厦门大学计算机科学系研究生课程

《分布式数据库技术》

专题二 分布式数据库系统的设计（2012年新版）

林子雨

厦门大学计算机科学系

E-mail: ziyulin@xmu.edu.cn

主页：http://www.cs.xmu.edu.cn/linziyu

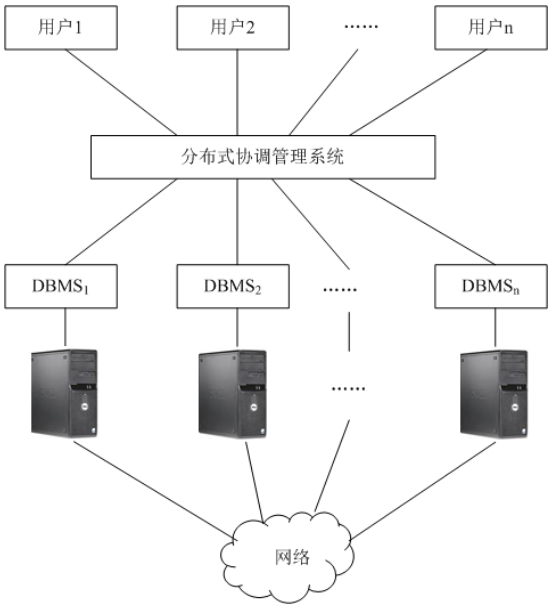
第3章　分布式数据库系统的设计

3.1 分布式数据库系统设计概述

3.1.1 分布式数据库系统的创建方法

(1) 采用组合法创建分布式数据库系统

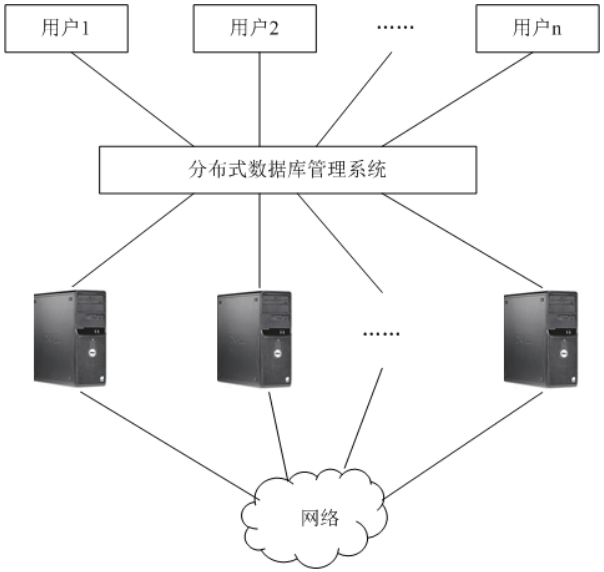
- 一种自底向上的创建方法，利用现有的计算机网络和独立存在于各个站点上的现代数据库系统，通过建立一个分布式协调管理系统，将它们集成为一个统一的分布式数据库系统。
- 用这种方法建立数据库系统，要对网络和各个站点数据库系统进行剖析，还需要解决数据的一致性、完整性和可靠性。如果各个站点的DBMS是不相同的，实施难度比较大。
- 由于利用现有的网络和数据库系统，仅仅需要建立一个分布式协调管理系统，因此，工作量小，周期短，花费人力、物力少，用户比较容易接受。
- 采用组合法的分布式数据库系统往往是异构或者同构异质的分布式数据库系统。法国IMAGE研究中心研制的POLYPHEME就采用组合法。



3.1.1 分布式数据库系统的创建方法

（2）采用重构法创建分布式数据库系统

- 重构法是根据系统的实现环境和用户需求，按照分布式数据库系统的设计思想和方法，采用统一的观点，从总体设计做起，包括各站点上的数据库系统，重新建立一个分布式数据库系统。
- 优点是，可以按照统一的思想来考虑分布式数据库系统中的各种问题，有效地解决分布式数据库系统的数据一致性、完整性和可靠性。但是，花费的人力、物力比较多，研制周期也比较长，系统建设代价大。
- 采用重构法，通常是同构异质，甚至是同构同质的分布式数据库系统。因为，同构型比异构型容易实现，如无特殊要求，一般采用重构法时，会选择同构型分布式数据库系统。
- 著名的POREL系统、SDD-1系统、SIRIUS-DELTA系统和MICROBE系统，都采用重构法。



3.1.2 分布式数据库设计的任务

(1) 数据库设计

对于一个给定的应用环境，构造最优的数据库模式，建立数据库及其应用系统，使之能够有效地存储数据，满足各种用户的应用需求。

(2) 分布式数据库设计的任务

- 分布式数据库设计包含以下任务：
  - 定义全局数据库的概念模式
  - 设计分片
  - 设计片段的分配
  - 设计物理数据库，将概念模式映射到存储区域，并确定适当的存储方法
- 在分布式数据库设计过程中，必须考虑分布式数据库应用的需求，包括：应用提交的场地、应用执行的频度、每个应用所存取数据的类型、次数及统计分布等信息
- 应该明确分布式数据库系统设计的基本策略：**从顶向下的设计处理或者从下向上的设计处理**

3.1.3 分布式数据库设计的目标

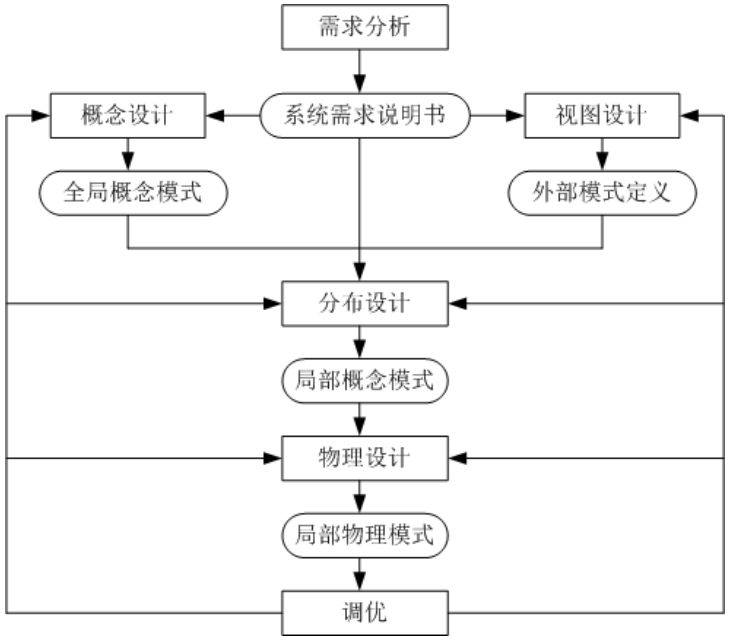
在设计分布式数据库时，不仅要实现集中式数据设计的目标，还要实现以下目标：

- 分布式数据库的本地性或近地性
  - 减少对网络的利用，尽可能减少站点之间的通信次数和通信量
  - 使得数据和应用实现最大程度的本地性
- 控制数据的适当冗余
  - 冗余提高了系统的本地性、并发度和可靠性
  - 为了维护数据一致性，减少同步更新的开销，需要控制数据副本的数量
- 工作负荷分布
  - 要充分利用每个站点计算机的能力和资源，提高应用执行的平衡度，提高性能
- 存储的能力和费用
  - 可以设置专门支持存储的站点，也可以设计不支持大容量存储的站点
  - 需要考虑站点可用存储空间的限制

3.1.4 分布式数据库设计的方法

- 分布式数据库系统的创建方法包括两种：组合法和重构法
- 相应地，分布式数据库的设计方法也包括两种
  - 自顶而下方法
  - 自底向上方法

3.1.4 分布式数据库设计的方法

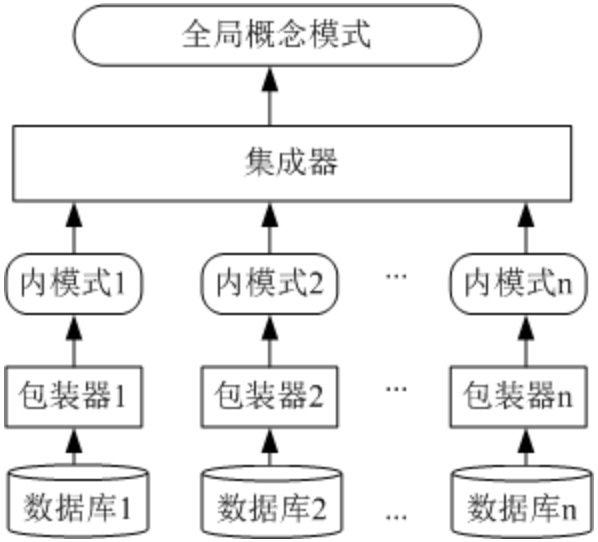


自顶向下设计过程示意图

自顶向下的设计过程，是从需求分析开始，进行概念设计、模式设计、分布设计（分片设计和分配设计）、物理设计以及性能调优等一系列设计过程。

- 第一步：系统需求分析。首先，根据用户的实际应用需求进行需求分析，形成系统需求说明书。该系统说明书是所要设计和实现的系统的预期目标。
- 第二步：根据系统需求说明书中的数据管理需求进行概念设计，得到全局概念模式，如E-R模型。同时根据系统说明中的应用需求，进行相应的外模式定义。
- 第三步：根据全局概念模式和外模式定义，结合实际应用需求和分布设计原则，进行分布设计，包括数据分片和分配设计，得到局部概念模式以及全局概念模式到局部概念模式的映射关系。
- 第四步，根据局部概念模式实现物理设计，包括片段存储、索引设计等。
- 第五步，进行系统调优。确定系统设计是否最好地满足系统需求，包括同用户沟通、系统性能模拟测试等，可能需要进行多次反馈，以使系统能最佳地满足用户的需求。

3.1.4 分布式数据库设计的方法



自底向上设计过程示意图

自底向上设计策略适合于已经存在多个数据库系统，并需要将它们集成为一个数据库的设计过程。自底向上的设计策略属于典型的数据库集成的研究范畴。有关异构数据库集成方法中，有基于集成器或包装器的数据库集成策略和基于联邦的数据库集成策略等。

基于集成器的多数据库集成系统的设计过程：

- 首先，各异构数据库系统经过相应的包装器转换为统一模式的内模式；
- 接着，集成器将各内模式集成为全局概念模式，集成过程中需要定义各内模式到全局模式的映射关系以及解决模式间的异构问题；
- 最后，全局概念模式即为自底向上策略设计得到的分布式数据库系统的全局概念模式。

3.2 数据分布的概念和方法

3.2.1 数据分布的概念和相关定义

3.2.1.1 数据分布的概念

数据分布的概念

逻辑上将全局概念模式（即全局关系模式），划分成若干逻辑片段（子关系）；再按一定的冗余度将片段分配到各个节点上，这时逻辑片段就成为具体的物理片段。

3.2.1.2 集中数据库的关系模式及形式化定义

为了讨论分布式数据库的模式定义，首先复习相关知识点。

（1）关系

定义3.1：域是一组具有相同数据类型的值的集合。用{　}表示。

定义3.2：给定一组域D_1，D_2，…，D_n，这些域中，可以有相同的。其中D_1，D_2，…，D_n的笛卡儿积为：

$D_1 \times D_2 \times \ldots \times D_n = \{ (d_1, d_2, \ldots, d_n) \mid d_i \in D_i, i=1, 2, \ldots, n \}$

其中每一个元素$(d_1, d_2, \ldots, d_n)$叫做一个n元组（或元组），元素中的每一个值d_i叫做一个分量

笛卡儿积的基数为$|D_1| \times |D_2| \times \ldots \times |D_n|$

定义3.3：$D_1 \times D_2 \times \ldots \times D_n$的子集叫做在$D_1$，$D_2$，…，$D_n$域上的关系，表示为：　$R(D_1, D_2, \ldots, D_n)$

其中，R表示关系名，n是关系的目（或称为度）。

3.2.1.2 集中数据库的关系模式及形式化定义

笛卡尔积可表示为一个二维表。表中每行对应一个元组，表中的每列对应一个域。

例1：给出三个域：

D1 =导师集合SUPERVISOR=张清玫，刘逸

D2 =专业集合SPECIALITY=计算机专业，信息专业

D3=研究生集合POSTGRADUATE=李勇，刘晨，王敏

则D1，D2，D3的笛卡尔积为：

D1×D2×D3={（张清玫，计算机专业,李勇），（张清玫，计算机专业，刘晨），
（张清玫，计算机专业，王敏），（张清玫，信息专业，李勇），
（张清玫，信息专业，刘晨），（张清玫，信息专业，王敏），
（刘逸，计算机专业，李勇），（刘逸，计算机专业，刘晨），
（刘逸，计算机专业，王敏），（刘逸，信息专业，李勇），
（刘逸，信息专业，刘晨），（刘逸，信息专业，王敏）}

其中（张清玫，计算机专业，李勇）、（张清玫，计算机专业，刘晨）等都是元组。张清玫、计算机专业、李勇、刘晨等都是分量。

3.2.1.2 集中数据库的关系模式及形式化定义

该笛卡尔积的基数为2×2×3=12，也就是说，D1×D2×D3一共有2×2×3=12个元组。这12个元组可列成一张二维表，如下：

D_1，D_2，D_3的笛卡尔积

SUPERVISOR	SPECIALITY	POSTGRADUATE
张清玫	计算机专业	李勇
张清玫	计算机专业	刘晨
张清玫	计算机专业	王敏
张清玫	信息专业	李勇
张清玫	信息专业	刘晨
张清玫	信息专业	王敏
刘逸	计算机专业	李勇
刘逸	计算机专业	刘晨
刘逸	计算机专业	王敏
刘逸	信息专业	李勇
刘逸	信息专业	刘晨
刘逸	信息专业	王敏

3.2.1.2 集中数据库的关系模式及形式化定义

定义3.3说明：

- 若关系中的某一属性组的值能唯一地标识一个元组，则称该属性组为侯选码。
- 若一个关系有多个侯选码，则选定其中一个为主码（Primary Key）。主码的诸属性称为主属性（Prime attribute）。
- 不包含在任何侯选码中的属性称为非码属性（Non-key attribute）。
- 在最简单的情况下，侯选码只包含一个属性；在最极端的情况下，关系模式的所有属性组是这个关系模式的侯选码；上述称为全码（All-key）。
- 关系是一个二维表，表的每行对应一个元组，表的每一列对应一个域。

3.2.1.2 集中数据库的关系模式及形式化定义

(2) 关系模式

定义3.4：关系的描述称为关系模式。它可以形式化地表示为：

R (U，D，dom，F)

其中：R为关系名，

U为组成该关系的属性名集合，

D为属性组U中属性所来自的域，

dom为属性向域的映象的集合，

F为属性间数据的依赖关系集合。

(3) 关系数据库

定义3.5：在一个给定的应用领域中，所有实体及实体之间联系的关系的集合构成一个关系数据库。

3.2.1.3 分布式数据库的模式定义

3.2.1.3.1 全局关系模式及关系

全局关系模式是一个多元组，表示为：R（U，D，dom，I，F，Q，S），其中：

R是关系名；　　U是组成R的有限属性集；

D是U中属性的值域；　　dom是属性列到域的所有映射的集合；

I 是一组完整性约束条件；　Q是关系所满足的限定条件（谓词）；

F是属性间的一组数据依赖；　S是关系的分布结构。

一个**关系r**是相应于全局关系模式R（U，D，dom，I，F，Q，S）按分布结构S组织起来的从属性集U到值域D上所有满足Q的映射的集合。其中，每个元素称为元组；每个关系有主键$K \subseteq U$。

关系和关系模式有以下关系：

✓关系模式描述关系的结构及语义约束，对于全局关系模式，同时还具有按一定谓词条件Q划分成子关系（局部关系）模式和子关系物理存储模式的约束

✓关系是关系模式在某一时刻的“当前值”

✓全局关系被划分以后，按分布结构组成子关系，以呈现现实世界某一时刻的状态。所有关系的当前值就是关系数据库

✓为了把问题集中于数据分布，可以把全局关系模式简化成R（U，Q，S）

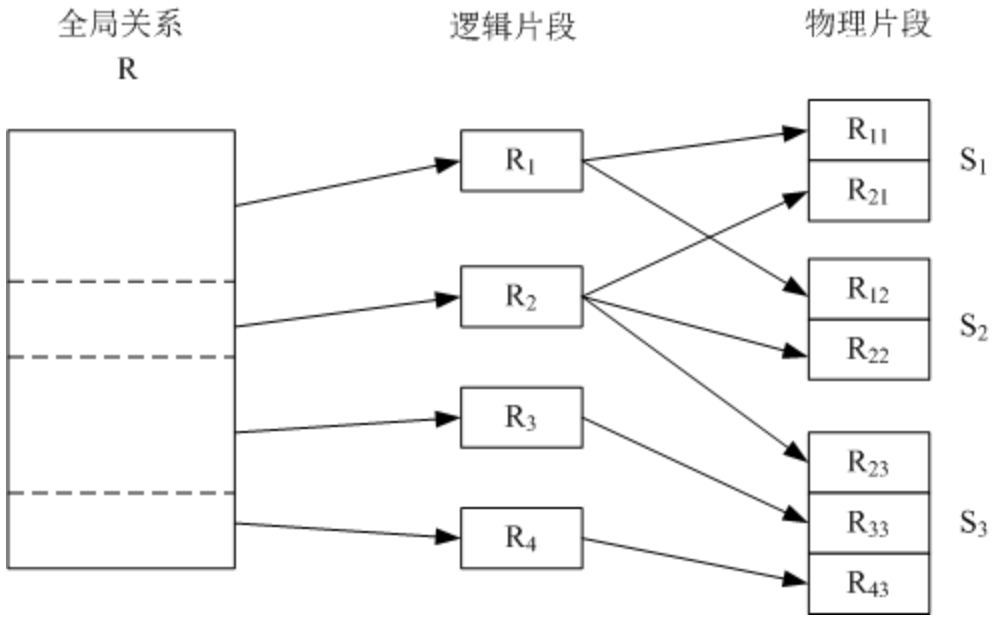
3.2.1.3.2 DDB中的三种关系

- **全局关系**：当一个关系R（U，Q，S）称为全局关系（GR），是指在分布式数据库系统中对用户是可见的，实际上是由若干子关系的逻辑片段和物理片段按分布结构S组成，具有Q=TRUE，S≠Ø的特点，且GR是虚拟的。
- **逻辑片段**：当一个关系R（U，Q，S）称为逻辑片段，是指这个关系在DDB中是实际存在的关系，不需要由其它关系组成，因而它是基本关系，即是构成DDB的实体。它是全局关系在某个场地上的子关系的逻辑成分，所以逻辑片段可简称为GR的逻辑关系，而全局关系与逻辑关系间有一定的映射，用分片模式定义，它们之间的映射性质为 1：n 。
- **物理片段**：当一个关系R（U，Q，S）称为物理片段，是指这个关系是在某一场地上的逻辑片段，即是在某场地上的基本关系，其特点是S=Ø。物理片段由分配模式定义。逻辑片段映射为某个场地上的物理片段，亦可称为某个场地的副本，或直接称物理关系，其映射有 1：1 和 1：n性质（因而一个逻辑关系可以对应一个物理关系，也可对应多个物理关系）。
- **结论：**

一个全局关系由分片操作（分片模式定义）分解成多个逻辑关系；一个逻辑关系在几个场地上放置副本（分配模式定义）就产生几个物理关系。这些分片信息和分配信息构成了全局关系的分布结构S。

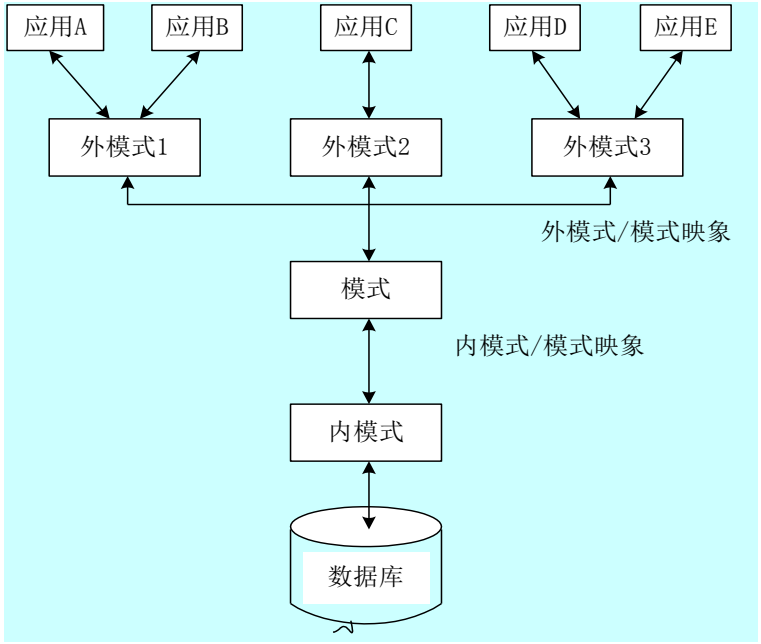
3.2.1.3.2 DDB中的三种关系

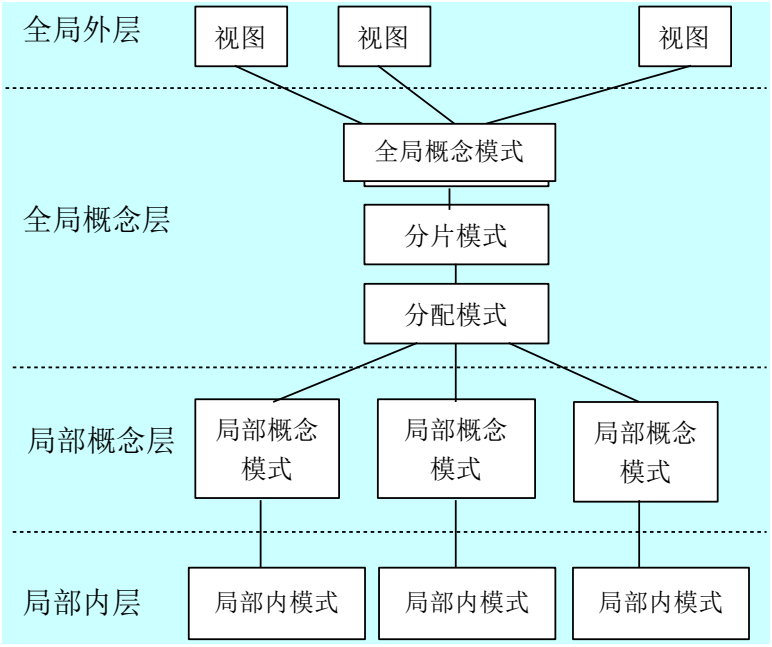
例：下图是全局关系R的分片和分配情况示意图，图中全局关系R被划分为四个逻辑片段：R_1,R_2,R_3,R_4，并以冗余的方式将这些片段分配到网络上的三个站点S_1,S_2,S_3中。R_1在站点S_1和S_2上重复存储，得到R_1的两个物理片段R_{11}和R_{12}，R_2在站点S_1,S_2,S_3上重复存储，得到R_2的三个物理片段R_{21},R_{22},R_{23},而R_3和R_4只存储在站点S_3上，得到物理片段分别为R_{33}和R_{43}。



3.2.1.3.2 DDB中的三种关系

集中式三层模式结构图





分布式四层模式结构图

问题：DDB三种关系分别属于哪一层？

3.2.1.3.3 DDB中的三种数据库

全局（虚拟）数据库：由所有全局关系组成的数据库称为全局（或虚拟）数据库GDB。
逻辑数据库：由所有逻辑片段（基本关系）组成的数据库称为逻辑数据库LgDB。
物理数据库：由所有物理关系组成的数据库称为物理数据库PDB。

三种数据库之间的关系可表示如下：

GDB —分片模式→ LgDB —分配模式→ PDB

当用户查询或更新操作分布式数据库时，只是对虚拟的全局数据库操作，它并不实际存在，而是由若干“片段”组成的（由若干片段的并行操作和自然联接操作实现的），这些片段映射为一个“物理关系”存在于物理数据库中。

三种数据库是通过分片模式定义和分配模式定义联系起来的。

3.2.1.3.4 分片模式（FS）定义

分片模式**FS**定义为一组操作，它将**GDB**中每个**GR**分解成**LgDB**中的片段，即：

FS（GDB）= LgDB

这种操作还应具有可逆性，这是分布式数据库系统的基本要求：

FS^{-1}（LgDB）＝GDB

3.2.1.3.5 分配模式（AS）定义

分配模式定义为一组操作，它将LgDB中每个逻辑关系映射到PDB中的一组物理关系上，即：

$$AS（LgDB）= PDB$$

分配模式的逆操作含义是，从PDB中选择出适当的物理关系作为逻辑关系，从而构成LgDB，即有：

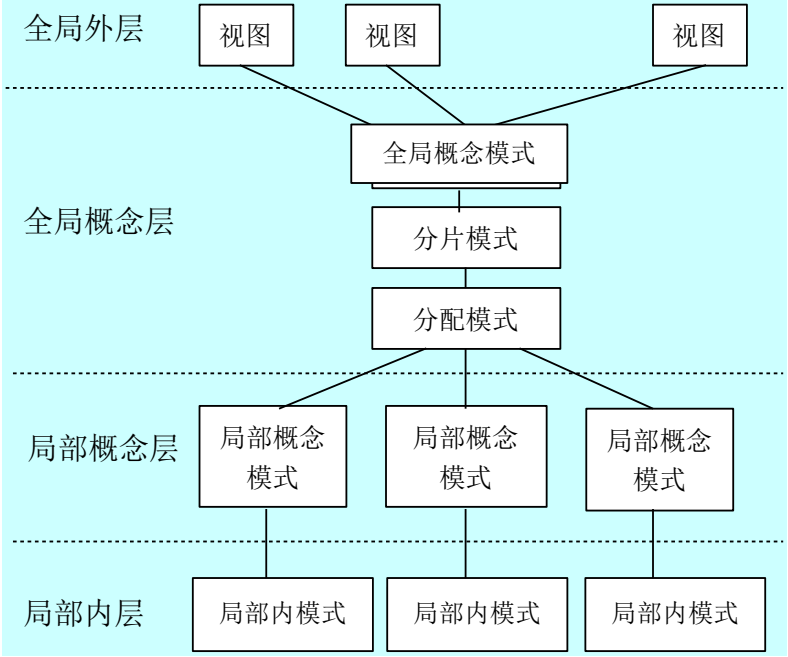
$$AS^{-1}（PDB）=LgDB。$$

3.2.1.3.6 关系的分布结构S

关系R（U，Q，S）的分布结构S是有关R被划分和分配的信息的集合

- ✓如果R是物理关系，那么它无需划分和分配，因此S=Ø
- ✓如果R是逻辑关系，那么它还需分配场地，因此S记载R的分配信息
- ✓如果R是全局关系，那么S记载了R如何分解成逻辑关系和物理关系

3.2.1.4. 分布式数据库的分布透明性



3.2.1.4.1 分片透明性

- **分片透明性**是分布透明性中的最高层次。在分布式数据库模式结构图中，分片透明位于全局概念模式与分片模式之间。
- 当分布式数据库具有分片透明性时，用户编写的应用程序只对全局关系进行操作，不必考虑数据的逻辑分片，当分片模式改变时，只要改变全局概念模式到分片模式之间的映射，不会影响用户程序，从而实现了数据分片透明性。
- 例：考虑全局关系SUPPLIER(SNO,SNAME,SCITY)，被划分成两个逻辑片段SUPPLIER1和SUPPLIER2，片段SUPPLIER1存放在站点S_1上，片段SUPPLIER2有一个副本，分别存在站点S_2和S_3上面。现在编写一个名为SUPQUERY的简单查询程序，它将从终端接收一个供应商号，查询该供应商号相应的供应商名，并显示在屏幕上。

具备分片透明性时的程序书写，用户只需要知道全局关系即可编写程序

```
Read(terminal, $SNO);
Select SNAME into $SNAME
From SUPPLIER
Where SNO=$SNO;
Write(terminal, $SNAME);
```

3.2.1.4.2 位置透明性

- **位置透明性**，也称**分配透明性**，是分布透明性的中间层。位于分布式数据库模式结构图的分片模式和分配模式之间。当片段及其副本的存储站点发生改变时，只要改变从分片模式到分配模式之间的映射，不会影响用户程序，从而实现数据片段的位置透明性。
- 位置透明性包含两种情形：
  - **复制透明性**（或数据冗余透明性）：不需要用户了解各片段被复制的情况，即是否被复制，复制了几个副本；
  - **位置透明性**：不需要用户了解各个副本的位置分布情况。

不具备分片透明性，具备位置透明性时的程序书写，用户需要知道全局关系的分片情况，但是不需要只为分片的存储位置

```
Read(terminal, $SNO);
Select SNAME into $SNAME
From SUPPLIER1
Where SNO=$SNO;
If not # FOUND then
Select SNAME into $SNAME
From SUPPLIER2
Where SNO=$SNO;
Write(terminal, $SNAME);
```

3.2.1.4.3 局部数据模型透明性

- **局部数据模型透明性**，也称局部映射透明性，即与数据库各站点上的数据库的数据模型无关，是分布透明的最低层。在分布式数据库模式结构图中位于分配模式和局部概念模式之间。
- 全局数据模型和每个站点上的局部数据模型的转换，由分配模式和局部概念模式之间的映射实现。当某个站点上的数据模型发生改变时，只需要修改分配模式到该站点的局部概念模式之间的映射，应用程序不会受到影响，从而实现了局部数据模型透明性。

具备局部数据模型透明性时的程序书写，用户不但要了解全局数据的逻辑分片情况，还要了解各个逻辑片段的副本复制和存储位置情况，但是，不需要了解各个站点上的数据库的数据模型及其对象的表示性质。

```
Read(terminal, $SNO);
Select SNAME into $SNAME
From SUPPLIER1 At S1
Where SNO=$SNO;
If not # FOUND then
Select SNAME into $SNAME
From SUPPLIER2 At S3
Where SNO=$SNO;
Write(terminal, $SNAME);
```

3.2.1.4.4 无透明性

- 假设站点S1上的本地DBMS为IMS，站点S3上面的DBMS为codasyl。此时，程序员必须编写实现所需功能的IMS程序和codasyl程序，并把这些程序安装在相应的站点上。

```
Read(terminal, $SNO);
execute $SUPIMS($SNO,$FOUND,$SNAME)  at S1;
If not $FOUND then
execute $SUPCODASYL($SNO,$FOUND,$SNAME)  at S3;
Write(terminal, $SNAME);
```

3.2.1.4.5 几种分布透明性之间的关系

- **高级分布透明性**
  - 如果一个分布式数据库提供了分片透明性，当然也会提供分配透明性和局部数据模型透明性，是分布透明性的最高级别。
  - 此时，对用户和应用程序而言，它们所面对的分布式数据库系统，如同集中式数据库一样，不必考虑数据的分片细节，不必考虑各片段的副本情况，不必考虑各个副本的分配细节，也不必考虑各站点上数据库是什么数据模型。
- **中级分布透明性**
  - 如果一个分布式数据库系统提供分配透明性，而没有提供分片透明性，当然也会提供局部数据模型透明性，所以也称为中级分布透明性。
  - 此时，对于用户和应用程序而言，必须知道分布式数据库全局数据的逻辑分片情况，在程序中必须给出所需要访问的逻辑片段名。但是，不必关心逻辑片段是否被复制，以及如何分配到各站点，也不必考虑各站点的数据模型。
- **低级分布透明性**
  - 如果一个分布式数据库系统只提供局部数据模型透明性，不提供分片透明性，也不提供分配透明性，称为低级分布透明性。
  - 此时，对于用户和应用程序而言，就必须知道分布式数据库的全局数据的逻辑分片情况，还必须知道各片段是否有副本，有多少副本，各个副本被分配到哪些站点，但是，不必考虑局部站点上的数据模型。
- **无分布透明性**
  - 需要用户自己处理异构数据模型的转换。

3.2.1.4.6 分布式数据库更新应用与分布透明性

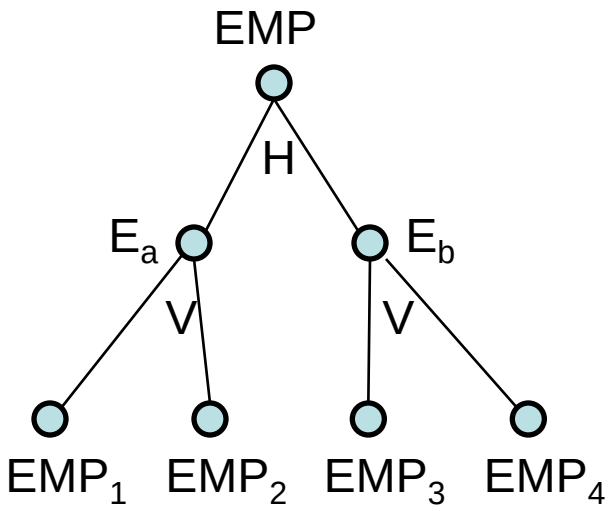
- 查询操作只需要对片段任意副本进行，但是，对于更新操作，必须针对所有副本进行。如果分布式数据库不提供位置透明性，就需要由应用程序来完成针对所有副本的更新。
- 在更新应用中，还有一个比更新某个数据项的所有副本更为复杂的问题，就是当被更新的属性，恰好是作为分片模式定义中所使用的属性时，可能引起元组从一个片段移动到另一个片段，会产生相当复杂的影响。
- 例：考虑全局关系EMP(ENO,ENAME,DNO,SAL,TAX,MGRNO)经过混合分片，先按照DNO的值小于10来进行水平分片，分成两个片段E_a和E_b，后来又经过垂直分片，划分为如下四个逻辑片段：

$EMP_1=E_a(ENO,ENAME,SAL,TAX)$

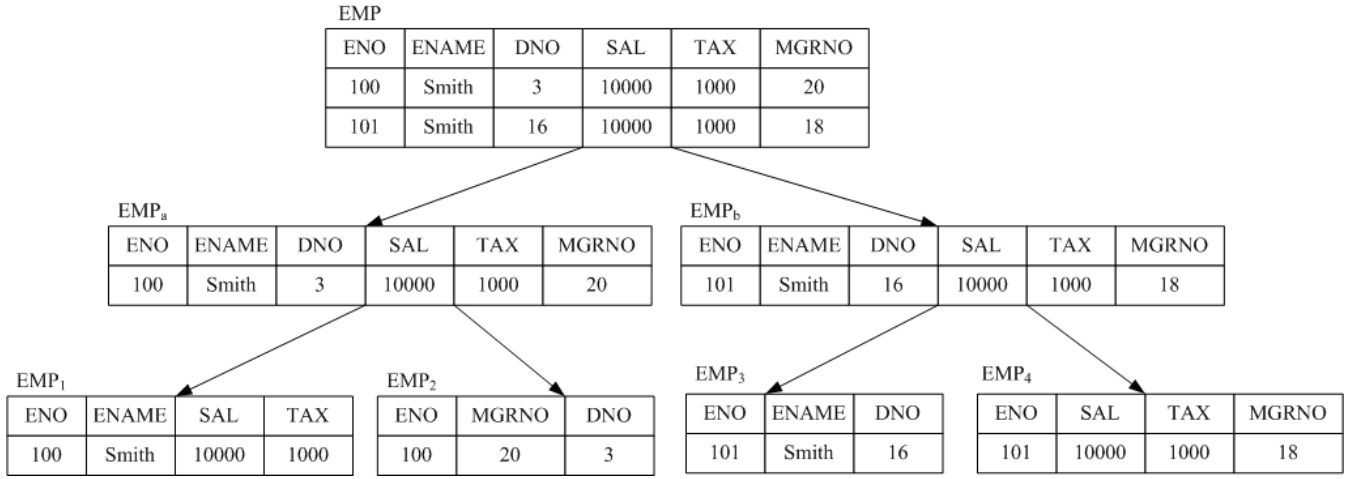
$EMP_2=E_a(ENO,MGRNO,DNO)$

$EMP_3=E_b(ENO,ENAME,DNO)$

$EMP_4=E_b(ENO,SAL,TAX,MGRNO)$



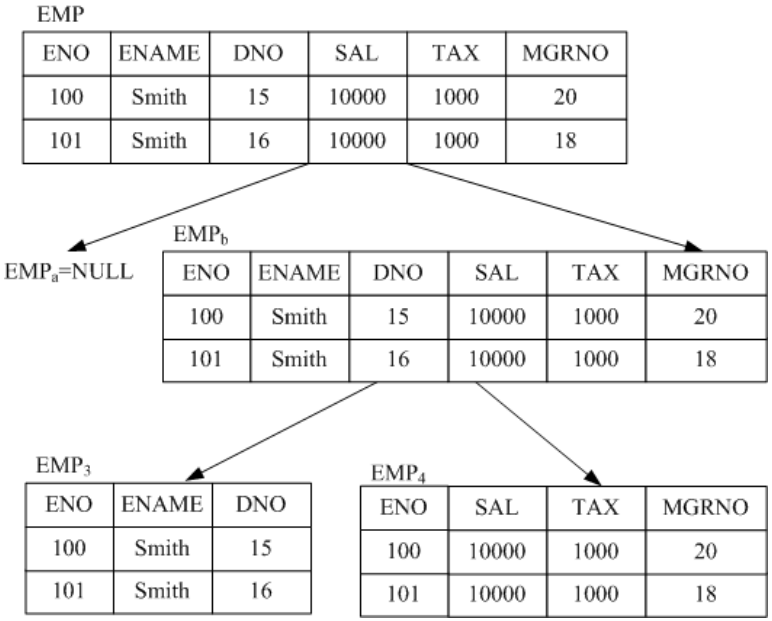
3.2.1.4.6 分布式数据库更新应用与分布透明性



更新前

更新后

更新操作：把ENO=100的元组的DNO=3属性更改为DNO=15，这个更新会导致EMP1和EMP2片段的元组ENO=100重新组合后，移动到EMP3和EMP4中。



3.2.1.4.6 分布式数据库更新应用与分布透明性

（1）分片透明性时的程序

应用程序就如同集中式数据库一样来执行更新操作，编程人员不必知道被更新的属性是否是分片模式定义中所使用的属性。

```
update emp set dno=15
where eno=100
```

3.2.1.4.6 分布式数据库更新应用与分布透明性

（2）位置透明性时的程序

编程人员必须知道分片情形，并给出明确的处理。

```
select ename,sal,tax  into  $ename, $sal, $tax from emp1
where eno=100;
select mgrno into $mgrno from emp2
where eno=100;
insert into emp3 (eno, ename, dno)
values (100, $ename, 15);
insert into emp4 (eno, sal, tax, mgrno)
values (100, $sal, $tax, $mgrno);
delete emp1 where eno=100;
delete emp2 where eno=100;
```

3.2.1.4.6 分布式数据库更新应用与分布透明性

（3）局部数据模型透明性时的程序

- 编程人员必须明确给出片段的位置，还要考虑片段副本。
- 假设EMP四个片段的副本分布如下：
  - EMP_1：站点S_1和站点S_5；
  - EMP_2：站点S_2和站点S_6；
  - EMP_3：站点S_3和站点S_7；
  - EMP_4：站点S_4和站点S_8；

```
select ename,sal,tax  into  $ename, $sal,
$tax from emp1 at S1
where eno=100;
select mgrno into $mgrno from emp2 at S2
where eno=100;
insert into emp3 (eno, ename, dno) at S3
values (100, $ename, 15);
insert into emp3 (eno, ename, dno) at S7
values (100, $ename, 15);
insert into emp4 (eno, sal, tax, mgrno) at S4
values (100, $sal, $tax, $mgrno);
insert into emp4 (eno, sal, tax, mgrno) at S8
values (100, $sal, $tax, $mgrno);
delete emp1 at S1 where eno=100;
delete emp1 at S5 where eno=100;
delete emp2 at S2 where eno=100;
delete emp2 at S6 where eno=100;
```

3.2.2 分片作用、操作原则及分片方法

3.2.2.1 分片的作用

对数据进行分片存储，便于分布地处理数据，对提高分布式数据库系统的性能至关重要。主要作用如下：

（1）**减少网络传输量**。网络上的数据传输量是影响分布式数据库系统中数据处理效率的主要代价之一。为了减少网络上的数据传输代价，分布式数据库中的数据允许复制存储，目的就是可以就近访问所需要的数据副本，将少网络上的数据传输量。因此，在数据分配设计时，设计人员需要根据应用需求，将频繁访问的数据分片存储在尽可能近的场地上。

（2）**增大事务处理的局部性**。数据分片按需分配在各自的局部场地上，可并行执行局部事务，就近访问局部数据，减少数据访问的时间，增强局部事务的处理效率。

（3）**提高数据的可用性和查询效率**。就近访问数据分片或副本，可提高访问效率。同时，当某一个场地出现故障时，若存在副本，非故障场地上的数据副本均可用，保证了数据的可用性和完整性以及系统的可靠性。

（4）**均衡负载**。有效利用局部数据处理资源，就近访问局部数据，可以避免访问集中式数据库所造成的数据访问瓶颈，有效提高整个系统效率。

3.2.2.2 分片操作原则

（1）数据划分的基本思路：首先按DDB外部特征划分数据，然后根据DDB的内部特征，提出应遵守的基本原则以检验数据划分的正确性。

所谓“外部特征”是指构成DDB的属性群集特性，包括属性值集和数据项集等。

例1：某公司数据库中有部门关系Dept，便于管理可按部门性质或部门所在地将部门关系Dept划分成若干片段（如部门号DNO，或部门所在地区AREA的属性值集划分）。这是一种按属性值集的划分。

例2：而上述数据库中有职员关系Emp，通常可按业务管理性质将Emp的属性组合值集划分（如财务部门对税收TAX、工资SAL等属性项有兴趣，行政部门对Emp承担工作业务的属性项有兴趣等）。这是按数据项集划分。

内部特征是指DDB的组成性质。

（2）基本原则：当对DDB划分后，仍应保持DDB原有的特质，所以划分后的各逻辑关系之间应遵守下列原则：

完整性原则、重构性原则、不相交原则

3.2.2.3 分片操作

3.2.2.3.1 水平分片

水平分片是将关系按行横向以某些条件划分成元组的子集，（即：满足条件的记录的集合）每个子集含有一定的逻辑意义，称逻辑片段。

定义1：组合关系R（U，Q，S）上的水平分片（H）是一操作，它将R按照一组给定的谓词 P_1，P_2，…，P_n，划分成一组关系模式：

R_1（U_1，Q_1，S_1），…，R_n（U_n，Q_n，S_n）

满足：1）$U_1=U_2=\ldots=U_n=U$；

2）$K_1=K_2=\ldots=K_n=K$；

3）$Q_i=Q \wedge P_i$；

4）$S_i \neq \emptyset$

其中：i，j∈{l，…，n}，$P_i \wedge P_j=\emptyset$，$\vee P_i$=True，记为：

R（H）<P> = { R_1，…，R_n } 式中P={P_1，…，P_n}。

由定义可得出：水平分片实际上是关系的选择操作。即属性="值"的具体条件的子关系R_i，因此片段可用$\sigma_q(R)$表示。

3.2.2.3.1 水平分片

例1：供应商全局关系Supplier（SNO，SNAME，SCITY）按供应商所在地SCITY 属性值划分。假设只有两个城市London和Paris，则按上述定义，其水平分片为：

$$Supplier1 = \sigma_{scity='London'}(Supplier)$$

$$Supplier2 = \sigma_{scity='Paris'}(Supplier)$$

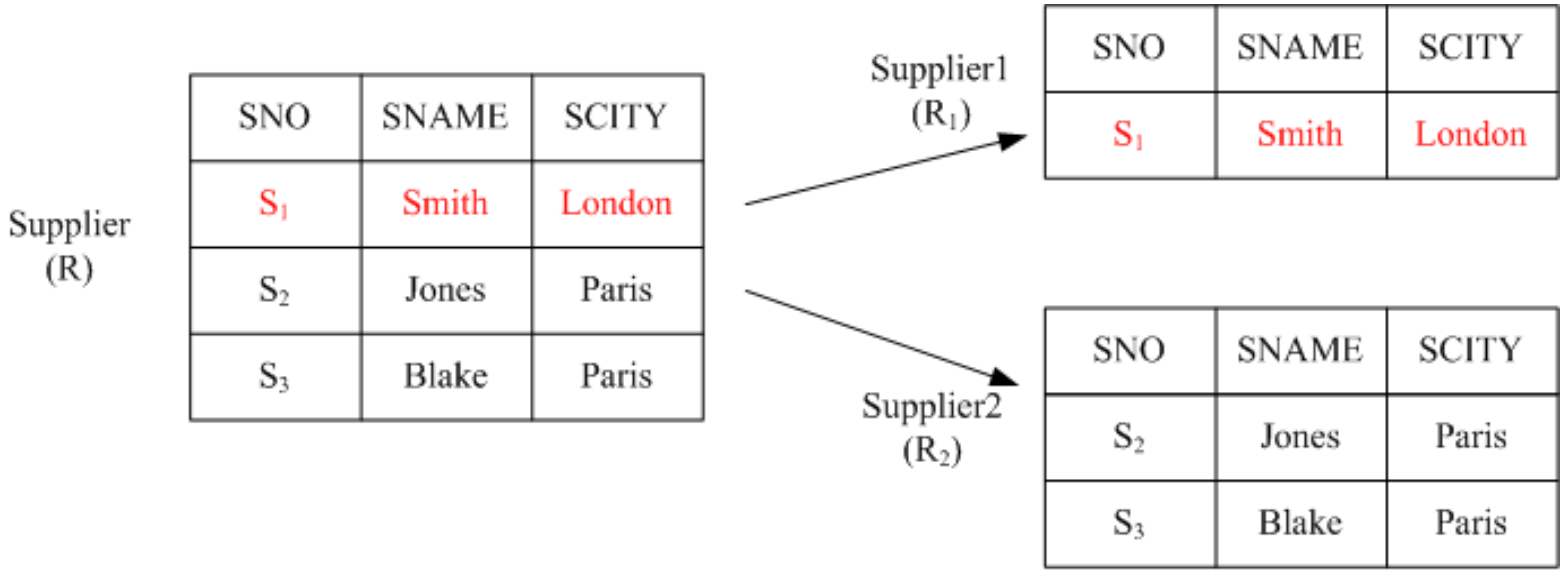


图 水平分片范例

3.2.2.3.2 垂直分片

垂直分片是将关系按列纵向以属性组划分成若干片段。在垂直分片时，为了保证片段的重构性，应将“键属性”属于各个片段中（放松的不相交性）。

定义2：组合关系R（U，Q，S）上垂直分片（V）是一操作，它将R按照一组给定的属性A_1，…，A_n划分成一组关系模式：

R_1（U_1，Q_1，S_1），…，R_n（U_n，Q_n，S_n）

满足：1）$K_1=K_2=\ldots=K_n=K$；

2）$U_i=A_i$；　　3）$Q_1=Q_2=\ldots Q_n=Q$；

4）$\pi_{k1}(R_1)=\pi_{k2}(R_2)=\ldots=\pi_{kn}(R_n)=\pi_k(R)$；

5）$S_i\neq\emptyset$

其中：i，j∈{1，…，n}，$A_i \subseteq U$，$A_i \cap A_j = K$，$\cup A_i = U$

记为：R（V）<A>={R_1，…，R_n} 式中A={A_1，…，A_n}

由定义可得出，关系的垂直分片实际上是对指定属性集上的投影操作。所以，R关系的垂直分片片段是R的部分属性组合子关系R_i，可用$\pi_{A_i}(R)$表示，其中$K \subseteq A_i$。

3.2.2.3.2 垂直分片

例 2：职员关系 Emp（ENO,ENAME,SAL,TAX,MGRSSN,DNO）划分成两个子关系：一个包括财务信息，对SAL,TAX属性感兴趣；一个包括工作信息，对ENO,ENAME,MGRSSN,DNO属性感兴趣。为了保证划分后重构，可将ENO作为公共属性分别包括在二个片段中。这样Emp可划分：

$$Emp1=\pi_{ENO,SAL,TAX}（Emp）$$

$$Emp2=\pi_{ENO,ENAME,MGRSSN,DNO}（Emp）$$

Emp(R)

ENO	ENAME	SAL	TAX	MGRSSN	DNO
E_1	Mike	100	10	480036	D1
E_2	John	20	20	33344	D2
E_3	Smith	300	30	987654	D3
E_4	Jack	200	20	333444	D2

$Emp2(R_2)$

ENO	ENAME	MGRSSN	DNO
E_1	Mike	480036	D1
E_2	John	33344	D2
E_3	Smith	987654	D3
E_4	Jack	333444	D2

$Emp1(R_1)$

ENO	SAL	TAX
E_1	100	10
E_2	20	20
E_3	300	30
E_4	200	20

图 垂直分片范例

3.2.2.3.2 垂直分片

垂直分片设计方法

垂直分片设计的核心是根据用户的应用需求正确地划分属性组。这里采用属性紧密度(affinity)来度量属性间的关系。

令Q={q_1,q_2,…,q_m}是用户的查询应用，关系R{A_1,A_2,…,A_n}包含属性A_1、A_2、…、A_n属性，则*aff*(A_i,A_j)表示属性A_i与A_j的紧密度，计算公式如下：

$$aff(A_i, A_j) = \sum_{l=1}^{s} \sum_{k=1}^{m} (ref_l(q_k) acc(q_k))$$

其中，*l*为场地，*s*为场地个数，*m*为查询个数，$ref_l(q_k)$为查询q_k在场地S_l上同时访问属性A_i与A_j的次数， $acc_l(q_k)$为查询q_k在场地S_l上的访问频率统计值。

方法是：根据两两属性间或局部范围内属性间的紧密度，通过线性地排序属性间的紧密度来实现分组，即*aff*(A_i,A_j)具有较大值的属性划分为一组，*aff*(A_i,A_j)具有较小值的属性划分为一组。

3.2.2.3.3 混合分片

混合分片是水平分片和垂直分片的内部混合。

定义3：组合关系R（U，Q，S）上的混合分片（M）是一操作，它将关系按照一组属性$A_1,\ldots,A_n$和一组谓词$P_1,\ldots,P_n$划分成：

$R_1(U_1,Q_1,S_1)$，…，$R_n(U_n,Q_n,S_n,)$

满足：1）$U_i = A_i$

2）$K_1=K_2=\ldots=K_n=K$;

3）$Q_i=Q \wedge P_i$;

4）$S_i \neq \varnothing$。

其中：i，j∈{l，…，n}，$A_i \subseteq U$，$A_i \cap A_j = K$，$\cup A_i = U$，$P_i \wedge P_j = \varnothing$，$\vee P_j = True$，记为：

$R(M)<AP> = \{ R_1, \ldots, R_n \}$ 式中$AP = \{ <A_i,P_i> \mid i=1,\ldots,n \}$。

上述定义说明混合分片是水平分片和垂直分片的混合操作，即对关系的选择和投影。当要重构混合分片的各片段，可按相应次序做合并（UNION）操作和联接（JOIN）操作。

3.2.2.3.3 混合分片

例3：将例2的Emp划分成Emp1,Emp2后，对Emp2可按部门号DNO=1、DNO=3分在一个片段，而DNO=2在另一片段，则Emp可划分为：

$Emp1=\pi_{ENO，SAL，TAX}（Emp）$

$Emp2=\sigma_{DNO=“D1”or\ DNO=“D3”}(\pi_{ENO,ENAME,MGRSSN,DNO}\ Emp)$

$Emp3=\sigma_{DNO=“D2”}(\pi_{ENO,ENAME,MGRSSN,DNO}\ Emp)$

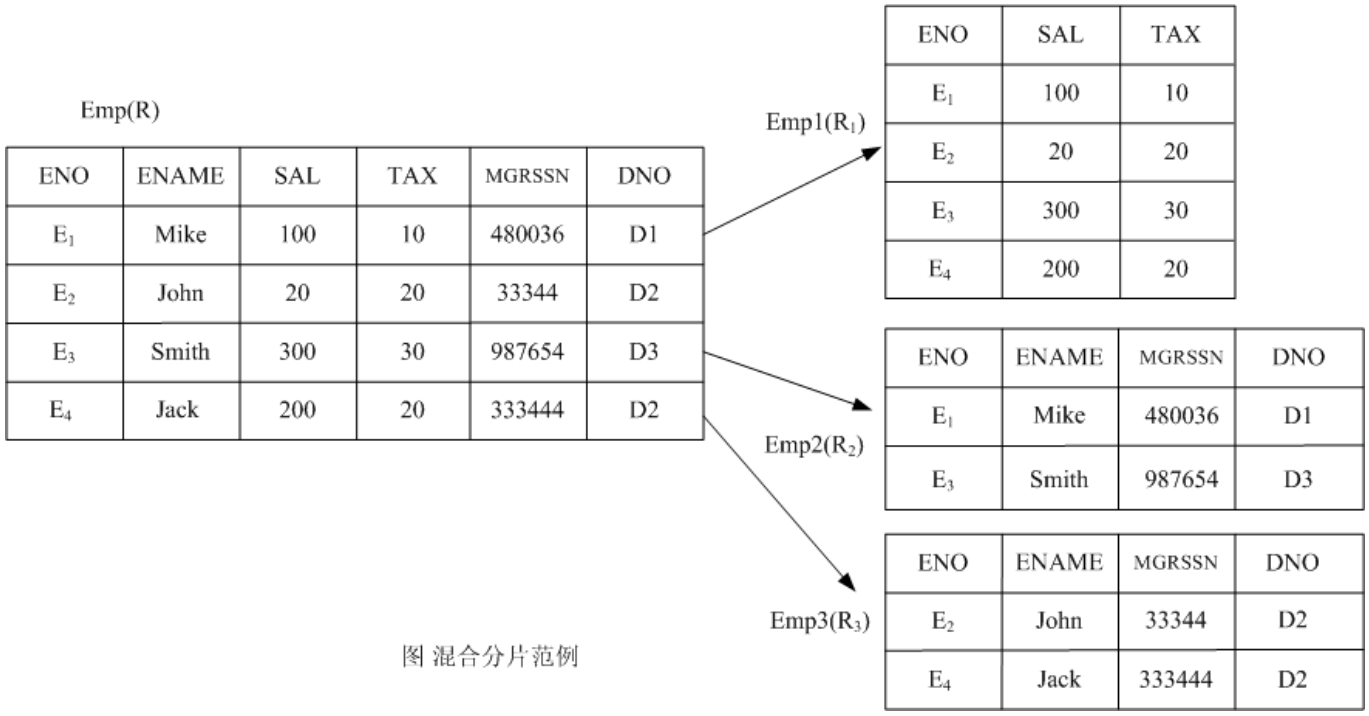


图 混合分片范例

3.2.2.3.4 诱导分片

一个关系的分片不是基于关系本身的属性，而是根据另一个与其有关联性质的关系的属性来划分。这种划分是只基于水平分片的诱导

定义4：组合关系R（U,Q,S），按另一个组合关系T（T是已经水平分片成T_1（U_1, Q_1, S_1）,…，T_n（U_n,Q_n, S_n））在公共属性A上的诱导分片（DH）是一操作，它将R划分为：R_1（U_1', Q_1', S_1'），…，R_n（U_n', Q_n', S_n'）

满足：

1）$U_1' = U_2' = \cdots = U_n' = U$;

2）$K_1' = K_2' = \cdots = K_n' = K$

3）$\pi_A(R_i) \subseteq \pi_A(T_i)$

4）$Q_i' = Q_i$;

5）$S'_i \neq \emptyset$

记为：R（DH）<T>={R_1，…，R_n} 式中：T={T_1，…，T_n}

- 诱导分片是一种相关分片操作，它是一种半联接操作
- 诱导分片实例

3.2.2.3.4 诱导分片

关于半联接操作

- 关系代数操作中的联接（JOIN）操作包括θ-联接和自然联接
- 半联接操作是关系代数操作中联接（JOIN）操作的一种缩减：关系R和S的半联接记为R∝S，其结果关系是R和S自然联接（Natural JOIN）后在R的属性上的投影，可用下述表达式表示：

$$R \propto S = \pi_R (R \infty S)$$ （实例）

- 计算R∝S的另一种等价的方法是：将S中与R有相同属性名的属性集投影出来，然后与R完成自然联接
- 半联接操作是一种不对称操作，即R∝S≠S∝R
- 在分布式数据库的查询优化中，常常用半联接操作实现联接操作的操作数（关系）的缩减（归约）

3.2.2.3.4 诱导分片

自然联接的结果是在R和S中的在它们的公共属性名字上相等的所有元组的组合。例如下面是表格“雇员”和“部门”和它们的自然联接：

雇员

Name	EmpId	DeptName
Harry	3415	财务
Sally	2241	销售
George	3401	财务
Harriet	2202	销售

部门

DeptName	Manager
财务	George
销售	Harriet
生产	Charles

雇员 ⋈ 部门

Name	EmpId	DeptName	Manager
Harry	3415	财务	George
Sally	2241	销售	Harriet
George	3401	财务	George
Harriet	2202	销售	Harriet

图 自然联接实例

3.2.2.3.4 诱导分片

θ-联接（等值联接是它的特例）

如果我们要组合来自两个关系的元组，而组合条件不是简单的共享属性上的相等，则有一种更一般形式的联接算子才方便，这就是 θ-联接(或 theta-联接)。θ 是在集合{<, ⩽, =, >, ⩾}中的二元关系。θ的结果由在 R 和 S 中满足关系 θ 的元素的所有组合构成。只有 S 和 R 的表头是不相交的，即不包含公共属性的情况下，θ-联接的结果才是有定义的。

实例：考虑分别列出车模和船模的价格的表“车”和“船”。假设一个顾客要购买一个车模和一个船模，但不想为船花费比车更多的钱。在关系上的θ-联接 *CarPrice* ≥ *BoatPrice* 生成所有可能选项的一个表。

车

CarModel	CarPrice
CarA	20'000
CarB	30'000
CarC	50'000

船

BoatModel	BoatPrice
Boat1	10'000
Boat2	40'000
Boat3	60'000

车 ⋈ 船 $CarPrice \geq BoatPrice$

CarModel	CarPrice	BoatModel	BoatPrice
CarA	20'000	Boat1	10'000
CarB	30'000	Boat1	10'000
CarC	50'000	Boat1	10'000
CarC	50'000	Boat2	40'000

图 θ-联接实例

3.2.2.3.4 诱导分片

半联接的结果只是在 S 中有在公共属性名字上相等的元组的所有 R 中的元组。

实例：“雇员”和“部门”和它们的半联接的表：

$R \propto S = \pi_R(R \infty S)$

R

雇员

Name	EmpId	DeptName
Harry	3415	财务
Sally	2241	销售
George	3401	财务
Harriet	2202	生产

S

部门

DeptName	Manager
销售	Harriet
生产	Charles

图 半联接实例

雇员 ⋉ 部门

Name	EmpId	DeptName	Manager
Sally	2241	销售	Harriet
Harriet	2202	生产	Charies

半联接结果

自然联接结果

3.2.2.3.4 诱导分片

例4：设供应关系Supply(SNO,PNO,QTY)，它的划分要求按供应商所在地SCITY属性值划分。

分析：虽然SCITY不是Supply关系的属性，但Supply是Supplier与另一个零件关系Part的关联（这种关联描述了供应商供应零件的细节）。Supply与Supplier和Part分别通过SNO和PNO建立联系，就Supply与Supplier而言，SNO是它们的公共的属性，充当Supply的外键（foreign key）。所以，Supply的划分可以通过公共属性SNO实现，在SCITY属性值上完成水平诱导划分。这时，SCITY属性称为Supply的诱导属性。

Supply

SNO	PNO	QTY
S1	P1	320
S1	P2	200
S1	P3	450
S2	P1	300
S2	P2	250
S3	P2	100

Supplier

SNO	SNAME	SCITY
S1	Smith	London
S2	Jones	Paris
S3	Blake	Paris

Part

PNO	PNAME	ITEM
P1	IBM	Harddisk
P2	HP	Mouse
P3	DELL	Keyword

3.2.2.3.4 诱导分片

前例已有：$Supplier1=\sigma_{SCITY='London'}Supplier$

$Supplier2=\sigma_{SCITY='Paris'}Supplier$

通过半联接操作实现对Supply的划分，则

$Supply1=Supply \propto Supplier1$

$Supply2=Supply \propto Supplier2$

根据半联接表达式，则上式分别为：　[注：$R \propto S=\pi_R$（$R \propto S$）]

$Supply1=\pi_{SNO,PNO,QTY}(Supply \propto (\sigma_{SCITY='London'}Supplier))$

$Supply2=\pi_{SNO,PNO,QTY}(Supply \propto (\sigma_{SCITY='Paris'}Supplier))$

从式中可看出：Supply诱导水平分片的谓词有两部分，一部分是它与关联关系的公共属性；另一部分是它应满足的关于诱导属性的谓词。可表示为：

Q1：Supply.SNO=Supplier. SNO and SCITY='London'

Q2：Supply.SNO=Supplier.SNO and SCITY='Paris'

3.2.2.3.4 诱导分片

Supplier

SNO	SNAME	SCITY
S1	Smith	London
S2	Jones	Paris
S3	Blake	Paris

Supplier1(T1)

SNO	SNAME	SCITY
S1	Smith	London

Supplier2(T2)

SNO	SNAME	SCITY
S2	Jones	Paris
S3	Blake	Paris

Supply

SNO	PNO	QTY
S1	P1	320
S1	P2	200
S1	P3	450
S2	P1	300
S2	P2	250
S3	P2	100

Supply1(R1)

SNO	PNO	QTY
S1	P1	320
S1	P2	200
S1	P3	450

Supply2(R2)

SNO	PNO	QTY
S2	P1	300
S2	P2	250
S3	P2	100

图 诱导分片范例

3.2.2.3.5 四种分片操作的统一表示

四种分片操作可以用一个统一的形式描述：

$R(O_j)$，$<C>=\{R_1, \ldots, R_n\}$

其中，$O_j \in \{H,V,M,DH\}$, $C=\{C_1,\ldots, C_n\}$。

当O_j=DH时，C_i表示关系；

当$O_j \neq$DH时，$C_i=\{<A_i,P_i> |$ 如果O_j=H,则A_i=U；如果O_j=V，则P_i=True}。

操作的作用是将组合关系R（U,Q,S）按操作O_j分解成一组满足条件的片段（子关系）：$R_1(U_1,Q_1,S_1)$，…，$R_n(U_n,Q_n,S_n)$，其中，

当$O_j \neq$ DH时，$U_i=A_i$，K_i=K, $Q_i=Q \wedge P_i$，$S_i \neq \emptyset$，i=1，…，n；

当O_j=DH时，U_i=U, K_i=K, $Q_i=C_i$，$S_i \neq \emptyset$，i=1，…，n。

3.2.2.4 分片操作的正确性

四种分片操作均是建立在关系代数基础上的。一个关系经过选择操作分成若干子关系，子关系间遵守三个内部特征原则。

一个关系经过投影操作划分成若干子关系，它们共享关系的主键属性，且相交属性只允许是主键。这种划分后的内部特征的检查如下：

- **完整性检查**：垂直分片定义中已定义了子关系属性项，并是关系的属性项（$\cup A_i=U$），即不会出现游离的属性项；
- **重构性检查**：将子关系做自然联接则还原成关系；
- **不相交性检查**：对于垂直分片是允许在键属性上相交。

一个关系的混合分片的检查，实际上是重复上述两种分片的检查。两个关系间的诱导分片的检查，主要是对半联接作的检查，半联接操作是一种基于关系基本操作的导出操作（选择、投影、联接的复合操作）。

3.2.2.4 分片操作的正确性

例1：检查Supplier被水平划分成Supplier1和Supplier2两个子关系的分片正确性。

- 完整性检查：选择操作包括了所有 SCITY值，即不出现游离的元组项；
- 重构性检查：将两个子关系做并操作还原为原来的Suppler关系；
- 不相交性检查：选择操作的两个谓词为：

  Q_1：SCITY='London'，

  Q_2：SCITY='Paris'，

它们之间是互斥的，所以片段中不存在相同的元组项

3.2.3 数据分配原则及方法

3.2.3.1 数据分配的一般准则

分配4种类型：集中型、分割型、全复制型、混合型
评估4个因素：存储代价、可靠性、检索代价、更新代价

评估分析

集中型评估

- 没有片段间的通讯；
- 存储代价没有额外开销；
- 可靠性差；
- 检索代价猛增；
- 没有同步更新的开销。

分割型评估

- 存储代价没有增加
- 可靠性比集中型好
- 没有通讯代价问题
- 不存在副本同步更新

全复制型评估

- 可靠性好
- 存储代价剧增
- 检索代价随更新同步机制而变化（分析集中式更新同步技术和全局锁技术的检索代价与更新复杂性）

混合型评估

上述三种的综合设置，考虑更新/检索比

数据分配一般准则：

- 处理局部性
- 数据的可用性和可靠性
- 工作负载分布的均匀性

3.2.3.2 分配操作定义

定义1：

对于逻辑片段R（U，Q，S）的分配操作（AO），是将R分配到一组场地r_1，…，r_n上，使得每个r_i上有一个相应的物理关系Rr_i（Ur_i，Qr_i，Sr_i），满足：

$Ur_i=U$；　$Qr_i=Q$；　$Sr_i=Ø$。

记为：　$R(AO)\langle r\rangle=\{Rr_1, Rr_2 \ldots Rr_n\}$，　$\{r=r_1, \ldots, r_n\}$

3.2.3.3 分配操作方法

总体思路和参考方法：

利用大量的性能测试和“应用”优化。即：分别根据需求，使用以下三种方法求出收益值，分析最佳分配。

方法1：非冗余分配“最佳”法

- 对每个方案都进行比较，选择最佳的方案。

方法2：冗余分配“选择所有收益场地法”

- 在全部场地内选择一组场地，当片段的一个副本分配到这一组场地时，其收益高于所花费的代价，则该副本就放置在这一组场地上。

方法3：冗余分配“添加副本”法

- 先用最佳法决定非冗余分配方案，然后增加副本，使其仍能获得收益，再添加副本直到没有收益为止。

3.2.4 数据分布结构模式定义

3.2.4.1数据分布结构模式定义概述

全局关系模式： R（U，D，dom，I，F，Q，S）

或简化成： R（U，Q，S），

其中，S是关系的分布结构 ，是有关R被划分和分配的信息的集合。

- 如果R是全局关系，那么S记载了R如何分解成逻辑关系和物理关系；
- 如果R是逻辑关系，那么它还需分配场地，因此S记载R的分配信息；
- 如果R是物理关系，那么它无需划分和分配，因此S=Ø；

本节讨论如下要点：

- 如何将分片和分配的信息在一个分布结构S中进行描述
- 表示“数据分布的模式定义”的语句格式，即以语句格式表现数据分布的实际操作

3.2.4.2 用分解树理论讨论分布结构的定义

3.2.4.2.1 分布结构的粗定义

定义1：一个分布结构是一个有向图G=（V，E，L），其中：

1） V是节点集：每个节点由关系模式和在相应关系上的操作所组成，用$R_i(U_i,Q_i,S_i)(O_j)$表示；

2） E是边集：如果有任意分片操作$R（O_j）<C>=\{R_1,\ldots,R_n\}$，则有从节点R（U,Q,S）（$O_j$）到$R_1（U_1,Q_1,S_1）(O_1)$,…，$R_n（U_n,Q_n,S_n）(O_n)$的n条有向边[R（R,Q,S）（$O_j$），$R_i（U_i,Q_i,S_i）(O_i)$],每条边上有符号$C_i$，i=1,2,…，n。

3） L是符号集：它由所有边上的C_i所组成。

如果一个分布结构是一棵树，就称它为分解树。

3.2.4.2.1 分布结构的粗定义

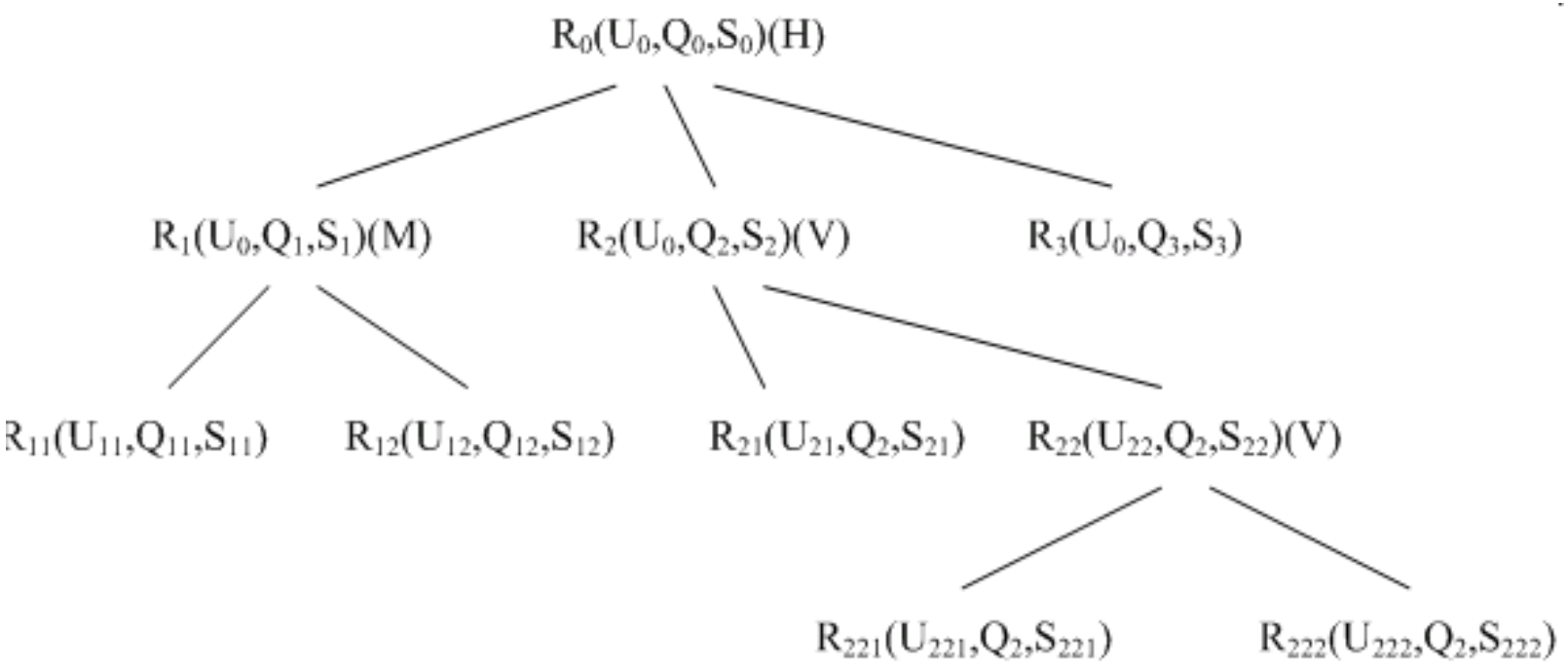


图 分解树结构

3.2.4.2.2 关于独立分片的分解树定义

定理1：在分解树中对任一符号为C_i=<A_i,P_i>的有向边[R（U,Q,S）(O)，R’（U’,Q’,S’）(O’)]，都有：

Q’=QΛP_i，U’=A_i，K’=K。

定理2：在分解树中，设任意两节点$R_0(U_0,Q_0,S_0)(O_0)$到$R_f(U_f,Q_f,S_f)(O_f)$间的所有边上的符号为 $C_1, C_2, \cdots, C_n (C_i =< A_i, P_i >, i = 1, \cdots, n)$，**有：**

1） $U_f = A_n \subseteq A_{n-1} \subseteq \cdots \subseteq A_1 \subseteq U$

2） $K_f = K$

3） $Q_f = P_n \Lambda P_{n-1} \Lambda \cdots \Lambda P_1 \Lambda Q_0$

4） $S_0 =$以$R_0(U_0, Q_0, S_0)(O_0)$ 为根节点的子树.

3.2.4.2.2 关于独立分片的分解树定义

定义2： 分解树是一棵树T=（V，E），其中：

1）V是节点集：每个节点结构为$R_i(U_i,Q_i,S_i)(O_j)$，Q_i是一谓词，O_j表示操作集{H，V，M}，对于根节点，有Q_i=True，S_i=T；

2）E是边集：对于任一独立分片操作，$R(O_j)<C>=\{R_1，\ldots，R_n\}$，其中，$C=\{C_1，\ldots，C_n\}$，$C_i=<A_i,P_i>$，（i=1,…,n）。则在分解树中，有节点$R(U,Q,S)(O_j)$到$R_i(U_i,Q_i,S_i)(O'_j)$的边，其中，i=1,…,n，且$U_i=A_i$，$K_i=K$，$Q_i=Q\wedge P_i$。

这个定义就是非常重要的**独立分片的分解树定义**

3.2.4.2.3 包含相关分片的分解树定义

定义3： 分解树是一棵树 $T=(V,E)$，其中：

1）V是节点集： 每个节点结构为$R_i(U_i,Q_i,S_i)(O_j)$,其中Q_i 是一谓词或是关系名，O_j 表示操作集$\{H,V,M,DH\}$, 根节点 $Q_i=True, S_i=T$

2）E是边集： 对于任一分片操作 $R(O_j)\langle\{C_1,\cdots,C_n\}\rangle=\{R_1,\cdots,R_n\}$,

如果，$C_i=<A_i,P_i>$，在分解树中有从节点 $R(U,Q,S)(O_j)$ 到$\mathbf{R_i(U_i,Q_i,S_i)(O_j')}$的，$i=1,\cdots,n$, 且$\mathbf{U_i=A_i}$，$\mathbf{K_i=K}$，$\mathbf{Q_i=Q\wedge P_i}$；

如果 $C_i\neq<A_i,P_i>$，那么 C_i 一定是关系名，这时 $O_j=DH$，在分解树T中有从节点 $R(U,Q,S)(DH)$ 到节点 $R_i(U_i,Q_i,S_i)(O_j')$ 的几条边，$i=1,\cdots,n$,且对每个 $R_i(U_i,Q_i,S_i)(O_j')$，有 $U_i=U,K_i=K,Q_i=C_i$

3.2.4.2.4 包含分配操作的分解树定义

定义4： 分解树是一棵树 $T=(V,E)$，其中：

1）V是节点集： 每个节点结构为 $R_i(U_i,Q_i,S_i)(O_j)$ 其中，

Q_i 是一谓词或是关系名或场地名，

O_j 表示操作集 $\{H,V,M,DH,$或空$\}$，对于根节点，有 $Q_i=True, S_i=T$

2）E是边集： 对于任一分片操作 $R(O_j)\langle\{C_1,\cdots,C_n\}\rangle=\{R_1,\cdots,R_n\}$，

如果$O_j\in\{H,V,M\}$，则有 $C_i=<A_i,P_i>$，在分解树中有从节点 $R(U,Q,S)(O_j)$ 到$R_i(U_i,Q_i,S_i)(O_j')$的边，$i=1,\cdots,n$，且$U_i=A_i$，$K_i=K$，$Q_i=Q\wedge P_i$；

S_i=以$R_i(U_i,Q_i,S_i)(O_j')$为根的子树，

如果$O_j\in\{DH,$空$\}$，那么

在分解树T中有从节点 $R(U,Q,S)(O_j)$ 到节点 $R_i(U_i,Q_i,S_i)(O_j')$ 的边，$i=1,\cdots,n$，且对每个 $R_i(U_i,Q_i,S_i)(O_j')$，有 $U_i=U,K_i=K,Q_i=C_i$

（注：“空”时，Q_j 表示场地名）

3.2.4.3 数据分布模式定义

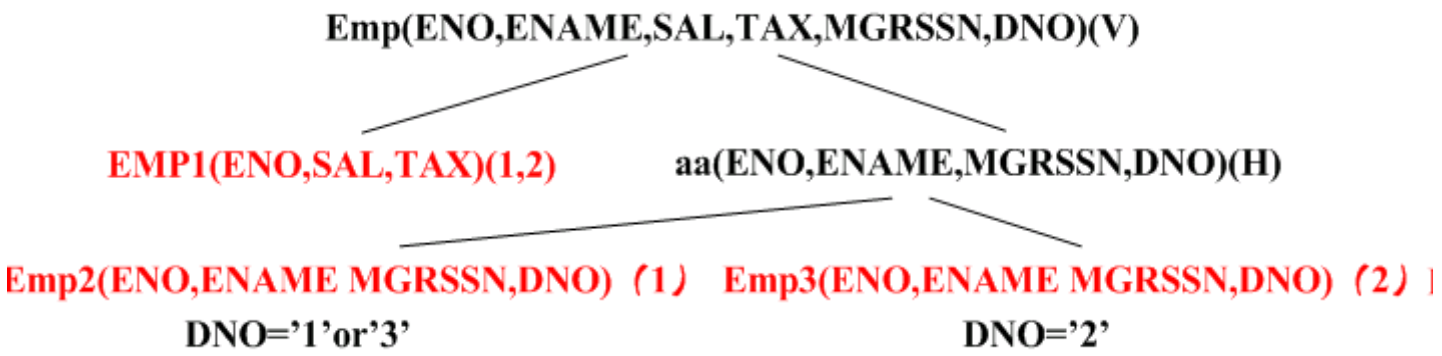
DDB的数据分布在四层模式中属全局概念层的描述，它是分布式数据库的整体抽象，也是分布式数据库与集中式数据库抽象的最特殊的一点。这里讨论描述全局概念层中关于数据分布的模式定义语句，而四层模式中其它三层的模式定义与集中式数据库类似。

在全局概念层中，有全局概念模式、分片模式、分配模式三种模式。所以，对全局层模式定义语句应包括这三方面的描述，语句的词法和语法视设计的风格而定。本课程采用武汉大学数据库研究组研制的WDDBS的L-SQL*语言。

3.2.4.3 数据分布模式定义

例6 有例2的职员关系Emp，将它按例3所示混合分片划分Emp时，用L-SQL*语句如下书写：

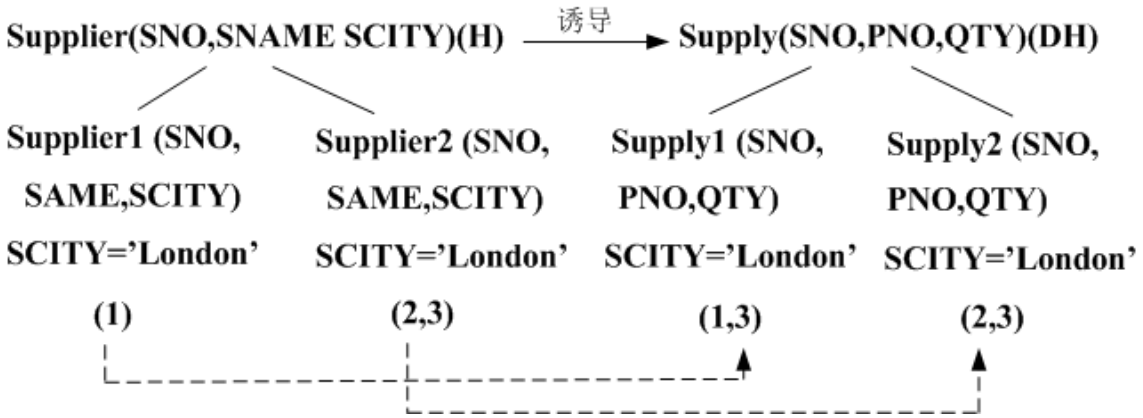
Create Table Emp (ENO n 4 ENAME c 12 SAL n 5 TAX n 4
MGRSSN n 8 DNO n 2)
Key ENO
Fragment aa=Emp (ENO,ENAME,MGRSSN,DNO)
Emp1=Emp (ENO, SAL, TAX)
Emp2=aa (DNO="1" or DNO="3")
Emp3=aa (DNO="2")
Site Empl(1,2) Emp2(1) Emp3(2)；



3.2.4.3 数据分布模式定义

例7 给出例1和例4的分片及分配模式定义语句

Create table Supplier (SNO n 4 SNAME c 12 SCITY c 10)
Key SNO
Fragment Supplier1=Supplier(SCITY='London')
Supplier2=Supplier(SCITY='Paris')
Site Supplier1 (1) Supplier2(2,3)；
Create Table Supply (SNO n 4 PNO n 4 QTY n 5)
Key SNO PNO
Fragment Supply1=Supply SJ［SNO=SNO］Supplier.Supplier1
Supply2=Supply SJ［SNO=SNO］Supplier.Supplier2
Site Supply1(1,3) Supply2(2,3)；



3.3 自顶向下设计分布式数据库

DATAID-D方法，是一种典型的自顶向下设计分布式数据库方法

3.3.1 DATAID-D方法概述

- DATAID-D方法，是一种典型的自顶向下设计分布式数据库方法，由意大利米兰工业大学提出
- 该方法强调的是给设计者提供一种方法学框架结构，指明有关的设计问题，为解决这些问题需要哪些参数，以及如何解决这些问题。
- DATAID-D方法是对集中式数据库设计DATAID-1方法论的扩充，在原有的需求分析、概念设计、逻辑设计和物理设计四个阶段的基础上，增加了两个阶段：
- **分布要求分析阶段**
  - 收集关于分布的信息，如水平分片的划分谓词，每一应用在各站点激活的频率等
  - 为了收集分布信息，必须从概念设计阶段的某些结果出发来收集关于分布的要求，因此，分布要求分析阶段位于概念设计阶段之后
- **分布设计阶段**
  - 这一阶段始于全局数据库模式的规格说明和所收集的分布要求，然后，产生全局数据的分片模式和片段的位置分配模式
  - 该阶段的输入是以前设计阶段产生的数据与应用规格说明，其形式为全局数据库模式和逻辑访问表。全局数据库模式使用ER模型描述，逻辑访问表指明各应用在每个实体上所执行的数据库访问操作的类型和频率。

3.3.1 DATAID-D方法概述

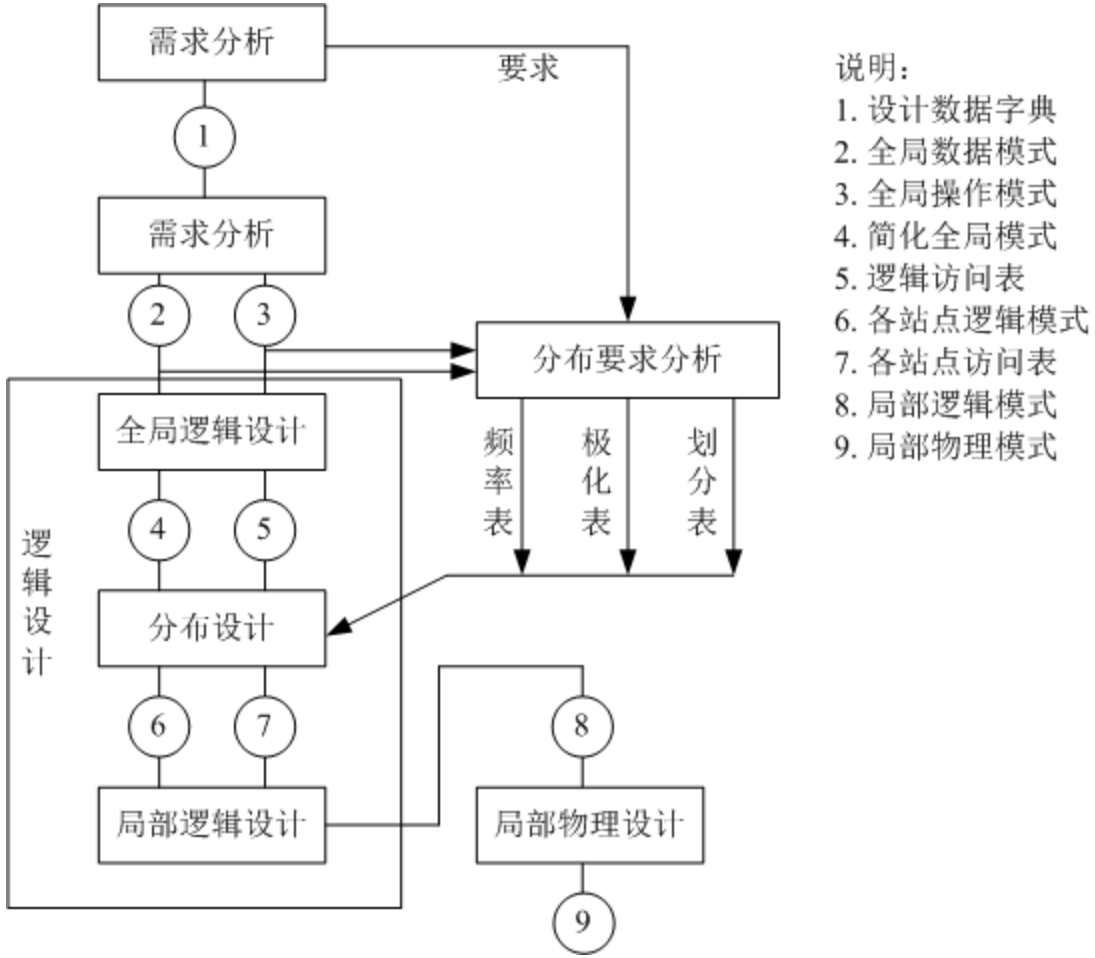


图 采用DATAID-D方法时的分布式数据库设计步骤

3.3.2 分布要求分析阶段

- 分布要求分析的目的是收集以后用于推动分布设计所需要的信息。
- 这一阶段的输入是用户对分布的要求和全局数据概念模式与操作模式。
- 这一阶段的输出是三种类型的表：
  - 应用的频率表
    - 给出各个站点上每一应用的激活次数
  - 实体的划分表
    - 指明可用于模式中各实体的潜在水平分片规则
  - 数据与应用的极化表
    - 基于定量分析方法来说明分片如何影响应用处理的本地性。一个极化值表指明由一给定场地发出的一给定应用访问一给定片段的概率。

3.3.3 分布设计阶段

- 分布设计的目标是从全局数据模式、逻辑访问表和分布要求出发，将数据分配在站点上。
- 分布设计阶段的输出是各站点的逻辑模式和逻辑访问表。在以后的局部逻辑设计阶段和在各场地上独立进行的物理设计阶段，要使用这些逻辑模式和访问表。
- 分布设计包括四个阶段：
  - 分片设计
    - 对实体进行水平和垂直分片等分片操作
  - 非冗余分配
    - 把各片段分配到使用最多的场地上
  - 冗余分配
    - 使用贪婪启发式算法，可以采用前面讲过的**冗余分配“选择所有收益场地法”和“添加副本法”**
  - 局部模式的重新构造
    - 重新构造片段分配场地上的局部模式

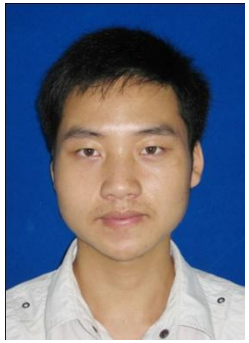
3.4 自底向上设计分布式数据库

- 把现有数据库集成起来构成分布式数据库时，可采用自底向上的方法。
- 这种方法的重点是把现有的各种不同的数据库模式集成为全局模式。所谓集成就是把公用数据定义合并起来，并解决对同一个数据的不同表示方法之间的冲突。
- 自底向上的设计方法，不适宜于水平分片关系的设计。同一个全局关系的各个水平分片必须具有相同的关系模式，但是，自底向上设计分布式数据库时，都是事先已经独立设计各个场地的数据库，然后，才把它们集成起来，各个数据库的模式很难相同。因为水平分片是分布式数据库一个重要特点，所以，在修改过程中，应该设法修改本地关系的定义，这样就可以把本地关系看成是一个公共全局关系的水平分片。
- 自底向上的设计方法要解决以下三个问题：
  - 选择公用数据库模型来描述数据库的全局模式
  - 把每个站点上的本地模式翻译成公用的数据模型
  - 把各个站点上的本地数据模式集成为一个公用的全局模式

附件：主讲教师和助教

主讲教师：林子雨

单位：厦门大学计算机科学系
E-mail: ziyulin@xmu.edu.cn
个人网页：http://www.cs.xmu.edu.cn/linziyu
数据库实验室网站：http://dblab.xmu.edu.cn



助教：赖明星

单位：厦门大学计算机科学系数据库实验室2011级硕士研究生
E-mail: mingxinglai@gmail.com

Department of Computer Science, Xiamen University, September, 2012