计算机科学 1994Vol. 21 №. 3

工程数据库模式设计的一种形式方法——尼杰森法

William J. Rasdorf 和 Osama Y. Abudayyeh

Rasd., WJ 李刚译

摘 要 本文介绍一种新的建立数据模型的方法，叫做尼杰森信息分析法(Nijssen's Information Analysis Methodology，NIAM)。NIAM 是一种图形建模语言，所设计的概念模式能被映象成数据库模型(例如关系和层次数据模型)。文中阐述了把 NIAM 概念模式映象成第五范式关系数据模型的步骤，叫做最优范式算法。

一、引言

工程系统中的数据不仅对于分析工程系统的性能很重要，而且在设计将来的系统时也是极为有用的。不幸的是，工程数据模型没有适用的建模方法，几乎全是用特定的方式进行开发，结果数据库模式不是最优的。本文介绍一种形式化建模方法，叫做尼杰森信息分析方法 NIAM(Nijssen's Information Analysis Methodology)，它所产生的数据模型，实际上是一份系统地表示很好的数据模型设计的工程图[1]。然后 NIAM 模型可以映象成任何一种数据库模型。本文中我们将其映象成为关系数据模型，因为关系模型提供了适合于工程应用的公认的标准化数据存贮和恢复机制，而且工程应用中关系模型正在增加。

二、信息工程

设计和开发一个信息管理系统，很类似于开发工程系统，它至少包括三个主要阶段：概念化、设计、开发(图 1)。

概念化包括识别问题和确定所希望解的规格说明集。对于一个信息系统，这包括确定数据项和用于采集与存贮数据的处理模型。这个阶段的产品应是所希望的工程信息管理系统的一份形式化规格说明，即所谓基本事实。

设计包括在概念化阶段所产生的规格说明集的基础上，开发一套工程图以提供所希望解的细节。对于一个信息系统，这包括对上一阶段确定的数据建立系统化的模型，在形式化说明的上下文内，对数据存贮和应用得出最好的结构和组织。在此阶段，用 NIAM 作为系统化的建模方法学。NIAM 在设计阶段的产品是一份图形表示的概念模式图，可以看作是一份工程图。事实上，NIAM 图已经由桑迪亚实验室用作为工程图，得到委托方的认可之后，方可进入开发阶段[2]。

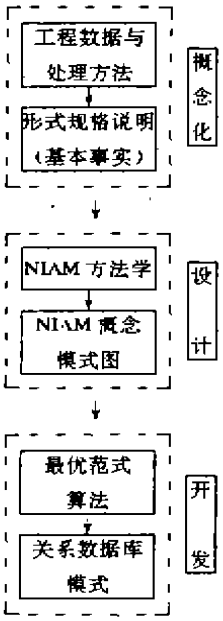


图 1 工程数据库的开发过程

开发包括利用设计阶段产生的工程图建造实际的工程制品。对一个信息系统，这包括进行计算机系统的物理开发。这个阶段利用最优范式算法把 NIAM 概念模式图转换成关系模式，然后在计算机上用一种数据库管理系统软件包来加以实现。

三、用 NIAM 形式方法做概念化设计

从考察信息系统的输入开始，为开发该信息管理系统的规格说明建立概念化模型。一般情况下，这些输入可以在报表、文档、输入表格和图中得到。这些类型的输入用于表示与给定领域有关的数据项的

含意和组织。然后，用 NIAM 形式方法(语言)对基本事实做公式化和形式化表示，得出称为基本事实的一个规格说明集。

基本事实来源于上述输入中抽出的"有代表性的事例"。基本事实是一种不能分割为更小的且仍维持原来信息的事实。在建立这些事实时，人们应该找出有代表性和有意义的事例(事实)集。称一个事例集是有意义的，是指它提供了与被建模的领域有关的相关信息和约束条件。

在 NIAM 中，把感兴趣的领域(有时称为论域，UoD)看作为起作用的实体集。UoD 中的一个事实类型确定某些实体，在一种关系中起某些作用。每个实体有一个类型(type)，定义它的所有可能的实例集。一个事实类型中一个实体类型的一个实例叫做标记(label)。实体类型的测量单位或参照基(reference base)叫做参照方式。每个实体类型必须至少起一种作用或作为某种关系中的组成部分。每一实体类型准确地起一种作用。作用名在事实类型的上下文中应该是唯一的。每个事实类型均有一个确认值(arity)，指明所包含的实体类型数。

表示事实类型所用的简写符号是：实体用实体名大写的第一个字符表示；参照方式括在括弧中；文本标记括在单引号内；数字标记写成数字；作用名用斜体字写成。此外，事实类型名写作黑体字，后跟一个冒号，其前面是基本事实实例。二元(确认值 2)和三元(确认值 3)事实类型的例子是：

- **WP-Level-Of-Detail**: WBS-Node (code) 134123 *has* Level-Of-Detail(number)6;
- **Material Budgets**: Control-Account(code)134123 *has-budget* Quantity(number)100 *for* Material (code)100.

四、用 NIAM 图形语言做设计

在从事例中抽出了基本事实之后，需要用一种标准的形式化方法(或语言)来表示之。NIAM 提供了一种形式图形语言来表示事实和作用[4,6]。事实的图形表示称之为概念模式图(CSD)。图 2 示出了上面所举的二元和三元事实类型的 CSD。在这种图形语言中，采用以下一些符号：

- 椭圆表示实体类型，实体名写在椭圆内，参照方式写在名字下面，并括在括弧内；
- 矩形表示作用；
- 用线段将实体类型和它所起的作用相连；
- 连续 n 个作用矩形序列，每一个都确切地与一个实体类型相接，表示一个 n 元事实类型；

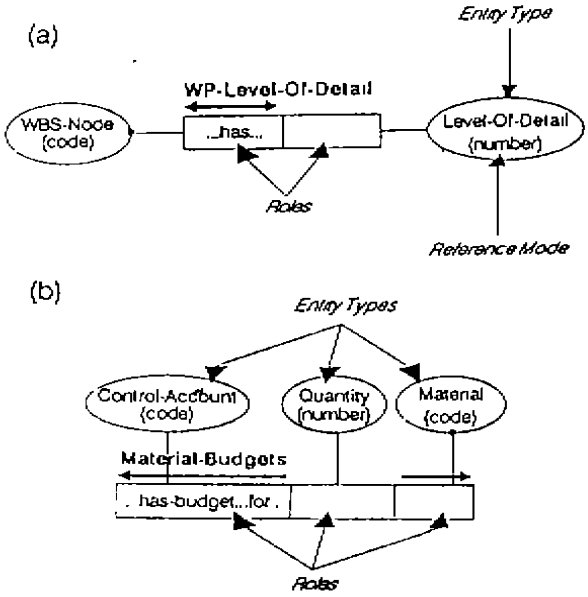


图 2　二元事实类型(a)和三元事实类型(b)的例子

- 作用写成谓词，它带有用'……'表示的空白间隙，写在 n 元事实类型的一端的矩形内。该记法表示第一个实体类型占据谓词中的第一个间隙。其余的间隙由事实类型的其余实体类型按从左到右顺序占据。

为了理解以上所列举的符号，让我们来看一下图 2(a)。图中，WBS-Node 和 Level-Of-Detail(在椭圆内)是二元事实类型 **WP-Level-Of-Detail** 的两个实体类型，"code"和"number"(在括弧内)是它们的参照方式。两个实体类型所起的作用表示为谓词"…has…"(在矩形内)，其中 WBS-Node 占据第一个间隙，Level-Of-Detail 占据第二个间隙。这样，此事实类型用英文一般地陈述为：a WBS-Node code has some corresponding numeric Level-Of-Detail(WBS-Node 代码有某个对应的数值 Level-Of-Detail)。

4.1　对概念模式求精

对概念模式求精包括消去不必要的事实类型、不必要的实体类型或者是二者兼而有之。所谓不必要的事实类型，是指一个事实类型可以从另外的事实型推出，而不必要的实体类型是指两个不同的实体类型可以组合为一个实体类型。这种组合的一个良好特征(indicator)是二者具有相同的参照方式基[4,6]。然而并不是具有相同参照方式的所有实体类型都需要组合起来，这要根据实体类型和它们建立模型的各自信息片断(piece)作出决策。例如，在图 2(b)中所示的 Control-Account 和 Material 实体类型都

有'code'作为它们的参照方式.但由于 Control-Account 实体类型表示的信息不同于 Material 实体类型表示的信息.所以这两个实体类型就未加组合。

在一些情况下.宁愿采用嵌套来代替使用展平(flattened)表示法(version)(事实类型无嵌套)。嵌套是由 NIAM 提供的一种机制.把实体类型之间的关系看作是一个实体类型自身.这种实体类型叫做具体化的关系类型(objectified relationship type)。正如实体之间的关系是由作用组成的一样。具体化的关系类型也是如此。包括具体化的关系类型的事实类型叫做嵌套的事实类型。像任何其它的实体类型一样.具体化的关系类型必须至少起一种作用.在 NIAM 中用一个将作用围在其中的椭圆表示。但是人们怎样来决定是采用嵌套表示法还是展平表示法呢?为回答这个问题.试考虑下面的通用法则:每当两个事实类型共享它们的所有实体类型且对这两个事实类型中的相同作用施加了唯一约束条件时.就采用嵌套的事实类型;否则.宁可采用展平表示法。有关嵌套机制的细节.可以在文[4]和[6]中找到。

为了理解嵌套的事实类型和具体化的关系类型的概念.我们来考虑一下图 3(a)中示出的两个事实类型(**Regular-Hours** 和 **Overtime-Hours**)。对这两个事实类型采用嵌套机制。图中所示的这两个事实类型叫作展平表示法.与下面将要简要研究的嵌套表示法相对应。采用嵌套机制.将 **Regular-Hours** 和 **Overtime-Hours** 修改成以下的嵌套表示法:

- **Regular-Hours**: Worker-ID(number)101 *has worked on* Task(name) 'install hardware' *on* Day (date) 'May 1, 1991'. This Activity *lasted regular* Period(hours)8;
- **Overtime-Hours**: Worker-ID(number) 101 *has worked on* Task(name) 'unload material' *on* Day (date) 'May 1,1991'.

  This Activity *lasted overtime* Period(hours)2.

注意如何将 **Regular-Hours** 和 **Overtime-Hours** 两个事实类型分解成两个句子。一个句子将 Worker-ID、Task 和 Day 分成一组.另一个句子用名字 Activity 引用第一句子中的实体类型。Activity 叫做具体化的关系类型.用作用名 *lasted regular* 或 *lasted overtime* 将其赋给第二个句子中的 Period。在这种表示中.称 **Regular-Hours** 和 **Overtime-Hours** 为嵌套事实类型.如图 3(b)所示。图中.两个嵌套事实类型共享公用的具体化的关系类型 Activity。注意 Activity 如何在每个嵌套事实类型中起两种不同的作用.即 *lasted regular* 和 *lasted overtime*。图 3(b)中示出的嵌套事实类型表示法与图 3(a)中的展平表示法等效。但是.在这种情况下.具体化的表示法是一种较好的表示.尤其是在它被映象成关系模型时更是这样。嵌套机制及其优点的详细论述可见文[4]和[6]。

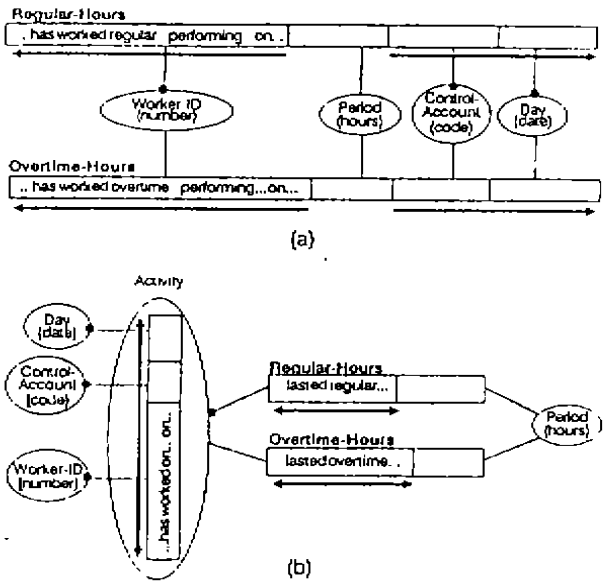


图 3 模式求精:(a)展平事实类型;(b)嵌套事实类型

4.2 规定约束条件

在完成了概念模式图之后.下一阶段就是表示约束条件以控制图上的基本事实类型和实体类型的行为。在将概念模式映象成关系数据模型时.这样的约束条件起一种关键作用。这里讨论三种约束条件:唯一约束条件、强制或可选作用约束条件以及子类约束条件。这些约束条件已用于建立建筑问题模型。其它约束条件也可在 NIAM 中表示.文[4]和[6]中有详细论述。

4.2.1 唯一约束条件 需要用唯一约束条件控制模式设计中的冗余。基本事实是不可重复的.这是一条原则。这表明事实的实例必须是唯一的.事实类型中的实体类型可能本身是唯一的.即它的实体实例是不重复的。这样就导致了在一列中不会有完全一样的值。另一方面.也存在这样的情况:虽然没有一种实体类型本身是唯一的.但是遍及事实类型的一些或全部实体类型的组合产生了唯一的事实实例.当一个实体类型是唯一的时候.称之为单码(single key).而当组合的实体类型是唯一的时候.它们产

生复合码(composite key)。

在NIAM中,唯一约束条件用双向箭头表示,画在码中所含作用的顶上或底下。图2(a)示出了一个二元事实类型 **WP-Level-Of-Detail**,带有 WBS-Node 实体类型,它是唯一的。注意在 WBS-Node 实体类型所起的作用 *has* 的顶上的双向箭头。图2(b)示出了一个三元事实类型 **Material-Budgets**. Control-Account 和 Material 实体类型形成了此事实类型的码。这两个实体类型不是邻近的。注意箭头是怎样横跨它们所起的作用的,以及如何在 Quantity 实体类型处断开。

唯一约束条件也适用于嵌套的事实类型。创建具体化的关系类型的一个要求是唯一约束条件箭头必须横跨具体化的关系中包含的所有作用,而具体化的关系类型在嵌套的事实类型中所起的作用必须是唯一的,如图3所示[4,6]。

4.2.2 *强制或可选作用约束条件* 一个输入表格上的信息可以是强制的或者是可选的。强制的意味着一个数据项必须要记录;可选的则意味着可以被忽略,在字段中留下一个空值。在NIAM中,为了形式说明信息的类型,相关的作用被标志为强制的或可选的。如果要求隶属于一个作用的实体类型的全体成员中每个都起这个作用,那么事实类型中的这个作用就是强制的;否则这个作用就是可选的。强制的作用在概念模式图中是这样表示的:在来自作用的弧线与实体类型接触点处加一黑点。如果一个实体类型只起一种作用,该作用就是强制的。图2(a)表示 WBS-Node 实体类型在二元事实类型中起强制作用,而 Level-Of-Detail 实体类型则起可选作用。在图2(a)中应注意的是:该事实只是一个子模式,但在考虑总体模式之前,子模式不应该加注起强制作用的黑点标记,原因是一个子模式实体类型似乎起强制作用,但是在子模式与其它的子模式合并形成总体模式时,该作用可能会变成可选的。

4.2.3 *子类约束条件* 当只由给定实体类型的全体的一个子集产生至少有一种作用时,采用子类约束条件。此作用可能已经被标志为可选的了,但是有时需要强调只有全体的一个子集才产生这一作用。为此,要为此子集全体创建一个新的实体类型。此实体类型变成了一个子类(subtype),而原来的实体类型则变成了一个超类(supertype)。子类/超类约束条件用从子类实体类型指向超类实体类型的箭头表示。在此体系中,子类除了起特别针对此子类的作用之外,还起隶属于超类的作用,此概念称为继承。这是因为超类仍包括子类的成员。

图4表示NIAM中的这种约束,其中,Control-Account 实体类型所起的作用只与此实体类型的全体有关。但是,Control-Account 实体类型的全体是 WBS-Node 的全体的一个子集(因为每个 Control account 事实上都是一个 WBS 元素),而与 Control account 相关的信息可能与 WBS 上的其它元素是不相关的。因此,为克服这个问题,创建了 Control-Account 实体类型以包括这个子集,并将子类约束施加给它,用从 Control-Account 实体类型指向 WBS-Node 实体类型的黑色箭头表示之。注意,由于这个子类约束,故 Control-Account 实体类型的全体继承了由 WBS-Node 实体类型所起的作用。

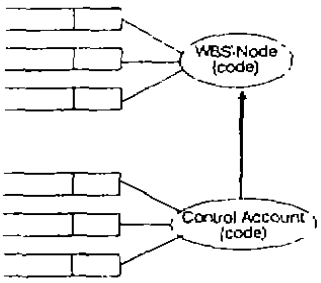


图4 子类约束

五、用NIAM的关系变换开发

一旦完成了概念模式图和加载了必要的约束条件,就将准备向任意的计算模型映射。现在,在计算数据模型中,关系、层次、网状和面向对象的模型都是适用的。本节把重点放在将NIAM模型映射成关系数据模型。NIAM模型同样也可以映射成其它适用的计算数据模型。

NIAM提供了一种关系变换算法,一般能保证开发出第五范式(5NF)关系模型。此算法叫做最优范式(ONF),由三个主要步骤组成,概述如下:

1. 对每个无单码的事实类型创建一个独立的关系。选择最短的码作为此关系的主码。

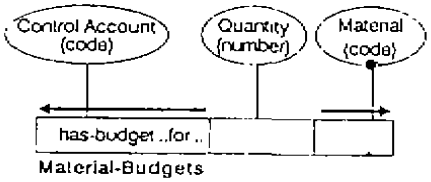
2. 将共享一个公共实体类型且在公共实体类型基础上有单码之事实类型分组到一个关系中去。根据此公共实体类型选择主码。

3. 对其余的每个事实类型创建一个独立的关系,并为每个关系选择主码。

此外,在遇到子类时,将它们映象成独立的关系,然后照字面表达相关属性之间的子类关系。而且,将嵌套事实类型中的具体化的关系类型看作是规范的实体类型。但是,在一个具体化的关系类型的

基础上创建的任何关系都必须具有与该具体化的关系类型中所包括的实体类型有关的属性。然后，一旦定义了关系，就在主码下面划线，将可选属性标为'OP'。

图 5 和图 6 给出了如何执行 ONF 的前两步的例子。在图 5 中，三元事实类型 Material-Budgets 符合 ONF 算法的第一步。这样，将其转换成了图中所示的关系。在图 6 中，Regular-Rate、Overtime-Rate、Craft-Code 和 Worker-ID 二元事实类型共享 Worker-ID 实体类型，并具有由 Worker-ID 起作用的单码，这样符合 ONF 算法的第二步。因此为它们创建了一个关系，如图中所示。



Control-Account-Budget-Materials

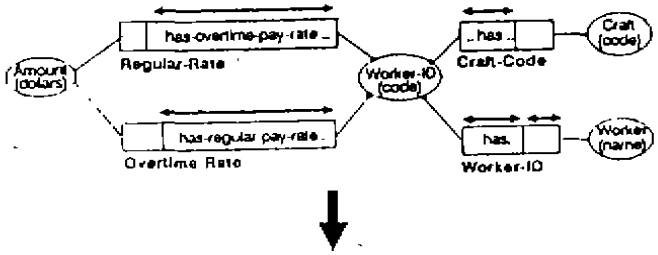
Control-Account-Code	MaterialCode	Quantity
134121	100	400
134122	110	1000
134123	120	10000

□ 码属性(复合码)

图 5 ONF 算法步骤 1

六、一个建筑管理的数据建模问题

在土木工程领域的施工过程中主要目标是按时完成计划和不超预算，且又满足规定的质量要求和其它规范。尤其是控制造价和工程进度需要采集、存贮和使用大量的数据，处理建筑管理系统中涌入的大量信息的一种有效的机制就是数据库管理系统。然而，在能够开发一个好的数据库设计之前，要对问题建立好模型。为此，我们采用了 NIAM。为确定控制造价和工程进度中所包含的数据项，我们实际利用了建筑数据采集表格和报告。此外，我们还利用了一个造价和工程进度控制模型来描述控制系统在数据管理和处理方法及机制方面的行为。利用造价和工程进度控



Worker-Information

WorkerID	WorkerName	CraftCode	RegularRate	OvertimeRate
101	D Callaway	10	15	20
102	S Hubert	20	20	30

□ 码属性(单码)

图 6 ONF 算法步骤 2

制模型所提供的这些表格和知识，能够以基本事实的形式表示对所要求的信息管理系统，形式化地开发出一套规格说明。

下一步，我们利用 NIAM 的图形语言，用图形表示出基本事实来。然后，把唯一的和强制的作用约束加到完整的 NIAM CSD 上。最后，应用 ONF 算法把完整的 NIAM CSD 转换成为等效的关系数据库模式。

关系转换产生了三组关系。第一组叫做静态数据库，包括那些在处理施工过程中，正常情况下不会变化的数据关系(例如预算和计划)。第二组叫做动态数据库，包括处理施工期间每日采集的数据和资料的关系。第三组叫做历史数据库，包括处理为将来应用而创建历史数据记录的关系。

七、结论

尼杰森法是一种简单且用自然语言建立数据和语义模型的方法，所以在将来的工程数据库设计中将会起到关键的作用。NIAM 的图形方法为其自己产生了由工程师们用来开发的工程图。此外，由于它是建立模型方面的系统化方法，使 NIAM 建模方法能容易实现自动化设计。事实上，帮助把 NIAM CSD 转换成为关系数据模型的自动化工具已经存在了。再者，NIAM 确保了数据库模式设计和生成的一致性，同时也确保了数据库的高度集成性(产生 5NF 关系模型)。这样，在各行各业的工程数据库的开发和实现方面，以及在数据交换方面，对标准化工作作出了卓有成效的贡献。

李明琪摘译自《Advances in Engineering Software》Vol. 14, No. 1, 1992