# 一个用户主导的情景数据集成应用构造环境

王桂玲[1+], 曹波[2,1], 张赛[1], 耿美珍[1], 周浩山[1], 张峰[2,1], 刘晨[1]

[1](北方工业大学 云计算研究中心, 北京市 中国 100144)

[2](山东科技大学 信息科学与工程学院, 青岛市 中国 266590)

**摘 要** 随着网络的普及和深入应用，人们希望共享和集成丰富的网络信息资源，满足其个性化需求。论文提出了一个用户主导的情景数据集成应用构造环境 DSS，使能大量不具备专业编程知识的最终用户自行利用既有的网络信息资源即时构造应用。DSS 支持当前常见的网络信息资源，实现了交互式的网页资源个性化服务封装，将 Spreadsheet 和嵌套关系模型相结合，提供了可视化的嵌套电子表格操作和公式语言，支持用户进行数据服务的组合。论文给出了案例和相关工作分析比较，说明了 DSS 上述功能及特色的有效性。

关键词 数据集成； 数据服务； 情景应用；电子表格；Mashup

**中图法分类号** TP311 DOI 号：

# A user-steered situational data integration application building environment

Guiling Wang[1+], Bo Cao[2,1], Sai Zhang[1], Meizhen Geng[1], Haoshan Zhou[1], Feng Zhang[2,1], Chen Liu[1]

[1](Research Center for Cloud Computing, North China University of Technology, 100144, China) ,

[2](College of Information Science and Engineering, Shandong University of Technology, Qingdao, 266590, China),

**Abstract** As Internet is being widely and deeply used, Internet has become a vast repository of online information resources. It is essential herein to enable end-users to develop Situational Data Integration Applications by themselves, and thus to profit from the online information resources' potential value. The paper presents a user-steered situational data integration environment called *DSS*, which supports the current most used network resources, implements the ability to personalized wrap HTML web page interactively, combines Spreadsheet programming style and nested relational model, provides a set of visualized nested table operators and formula language, offers required agility and expressive power to support situational data integration by non-professional users. Use case and related work analysis reveal the potentials of *DSS* in situational data integration.

**Key words** Data Integration; Data Service; Situational Application; Spreadsheet; Mashup

## 1 引言

情景应用指一类为满足小规模用户群体的特定需求而构造的软件[1][2]。这里，特定需求常指一类特定环境下用户自发的、难以预知的需求，具有高度的个性化和动态性。例如，用户在出行时希望能有一幅此次出行相关的路线地图，上面标识出用户感兴趣的餐馆、景点的详细信息。用户的这种需求是在出行的时候即时发生的。近年来流行的 Mashup 即是情境应用的典型代表。在数据集成背景下，情景数据集成应用是指利用多种不同来源、

收稿日期：年-月-日*投稿时不填写此项*；最终修改稿收到日期：年-月-日 *投稿时不填写此项*. 本课题得到国家自然科学基金重点项目面向服务的软件理论、方法及其应用 (No. 61033006)；北京市自然基金重点面向大规模流式数据处理的数据空间理论与关键技术研究(No. 4131001)；北京市教委科技计划重点项目支持数据资源联动的云服务社区研究(No.KZ201310009009)以及北方工业大学科研基金资助. 王桂玲，女，1978年生，博士，E-mail: wangguiling@ict.ac.cn，副研究员，CCF会员，主要研究领域为云计算、服务计算、数据集成等. 曹波，男，1990年生，硕士生，E-mail:kusenanren@126.com；张赛，女，1988年生，硕士生, E-mail: qingqingbixue@163.com；耿美珍，女，1989年生，硕士生, E-mail: xinyu0889@163.com，硕士生；周浩山，男，1990年生，硕士生, E-mail: abcde237478460@qq.com；张峰，男，1981年生，博士生，E-mail: 15153206609@163.com，讲师，主要研究领域为服务计算、数据集成等。刘晨，男，1980年生，博士，E-mail: liuchen@ict.ac.cn. 主要研究方向为数据集成、云计算、服务计算等。

第1作者手机号码:18601063480, E-mail:wangguiling@ict.ac.cn

不同格式、不同特点的信息资源而构造的满足小规模用户群体特定需求的数据集成应用，本文称之为情景数据集成应用。

情景数据集成应用有以下需求特点：1）数据来自多个信息源，具有不可枚举性；2）不同用户集成的目标信息源常常不同，即便相同的信息源，不同用户也可能会有不同的集成需求；3）用户需求是即时发生的，其流程无法被系统预先建立的流程模型所涵盖，并且需要在较短时间内通过简单的操作即可由最终用户直接完成；4）具有阶段稳定性，用户具有重用以前构造的情景应用的需求。

针对上述需求，本文提出了一种基于数据服务的用户主导的情景数据集成应用构造环境（我们称之为 Data Service Space，简称 DSS），使能不具备专业 IT 编程知识的最终用户利用既有的网络信息资源即时完成情景应用的构造。

本文下面的内容组织如下：第 2 节介绍了系统设计的基本思想；第 3 节介绍了情景数据集成应用的设计原理；第 4 节介绍了原型案例；在第 6 节与相关工作分类进行了比较；第 7 节是本文结论。

## 2 基本思想

### 2.1 基于服务的数据资源一体化

不同类型的网络信息资源具有不同的信息表示形式，使得在访问网络信息资源时，需要针对其特定的表示形式采取相应的解析方式，无法统一处理，从而增加了网络信息资源的访问复杂性。针对这一问题，本文基于服务对数据资源进行一体化抽象，提出数据服务来支持统一的信息表示形式和访问接口。当前 DSS 支持的网络信息资源包括 HTML 信息源、关系数据库信息源、RSS/ATOM 信息源、以及以 XML/JSON 资源表述格式的 REST 服务等。我们将在 3.1 节介绍本文数据服务模型。

### 2.2 交互式的网页数据服务化

对于关系数据库信息源、RSS/ATOM 信息源、以及 XML/JSON 资源表述格式的 REST 服务，其结构化程度相对较好，相对易于映射为嵌套关系数据模型，具体的映射方法见我们之前发表的论文[3]。与这几类资源相比，HTML 信息资源没有显式给出数据模式定义，结构化程度最差，是网络信息资源一体化的难点。与传统的网页结构化信息抽取问题不同，本文面临的问题是让最终用户自主构造情景应用。为此，在 DSS 中设计和实现了交互式的网页数据服务化方法，为资源封装过程提供了所见即所得的图形化用户交互界面，抽取网页上的哪些元素由最终用户来进行自主定制。

### 2.3 借鉴Spreadsheet和嵌套关系模型

在各种最终用户编程的手段中，Spreadsheet 是一种相对比较成功的编程模式。但是，Spreadsheet 是扁平化的二维数据结构，不适合用来呈现和处理常见的 XML 等复杂结构的网络信息资源内容。嵌套关系模型（nested relational model）是一种放松了关系数据库理论中第一范式的限制的数据模型，它允许关系的分量为关系，从而属性的取值可以是一个关系[4]。嵌套关系数据模型简单、直观，可用来表示和呈现 XML 等复杂结构的网络信息资源，而且，它以嵌套关系代数为基础，具有丰富的表达能力。因此，本文提出将 Spreadsheet 和嵌套关系模型相结合，采用了嵌套关系模型作为基本的数据模型和数据可视化呈现方式，同时借鉴 Spreadsheet 的编程模式，支持类似 Spreadsheet 的可视化菜单，允许菜单操作直接作用在数据上，并允许用户通过编辑公式来灵活地进行数据处理。

### 2.4 借鉴示例编程

“示例编程(Programming by Example)”或“演示编程(Programming By Demonstration)”是比较常用的一种最终用户编程技术[5]，这种技术通过用户在某实例数据之上的操作来记录可重用的用户操作序列或推知程序的结构，从而为用户生成可执行的程序。本文拟借鉴示例编程的思想，在构造时，允许用户在一个样例嵌套表格之上进行数据的处理操作，系统将这些操作参数化后记录下来，生成一段组合脚本，用来表示数据服务的组合逻辑。以后用户重新执行该应用时，执行引擎负责解析、执行该组合脚本，从而将用户的组合逻辑应用到新的网络信息资源的实例上。

## 3 情景数据集成应用构造环境

### 3.1 数据服务

本文提出数据服务作为网络信息资源的统一抽象形式。图 1 示出了 DSS 数据服务的结构。数据服务以统一的访问接口接收客户端的输入，构造并发送 HTTP 请求消息，然后接收网络信息资源所在服务器的响应，并将服务器返回的响应文档转换为

符合嵌套关系模型的数据实例，最后把结果返回给客户端。

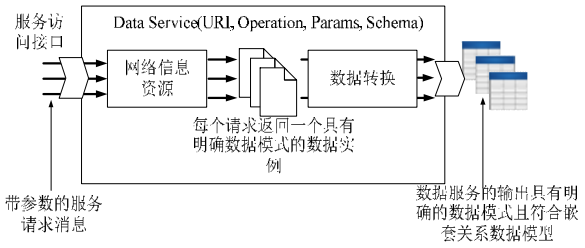


图 1．DSS 数据服务的结构

在 DSS 中，一个基本的数据服务至少需要对外暴露几个元素：访问地址、输入参数、服务操作和明确的数据模式。其中，一个 DSS 数据服务的访问地址对应一个网络信息资源的惟一 URI；DSS 数据服务使用 HTTP 协议标准的方法，数据服务的请求输入参数放在数据服务的 URI 中或者 HTTP 请求的消息体中传递；每个数据服务都有一个明确的数据模式 schema，它基于嵌套关系模型，用于描述服务返回结果，对于每次给定请求参数值的服务请求，数据服务都会返回一个数据实例，该实例遵循 schema。

### 3.2 交互式的网页数据服务化

DSS 中的网页数据服务化是一个用户交互和自动学习的迭代过程。如图 2 所示。

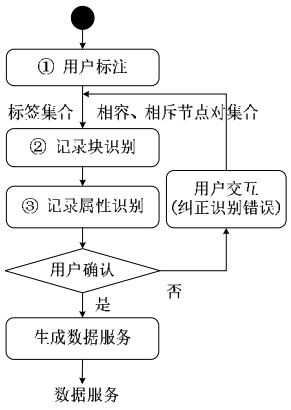


图 2．交互式的网页数据服务化流程

1）首先，用户在样例中标注出一个数据记录，结果表示为一组标签；2）接着，系统根据用户标注得到的标签集合从网页样例抽取记录块，即“记录块识别”。记录块识别的依据是网页中数据记录块相似性假设，即在同一个 HTML 信息源中，网页中描述一组同类数据记录的记录块具有一定的相似性；3）然后，将上一步识别出来的非样例记录块与用户标注的样例记录块的节点对齐，可以实现从非样例记录块中提取记录属性的值，形成数据记录；4）最后用户进行抽取结果确认，若确认无误，生成数据服务，若存在抽取错误，用户进行纠正后重新进行学习。该过程的核心是记录块特征值的提取及基于相似度计算的记录块识别算法，具体细节请参见我们的论文[6]。

### 3.3 基于嵌套电子表格的数据服务组合

#### 3.3.1 嵌套电子表格

在 DSS 中，在将网络信息资源进行服务化封装后，用户进行情景数据集成应用构造的过程就是对这些数据服务进行组合的过程。根据 2.4 节所述示例编程的思想，数据服务组合的过程是通过用户在数据服务的实例数据之上的操作来完成的。其基本过程如下：用户导入已经封装好的数据服务并配置其样例输入参数，系统调用数据服务，将数据服务返回的数据实例可视化为一系列样例嵌套表格（称为工作集）呈现给用户。用户利用 DSS 提供的各种操作在这些样例嵌套表格上进行数据的处理或调用新的数据服务。所有操作完成后，用户指定工作集中的哪个嵌套表需要发布为数据集成的最终结果。系统将其相应的复合数据服务发布出来（复合数据服务仍然符合 3.1 的数据服务结构）。这个过程是交互式的，系统不仅记录用户的参数化操作还记录用户操作的中间结果，从而支持用户根据每一步执行的结果来进行回退、调整等。

在嵌套电子表格中把所有数据组织为工作集，一个工作集包含若干有序的嵌套表，每个表都有一个在工作集内唯一的名称。工作集带有输入参数，它根据数据服务的输入参数设定。嵌套电子表格提供了一组定义在嵌套表、表中的列、行以及单元格之上的多粒度操作，图 3 示出了一个工作集的例子。工作集除了可以容纳来自于数据服务的动态数据之外，还可以同时容纳内容不会发生变化的静态数据。动态数据和静态数据在工作集中以一致的方式操作。

#### 3.3.2 用户操作和公式语言

在工作集中，数据被划分为三个粒度：表，列和单元格（如图 3 所示）。用户操作对应这些粒度而定义，可以分类为表操作，列操作和单元格操作。附录 1 列出了这些操作的形式化定义及文字说明的含义。

所有粒度上的编辑操作都依靠公式来进行。这些公式使用公共的表达式语言，其语法的 EBNF 表示见附录 2。与一般的 Spreadsheet 不同，DSS 提供了涉及集合的运算。例如，集合运算表达式计算两个集合的并、交、差。这些表达式可用于在表之间

进行计算。

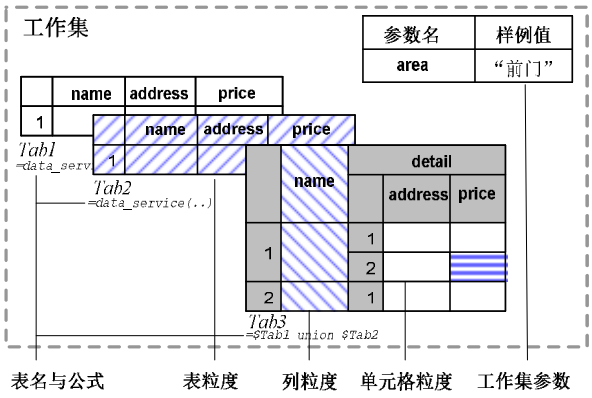


图 3 工作集示例

DSS 还提供了一系列的函数用以扩展用户操作和公式语言的能力。其中，函数 data_service 用于调用数据服务，它以数据服务的标识和参数作为参数。此外还有 count，sum 和 avg 等聚集函数。而函数 index 用于返回当前行的序号。

另一个与一般的电子表格不同的是，工作集中的表通过“$”符号加上表名得以引用；表中的列通过表引用加上路径表达式得以引用。路径表达式借用了 XPath 的路径表达式语法，例如表 Tab 中 detail 列的子列 address 列通过$Tab3/detail/address 引用。列中的单元格通过带有行号的列引用得以引用。例如，Tab 中的 address 列的第二个单元格通过$Tab3/detail[1]/address[2]得以引用。

DSS 中的公式语言尚没有可视化，普通用户掌握起来有难度，这是我们下一步的工作。

3.3.3 参数化机制

将用户对嵌套表实例上的操作和公式定义转换为复合数据服务的关键是将用户操作和公式定义参数化。参数化的难点在于如何在服务运行时解析数据引用。由于表引用和列引用分别通过表名和路径表达式进行，与特定的嵌套表实例无关，无需进行参数化。但是由于单元格是通过列名和行号引用的，而行号会在不同的数据实例上有所不同。因此，单元格引用需要进行参数化。我们通过条件列来实现单元格引用的参数化，用条件列上的值构成解析单元格引用的完整条件。例如，对单元格引用 detail[1]/address[2]来说，系统选择 name 和 address 作为条件列，将其解析为 detail[name=’北京’]/address[price=’30’]。

# 4 案例

我们通过一个留学生出行定制个性化餐馆导航服务的场景来说明本文提出的用户主导的情景数据集成应用构造环境的有效性。在这个“个性化餐馆导航”场景中，大众点评网站提供了餐馆的地址、价格等基本信息，餐馆英文菜单查询的 Open API 提供了餐馆有无英文菜单的信息，用户需要得到一个线路地图，上面显示了某地标附近餐馆的详细信息（既包含了地址、价格等基本信息，也包含了有无菜单的信息）。下面简要介绍用户利用 DSS 环境完成上述应用构造的简要步骤。

首先，用户通过 2.1 和 3.2 所介绍的数据资源服务化技术，可以将大众点评网站餐馆查询页面和餐馆英文菜单查询的 Open API 转化为数据服务。设 $S_1$ 表示封装大众点评网页的数据服务，$S_2$ 表示封装餐馆有无英文菜单 API 的数据服务，在嵌套表格中采取如图 4 所示步骤，即可得到输出结果为某地标附近餐馆的详细信息的复合数据服务。

步骤 1~2：导入大众点评数据服务 $S_1$ 创建表格 $T$；然后，为创建 $T$ 新的一列 $R_5$，这一列是大众点评数据服务和有无英文菜单进行组合的结果输出列；

步骤 3：在 $R_5$ 列编辑列公式输出结果。公式以 name 列为输入参数通过 data_service 函数调用英文菜单数据服务 $S_2$。

步骤 4：对 $R_5$ 列实施解嵌套操作，得到输出结果，保存后生成输出结果为某地标附近餐馆的详细信息的复合数据服务。

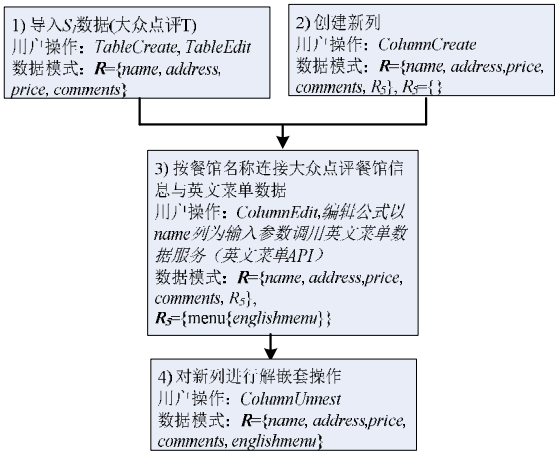


图 4 嵌套表中生成某地标附近餐馆详细信息的步骤

接下来，用户打开数据服务的地图定制应用，选择刚刚创建的复合数据服务，可将其呈现到地图上，并完成路线规划等任务。用户还可以通过应用

提供的个性化景点选择应用在地图中定制选取自己感兴趣的餐馆。经过上述操作，用户完成了一个情景数据集成应用的构造。

## 5 相关工作

用户主导的情景数据集成工具大体可以分为可视化语言[7, 8]，可视化数据流编程[9, 10]，以及基于 Spreadsheet 的编程等形式。其中，基于 Spreadsheet 的编程相关工作主要有 SheetMusiq [11], AMICO [12] 和 MashSheet [13]等。表 1 对它们与本文的 DSS 分别从支持的数据源类型、表达能力以及电子表格上的操作粒度等几个维度进行了比较。篇幅限制，具体分析在此不赘述。其中，DSS 表达能力的分析和证明请参考文献[3]。

表 1 相关工作比较

| 系统名称 | | Sheet Musiq | AMICO | MashSheet | DSS |
|---|---|---|---|---|---|
| 支持的数据源 | HTML | - | - | - | + |
| | XML/ JSON | - | + | + | + |
| 表达能力 | | 支持 SQL 查询 | 只支持控制流模式 | 支持控制流和数据流 | 表达能力大于 N1NF 与 XQuery 相当 |
| 操作粒度 | Table | + | - | - | + |
| | Column | + | - | - | + |
| | Cell | + | + | + | + |

## 6 结论

本文提出了一个用户主导的情景数据集成应用构造环境 DSS 以数据服务作为网络信息资源的统一抽象形式，支持包括 HTML 信息源、关系数据库信息源、RSS/ATOM 信息源、以及以 XML/JSON 资源表述格式的 REST 服务等在内的当前常见的网络信息资源；对于 HTML 信息源，DSS 支持普通用户交互式的网页资源个性化服务封装；DSS 将 Spreadsheet 和嵌套关系模型相结合，提供了可视化的嵌套电子表格操作和公式语言，支持用户进行数据服务的组合。案例和相关工作分析比较表明了 DSS 的有效性。

**致 谢** 本文要感谢北方工业大学韩燕波教授的指导和大力支持; 感谢 DSS 前期版本的开发者中国院计算所杨少华博士和季光博士; 参与 DSS 开发的德国 University of Applied Sciences Darmstadt 计算机系留学生 Matthias Wiatrok，以及参与 DSS 开发、测试的北方工业大学研究生孟昭彤、刁玺、朱美玲、张仲妹等。

### 参 考 文 献

[1] Jhingran., Enterprise information mashups: integrating information, simply. In: Proceedings of the 32nd International Conference on Very Large Databases, Seoul, Korea. 2006. pp 3-4

[2] Situational application, http://en.wikipedia.org/wiki/Situational_application

[3] Yanbo Han, Guiling Wang, Guang Ji, Peng Zhang: Situational data integration with data services and nested table. Service Oriented Computing and Applications 7(2): 129-150 (2013)

[4] L.S. Colby, A Recursive Algebra and Query Optimization for Nested Relations, In: *Proc. of the International Conference on Management of Data (SIGMOD'89)*, Portland, Oregon, USA, 1989, 273-283.

[5] A. Cypher, "Watch What I Do: Programming by Demonstration", *MIT Press*, 1993.

[6] Shaohua Yang, Guiling Wang, Yanbo Han. Grubber: Allowing End-Users to Develop XML-based Wrappers for Web Data Sources. The Joint International Conferences on Asia-Pacific Web Conference (APWeb) and Web-Age Information Management (WAIM), Suzhou, China. 2009, pp. 645–650.

[7] Braga D, Campi A, Ceri S. XQBE (XQuery By Example): A visual interface to the standard XML query language[J]. ACM Trans. Database Syst. 2005, 30 (2): 398-443.

[8] Borkar V, Carey M, Lychagin D, et al. Query processing in the aqualogic data services platform [C]. Seoul, Korea : VLDB Endowment, 2006.

[9] Altinel M, Brown P, Cline S, et al. Damia: a data mashup fabric for intranet applications[C]. Vienna, Austria: VLDB Endowment, 2007.

[10] Yahoo Pipes, Inc. http://pipes.yahoo.com/, 2013

[11]B. Liu, H. Jagadish (2009) A spreadsheet algebra for a direct data manipulation query interface. In: Proceedings of the 35th International Conference on Very Large Databases , pp 417-428

[12] Ž. Obrenović, D. Gašević (2008) End-user service computing: spreadsheets as a service composition tool. IEEE Transactions on Services Computing 1(4): 229-242

[13]Hoang D, Paik H-Y, Ngu A. Spreadsheet as a Generic Purpose Mashup Development Environment. In: Maglio P, Weske M, Yang J, Fantinato M, eds. Service-Oriented Computing. Vol 6470: Springer Berlin / Heidelberg; 2010:273-287

**附录 1. DSS 定义的用户操作**

| | 操作名称 | 备注 |
|---|---|---|
| 表操作 | ***TableImport<W, N, d>*** | 把一个新表 N 加入到工作集 W 中，把静态 JSON/XML 文档 d 中的数据填充到表 N 中 |
| | ***TableCreate<W,N>*** | 向工作集 W 加入一个空表 N |
| | ***TableEdit<W,i,F>*** | 用公式 F 计算结果填充工作集 W 中序号为 i 的表 |
| | ***TableCopy<W,i,N'>*** | 复制工作集 W 中的第 i 个表改名为 N' |
| | ***TableMove<W,i,j>*** | 把第 *i* 个表移动到新位置 *j* |
| | ***TableDelete<W,i>*** | 把第 *i* 个表从工作集 W 中删除 |
| | ***TableRename<W, i, N'>*** | 把工作集 W 中第 *i* 个表的名字改为 *N'* |

| | 操作名称 | 备注 |
|---|---|---|
| 单元格操作 | ***CellEdit<W, i, R, j, t, F>*** | 通过公式改变工作集 *W* 中 $T_i$ 表上的原子列 $R_j$ 下的单元格 t[$R_j$]内容为 F 的计算值 |

| | 操作名称 | 备注 |
|---|---|---|
| 列操作 | ***ColumnCreate<W, i, R, j, R'>*** | 创建与工作集 *W* 中 $T_i$ 表上的 $R_j$ 列平行的 R' |
| | ***ColumnEdit< W, i, R, j, F >*** | 在工作集 *W* 中 $T_i$ 表上的 $R_j$ 列上设置公式 F，从而改变该列上所有单元格的值 |
| | ***RowSort< W, i, R, j, Order>*** | 基于工作集 *W* 中 $T_i$ 表上的 $R_j$ 列上单元格的数值对行实施排序 |
| | ***RowFilter< W, i, R, j, V'>*** | 基于工作集 *W* 中 $T_i$ 表上的 $R_j$ 列上单元格的数值 V 实施筛选 |
| | ***ColumnNest<W, i, R, j, k>*** | 把工作集 *W* 中 $T_i$ 表上从 $R_j$ 到 $R_k$ 的一组相邻的列设置为一个新列的子列，并将这些列的行聚集成新列的单元格 |
| | ***ColumnUnnest <W, i, R, j>*** | 是嵌套的相反操作，它连接父列内部的行和外部的行，让表变得扁平 |
| | ***ColumnCopy <W, i, R, j, R'>*** | 复制列 Rj 并为新列取名为 R' |
| | ***ColumnMove<W, i, R, j, k>*** | 把列 Rj 移动到新位置 k |
| | ***ColumnDelete<W, i, R, j>*** | 用于删除列 Rj |
| | ***ColumnRename<W, i, R, j, R' >*** | 用于把列 Rj 重命名为 R' |

**附录 2. DSS 公式语言的定义**

公式定义

```
formular ::= expr;
expr ::= exprSingle ("," exprSingle)*;
exprSingle ::= ifExpr | orExpr | quantified;
```

基本表达式

```
ifExpr ::= "if" "(" expr ")" "then" exprSingle "else" exprSingle;
orExpr ::= andExpr ( "or" andExpr )*;
andExpr ::= comparison ( "and" comparison )*;
comparison ::= range (("="|"!="|"<"|"<="|">"|">=") range)?;
range ::= additive ( "to" additive )?;
additive ::= multiplicative (("+"|"-") multiplicative )*;
multiplicative ::= cartesian (("*"|"div"|"idiv"|"mod") artesian)*;
```

*集合表达式*

```
cartesian ::= union ( "^" union )*;
union ::= intersectExcept (("union"|"|") intersectExcept)*;
intersectExcept::= unary (("intersect"|"except") unary)*;
```

*路径表达式*

```
unary ::= ("-" | "+")* path;
path ::= ("/" relativePath?) | relativePath;
relativePath ::= step ("/" step)*;
step ::= (".."|QName|primary) (predicate)*;
predicate ::= "[" expr "]";
```

*量化表达式*

```
quantified ::= ("some"|"every") "$" QName "in" exprSingle("," "$" QName "in"
exprSingle)* "satisfies" exprSingle;
```

*数据引用与函数调用*

```
primary ::= Literal | varRef | parenthesized | functionCall;
varRef ::= ("$" QName) | ("~" QName) | "~..";
parenthesized ::= "(" expr? ")";
functionCall ::= "~"? QName "("(exprSingle("," exprSingle)*)?")";
```

*注: Literal 和 QName 的定义被省略。*