arXiv:2304.06811v1 [cs.DB] 13 Apr 2023

SIGNAL - SAP Signavio 分析查询语言

Timotheus Kampik、Andre L¨ucke、J¨orn Horstmann、Mark Wheeler 和 David Eickhoff

SAP Signavio，柏林，德国{
timotheus.kampik,a.luecke,joern.horstmann,
mark.wheeler,david.eickhoff}@sap.com

摘要。 本文介绍并讨论了 SIGNAL，这是一种由 SAP Signavio 开发、面向大规模云端系统的工业级流程数据查询语言及引擎。SIGNAL 针对以事件日志格式存储的流程数据进行了优化，可实现快速读取，并为此采用内存列式存储。为便于使用，SIGNAL 采用类 SQL 语法，并加入了额外的领域特定查询功能，尤其是基于行模式匹配的时态运算符。此外，本文还着重探讨了与流程查询相关的研究挑战，这些挑战源自 SIGNAL 的实现与应用。

关键词： 业务流程管理 · 流程挖掘 · 查询语言

1 引言

业务流程管理的一项关键任务，是从流程数据和知识中推导出可执行洞见。为了加强技术对这项任务的支持，流程查询已成为学术研究和行业实践中的一个重要子领域 [13]。随着行业内业务流程管理与分析技术不断整合，能够应对大规模挑战的技术生态系统日趋成熟，人们也愈发关注如何将流程查询领域的学术知识体系与面向工程的现实视角相结合。为此，本文介绍并探讨 SIGNAL，即 SAP Signavio 分析查询语言。SAP 使用该语言进行流程分析，尤其是基于事件日志的流程挖掘。SIGNAL 既是一种查询语言，也是支撑该语言的查询引擎的名称。尽管 SIGNAL 的语法类似 SQL（实际上，它是 SQL 的一个子集，并扩展了对嵌套事件日志数据的处理能力），其底层引擎采用列式数据库，并针对云端多租户环境中的快速读取操作进行了优化。与将流程领域特有的时间和控制流功能提升到运算符层级的查询语言不同，SIGNAL 使用基于正则表达式的查询来匹配案例；这一做法类似于 SQL 标准 2016 年更新中引入的行模式匹配 [11]，也与流程查询语言 (PQL) [14,15]的某些特性颇为相似。这种设计确保了传统 SQL 功能与流程查询专用扩展之间实现关注点分离。本文将介绍 SIGNAL，重点讨论其设计动机和

设计原则（第 3 节）、架构（第 4 节），以及查询语言的关键特性及其与底层数据库的关系（第 5 节）。我们还将结合其他流程查询语言，并考虑更广泛的查询和数据库技术趋势，探讨 SIGNAL；随后重点指出我们认为与工业级流程查询语言和引擎相关的长期挑战（第 6 节），最后总结全文（第 7 节）。

2 流程查询

业务流程这一概念为管理组织内部及组织之间的工作开展方式提供了强大的抽象。为了促进业务流程管理，人们开发了各种用于流程设计、执行和分析的技术。这些技术的关键在于提取、转换和分析流程数据与知识，通常分别以事件日志 [1] （数据驱动）和流程模型 [20]（知识驱动）的形式呈现。在业务流程管理软件兴起之初，这些任务由通用数据库技术处理，通常借助当时流行的基于结构化查询语言（SQL）的工具，例如 MySQL[1]。然而，随着时间推移，流程查询技术日益复杂。这一发展得益于两种趋势：

1. 通常，人们开始质疑基于 SQL 的关系型数据库能否在大规模且通常基于云的分布式系统中处理海量数据。这最终促成了所谓的*NoSQL* 数据库技术的出现，这类技术放宽了传统约束条件 [6]。然而，从实际工程的角度来看，也可以认为，某些 NoSQL 技术的广泛采用及其带来的挑战，还凸显了一个事实：放弃 SQL 并非灵丹妙药。因此，工程师开始认真权衡在不同数据库系统之间进行选择，或在选择要实现的新数据库系统功能时必须考虑的各种取舍 [16]。

2. 具体而言，流程管理技术在实际应用规模和方法论成熟度方面都取得了发展。就前者而言，业务流程管理软件，尤其是数据密集型流程挖掘工具，如今已整合到大多数大型企业软件供应商的产品和应用生态系统中。就后者而言，流程管理软件的应用范围不断扩大，重点从建模和执行转向高级挖掘和分析方法，这些方法通常采用专用算法，例如 从数据中发现流程模型或进行（时态）符合性检查。

[1]例如，APROMORE 平台（至少在最初）基于底层 MySQL 数据库实现 [10]。不过需要指出，本文中一些以技术为中心的评估和论断基于隐性的工程知识，有些涉及专有技术，因此无法追溯到学术论文，甚至无法通过可靠可用的开源代码库加以验证。

第二个趋势在业务流程管理学术界体现为类似的转变：从流程建模转向流程挖掘，并发展流程查询方法，其重心也日益从前者转向后者。然而，研究人员通常关注的目标包括应用新的基础分析方法 [12]；可以说，现实世界中大规模流程查询的工程问题鲜少得到考虑[2]。一些学术界的流程数据查询方法旨在支持对“大数据”的处理（例如 [4]）。然而，这些方法的架构——例如围绕 SparkQL 和 Hadoop 等技术构建——并未反映出在基于云的系统中同时为数千个组织查询流程数据（尤其是每个事件日志包含数千万行的事件日志）的需求。本文通过介绍和探讨 SAP Signavio 的流程查询语言及引擎 SIGNAL，旨在进一步阐明这些实际问题。

3 目的与原则

SIGNAL 的主要目的是促进事件日志的分析[3]。具体而言，SIGNAL 查询构成了通过 SAP Signavio 流程挖掘软件开展的所有事件日志分析的核心抽象。例如，便于业务用户使用的流程分析“小部件”实际上是图形用户界面，可触发基于模板的 SIGNAL 查询实例化（见图 1）；第三方系统也可能使用 SIGNAL 查询，以无界面方式运用流程挖掘功能。

由于现代企业软件通常以即服务的形式交付，SIGNAL 的设计原则体现了对高可扩展性、多租户基于云的软件系统的工程与产品需求。具体而言，在查询语言和引擎的设计过程中，以下原则被优先考虑。

基于云。 引擎应设计为可在基于云的系统所使用的超大规模云服务商环境中运行。

多租户。 底层架构需要支持租户之间的严格隔离即不同客户组织的环境。

无服务器。 从用户的角度来看，查询语言和引擎需要无服务器 化（SIGNAL 用户包括数据工程师、开发人员和技术数据分析师）：用户可以专注于基于查询的推断这一核心任务，而资源分配、优化和执行等技术细节则由底层系统负责。

[2]一个值得注意的例外是，有一章书籍内容介绍了 Celonis 流程查询语言 [19]，我们会在讨论（第 6 节）中介绍它，而该章的描述并未涵盖本文所探讨的许多流程查询工程视角和通用数据库技术。[3]请回顾一下，事件日志是由事件组成的集合，传统上，每个事件至少包含一个事件 ID、一个案例 ID 和一个时间戳。事件日志是流程挖掘算法的经典数据输入，而最基本的事件日志格式（一组（事件 ID、案例 ID、时间戳）三元组）足以应用流程发现和符合性检查的核心算法。

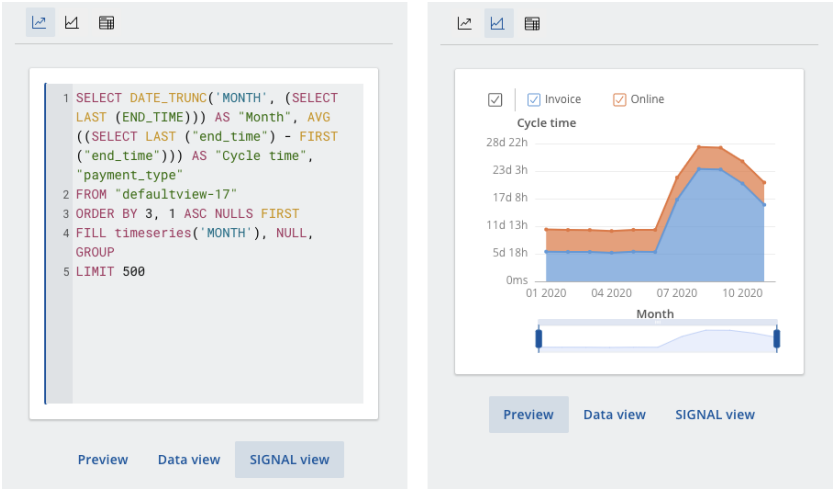


图 1：预览模式下的周期时间小组件及其底层 SIGNAL 查询。

以挖掘为核心。 该语言的主要用途是分析流程数据。因此，该语言应为典型的流程挖掘任务提供一流的抽象，例如筛选行为模式、符合性检查和流程绩效指标（PPI）分析；同时，它也必须支持常见的数据分析任务。

便于业务用户使用。 这种查询语言必须相对易于使用，适合仅具备基础编程技能的业务用户。

这些关键原则指导了 SIGNAL 的架构和功能设计，具体内容将在以下两节中介绍。

4 架构

SIGNAL 的架构由以下组件构成（另见图 2）。

REST API 和引擎接口。 SIGNAL 查询引擎接口通过 RESTful 应用程序编程接口（API）开放，用户（个人或系统）可通过该接口注册 SIGNAL 查询，并让引擎执行查询并返回结果。SIGNAL 集成的主要流程挖掘工具通过 REST API 使用 SIGNAL，将其作为生成流程挖掘分析的核心语言。用户既可以直接使用 SIGNAL（此时，工具会屏蔽当前流程范围等已知上下文），也可以在图形用户界面中使用所谓的 小部件，再由前端系统将其编译为 SIGNAL 查询。除了 REST API，数据摄取 API 还支持一次性上传数据，以及定期摄取流程数据更新。支持的格式包括可扩展事件流（XES）[3] 以及逗号分隔值文件和制表符分隔值文件（CSV 和 TSV）。为便于实施流程挖掘和

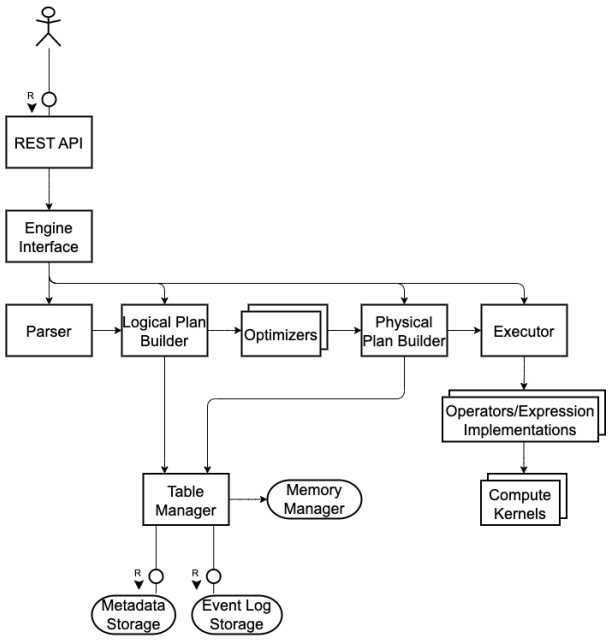


图 2：SIGNAL 引擎架构。

解决方案，SIGNAL 周边的技术生态系统还提供先进的数据接入工具。

解析器。 基于 SIGNAL 语言定义（见第 5 节），SIGNAL 查询会经过解析，生成的抽象语法树随后交给逻辑计划构建器。**逻辑计划构建器**。 逻辑计划构建器接收解析后的查询，执行类型检查，并优化查询执行计划。在逻辑层面，存在案例级与事件级之分。这两个层级均支持布尔值、数值、字符串、时间戳和持续时间类型。

优化器。 工作负载会被动态调度和优化。在此过程中，多个优化步骤会利用预排序的事件日志存储（事件的全序）。例如，去重计数和窗口函数/分区就会采用这种方式。为了优化交互性能，LIMIT 运算符会尽早缩减待处理数据的规模。这对于通过基于浏览器的用户界面分析流程数据尤其有帮助，因为按照设计，探索性查询通常只针对数据的有限子集执行。

物理计划构建器。 优化后，逻辑计划会转换为物理执行计划。转换过程中会创建所需表列的快照。物理计划是由一系列算子构成的树或列表，并在单独的线程中执行，以隔离意外行为可能造成的负面性能影响。

包含算子和表达式实现以及计算内核的执行器。
SIGNAL 支持基于 SQL 的数据分析常用的算子和函数，还提供用于分析（时序）控制流的其他算子。用于关系代数操作的底层计算内核经过优化，可实现高吞吐量处理，并直接作用于列式数据。

表管理器和内存管理器。 内存和表管理由开源框架提供支持，这些框架采用列式的数据管理方法来实现语言无关的内存分析，并允许将案例嵌套到事件日志表中。

元数据存储和事件日志存储。 计算和存储基础设施由不同的超大规模云服务商提供。元数据存储在传统的关系数据库中，而事件日志则以 Apache Arrow 列式格式单独存储[4]，无需执行复杂联接，因此可以快速读取。

请注意，我们有意将架构描述得较为抽象，因为实现细节（例如特定层所用的确切技术）容易发生变化；而且在某些情况下，不同实例可能运行在略有差异的技术栈上。

SIGNAL 的一个关键特点是采用列式方法进行数据持久化。如表 1 所示，SIGNAL 将事件日志表示为案例表，其中每个案例都包含一个嵌套表，存储构成该案例的事件。每个案例都必须具有案例 ID 属性，每个事件都必须具有事件名称和结束时间属性，并且可以具有可选的开始时间属性[5]。案例和事件都可以包含多个其他属性（通常与特定领域相关）。每个案例中的事件按其结束时间戳排序。这有助于执行快速读取操作。

5 查询语言

SIGNAL 语言旨在使用基于 SQL 的语法查询以上所述以列式格式表示的事件日志（图 1），重点支持读操作。此处，被查询的表通常是固定的，即只需提供相应已提取事件日志的标识符指针即可；在许多场景中，这由集成系统处理，例如流程分析前端应用程序。除了用于数据提取和聚合的标准 SQL 运算符外，SIGNAL 还提供一种行模式匹配方法来管理逻辑时间（顺序），即识别其行为（事件）符合指定时序约束条件的案例。

[4]https://arrow.apache.org/docs/format/Columnar.html，访问日期：2022年1月30日。
[5]在示例中，我们仅使用结束时间，这反映了事件日志在现实世界中的典型属性：在大多数情况下，一个事件恰好有一个时间戳，表示该事件在系统中登记的时间。

表 1：用于表示事件日志的列式模型。

案例 ID	客户 ID	最终状态	事件		
			事件名称	结束时间	状态
1001	C2001	done	打开工单	1675086864052	none
			分配工单	1675160180724	open
			关闭工单	1675220315296	done
1002	C2002	已阻塞	事件名称	结束时间	状态
			打开工单	1675147138009	none
			分配工单	1675213914098	open
			关闭工单	1675282027657	已阻塞
			打开工单	1675414104525	已阻塞

在详细介绍使用 SIGNAL 进行时序模式匹配之前，我们先说明一些关于案例级和事件级可用信息的假设，尤其是时间戳的可用性。最基本的案例定义包括案例标识符、按顺序排列的事件名称列表、事件开始时间戳（缺失时为空）以及事件结束时间戳。这意味着，事件日志中事件的开始时间戳构成事件上的偏序，而结束时间戳则保证构成全预序。不过，由于在事件日志级别会强制采用序列，并在时间戳不唯一时打破平局，因此我们可以分别假设它们构成偏序和全序。实际中，事件开始时间戳通常缺失，因为活动的执行时间往往没有记录；企业系统是事件日志的最终来源，而在这些系统中，通常只有活动完成时才会创建事件。例如，创建采购订单时通常没有开始时间；只有提交表单才会触发事件。事实上，即使打开表单时记录了开始时间戳，也可能产生误导：毕竟，用户可能在打开表单之前就花了大量时间准备采购订单。如果用户希望只依赖经典 SQL（不使用行模式匹配，但使用一些用于处理事件间时间关系的基本附加算子），则可以将嵌套在事件日志表中的案例表展平。

图 3 展示了一个嵌套 SIGNAL 查询，该查询从持久化在 SIGNAL 数据库中的事件日志获取某个流程的平均周期时间，并直观呈现结果的计算方式。请注意，这里没有使用行模式匹配；相反，该查询依赖 FIRST 和 LAST 算子。事件级子查询（SELECT LAST(end time) - FIRST(end time)）可以看作隐含了一个 FROM events 子句。这意味着，从逻辑上讲，聚合中的表达式作用于嵌套 EVENTS 表中的各行。FIRST 和 LAST 是聚合函数（与 MIN 或 MAX 一样），可将多个输入归约为单个值。事件级子查询的结果必须是单个值；我们可以通过要求子查询中包含聚合函数来确保这一点。从类型检查的角度来看，事件级子查询很重要。在其外部，涉及 end time 的表达式会引用多个值，这是不允许的。

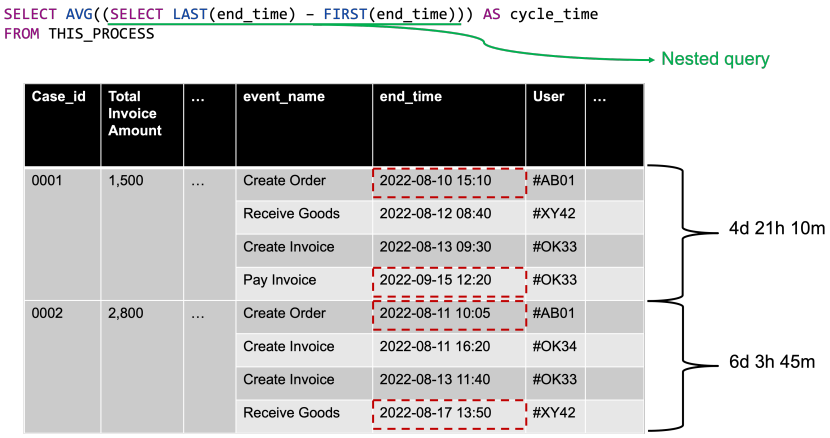


图 3：使用 SIGNAL 查询嵌套表：平均周期时间。所选周期时间（随后据此计算平均值）显示在右侧。

可以通过案例的简单基于图的模型，以半形式化的方式展示行模式匹配方法的行为。我们将一个案例 $C = (E,\ R)$ 定义为由事件 E 和 $R \subseteq E \times E$构成的有限集合，其中我们用 $(a,\ b) \in R$ 表示 $a \succeq b$，用 $(a,\ b) \in R,\ (b,\ a) \notin R$ 表示 $a \succ b$，并规定 R 是 E 上的全序[6]，该全序可以根据案例事件的结束时间戳推断得出。为了检查案例是否满足时间约束，我们引入一个有限的行为约束集合 B，以及一个行为匹配函数（偏函数）$f\colon B \to 2^E$，它将行为映射到事件集合。

下面简要介绍 SIGNAL 语言：先提供一些总体直觉，再更准确地定义行模式匹配方法。例如，在客户支持请求处理流程中，我们可能希望定义这样一些案例：工单在关闭后又被重新打开。借助 SIGNAL，我们可以如下查询这种行为（并对所提供的事件名称作出一些假设）。

```
SELECT 案例_id
来自 此处_流程
WHERE 事件_匹配 ('关闭工单' ~>'打开工单')
```

请注意，当前流程是一个依赖上下文的指针，指向当前作用域中的流程；例如，在业务用户通过图形用户界面执行可视化数据分析的上下文中。

[6] 请记住，由于 R 是全序，因此 $\forall a,\ b,\ c \in R$ 有 $a \succeq a$（自反性）；如果 $a \succeq b$ 且 $b \succeq c$，则有 $a \succeq c$（传递性）；如果 $a \succeq b$ 且 $b \succeq a$，则有 $a = b$；此外还有 $a \succeq b$ 或 $b \succeq a$。请注意，即使时间戳不唯一，我们使用的也是全序，而非全预序，因为事件日志中的事件序列按时间排序后，其顺序便已“固定”；对于不唯一的时间戳，则会打破平局。

MATCHES 子句的工作方式如下：

– 匹配的范围是一个案例（即一组事件），而不是单个事件，正如我们的行为匹配函数 f 所规定的那样。– 我们的行为 由事件名称（列名）隐式定义，然后为其分配 '关闭工单' 和 '打开工单' 这两个值。也可以通过选择满足指定约束条件的事件来显式定义行为，具体示例见下文。根据隐式定义，所有事件名称分别为 '关闭工单' 和 '打开工单' 的事件都会被匹配。– 然后，时间模式匹配运算符 $\sim>$ （直接或间接跟随）[7] 会匹配符合指定时间关系的行为发生的案例。在此示例中，所有事件名称为 ='关闭工单' 的事件被事件名称为

-

- e

= '打开工单' 的事件直接或间接跟随。

此外，我们希望识别这样的案例：工单已关闭，但其状态被标记为“已阻塞”（而不是通常应在工单关闭时设置的预期状态“已完成”）。借助 SIGNAL，我们可以按如下方式查询此行为（同时对所提供的属性作出一些额外假设）。

```
SELECT 案例 ID_FROM当前流程_WHERE (事件_名称 = '关闭工单'AND"状态"
 = '已阻塞')
```

最后，我们来识别这样一些案例：工单先被直接关闭，然后（最终）又重新打开。这里，我们使用了行为这一概念。请注意，可以使用任意表达式（返回布尔值的表达式）来定义行为。

```
SELECT 案例 ID_FROM当前流程_WHERE 行为 (事件_名称 = '关闭工单
'AND"状态" = '已阻塞') ) 作为阻塞时已关闭_ _匹配(阻塞时已关闭
_时_已阻塞 ~> '打开工单')
```

在我们的框架中，我们可以将后一个查询建模如下：

– 行为集合为 $B = \{x, y\}$，其中 x 表示已阻塞状态下关闭，y 表示活动名称为“打开工单”的事件。– 我们的案例中的事件为 $E = \{e_0, ..., e_n\}$，并假设它们按索引顺序构成全序，即即，我们有一个图 $C = (E, R)$，其中 $R = \{(e_i, e_j) | 0 \leq i \leq j \leq n\}$（这意味着 (E, R) 等价于序列 $\langle e_0, ..., e_n \rangle$）。– 恰好有一个事件 $e \in E$，使得 e 是一个在状态为 “ 已阻塞 ” 时关闭的工单；恰好有两个事件 e', $e'' \in E$ 的活动名称为“打开工单”： $f = \{(x, \{e\}), (y, \{e', e''\})\}$。假设 $e' \succ e$ 和 $e \succ e''$ 成立。

[7] ~>下文将对此作更精确的解释。

– 匹配运算符 $\sim>$（直接或间接跟随）随后被转换为语句 $\exists c \in f(x)\ d \in f(y)$: $c \succ d$。情况确实如此： $e \in f(x)\ e'' \in f(y)\ e \succ e''$。因此，我们的查询与该案例匹配。

请注意，对于任意算子，匹配都是存在性的（而非全称性的），即，只需匹配到一个行为即可，如上所示。

我们基于已有的成熟研究提出，对于管理有限轨迹，使用单个二元关系（R）足以表达有限轨迹的线性时序逻辑 [8]。下文中，我们假设一个案例 $C = (E, R)$，其中有 $a, b \in E$ 以及一个行为集合B，其中包含行为 $a', b' \in B$，使得 a' 和 b' 匹配唯一事件，即，存在一个行为匹配函数 f，使得 $f(a') = \{a\}$ 和 $f(b') = \{b\}$ 成立。

可使用以下行模式匹配运算符。

直接跟随（$->$）。匹配该行为的两个事件彼此直接跟随。$a' -> b'$ 匹配我们的案例，当且仅当 $a \succ b$ 和 $\nexists c \in E$ 使得 $c \neq a$、$c \neq b$、$a \succ c$和 $c \succ b$成立。间接或直接跟随（$\sim>$）。匹配该行为的两个事件彼此直接或间接跟随。如上所示，$a' \sim> b'$ 匹配我们的案例，当且仅当$a \succ b$成立。以.....开始（^）。匹配该行为的事件是案例中的第一个事件。^ a'匹配我们的案例，当且仅当 $\forall c \in E \setminus \{a\}$ 满足 $a \succ c$。以.....结束（$）。匹配该行为的事件是案例中的最后一个事件。$a'$ $ 匹配我们的案例，当且仅当 $\forall c \in E \setminus \{a\}$ 满足 $c \succ a$。包含任意项（ANY）。指定必须发生任意事件。a' ANY b' 匹配我们的案例，当且仅当 $\exists c \in E$ 满足：对于某个 $c' \in B$，$c \in f(c')$成立，且 $a' -> c'$ 和 $c' -> b'$。不包含（NOT）。对行为匹配取反。例如，a' NOT b' 匹配我们的案例，当且仅当 $\exists c \in E$ 满足 $a \succ c$，并且 $\nexists d \in E$ 满足 $d \neq a$、$d \neq c$、$a \succ d$和$d \succ c$，且 $c \notin f(b')$。或（|）。两种行为中的任意一种发生。例如，^ $(a'|b')$匹配我们的案例，当且仅当 ^ a'或 ^ b' 匹配我们的案例。重复（*）。指定某个行为发生零次或多次。例如，a'ANY* b' 匹配我们的案例，当且仅当 $a' \sim> b'$。

需要强调的是，全称量化可以通过否定实现。例如，如果我们想表达行为 a' 应始终直接跟随行为b'，可以用表达式 (^ (NOT a' | (a' b'))* $) 来规定；当且仅当事件全序所规定的整个序列由任何非 a' 的内容，或 a' 后跟 b' 构成时，该表达式才匹配。即，当且仅当 a' 出现时，它总是直接跟随 b'。表 2 展示了模式匹配运算符的工作方式，假设事件再次按全序（序列）排列，并采用上例中的行为和行为匹配函数。

图 4 展示了一个查询，其中使用正则表达式进行行模式匹配，以指定控制流约束。具体而言，对于特定案例，活动“创建订单”必须首先出现在按时间戳排序的事件序列中，随后（直接或间接）是“创建订单”，而后者又必须随后（直接或间接）是“创建发票”。

模式语法	匹配序列
$a' -> b'$	$\langle ..., a, b, ... \rangle$
$a' \sim> b'$	$\langle ..., a, ..., b, ... \rangle$
$\hat{} \ a'$	$\langle a, ... \rangle$
a' \$	$\langle ..., a \rangle$
a' 任意 b'	$\langle ..., a, c, ^{b,\ \ ...} \rangle$ （给定某个事件 $c \in E$）
a' 非 b'	$\langle ..., a, c... \rangle$ (给定某个事件 $c \in E$： $c \notin f(b')$)
$\hat{} \ (a'\|b')$	$\langle c, ... \rangle$ (给定 $c \in \{a, b\}$)
a' 任意* b'	$\langle ..., a, ..., b, ... \rangle$

表 2：模式匹配运算符及其匹配的序列（全序）。

6 讨论

据我们所知，SIGNAL 是学术文献中介绍的第二种商业级、行业规模的流程查询语言；此前介绍的唯一一种语言是 Celonis 流程查询语言（CPQL）[19]。SIGNAL 与 CPQL 之间存在一些相似之处。显然，SIGNAL 和 CPQL 覆盖了相似的使用场景，重点支持大规模流程挖掘。此外，两种语言都使用内存数据库，并注重快速读取操作以及用于管理逻辑时间的自定义运算符。然而，与 CPQL 不同，SIGNAL 采用列式方法存储数据，以实现快速读取访问（据报道，CPQL 依赖雪花模式），并且 SIGNAL 语法明确区分传统 SQL 运算符与用于事件日志分析的自定义时态运算符（后者通过行模式匹配处理）。此外，据报道，CPQL“受 SQL 启发”，而 SIGNAL 可以说更接近 SQL 模型，通过引入嵌套事件表这一概念对其进行扩展。在该嵌套表上，SIGNAL 可以执行与案例级别相同的 SQL 操作和聚合。相比之下，CPQL 需要使用特殊语法或函数，将例如从活动表聚合的值提取到案例级别。SIGNAL 与 CPQL 之间一个尤其值得注意的区别在于，据报道，CPQL 依赖基于佩特里网的表示来进行流程符合性检查；而 SIGNAL 则采取基于原则的立场，认为其基于行模式匹配的时态推断是进行时态推理的唯一一等抽象，并假设这将带来更简单、更易维护、最终性能更高且更便于用户使用的语言和查询引擎。据我们所知，在撰写本文时，与学术界的流程查询语言和引擎相比，SIGNAL 还因采用列式数据库以及带有行模式匹配扩展的基于 SQL 的语法来进行管理而独树一帜

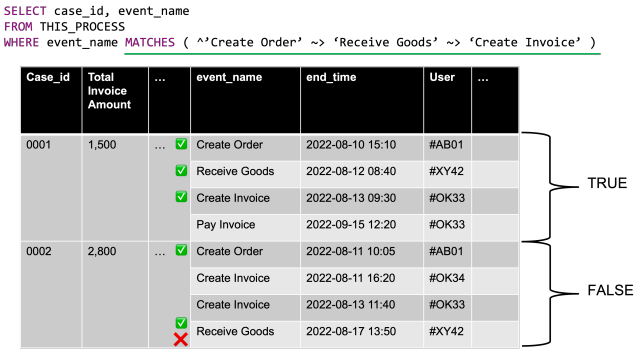


图 4：使用 SIGNAL 应用控制流约束：右侧括号中的标签表示哪些案例符合这些约束条件。

逻辑时间。这些关键设计决策可能会对学术研究产生有趣的影响：在 SAP Signavio 流程挖掘生态系统中，所有事件日志分析工具和手段（如图形用户界面小部件）最终都会被编译为 SIGNAL 查询。在学术界以及更广泛的开放技术生态系统中，类似的方法若聚焦于一种可能标准化的通用语，或有助于进一步汇聚流程挖掘领域的力量，并推动长期创新，尤其是考虑到其他数据库技术（如 Apache Flink） [5]至少部分源自学术界。

基于 SIGNAL 的实现经验以及构建解决方案的实践，我们确定了现代、工业级流程查询语言和引擎面临的以下趋势与挑战。

支持数据转换。 流程挖掘的一项关键挑战是从企业软件数据库中提取事件日志 [9]。通常，ETL 流程复杂且需要反复迭代；在开展基于流程挖掘的分析之前，就需要深入了解业务流程、涉及的信息技术系统及其生成的数据。流程查询语言可能在 ETL 流程中就发挥作用，例如用于融合不同的数据源（如上一节所述）。为了进一步探索这一方向，未来的研究可以调查流程挖掘应用场景中数据融合面临的挑战。

无需事件日志的流程数据分析。更进一步，理想情况下，可以直接从一个或多个 IT 系统的数据库中查询流程数据。在许多场景下，这种做法并不现实，例如，底层数据库的性能不足，或者要将数据库中所表示的流程数据转换为便于分析的表示形式，所需的领域知识使中间转换不可避免。不过，在许多场景下，数据转换这一挑战无论如何都需要深入的源系统数据库专业知识，而且还可能通过在企业数据上持久化适用于流程挖掘的视图来弥补性能瓶颈

因此，未来流程数据查询有望进一步接近源系统。

多个视图，取决于对案例 ID 的不同定义。从实际角度来看，传统的扁平事件日志仅提供了业务流程的一种特定视图，并且取决于按照特定案例 ID 定义对流程进行范围界定。例如，订单处理流程可以使用订单标识、订单项标识或货运标识作为案例 ID，由此生成的事件日志能够从截然不同的角度呈现订单处理方式，关注粒度不同、但不一定存在层级关系的项目（例如，一个订单可能拆分为多批货物发运，而一批货物也可能包含多个订单中的项目）。在学术研究中，这一挑战通过引入以对象为中心的流程挖掘来应对 [2], ，即在事件日志中加入面向对象的抽象（此时事件日志不再是简单的表格）。不过，也可以认为，以对象为中心的流程挖掘只是将更多结构从通常用于提取事件日志的关系数据库转移到了事件日志本身。另一种可能的做法是在 ETL 流程中提供动态流程范围界定支持，并在流程查询语言中实现一流抽象，以便针对同一组流程数据探索多个范围（案例 ID 定义）。例如，在订单到交付流程中，可以探索订单、交付，甚至某个特定采购项的历程范围。

挖掘与建模的集成。SIGNAL完全专注于查询事件日志（流程数据），尽管在其产品应用场景中，流程模型（即流程模型的表示形式）通常是可用的。SIGNAL也可能用于查询流程模型。不过，与事件日志或其集合相比，流程模型集合通常规模相对较小，而且查询时，模型中活动的时间顺序（例如：“返回所有创建采购订单后紧接着执行审批步骤的流程模型”）很少受到关注；即，尽管我们认为流程模型查询能够产生重大的业务影响，例如在变体管理和最佳实践推荐方面，但我们假设在此处，精确查询“时间性”控制流的重要性较低。

面向组织挖掘和经验挖掘的一等抽象。归根结底，流程管理关注的是组织的管理。然而，在典型的流程查询语言中，关键的组织视角，尤其是角色和资源，并未被视为一等抽象。尽管如此，已有大量研究致力于组织挖掘 [17][8]，即从事件日志中提取和分析组织信息。近来，智能体系统挖掘 [18]领域出现了首个方法，该方法将执行特定活动的组织资源作为一等抽象，重点加以研究。尤其是后者提出了这样一个问题：流程查询语言在多大程度上能够处理特定行动者所特有的组织视角，即身处其中的人类或人工智能体的经验（或信念）。例如，当分析来自多个信息技术系统的流程数据，而这些系统对工作流或工作状态的表示彼此不一致时，便可以通过查询观察到这种不一致，并据此推断出使用其中某个信息技术系统的行动者持有错误信念，而这可能导致错误决策；

[8] 最初称为组织挖掘，但我们认为，挖掘的并非组织层面的事物；相反，被挖掘的对象是组织。

从基础理论的角度来看，此类分析可借助非经典逻辑的工具，例如采用次协调逻辑或非单调逻辑。

可解释推理。 流程查询可视为一种符号推理方法（基于数据或知识）。符号推理的一个近期趋势是生成（通常简洁，即在某种意义上尽可能精简的）解释：能够令人信服地概述某项推理为何得出的查询或查询结果，例如运用必要性、充分性或反事实性的概念，并且理想情况下可转换为人类可理解的形式。借鉴丰富的推理可解释性研究（综述见 [7] ），有望推动流程查询应用于业务流程智能软件的关键应用领域，例如根因分析和流程变更建议生成。

跨事件日志、跨组织和跨流程挖掘。 与数据融合和流程范围界定方面的挑战类似，流程查询语言理想情况下应支持同时从多个事件日志中进行联合推理。传统上，人们假定需要联合分析的事件日志中的实例彼此相关。然而，跨组织联合分析同一流程的多个实现也是一个很有前景的应用场景，尤其是因为这能大幅减少各个组织的分析工作量，同时有助于获得准确且可采取行动的分析结果。

7 结论

本文介绍了一种工业级流程查询语言及其引擎。我们希望本文能让读者更深入地了解用于事件日志分析的工业级流程查询方法，并为该领域的未来研究提供参考。尤其是，我们认为，兼顾实际约束条件与挑战的学术界和产业界方法若能相互融合，将推动流程查询领域的创新，理想情况下，甚至能产生更广泛适用于通用数据库技术的见解。

致谢 我们感谢众多参与 SIGNAL 开发及使用工作的工程师和分析师。我们还要感谢 Artem Polyvyanyy、Adrian Rebmann、Han vander Aa 和MatthiasWeidlich，感谢他们深入的讨论与反馈。

参考资料

1. van der Aalst, W.M.P.: 《流程挖掘——行动中的数据科学》，第二版。Springer（2016年）。https://doi.org/10.1007/978-3-662-49851-4，https://doi.org/10. 1007/978-3-662-49851-4

2. van der Aalst, W.M.P.：以对象为中心的流程挖掘：处理事件数据中的分歧与汇合。载于： ¨Olveczky, P.C., Sala¨un, G.（编）《软件工程与形式化方法——第17届国际会议，SEFM 2019》，挪威奥斯陆，2019年9月18日至20日，会议论文集。《计算机科学讲义》，第11724卷，第3–25页。Springer（2019年）。 https://doi.org/10.1007/978-3-030-30446-1 1，https://doi.org/10. 1007/978-3-030-30446-1_13. Acampora, G., Vitiello, A., Stefano, B.N.D., van der Aalst, W.M.P., G¨unther, C.W., Ver-beek, E.：IEEE 1849：XES标准：由IEEE计算智能学会赞助的第二项IEEE标准 [学会简报]。IEEE《计算智能杂志》**12**(2)，4–8（2017年）。 https://doi.org/10.1109/MCI.2017.2670420，https://doi.org/10.1109/ MCI.2017.2670420 4. Beheshti, A., Benatallah, B., Motahari-Nezhad, H.R., Ghodratnama, S., Amouz-gar, F.：BP-SPARQL：一种用于汇总和分析大型流程数据的查询语言。载于：Polyvyanyy, A.（编）《流程查询方法》，第21–48页。Springer（ 2022年）。https://doi.org/10.1007/978-3-030-92875-9 2，https://doi.org/10.1007/ 978-3-030-92875-9 2_5. Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., Tzoumas, K.：Apache Flink™：在单一引擎中实现流处理与批处理。IEEE《数据工程公报》**38**(4)，28–38（2015年），http://sites.computer.org/debull/A15dec/p28.pdf 6. Corbellini, A., Mateos, C., Zunino, A., Godoy, D., Schiaffino, S.：持久化大数据：NoSQL概览。《信息系统》**63**，1–23（2017年）。 https://doi.org/https://doi.org/10.1016/j.is.2016.07.009，https://www. sciencedirect.com/science/article/pii/S0306437916303210 7. Cyras, K., Badrinath, R., Mohalik, S.K., Mujumdar, A., Nikou, A., Previti, A., Sundararajan, V., Feljan, A.V.：机器推理可解释性。CoRR **abs/2009.00418**（2020年），https: // arxiv.org/abs/2009.00418 8. Giacomo, G.D., Vardi, M.Y.：有限轨迹上的线性时序逻辑与线性动态逻辑。载于：Rossi, F.（编）IJCAI 2013，第23届人工智能国际联合会议论文集，中国北京，2013年8月3日至9日。第854–860页。IJCAI/AAAI（2013年），http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/ paper/view/6997 9. Kampik, T., Weske, M.：事件日志生成：行业视角。载于：Augusto, A., Gill, A., Bork, D., Nurcan, S., Reinhartz-Berger, I., Schmidt, R.（编）《企业、业务流程与信息系统建模——第23届国际会议BPMDS 2022和第27届国际会议EMMSAD 2022，CAiSE 2022期间举行》，比利时鲁汶，2022年6月6日至7日，会议论文集。《商业信息处理讲义》，第450卷，第123–136页。Springer（2022年）。https://doi.org/10.1007/978-3-031-07475-2 9， https://doi. org/10.1007/978-3-031-07475-2_910. La Rosa, M., Reijers, H.A., van der Aalst, W.M., Dijkman, R.M., Mendling, J., Dumas, M., Garc´ıa-Ba˜nuelos, L.：APROMORE：先进的流程模型存储库。《应用专家系统》**38**(6)，7029–7040（2011年） 11. Michels, J., Hare, K., Kulkarni, K., Zuzarte, C., Liu, Z.H., Hammerschmidt, B., Zemke, F.：新版改进型SQL：2016标准。SIGMOD Rec.**47**(2)，51–60（2018年12月）。 https://doi.org/10.1145/3299887.3299897，https://doi.org/10.1145/3299887. 3299897 12. de Murillas, E.G.L., Reijers, H.A., van der Aalst, W.M.P.：数据感知型流程导向查询语言。载于：Polyvyanyy, A.（编）《流程查询方法》，第49–83页。Springer（2022年）。 https://doi.org/10.1007/978-3-030-92875-9 3，https://doi.org/10.1007/ 978-3-030-92875-9 3_

13. Polyvyanyy, A.（编）：流程查询方法。Springer（2022）。
https://doi.org/10.1007/978-3-030-92875-9, https://doi.org/10.1007/ 978-3-030-92875-9
14. Polyvyanyy, A., Corno, L., Conforti, R., Raboczi, S., Rosa, M.L., Fortino, G.：在APROMORE 中进行流程查询。载于：Daniel, F., Zugal, S.（编）《第13届国际业务流程管理会议（BPM 2015）同期举行的 BPM 2015演示会议论文集》，奥地利因斯布鲁克，2015年9月2日。CEUR研讨会论文集，第1418卷，第105–109页。CEUR-WS.org（2015），http://ceur-ws.org/Vol-1418/paper22. pdf 15. Polyvyanyy, A., Pika, A., ter Hofstede, A.H.M.：面向合规性、重用和标准化的基于场景的流程查询。信息系统 **93**，101563（2020）。https://doi.org/10.1016/j.is.2020.101563, https://doi.org/10.1016/j.is.2020.101563 16. Roy-Hubara, N., Shoval, P., Sturm, A.：为多语言持久化应用选择数据库。数据与知识工程 **137**，101950（2022）。
https://doi.org/https://doi.org/10.1016/j.datak.2021.101950, https://www.sciencedirect.com/science/article/pii/S0169023X21000744 17. Song, M., van der Aalst, W.M.P.：迈向全面支持组织挖掘。决策支持系统 **46**(1)，300–317（2008）。
https://doi.org/10.1016/j.dss.2008.07.002, https://doi.org/10 002 18 1016 2008 07 . / j.dss. . . . Tour, A., Polyvyanyy, A., Kalenkova, A.A.：智能体系统挖掘：愿景、优势与挑战。IEEE Access **9**，99480–99494（2021）。
https://doi.org/10.1109/ACCESS.2021.3095464, https://doi.org/10.1109/ACCESS.2021.3095464 19. Vogelgesang, T., Ambrosy, J., Becher, D., Seilbeck, R., Geyer-Klingeberg, J., Klenk, M.：Celonis PQL：一种用于流程挖掘的查询语言。载于：Polyvyanyy, A.（编）《流程查询方法》，第377–408页。Springer（2022）。
https://doi.org/10.1007/978-3-030-92875-9 13, https://doi.org/ 10 1007 978 3 030 92875 9_13./----. Weske, M.：
20《业务流程管理——概念、语言与架构》第三版。Springer（2019）。
https://doi.org/10.1007/978-3-662-59432-2, https://doi.org/ 10.1007/978-3-662-59432-2