

分布式数据库设计方法学(续)

Stefano Ceri, Barbara Pernici和Gio Wiederhold

2) 飞机订票系统中的分布要求分析 这一节考察在分布要求分析阶段之后, 如何用 DATA_ID_D 方法学表示分布要求。

图4(b)中示出的频率表说明了图3中在结点1(丹佛), 2(纽约), 3(亚特兰大)上全局操作模式所描述的应用a, b, c的频率。

图5示出了飞机订票数据库的划分表。图5(a)给出实体机场和旅客的基本划分表。设计者将区域属性选为机场实体的划分准则, 并且将电话号码前三位(区域码)选为旅客实体的划分属性。谓词选择性用每一可能的划分属性值给出该实体元组的百分率。

基本划分表			
实体	划分名	谓 词	谓词选择性
机场	区 域	区域='W'	30
		区域='N'	45
		区域='S'	25
旅客	电 话 (前3位)	电话='415*'	35
		或'408*', 或...	
		电话='904*'	35
		或'713*', 或...	
		电话='212*'	30
		或'617*' 或...	

(a)

然后设计者考虑由机场到区域的基本划分产生的可能导出划分。图5(b)中考虑了四种导出划分选择,

• 可以用两种方法来划分班机实体: 基于关系“从”(起飞机场)或“到”(到达机场)和把机场划分成区域。注意, 依据使用两个不同关系, 同一基本划分将产生两种不同导出划分。

导出划分表				
实体	导出相关	基实体	划分名	谓词选择性
班机	从	机场	区域	同
班机	到	机场	区域	同
旅客	订票	班机	第一订票地点	同
旅客	订票	班机	班机起飞区域	注释

注释: 班机起飞 划分基于7个谓词:		谓词选择性
P1: 所有班机由离开区域A的旅客预定		20
P2: 所有班机由离开区域B的旅客预定		20
P3: 所有班机由离开区域C的旅客预定		20
P4: 所有班机由离开区域A和B的旅客预定		10
P5: 所有班机由离开区域A和C的旅客预定		10
P6: 所有班机由离开区域B和C的旅客预定		10
P7: 所有班机由离开所有区域的旅客预定		10

(b)

图 5 (a) 基本划分表。(b) 导出划分表

• 依据关系从预定到班机, 按起飞区域划分, 最后两行给出了划分旅客实体的两个其它方法。注意, 在这种情况下, 导出机制分两步, 因为它从机场作用到班机和然后从班机作用到旅客。在前一情形中, 依据他们第一次预订的班机来划分旅客; 在后一情形中, 依据它们各次的预订来划分旅客。注释表中说明了七种可能情形: 旅客可能预定只离开一个区域(A, B, C)的班机, 或离开两个区域(AB, BC, AC)的班机, 或离开所有区域(ABC)的班机。由于订票是一种多对多关系(每个旅客可能多次订票), 所以需要以上七种情况, 然而照分割的定义, 我们想把现实世界中的每个旅客映射到旅客实体的恰

好一个实例。在第一种情形，当进行第一次预订时，静态地给每一旅客示例分配一划分，而在第二种情形，旅客示例到划分的映射是动态的：凡某一旅客更改订票时，相应的实体示例就可能从一个划分移动到另一划分。

图6示出了极化表。列关系到每一结点上应用的激活，行关系到划分谓词。如果选择了特定的划分标准，那么每一实体表明访问一个段的百分率。在实践中，只说明少数实体；而其余的实体可以通过假设其余示例的一致分布来进行计算。有些应用不使用实体，所以某些子矩阵不相关，在表中就画上叉。

		a			b			c		
		1	2	3	1	2	3	1	2	3
按区域划分机场	P1	80						100		
	P2		75			×			100	
	P3			80						100
...	...									

图6 极化表

为了说明，让我们讨论第一个子矩阵，它与订票应用(图3(a))有关联并且使用按区域划分机场实体(P1代表区域1, P2代表区域2, P3代表区域3)。这个子矩阵表明，如果我们按区域划分机场，那么在区域1发出的关于订票查询有80%的概率是关系到区域1的机场的。对于第1列中的其它两个位置，我们假定其余20%访问是均匀的。

3) 飞机订票系统中的分布设计 分布设计由四步组成：对每一实体选择分割准则，确定非冗余分配，在非冗余分配上引入冗余，最后在每一结点上重新构造局部模式(见图7)。

a) 分割设计 给定了一些包含在极化表中的可能划分准则，这里设计者必须为各实体选择最合适的准则，并确认划分自身

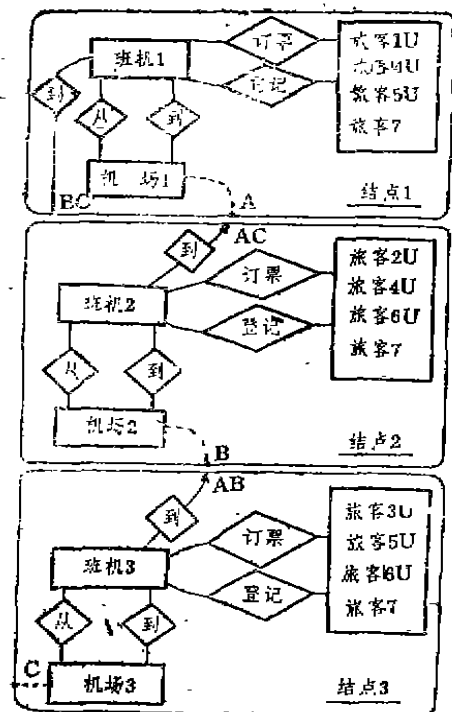


图7 分布设计结果：结点1-3的局部模式

是方便的。这要求定量分析有关的应用，可以把这些应用分成三类：通过划分变得容易的应用，变得更困难的应用和不受影响的应用。这样，如果第一类“大于”第二类，那么划分是方便的。在我们的实例研究中：

1) 对于确定分配单位，垂直划分是没用的，事实上，通过垂直划分，没有哪个应用是明显容易的。

2) 相反地，所有实体都有水平分割。

· 机场实体有一个基于区域的水平分割(段机场1, 机场2, 机场3)；

· 航班实体有一个基于起飞机场的导出水平分割(段航班1, 航班2, 航班3)；

· 旅客实体有一个基于旅客预定的所有航班起飞的导出水平分割(段旅客1到旅客7)。

b) 非冗余分配 在有些情形中，根据选择的划分准则，容易得出非冗余分配。例如，机场1, 航班1和旅客1被直接分配在结点1，同样地，机场2, 航班2和旅客2被分配在结点2；机场3, 航班3和旅客3被分配在结点3。对于实体旅客的其它段，我们必须根

据极化表和频率表来选择使用该段最多的结点；因此我们把旅客4，旅客6和旅客7分配到结点2，旅客5到结点3。

c) 冗余分配 在许多情形中，与冗余有关的代价超过了同一实体所有段的效益。机场和班机实体就是这种情况。对于旅客实体段，我们引入有限的冗余度：旅客4，旅客5和旅客6，相应于预定离开两个区域的班机的旅客，被存储在两个相应的结点上。同样地，旅客7(包括乘坐离开所有三个区域班机的旅客)被存储在各结点上。

d) 局部模式的重构 这一步主要涉及ER全局数据模型关系的分割和分配。根据上述的段分配，关系订票和登记碰巧有一自然分配，使得所有连接都是局部的(因为各个旅客在他预定或登记的各结点上与一个旅客示例相联系)。“从”关系也是自然分配，因为机场和班机实体是根据“从”关系进行水平分段的；因此所有的连接都是局部的。不过，得讨论“到”关系的分布，因为它连接可能存储在两个不同结点上的实体示例。我们在解决时，“到”关系示例被存储在与班机实体的相应示例相同的结点上。

这种解决办法的最大特征是，通过只查看分配在该班机起飞结点上的数据，就可以回答班机的所有请求；当数据库受到大量使用时，为准备班机起飞，不需要远程信息。这种解法的缺点在于，旅客信息重复，并且在进行订票时必须仔细地管理旅客信息。

四、 自底向上分布设计

在自底向上方法中，表示存储在各单个结点上的数据部分的模式构成设计的出发点，并且分布设计包括弄清哪些数据是模式共有的，以及其差别如何。

在操作期间，大多数多数据库系统只提供全局查询能力和局部更新能力，因此各个局部系统只能通过该结点发送的事务进行更新。如果设计者不能修改多数据库系统中的

本地数据库，那么必须在系统的查询处理能力中引入冲突判决。

多数据库支撑提供一种查询的自动映射，将按照全局视图发送的查询映射成适合局部模式的查询，并且协调查询的执行和结果的收集。

4.1 构造全局模式的设计问题 多数据库系统的自底向上设计中的问题是需要构造一个全局模式(亦称超视图^[12,36])。综合过程识别实体与其属性的匹配。

为了综合数据库，我们必须为全局模式选择一个合适类型的数据模型。视图综合的以前研究工作已说明了概括分层结构有支持视图综合的能力。概括分层允许定义两个实体间的类型子类型关系；当两个视图对同一实体进行部分相交描述时，这一点很有用。这样，视图综合问题的经典解法是生成三个实体，一个具有共同属性，另两个具有非相交属性^[22]。在图8示出的例子中，班机实体的两个视图具有某些共同属性；在全局视图中，共同属性与子类型相关联，并且对包含非相交属性的各个视图生成一子类型。双线箭头表示概括分层。

之所以需要概括分层，是因为如象(用概括分层扩充的)ER模型，结构模型，或功能模型之类的概念模型都是视图综合过程的良好候选对象。在本文，我们使用以概括分层扩充的ER模型。

在图8的例子中，班机实体的两个视图具有一些共同属性；在全局视图中，我们生成一概括分层结构，有一个超类型和两个子类型；超类型具有共同属性，而子类型各对应一个视图；“差”属性与子类型有关。

另一个一般性问题是视图综合的次序。当出现几个视图时，一般通过一次把一个视图和全局模式合并来进行综合，逐步构造起全局视图。因此我们考虑的一般问题是如何构造两个视图的超视图。通常最好首先综合最大或最重要的视图，然后跟着综合最小的或最不重要的视图。

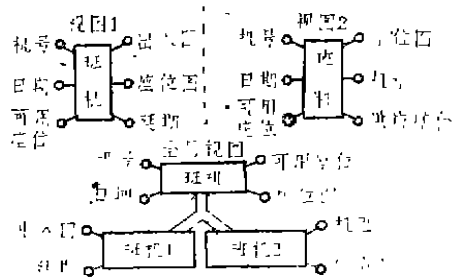


图8 使用概括分层的两视图合并

1) 识别相似性 综合两个模式的第一步是识别它们的相似性；这些相似性是综合的出发点。因为模式相交部分的命名相似性或结构相似性，可以识别匹配。从先前存在的数据库或文件中数据的相似性也可以推得匹配。相似的值集表明相交。通过比较属性的投影，可以匹配属性域。观察文件或域具有相同或近似相同的基数性，可以得到初步提示。

如果在不同数据库位置上的不同应用使用了同一源数据副本，那么数据库之间肯定有某些相交。

2) 识别冲突 综合也应弄清楚冲突，即弄清不同模式中相似数据的不同表示或域定义^[2,22,23]。通过在全局模型中引入差异或在源模型中做一些折衷，可以解决冲突^[47]。对于设计时不能解决的冲突，需要选择策略回答有不一致数据的查询。

模式差异包括命名冲突、定标差异和结构差异。

• 命名冲突 命名冲突有两种类型：同物异名和异物同名，前者出现在表示同一世界对象的数据对象在两个视图中给以了不同的名字之时；后者出现在两个视图中具有同名的数据对象表示不同的现实世界对象之时。一旦检测到命名冲突，通过在全局模式中存储名字对应表就能方便地解决。

• 域差异 最麻烦的问题归咎于域差异。在处理工资的结点上，一个关系人员也许不含合同承包人员，因为不直接给他们付工

资。在管理项目的结点上，包含合同承包人，但不包括没参与任一项目的审计人员。检测这些问题最好是通过比较源数据库或文件并注意不一致性来进行。概括分层可以用来表示这一问题的解。在上述情形中，正式人员、合同承包人员和审计人员都可以是同一实体人员的子类型。根据应用使用的子类型，用不同的方法来执行工资这样的应用。

• 定标差异 在具有同一数值的不同视图中可以见到定标差异。若可能，应使用更精确的定标来检索数据，并使用换算公式进行连接或输出。如果分布式DBMS可以支持换算公式，那么它们应作为全局模式的一部分加以存储。

• 结构差异 结构差异可以归咎于各视图中不同设计选择；例如，同一现实世界对象可能在一个视图中作为一属性进行建模，而在另一个视图中作为一个实体进行建模。使用全局模式中的概括只能处理少数结构差异。在视图设计中，一般通过改变一个或两个视图来解决结构差异。当涉及自主数据库时，如果分布式DBMS可以支持复杂的查询修改，那么解决结构差异要求编写这样的查询修改过程，并存放在全局模式中^[18]。

在综合前应报告和确定上述种种冲突，接着应修改局部系统以反映可能出现的综合。否则，全局模式应含有关于冲突的信息和判决冲突的策略。显然，这一节提及的换算公式管理、查询修改过程或冲突判决策略要求沿着更完善的多数据库系统方向扩充传统DBMS的能力；没有这种扩充，则必须使用应用程序来精心支撑上述特征。

3) 处理操作期间的不一致数据 实际上，工作的多数据库有错误，每存储记录高达1%的错误是常见的。这些错误可能归咎于输入抄写，同步更新中的故障或遗漏，和从系统错误的非正确恢复。数据库设计者必须决定处理全局数据库工作期间出现的不一致性的策略。

例如，考虑这样的情形，存储在不同结

点的雇员实体的两个示例碰巧有相同的标识符,但薪水属性值不同。这种情形可能有几个原因:

- 同一标识符对应于两个不同雇员是可能的;当从数据库抽取值时系统地进行修改,就可以解决这种情况。例如,可以引入系统范围的独特标识符,把所有局部标识符和结点标识符串起来。

- 另一种可能性是,同一雇员在两个结点有两种不同的工作,并且薪水属性正好涉及到这两种不同的工作。在模式综合级可以考虑这种情况,把两个薪水属性处理成异物同名。否则,有可能管理这种数据不一致性,在全局模式中指明,关系到全局薪水的所有查询应计算这个薪水为两个局部薪水之和。

- 不一致性的第三个原因可能是逐渐过时所致。这种情况也可以在模式级解决,把两个薪水处理成异物同名(即老薪水和新薪水)。否则,有可能在查询修改级工作,指明应使用最新值(在这种情况下,薪水属性必须盖上时间戳,显然这是一个额外的主要费用)。

- 上述的这种不一致性的第四个原因是由于不合逻辑错误的实际不正确性。

以上四种情况表明,数据库设计者对如何管理不一致性有多种选择。当在执行时检测到不一致性时,确定解决不一致性的策略是一个设计问题。这些策略包括:

- 显示任一不一致值而不通知用户:这是最直截了当,同时也是最危险的解决办法。

- 显示所有不一致值,并告诉用户信息源。在这种情况下,用户应能评价不一致性的原因。

- 求值不一致值的某些组合函数,并给用户显示此函数的结果。可能的组合函数包括求平均、求最小值、求最大值。使用这种技术是在不同时间出现时预期观察值也不同的场合^[4]。

- 显示最新值。这一策略需要更新操作

的时间戳(肯定代价大些)。它所依据的假设是不一致性归因于更新不及时,因此,最新的值也是最可能的值。

- 显示最可靠系统的值。这一策略所依据的假设是,设计者可以评价分布式数据库中结点的可靠性。

4.2 自底向上综合的一个例子

假设两条可供使用的航线决定有一个多数据库系统以便从两个公司的任一办公室查询班机可用情况。设图2中示出的并在3.3节讨论的概念模式与航线A联系,设图9的概念模式代表航线B的数据。让我们观察一下航线B模式的特征并将它与航线A的模式进行比较。显然,新的模式更简单;特别是,没有关于机场和登记的信息。关于班机起飞和到达的信息被表示成实体班机的属性。然而,关于旅客和班机的基本信息却十分类似。

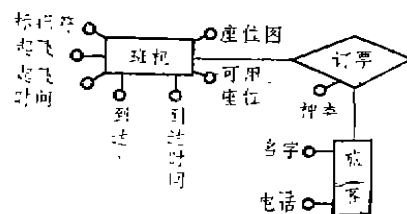


图9 航线B的概念模式

图10表示综合后建立起来的全局模式。一概括分层被用来表示两个子类型班机A和班机B,而子类型班机包含所有公共属性。这些属性包括班机标识符、模式A中被调用的班机号;这里,识别一个同物异名。公共属性还包括班机日期(在属性起飞时间的第一位置中进行编码);这里,需要进行更难的分析。实体班机A通过关系‘登记’与实体旅客相关,表明这一信息只为模式A的一些实体所用。最后,对实体班机A和班机B,用不同的方法表示有关起飞和到达机场的信息,这反映了不同的表示。注意:对两个局部模式,应以不同的方法翻译一个需要有关班机从一给定机场起飞的信息的查询。

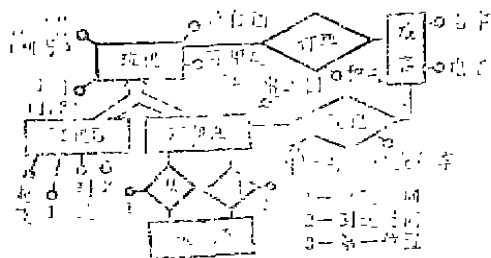


图10 综合后建立的全局模式

4.3 自底向上分布设计的早期工作

本节许多想法出自Dayal和Hwang^[18]的文章。Breitbart和Paolini^[6]、Litwin^[20]、Wong和Bazex^[43]以及Dayal^[17]也讨论了多数据库的综合。这些文章描述了应如何把查询分解成各局部系统的子查询；我们不讨论这一课题，因为这一课题更多的是关系到多数据库系统的开发，而不是这种系统的应用设计。

EIMasri和Wiederhold^[22]、Batini、Lenzerini和Moscarini^[21]以及Navathe、Sashidar和EIMasri^[26]在集中式系统的数据库设计框架中主要讨论了视图综合。虽然视图层次建模的方法稍有不同，但在视图综合方面的大多数参考文献都建议在视图综合中使用概括分层，其中有文献[22]（使用结构模型），[3]（使用EER模型）和[18]（使用功能数据模型）。

4.4 自顶向下与自底向上方法间的相互联系 在第二节，我们介绍了单纯的自顶向下方法，其中数据库设计者在进行概念设计时，忽略了任何物理细节（包括分布）。尽管这种方法理论上很有价值，但在一些实际场合中不大适用。例如，假设一数据库描述一个由功能组成的企业，每个功能直接映射到一数据库的位置上。

即使设计是从头开始，但在这种情形中分布信息的提取可能不自然。一种可能的设计方法是部分使用自顶向下和部分使用自底向上，对要求进行收集并对各功能独立地进行概念设计，然后综合所有概念模式成一个全局模式，最后重新分布信息。显然，使用这种方法，我们保留了视图的概念，其中每一对象是原先定义的，又因为这种概念将暗示着对象的分配。不过，对于引入复制和将一些对象从一局部模式“移”到另一局部模式，重新分配是十分有用的。

总之，一种纯的自顶向下分布设计方法在视图与分布不对应时是合理的，而自底向上方法在视图与数据库位置直接对应时是合理的；还可能存在除以上两种情形以外的种种中间情形，对于这些中间情形，应使用中间方法（部分自顶向下和部分自底向上）处理。

五、 结 论

随着分布式应用变成一种现实，分布设计成了数据库设计中的一个新的不可缺少的领域。分布需要自己的理论、问题定义、求解方法和方法学。

本文是自顶向下和自底向上分布设计方法的一个综述，主要论述了DATAID、D自顶向下方法学，举例说明这些方法并比较之。

参考文献（共48篇略）

（钟 源 译自proceedings of the
IEEE, vol. 75, No. 5, 1987,
P. 533-540 声 钟校）

下期主要内容预告

连接机制：向传统计算机结构的挑战；第六代计算机的核心——神经网络计算机，神经网络理论——学习、识别与计算的现代途径；人工智能与软件工程交叉领域分析；分布式数据库系统的现状和未来；程序设计的法则；新一代计算的若干问题；