数据核算与错误计数

Michał J. Gajda[1]

[1]*Migamake Pte Ltd*

摘要

我们能否从复杂数据分析软件的输出中推断错误来源？双向编程承诺，我们可以反向追踪软件中的数据流，并将对输出的修正转化为对输入或数据分析的修正。

这使我们能够实现自动化调试、风险报告和大规模分布式错误跟踪的终极目标。

由于风险报告和数据分析管道的处理通常可以表示为一系列关系代数运算，我们提出以数据汇总代数取代这种传统方法，从而帮助确定错误的影响。该方法通过将数据分析定义为对数据集进行必然完整的汇总来实现，并且可以沿多个维度以多种方式进行汇总。我们还对这一方法进行了说明，以便更好地阐明输入数据的完整汇总如何有助于简化调试，并提高分析流水线的开发效率。这种方法也可以描述为将会计公理理论推广到数据分析领域，因此称为数据核算。

我们还提出了形式性质，使我们能够透明地断言单条记录对聚合数据的影响，并通过找出能够改变数据分析行为的最小变更，按记录粒度简化调试。

arXiv:2301.12583v1 [cs.PL] 29 Jan 2023

1. 引言

错误追踪和错误影响估算一直是实用自动化方法在风险报告中的圣杯（巴塞尔银行监管委员会，2013年）、大规模分布式错误追踪（Julian，2017年）以及自动化调试。

由于风险报告处理和数据分析管道通常可以表示为一系列关系代数运算，因此我们提出以数据汇总代数取代这种传统方法，以帮助确定错误的影响。该方法通过定义对数据集进行数据分析所需的完整汇总来实现，并且可以沿多个维度采用多种方式进行。

我们还提供了相关描述，以便更好地说明对输入数据进行完整汇总如何有助于简化调试，并提高分析流水线的开发效率。

这种方法也可以描述为将会计公理理论推广到数据分析领域，因此称为数据核算。

我们还提出了形式性质，使人们能够透明地断言单条记录对聚合数据的影响，并通过找出会改变逐条记录的数据分析行为的最小改动来简化调试。

arXiv©2022 年本文作者版权所有。CEUR研讨会论文集（
VS.org）

答案确实基于完全双向计算及其向保数据变换的推广。

2. 文献综述

巴塞尔银行监管委员会《最佳实践》（BCBS 239）（巴塞尔银行监管委员会，2013年）是一套关于金融机构如何组织风险报告管道的建议。这些建议还对数据聚合与报告的一般原则进行了审慎考量，因此成为其他汇总报告最佳实践相关工作的参考。

我们还受到近期关于数据流水线构建方法的观点中所提出的实践规则的启发（Gajda 2020a，2020b）。

2.1. 双向计算

可逆计算（Bennett 1973）旨在从输出反向推导输入。双向计算（Bancilhon 和 Spyratos 1985）试图推广程序，使其能够选取输出和输入中的任意一部分，并计算其余部分。

自动化调试旨在自动指出程序中可能存在错误的位置。自动化程序推导旨在根据给定的输入和输出提出有效的程序。

可微计算（Levitt 1983；Abadi 和 Plotkin 2019）旨在通过计算函数在枢轴点处的值和微分，找到该函数的一个最小值。

2.2. 数据库视图更新问题

数据库视图是从输入关系到输出关系的一种计算。数据库领域长期以来一直致力于解决视图更新问题（Chen 和 Liao 2011），即尝试对视图的计算进行逆向操作，从而将对结果的更新转换为对源关系的更新。

尽管这种方法的适用范围仅限于关系，但它对数据分析产生了重大影响，因为大型数据集通常直接建模为关系，或可转换为关系（例如，将图转换为关联关系）。

对这一问题的分析促使人们识别出视图补全，它包含视图中缺失但对计算更新至关重要的关系。

从这个意义上说，数据库领域将视图认定为非单射。为了实现可更新视图，该领域尝试通过计算视图补全来恢复视图的单射性。

2.3. 事件溯源

事件溯源用已发生事件的历史记录取代数据库，作为事实来源。这个事件日志 类似于会计中的总账，因为它只能追加记录，而状态不过是总账上的视图 或余额 。

2.4. 自动化调试

自动化调试方法旨在根据输出中的故障，找出程序中程序错误的位置。这可以通过类似双向程序的方法实现，但应用于程序源代码，而不是输入。

2.5. 会计的哲学与公理化

会计是一门科学，旨在通过洞察现金流和指标，预测公司的未来业绩（Ijiri 2018, 1975）。会计的数学公理化（Ijiri 2018）基于一种时间不变测度 的概念：它以一种随时间维度变化的方式划分货币，同时保持总和不变。

这与会计师的一项传统观察一致：相比只记录当前可用货币的总和，以及不断增加或减少货币，最好采用单调 的现金流测度，并在不同账户之间转移货币。

这种方法称为复式记账（Ellerman 2007），最适合通过将总余额表示为借方 和贷方 两方来说明。贷方减去借方 即可得出总余额。两方均为非负数，且随时间严格单调递增。这意味着我们拥有一种严格单调且易于跨时间验证的测度。

下一步是将复式记账推广到多个用途账户：例如，我们可以分别设立现金余额账户和支出账户，

为了解决难以用货币尺度衡量企业所有要素的问题，现代会计师编制报表时所依据的不仅是过往交易的分类账，还包括信息库 ，即用于编制财务报表的整个数据库（Warsono、Ridha 和 Darmawan，2009 年）。

请注意，以估值为中心的方法（如公允价值会计 – FVA）难以传达非货币信息，而且受货币稳定性的影响。此外，在易货交易中，一家公司直接以服务换取另一项服务、没有货币往来的情况下，这种方法也面临困难（Balzer 和 Mattessich，1991 年）。

与 FVA 相反，传统会计以“持续经营”为基础，按非清算基础对资产进行估值，因此着眼于公司的未来业绩。

信息含量学派强调，披露商业信息旨在减少有关主体的不确定性。该学派将会计信息视为一种主动的管理支持工具，其核心并非**公允价值**。

2.5.1. 会计计量

关于会计计量公理的研究Ijiri（Ijiri 2018）指出，所有相关对象都可按数量、体积、重量或其他测度进行量化，并划分为可数个类别。每个类别都必须有一种物理计量，但不同类别的物理计量不必相同。可替代性原则指出，数量相同的两个对象集合可以在交换时相互替代。不可替代的对象必须单独归为一类。测度必须是正确的数学测度，即非负、类别为空时为零，且可数可加（单调）。有些对象可能属于多个不同类别，这些类别通常对应不同的报告维度。时间是一个实变量，有些对象会随时间改变其类别。但对象的总计量

随时间推移必须保持不变。测度可以根据资产的当前价值和未来价值进行分配、估算和比较。默认采用截至某一时点的重置成本。

例如，我们可以估算出，2020 年一台工作站的购置成本为 1000 美元。由于磨损和陈旧，工作站的价值到 2021 年将降至 800 美元。但由于总计量必须保持为 1000 美元，我们将 2021 年的这一价值划分为两类：工作站当前价值 800 美元，以及已核销价值 200 美元。

Balzer 和 Mattesich（Balzer and Mattessich 1991）以及 Ellerman（Ellerman 2007）尝试对基于价值的会计进行公理化，使其将资源划分为互不相交的组别，但他们遵循的原则相似。

例如，Ellerman（Ellerman 2007）提出了一种解决方案，通过将信息拆分为一对单调递增的借方和贷方数值，对传统复式记账系统中现金流入和流出所包含的信息进行单调追踪。

一些会计哲学家补充指出，会计的目标不仅仅是汇总数量，还要预测企业未来的状态，并减少对这一状态的不确定性（Warsono、Ridha 和 Darmawan 2009）。这意味着，会计成为一门关于数据分析和汇总的科学，而会计计量只是在庞大的信息汇总体系中占据很小的一部分。

2.6. 错误影响报告要求

这里概述批量错误影响报告的基本要求：

分析流水线不会因记录失败而**崩溃**，而是针对给定的代数表达式整体运行，并提供正确关系 （其中包含正确且相关的结果）**以及** 错误追踪关系，以补充正确结果，并提供错误报告所需的信息。这意味着函数必须是全函数。

不会有任何信息被简单地丢失或丢弃。 每当相关信息不应影响输出时，我们都会将其归入所提供信息的**其他类别**，但该类别与当前处理无关，因此可以将其放入错误追踪关系中。我们可以在不同信息类型之间进行转换，但非空信息必须转换为非空信息实体。我们将信息视为单调的（只能增加）且可加的。

对于每个正确关系，其中包含在流水线当前阶段有效且相关的数据，我们还会保留一份错误追踪，记录那些已被丢弃、因而无法影响处理的记录。这样，我们便能估算错误记录（如果它们作为输入提供）可能造成的影响。这意味着我们会以错误输出的形式保留错误输入。

2.7. 铁路导向编程

在分析流水线的末尾，我们应该始终能够自动提供错误仪表板，汇总正确处理并纳入输出汇总的数据量，以及被丢弃的数据量和丢弃原因。这也意味着，用于提供输出最终摘要的仪表板 必须能够感知 所有输入和所有形式的错误。

我们应该自动为错误追踪添加所有必要信息，以便找出数据在流水线的哪个阶段被丢弃。这意味着我们应该将分析流水线划分为汇总阶段和转换阶段；转换阶段应保留足够的输入片段信息，以便一直评估其影响，直到最终汇总。

我们得出结论，满足这些要求的最佳方法是尝试保留所有**错误记录** 以及正确记录，并行汇总这些记录。

铁路导向编程 方法论（Wlaschin，2012）建议将所有**错误记录** 合并到单独的输出流中，并像汇总所谓的**快乐路径** 数据一样对其进行汇总（快乐路径是产生用于汇总的已验证记录的主路径）。这样便形成了两个并行处理流：一个转发有效数据，另一个转发错误和异常。

domains.	
es	Railroad-oriented programming switching between *happy path* and *error path*
or	N/A
	N/A
injectivity	N/A

3. 提议的工作

这项工作试图统一双向编程、铁路导向编程和会计学的方法，以兼具三者的最佳特性。也就是说，我们旨在提供调试数据管道的单射性（理解为找到能够修复程序的差集），为此，我们保持视图/过滤计算的严格单射性以及单调数据聚合。在这一过程中，我们发现，双向计算**和**调试的统一理论可以建立在会计学数学理论的基础上。

据我们所知，双向计算迄今尚未应用于单调数据聚合。我们还发现，这是首次探讨公理化会计学（Ijiri 2018）与双向计算及单调摘要（Ross 和 Sagiv 1997）之间联系的研究。

3.1. 动机示例

我们提出以下运行示例：集装箱船的装载内容。集装箱船示例表“items”：

描述	购买价格	商品	数量	Unit	保险
水手	无价的		17	人	2,000,000$
Nutella	10$		20,000,000	jar	
谷物		小麦籽粒	200	吨	
Milk		milk	5,000	升	
Cat			1	动物	
坚果油	未知	坚果油	1,000	升	
Ship	2,000,000,000$		1		1,000,000,000$

商品价格参考表“当前价格”：

市场	商品	价格	货币	每单位	报价 date	期货 date
远期	小麦 谷物	6.0981	USD	蒲式耳	2021-12-01	2022年2月01
spot	小麦 谷物	6.0575	USD	蒲式耳	2021年9月01	
spot	小麦 谷物	已关闭	USD	蒲式耳	2021-12-01	
spot	milk	33.98	EUR	100千克	2021-12-01	

报告表：

Name	描述	已入账	未入账
保险价值	保险人支付给所有各方	保险价值	猫、坚果 oil
重置 cost	所有者的重置成本发生损失时	购买价格或租用成本	猫，坚果 oil
重量	运输总质量	商品重量	水手，cat
重量	运输总质量	商品重量	水手，cat

重点在于展示，根据我们考察的是保险公司为船舶损失支付的总赔偿金、船舶所有者更换船舶的成本，还是因不合理延误而在市场上购买商品的成本，计算出的价值会有所不同。由于我们只关注数据集的完整汇总，因此只会构建汇总相关数据并汇总未纳入或被忽略数据的查询。这被视为最佳实践（巴塞尔银行监管委员会 2013年）。

3.2. 概述

影响报告最佳实践（巴塞尔银行监管委员会，2013年）与会计计量的数学理论（Ijiri 2018）之间的相似性并非偶然。我们可以形式化输入数据集的完整汇总，使之同时推广错误影响报告和上文定义的会计计量这两个概念。

不过，我们不采用较为狭义的数值测度，而是推广为一种信息概念，即给定集合上的加法幺半群。所有权概念被转化为该集合的划分。总计量随时间保持不变这一性质，则被处理过程中总信息幺半群保持不变这一性质所取代。

数据空间表示对数据集的完整汇总。只能表达完整汇总的语言，其运作方式不同于传统关系代数，因此，要对其进行定义，我们需要一种数学理论。

3.3. 定义

划分 一个集合是指一族互不相交的集合，这些集合覆盖整个空间。

实数 $\mathbb{R}$ 和非负实数 $\mathbb{R}_+ = \{x \| x \geq 0 \wedge x \in \mathbb{R}\}$.

我们用符号 $\leq$ 表示任意偏序，用 $\cup$ 表示带标签的不交并，其中 in_r 表示右标记， in_l 表示左标记。

我们用 $\cup$ 表示经典集合并，用 $\dot{\cup}$ 表示由集合组成的并集，其中右标签位于 $_r()$，左标签位于 $_l()$。

3.3.1. 数据空间

如前所述，我们将专注于给定输入的 完整汇总 。所谓完整，是指它们不会丢失信息。

我们将上述概念形式化为数据空间：
数据空间 是一个元组，包含以下内容（$\mathbb{D}$，$\leq_{\mathbb{D}}$，$\|_\|_{\mathbb{D}}$，（$\mathbb{M}$，e，$\diamond$）：

- 数据载体集 $\mathbb{D}$;
- 信息集 $\mathbb{M}$;
- 表示信息增长的偏序 $\leq_{\mathbb{D}}$[1];
- 集合元素的数据度量 $\mathbb{M}$: $\|_\| : \mathbf{2}^{\mathbb{D}} \to \mathbb{M}$。
- 信息融合幺半群 $\diamond_{\mathbb{D}}$： $\mathbb{M} \to \mathbb{M} \to \mathbb{M}$，它在 m 上是可加的;
- 融合幺半群的单位元 e 包含的是 $\mathbb{M}$ 的单位元，且其等于$\|\emptyset\|_{\mathbb{D}}$

- 我们期望信息融合 在载体集合 $\mathbb{D}$ 的任意划分下保持不变;
- 偏序的单调性 相对于信息融合：

(1)
$$a \leq b \wedge c \leq d \implies a \diamond c \leq b \diamond d$$

请注意，我们的偏序通常可以通过等式定律从性质良好的信息幺半群中派生出来：
$a \diamond b \leq a$ 和 $a \diamond b \leq b$。
它描述了一个带有偏序的加法幺半群，该偏序与此幺半群相容。这是因为我们只关注给定载体集合 $\mathbb{D}$ 上数据的完整汇总。

数据空间示例

- 会计计量(Ijiri2018)
- 任何带有有限可加度量的部分整体论空间，即一个由集合构成的族，配备了布尔代数的运算，以及定义在这些集合上的加法度量（Arntzenius 2004；Barbieri 和 Gerla 2021）；

- 用于记录并类型（Gajda 2022）的类类型，其偏序由信息融合幺半群派生，如 $a \diamond b \leq a$ 和 $a \diamond b \leq b$，而度量 $\|_\|$ 则作为类型推断运算。

以集合并作为信息融合的巴拿赫空间（度量空间）。

每个集合还配备了一个恒等数据空间：

(2)
$$\begin{aligned} \|_\|_{id} &:= id \\ \mathbb{D}_{id} &:= \mathbb{M}_{id} \\ \leq &:= \subset \\ \diamond_{id} &:= \cup \end{aligned}$$

[1]当然，$\leq_{\mathbb{D}}$ 是一个幺半预序（Fong and Spivak 2021）

另一个有趣的例子是帕西奥利代数，它描述了复式记账：

(3)

$$\begin{aligned} \mathbb{D}_p &:= \mathbb{R}_+ \times \mathbb{R}_+ \\ (d_1, c_1) \diamond (d_1, c_2) &:= (d_1 + d_2, c_1 + c_2) \\ \text{partial ordering } (d_1, c_1) \leq_p (d_2, c_2), &\quad \text{iff} \quad d_1 \leq d_2 \text{ and } c_1 \leq c_2 \end{aligned}$$

这是一个示例，说明如何通过将每个余额转换为一对严格为正的贷方和借方，基于账户余额创建加性数据空间。

数据空间（汇总）示例 每当对数据进行汇总时，我们都应尝试对错误路径中的字段进行类似的汇总；如果这些字段不存在，则应尝试统计其他影响度量。

对于每个表，我们通过为某一列指定合适的幺半群来表示可能的汇总，并指出可用于对结果进行分组的其他列。
例如，典型模式中的表 `order_details`：

表 5：汇总示例。

列	type	唯一	结果	聚合
订单 ID_	整数	唯一	集合	并集
产品_id	整数	-	标识符一组	并集
单位_价格	数量	-	标识符数量	最小值、最大值、平均值
数量	数量	-	数量	总和 （针对每种单位类型）
单位_价格 * 数量	数量	派生	数量	sum

读完这个示例后，我们还可以看到，报告幺半群呈现出某些趋势：

- 对象标识符、唯一标识符和维度通常用集合汇总
- 数值事实通常按总和、最小值、最大值和平均值进行汇总

通常会对相同数据类型（单位）采用相同的汇总方式。将不同数据类型（单位）汇总在一起通常是错误的；在这种情况下，我们会使用包含类别，例如统计不同类型的错误记录数量，而不是将它们的值相加。

自动汇总可以对所有允许的聚合执行汇总，以检测常见模式。

我们可以推断，对于那些在数据不完整时仍可使用的幺半群，可以采用这样一条启发式规则：对正确数据的每次汇总也应同时汇总不正确的记录。

这应当为分析主路径（或“快乐路径”）之外的数据提供合理的默认汇总方式。

3.3.2. 信息保持操作

信息保持操作可以用一种一般的方式表述为双向且因此可逆的操作，也可以表述为保持信息度量的操作。

猜想： 可逆操作总是能保持上述定义的任意特定信息度量。

数据空间划分 无损操作不是从输入中选择或投影数据，而是将数据集**划分** 为具有所需属性的子集。

例如

对于谓词 p，我们可以将其标记为：

$$(\mathbb{D}_{accepted}, \mathbb{D}_{rejected}) = partition(p, \mathbb{D})$$

$$\mathbb{D}_{accepted} = \{x | x \in \mathbb{D} \wedge p(x)\}$$

$$\mathbb{D}_{rejected} = \{x | x \in \mathbb{D} \wedge p(x)\}$$

信息保持的并集如果两个集合的成员都明确标记，使我们能够推断它们来自哪个输入集合，那么这两个集合的并集就是信息保持的。这可能是因为它们来自不相交集合，也可能是因为它们被明确标记：

$$\mathbb{D}_u = \mathbb{D}_1 \dot{\cup} \mathbb{D}_2 = \{\mathsf{in}_l(d_1) \| d_1 \in \mathbb{D}_1\} \cup \{\mathsf{in}_r(d_1) \| d_1 \in \mathbb{D}_1\}$$

在某些情况下，我们已经有了用途相同的标签，也就是说，存在一个来源函数 f：

(4)

$$\exists f. \forall x. f(x) = \begin{cases} \mathsf{in}_l() & x \in \mathbb{D}_1 \\ \mathsf{in}_r() & x \in \mathbb{D}_2 \end{cases}$$

信息保持函数如果我们只关心信息度量$\mathbb{M}_1$，..，$\mathbb{M}_n$和$\mathbb{M}'_1$，..，$\mathbb{M}'_n$，那么我们可以将范围缩小到对这些函数保持信息的函数，也就是说：

给定函数 f：$\mathbb{D} \to \mathbb{D}'$，我们可以计算出与 g：$\mathbb{M}_i \to \mathbb{M}'_i$ 对应的映射，使得 $g(m(x)) = m'(f(x))$。

信息保持函数示例在最简单的示例中，可能会出现这种情况$\mathbb{D} = \mathbb{D}'$ 和 $\mathbb{M}_i = \mathbb{M}'_i$。

另一个例子是根据某个谓词重新划分数据集，或为每条记录计算额外的量。

注意：聚合通常只保留与其维度相对应的信息（例如，对数量求和只保留这一维度的信息，但不会保留记录的数量）。后文将讨论如何通过对各个维度上的聚合取张量积，来构造能够保留多个维度信息的聚合。

3.3.3. 从较简单的数据空间派生复杂数据空间

现在我们来讨论如何从较简单的数据空间派生复杂数据空间。

3.3.4. 张量积

我们已经看到，数据度量空间可以从性质良好的信息融合幺半群派生得到。

现在，我们尝试将符合不同数据度量的子空间组装成数据空间：

不交张量积 两个数据空间的不交张量积 $\mathbb{D}_1 \otimes \mathbb{D}_2$ 是满足以下条件的数据空间：

- 数据载体集为 $\mathbb{D} = \mathbb{D}_1 \times \mathbb{D}_2$;
- 偏序 $a \leq c \wedge b \leq d \implies (a, b) \leq (c, d)$;
- 信息融合 $(a, b) \diamond (c, d) = (a \diamond_1 c, b \diamond_2 d)$
- 单位元 $e = (e_1, e_2)$

不交张量积 给定两个定义在同一集合上的数据空间，我们可能希望使用复合数据度量 来表示两个空间中的数据度量所传达的信息：两个数据空间 $\mathbb{D}_1 \bigoplus \mathbb{D}_2$ 的平行积 是满足以下条件的数据空间：

- 载体集合 i1s $\mathbb{D} = \mathbb{D}_1 = \mathbb{D}_2$;
- 偏序 $a \leq c \wedge b \leq d \implies (a, b) \leq (c, d)$;
- 信息融合 $(a, b) \diamond (c, d) = (a \diamond_1 c, b \diamond_2 d)$
- 单位元 $e = (e_1, e_2)$

信息重构与信息投影 给定两个数据空间 $\mathbb{D}_1$和 $\mathbb{D}_2$，其数据载体相同，但信息载体 $\mathbb{M}_1$ 和 $\mathbb{M}_2$不同，我们可以通过对信息载体 $\mathbb{M} = \mathbb{M}_1 \bigoplus \mathbb{M}_2$取张量积，并使用诱导偏序和信息融合幺半群来构造一个 $\mathbb{D} = \mathbb{D}_1 \oplus \mathbb{D}_2$：

- $a \leq b$ 当两者都 $\|a\|_{\mathbb{D}_1} \leq_{\mathbb{D}_1} \|b\|$ 且 $\|a\|_{\mathbb{D}_2} \leq_{\mathbb{D}_2} \|b\|_{\mathbb{D}_2}$
- $m \diamond_{\mathbb{D}_1 \oplus \mathbb{D}_2} n = (m \diamond_{\mathbb{D}_1} n, \ m \diamond_{\mathbb{D}_2} n)$

对于互不相交的信息载体，我们可以将组件数据空间 $\mathbb{D}_1$ 和 $\mathbb{D}_2$ 投影称为：

$$\text{Proj}_{\mathbb{M}_1}(\mathbb{D}) = \mathbb{D}_1$$

$$\text{Proj}_{\mathbb{M}_2}(\mathbb{D}) = \mathbb{D}_2$$

函数乘积 由于我们关注的是对数据空间的变换，因此我们将数据空间上与上述函数乘积相对应的函数乘积记作：

$$(f \otimes g)(x) \begin{cases} \text{in}_r(x) & \to f(x) \\ \text{in}_l(y) & \to g(y) \end{cases}$$

$$(f \Cup g)(x) = (f(x), g(x))$$

偏函数 每个偏函数 $f : \mathbb{D} \mapsto \mathbb{M}_1$ 都可以通过将数据空间 $\mathbb{D}$ 划分为两个不同的片段 $\mathbb{D}_1$ 和$\mathbb{D}_2$，扩展为全函数$\mathbb{T}(f) : \mathbb{D} \to \mathbb{M}$：

$$\mathbb{T}(f)(a) \begin{cases} \text{in}_r(f(a)), & \text{if } x \in \mathbb{D}_1 \\ \text{in}_l(a), & \text{otherwise} \end{cases}$$

这意味着 $f : \mathbb{D} \mapsto \mathbb{M}_1 \dot\cup D_1$ ——

查找和连接 连接可以表示对信息的查找，而这些信息可能会改变已处理记录的影响。但也可能出现这种情况：即使在正确数据上计算内连接，部分错误流数据仍会缺少连接所需的键字段。因此，对于每个正确数据的连接，我们都会在错误路径数据上执行类似的**外连接**。因此，我们需要**始终**在错误路径数据上执行外连接，同时包含位于正确路径上的记录和错误记录

我们将信息保持操作限制为外连接，并将结果标记为左侧、右侧和内部输出。内部输出由根据连接条件 c匹配的元组组成，左侧部分由来自 $\mathbb{D}_1$ 且未与 $\mathbb{D}_2$中的任何元组匹配的元组组成，右侧部分由来自 $\mathbb{D}_2$ 且未与 $\mathbb{D}_1$中的任何元组匹配的元组组成。

$$\mathbb{D}_1 \bowtie_c \mathbb{D}_2 = (\mathbb{D}_{left}, \mathbb{D}_{inner}, \mathbb{D}_{right})$$

对于谓词 p，我们可以将其标记为：

$$\mathbb{D}_{inner}) = \{(x, y) | x \in \mathbb{D}_1 \wedge y \in \mathbb{D}_2 \wedge p(x, y)\}$$

$$\mathbb{D}_{left} = \{x | x \in \mathbb{D}_1 \wedge \nexists_{y \in \mathbb{D}_2} p(x, y)\}$$

$$\mathbb{D}_{right} = \{x | x \in \mathbb{D}_1 \wedge \nexists_{y \in \mathbb{D}_2} p(x, y)\}$$

请注意，连接条件 c 是对 $\mathbb{D}_1$ 的笛卡尔积的筛选条件， $\mathbb{D}_2$ 且只依赖于连接列（通常是键）。

值查找示例 我们以在包含 product 的表中查找商品当前价格为例， quantity 持仓。参考表以 product 为键，包含 description 和 current_price。当我们能保证 product_id 匹配时，常规查找采用内连接；我们将其转换为外连接，其中左侧对应于参考表中缺少 missing 的 product_id 条目，右侧对应于参考表中缺少 unused 的 product_id 条目。

左右两侧分别汇总为针对左侧表的错误反馈和针对右侧表的 unused反馈，以确保记录不会丢失。

这样我们就可以轻松发现过时引用，或像用 oli 代替 oil 这样的拼写错误，并将其正确报告为汇总行：共有 14 行，5 个单位的产品名称未知， Nut oli 是由拼写错误造成的。

值查找示例 我们以在包含 product 的表中查找商品当前价格为例， quantity 持仓。参考表以 product 为键，包含 description 和 current_price。当我们能保证 product_id 匹配时，常规查找采用内连接；我们将其转换为外连接，其中左侧对应于参考表中缺少 missing 的 product_id 条目，右侧对应于参考表中缺少 unused 的 product_id 条目。

左侧和右侧要么都汇总为针对左侧表的错误反馈，以及针对右侧表的 unused反馈，以确保记录永不丢失。

这样我们就能轻松找到过时引用或拼写错误，例如用 oli 代替 oil ，并将其正确报告为汇总行：包含 14 行，以及 5 个未知产品名称为 oli的单位。

列式投影和行格式 由于我们坚持不丢弃任何可能与调试工作相关的数据，因此不能简单地通过投影运算符丢弃字段。因此，我们的记录包含两种类型的字段：

- 与最终输出**相关的**字段
- 与最终输出**无关的**字段，仅为调试目的而保留

对于每条记录，每个**相关的**字段都只有一个值，但每个**无关的**字段可能有一组值，以便在丢弃重复的**相关的**记录时将其考虑在内。这意味着每条记录都有一个仅保留无关字段的子关系。

这使我们能够实现一种无损投影 操作，将相关字段移入无关字段集。

请注意，投影操作对正确关系及其对应的错误跟踪关系的执行方式相同。

3.4. 逐行数据丰富操作

在 SQL 语法中，通常会从单个表的查询中添加一个计算字段：

```
select "Purchase price"*"Amount" as "Total price"
  from items;
```

我们将这些操作称为数据丰富，并且可以对所有行执行这些操作。然而，为了最大限度地发挥调试作用，我们提出了一种方法：在操作流水线中尽可能早地自动安排计算额外字段。这样，我们就能提供更多信息。记录离开正确流并进入错误流后，为防止级联故障，我们不再为其填充与主要计算相关的新字段值。

W我们只能为这些记录补充更多特定于错误的信息，例如添加错误描述
p- tion
以及估算这些信息的影响。

因此，除了在 fmap 操作正确数据流上执行操作外，我们还允许在错误流上执行 emap 操作，以丰富错误信息。我们可以选择在流水线到达相应阶段时、且在任何记录被添加到错误流之前，提前安排计算这些信息：

```
pipeline = doforM filename -> do
emap (@"Filename" :-> filename) $ do…
```

组合代数最后一步是以恰当的方式使用这些操作：这意味着所有数据空间都必须包含在最终聚合中，或者被为该聚合做出贡献的数据空间使用。就像在线性逻辑中，我们不允许多次使用某些变量一样，在数据空间代数中，我们也不允许遗忘某个变量。

3.5. 转换关系代数

关系代数是一种标准代数，用作描述数据库操作和分析流水线的基础。

这些基本操作通常在由行（形状相同的记录）组成的集合上实现，这种集合称为**关系**。不过，它们同样也可以在多重集或记录列表上实现[2]
它使用以下基本操作：

- 投影会丢弃一组列，只保留与后续处理相关的列

- 选择会选取记录的一个子集
- 重命名会将某些字段重命名为其他名称
- 两个关系的笛卡尔积（叉积），输出为由两个输入中所有可能的行对组成的列表

[2]在这种情况下，通常还会有一个用于排序并提取唯一、不重复记录子集的操作。

- 并集、差集和交集运算

请注意，所有连接、交集和差集运算在技术上都是通过外连接加后处理步骤实现的。

为方便起见，我们还提供了 lambda 或映射操作，用于根据其他字段计算出一个或多个新字段（在逻辑上是冗余的，但提供此功能是为了方便）。

表 6：关系代数和 SQL 中的运算到数据空间代数的转换。关系

代数	SQL	数据空间代数
投影	`SELECT field FROM ...`	提取
选择	`... 或 WHERE ... HAVING ...`	划分
重命名	`WHERE ... 或 HAVING ...`	映射函数
叉产品	`SELECT ... FROM table1, table2`	不交张量积
外连接	`OUTER JOIN`	先对每个表进行由 NULL 诱导的划分，然后并行张量积
自然 join	`INNER JOIN`	对由非 NULL 诱导的划分进行并行张量积 对每个表
并集	`UNION`	不交张量积（带标签并集），然后划分 移除重复项，并映射以移除标签
多重集并集	`UNION ALL`	带标签并集，然后应用信息保持函数，该函数 移除标签
差集	`MINUS`	对第一个表进行划分，将共有元素放入错误集
交集	`INTERSECT`	对每个表进行划分，将共有元素放入输出集

使用这些操作，我们还可以实现连接。此外，还提供了连接和聚合操作。

连接是接收两个关系并生成行的操作；这些行包含更大的记录，由两个输入合并而成。

此外，我们还添加了聚合操作：

- 分组会生成一个关系，其中的记录是子关系，包含来自同一子组的字段
- 聚合操作：对给定字段执行任意幺半群运算，并生成关系中所有记录的幺半群求和行

上述操作集合被广泛认为足以满足大多数提取、转换、加载和数据聚合需求。每当有人提出一种新的数据库或数据分析操作时，通常都会将其与基本关系代数操作进行比较。因此，我们将提出一组操作，使我们能够获得与关系代数表达式相同的结果。

4. 讨论

4.1. 错误聚合与错误估计

使用错误追踪关系代数，我们可以轻松地对单条记录如何影响处理做出一般性断言（因为每次整体汇总都必须考虑所有记录）。当我们期望某条给定记录完全正确时，只需断言其字段最终包含在正确路径上的正确记录集中。

如果我们期望某条给定记录被丢弃，只需断言它存在于错误集中。

这也意味着，断言失败时，我们可以快速定位所有可能需要修改以修复问题的代码位置或数据输入。

4.2. 在分析中替换关系代数

我们提出用一组操作取代关系代数，这组操作能够执行相同的计算，但其组织方式可以确保所有数据都得到汇总并纳入其中。我们借鉴会计理论为此提供依据，因此将为确保符合统计与分析最佳实践而对数据集进行完整汇总的过程称为“数据核算”。我们阐述了如何将经典关系代数转换为这种新的“数据空间代数”。这种代数支持对数据进行可逆变换，并能在汇总数据的同时保留会计数学原理。文中还简要介绍了如何在传统 SQL 之上实现“数据空间代数”操作。

5. 结论

我们还建议对错误表中所有定义明确的列采用统一的汇总机制，以便发现错误中的模式。

我们提出将数据空间代数作为传统关系代数的替代方案，以实现更稳健的报告，并建议运用会计原则开展更严谨的数据分析。我们简要说明了如何转换传统关系代数表达式，使其中一个输出能够提供与关系查询相同的信息。自然，为了完整核算输入数据，我们还需要汇总其他信息，以确保所有数据均得到核算。

参考文献

Abadi, Martin, and Gordon D. Plotkin. 2019. “简单的可微编程语言。”《*ACM*程序设计语言会议论文集》4（POPL）。https://doi.org/10.1145/3371106。

Arntzenius, Frank. 2004. “粘块、拓扑学与测度。”载于《牛津形而上学研究：第4卷》，Dean Zimmerman 编。牛津大学出版社。Balzer, Wolfgang, and Richard Mattessich. 1991. “会计公理基础：一种结构主义重构。”《理论与决策》30 (3): 213–43。Bancilhon, François, and Nicolas Spyratos. 1985. “数据库中的代数独立性与概率独立性。”载于第四届ACM SIGACT-SIGMOD 数据库系统原理研讨会论文集，1985年3月25日至27日，美国俄勒冈州波特兰，149–53。ACM。https://doi.or g/10.1145/325405.325424。Barbieri, Giuseppina, and Giangiacomo Gerla. 2021. “部分整体论空间中的测度定义（探索性论文）。”《逻辑与逻辑哲学》31 (1): 57–74。https://doi.org/10 .12775/LLP.2021.005。巴塞尔银行监管委员会。2013年。“BCBS 239：有效风险汇总与风险报告原则。”https://www.bis.org/publ/bcbs239.htm。Bennett, C. H. 1973. “计算的逻辑可逆性。”《IBM研究与开发杂志》17 (6): 525–32。https://doi.org/10.1147/rd.176.0525。Chen, Haitao, and Husheng Liao. 2011. “视图更新问题综述。”《国际计算机理论与工程杂志》3: 23–31。Ellerman, David. 2007. “复式记账：数学表述与推广。”SSRN电子期刊，12月。https://doi.org/10.2139/ssrn.1340619。Fong, Brendan, and David I. Spivak. 2021. “对称幺半预序。”麻省理工学院。Gajda, Michal Jan. 2020a. “敏捷数据管道：2020年的提取、转换、加载。”Data Science Europe。2020年。https://youtu.be/aHAc8ght9Gw。———。2020b。“使用Haskell敏捷生成云端 API 绑定。”Haskell.Love。2020年。
h
t
t
p
s://skillsmatter.com/skillscasts/14905-agile-functional-data-pipeline-in-haskell-a-case-study-of-multicloud-api-binding。———。2022。“迈向更完善的联合类型。”Ijiri, Yuji. 1975.《会计计量理论》。10。美国会计学会。———。2018。“传统会计计量的公理与结构。”《会计、经济学与法律：思想交流》8 (1): 20170057。https://doi.org/https://doi.org/10.1515/ae l-2017-0057。Julian, Mike. 2017. “实用监控。”Levitt, Michael. 1983. “天然蛋白质的分子动力学：I. 轨迹的计算机模拟。”《分子生物学杂志》168 (3): 595–617。https://doi.org/https://doi.org/ 10.1016/S0022-2836(83)80304-0。Ross, Kenneth A, and Yehoshua Sagiv. 1997. “演绎数据库中的单调聚合。”《计算机与系统科学杂志》54 (1): 79–97。https://doi.org/https://doi.org/10.100 6/jcss.1997.1453。Warsono, Sony, Muhammad Ridha, and Arif Darmawan. 2009. “会计数学：一个重大未解问题。”SSRN电子期刊，7月。https://doi.org/10.2139/ssrn.14390 84。Wlaschin, Scott. 2012. “F#乐趣与收益。”2012年。https://swlaschin.gitbooks.io/fsharpforfunandprofit/content/posts/recipe-part2.html。