Hans Journal of Data Mining 数据挖掘, 2021, 11(2), 84-99
Published Online April 2021 in Hans. http://www.hanspub.org/journal/hjdm
https://doi.org/10.12677/hjdm.2021.112009

Hans汉斯

带有范例元组的交互式数据转换映射方法研究

李　静，李　贵，李征宇，韩子扬，曹科研

沈阳建筑大学，信息与控制工程学院，辽宁 沈阳
Email: 2868896342@qq.com, ligui21c@sina.com

收稿日期：2021年3月19日；录用日期：2021年4月20日；发布日期：2021年4月27日

摘　要

模式映射是Web异构大数据集成的重要研究内容之一，通常包含实例层和模式层两方面的研究，本文的研究重点主要集中在模式层。要想在短时间内完全掌握这门技术并且加以运用，这对于那些不熟悉模式转换所涉及的转换语义和语言的非专家用户来说几乎是不可能的。因此，本文在已有的关于数据转换研究成果的基础之上提出了一个适用于非专家用户的交互式模式映射设计框架系统。首先，对由非专家用户提供的不完整的表达性较差的数据转换范例元组进行预处理。然后，再通过简单的用户交互递归地对初始范例元组的有效性进行布尔查询从而得到最终映射规则。其次，本文提出了两种探索所有数据转换映射空间的策略以满足任意用户范例元组。在探索过程中系统会根据与用户交互的结果来保留最适合用户需求的规则，并动态地剪枝搜索空间从而减少与用户交互的次数，本文实验采用来自中国土地市场网的数据集成转换来验证本文方法的有效性。

关键词

Web大数据，数据集成，数据转换，模式映射，布尔查询

Research on Interactive Data Conversion Mapping Method with Example Tuples

Jing Li, Gui Li, Zhengyu Li, Ziyang Han, Keyan Cao

School of Information & Control Engineering, Shenyang Jianzhu University, Shenyang Liaoning
Email: 2868896342@qq.com, ligui21c@sina.com

Received: Mar. 19th, 2021; accepted: Apr. 20th, 2021; published: Apr. 27th, 2021

Abstract

Schema mapping is one of the important research contents of heterogeneous big data integration

文章引用: 李静, 李贵, 李征宇, 韩子扬, 曹科研. 带有范例元组的交互式数据转换映射方法研究[J]. 数据挖掘, 2021, 11(2): 84-99. DOI: 10.12677/hjdm.2021.112009

on Web, which usually includes two aspects: instance layer and schema layer. The focus of this paper is mainly on schema layer. It is almost impossible for non-expert users who are not familiar with the semantics and language involved in schema transformation to master this technology and apply it in a short time. Therefore, based on the existing research results on data conversion, this paper proposes an interactive schema mapping design framework system for non-expert users. Firstly, the incomplete data transformation paradigm tuples with poor expressiveness provided by non-expert users are preprocessed. Then, the validity of the initial example tuple is recursively queried by simple user interaction, and the final mapping rules are obtained. Secondly, this paper proposes two strategies to explore the mapping space of all data transformations to satisfy any user paradigm tuple. In the process of exploration, the system will keep the rules that are most suitable for users' needs according to the results of interaction with users, and prune the search space dynamically to reduce the number of interactions with users. In this experiment, the data integration transformation from China Land Market Network is used to verify the effectiveness of this method.

Keywords

Web Big Data, Data Integration, Data Exchange, Schema Mapping, Boolean Query

1. 引言

随着 Web 数据源的不断丰富和互联网技术的不断提高，Web 数据集成技术得到了前所未有的发展。这使得人们获得的 Web 信息资源大量增加，方便了人们的工作和生活。于此同时，集成来自不同数据源的数据是开发 Web 大数据应用的重要任务，现行的数据管理应用常常从多个异构 Web 数据源集成数据，以构建或补充特定领域或多领域的知识库。另外当前的数据集成系统在进行数据交换时至少要解决两方面的问题：一方面是对于映射实例层的研究，主要研究内容是如何寻找无冗余的数据交换解。另一方面是对数据集成过程中模式层的研究，主要涉及映射规则的生成、选择及优化[1]。

首先，数据交换问题(data exchange problem) [2]最初是由 Fagin、Kolaitis 等人提出的并且现如今大部分关于数据交换的研究皆是以此为基础研究而来，即数据交换是在源模式下获取结构化数据，并创建目标模式的实例以尽可能准确地反映源数据的问题。其次他们还给出了一个代数规范，该规范在数据交换问题的所有解决方案中选择一类我们称之为通用的特殊的解决方案。且通用解决方案能够很好的解释源和目标数据。它代表了所有可能的解决方案的整个空间。

其次，针对数据集成中的模式映射的问题。如文献[3]模式映射可分为模式匹配和映射生成两步。模式匹配的目标为发现模式元素间的对应关系，映射生成的目标为产生符合模式语义约束的元素集与元素集之间具有蕴含或相等关系的逻辑表达式。其关键在于如何从大量的潜在的映射规则集里面选择一些能最好的解释源数据库和目标数据库的映射规则，并且要尽可能地减少目标数据库里面的数据冗余度。如文献[4]提出了一种基于概率软逻辑(PSL)的方法来解决模式映射问题。即在给定元数据约束和数据实例的情况下，从潜在映射空间中选择一组最佳映射。但是该方法通常针对小型映射场景有比较好的性能表现，然而针对大型的映射场景就显得有些乏力，并且该文献里面并未过多地详述初始映射规则集的由来。

本文的研究重点主要集中在初始映射规则集上，实际上模式映射就是一系列从源表到目标表的对应关系的声明性规范，通常采用一阶逻辑。它们构成了关键的数据可编程原语，也使数据库用户能够在大型的共享数据库上使用编程工具来设计映射规则。与此同时，在大多数研究异构或者异源的数据集成问题中，数据库管理专家不可避免的会在考虑这些数据集的不同的映射规则集的前提下对这些不同来源且异构的数据集进行设计、实现以及运用来达到数据集成的目的。此外映射规则还通常在企业 IT 和一些其他的大数据领域得到广泛的运用。例如在企业 IT 中，数据分析师(也叫工程映射的开发人员)需要预先定义要访问或集成的不同来源的数据的模式之间的工程映射(数据转换模式) [5]。通过这种预定义，也就意味着对于每个不同的应用来说，这些映射规则都是被精确定义和测试的。目前已经有许多文献对工程映射相关问题进行了研究，首先能想到的就是使用可视化的图形界面来让使用者设计映射规则。就如文献[6] [7]中提到的一样，该文献提出了几个基于图形界面的映射设计器，这些映射设计工具能帮助数据分析师以通俗易懂的符号设计不同模式的数据库表之间的映射规则。然而以上提到的解决方法的一个主要的缺点就是，依靠图形语言生成的可编程语言或查询语言中的映射规则都依赖于特定的映射生成工具。因此，同样的图形语言规范可能会被两个或多个不同的映射生成工具转换为不同的、无法比较的声明性映射规范，从而使得两边数据库不一致。为了解决这种数据不匹配问题，在文献[5]中提出了模型管理操作符，以提供一种通用的映射设计器，该设计器可以适用于多种数据可编程性的工具，然而该模型管理系统仍然仅仅适用与专家用户。其次就是依靠具有代表性的数据实例[8] [9] [10]来生成用户心目中所想的映射规则。即先由熟悉模式映射语义的专家用户提供一组源和目标实例。再通过对这些实例进行分析来生成规则集。但是，这些数据实例通常被预先假定为当前映射的解决方案和所有其他解决方案的典型代表。尽管由于上述方法在映射生成方面取得了一定的进展，但它们都有一个共同的特点，即它们都是面向专家用户的。这些用户通常都熟悉映射生成工具，并拥有完备的数据转换领域、映射的基本语义及其解决方案的知识。最终使得他们能够独立的设计较好的数据库查询代码或编写定制的代码。

正如文献[5]所讲述的一样，映射规则集的另一端通常是终端用户，他们通过挖掘异类数据源、Web 大数据、科学数据管理以及某些特定领域中各类型数据之间的关系来构建映射实例。一方面，构建映射实例的关键在于设计查询规范，然而查询规范对于非专业用户来说具有挑战性，并且比执行查询本身[11]更耗时。我们认为设计映射规范对这样的用户来说更加困难，这是因为映射包含了许多复杂的查询之间的语义关系。尽管近年来在非专业用户的查询规范方面做了很多努力[12] [13] [14]，但这些文献的研究成果对非专业用户来说并不适用。另一方面，越来越多的普通用户每天都要面对基于用户驱动的数据探索场景，例如文献[15]里面提到的几个具有代表性的数据空间项目。因此，针对这类用户相关的映射规则的设计就变得十分有必要了。

最后，为了解决上述已有研究存在的部分问题以及结合实际的地产大数据项目需要。本文在结合原有的文献的研究基础上提出了一种改进的基于数据实例(后文也叫做范例)的用户交互式映射设计方法。该方法使用范例元组做引导，对应于非专家用户提供的数量有限的元组。通过用户与这些元组之间简单的布尔交互来挑战用户，这些布尔查询问题旨在驱动用户所想到且事先未知的映射的推理过程。

2. 例子

结合实际的房地产项目数据，具体考虑如下几个 Web 源数据和目标数据表格，表 1~表 3 代表由非专家用户提供的源范例元组，同理表 4 和表 5 代表目标范例元组。

针对以上范例元组，对于房地产领域的数据分析师用户而言他们很容易会想到的初始映射规则集 M 如下：

$$\mathrm{LAT}(\mathrm{Idin}_1, \mathrm{Idin}_2, \mathrm{Idli}) \wedge \mathrm{LT}(\mathrm{Idli}, \mathrm{Name}, \mathrm{L})$$

$\wedge DT(Idk_1, Name', L) \wedge BT(Idin_1, St_1, L, Idk_1)$

$\wedge BT(Idin_2, St_2, L', Idk_1)$

$\rightarrow \exists idS_0, idS_1, idS_3, idS_2, idS_4 : T3(idS_0, Name, L)$

$\wedge T1(St_1, Idin_1, idS_1) \wedge T3(idS_1, Name', L)$

$\wedge T2(L, Idin_1, idS_3) \wedge T3(idS_3, Name', L)$

$\wedge T1(St_2, Idin_2, idS_2) \wedge T3(idS_2, Name', L)$

$\wedge T2(L', Idin_2, idS_4) \wedge T3(idS_4, Name', L)$

Table 1. Source instance E_S
表 1. 源实例 E_S

LAT (License Association Table)			LT (License Table)		
Ldnum	Zdnum	Idlicense	Idlicense	Name	Location
楼栋号	宗地号	许可证号	许可证号	楼盘名称	项目所在地
1Z#	2016-001	16357	16357	万科金地中山公园	沈阳市和平区

Table 2. Source instance E_S
表 2. 源实例 E_S

DT (Developer Table)		
Iddeveloper	Name	Location
开发商编号	开发商名称	总部地址
K1	沈阳万科中山置业有限公司	沈阳市和平区

Table 3. Source instance E_S
表 3. 源实例 E_S

BT (Building Parcel Association Table)			
Idinformation	State	Position	Iddeveloper
编号信息	状态	位置	开发商编号
1Z#	已发证	沈阳市和平区	K1
2016-001	已出让	和平区水上运动学校	K1

Table 4. Target instance E_T
表 4. 目标实例 E_T

T1 (State Table)			T2 (Address Table)		
State	Idinformation	Id	Location	Idinformation	Id
状态	编号信息	开发商编号	地址信息	编号信息	开发商编号
已发证	1Z#	a1	沈阳市和平区	1Z#	a3
已出让	2016-001	a2	和平区水上运动学校	2016-001	a4

Table 5. Target instance E_T
表 5. 目标实例 E_T

T3 (Developer License Association Table)		
Id	Name	Location
编号	名称信息	地址信息
a0	万科金地中山公园	沈阳市和平区
a1	沈阳万科中山置业有限公司	沈阳市和平区
a2	沈阳万科中山置业有限公司	沈阳市和平区
a3	沈阳万科中山置业有限公司	沈阳市和平区
a4	沈阳万科中山置业有限公司	沈阳市和平区

首先通过以上的映射场景，我们可以观察到用户提供的范例元组可能没有被很好地选择，甚至与用户所想的映射 M 不一致。此外，用户提供的范例元组往往也不会是需要推断的映射规则集的解决方案或通用解决方案。在文献[16]的开创性工作的基础之上，学者们开始广泛研究关于映射规则集的生成和优化的问题例如文献[17] [18]，但以上文献都假设了非常复杂的输入或更复杂的用户交互，显然这些文献里面的研究成果有待进一步改进。另外虽然本文中的目标元组回忆了数据实例，但它们与以上文献里面的实例相比，根本不同在于本文给出的目标解决方案并不是通用解决方案。

其次以上示例演示了本文部分的运行场景，其中一个非专业用户需要在显示城市房地产信息的两个数据库之间建立映射。源数据库模式由 License Association Table、License Table、Developer Table 和 Building Parcel Association Table (分别缩写为 LAT、LT、DT 和 BT)四个关系表组成，目标数据库模式由 State Table、Address Table、Developer License Association Table (分别抽象为 T1、T2、T3 和 T4)三个关系表组成。以上示例中用户为源和目标数据库提供的范例元组又可分别记为 E_S 和 E_T。

通过以上示例我们还可以观察到每个表所包含的元组数量都非常少，且用户并不打算提供完整的实例，而只是提供一小组具有代表性的元组。我们也可以很容易地观察到以上示例元组中的一些固有的歧义，比如常量沈阳市和平区既代表预售许可表项目地址(在关系 LT 中)，也代表具体的楼栋地址(在相应的关系 BT 中)，此外以上示例中的具体楼栋地址和宗地地址恰好与开发商总部的地址在同一城市的同一个行政区。如果我们将这些示例元组视为基础事实，我们将会把它们翻译为如上所示的映射规则集 M。然而，这样的映射并没有很好地解释范例元组，因为首先它假设所有的解决方案都必须包含同一家房地产开发商，且此开发商的总部地址与所开发的楼盘的详细楼栋地址和其所获得的宗地地址恰好与位于同一个城市。其次是预售楼盘的项目地址必须和地产开发公司的总部在同一城市的同一行政区。因此，从逻辑的角度来看，这样的映射方式太过于具体。此外，这种映射在真实的映射场景中可能会非常庞大并且不利于后续映射设计人员理解，因为它完全嵌入了所有的范例元组正如前文所说，我们的映射规范过程是建立在最终用户示例元组之上，这就可能导致我们设计的映射规则集不能够很好的解释原始数据。因此，本文的研究目的旨在构建一个能够为非专家用户提供映射规则集设计服务的通用框架系统，同时将以上由非专家用户提供的初始范例元组抽象为形如(E_S, E_T)这样的输入对作为系统的输入，则交互式映射规范问题就是通过与用户进行布尔交互来发现映射 M'，使得(E_S, E_T)满足 M'并且 M'逻辑蕴含 M，并最终导出更小的或者更加接近用户心目中的可控的映射规则集。本文的主要贡献总结如下：

1) 映射通常在企业 IT 和其他一些领域例如数据架构师(也称为工程映射的开发人员)中被指定和测试。本文仔细分析了以往用于帮助工程映射人员设计映射规则的一些范例通过大量的文献查阅发现其中的不

足之处，在借鉴别人的研究方法的基础上提出了一个更适用于非专家用户的模式映射规范设计框架系统，并适用于一般的 GLAV [19]映射。其中每优化一条映射规则，系统都会就布尔查询问题自动的与用户进行交互。

2) 经过本文方法(即初始规则预处理、原子优化、连接优化)处理后所生成的映射具有不可约的右边。再结合冗余映射消除这就保证了经过优化后的映射是标准形式的映射[20]。直观地说就是与庞大的初始映射规则集相比，优化后的映射规则对于非专家用户来说更容易理解。

3) 本文提出并实验测量了与可能映射空间相对应的类树形结构的两种映射空间探索策略(即自上而下探索策略和自下而上探索策略)，并最终确定了自上而下的探索策略的效率比自下而上的探索策略效率高。

4) 结合具体的项目在实际的地产数据集上进行了实验，验证了本文方法的可行性。

3. 相关概念

本节简单介绍了本文中所使用的各种概念，具体定义如下：

概念 1. 数据模型

给定两个可数的不相交的有限的常量集合 C 和变量集合 V，另外假设一个双射函数 $\bar{\theta}$，假如 $\bar{\theta}(x)=c$ 且 $c_i \in C$ 是与变量 $x_i \in V$ 相关的常量，那么 $\bar{\theta}^{-1}(C)=x$。以及一组属性 $\{A_0, A_1, \cdots\}$ 和一组关系 $\{R_0, R_1, \cdots\}$，每个关系 R 都有对应于属性的关系模式 $R(A_1, \cdots, A_K)$。模式 S 可以由若干关系构成，即 $S=(R_1, \cdots, R_n)$。关系 $R(A_1, \cdots, A_K)$ 的实例表示为一个有限元组 $R(A_1:V_1, \cdots, A_K:V_K)$，其中 V_i 可以是常量也可以是不确定值即变量。模式 S 的实例是该模式包含的所有关系实例的集合。在一个关系 R 上的元组有如下的形式，$R(c_1, \cdots, c_n)$，其中 $c_i \in C$。同理原子有如下形式，$R(x_1, \cdots, x_n)$，其中 $x_i \in V$。

概念 2. 依赖关系(模式映射关系)

一个映射是一个形如 $M=(S,T,\Sigma)$ 这样的三元组，其中 S 是源模式，T 是和源模式 S 不相交的目标模式。Σ 是一组建立在源模式 S 和目标模式 T 上的候选映射规则集 tuple-generating dependency (简称 tgd)。tgd 是形如 $\varphi(\bar{X}) \to \exists \bar{Y}, \psi(\bar{X}, \bar{Y})$ 这种形式的一阶逻辑公式，其中 $\bar{X}, \bar{Y}$ 代表含有变量的向量，$\psi(\bar{X}, \bar{Y})$ 和 $\varphi(\bar{X})$ 是原子公式的合取式且 $\varphi(\bar{X})$ 只含有 $\bar{X}$ 中的变量，此外，$\varphi(\bar{X})$ 和 $\psi(\bar{X}, \bar{Y})$ 还可包含形式为 $V_i = V_j$ 的等式，V_i 和 V_j 都是变量。

若 $\varphi(\bar{X})$ 和 $\psi(\bar{X}, \bar{Y})$ 分别为源模式 S 和目标模式 T 上的公式，则该依赖关系为源到目标依赖关系(source-to-target tgd)。若 $\varphi(\bar{X})$ 和 $\psi(\bar{X}, \bar{Y})$ 均为目标模式 T 上的公式，则该依赖关系为目标依赖关系(target tgd)。需要说明的是本文只考虑 source-to-target tgd (简称 s-t tgds)，其中 φ 中的原子来源于关系 S，ψ 中的原子来源于关系 T。另外本文也只研究 GLAV [17]映射，其中的 tgd 可以包含多个在 φ 和 ψ 中的原子。

概念 3. 范例元组

通常范例元组是由非专家用户提供的一对输入对其形式如下：(E_S, E_T)，E_S 代表源实例，E_T 代表目标实例。并且请注意，用户提供的范例元组可能跟最初的映射规则不是特别匹配，甚至与用户所想的的映射场景 M 不一致。此外，范例元组不应该是需要推断的映射规则的解决方案或通用解决方案。

概念 4. 初始映射规则

本文中需要被优化的初始映射规则是建立在非专家用户提供的初始范例元组的基础上的，假设与(E_S, E_T)相关的初始映射规则 tgd 是形如 $\varphi \to \psi$ 这样的形式，其中 $\varphi = \bar{\theta}^{-1}(E_S)$，$\psi = \bar{\theta}^{-1}(E_T)$ 也就是说左侧 φ 是由 E_S 构成，其方法是通过 φ 里面的原子配对物取代 E_S 里面的所有元组，并且用变量取代常量。同理右侧 ψ 也可以通过类似的原理得到。

概念 5. 同态和通用解决方案

假设 E_S 和 E'_S 是同一个模式上的两个实例，实例之间的同态 $h: E_S \to E_S$ 需要同时满足以下两个条件：

1) 对于每个每一个属于 $E'_S \cup E_S$ 中的常量 c，有 $h(c)=c$。

2) 对于每一个在实例 E_S 中的 $R(c_1,\cdots,c_n)$， $R(h(c_1),\cdots,h(c_n))$ 在 E_S 中。

对于一个映射 M 和一个实例 E_S，如果 E_T 是一个通用解决方案那么其必须满足以下条件：

1) E_T 首先必须是实例 E_S 的一个解决方案。

2) 对于每个别的方案 E'_T，存在一个同态 $h: E_T \to E'_T$ （$h(c)=c$， $c \in \text{constants}(E_T \cup E'_T)$）。

概念 6. 分裂减少映射(split-reduced)和 σ 冗余映射(σ-redundant)

将一个 tgd 拆分为多个逻辑上等价的 tgd 并且这些 tgd 的右边不存在相同的存在量词，那么这就满足了分裂减少映射的定义。标准的定义如下：

1) 假设 σ 是形如 $\varphi(\bar{X}) \to \exists \bar{Y}, \psi(\bar{X},\bar{Y})$ 这样的一阶逻辑。

2) 不存在形如 $\sigma_1: \varphi_1(\bar{X}) \to \exists \bar{Y}_1, \psi(\bar{X},\bar{Y}_1)$， $\sigma_2: \varphi_2(\bar{X}) \to \exists \bar{Y}_2, \psi(\bar{X},\bar{Y}_2)$ 这样的 tgds 使其满足 $\bar{Y}_1 \cap \bar{Y}_2 = \varnothing$ 且 $\{\sigma\} \equiv_L \{\sigma_1;\sigma_2\}$

σ 冗余映射的主要目的就是排除映射规则集中多余的重复定义的规则，其标准定义如下：

1) 假设 $M=(S,T,\Sigma)$， $\sigma \in \Sigma$。

2) $\Sigma \setminus \{\sigma\} \equiv_L \Sigma$。

概念 7. Chase 算法

Chase 算法的核心思想可以归纳为以下两个步骤：

首先，依次查看每条映射规则并将约束条件依次应用到初始的源数据实例上。其次，根据相应的映射规则计算出源数据实例的解决方案并使其能够满足规则所有的约束条件。

4. 映射优化

本节将详细描述交互式映射设计方法的关键部分首先是预处理，其目的是将一个看起来非常庞大的 tgd 划分为一组逻辑上等效的较小的 tgd (本文称之为标准映射)。其次是通过原子优化和连接优化对标准映射进行改进从而简化 tgds 的左侧。后续本文将在一小节详细解释第一步、二小节详细介绍本文所研究系统的两个核心步骤、而映射优化相关的算法将在三、四小节进行详细的描述。

4.1. 规则的预处理

为了便于后续对规则的优化处理，将初始的映射规则分割为较小的规范化的 tgd，这对于映射优化是很有必要的。因为它可以让映射设计人员只关注每个简化的 tgd 左边所隐含的必要原子。同时本小节还结合前面提到的 split-reduced mappings 和 σ-redundant mappings 的概念提出了一个评价映射好坏的标准定义，即每一个在 Σ 中的 tgd 都应该具有最小的右边并且在 Σ 中没有冗余的 tgd。

然而在对规则进行分解的时候,我们可能会在集合 Σ 中得到冗余的 tgds。这些冗余的 tgd 是不必要的，需要删除它们以避免后面系统与映射设计人员无用的交互。最后,我们说(*S*,*T*,Σ)是规范化的,当且仅当每一个在 Σ 中的 tgd 是 split-reduced 并且在 Σ 中没有冗余的 tgd。结合前面的实例具体的预处理过程如下：

假设 φ = LAT(Idin$_1$, Idin$_2$, Idli) ∧ LT(Idli, Name, L)

∧DT(Idk$_1$, Name', L) ∧ BT(Idin$_1$, St$_1$, L, Idk$_1$)

∧BT(Idin$_2$, St$_2$, L', Idk$_1$)

由以上定义可以导出 Σnorm = {

$$\varphi \rightarrow \exists \text{idS}_0,\ \text{T3}(\text{idS}_0,\ \text{Name},\ \text{L}) \tag{1}$$

$$\varphi \rightarrow \exists idS_1,\ T1(St_1,\ Idin_1,\ idS_1) \wedge T3(idS_1,\ Name',\ L) \quad (2)$$

$$\varphi \rightarrow \exists idS_3,\ T2(L,\ Idin_1,\ idS_3) \wedge T3(idS_3,\ Name',\ L) \quad (3)$$

$$\varphi \rightarrow \exists idS_2,\ T1(St_2,\ Idin_2,\ idS_2) \wedge T3(idS_2,\ Name',\ L) \quad (4)$$

$$\varphi \rightarrow \exists idS_4,\ T2(L',\ Idin_2,\ idS_4) \wedge T3(idS_4,\ Name',\ L) \quad (5)$$

}

4.2. 交互式映射设计执行过程

上一小节定义了生成规范化映射的预处理步骤。本小节将介绍交互式映射框架的两个核心优化步骤。首先，本文方法所基于的假设是非专家用户提供一对具体实例(E_S, E_T)，其中 E_T 不是通用解决方案。其次，对由以上实例导出的初始映射规则进行规范化处理。最后，在映射优化步骤中通过与用户进行简单的布尔交互从而得出最终的映射规则集。但同时我们也注意到对于给定的输入对(E_S, E_T)，满足它的映射的数量可能很庞大。因此，要想提高本文所述框架的效率，最重要的是提供有效的映射空间探索策略，以减少向系统与用户的交互次数。在本节的其余部分中为了便于说明，本文假设用户只提供了一对范例元组(E_S, E_T))。但是在实际的映射场景中，用户更可能提供的是一组小的实例，例如$\left(E_S^1, E_T^1\right),\cdots,\left(E_S^n, E_T^n\right)$。由于系统每次向用户提出的问题都集中在一个 tgd 上，并且由于我们不假定每个(E_S^j， E_T^j)中的 E_T 都是通用解决方案，因此本文的方法完全适用于实例对的集合。结合上面的阐述这里正式的定义交互式映射设计的具体过程：

a. 首先计算和每对(E_S^j， E_T^j)相关的初始映射规则 σ^j；

b. 将上一步的每个 σ^j 分解为 Σ_{norm}^i；

c. 删除上述结果中冗余的 tgds；

d. 对上一步的结果集中的每个 Σ_{norm}^i 执行原子优化和连接优化。

4.3. 原子优化

正如 4.1 节中所讨论的，规则预处理就是将初始映射规则集转换为符合 split-reduced 定义的映射规则集，其中每个 tgd 的左边都包含多个原子，如 φ。然而就实际的映射场景而言，φ 中的一些原子可能是多余的并且还可能进一步导致歧义。为了缓和以上歧义和简化规则，本文以前人的研究结果为基础提出如下算法：

算法 1：原子优化

输入：将要被原子优化的一组映射规则集 Σ
输出：将经过原子优化后的规则输出到 Σ′中
1: Σ′←∅
2: **for all** σ ∈Σ **do**
3: **let** σ =φ → ψ
4: C_{cand} ← generate set of possibles candidates from φ
5: C_v ←∅
6: **repeat**
7: e ← SelectAtomSet(C_{cand}, C_v)
8: **if** AskAtomSetValidity(σ ,e)**then**
9: add e to C_v
10: remove e and its supersets from C_{cand}
11: **else**
12: remove e from C_{cand}
13: **end if**
14: ++i
15: **until** C_{cand} =∅
16: **for all** e∈C_v **do**

Continued

```
17:            add the tgd (e →ψ) to Σ′
18:    end for
19:    j=i+j
20: end for
21: return Σ′
```

这里首先对上述算法 1 进行简单介绍，更详细的讲解将在下面小节通过具体的例子来说明。首先上述算法第 4 行根据每个待优化的 tgd 的左边原子构造候选映射空间，其次算法 1 第 6 到 15 行递归地遍历候选映射空间直到该空间不存在任何原子，最后再将每个优化后的 tgd 存储到 Σ′(16~17 行)。

4.3.1. 原子优化中树的应用

算法 1 采用类树的数据结构来处理数据主要是基于树是一种非线性的数据结构其叶子节点可以存储各种类型的数据元素，并且算法 1 需要处理的数据类型也并非常规的数据类型。

例 1. 考虑 4.1 节中 Σnorm 中的规则集，可以看出它们都具有同一个特征那就是具有相同的左边即集合{LAT(Idin$_1$, Idin$_2$, Idli)；LT(Idli, Name, L)；DT(Idk$_1$, Name', L)；BT(Idin$_1$, St$_1$, L, Idk$_1$)；BT(Idin$_2$, St$_2$, L', Idk$_1$)}，对于 Σnorm 中的每条规则来说以上原子的集合的子集，即为需要探索的全部映射空间同时也代表了树的叶子节点。

接下来将以树为数据载体结合 4.1 节中的规则 3 来解释算法 1 中的第 4 行代码是如何构造出一棵完整的候选映射空间树的，当然对于 4.1 节 Σnorm 中的其它规则来说也有类似处理过程。首先，通过观察我们可以注意到 Σnorm 中的规则 3 的右边的是形如 ∃idS$_3$，T2(L, Idin$_1$, idS$_3$) ∧ T3(idS$_3$, Name', L)这样的逻辑表达式，并且其中的原子 T2 和 T3 所包含的全部普通变量的集合为{L, Idin$_1$, Name'}。

其次，一个优化后的 tgd 必须满足一个基本条件，即其左边应该至少包含以上普通变量，就例 1 中给出的原子的集合来说，假设满足以上条件的最小的子集为如下两个集合：

{BT(Idin$_1$, St$_1$, L, Idk$_1$)，DT(Idk$_1$, Name', L)}和{LAT(Idin$_1$, Idin$_2$, Idli)，DT(Idk$_1$, Name', L)}并且所有不是以上任意一个集合的超集的集合都将被剪枝。

最后综合上述原理构造出的候选映射空间树图 1 所示：

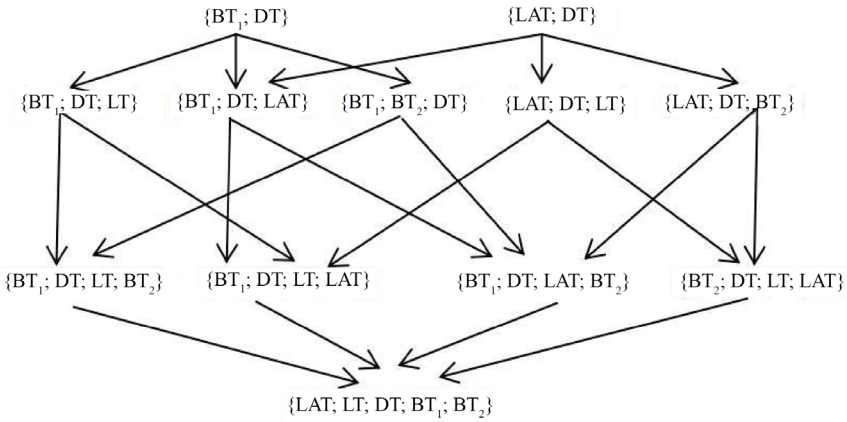

Figure 1. Candidate mapping space tree for examples 1 and 2
图 1. 例 1 和例 2 的候选映射空间树

以上空间树中的原子集：LAT = LAT(Idin$_1$, Idin$_2$, Idli), LT= LT(Idli, Name, L), DT = DT(Idk$_1$, Name', L), BT$_1$ = BT(Idin$_1$, St$_1$, L, Idk$_1$)，BT$_2$ = BT(Idin$_2$, St$_2$, L', Idk$_1$)

4.3.2. 候选映射空间树的遍历

在探索可能的映射空间(也就是对映射空间树进行遍历)的过程中，每经过一个叶子结点系统都会和非专家用户进行一次交互(算法 1 第 8 行)，同时选择叶子结点的方式将由算法 1 的第 7 行给出的探索策略决定。

例 2. 假设这里采用自顶向下宽度优先的探索策略对图 1 的映射空间树进行遍历。显然依据上一节的描述对图 1 的遍历就是对 4.1 节中的规则 3 进行优化，具体的优化过程如下：

首先当遍历节点{BT_1; DT}的时候，系统对用户提出如下问题：

“通过 BT(1Z#，已发证，沈阳市和平区，K1)和 DT(k1，沈阳万科中山置业有限公司，沈阳市和平区)能否导出 T2(沈阳市和平区，1Z#，a3)和 T3(a3，沈阳万科中山置业有限公司，沈阳市和平区)?”假如用户对这个问题回答为能，那么{BT_1; DT}的所有超集都将会被剪枝(也就是图 1 中的蓝色节点)。系统随后将输出如下规则：

$$BT(Idin_1,\ St_1,\ L,\ Idk_1) \wedge DT(Idk_1,\ Name',\ L) \rightarrow \exists idS_3,\ T2(L,\ Idin_1,\ idS_3) \wedge T3(idS_3,\ Name',\ L) \quad (6)$$

接下来就是遍历节点{LAT; DT}，此时系统对用户提出如下问题：

“通过 LAT(1Z#，2016-001，16357)和 DT(k1，沈阳万科中山置业有限公司，沈阳市和平区)能否导出 T2(沈阳市和平区，1Z#，a3)和 T3(a3，沈阳万科中山置业有限公司，沈阳市和平区)?”用户很可能对这个问题回答不能，因为这里存在一定的歧义，即开发商的总部地址必须和其负责开发的楼盘里面的具体住宅楼栋地址相同，显然这样的约束与现实的情况并不相同。

再接下来系统会继续遍历其下层的叶子节点，也就是集合{LAT; DT; LT}和{LAT; DT; BT_2}。针对这两个集合系统与用户最终的交互结果都是不能，最后对于集合{LAT; DT; LT; BT_2}采用同样的交互流程所得到的结果显然还是不能。最后算法 1 将输出最终的结果(6)。

4.4. 连接优化

在关系数据库中，相同值的多次出现并不一定意味着包含该值的属性之间存在相关的语义关系。本文的运行场景中的一个例子是常量“沈阳市和平区”的出现。它既是楼盘项目所在地也是开发商总部所在地，同时还是具体的住宅楼栋所在地。然而，初始映射规则强化了这种可能由于虚假使用同一变量而导致的以上共存现象。因此，初始映射规则可能会在 tgds 的左侧引入不相关的连接。要想最终生成既满足用户心中所想又没有歧义的相对完美的映射，我们首先需要从这些不相关的连接中区分相关的连接。在正式介绍连接优化之前不得不先引用两个数学概念：

集合的覆盖：若把一个集合 A 划分成若干叫做块的非空子集使得 A 中的每个元素至少属于一个分块，则这些分块的全体叫做 A 的一个覆盖。

集合的划分：给定一个集合 A 的一个覆盖 S，若 A 中的每个元素属于且仅属于 S 的一个分块。则将这样的 S 称作 A 的一个划分。

在以上概念的基础之上本文又给出了如下两个简单的定义：

定义 1：假设集合 A 的一个划分被表示为 P，那么表达式：$a \equiv_P b$ 指示集合 A 中的任意两个不同的元素在一个划分 P 的同一个块中。

定义 2：在定义 1 的基础上定义偏序 $P_0 \le P_1 \Leftrightarrow \forall x, y \in A, \left(x \equiv P_0 y \Rightarrow x \equiv P_1 y\right)$。

在数据库的连接查询中，连接通常由变量的多次出现来编码，我们将这些多次重复出现的变量称为连接变量。连接优化的主要原理就是用新的变量取代那些旧的被误用的变量以此来排除不相关的连接。综合以上背景和实际的房地产项目需要本文提出如下算法：

算法 2：连接优化

输入：一组在 σ 中且将要被连接优化的 tgds
输出：连接优化后的规则集 Σ'
1: $\Sigma' \leftarrow \emptyset$
2: **for all** $\sigma \in \Sigma$ **do**
3: **let** $\sigma = \varphi(\bar{X}) \rightarrow \exists \bar{Y}, \psi(\bar{X}, \bar{Y})$
4: $\Sigma_t \leftarrow \{\sigma\}$
5: **for all** $x \in \bar{X}$ **do**
6: **if** variable x occurs more than once in φ **then**
7: Cexplored $\leftarrow \Sigma_t$
8: $\Sigma_t \leftarrow \emptyset$
9: $\Sigma_t \leftarrow \Sigma_t \cup$ VarJoinsRefinement(Cexplored, x)
10: **end if**
11: **end for**
12: $\Sigma' \leftarrow \Sigma' \cup \Sigma_t$
13: $\Sigma_t \leftarrow \emptyset$
14: **end for**
15: **return** Σ'

上述算法 2 的简要执行步骤如下：

1) 对将要进行连接优化的规则初始化为标准的一阶逻辑表达式：$\varphi(\bar{X}) \rightarrow \exists \bar{Y}, \psi(\bar{X}, \bar{Y})$ (算法第 3 行)。

2) 对每一个属于 $\bar{x}$ 的普通变量，只要其满足在 φ 中的出现次数大于 2 就对包含该变量的连接进行优化，直到遍历完所有在 $\bar{X}$ 的普通变量(5~11 行)。

3) 最后就是将结果赋值给 Σ'并进行返回(12~15 行)。

接下来本文将结合具体的例子进一步解释以上算法的详细执行过程考虑如下例子：

例 3. 回顾例 2 的 tgd(6)如下：

$BT(Idin_1, St_1, L, Idk_1) \wedge DT(Idk_1, Name', L) \rightarrow \exists idS_3, T2(L, Idin_1, idS_3) \wedge T3(idS_3, Name', L)$

通过观察以上 tgd 可以很容易注意到它存在歧义，那就是在变量 L 的使用上即它既代表楼盘项目所在地也代表开发商总部所在地，同时还代表具体的住宅楼栋所在地。之所以出现这样的歧义是因为变量 L 的多次重复使用而造成的。算法 2 中第 9 行所包含的子程序 VarJoinsRefinemen 就完美的解决了这样的歧义。具体的伪代码如下所示：

算法 3：子程序 VarJoinsRefinement(σ, x)

输入：一个待优化的 tgd σ 和一个来自于 σ 的变量 x
输出：一组已经连接优化完毕的 tgds Σ'
1: 初始化 σ 得到 σ'
2: **let** $\sigma' = \varphi' \rightarrow \psi'$
3: $C_{cand} \leftarrow$ generate set of possibles candidates from σ'
4: $C_v \leftarrow \emptyset$
5: **repeat**
6: P $\leftarrow$ SelectPartition(C_{cand}, C_v)
7: $\sigma'' \leftarrow$ UnifyVariables(σ', P)
8: **if** AskJoinsValidity(σ'') **then**
9: add P to C_v
10: remove P and its More rough partitions from C_{cand}
11: **else**
12: remove P from C_{cand}
13: **end if**
14: **until** $C_{cand} = \emptyset$
15: $\Sigma' \leftarrow \emptyset$
16: **for all** P $\in C_v$ **do**
17: $\sigma'' \leftarrow$ UnifyVariables(σ', P)
18: add σ'' to Σ'
19: **end for**
20: **return** Σ'

在介绍算法 3 的执行原理之前，需要先介绍与之相关的两个定义具体如下：

定义 1：合格的划分 P

考虑前文例 2 中的 tgd(6)：$BT(Idin_1, St_1, L, Idk_1) \wedge DT(Idk_1, Name', L) \rightarrow \exists idS_3, T2(L, Idin_1, idS_3) \wedge T3(idS_3, Name', L)$，将其中的变量 L 根据其在规则中多次出现的位置分别用新的变量取代(即 L_1，L_2，L_3，L_4)得到如下规则：

$BT(Idin_1, St_1, L_1, Idk_1) \wedge DT(Idk_1, Name', L_2)$

$\rightarrow \exists idS_3, T2(L_3, Idin_1, idS_3) \wedge T3(idS_3, Name', L_4)$

由于本文不考虑在 tgd 中引入新的存在量词，因此每个出现在 ψ 中的变量都必须满足至少与一个出现在 φ 中的变量相绑定。如划分{{L_1; L_2}; {L_3; L_4}}为变量 L 的一个合格的划分，划分{{L_3}; {L_1; L_2; L_4}}为不合格的划分。综上更正式的定义如下：

考虑一个形如 $\sigma = \varphi \rightarrow \psi$ 的 tgd，存在一个变量 $x \in \varphi \bigcup \psi$ 且在该 tgd 中多次出现。我们将变量 x 的所有划分集合里面，那些满足其所有块里面都至少包含一个在 φ 中的变量的划分称为一个合格的划分。

定义 2：数据的存储结构

算法 3 和算法 1 一样同样采用类树的结构来存储数据，唯一不同的就是剪枝叶子节点的理论标准不一样。算法 3 的主要灵感来自于集合的划分，即给出任意两个不同的划分 P 和 P′，如果 $P \leq P'$且 P 是一个合格的划分，那么 P′也是一个合格的划分。这些合格的划分的不同的集合构成了树的不同节点最终构成了算法 3 的探索空间。

接下来本文正式开始讲解算法 3：

1) 对每个出现在σ中的 x 都使用一个新的变量去取代它，从而产生 σ'(算法第 1 行)。

2) 参照上面的定义 1 和定义 2 初始化树的叶子节点，保留合格划分的节点排除不合格划分的节点。(算法第 3 行)

3) 算法第 6 行主要是探索合适的遍历方式，任何有助于提高搜索效率的策略都可以运用在这里。

4) 算法的第 7 行和 17 行主要是统一变量，即存在于一个划分的同一个块里的不同名字的变量将被统一为相同名字的变量。

5) 算法第 8 到 13 行主要实现系统与用户之间的交互，通过交互的结果来决定后续系统的输出。

下面本文将通过一个具体例子来模拟以其执行过程：

例 3. 再次回顾例 2 中的 tgd(6)：

$BT(Idin_1, St_1, L, Idk_1) \wedge DT(Idk_1, Name', L)$

$\rightarrow \exists idS_3, T2(L, Idin_1, idS_3) \wedge T3(idS_3, Name', L)$

以上 tgd 所含的普通变量为 $\bar{X}$ = {$Idin_1$, St_1, L, Idk_1, Name'}根据算法 2 这里只考虑在 φ 中出现次数大于 2 的变量即 L 和 Idk_1。首先考虑 Idk_1 依据算法 3 将其分别重命名为 t_1 和 t_2，则可推导出如下的 tgd：

$BT(Idin_1, St_1, L, t_1) \wedge DT(t_2, Name', L)$

$\rightarrow \exists idS_3, T2(L, Idin_1, idS_3) \wedge T3(idS_3, Name', L)$

正如前文所讲此时树中存在两个节点，一个是{{t_1}; {t_2}}另外一个是{{t_1; t_2}}。接下来就是遍历这两个节点并且与用户进行交互，判断新引进的变量是否存在相关性。对于节点{{t_1}; {t_2}}用户可能会回答不相关此节点将被剪枝，因为此时开发商拥有与其所开发楼盘不相关的标识符。最后显然只剩下节点{{t_1; t_2}}，此 tgd 不做任何的改变。

然后考虑变量 L，经过重命名后导出如下 tgd：

$BT(Idin_1, St_1, L_1, Idk_1) \wedge DT(Idk_1, Name', L_2)$

$\rightarrow \exists idS_3, T2(L_3, Idin_1, idS_3) \wedge T3(idS_3, Name', L_4)$

可以观察到这里存在如下 5 个合格的划分：

$\{\{L_1; L_3\}; \{L_2; L_4\}\}$, $\{\{L_1; L_4\}; \{L_2; L_3\}\}$, $\{\{L_1; L_3; L_4\}; \{L_2\}\}$, $\{\{L_1\}; \{L_2; L_3; L_4\}\}$, $\{\{L_1; L_2; L_3; L_4\}\}$同时以上划分组成了四颗不同的树具体如下图 2 所示：

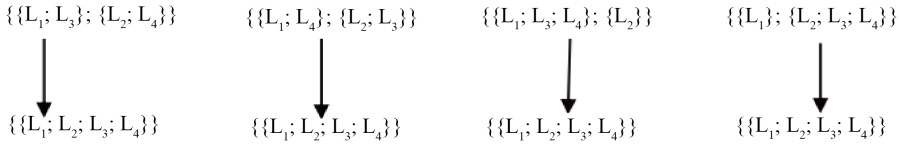

Figure 2. Diagram of example 3
图 2. 例 3 的划分图

首先遍历第一颗树，对于节点$\{\{L_1; L_3\}; \{L_2; L_4\}\}$来讲系统将对用户提出如下问题：

“通过 BT(1Z#，已发证，沈阳市和平区 $_1$，K1)和 DT(k1，沈阳万科中山置业有限公司，沈阳市和平区 $_2$)是否足够导出 T2(沈阳市和平区 $_1$，1Z#，a3)和 T3(a3，沈阳万科中山置业有限公司，沈阳市和平区 $_2$)?”显然以上划分对用户来说是可以接受的，即这些新引进的变量之间存在相关性。此时节点$\{\{L1; L3\}; \{L2; L4\}\}$的叶子节点将被剪枝，最后经过变量的统一得出如下 tgd：

$$BT(Idin_1, St_1, L_1, Idk_1) \wedge DT(Idk_1, Name', L_2) \rightarrow \exists idS_3, T2(L_1, Idin_1, idS_3) \wedge T3(idS_3, Name', L_2) \quad (7)$$

接下来同理遍历后面的树，然而剩下的划分无一例外全都将楼栋的具体地址与开发商的总部地址关联起来，这在现实环境中太过于具体了。因此用户将持续回答不相关最后系统输出最终结果 tgd(7)。

4.5. 最终的映射

最后对于本文所给定的一对范例元组来说，经过初始规则的生成、规则的预处理、原子优化、连接优化后得出的最终映射规则如下：

$$LT(Idli, Name, L) \rightarrow \exists idS_0, T3(idS_0, Name, L)$$

$$BT(Idin_1, St_1, L_1, Idk_1) \wedge DT(Idk_1, Name', L_2) \rightarrow \exists idS_1, T1(St_2, Idin_2, idS_1) \wedge T3(idS_1, Name', L_2)$$

$$BT(Idin_1, St_1, L_1, Idk_1) \wedge DT(Idk_1, Name', L_2) \rightarrow \exists idS_3, T2(L_1, Idin_1, idS_3) \wedge T3(idS_3, Name', L_2)$$

$$BT(Idin_2, St_2, L', Idk_1) \wedge DT(Idk_1, Name', L) \rightarrow \exists idS_2, T1(St_2, Idin_2, idS_2) \wedge T3(idS_2, Name', L)$$

$$BT(Idin_2, St_2, L', Idk_1) \wedge DT(Idk_1, Name', L) \rightarrow \exists idS_4, T2(L', Idin_2, idS_4) \wedge T3(idS_4, Name', L)$$

5. 实验验证

5.1. 数据集

本文首先，将从中国土地网、安居客、房天下等房产信息网站爬取的 300 多万条数据，经过垃圾清理、冗余信息删除两个预处理过程后得到的数据集作为本文实验的基本数据集。其次，为了更好的验证本文所提出的映射系统的有效性和可行性，我们又从以上数据集中筛选出部分典型的房产信息数据构建实验数据集。最后，借助映射设计工具 iBench [21]人工模拟了几个映射场景，这些映射场景能够基本满足本文对范例元组的要求。

5.2. 实验设置

本文的实验目标主要有两个：第一实验评估出与最终用户交互较少的映射空间探索策略。第二实验

评估出本文方法的可行性。

为了实现以上实验目的本文首先人为模拟了 5 个映射场景，并且以这 5 个场景为基础来构建范例元组。此时的范例元组可以看成是由一个完美的用户所提供即这些范例元组所包含的映射规则就是用户心中所想的完美映射规则。

其次，针对以上范例元组人为引入歧义或者错误，以此来模拟非专家用户所提供的范例元组的情况。本文将此操作定义为两次退化具体方法可以通过如下例子来说明：

例 4. 假设任意一条规则σ所产生的实例如下：

$$E_S^\sigma = \{\text{J(f1, f2, f3, a1); K(a1, f4, f5)}\}\text{和}\ E_T^\sigma = \{\text{J}_1\text{(A6, f3, f1, a2); K}_1\text{(a2, f4, f5)}\}$$

第一次退化首先在 E_S^σ 中随机选择一个元组进行复制，然后在该元组内随机选择一个常量并且用一个新的常量代替它，最后将这个新的元组加入到 E_S^σ 中。第二次退化主要是针对连接的即首先随机选择两个出现在 E_S^σ 中的常量，然后在 E_S^σ 和 E_T^σ 的范围内找到其中一个常量出现的所有位置并且用另一个常量代替这些位置上的常量值。综上最后得到退化后的实例如下：

$$E_{S1}^\sigma = \{\text{J(f1, f3, f3, a1); K(a1, f4, f5); J(f1, f3, f3, a1)}\};\ E_{T1}^\sigma = \{\text{J}_1\text{(A6, f3, f1, a2); K}_1\text{(a2, f4, f5)}\}$$

其中黑色下划线表示退化的地方。

最后以上面的阐述为基础给出如下表 6。

Table 6. Experimental mapping scenario
表 6. 实验映射场景

场景序号	|Σ|	$\bar{N}$	$\bar{Q}$
A1	10	1.2	2.8
A2	86	1.3	0.35
A3	6	3.5	1.3
A4	15	1.5	2.5
A5	12	1.5	3.9

其中|Σ|代表每个映射场景所包含的 tgds 的总的数量。$\bar{N} = \left(Nv/|V|\right)\times 100\%$，其中 Nv 是 tgds 中每个不同变量 v 出现的次数总和且 $v \in V$，V 是 tgds 中不同的变量的集合。同时随着退化的持续进行 $\bar{N}$ 会发生改变。$\bar{Q} = \left(P/S\right)\times 100\%$，其中 P 代表系统处理一个映射场景时与用户交互的总次数，S 代表该映射场景所包含的全部 tgd 的数量。

5.3. 实验结果

图 3 中横轴表示 $\bar{N}$，纵轴表示 $\bar{Q}$。从实验结果图 3 我们可以观察到除了场景 A3 以外，在其它的场景中都能观察到系统与用户之间的交互次数与 $\bar{N}$ 存在很强的线性相关性，换句话说就是本文提出的方法在实际的映射场景中具有可行性，因为系统与用户的问题交互次数并不会因为范例元组的增加而剧烈增长，而这也保证了本文方法的高效性。

其次还可以观察到在每个场景中，自上而下广度优先的映射空间探索策略所表现出的线性相关性要强于自下而上广度优先的映射空间探索策略。也就是说在实际的运用中我们应当采用自上而下广度优先的映射空间探索策略，因为此时算法的效率最高。

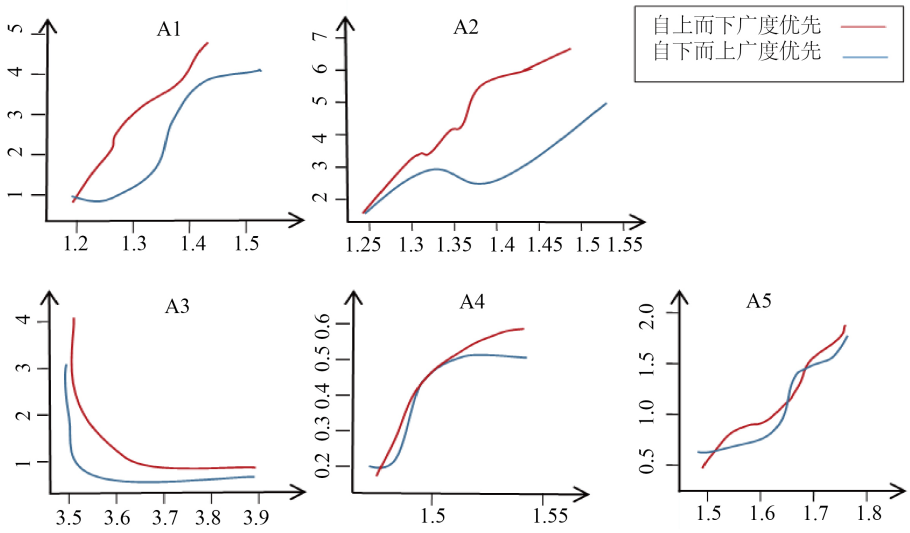

Figure 3. Diagram of experimental results
图 3. 实验结果图

6. 结论

首先，本文分析了目前存在的一些适用于普通用户的映射设计工具，从中发现其不足并在已有研究的基础之上设计了一个适用于非专家用户使用的映射设计系统。同时本文所设计的系统的主要工作原理就是，对由非专家用户提供的范例元组先进行规则的预处理，然后再将预处理后的规则输入到本文所设计的系统里面，经过原子优化(即原子的分解与选择)和连接优化(排除 tgd 中变量之间不相关的连接)两个步骤后最终输出更符合用户心中所想的映射规则。

其次，本文以 Web 地产数据集为实验数据，通过对数据的简单预处理和集成后人为的模拟了 5 个映射场景对本文提出的方法进行了实验，验证了本文方法的可行性。当然本文也存在不足的地方，比如在设计探索空间策略的时候可以考虑使用机器学习的方法，或者在用户交互阶段可以设计出相关的错误接受机制而不是简单的回答不能。

参考文献

[1] 纪宇航, 李贵, 李征宇, 韩子扬, 曹科研. Web 数据转换模式映射优化方法[J]. 数据挖掘, 2020, 10(1): 76-89. https://doi.org/10.12677/HJDM.2020.101008

[2] Fagin, R., Kolaitis, P.G., Miller, R.J. and Popa, L. (2005) Data Exchange: Semantics and Query Answering. *Theoretical Computer Science*, **336**, 89-124. https://doi.org/10.1016/j.tcs.2004.10.033

[3] 杨雪梅, 董逸生, 王永利, 钱江波, 钱刚. 异构数据源集成中的模式映射技术[J]. 计算机科学, 2006, 11(7): 1-5.

[4] Kimmig, A., Memory, A., Miller, R.J. and Getoor, L. (2017) A Collective, Probabilistic Approach to Schema Mapping. 2017 *IEEE* 33*rd International Conference on Data Engineering* (*ICDE*), San Diego, 19-22 April 2017, 915-932. https://doi.org/10.1109/ICDE.2017.140

[5] Bernstein, P.A. and Melnik, S. (2007) Model Management 2.0: Manipulating Richer Mappings. *SIGMOD*'07: *Proceedings of the* 2007 *ACM SIGMOD International Conference on Management of Data*, June 2007, 1-12. https://doi.org/10.1145/1247480.1247482

[6] Shvaiko, P. and Euzenat, J. (2005) A Survey of Schema-Based Matching Approaches. In: Spaccapietra, S., Ed., *Journal on Data Semantics IV*. *Lecture Notes in Computer Science*, Vol. 3730, Springer, Berlin, Heidelberg, 140-171. https://doi.org/10.1007/11603412_5

[7] Bonifati, A., Bellahsene, Z. and Rahm, E., Eds. (2011) Schema Matching and Mapping. Data-Centric Systems and Applications. Springer, Berlin. https://doi.org/10.1007/978-3-642-16518-4

[8] Alexe, B., Chiticariu, L., Miller, R.J. and Tan, W.C. (2008) Muse: Mapping Understanding and Design by Example. In 2008 *IEEE* 24*th International Conference on Data Engineering*, Cancun, 7-12 April 2008, 10-19. https://doi.org/10.1109/ICDE.2008.4497409

[9] Alexe, B., ten Cate, B., Kolaitis, P.G. and Tan, W.C. (2011) Designing and Refifining Schema Mappings via Data Examples. *SIGMOD*'11: *Proceedings of the* 2011 *ACM SIGMOD International Conference on Management of Data*, June 2011, 130-135. https://doi.org/10.1145/1989323.1989338

[10] Gottlob, G. and Senellart, P. (2010) Schema Mapping Discovery from Data Instances. *Journal of the ACM*, **57**, 6. https://doi.org/10.1145/1667053.1667055

[11] Jagadish, H.V., Chapman, A., Elkiss, A., Jayapandian, M., Li, Y., Nandi, A. and Yu, C. (2007) Making Database Systems Usable. *SIGMOD*'07: *Proceedings of the* 2007 *ACM SIGMOD International Conference on Management of Data*, June 2007, 13-24. https://doi.org/10.1145/1247480.1247483

[12] Abouzied, A., Hellerstein, J.M. and Silberschatz, A. (2012) Playful Query Specification with Dataplay. *Proceedings of the VLDB Endowment*, **5**, 1938-1941. https://doi.org/10.14778/2367502.2367542

[13] Papadimitriou, C.H., Abouzied, A., Angluin, D., Hellerstein, J.M. and Silberschatz, A. (2013) Learning and Verifying Quantified Boolean Queries by Example. *PODS*'13: *Proceedings of the* 32*nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, June 2013, 49-60.

[14] Diaz, G.I., Arenas, M. and Benedikt, M. (2016) Sparqlbye: Querying RDF Data by Example. *Proceedings of the VLDB Endowment*, **9**, 1530-1535. https://doi.org/10.14778/3007263.3007302

[15] Franklin, M.J., Halevy, A.Y. and Maier, D. (2008) A First Tutorial on Dataspaces. *Proceedings of the VLDB Endowment*, **1**, 1516-1519. https://doi.org/10.14778/1454159.1454217

[16] Popa, L., Velegrakis, Y., Miller, R.J., Hernández, M.A. and Fagin, R. (2002) Translating Web Data. *VLDB*'02: *Proceedings of the* 28*th International Conference on Very Large Databases*, Hong Kong, 20-23 August 2002, 598-609. https://doi.org/10.1016/B978-155860869-6/50059-7

[17] Francis, N. and Libkin, L. (2017) Schema Mappings for Data Graphs. *Proceedings of the* 36*th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, May 2017, 389-401. https://doi.org/10.1145/3034786.3056113

[18] Deutch, D., Gilad, A. and Moskovitch, Y. (2018) Efficient Provenance Tracking for Datalog Using Top-K Queries. *The VLDB Journal*, **27**, 245-269. https://doi.org/10.1007/s00778-018-0496-7

[19] 赵雨蒙. 基于模式映射的异构数据集成模型研究[D]: [硕士学位论文]. 济南: 山东大学, 2010.

[20] Gottlob, G., Pichler, R. and Savenkov, V. (2011) Normalization and Optimization of Schema Mappings. *The VLDB Journal*, **20**, 277-302. https://doi.org/10.1007/s00778-011-0226-x

[21] Glavic, B., Arocena, P.C., Ciucanu, R. and Miller, R.J. (2015) The iBench Integration Metadata Generator. *Proceedings of VLDB*, **9**, 108-120. https://doi.org/10.14778/2850583.2850586