University Library



A gateway to Melbourne's research publications

Minerva Access is the Institutional Repository of The University of Melbourne

Author/s:
Polyvyanyy, A

Title:
Process Querying: Methods, Techniques, and Applications

Date:
2022

Citation:
Polyvyanyy, A. (2022). Process Querying: Methods, Techniques, and Applications. Polyvyanyy, A (Ed.). Process Querying Methods, (1), pp.511-524. Springer International Publishing.

Persistent Link:
https://hdl.handle.net/11343/312577

Process Querying: Methods, Techniques, and Applications

Artem Polyvyanyy

Abstract Process querying studies concepts and methods from fields like big data, process modeling and analysis, business process intelligence, and process analytics and applies them to retrieve and manipulate real-world and designed processes. This chapter reviews state-of-the-art methods for process querying, summarizes techniques used to implement process querying methods, discusses typical applications of process querying, and identifies research gaps and suggests directions for future research in process querying.

1 Introduction

Process querying aims to combine concepts studied by disciplines that look into the use of large and complex data sets, like big data, and research areas that investigate aspects related to process modeling and analysis, like business process management, process mining, business process intelligence, and process analytics to develop methods and tools for automatically manipulating, e.g., redesigning and optimizing, real-world and designed processes, and systematically extracting process-related information for subsequent use [28]. A *process querying method* is a technique that given a collection of processes and a process query, i.e., a statement that articulates a process-related information need or specifies an instruction for process manipulations, systematically implements the query over the processes.

The idea of process querying started to shape at the beginning of the twenty-first century. However, the lion's share of concepts, principles, and methods for process

Artem Polyvyanyy
School of Computing and Information Systems
Faculty of Engineering and Information Technology
The University of Melbourne
Victoria 3010, Australia
e-mail: artem.polyvyanyy@unimelb.edu.au

querying appeared only recently. The reason for this is at least twofold. First, recent large-scale digitization of real-world processes governed by organizations has led to the generation of large volumes of digital footprints of real-world processes and supporting data. Second, increasing evidence of added value due to the use of data generated by processes [32], as well as of information about the processes in forms of models and ontologies, for improving future real-world processes has spawn research on new analysis techniques aimed to understand and interpret process phenomena, giving the birth to the discipline of Process Science.

This chapter gives an overview of the state-of-the-art methods for process querying. The methods are classified into those for querying observed and recorded processes, i.e., event logs, as studied in process mining [1], process models, either automatically discovered from event logs using process mining algorithms, or manually designed, as traditionally studied in business process management [10, 42], and such that address the querying of both logs and models. Besides, the chapter discusses techniques that underpin the existing process querying methods and practical applications of process querying methods as recommended and employed by academia and industry. The content of this chapter is based on the literature reviews reported in [24, 28] and extends them by further insights reported by the contributors of this book.

The rest of this chapter unfolds as follows. The next section provides the foundations necessary for following the subsequent discussions. Section 3 discusses the existing methods for process querying. Next, Section 4 gives an overview of techniques that are commonly used to implement process querying methods. Section 5 summarizes the main applications of process querying, as recommended or evaluated by the authors of the process querying methods. Finally, Section 6 summarizes research gaps and suggests directions for future research in process querying.

2 Foundations

This section lists basic concepts in process querying that are necessary to support the subsequent discussions. Let $\mathcal{U}_{an}$ and $\mathcal{U}_{av}$ be the universe of *attribute names* and *attribute values*, respectively.

Event. An *event* e is a relation between some attribute names and attribute values with the property that each attribute name that participates in the relation is related to exactly one attribute value, i.e., it is a partial function $e : \mathcal{U}_{an} \nrightarrow \mathcal{U}_{av}$.

Set $e = \{(case, 120327), (time, 2020\text{-}02\text{-}03T00{:}27{:}29Z), (act, \text{“Open claim”})\}$ is an example event with three attributes. A possible interpretation of these attributes is that event e belongs to the process with case identifier $e(case) = 120327$, was recorded at timestamp $e(time) = 2019\text{-}10\text{-}22T11{:}37{:}21Z$ (ISO 8601), and was generated by the activity with name $e(act) =$ “Open claim”.

Process. A *process* π is a partially ordered set $(E, \leq)$, where E is a set of events and $\leq$ is a partial order over E.

Trace. A *trace* τ is a process $(E, <)$, where $<$ is a total order over E.

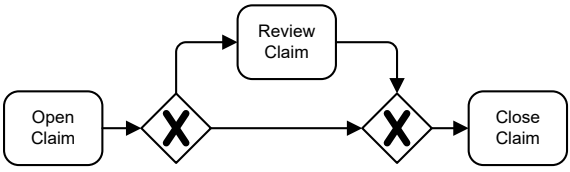


Fig. 1: A process model.

A trace is often specified as a sequence of events with the same case identifier ordered by timestamps in ascending order. Hence, a trace groups events that were generated by the same process instance. For example, $\langle e_1, e_2, e_3 \rangle$ is a trace composed of three events. In a process, in general, some events may be unordered to reflect that they are independent and, thus, can be, or were already, executed simultaneously.

Behavior. A *behavior b* is a multiset of processes.

The fact that a behavior is a multiset of processes can be used when describing that a process was observed and recorded several times. Behavior $b_1 = [\tau_1^9, \tau_2^3]$, where $\tau_1 = \langle \{(act, \text{“Open claim”})\}, \{(act, \text{“Review claim”})\}, \{(act, \text{“Close claim”})\} \rangle$ and $\tau_2 = \langle \{(act, \text{“Open claim”})\}, \{(act, \text{“Close claim”})\} \rangle$, specifies that a claim is opened, then reviewed, and finally closed nine times, and is opened and then immediately closed three times.

Behavior model. A *behavior model* is a simplified representation of real-world or envisioned behaviors, and relationships between the behaviors, that serves a particular purpose for a target audience.

The model in Figure 1 is a model of an envisioned behavior captured as a BPMN diagram. According to the semantics of BPMN, the model specifies behavior composed of traces τ_1 and τ_2, whereas behavior b_1 is a possible description of the observed behavior induced by the executions of the diagram. In [28], we suggest several classes of behavior models. According to this classification, the diagram in Figure 1 identifies a *process model*, while behavior b_1 identifies an *event log*. In general, a process model describes a potentially infinite collection of envisioned processes, whereas an event log captures a finite collection of already executed processes. The reader can refer to [28] for further details on the different classes of behavior models.

Process repository. A *process repository* is an organized collection of behavior models.

For example, a process repository can be composed of process models and/or logs organized in a folder hierarchy.

Process query. A *process query* is a statement that describes an information need in a process repository or specifies an instruction to manipulate a process repository.

A sample process query q can specify an instruction to retrieve all process models from a repository that describe behaviors with processes in which activity “Review claim” occurs.

Let $\mathscr{U}_{pr}$ and $\mathscr{U}_{pq}$ be the universe of *process repositories* and the universe of *process queries*, respectively.

Process querying method. A *process querying method m* is a relation between ordered pairs, where in each pair the first entry is a process repository and the second entry is a process query, and process repositories with the property that each pair is related to exactly one process repository, i.e., it is a function $m : \mathscr{U}_{pr} \times \mathscr{U}_{pq} \rightarrow \mathscr{U}_{pr}$.

Therefore, a process querying method maps an input process repository and a process query onto a resulting process repository obtained by implementing the query statement on the input repository. For instance, given that the model in Figure 1 is contained in the input repository, query q mentioned above will result in a repository that contains the model.

3 Process Querying Methods

This section summarizes all the process querying methods covered in this book. The methods are grouped based on the types of *behavior models* that can be taken as input in *process repositories* and are referred to by the short names of the corresponding languages for specifying *process queries*.

3.1 Log Querying

Process querying methods discussed in this section operate over event logs.

BP-SPARQL. BP-SPARQL is a textual language for summarizing and analyzing process execution data, for example, event logs [4–7]. The language extends SPARQL with constructs for querying Big Process Data described in an RDF graph of process-related entities. Such process-related entities are, for instance, events, actors, and relationships. Examples of relationships include the ordering relations between events and relations between entities and their attributes. The language is implemented using Hadoop, Map Reduce, and Pig-Latin technologies. Being an extension of SPARQL, BP-SPARQL allows querying using standard SPARQL capabilities.

DAPOQ-Lang. The Data-Aware Process Oriented Query Language (DAPOQ-Lang) is a textual language for retrieving sub-logs of event logs and querying data constraints [19, 20]. In this way, DAPOQ-Lang aims to support answering business questions that arise in process mining [1]. DAPOQ-Lang is an SQL wrapper that aims to simplify SQL queries by defining and operating over a concrete meta-model for representing event log data, called the OpenSLEX meta-model. Hence, the focus is on the usability of the language as compared to corresponding SQL queries. The language supports querying over events in traces and related data objects, their

versions, and data schemas, as well as over the temporal properties of all these elements.

PIQL. Process Instance Query Language (PIQL) is a textual language for computing Boolean and numeric process performance indicators over traces and instances of process tasks [23]. The language aims to support business users in decision-making and monitoring and management of business processes. PIQL is user friendly to nonexperts, as in addition to machine-readable, it also offers a human-readable syntax formulated in terms of natural language statements. The language can be used to define decision logic that depends on the information kept in the historical traces and used during future process execution. PIQL can be integrated with Decision Model and Notation (DMN) to support process decisions, obtain measurements to display on process dashboards, and support the data flow.

3.2 Model Querying

This section summarizes methods for querying process models. These methods can be split into those that operate over process model specifications and those that target behaviors encoded in process models. The former methods can be further split into two subclasses. The first subclass comprises methods originally proposed for querying general conceptual models and reported useful for querying process models. DMQL and VM* belong to this subclass. The second subclass consists of dedicated techniques for querying process models. BPMN VQL and Descriptive PQL are languages that belong to the second subclass. Finally, CRL, QuBPAL, and PQL operate over behaviors encoded in process models.

DMQL. The Diagramed Model Query Language (DMQL) is a visual language for querying collections of conceptual models created in arbitrary graph-based modeling languages [9]. The language also supports approximate analysis of the executions of process models in the presence of loop structures. DMQL querying is implemented as searching for model subgraphs that correspond to a predefined pattern query. DMQL is proposed not only to query process models but can also be used to query data models, organizational charts, and other model types.

VM*. The Visual Model Manipulation Language (VM*) is a family of languages for expressing queries (VMQL), constraints (VMCL), and transformations (VMTL) over conceptual models [2, 38, 39]. The authors of VM* languages advocate their application over process models, for example, UML Activity Diagrams and BPMN models. VM* extends the meta-model of the host language with several intuitive annotations. Such an approach broadens the usability of VM* queries, as they resemble models captured in the host language, which decreases the user effort for reading and writing queries. For example, VMQL for querying BPMN models subsumes the syntax and notation of BPMN. The matching of VM* queries is implemented through pattern matching over model graphs.

BPMN VQL. BPMN Visual Query Language (BPMN VQL) is a visual process query language. BPMN VQL can be used to retrieve structural information about the queried models and knowledge related to the domain of the models. BPMN VQL queries follow the structure of SQL, borrow the syntax from BPMN, while the semantics of BPMN VQL queries is grounded in SPARQL [13]. A BPMN VQL query consists of two parts. The matching part of the query specifies a structural condition to match in process models, whereas the selection part specifies parts of models to retrieve as a query result. When executed, BPMN VQL queries get translated to SPARQL using a formalization of the BPMN meta-model as an RDF ontology. The authors of the language have conducted empirical studies that demonstrate that BPMN VQL queries are easier to understand than natural language queries and that it is easier to formulate BPMN VQL queries than to match natural language queries against process models.

Descriptive PQL. Descriptive Process Query Language (Descriptive PQL) is a textual language for retrieving process models and specifying abstractions and changes over process models [16]. The manipulations on models are defined using Single-Entry-Single-Exit subgraphs [30] and implemented via translations to the Cypher graph query language, where Cypher is a declarative query language for querying and changing graphs stored in graph database [22].

QuBPAL. Querying Business Process Abstract modeling Language (QuBPAL) is a textual ontology-based language for retrieving process fragments and their subsequent reuse, e.g., when constructing fresh process models. QuBPAL queries resemble SPARQL queries and are executed over collections of BPMN process models, their meta-model and behavioral semantics, domain knowledge, and semantic annotations. When executed, QuBPAL queries are translated into Logic Programming queries and evaluated using the Prolog inference engine [34–36]. The authors suggest that QuBPAL can also be used for querying at run-time over running process instances and over the logs of completed processes.

CRL. Compliance Request Language (CRL) is a process query language grounded in temporal logic designed to support standard process compliance rules [12]. Specifically, CRL supports process compliance rules that address control flow, resource, and temporal aspects of business processes. CRL queries are executed over BPEL specifications by translating the queries to LTL or MTL and then model checking the resulting temporal logic properties over BPEL specifications using the SPIN model checker [15]. CRL is designed with the relevance and usability of the supported queries in mind. For instance, the control flow compliance rules are grounded in the patterns identified in a comprehensive survey [11].

PQL. Process Query Language (PQL) is a textual query language based on executions and behavioral relations, e.g., ordering, mutual exclusion, and concurrency, between tasks in executions of process models [26, 27, 29]. PQL reuses parts of the abstract syntax of APQL and has an SQL-like concrete syntax. The semantics of PQL is defined over all possible executions of process models and is independent of notations used to describe process models.

The core of PQL is grounded in the behavioral relations of the 4C spectrum [31]—a systematic collection of the co-occurrence, conflict, causality, and concurrency relations. In addition, PQL can reason at the level of process scenario templates (a.k.a. sample process executions with placeholders). For example, one can retrieve models from a process model collection that can execute a specified process scenario or augment models to describe a fresh process scenario template. The latter query type is implemented in PQL as a solution to the process model repair problem [25].

3.3 Log and Model Querying

Methods from this section can be used to query process models and event logs.

BPQL. Business Process Query Language (BPQL) is a textual language for querying over process models and event logs [17, 18]. It aims at making process specification more flexible by defining such elements as resource assignment and transition conditions via BPQL queries evaluated during process execution. The language is defined as an extension of the Stack-Based Query Language (SBQL) [40]. The semantics of the language is defined over the proposed abstract model for capturing process specifications and execution traces. The authors of BPQL proposed BPQL templates for monitoring process execution and integrated the language with the BPMN standard. BPQL can be used to aggregate information over the attributes of tasks, for instance, to compute the cost of a trace based on the costs of individual tasks in the trace.

Celonis PQL. Celonis Process Query Language (Celonis PQL) is a domain-specific language that operates over a process data model that combines data about a process of interest from various systems into one snowflake schema [41]; snowflake schemas are often used to develop data warehouses. Celonis PQL is designed for business users and aims to support process discovery, enhancement, and monitoring, three well-studied problems in process mining [1]. Business users can use Celonis PQL to formalize their process questions and execute them automatically to gain valuable process-related insights. The language supports over 150 operators, including process-related functions, machine learning, and mathematical operators. The syntax of the language resembles SQL. Thousands of users from different industries use Celonis PQL daily.

4 Process Querying Techniques

Existing process querying methods are often founded on well-established techniques. Some most commonly used techniques used to implement process querying are summarized in this section.

Structural Analysis. Behavior models like process models or event logs can be formalized as graphs. Several methods perform process querying by analyzing structural properties of graphs used to describe behavior models, e.g., DMQL and VM*. Examples of graph analysis tasks used for process querying include the *problem of determining if a path exists in a graph* and *subgraph isomorphism problem*.

Behavioral Analysis. Several existing process querying methods can support the analysis of properties of behaviors encoded in models, e.g., QuBPAL. These methods can be used to issue process queries that specify conditions over all the processes (of which there can be potentially infinitely many) encoded in the behaviors of models stored in a process repository.

Given a model of a finite-state system, e.g., a behavior model or an event log, and a formal property, *model checking* techniques verify whether the property holds for the system [3]. The formal properties are usually specified using formal logics.

Temporal Logic. A temporal logic is a formal language for specifying and reasoning about propositions qualified in terms of time [21]. Several process querying methods are grounded in temporal logics, e.g., CRL. Given a process repository and a process query, these methods proceed by translating the query into a temporal logic expression, e.g., Linear Temporal Logic (LTL), Computation Tree Logic (CTL), or Metrical Temporal Logic (MTL) expression. Then, each behavior model from the repository is translated into a finite-state system and verified against the expression. Those behavior models that translate to systems for which the property captured in the expression holds constitute the query result.

First-order Logic. First-order Logic (FOL) is a formal logic that extends propositional logic, which operates over declarative propositions, with the use of predicates and quantifiers [37]. The QuBPAL language for process querying is an example language that operates by translating its queries to FOL and model checking them against the models in the repository. Hence, the process repository is interpreted as a formal FOL theory, while queries verify the formal properties of the theory.

Process querying can be grounded in techniques for *data querying*.

Data Querying. Data querying is a technique for retrieving and augmenting data. Structured Query Language (SQL) is a language for managing data stored in a relational database [8]. DAPOQ-Lang is an example language for process querying grounded on SQL. It is an SQL wrapper that aims to simplify query formulation over a relational model for storing collections of event logs.

SPARQL Protocol and RDF Query Language (SPARQL). SPARQL is a semantic query language for retrieving and manipulating data captured in the Resource Description Framework (RDF) format [14]. Hence, SPARQL queries operate over data stored as a collection of "subject-predicate-object" triples. Several methods for

process querying are based on SPARQL, e.g., BPMN VQL and BP-SPARQL. These methods encode process repositories as RDF data and implement process querying by first transforming process queries into SPARQL queries and then executing SPARQL queries over the RDF data.

Graph Querying. A graph querying technique can be used for data querying in annotated graphs [33]. For instance, Descriptive PQL uses the Cypher graph query language [22] to implement process querying; Cypher is a declarative graph query language used in the Neo4J graph database. By relying on Cypher, Descriptive PQL queries can formulate intents to retrieve and augment underlying graph structures of the queried process models.

5 Process Querying Applications

Due to their generic purpose, namely retrieval, manipulation, and management of behavior models stored in process repositories, process querying methods have many applications in process mining and business process management. Some example applications of process querying are listed below. These applications were mentioned and exemplified by the authors of the process querying methods covered in this book. The list is not meant to be exhaustive but should provide an impression about the broad range of process querying applications.

Business process compliance management. Business process compliance management studies approaches to check and ensure that business processes satisfy relevant compliance requirements, for example, legal regulations and policies. Process queries are often used to specify conditions that lead to violations of compliance requirements, while the corresponding query results contain models that violate the requirements.

Business process weakness detection. A weakness of a business process is its part that hinders process execution or has a negative impact on process performance. Such weaknesses are often modeled using syntactically correct model fragments, but according to their semantic, they are undesirable or even harmful. An example of a weakness in a process is when a document is first printed and later scanned. A condition that represents a business process weakness can be formulated as a process query and then executed over a process repository to identify all occurrences of the weakness.

Infrequent behavior detection. Real-world event data are full of noise and rare anomalous behavior. Using such raw data as input to process mining techniques is detrimental to the results and, hence, insights about real-world processes these techniques deliver. Infrequent behavior detection studies algorithms for identifying noise and irregular event patterns in event data to be filtered before applying process mining. Some process querying methods allow specifying conditions and, thus, querying for infrequent process behavior.

Process discovery. Process discovery is a problem studied in process mining and consists of automating the task of process modeling. That is, given event data, e.g., an event log of an IT system, a process discovery algorithm automatically constructs a process model that describes the data. Process querying methods can be used to support process discovery, for example, by retrieving event data of interest and filtering out the data that discovery algorithms deem not important for fulfilling their tasks.

Process enhancement. Process enhancement can be seen as a problem that generalizes the problem of process discovery. Process enhancement aims to improve, for example, extend, correct, or annotate, an existing process model based on event data about its actual executions. Hence, process discovery can be seen as process enhancement when the original process model is empty. Similar to process discovery, various process queries can be used to support steps of process enhancement algorithms concerning event data selection.

Process instance migration. Process instance migration is the task of adapting an incomplete execution of a process model to continue execution according to the rules of a different process model. This different process model can be a redesigned version of the original model that caters to new requirements. The migration instructions can be formalized as process queries and executed using process querying methods.

Process model comparison. The problem of process model comparison deals with assessing how similar two given process models are. Note that the problem can be instantiated with different notions of similarity, including structural and behavioral criteria. Such criteria for assessing similarity can be specified as process queries. Then, models that get included in the results of most of such similarity queries can be accepted as most similar.

Process model translation. Process model translation deals with translating process model labels from one natural language to another, for example, from English to Ukrainian, or vice versa. Solving this problem is useful when process models are used in multinational companies, as process models can be reused at different branches of the company around the globe. The operationalization of the relabeling of the process model concepts can be implemented through process querying.

Process monitoring. Process monitoring is the task of identifying problematic and successfully performing processes. The aggregated information about the performance of currently executing processes can often be implemented via process queries and then visualized via process performance dashboards.

Process reuse. Process reuse refers to the problem of constructing process models from the existing ones or their fragments. Hence, existing process models, fragments, and process patterns from other contexts are reused instead of creating them from scratch. Process querying can be used to discover reusable process pieces with desired characteristics for subsequent composition in new process models.

Process scheduling. As resources are in general scarce and usually follow certain availability patterns, for example, due to the work shifts or maintenance cycles, their

availability needs to be scheduled. Process scheduling is concerned with determining which resources and when to utilize in order to maximize the performance of currently executing processes or process parts. Process querying can support process scheduling, for example, to inquire about the status of the resources, execute the scheduling logic, or assign resources to pending process tasks.

Process selection. Process selection refers to the practice of choosing how an organization carries out its operations, for instance, customer claim handling. Indeed, the exact process followed may differ for customers of various demographic groups. Such process selection rules, or business rules, can be encoded as process queries that use process case information and information on past process executions to implement the decision logic.

Process standardization. Standard process models are models that should be used as references; that is, they are exemplar models for describing best practices for certain classes of behaviors. Process standardization refers to the problem of replacing similar process models or similar model fragments with a single unified model or fragment. Process querying is useful at the early stages of process standardization, as queries can help identify similar fragments that should be standardized. Such similar models or model fragments can be included in the result of a corresponding carefully designed process query.

Process variance management. Process variance management is the task of identifying, maintaining, and improving the handling of variants of the same process in an organization. This task can be supported by process querying at various stages. For example, process queries can specify conditions for identifying process variants, pinpointing their differences, and supporting their standardization.

Syntactical correctness checking. Syntactical correctness checking is the task of identifying process models that violate the syntax rules of the modeling language used to capture them. Alternatively, syntactical correctness checking can be used to verify whether process models adhere to the modeling guidelines established by the organization. For example, the organization can establish that only a subset of the modeling constructs is allowed in process models. Rules for checking the syntactical correctness of process models can be captured as process queries.

6 Past, Present, and Future of Process Querying

Future work on process querying will aim to achieve *process querying compromise* [28], i.e., it will propose new and improve the existing process querying methods to support more practically relevant process queries that can be computed efficiently. As of today, the development of process querying methods still constitutes a non-coordinated effort. Many languages and systems for process querying were designed and developed in silos. Future work should contribute to the consolidation of various methods leading to the definition of a standard meta-model for behavior models

and behaviors that these models describe, and a standard language for capturing process queries. Such consolidation may take place, for instance, around the Process Querying Framework [28], which defines typical components of a process querying solution as well as their interfaces and details roles of the constituent components. This book is the first step in this direction.

The existing process querying languages and methods cover a wide range of use cases and applications. A small tendency is observed toward the design of more methods that operate over specifications of behavior models. Besides, while there is a plethora of languages and techniques for capturing and executing instructions for filtering process repositories, i.e., selecting processes with specific characteristics, methods for manipulating process repositories, i.e., changing processes, are scarce and are still in their infancy. Future research will close these gaps by devising techniques that operate over the behaviors process models describe rather than their structures and developing methods that manipulate models to include or exclude certain behaviors they describe.

Many process querying techniques suffer from performance issues, if not for most basic queries then for the intricate ones. To overcome such performance limitations, dedicated indexes can be designed and constructed for certain classes of process queries to allow trading the additional space for storing the indexes for the speedup in the processing of queries. This design of indexes should be guided by empirical investigations on which queries are considered most useful by the users. Indeed, those intricate and useful queries should inform the development of process querying indexes. Unfortunately, only several such empirical works exist, and this gap must be addressed in future work.

References

1. van der Aalst, W.M.P.: Process Mining – Data Science in Action, 2nd edn. Springer Berlin Heidelberg (2016). DOI 10.1007/978-3-662-49851-4
2. Acretoaie, V., Störrle, H., Strüber, D.: VMTL: a language for end-user model transformation. Software and Systems Modeling **17**(4), 1139–1167 (2018). DOI 10.1007/s10270-016-0546-9
3. Baier, C., Katoen, J.: Principles of Model Checking. MIT Press (2008)
4. Beheshti, A., Benatallah, B., Motahari-Nezhad, H.R.: ProcessAtlas: A scalable and extensible platform for business process analytics. Software: Practice and Experience **48**(4), 842–866 (2018). DOI 10.1002/spe.2558
5. Beheshti, S., Benatallah, B., Motahari-Nezhad, H.R.: Scalable graph-based OLAP analytics over process execution data. Distributed and Parallel Databases **34**(3), 379–423 (2016). DOI 10.1007/s10619-014-7171-9
6. Beheshti, S., Benatallah, B., Nezhad, H.R.M.: Enabling the analysis of cross-cutting aspects in ad-hoc processes. In: Advanced Information Systems Engineering, *Lecture Notes in Computer Science*, vol. 7908, pp. 51–67. Springer (2013). DOI 10.1007/978-3-642-38709-8_4
7. Beheshti, S., Benatallah, B., Nezhad, H.R.M., Sakr, S.: A query language for analyzing business processes execution. In: Business Process Management, *Lecture Notes in Computer Science*, vol. 6896, pp. 281–297. Springer Berlin Heidelberg (2011). DOI 10.1007/978-3-642-23059-2_22
8. Date, C., Darwen, H.: A Guide to the SQL Standard: A User's Guide to the Standard Database Language SQL. Addison-Wesley (1997)

9. Delfmann, P., Breuker, D., Matzner, M., Becker, J.: Supporting information systems analysis through conceptual model query – the diagramed model query language (DMQL). Communications of the Association for Information Systems **37**, 24 (2015)
10. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management, 2nd edn. Springer (2018). DOI 10.1007/978-3-662-56509-4
11. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. In: Proceedings of the 21st international conference on Software engineering - ICSE '99, pp. 411–420. ACM Press (1999). DOI 10.1145/302405.302672
12. Elgammal, A., Turetken, O., Heuvel, W.J., Papazoglou, M.: Formalizing and applying compliance patterns for business process compliance. Software & Systems Modeling **15**(1), 119–146 (2016). DOI 10.1007/s10270-014-0395-3
13. Francescomarino, C.D., Tonella, P.: Crosscutting concern documentation by visual query of business processes. In: Business Process Management Workshops, *Lecture Notes in Business Information Processing*, vol. 17, pp. 18–31. Springer Berlin Heidelberg (2008). DOI 10.1007/978-3-642-00328-8_3
14. Hebeler, J., Fisher, M., Blace, R., Perez-Lopez, A., Dean, M.: Semantic Web Programming, 1st edn. Wiley (2009)
15. Holzmann, G.J.: The model checker SPIN. IEEE Trans. Software Eng. **23**(5), 279–295 (1997). DOI 10.1109/32.588521
16. Kammerer, K., Kolb, J., Reichert, M.: PQL – A descriptive language for querying, abstracting and changing process models. In: Enterprise, Business-Process and Information Systems Modeling, *Lecture Notes in Business Information Processing*, vol. 214, pp. 135–150. Springer (2015). DOI 10.1007/978-3-319-19237-6_9
17. Momotko, M.: Tools for monitoring workflow processes to support dynamic workflow changes. Ph.D. thesis, Polish Academy of Sciences (2005)
18. Momotko, M., Subieta, K.: Process query language: A way to make workflow processes more flexible. In: Advances in Databases and Information Systems, *Lecture Notes in Computer Science*, vol. 3255, pp. 306–321. Springer Berlin Heidelberg (2004). DOI 10.1007/978-3-540-30204-9_21
19. de Murillas, E.G.L., Reijers, H.A., van der Aalst, W.M.P.: Everything you always wanted to know about your process, but did not know how to ask. In: Business Process Management Workshops: Process Querying, *Lecture Notes in Business Information Processing*, vol. 281, pp. 296–309 (2016). DOI 10.1007/978-3-319-58457-7_22
20. de Murillas, E.G.L., Reijers, H.A., van der Aalst, W.M.P.: Connecting databases with process mining: a meta model and toolset. Software & Systems Modeling **18**(2), 1209–1247 (2018). DOI 10.1007/s10270-018-0664-7
21. Ohrstrom, P., Hasle, P.F.V.: Temporal Logic: From Ancient Ideas to Artificial Intelligence. Studies in Linguistics and Philosophy. Springer Netherlands (2007)
22. Panzarino, O.: Learning Cypher. Packt Publishing (2014)
23. Pérez-Álvarez, J.M., López, M.T.G., Parody, L., Gasca, R.M.: Process instance query language to include process performance indicators in DMN. In: IEEE Enterprise Distributed Object Computing Workshops, pp. 1–8. IEEE Computer Society (2016). DOI 10.1109/EDOCW.2016.7584381
24. Polyvyanyy, A.: Business process querying. In: Encyclopedia of Big Data Technologies. Springer (2019). DOI 10.1007/978-3-319-63962-8_108-1. URL https://doi.org/10.1007/978-3-319-63962-8_108-1
25. Polyvyanyy, A., van der Aalst, W.M.P., ter Hofstede, A.H.M., Wynn, M.T.: Impact-driven process model repair. ACM Transactions on Software Engineering and Methodology **25**(4), 1–60 (2017). DOI 10.1145/2980764
26. Polyvyanyy, A., Corno, L., Conforti, R., Raboczi, S., Rosa, M.L., Fortino, G.: Process querying in Apromore. In: Business Process Management Demo Session, *CEUR Workshop Proceedings*, vol. 1418, pp. 105–109. CEUR-WS.org (2015)
27. Polyvyanyy, A., ter Hofstede, A.H.M., Rosa, M.L., Ouyang, C., Pika, A.: Process query language: Design, implementation, and evaluation. CoRR **abs/1909.09543** (2019)

28. Polyvyanyy, A., Ouyang, C., Barros, A., van der Aalst, W.M.P.: Process querying: Enabling business intelligence through query-based process analytics. Decision Support Systems **100**, 41–56 (2017). DOI 10.1016/j.dss.2017.04.011
29. Polyvyanyy, A., Pika, A., ter Hofstede, A.H.M.: Scenario-based process querying for compliance, reuse, and standardization. Inf. Syst. **93**, 101,563 (2020)
30. Polyvyanyy, A., Vanhatalo, J., Völzer, H.: Simplified computation and generalization of the refined process structure tree. In: Web Services and Formal Methods, *Lecture Notes in Computer Science*, vol. 6551, pp. 25–41. Springer (2010). DOI 10.1007/978-3-642-19589-1_2
31. Polyvyanyy, A., Weidlich, M., Conforti, R., Rosa, M.L., ter Hofstede, A.H.M.: The 4C spectrum of fundamental behavioral relations for concurrent systems. In: Application and Theory of Petri Nets and Concurrency, *Lecture Notes in Computer Science*, vol. 8489, pp. 210–232. Springer International Publishing (2014). DOI 10.1007/978-3-319-07734-5_12
32. Reinkemeyer, L.: Process Mining in Action: Principles, Use Cases and Outlook. Springer International Publishing (2020). URL https://books.google.com.au/books?id=OrHWDwAAQBAJ
33. Robinson, I., Webber, J., Eifrem, E.: Graph Databases. O'Reilly (2013)
34. Smith, F., Missikoff, M., Proietti, M.: Ontology-based querying of composite services. In: Business System Management and Engineering, *Lecture Notes in Computer Science*, vol. 7350, pp. 159–180. Springer Berlin Heidelberg (2010). DOI 10.1007/978-3-642-32439-0_10
35. Smith, F., Proietti, M.: Rule-based behavioral reasoning on semantic business processes. In: International Conference on Agents and Artificial Intelligence, pp. 130–143. SciTePress (2013)
36. Smith, F., Proietti, M.: Ontology-based representation and reasoning on process models: A logic programming approach. CoRR **abs/1410.1776** (2014)
37. Smullyan, R.M.: First-order Logic. Courier Corporation (1995)
38. Störrle, H.: VMQL: A generic visual model query language. In: IEEE Visual Languages and Human-Centric Computing, pp. 199–206. IEEE Computer Society (2009). DOI 10.1109/VLHCC.2009.5295261
39. Störrle, H.: VMQL: A visual language for ad-hoc model querying. Journal of Visual Languages & Computing **22**(1), 3–29 (2011). DOI 10.1016/j.jvlc.2010.11.004
40. Subieta, K.: Stack-based query language. In: Encyclopedia of Database Systems, pp. 2771–2772. Springer US (2009). DOI 10.1007/978-0-387-39940-9_1115
41. Vogelgesang, T., Kaufmann, J., Becher, D., Seilbeck, R., Geyer-Klingeberg, J., Klenk, M.: Celonis PQL: A Query Language for Process Mining. Springer (2020). (In Press)
42. Weske, M.: Business Process Management: Concepts, Languages, Architectures, 3rd edn. Springer (2019). DOI 10.1007/978-3-662-59432-2