一种异构数据源模型转换和模式集成框架

王 博 郭 波

(国防科学技术大学信息系统与管理学院 长沙 410073)

摘 要 异构数据源集成系统的主要任务是屏蔽数据源数据模型的异构性,提供对数据的统一访问。公共数据模型、模型变换和中间数据模型被广泛用来解决该问题。由于数据集成工作的工作量和任务对象日益复杂、规模日趋庞大,仅采用公共数据模型不能满足现有集成要求。异构数据源数据模型内在的关联性虽然使得面向语义集成变得更加复杂,但更利于模型转换方法在数据集成中的应用。模型转换是模式集成的基础,本文给出了异构数据源模型的形式化描述方法和基础模型转换操作形式化框架。该框架能够保证模型、实例和约束三者的相互独立,适用于大多数基础数据模型及其之间的转换和集成应用。通过该框架可简化和规范异构数据源数据模型转换和模式集成过程。

关键词 异构数据模型,模型转换,模式集成

A Framework for Model Transformation and Schema Integration of Heterogeneous Data Source

WANG Bo GUO Bo

(College of Information System and Management, National University of Defense Technology, Changsha 410073)

Abstract The main task of heterogeneous data source integration is to providing a uniform way for accessing data source without considering the heterogeneous characteristic of data source. Common data model, model transformation and middle data model are widely used to solving this problem. As the work of data integration becomes more and more complicated, common data model can not satisfy the need of data integration now. Although works of semantic integration becomes more complicated for the inner relationship between data sources, but the transformation of model becomes much convenient, model transformation is the foundation of schema integration, in this paper we give a formal method for the description of heterogeneous data source and the formal framework for the primitive transformation between basic data model, this framework can maintain the independency among model, instance and constraints, which is more suitable for transformations between data models and the application of integration, the process of the transformation between heterogeneous data sources and schema integration will be simplified and formalize by using of this framework.

Keywords Heterogeneous data models, Model transformation, Schema integration

1 引言

异构数据源集成是解决集中访问存储在不同数据源的数据的问题。异构数据源通常指地理位置分散、状态各异、结构不同的数据源,数据源异构主要体现在两个方面:物理异构和逻辑异构。物理异构包括如下几个方面:硬件和操作系统异构、数据管理软件异构(例如 DBMS,MIS 系统异构)等。逻辑异构主要包括:数据模型、模式和语义异构、中间件异构、用户接口异构、业务逻辑和完整性约束异构等[1],其中数据模型、模式的异构是异构数据源集成技术研究的主要问题。

实现异构数据源之间的数据交换和共享、异构数据源的集中访问,首先需要正确理解异构数据模型、数据模式,构建基于公共数据模型(Common Data Model,简称为 CDM)的集成模式[2]。由于不同数据模式设计独立,在集成时存在模式类型、模型结构和模式语义的冲突,因此用 CDM 来解决模式集成问题在模型差异适中、种类不多的情况下可以有效解决集成问题,但随着数据模型的不断丰富、种类不断增加、设计独立性越来越明显,公共数据模型的构建必然是十分繁琐的工作。为消除模式之间类型和结构的差异,需要一种能够实现各种类型和各种结构数据模型互相转换和映射的机制和方法来实现模式集成。目前,数据库设计和信息系统设计常用的数据模型包括 ER 模型、关系模型、UML 模型、XML 数据模型以及 Web 信息模型,还包括一些特殊类型的数据模型,例如半结构化、非结构化数据模型以及多媒体数据模型等。就某一数据模型本身来说,例如 UML 模型,不同设计人员对相同的信息实体的设计方式不尽相同,例如具有父子类关系的实体在某些情况下也可以设计为实体属性关系(见图 2),因此数据模式设计结构的差异、数据模型本身的差异以及数据模式的语义差别使模式集成问题变得十分复杂,有必要提供一个统一的异构数据模型描述方法以及异构数据模型相互转化的机制来实现异构数据模型的相互理解和集成。本文首先分析了常用公共数据模型的共同特点,给出了一种用于描述基础数据模型转换的形式化框架。该框架无需考虑数据模型本身的不同,只需要注意数据模式的设计差异,从而实现异构数据源数据模式之间的相互映射和转换,最终实现异构数据源数据模式集成。

2 数据模型分析与形式化建模

数据模型分为静态模型和动态模型两种[3],目前数据集成面对的数据大多是在已有的信息系统中,其模型大多是静态模型。而动态数据模型主要来自系统的业务建模过程,例如 UML 模型中的顺序图、交互图等,这里暂不考虑数据模型

王 博 博士研究生;郭 波 教授,博士生导师。

的动态特性，只考虑数据实体本身。Peter 等分别分析了 ER 数据模型、UML 和 WWW 等数据模型的形式化描述方法[4]，但未能进行统一，不同的模型需要中间模型—超图[5]来转换。常用数据模式主要是对信息实体进行建模，因此主要考虑实体本身以及相互关系，例如实体的名称、属性、实体间的层次关系和关联关系等。在构建异构数据模型转换和模式集成形式化框架之前首先定义两个不相交集合：

1. 数据值集合定义为：$Val(values)$，表示某类数据模型代表的数据源中的真实内容。

2. 实体、属性和关系名称集合定义为：$Name(names\ of\ entity\ types, attributes\ and\ relationships)$。常量 $Null \in Names$，表示空名称。

令 $Sequence(Vals)$ 表示值的有限序列，令 $Cards$ 表示实体关系的基数约束，本文简称势约束。对 $c \in Cards$，c 具有 $l:u$ 形式，其中 l 是势约束下限，u 是势约束上限。以实体学生和姓名之间关系为例，$c_{sd}=1:1$ 为学生与姓名关系中学生对姓名的势约束，表示一个学生最多有一个姓名，最少也有一个姓名。

定义在一般数据实例上的普通函数为 $function(args)$，令 $Range(function)$ 表示函数 $function(args)$ 值域。

定义 1 Data Model Schema S 为一个四元组 $\langle Ents, Incs, Atts, Assocs\rangle$，其中：

$Ents \subseteq Names$ 是实体类型的名称集合，该实体可以是 ER 模型中的 Entity，也可以是 UML 中 class 实体，或是 XML 文档中的一个 Node 等。

$Incs \subseteq (Ents \times Ents)$，有序值对 $\langle e_1, e_2\rangle \in Incs$ 的含义为 e_1 是 e_2 的子类型，$Incs$ 是无环的，即不存在某个实体的子孙类是其本身的父类。

$Atts \subseteq Names$ 是属性名称集合，例如关系数据库中表的属性列、UML 模型中类的属性以及 XML 模型中节点的属性等。

$Assocs \subseteq (Names \times Names \times Names \times Cards \times Cards)$ 为关系集合。

对实体 $e_1, e_2 \in Ents$，存在元组 $\langle rel\_name, e_1, e_2, c_1, c_2\rangle \in Assocs$，则称实体 e_1, e_2 存在关系。c_1, c_2 为势约束，c_1 表示对于每个实体 e_1，在关系 $\langle rel\_name, e_1, e_2, c_1, c_2\rangle$ 中所允许对应的实体 e_2 个数的上限和下限，即 $c_1=l_1:u_1$ 中的 l_1 和 u_1，c_2 的含义同 c_1。若 e_1, e_2 之间关系唯一，rel_name 可为 $Null$。例如 XML DTD 中定义了 XML 文档的结构格式和内容格式，其中 ELEMENT 节点的属性取值即规定了 XML 节点之间的势约束关系[6]。

同理，属性和实体同样存在关系，用元组 $\langle Null, e, a, c_1, c_2\rangle \in Assocs$ 表示。由于属性和实体之间有唯一关系，因此其名称为 $Null$，其中 a 为属性实体。

定义 2 给定模式 $Schema\ S=\langle Ents, Incs, Atts, Assocs\rangle$，令 $Schemes=\{\langle n_0, n_1, n_2\rangle \mid \langle n_0, n_1, n_2, c_1, c_2\rangle \in Assocs\}$，$Schemes$ 可理解为实体关系对，通过约束限制，$Schemes$ 对应了一系列关系实例。

模式 S 的实例（Instance，记为 I）定义为 $I \subseteq P(Seq(Vals))$，I 是一个值序列集合，其中 $Vals$ 可理解为数据模型对应的数据实例，I 为有效值序列集合。

定义函数：$Ext_{S,I}: Ents \cup Atts \cup Schems \to P(Seq(Vals))$，其中 $P(args)$ 可理解为具有强度的值序列，函数 $Ext_{S,I}$ 理解为实体、属性和关系上的外延。显然，某个模式 S 上的外延必然是模式 S 实例的子集。以图 1 为例，教师实体的外延是集合 $\{ZhangSan, LiSi\}$，课程外延为 $\{Math, English, Music\}$，中间部分为关系外延。图 1 下半部分所有子集构成整个数据模式的实例。这里称 $Ext_{S,I}$ 为一个从 S 到 I 的映射函数。

函数 $Ext_{S,I}$ 满足：

①每个 $Range(Ext_{S,I})$ 中的子集都可以通过 I 集合上的查询语言 L 的表达式来得到；

②相反，每个 I 中的集合都可以通过 $Range(Ext_{S,I})$ 中集合上的查询语言 L 的表达式来得到；

③$\forall \langle e_1, e_2\rangle \in Incs, Ext_{S,I}(e_1) \subseteq Ext_{S,I}(e_2)$。

定理 1 对于每一个 $\langle n_0, n_1, n_2\rangle \in Schemes$，$Ext_{S,I}(\langle n_0, n_1, n_2\rangle)$ 表示 $\langle n_0, n_1, n_2\rangle \in Schemes$ 外延值序列，则对于每个值序列 $s \in Ext_{S,I}(\langle n_0, n_1, n_2\rangle)$ 包含两个可能的有交叉项的子序列 s_1, s_2，其中 $s_1 \in Ext_{S,L}(n_1)$，$s_2 \in Ext_{S,I}(n_2)$；用 $n_1(s), n_2(s)$ 分别表示 s 的子序列，满足如下域和势约束条件：

如果 $\langle n_o, n_1, n_2\rangle$ 的势约束为 $\langle l_1 : u_1 : l_2 : u_2\rangle$ 则

$$\forall s_1 \in Ext_{S,I}(n_1)\, l_1 \leqslant |\{n_2(s) \mid s \in Ext_{S,I}(\langle n_0, n_1, n_2\rangle) \wedge n_1(s)=s_1\}| \leqslant u_1$$

且 $$\forall s_2 \in Ext_{S,I}(n_2)\, l_2 \leqslant |\{n_1(s) \mid s \in Ext_{S,I}(\langle n_0, n_1, n_2\rangle) \wedge n_2(s)=s_2\}| \leqslant u_2$$

证明：l_1 表示每个 n_1 实例在关系 $\langle n_0, n_1, n_2\rangle$ 中对应 n_2 的实例的最大最小基数，即个数。$\langle n_0, n_1, n_2\rangle$ 实例满足势约束，即 $Ext_{S,I}(\langle n_0, n_1, n_2\rangle)$ 实例中值对 $\langle n_1, n_2\rangle$ 满足势约束，因此 $Ext_{S,I}(\langle n_0, n_1, n_2\rangle)$ 中对应 n_2 的 n_1 也满足势约束。定理结论第一部分成立，同理第二部分亦成立。证毕。

定义 3 对于复杂数据模型需要建立更为灵活和全面的约束模型。将约束模型定义如下：

$$Cons \subseteq \{c\{s_1, \cdots, s_n\} \mid c \in Funce \wedge s_1 \in Schemes \wedge, \cdots, \wedge s_n \in Schemes\}$$

其中 $Val(Cons)=\{true, false\}$，约束的取值为 BOOL 型，表示约束是否满足。约束模型是保证数据实体完整性要求的必要条件，也是不同模型间的相互转换是否成功的约束条件，其中 Funcs 是具体约束内容。定义 1 中势约束 Cards 是 Cons 中常见的一种。目前在各种数据模型中常用的约束模型包括包含约束、联合约束、唯一性约束等[7]。由于后面定义的模型基础转换对约束独立，本文仅就势约束进行讨论。

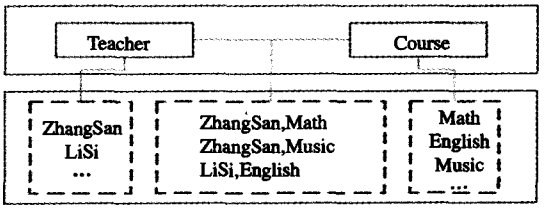


图 1 模式和外延

定义 4 数据源模型 M 定义为一个三元组：$\langle S, I, Ext_{I,S}\rangle$，其中 $S \in Schema$。定义模式等价（Schema Equivalence）如下：

定义 5 用 $Ins(S)$ 表示模式 S 的实例集合，给出模式蕴含和等价定义：

$Ins(S') \subseteq Ins(S) \Rightarrow$ 模式 S 蕴含模式 S'，即 S 的实例包含 S' 的实例；

$Ins(S')=Ins(S) \Rightarrow$ 模式 S 等价于模式 S'，即 S 的实例和 S' 的实例相同。

定义 6 $Ins(S,f)$表示满足给定条件 f 的模式 S 的实例集合，给出模式条件蕴含和条件等价定义如下：

$Ins(S',f)\subseteq Ins(S,f)\Rightarrow$模式 S 条件蕴含(c-subsumes)模式 S'；

$Ins(S',f)=Ins(S,f)\Rightarrow$模式 S 条件等价(c-equivalent)模式 S'。

上面给出了常用基础数据源模型的统一的形式化定义。部分数据源模型有更广泛的概念外延，例如实体关系约束还包括递归、可选等关系[7]，可以通过修改势约束的表示方法来表示。此外，UML 模型中还包括共同属性和私有属性[8]，可以通过向实体名称或属性名称前面加前缀的方式来予以区别。额外的数据源模型规范可能还需要进一步扩充上述数据模型的形式化描述内容，而核心部分不改变。模式等价是正确实现模式转换的重要保证条件，模式转换必须在等价条件下进行，按照模式等价定义可理解为真实存储在异构数据源中的数据必须在上层的数据模式转换过程中保证数据完整性。

3 数据源模型转换

数据源模型之间通过相互转换达到共同理解，通过向统一的模式转换可以进一步实现数据模式集成。本节给出基础数据源模型的一些基础变换操作，通过这些基础操作和操作组合可以在不同数据源模型之间实施变换。转换过程可以形式化描述为 $Trans(args,M)=M'$，其中 M 为被转换数据源模型，$args$ 为一些必要的转换参数，$Trans$ 代表所有基础转换操作。因此转换过程可以理解为以一个数据源模型为输入，通过适当参数在基础变换下输出需要的等价的数据模型。

模式变换的目的是实现异构数据模型之间数据的共享，数据模式变换过程必须保证数据实例完整性。如图 2 左边模型中属性 B、C 经模型变换后成为图 2 右侧类 A 的子类，但原有模型中类 A 对应的实例同右侧父子类形成的数据模型对应的数据实例相同。这类施加在模型上的变换称为等价基础变换。等价转换的结果是实例部分不变化，而模式和模式与实例的映射关系 $Ext_{S,I}$ 发生变化。

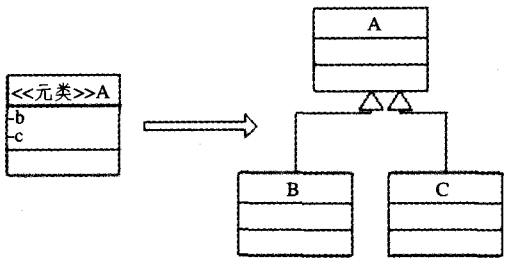


图 2 等价基础变换

可证明下面列出的基础转换(primitive transformation)都是等价转换。每个变换具有一定的附加条件，条件不满足则转换结果为 ϕ，转换过程表示为：$Trans(args,f,M)=M'$，ϕ 的转换结果也为 ϕ，表示为 $Trans(args,f,\phi)=\phi$。

定义 7 模型基础转换

(1)$rename_x(Names\times Names)$，该转换对模式 S 中的一些项进行重命名，其中 X 在 ER 模型中代表属性、实体和关系等名称，在 UML 模型中代表类名、属性名、关联关系以及其他关系名，在 XML 模型中表示 XML Node、属性以及 DTD 中 ELEMENT 约束等，其他模型中类似定义。$rename_x$ 参数为$\langle fromName,toName\rangle$、命名对象和被转换模式，其结果是所有在被转换模型中出现 $fromName$ 的被替换成 $toName$，转换执行条件 f 为 $toName$ 未被使用。

(2)$expand(Names\times Names\times Names\times Cards\times Cards)$，用来在实体之间进行约束扩展，参数为被转换模型和关系-势约束元组：$\langle n_0,n_1,n_2,c_1,c_2\rangle$，其结果是将原有势约束替换成现有的更松弛的势约束。其条件是原有约束元组$\langle n_0,n_1,n_2,oc_1,oc_2\rangle$中的势约束 oc_1,oc_2 必须存在，且满足 $oc_1\subseteq c_1$，$oc_2\subseteq c_2$，其中“$\subseteq$”定义为一种新含意的包含关系，例如势约束 $c_2=l_2:u$ 包含 $c_1=l_1:u$，则有 $c_1\subseteq c\Rightarrow l_1\geqslant l_2\wedge u_1\leqslant u_2$。

(3)$contract(Names\times Names\times Names\times Cards\times Cards)$，用来在实体之间进行约束收缩，参数为被转换模型和一个关系-势约束元组$\langle n_0,n_1,n_2,c_1,c_2\rangle$，转换结果为将原有势约束替换成现有的更严格的势约束。其条件是原有约束元组$\langle n_0,n_1,n_2,oc_1,oc_2\rangle$中 oc_1,oc_2 必须存在，且满足 $oc_1\supseteq c_1$，$oc_2\supseteq c_2$，其中“$\subseteq$”理解同上。

(4)$add_E(Names\times Queries)$，用来增加一个实体。参数$\langle e,q\rangle$中 e 是增加实体名称，查询 $q\in Queries$ 定义了所增加实体的外延。其执行条件是所增加实体在原有模式中不存在。

(5)$del_E(Names)$，用来删除一个实体，条件是所删除实体 e 不包含属性，并不属于任何关系。同时，为了满足定义 2 函数 $Ext_{S,I}$ 的条件 2，在删除实体之后所得到的新模式 S' 中将 $Ext_{S,I}(e)$ 定义为 ϕ。

(6)$add_R(Names\times Names\times Names\times Cards\times Cards\times Queries)$，用来增加一个新的关系，参数为被转换模型和关系元组$\langle r,e_1,e_2,c_1,c_2\rangle$，目的是将$\langle r,e_1,e_2,c_1,c_2\rangle$插入到数据模型对应的模式的关系集合 $Assocs$ 中。查询 q 定义了关系外延，执行条件是实体 e_1,e_2 必须存在，r 未被使用且外延满足域和势约束。

(7)$del_R(Names\times Names\times Names)$，用来删除一个关系，参数是被转换模型和关系元组$\langle r,e_1,e_2\rangle$，条件是关系 r 在原有模式中存在，且删除后为了满足定义 2 函数 $Ext_{S,I}$ 的条件 2，在删除 r 之后所得到的新模式 S' 中将 $Ext_{S',I}(\langle r,e_1,e_2\rangle)$定义为 ϕ。

(8)$add_A(Names\times Names\times Cards\times Cards\times Queries)$为一个实体类型增加一个属性类型，该属性在源模式 S 中可以存在，也可以是新的属性。其参数为一个被转换模型和属性-实体元组$\langle e,a,c_1,c_2,q_{att},q_{assoc}\rangle$，操作结果是增加属性 a，同时向被转换模式 S 增加一个新的关系$\langle Null,e,a,c_1,c_2\rangle$，查询 q_{att},q_{assoc} 分别给定义了增加的属性和新关系的外延。执行条件是实体 e 存在且没有属性 a，同时关系$\langle Null,e,a,c_1,c_2\rangle$外延要满足域约束和势约束。

(9)$del_A(Names\times Names)$，从一个给定实体中删除一个属性，参数为被转换模型 S 和元组$\langle e,a\rangle$，其中 a 为被删除属性，e 为包含该属性的实体类型。其操作结果是删除属性 a，同时从 S 的关系集合中删除对应关系，条件是关系$\langle Null,e,a\rangle$存在，同时为了满足定义 2 函数 $Ext_{S,I}$ 的条件 2，删除属性 a 后得到的新模式 S' 中将 $Ext_{S',I}(\langle Null,e,a\rangle)$定义为 ϕ。

(10)$add_I(Names\times Names)$，增加一个子类型关系，参数为被转换模型 S 和元组$\langle e_1,e_2\rangle$，目的是建立实体 e_1 和 e_2 之间的父子类关系，即通过转换在新的模型 S' 中 e_1 成为 e_2 的子类型。条件是实体 e_1 和 e_2 存在，且关系$\langle e_1,e_2\rangle$不在新模式 S' 的 $Incs'$ 的传递闭包中，表示为$\langle e_1,e_2\rangle\notin transitiveClosure(Incs')$，即 $Incs'$ 不存在如下父子关系序列$\langle e_1,e_{n_1}\rangle,\cdots,$

$\langle e_{n_m}, e_2\rangle$。满足 $Ext_{S,I}(e_1)\subseteq Ext_{S,I}(e_2)$。

(11)$del_I(Names\times Names)$，删除父子关系，参数为元组 $\langle e_1, e_2\rangle$ 和被转换模型，条件是被转换模式父子关系存在。

上述转换过程在没有条件约束的情况下可以进行任何异构数据模型的转换，在条件约束的情况下得到的模型是合理的模型。具有相同模式的各种模型，在经过相同转换后，得到的模型的模式依然相同。连续施加在被变换模型的基础变换序列成为组合变换。通过上述基础变换可以实现基本的异构数据模型之间的转换，例如图 2 的转换过程可以表示如下：

$add_E\langle B,\{e(s)\mid s\in\langle Null,e,a\rangle\wedge a(s)=b\}\rangle$，

$add_E\langle C,\{e(s)\mid s\in\langle Null,e,a\rangle\wedge a(s)=c\}\rangle$，

$add_I\langle B,A\rangle; add_I\langle C,A\rangle$，

$del_A\langle A,b\rangle; del_A\langle A,c\rangle$。

其中实体 B,C 是新增实体，其变换前为实体 A 的属性 b,c。其他变换包括实体/属性关系到实体关系等价变换、实体和关系等价变换以及实体融合集成等，也可通过集成变换组合实现，这里不一一介绍。

定义 8 t 变换对模式 S 的作用为 $Schema(t,S)$；

模式算子定义为 $Schemaof(M)\in Schema, M\in Models$。

$\forall m_1, m_2, \cdots, m_n\in Models$，令 $Trans(m)=m'$

若有 $Schemaof(m_1)=Schemaof(m_2)=\cdots=Schemaof(m_n)=S$，则有

$$Schemaof(Trans_i(m_1))=Schemaof(Trans_i(m_2))=\cdots=Schemaof(Trans_i(m_n))=Schema(t,S)$$

上述过程可理解为具有相同模式的模型无论是 ER、UML 或是 XML，经过相同的基础模式变换后得到的新模型的模式依然相同。

定义 9 基础转换 t 是模式依赖的(schema-dependent)$\Rightarrow\forall S, Schema(t,S)\neq\phi$。$\gamma$ 否则为实例依赖(*instance-dependent*)。

由 t 是模式依赖的，得出 $Schema(t,S)\Rightarrow S$。

证明：令 $t=t_1t_1^{-1}$，显然 $Schema(t,S)=Schema(t_1t_1^{-1},S)=S$，则 $Schema(t,S)\Rightarrow S$

定理 2 如果模式 S 经过一系列模式独立(*s-d*)的基础变换得到模式 S'，则 S' 经过相反变化过程可以得到 S，同时 S 和 S' 是等价的。

证明：首先有任意基础变换 t 存在可逆变换 t^{-1}，令变换序列为 $t_1, t_2, \cdots, t_n$，则组合变换 $t=t_1t_2\cdots t_n$ 的逆变换为 $t^{-1}=t_n^{-1}t_{n-1}^{-1}\cdots t_1^{-1}$。

由于 $t_1, t_2, \cdots, t_n$ 都是模式依赖的，则可递归描述组合变换如下：

$$Schema(t,S)=Schema(t_1t_2,\cdots,t_n,S)=Schema(t_n,Schema(t_1t_2,\cdots,t_{n-1},S))$$

有每个基础变换可逆，则有

$$Schema(t^{-1},S')=Schema(t_n^{-1}t_{n-1}^{-1},\cdots,t_1^{-1},S')=Schema(t_1^{-1},Schema(t_n^{-1}t_{n-1}^{-1},\cdots,t_2^{-1},S'))=,\cdots,=S$$

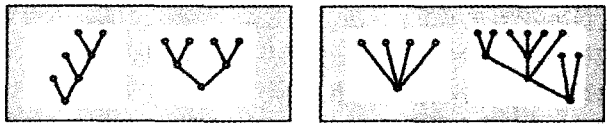
结论一证毕。由于基础变换过程中模式外延保持不变，因此 $Ins(S')=Ins(S)$，则模式 S 和 S' 是等价的，定理证毕。同理易证如下定理 3。

如果 t 是实例依赖的，模式 S 具有约束条件 f，则 $Schema(t,S)$ 条件蕴含 S，表示为 $Schema(t,S)\xrightarrow{t}S$。

定理 3 如果模式 S 经过一系列的 i-d 和 s-d 变换得到 S'，并考虑约束 f，则模式 S 条件等价(c-equivalent)模式 S'。

条件等价和等价关系均能够保证转换前后模式对应的实例保持不变，使得在进行数据集成工作的时候数据不至缺失，数据完整性约束能够得到保证。前面已经提到基础转换的组合转换依然能够保持等价性，因此不同数据模型可以在丰富本文构建的基础转换框架基础上实现交换和共享。

模式集成过程包含具有层次关系的两类[9]：第一类是二元集成，第二类是直接的多元集成。二元集成和多元集成如图 3 所示。



二元集成过程　　多元集成过程

图 3　集成过程

上述集成过程采用自底向上的方式。如前文所述包括两两集成和多模式集成两类，模式之间的集成通过彼此转换或者转换成统一的中间模型来逐层实现，如图 3。若不同异构数据源模型实现相互类型转换比较困难，则可以通过中间节点实现模型之间的互联。例如 XML 数据模型中，某些节点可能对应 UML 的对象模型，可以通过形成中间模型来实现二者的松散集成。

小结　异构数据模式集成首先需要实现模式之间的相互理解，通过模型转换以及中间模型可以消除一些基本的结构和模型类型差别。常用的方法是通过构建公共数据模型(CDM)来实现不同数据模型向公共数据模型的映射，为进一步模式集成提供基础，而公共数据模型表述能力有限，不能做大规模、复杂模式的集成工作，因此需要在预集成过程中做适当的模型转换工作，为后续集成提供基础。通过中间模型既可以实现多个高层概念模型向中间模型的转换，也可以使不同模型在中间概念层形成统一理解，例如 McBrien 提出的基于 HDM(超图模型)的中间数据模型[2, 10]。本文给出了基本的异构数据模型的形式化表示方法，给出了数据模型之间转换的基本操作的形式化描述。数据模型主要包括数据表述部分、约束部分和关联部分，在本文给出的形式化框架下相互独立，因此可以适应大多数据模型之间的转换，也可以进一步针对某类型数据模型进行建模转换工作。

参 考 文 献

1 Ziegler P, Dittric K R. Three Decades of Data Integration All Problems Solved?. In: 18th IFIP, Paris, France, 2002

2 Poulovassilis A, McBrien P. A formalisation of semantic schema integration: [Technical Report]. London: King's Colloge, 1996

3 Lawrence R, Barker K. Unity-A Database Integration Tool: [Technical Report]. Department of Computer Science, University of Manitoba, 2000

4 McBrien P, Poulovassilis A. A Uniform Approach to Inter-Model Transformations. In: CAiSE'99, Heidelberg, Germany, 1999

5 Boyd M, McBrien P. Comparing and Transforming Between Data Models via an Intermediate Hypergraph Data Model. Computer Science, 2005(3730): 69～109

6 Williamson H. XML: The Complete Reference. 北京：机械工业出版社，2002

7 Stephens R K, Plew R R. Database Design. 北京：机械工业出版社，2001

8 Eriksson H-E, Penker M. Business Modeling with UML. 北京：机械工业出版社，2004

9 Batini C, Lenzerini M, Navathe S. A Comparative Analysis of Methodologies for Database Schema Integration. ACM Computing Surveys, 1986, 18(6): 323～364

10 McBrien P, Poulovassilis A. A General Formal Framework for Schema Transformation. Data and Knowledge Engineering, 1998, 28(1): 47～71