The VLDB Journal (2019) 28:575–595
https://doi.org/10.1007/s00778-019-00539-y

REGULAR PAPER

Prescriptive analytics: a survey of emerging trends and technologies

Davide Frazzetto[1] · Thomas Dyhre Nielsen[1] · Torben Bach Pedersen[1] · Laurynas Šikšnys[1]

Received: 29 March 2018 / Revised: 15 March 2019 / Accepted: 28 March 2019 / Published online: 23 May 2019

Abstract
This paper provides a survey of the state-of-the-art and future directions of one of the most important emerging technologies within business analytics (BA), namely *prescriptive analytics* (PSA). BA focuses on data-driven decision-making and consists of three phases: descriptive, predictive, and prescriptive analytics. While descriptive and predictive analytics allow us to analyze past and predict future events, respectively, these activities do not provide any direct support for decision-making. Here, PSA fills the gap between data and decisions. We have observed an increasing interest for in-DBMS PSA systems in both research and industry. Thus, this paper aims to provide a foundation for PSA as a separate field of study. To do this, we first describe the different phases of BA. We then survey classical analytics systems and identify their main limitations for supporting PSA, based on which we introduce the criteria and methodology used in our analysis. We next survey, categorize, and discuss the state-of-the-art within emerging, so-called PSA$^+$, systems, followed by a presentation of the main challenges and opportunities for next-generation PSA systems. Finally, the main findings are discussed and directions for future research are outlined.

Keywords Business intelligence · Database systems · Data analytics · Decision support systems

1 Introduction

Today's world is fast becoming inextricably connected to information technologies. Cloud services, smart machines, Internet of things, and mobile computing are some of the top technological trends reported by Gartner research in 2016 [33], showing how both business enterprises and users are going through a process of digitalization of their activities, rapidly leading to a higher data production rate. This massive production of data poses new questions: how do we efficiently store and manipulate such data and, most of all, how do we generate value from it? Data describe facts about the present and the past. Thus, when collected and stored, data are already *dead*, meaning that by itself it does not provide any new understanding beyond those of the mere historical facts [41]. If no efficient ways are found to make use of the data we generate, the advantage of being able to collect such data simply vanishes.

In a recent report [32], Gartner identifies *business analytics* (BA) as a top priority on the chief information officers' agendas, by accounting for as much as 50% of the planned investments, thereby largely surpassing other technologies such as infrastructures and data centers, cloud computing, and service digitalization. BA is a set of information technologies (IT) whose objective is to drive business planning by utilizing data about the past to gain new insights about the future [64]. For years, advancements in BA have provided efficient ways to store and analyze data: from simple spreadsheets to advanced DBMSes with integrated analytics functionality such as online analytical processing (OLAP), data mining, machine learning, data visualization, etc [45]. The evolution of the data itself, from its increasing complexity to the velocity with which it is produced, together with the enterprises' need for more effective data analytics solutions, have been the key factors that triggered the advancement of

✉ Davide Frazzetto
david.frazzetto@gmail.com

Thomas Dyhre Nielsen
tdn@cs.aau.dk

Torben Bach Pedersen
tbp@cs.aau.dk

Laurynas Šikšnys
siksnys@cs.aau.dk

1 Department of Computer Science, Aalborg University, Aalborg, Denmark

BA [41]. BA has so far established a way to analyze the current status of an activity and to predict the possible outcomes in the future. However, to bring new value to the process, another step is necessary: find and evaluate the best course of action to achieve the business goal. From this perspective, it is possible to recognize three phases of analytics within BA, each with a different scope: descriptive analytics (DA), predictive analytics (PDA), and prescriptive analytics (PSA). After answering the questions *what happened?* (DA) and *what will happen in the future?* (PDA), PSA answers *how to make it happen?* allowing users to plan and perform a sequence of actions to optimize the performance of a process. PSA is a new type of data analytics [64], extending the capabilities of the well-known DA and PDA, by enabling data-driven optimization for decision support and planning.

While the term *prescriptive analytics* itself is relatively new, introduced by IBM [64] and trademarked by Ayata only in 2010, the underlying fundamental concepts and techniques have been around for years and cannot be considered novel in themselves. Instead, the novelty of PSA lies in the *combination and integration* of these concepts and techniques in a *synergetic way* taking optimal advantage of hybrid data, business rules, and mathematical/computational models. Thus, PSA is an emerging and not yet established field. To first evaluate how widespread the term *prescriptive analytics* currently is in academia, we found (with Google Scholar) that since 2010 only 67 published papers mention *prescriptive analytics* in their title. Most of these papers analyze problems and present PSA solutions for a specific application domain, such as transportation [77,112], health care [53,107], business process optimization [39], car rental [46], supply chain [98], and smart grids and energy management [87]. The most comprehensive overview of PSA that we have found is given by Soltanpoor et al. [96], where the authors define the distinction between DA, PDA, and PSA, and propose an abstract architecture and a conceptual framework for PSA, specifically for the field of educational research. However, as our goal is to thoroughly explore the current state-of-the-art of PSA, beyond the use of a specific term, we broaden our survey to include papers that, although not explicitly referring to PSA, propose contributions toward PSA development. Here, we focus on the system aspect of PSA, i.e., technical/tool contributions rather than specific solutions. Within this scope, the paper provides the following contributions:

- **Analyzing** the current status of PSA technologies and identifying challenges and opportunities for future research on these. To do so, we begin by briefly outlining the historical evolution of the broader area of BA, following the steps that have lead from simple reports to advanced analytics. We give an overview of the three main phases of BA—namely DA, PDA, and PSA, defining the objectives and requirements for each of them.
- **Identifying** the typical tasks and the different approaches that have been pursued for PSA. To delimit the scope of the paper, we focus on categorizing and comparing representative research papers and software systems that propose steps to advance data management systems toward PSA. By doing so, we identify state-of-the-art in the field, the trends in the development of PSA systems, and the differences between the emerging approaches. For this survey, we collect the papers presenting the systems that *explicitly* focus on (some of the) aspects of PSA development. To our knowledge, no previous study has surveyed the state-of-the-art from the point of view of PSA support.
- **Defining** the major challenges and opportunities of PSA, together with an overview of the solutions proposed by the state-of-the-art systems.

This paper is structured as follows. Section 2 describes the evolution of BA, followed by the identification of the different BA phases and tasks. Section 3 gives an overview of the *classical* software systems used in BA applications. Section 4 discusses evaluation criteria of the new emerging systems as well as our survey methodology. Based on these criteria and methodology, Section 5 surveys, evaluates, and compares a number of new emerging systems. Section 6 discusses remaining challenges and opportunities, and Sect. 7 concludes the paper.

2 The evolution and phases of business analytics

In this section, we give an overview of BA to position PSA in its proper context. To do this, we start by describing the evolution of BA and the characteristics of BA systems, followed by the identification of the different BA phases and tasks. Lastly, we give a real-world use case example of a PSA application.

2.1 Introduction to business analytics

The first appearance of the term business intelligence, or business analytics (BA), can be traced back to 1958, when Hans Peter Luhn published in an IBM journal the article “A Business Intelligence System” [63]. The author defined the term business as a “[...] collection of activities carried on for whatever purpose, be it science, technology, commerce, industry, law, government, defense, et cetera.”, and the term intelligence as “[...]the ability to apprehend the interrelationships of presented facts in such a way as to guide action towards a desired goal[...].” BA aims at providing sophisticated information analysis and supporting managerial decisions by making use of large amounts of data [17].

Considering the technological applications of data analytics, the evolution of BA has spanned across multiple fields, from the appearance of a computerized weather forecast system in 1950 and the release of Visicalc in 1979 (the first commercial spreadsheet software), over to the widespread use of modern BA software suites [45]. The classic BA setup is built around a (typically large) data warehouse [49], on which various operations are possible: from simple reports and Structured Query Language (SQL) queries to slicing and dicing of multidimensional data [19] and applying advanced data mining algorithms [64]. The combination of data management and analytical tools makes it possible to support business-level decision-making. Therefore, BA can be seen as the intersection of two main research areas—decision support systems (DSSs) and data management systems.

DSSs are computer technology solutions that are used for supporting decision-making and problem solving. Over the last three decades, research in DSS has evolved from the early work that started around 1985 (see [13,26,54]), to modern solutions [81,90] that comprehend (i) advanced DBMSes, (ii) mathematical modeling functions, and (iii) user interfaces offering querying and analysis tools as well as graphical visualization capabilities. The evolution of DSS has been the fruit of the research in different fields, e.g., computer science, mathematics, and operations research, leading to a combination of approaches belonging to different disciplines [91].

A crucial contribution to the evolution of DSSs has been provided by the database community research, specifically with the 1990s work in data warehousing, OLAP, and data mining. The research in the database field has contributed to the appearance of *data-driven* DSSs, where the source and focus of the analysis have shifted from the mathematical models that were previously the main DSS tools, to the data, thanks to the large quantity of information now being available. As the amount of data that can be stored and processed increases, so do the possibilities of utilizing such data for analytical purposes.

We start now by first presenting the recognized structure of BA. Then, we focus on the aspects most relevant for PSA, including the typical tasks involved and the application workflow.

2.2 The structure of business analytics

The consolidation of database solutions and data warehousing, and the diffusion of DSSs, have lead to BA becoming an umbrella term that covers a broad range of IT technologies [45,64], among others: data management, data warehousing, OLAP, statistics, data mining, machine learning, operations research, data visualization, etc. (Fig. 1). Because of this increase in the number of tools, and therefore also in the number of scenarios in which BA has found application, it is impossible to associate BA with a single field. Nevertheless, the following three core objectives of BA can be identified [64]:

Hindsight:	Gain understanding of the decision process, to obtain a structured description and view of the past and of the current state.
Insight:	Utilize the past to predict future events.
Foresight:	Combine knowledge about the past and the future to drive decision-making and optimization of the decision process.

Considering these objectives, it has been attempted to divide BA into three phases—*descriptive analytics* (DA), *predictive analytics* (PDA), and *prescriptive analytics* (PSA) [64,93]:

DA:	*What happened?* Involves techniques that provide historical data analysis, typically based on data aggregation and data mining.
PDA:	*What will happen?* Involves techniques, typically based on machine learning, that aim at producing predictions and forecasts.
PSA:	*How to make it happen?* Involves techniques to evaluate and find the best alternatives for a decision process, given a (complex) set of objectives, requirements, and constraints.

The differences between DA, PDA, and PSA can clearly be seen in the context of a typical generic decision-making workflow, as shown in Fig. 2. This includes a number of tasks carried out by human analysts, using one or more BA tools. In this context, DA, PDA, and PSA offer different levels of support to the user (human analyst) for solving a complex decision-making problem. Here, a specific BA phase consists of a set of tasks, for which the user is offered support, assistance, or automation by a given BA tool. We now give a short overview of the tasks in this workflow:

1. *Collect and consolidate data*
   In this task, data on the decision process are collected and stored, e.g., in DBMSes, data sheets, distributed file systems, etc. Moreover, cleaning and consolidating the data to eliminate errors and inconsistencies, or data mining to

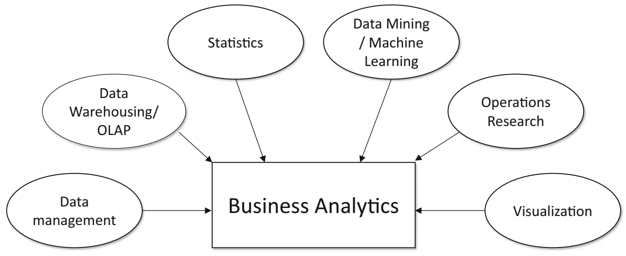

Fig. 1 Key areas that contribute to BA

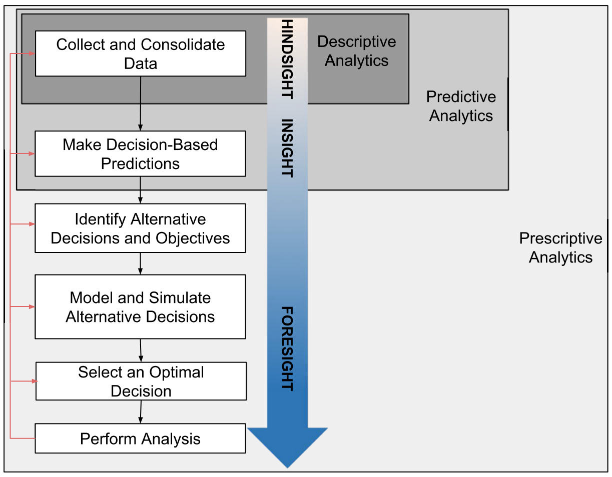
Collect and Consolidate Data
Make Decision-Based Predictions
Identify Alternative Decisions and Objectives
Model and Simulate Alternative Decisions
Select an Optimal Decision
Perform Analysis
ITERATE
HINDSIGHT
INSIGHT
FORESIGHT
Descriptive Analytics
Predictive Analytics
Prescriptive Analytics

Fig. 2 Phases and tasks of a decision-making workflow . (Adapted from [101, p. 5])

extract information and features, are common operations within this task.

2. *Make decision-based predictions*
   Using techniques from machine learning (e.g., time series forecasting, Markov models), the goal of this task is to perform an exploratory analysis of the events and trends that condition the decision process by analyzing historical data (provided by *Task 1*). This task is an intermediate activity, which the analyst can perform to obtain predictions about the future of the decision process, which can in turn be used to guide decision-making in the subsequent tasks.
3. *Identify alternative decisions and objectives*
   The first goal of this task is to identify the objective(s), rules, and constraints of the decision task. To support this task, analysis of the decision process is performed with techniques such as *business rule management* and *process mining* [1,110]. After having specified the objective(s), the second goal is to identify alternative decision options, together with the respective cost/gain functions and associated constraints.
4. *Model and simulate alternative decisions*
   Next, the effect that the decision options will have on the decision process has to be estimated. This task is connected to *Task 2*, where prediction and simulation models can be used to help *simulate* the behavior of the system under different settings (decision alternatives). Generally, simulation models can be either manually defined by the user (using purpose oriented languages and software) or automatically inferred from the available historical data.
5. *Select an optimal decision*
   Selecting an optimal decision is what we refer to as a *prescription*. With *Task 2* we know how to model the future, and *Task 4* tells us how our actions will affect the decision process. Therefore, it is possible to utilize techniques such as *optimization* or *game theory* to find an optimal course of action relative to the objectives and decision options identified in *Task 3* and the events predicted in *Task 2*.
6. *Perform analysis*
   This is an intrinsically iterative workflow: after the (prescribed) decision options have been realized, the resulting process events are observed. Collecting the data pertaining to these events can now trigger a new iteration of the BA workflow, starting again with *Task 1*. To understand the results of the prescriptions and guide the workflow process, we can analyze the effects of the changes on the decision process using, for example, visualization tools or business key performance indicators (KPIs).

Within the context of this workflow, DA, PDA, and PSA are interdependent—DA is a sub-phase of PDA and PDA is a sub-phase of PSA. Among these, PSA offers the highest level of support within this workflow and therefore has the highest value among the different phases. Based on the described workflow, we will provide more detailed descriptions of BA phases in the next sections.

2.3 Descriptive analytics

DA is the most widespread and established phase of BA, as the vast majority of the tools currently used for analytics falls within this phase. The focus in this phase is on collecting, categorizing, and classifying data, as well as on identifying and visualizing relevant patterns in the data [64]. It is possible to recognize in the research of the data management field some of the most important technologies that have enabled advanced data analytics, laying the basis for the development of BA [51]. The introduction of data warehousing and OLAP alongside the more traditional DBMSes opened new possibilities for sophisticated and easily accessible data analytics (see [56,68]). Furthermore, other techniques that have become standard toolboxes for DA applications are data visualization, dashboards, statistical analysis, and data mining [76]. Methods such as pattern matching and clustering are often a standard starting point in the decision process, allowing the user to extract, translate, and visualize the information contained in the data in a more meaningful and simple way.

2.4 Predictive analytics

DA tools lack the capability to perform *predictions* about future events. PDA borrows many ideas and techniques from machine learning, data mining, and statistics [3,24,62,67,80], generally making use of large volumes of historical data to extract and synthesize novel information [36]. These techniques provide ways to, for example, forecast the probability of certain events, find patterns that may repeat in the future, and determine relationships between events. PDA aims at providing support for planning and decision-making by modeling the process not only in terms of what has happened in the past, but also of what will happen in the future. PDA has been applied in different BA contexts, e.g., marketing and financial services, health care, supply chains, capacity planning, etc. [82,92,109]

2.5 Prescriptive analytics

PSA has already been successfully applied in many industrial and research scenarios (see [37,96,97,107]) and logically follows the path lead by the two previous phases of BA: If the past has been understood (DA), and predictions about the future are available (PDA), then it is possible to actively suggest (prescribe) a best option for adapting and shaping the plans according to the predicted future. In comparison with the other phases of BA, PSA allows decision-makers to not only identify issues and opportunities (by looking into past, present, or future), but also to directly prescribe the best decision options according to certain objectives and to evaluate their results.

Although optimization techniques are already a well-established and largely adopted way of solving decision problems, through the use of mathematical tools (see [9,10,108]), it is the combination of predictions and optimization that opens new possibilities for decision support. Moreover, as PSA is often applied to real-world cases with significant uncertainty, the optimization heavily relies on the accuracy of the predictions and, in some cases, the ability to quantify the uncertainty about the predictions. In such situations, the optimization may explicitly take into account the uncertainty inherent in the domain through the (combined) use of statistical and simulation-based models [6,41].

To further elucidate and contrast the three BA phases, we will in the following section exemplify the phases w.r.t. a concrete use case.

2.6 A prescriptive analytics use case

Consider a shopkeeper who needs to decide on which items to keep in storage in order to maximize sales profits. Figure 3 outlines the workflow of this decision process. The shopkeeper has access to item characteristics as well as previous sales and promotion data, which can be exploited by a BA solution. The storage can hold a maximum of 70 items (for simplicity, we can assume that each item occupies a single space unit); hence, the BA solution should prescribe an optimal storage management strategy conditioned on this storage constraint.

The first task (1 and 2 in Fig. 3) consists in the collection and integration of the disparate data sources. For this particular example, the data sources may include both structured and unstructured information; hence, the data may have to be cleaned and processed. These tasks are generally performed with DA tools, which can also help find patterns and relationships in the data.

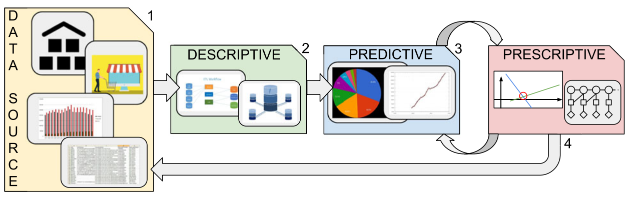

Fig. 3 Workflow for the shopkeeper use case. Data are collected (1), transformed, and stored in a database (2). The future sales trends are predicted (3), and the optimal storage management is (iteratively) prescribed and applied (4)

PDA lays the bridge between data and the subsequent decision-making (3 in Fig. 3). Here, machine learning and data mining techniques may be used to predict future sales of particular items based on item characteristics and previous sales information. However, predicting future events and sales does not in itself provide a strategy for storage management. For this an additional task is required.

PSA addresses this type of decision problem (4 in Fig. 3). In our example, the optimization objective can be addressed by considering the predictions obtained in the previous phase, analyzing the effects of the possible decisions, and updating the probability distributions over the sales to ultimately prescribe a storage strategy that maximize the expected sales profits.

2.7 Discussion

So far, we have presented how DA, PDA, and PSA fit in the BA field and how they are interlinked. Among these, PSA has the highest potential value for users. It is again important to note that the underlying techniques used in PSA are not necessarily novel. Similar to the way in which PDA was coined as a new term for already existing machine learning and statistical methods applied in business analytics, PSA considers approaches already in use in operations research [111] and normative decision support systems [13]. However, the value of PSA as a new field of study lies in defining, clarifying, and integrating the *entire* BA workflow, and in how PSA can be made easily available to its users in a general and standardized way. Our definition of PSA is given from a conceptual standpoint. However, in practice PSA is not yet an established phase of BA, and therefore not yet as widespread as DA and PDA. To further investigate the reason for this lack of standardized PSA tools and applications, we will expand on this discussion in the next chapter, where we will compare different state-of-the-art PSA systems.

Table 1 Overview of traditional BA systems and their support levels for both individual descriptive (DS), predictive (PS), and optimization (OS) *tasks* and full DA, PDA, and PSA (·—basic support; ⊙—intermediate support; ⨀—advanced support)

Group	System class	Key representative systems	DS	PS	OS	DA	PDA	PSA
BA Tools	Reporting and spreadsheet tools	Excel, Google Sheets	⨀	·	·	⨀	·	·
	Data Mining & ML libraries	Spark MLlib, Mahout	·	⨀	·	·	·	·
	Data Mining & ML GUI tools	Weka, Hugin	⊙	⨀	·	⊙	⊙	·
	Online ML cloud services	Watson, Azure ML	⊙	⨀	·	⊙	⊙	·
	Mathematical optimization tools	Gurobi, CPLEX, OptaPlanner	·	·	⨀	·	·	·
	Computer algebra tools	Mathematica, Mathcad	⊙	⊙	⨀	⊙	⊙	⊙
	System modeling tools	Dymola, Simulink	⊙	⊙	⊙	⊙	⊙	⊙
BA Suites	Statistical computing suites	MATLAB, R, Julia	⨀	⨀	⊙	⨀	⨀	⊙
	Statistical GUI suites	SAS, SPSS	⨀	⨀	⊙	⨀	⨀	⊙

3 Classical analytics systems

In this section, we first give an overview of the *classical* software systems utilized in both general BA and PSA applications. Drawing on the characteristics and limitations of these systems, we then in the next section define a number of general criteria and properties for evaluating new emerging PSA systems.

To begin, we review software systems that aim at supporting (some of) the tasks described in Fig. 2. We have selected the systems that are already in widespread use in BA applications, defining them as *classical systems* for BA. This selection is not an exhaustive list of the existing systems. Instead, it constitutes a representative list of software systems in a number of system classes.

As seen in Table 1, we divide these classical systems into two main groups: *BA Tools*, specialized software for individual tasks and/or specific models and domains, and *BA Suites*, composed of a set of tools for generic BA applications. For the different classes of systems within these groups, we identify levels to which users are provided support by the system when performing analytics tasks in the PSA workflow (Fig. 2). Further, based on the analytical tasks supported, we also identify the level of DA, PDA, and PSA support for the different tools/suites as basic, intermediate, or advanced.

BA tools from different classes mentioned in Table 1 have been used in BA scenarios for many years. Among these, Excel [75] is one of the most widely used spreadsheet-based BA tools for data analysis, reporting, and charting used in many typical DA cases. As it offers very basic support for predictions and optimizations, the overall PDA and PSA support is fairly limited. Data mining and machine learning (ML) libraries and tools, such as Weka [42], Mahout [80], and MLlib [74], offer graphical user interfaces (GUI) providing support for exploratory and predictive machine learning algorithms integrated with some basic data management primitives. They are therefore suitable and actively used as standalone tools, for basic DA and PDA activities. In this category, Hugin [65] provides support for probabilistic graphical models (PGMs) enabling probabilistic inference and reasoning under uncertainty, as well as models and methods for defining and solving decision problems. Existing online ML cloud services such as Watson [44] and Azure ML [8] are capable of handling much larger data volumes and information processing tasks. However, they are typically used for DA and PDA applications, and the support for PSA is fairly limited. On the other hand, mathematical optimization tools such as Gurobi [40], CPLEX [60], and OptaPlanner [95] focus on mathematical programming and thus offer dedicated languages and generic high-performance solvers for different problem classes such as linear programming and mixed integer programming. Despite substantial support for optimization, these tools alone are not suitable for exploratory data analysis and predictive modeling, and they therefore need to be combined with other tools to provide full DA, PDA, and PSA support. Computer algebra tools such as Mathematica [2] and Mathcad [83] offer a rich set of tools for manipulating mathematical expressions in a way similar to the traditional manual computations of mathematicians. Among other things, these tools offer solvers for effectively solving systems of equations, ordinary differential equations (ODEs), etc. However, they lack general-purpose data management and predictive functionalities, as required for DA, PDA, and PSA. Similarly, system modeling tools such as Dymola and Simulink offer rich environments for analyzing and optimizing complex systems composed of mathematical equations that describe the dynamic behavior of a system. While these system modeling tools can be used in specialized DA, PDA, and PSA applications, they do not offer support for data-driven (as opposed to model-driven) exploratory analyses, predictions, and optimizations.

Irrespective of their differences in focus, these systems share a common characteristic: none of them offer any substantial support for the tasks of the PSA workflow, requiring

Table 2 Productivity feature categorization criteria and values

Productivity features	Traditional $(\cdot)$	Modern $(\odot)$	Advanced $(\bigodot)$
Workflow Support	*algorithm-oriented*	*task-oriented*	*process-oriented*
System Extensibility	*closed system*	–	*extensible systems*
Language Integration	*multiple languages*	–	*unified language*

instead the use of multiple separate tools or integrated *BA suites*.

BA suites provide access to multiple tools in a single integrated environment, facilitating the development of more complex DA, PDA, and PSA applications using a single ecosystem. BA suites can typically provide functionality offered by individual BA tools. For example, MATLAB [71] supplies a programming environment and an engine targeting numerical computing, where analytics components can be utilized as toolboxes. In a similar way, softwares like SAS [89], IBM SPSS/CPLEX [48], and Julia [7] are high-performance suites with integrated analytics more oriented toward business applications. A non-exhaustive survey and comparison of these systems is provided in [100].

In summary, the *BA tools* and *BA suites* do not offer a convenient way of expressing and executing user-defined PSA workflows (Fig. 2). The major limitations are as follows:

- First, they typically support only procedural programming languages with no declarative primitives for expressing the PSA workflow tasks. While software developers are familiar with procedural programming, and often have a high level of expertise with multiple programming languages, data analysts often benefit from a declarative approach, where the focus is on data analysis and not on algorithmic specifications. For example, *BA tools* and *BA suites* still require the use of multiple languages and imperative constructs for expressing different tasks of the PSA workflow, e.g., data collection and prediction.
- Second, there is no native support of the PSA workflow of Fig. 2. Although *BA tools* and *BA suites* let the user develop PSA applications, their design is not directly focused on PSA applications. Therefore, no specific support is provided to the user for these types of use cases. As a result, the different tools required for PSA have to be interconnected manually in an ad hoc fashion, resulting in less structured and more time-consuming and error-prone specifications of PSA workflows. Furthermore, if new algorithms or specialized models (e.g., energy flexibility models [78,79]) need to be used in the application, the closed structure of the architectures might completely prevent this, or lead to the ad hoc integration of new tools, e.g., by the connection of external programs, to which the data have to be transferred via an API.
- Third, the analytics computations are still performed on a single-node machine, and often far away from where the data are stored. Hereby, *BA tools* and *BA suites* often lack highly scalable distributed analytics algorithms, which minimize the overhead from data exchanges and transformations while performing the analytics computations that are part of the PSA workflow. As such, they do not optimize the interleaved data management and analytics workflows, but instead use pre-defined code that makes API calls for accessing common DBMSes.

In the next section, we outline the evaluation criteria for a class of more recent emerging systems, denoted as *PSA$^+$ systems*, that aim to address (some of) these limitations and thus enhance the overall support for PSA applications.

4 PSA$^+$ criteria and methodology

In this section, we first define the PSA$^+$ system evaluation criteria, followed by our chosen survey methodology.

4.1 PSA$^+$ system evaluation criteria

We now propose a number of PSA$^+$ system evaluation criteria within the following three feature categories: *productivity features*, *technological features*, and *analytics features*.

Productivity features First, we have identified that a common thread among recent PSA$^+$ systems appears to be the focus on *developer productivity*. That is, efforts are targeted toward increased usability while also offering high-productivity features and providing a common easy-to-use framework for data analytics. We therefore intend to evaluate the contributions based on the three criteria—*workflow support*, *system extensibility*, and *language integration*, each taking the values shown in Table 2.

Technological features Second, we have identified basic properties for characterizing the more technological side of the advancements proposed in the literature. This includes the following criteria: *distributed computation* to characterize the potential for system scalability, *data independence* to characterize data flow optimization opportunities, and *implementation independence* to characterize auto-selection of algorithms and optimization opportunities.

Analytics features Lastly, we evaluate the levels of support for DA, PDA, and PSA tasks. For this, we define the following criteria: *descriptive primitives*, *predictive primitives*, and *optimization primitives*. These criteria describe whether, for each of the phases of BA, the fundamental analytics operations are supported natively in the system.

In the following sections, we will describe the proposed criteria and their properties in more detail.

4.2 Productivity features

Workflow support describes the capability of the system to support the user in the entire decision process, bringing the data and the results from one phase to the next while assisting in the natural iterative process of performing the analytics shown in Fig. 2. On the one end of the spectrum, we identify *algorithm-oriented* approaches, aiming at simplifying the development of analytics algorithms by separating the algorithm definition from the underlying data representation. For example, SystemML [35] proposes an *algorithm-oriented* approach, supporting low-level operations such as reading/writing data, iterations, matrix operations, and binary operations. *Task-oriented* approaches describe systems focusing on separating the analytics process from the algorithmic level, by providing the user with support for specific analytics applications, such as predictions or optimization. For example, MLBase [57] proposes a *task-oriented* approach, supporting high-level operations such as classification, features generation, and clustering. Finally, *process-oriented* approaches use high-level instructions and core optimization techniques to support the user *throughout* the PSA workflow tasks, from data to decisions.

System extensibility describes the possibility of extending the system's tool set/algorithms. Traditional solutions often follow a *closed-system* approach, where the user has no possibility of extending the system with custom tools, or where the only possibility is the connection to external programs. More advanced *extensible systems* instead provide interfaces for user extensions within the system itself. This type of approach allows the user to integrate custom algorithms within the same core architecture while taking advantage of the available ecosystem of existing algorithms.

Language integration describes the effort to reduce the number of languages required for PSA applications. While traditional systems use *multiple languages* for data management, analytics, and decision support, recent developments attempt to integrate these tasks in a single *unified language*, with the objective of reducing the developing time and cost. For example, in Tiresias [73], data management and manipulation are performed via standard SQL, while optimization problem modeling and solving are supported via a Datalog-based language. On the other hand, SolveDB [103] uses a single SQL-based language for both data management and optimization problem specification and solving. Improved productivity from a single declarative language is reported both for SolveDB [103] and for general big data analytics systems [70].

4.3 Technological features

Distributed computation describes whether the system allows the analytics computation to be effectively run in a distributed setting. The system language/interface hides from the user the distributed execution of algorithms, tasks, and processes.

Data independence describes whether the system can run a user application correctly irrespective of the physical organization of data. In traditional DBMSes, physical and logical data independence, as initially defined [50], shields the user application from changes to the physical organization of the data. Systems that bring this property to the analytics functions satisfy *data independence*. For example, SolveDB [103] fulfills the property by making query and solver implementations immune to the physical data organization and the data management optimizations performed.

Implementation independence determines whether the system decouples the high-level specification of an analytics application from its physical implementation. When the property is satisfied, results are correct and equivalent, independent of, for example, the chosen algorithm, operator implementation, and optimization strategy. Standard DBMSes already guarantee this property [50]. For example, in SystemML [35], the property is fulfilled via distributed/centralized deterministic operations, of which the algorithms are composed. In MADlib [43], this property is not guaranteed, as the algorithms are user-defined.

4.4 Analytics features

Descriptive primitives appear in traditional DBMSes and provide the user with basic primitives for data manipulation/-operations. The primitives are known by the system in order to generate execution plans for the operations. The user is provided with primitive operations that allow the implementation of BA tasks, algorithms, or processes. The operational semantics resulting from the standard primitives allows the system to reason about equivalences and cost of alternative execution strategies. *Descriptive primitives* refer to, for example, *relational algebra*, or built-in aggregation functions such as `GROUP BY` or `COUNT`, and OLAP operations.

Predictive primitives define whether the system is equipped with special primitives to manipulate predictive algorithms and models. For predictive *algorithms*, this includes primitives for linear algebra, matrix operations, statistical functions, whereas for predictive *tasks*, this includes specific model functions, such as *fit* and *predict*. The operational semantics of predictive primitives allows the system to evaluate execution plans for the operations supported by the primitives. An example of a system supporting *predictive primitives* is F^{2}DB [30], with declarative primitives for time series forecasting in SQL, e.g., `SELECT c AS OF time interval`.

Optimization primitives define whether the system is equipped with operations to specify and manipulate optimization models (problems), such as specifying objective variables, loss functions, objective functions, and constraints. Similar to *descriptive* and *predictive primitives*, the known semantics of the optimization primitives makes meta-optimization of the execution plan of the primitives possible. An example of a system offering *optimization primitives* is SolveDB [103], where `SOLVESELECT t(x) AS (...) MINIMIZE (SELECT sum(x) FROM t)...`, defines decision variables and the objective function of an optimization problem in SQL.

We next describe our literature survey methodology, which will form the basis for Sect. 5, where we present our categorization of the relevant literature, describe the different categories in detail, and classify the selected systems according to the criteria given in Sect. 4.1.

Table 3 Summary of the PSA$^+$ system comparison. WS—*workflow support*; SE—*system extensibility*; LI—*language integration*; DC—*distributed computation*; DI—*data independence*; II—*implementation independence*; DP—*descriptive primitives*; PP—*predictive primitives*; OP—*optimization primitives*

		System	WS	SE	LI	DC	DI	II	DP	PP	OP
Emerging PSA Systems	Analytical Frameworks	MLBase [56]	⊙	⨀	·	✓	✓	✓	✓	✓	✗
		SystemML [35]	·	⨀	·	✓	✓	✓	✓	✓	✗
		Tupleware [23]	⊙	⨀	·	✓	✗	✓	✗	✗	✗
		MADlib [42]	⊙	⨀	⨀	✓	✗	✗	✗	✗	✗
		Bismarck [28]	·	·	⨀	✗	✗	✗	✗	✗	✓
	Predictive DBMSes	LongView [3]	⊙	·	⨀	✗	✓	✓	✓	✓	✗
		SciDB [15]	·	⨀	·	✓	✓	✓	✓	✓	✗
		BayesDB [68]	⊙	·	⨀	✗	✓	✓	✓	✓	✗
		F^{2}DB [30]	⊙	⨀	⨀	✗	✓	✓	✓	✓	✗
	Optimization DBMSes	Tiresias [72]	⊙	·	·	✗	✓	✓	✓	✗	✓
		LogicBlox [4]	⊙	·	·	✗	✓	✓	✓	✗	✓
		PaQL [16]	⊙	·	⨀	✗	✓	✓	✓	✗	✓
		SolveDB [107]	⊙	⨀	⨀	✗	✓	✓	✓	✗	✓

4.5 Survey methodology and categorization

As already discussed in Sect. 1, the number of papers and systems focusing directly on PSA is limited. In our review, we therefore survey recent literature that, while not necessarily mentioning PSA explicitly, indirectly contributes to enhancing user PSA applications. First, we performed simple searches using Google Scholar to provide an overview of the existing work in PSA and to identify conferences, researchers in the field, and relevant keywords, e.g., recurring terms, common research topics, etc. Our research produced an initial pool of papers that have been used as a foundation for an exhaustive iterative structured search. In each iteration, we selected new papers deemed relevant to our PSA survey, adding them to our paper pool until the selection process converged to a point in which no more (relevant) papers could be found. The selection of papers was determined by the following criteria:

- Papers including the identified keywords.
- Papers whose title and abstract refer to the PSA characteristics defined in Sect. 4.1.
- Papers referenced by the papers in the current paper pool.
- Papers citing the papers in the current paper pool (citations found using Google Scholar).
- The publication history of each paper's author in the current paper pool.
- All conference proceedings or journal issues published after 2010 (included) in which the papers in the paper pool have been found. The 8 most common outlets were ACM SIGMOD, PVLDB, IEEE ICDE, TKDE, CIDR, DOLAP, Decision Support Systems, and Journal of Machine Learning Research.

Finally, we selected the systems proposed by the papers from the collected literature based on two constraints: (1) systems presented in papers published after 2010 (included), and (2) systems proposing advancements or directly addressing the aforementioned problems (see Sect. 3) of *limited language support*, *lack of high-productivity features*, and *lack of PSA workflow optimizations*. Only systems meeting both constraints were selected.

5 Emerging PSA$^+$ systems

In this section, we survey, evaluate, and compare a number of PSA$^+$ systems based on our system evaluation criteria and methodology.

5.1 Summary of emerging PSA$^+$ systems

Table 3 gives an overview of the thirteen systems we have evaluated and shows how the systems compare according to the different criteria specified in Sect. 4. In general, these contributions range from PSA-oriented architecture proposals over user programming/interaction interfaces to newly designed DBMSes for improved PSA applications. We have also found that the two main trends among all the contributions appear to be (1) strengthening the connection between analytics and data management and (2) declarative approaches for data analytics. In fact, all the emerging PSA$^+$

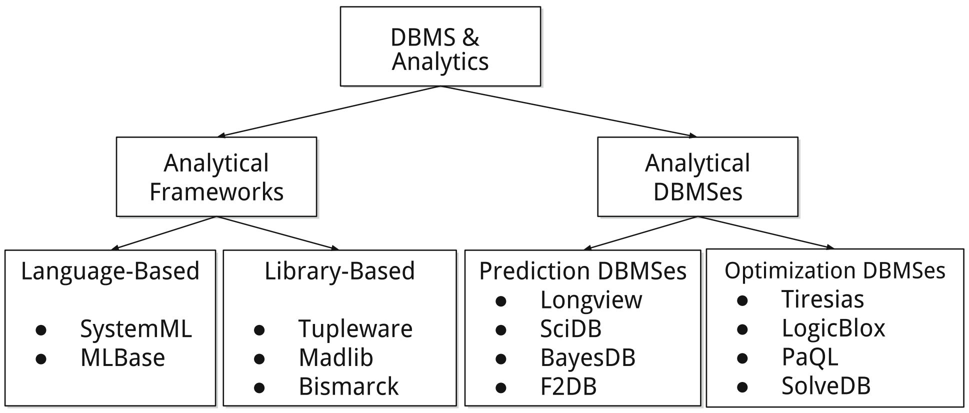

Fig. 4 Taxonomy of the PSA$^+$ systems

systems in Table 3 aim at integrating the DBMS with analytics framework technologies.

By looking at the contributions and focus points of the presented software systems and papers, we have synthesized a system taxonomy and grouped the systems according to this taxonomy. As seen in Fig. 4, we recognize a single root, denoted *DBMS & analytics*, encompassing the efforts of integrating DBMS and analytics framework functionalities. This root has two main system branches: *analytical frameworks* and *analytical DBMSes*. Analytical frameworks denote a class of systems that aim at unifying analytics tools with a data management layer in order to make the PSA process more efficient and *developer productivity* oriented. The frameworks aim at providing high-level specifications of analytics tasks or algorithms and provide a tighter connection between the data and analytics layers. Broader surveys of the systems in this category are available [12,59] (out of the scope of this paper). In this category, two sub-branches have already been identified [43]: *Language-based* and *Library-based*. In the other top branch, *analytical DBMSes* propose DBMS architectures integrating analytics functionality and BA support directly within the DBMS. This is achieved by extending the DBMS architecture, query language, and optimization techniques for in-DBMS analytics. On the one hand, we find systems proposing the idea of *predictive DBMSes*, by focusing on extending DBMSes with predictive algorithms and machine learning-focused languages. On the other hand, we have identified *optimization DBMSes*, systems aiming at integrating DBMSes with optimization solving capabilities.

We now survey the emerging PSA$^+$ systems in these categories.

5.2 Language-based analytical frameworks

Language-based analytical frameworks focus on high-level declarative programming languages to increase *developer productivity*. To support such languages, the frameworks deliver a data processing infrastructure for analytics processing. Examples of this category are SystemML [35] and MLBase [57].

SystemML provides a framework for the development of machine learning algorithms for both single node and distributed computation (MapReduce and Spark). SystemML introduces a declarative machine learning language (DML) with the objective of providing a framework that abstracts away the low-level details of distributed machine learning algorithms from the user. The solution proposed by SystemML follows an *algorithm-oriented* workflow approach, where the declarative support focuses on hiding the map reduce details from the user. DML allows the user to define machine learning algorithms based on *descriptive primitives* and *predictive primitives*, and iterative numerical optimization procedures [35]. By making the map reduce details transparent, DML allows the developer to perform distributed data analytics within a *unified language* framework. This approach satisfies both *data independence* and *implementation independence*, by exposing only the abstract data types frame, matrix, and scalar without their physical data structures [12].

In the same category of language-based frameworks, a different approach is presented by MLBase. Similarly to SystemML, MLBase provides a framework for DA and PDA *distributed* machine learning techniques on a map reduce architecture. However, contrary to SystemML's *algorithm-oriented* approach, MLBase introduces a syntax to specify *task-oriented* workflows [12]. This approach provides the user with high-level *descriptive primitives* and *predictive primitives*, in order to define standard analytics tasks such as `classify` or `predict`. By hiding both map reduce and algorithmic specification details in the underlying system, and with annotated algorithm characteristics and deterministic results, MLBase also satisfies the *data independence* and *implementation independence* properties. The translation from high-level tasks to map reduce operations is aided by techniques for selecting the choice of learning algorithm and by having the runtime execution optimized for the data processing of these tasks [57].

The two approaches, *algorithm-oriented* and *task-oriented*, show both advantages and disadvantages. On the one hand, both approaches satisfy *data independence* and *implementation independence* by providing the user with high-level primitives for a simplified specification of the analytics algorithms. On the other hand, the task of manually defining analytics algorithms is often viable only to programmers, thus limiting the impact of *algorithm-oriented* systems. *Task-oriented* systems provide a more coarse-grained scope, lowering the flexibility in exchange for developer productivity.

An example of a *task-oriented* approach is given by the simple PSA problem introduced in Sect. 2.6. For this problem, MLBase (for example) can be utilized to predict the probability with which the items will be sold in the future, guiding the shopkeeper to keep only the most probable ones. The shop possesses historical data related to the sales they have made in the past months. These data are organized in their DBMS, under the table `sales-facts`. In MLBase, the prediction task can be solved by declaratively calling a *prediction* function, which will find the best prediction model for the dataset to determine if an item will be sold or not. The code in Listing 1 shows an example of the specification for this problem in the MLBase language. While the variable X holds the model features from the DB (selected from the columns 2 to 10 with the *load* operation) and y the prediction labels (extracted from the first column, again with the *load* operation), `doPredict` selects and fits a model for predicting sales, saving the model and the results summary in the `fn-model` and `summary` variables. Using this approach, the prediction problem becomes a sequence of *tasks*.

Listing 1 Example of MLBase program for forecasting shop sales

```
var X = load("sales-facts",2 to 10)
var y = load("sales-facts",1)
var (fn-model, summary)=doPredict
  (X,y)
```

5.3 Library-based analytical frameworks

The second category of analytical frameworks is library-based frameworks, exemplified by MADlib [43], Bismarck [28], and TupleWare [23]. Library-based frameworks share the goal of providing a set of analytics blocks/tools, mainly focused on DA/PDA, together with library support for the user development process.

MADlib is an open-source library that collects a suite of SQL-based in-DBMS algorithms for DA and PDA, for both *centralized* and *distributed* computation architectures. MADlib follows a *task-oriented* approach, by introducing *User-Defined Aggregates* (UDAs) that can be utilized by the user as standard SQL aggregation functions, such as `SUM` or `COUNT`. UDAs can take advantage of the execution capabilities (e.g., multithreading, multiple nodes) of the DBMS, without requiring to modify the DBMS code to integrate them.

MADlib gives the user the possibility to extend the system with additional UDAs. To do this, MADlib provides an abstraction layer to facilitate the specification of UDAs and to encapsulate DBMS-specific logic inside the abstraction layer. By integrating the methods as SQL aggregate functions, MADlib succeeds at combining data management and data analytics tasks within the same *unified language* environment.

Tupleware proposes an approach similar to the one followed by MADlib, focusing on map reduce implementations for small clusters. Tupleware proposes a *task-oriented* approach, where the authors describe an architecture for automatic compilation of user-defined function (UDF) workflows for *in-DBMS analytics* methods. Tupleware presents an *extensible system*, supported by the possibility of developing the UDFs using generic programming languages. The architecture is based on the LLVM [61] compiler, providing a language-agnostic front end to allow the user to choose from different programming languages and to optimize UDF workflows at code generation [22]. However, both Tupleware and MADlib fail at satisfying the *data* and *implementation independence* properties as the UDAs/UDFs are implemented against custom data structures, and the operational semantics of UDAs/UDFs are by definition unknown in the system and not based on standard system primitives.

A different point of view is given by Feng et al. [28] with their Bismarck architecture. In their work, the authors advocate that the key bottleneck in the race for analytical DBMSes is that each new data analytics tool requires several expensive ad hoc steps every time it is installed in a new DBMS. This is caused by a lack of unification in data management architectures and algorithms. Common analytics tasks can be defined as convex programming problems [14], e.g., the learning task of machine learning algorithms often reduces to minimizing an error function while fitting a set of parameters to the model. The paper suggests that, since a number of statistical methods already fall into this category (e.g., support vector machines, logistic regression, and localized matrix factorization), the goal should be to unify the algorithmic diversities under the same theoretical framework. Thus, the Bismarck architecture attempts to unify in-DBMS analytics, providing a single level of abstraction for the definition of general-purpose optimization UDFs, by following an *algorithm-oriented* approach via *optimization primitives*.

On the one hand, the paper does not directly address PSA, and further research has to be conducted to verify that the broad range of algorithms and models required by PSA use cases fall into the convex programming problem category. On the other hand, the vision of designing a framework facilitating analytics extensions is a promising approach, which would allow both more control over the analytics process and easier implementation and use of the algorithms. Nevertheless, similar to the other UDF-based systems described so far, Bismarck does not satisfy the property of *data independence* and the Hogwild!-style [12,85] model updates deny the *implementation independence* property.

To conclude, the systems belonging to the analytical frameworks category present several advantages, proposing extensible and efficient frameworks with integrated DBMS and analytics capabilities, easy UDF implementation, and *declarative languages* for querying the models and the data. *Task-oriented* approaches also give the user an interface for easy implementation of the analytics workflow as high-level functions that can be combined to fulfill the analytics process, thereby possibly reducing the development of user PSA workflows to a combination of UDFs. Nevertheless, the systems in this category mostly focus on the preliminary tasks of DA and PDA, and do therefore not extend to the entire PSA workflow. Moreover, some of the systems do not satisfy the properties of *data independence* and *implementation independence*.

5.4 Predictive DBMSes

The systems in this branch propose extensions to well-known DBMSes with functionality addressing predictive tasks. While the approaches discussed in Sect. 5.2 provide analytical frameworks that can be used on top of the existing data management layers, predictive DBMSes focus on integrating the analytical tools directly into relational DBMSes. Representatives of this type of approach are Longview [3], SciDB [15], BayesDB [69], and F^{2}DB [30].

To start with an example, we consider again the shopkeeper problem. The query in Listing 2 is written in Bayesian Query Language (an extension of SQL), the language used in BayesDB. The query can be used to predict which items will most probably be sold. `INFER` (line 1) is used to declare a prediction query, as a generalization of the `SELECT` command, in which the results, the sold items, will be predicted in `inferred-sales` (line 2) by the use of the command `PREDICT`. In line 3, `CONFIDENCE` obtains the prediction confidence for each predicted sale, saved in `inferred-sales-confidence`. The SQL language gives the possibility to easily specify constraints on the predictions (line 5), where the prediction can be narrowed to items sold in the shops in Ohio. The resulting view, now containing the predicted data, can then be queried by the user via standard SQL queries to extract the results. As the reader can see, the code in Listing 2 can be used to solve the predictive phase of the problem, while the prescription (the optimization of the storage) still requires manual specification by the user.

Listing 2 Example of Bayesian Query Language (BQL) for predicting sales

```
1 INFER orderdate
2 PREDICT sales AS inferred-sales
3 CONFIDENCE inferred-sales-confidence
4 FROM sales-facts
5 WHERE state = 'Ohio'
```

In general, the systems belonging to the *predictive DBMSes* category present similar characteristics. The main goal of these systems is to integrate PDA techniques within a relational DBMS. The systems offer a wide range of common machine learning techniques, from clustering to classification and forecasting. As already noted in [58], the main contributions can be categorized in terms of model management, providing support for querying and model maintenance, feature engineering, algorithm selection, and parameter tuning. The systems support techniques for transparently selecting, processing, and maintaining the forecasting models; thus, the choice and use of the forecasting models can be kept hidden to the user. With the design of new *predictive query languages*, these systems allow the end user to easily apply PDA tools with a *declarative* and *task-oriented* approach. Moreover, the extension of the SQL language allows for seamless *unification* between the data processing and predictive tasks. The properties of *data independence* and *implementation independence* are satisfied by the underlying DBMS, and by the use of a standardized workflow architecture for model creation, maintenance, and usage. The systems present both *descriptive primitives*, inherited from the DBMS, and *predictive primitives*, by providing predictive task operations as first-class citizens in the DBMS.

Among the presented systems, F^{2}DB takes a narrower approach, focusing on integrating time series forecasting within a DBMS [29]. The authors of F^{2}DB argue that, in PDA applications and decision-making in general, one of the key statistical methods is time series forecasting, and that a deeper integration of these techniques will contribute to improving efficiency and usability in such use cases. F^{2}DB allows for *in-DBMS* time series forecasting, with an integrated SQL-based language to define the forecasting queries. F^{2}DB also provides the user with an *extensible interface* for integrating new algorithms.

5.5 Optimization DBMSes

Predictive DBMSes cover both DA and PDA, providing efficient and easy-to-use solutions for these phases. Nevertheless, predictions are only an intermediate step in the PSA workflow, and the optimization phase is not directly integrated into the workflow of the *predictive DBMSes*. Alternatively, in *optimization DBMSes*, we have grouped systems that specifically address mathematical optimization tasks. This group includes PaQL [16], Tiresias [73], SolveDB

[103], and LogicBlox [4] as representatives of DBMSes with integrated optimization problem solving capabilities. These systems provide the user with *optimization primitives* for linear programming (LP), mixed integer programming (MIP) [20], constraint programming (CP), global optimization, scheduling, etc. In this area, we have recognized two high-level approaches, based on the type of language provided to the user: (1) *Datalog-based* optimization DBMSes and (2), *SQL-based* optimization DBMSes. While both approaches aim at providing the user with declarative query languages, the choice between SQL and Datalog leads to either an extended SQL with PSA methods, or to the inclusion into the DBMS of a separate analytics-oriented language, in this case, Datalog.

Tiresias belongs to the class of Datalog-based *optimization DBMSes* by providing a system that can be interfaced with any relational DBMS.[1] In the shop sales scenario, a possible query could, for example, be *which items should be kept in storage in order to maximize the profit?* This is done by utilizing *hypothetical tables*, which form, together with the traditional DB tables, a Hypothetical DB (HDB). Hypothetical tables present the same schema of the traditional tables, with the addition of hypothetical columns that define the objective variables to optimize. Tiresias offers a new language based on Datalog, TiQL, with which it is possible to specify the hypothetical tables, the constraints, and the minimization/maximization objective. Recalling the storage optimization problem described before, the code in Listing 3 shows such an approach.

The program is divided into three parts, `HTABLE` specifies the hypothetical tables, `RULES` the optimization constraints, and `MAXIMIZE/MINIMIZE` the objective function. Intuitively, `HItemFacts` will be initialized as a hypothetical table, where `qnt?` (quantity) is *non-deterministically* set by the DBMS to comply with user-defined constraints. After having defined the specifications for the optimization problem in the section `RULES`, the system translates them into a mixed integer programming (MIP) problem, with the objective given in `MAXIMIZE`. The program is then handed to a MIP solver, which will output a solution that will be used to populate the HDB.

Listing 3 Example of TiQL program for the storage optimization problem.

```
HTABLE:
 HItemFacts(item, profit, qnt?) :-
  KEY(item,price)
RULES:
 HItemFacts(item, price, qnt?)  :-
  ItemFacts(item, price, qnt)
 [SUM(qnt?) <= 70]            :-
  HItemFacts(item, price, qnt?)
MAXIMIZE(SUM(profit*qnt?))
```

[1] At the time of publication [73], Tiresias has been tested only with PostgreSQL.

Similarly to the other declarative approaches described so far, the user does not directly specify optimization solving details. Besides the constraints and objective functions, it is the DBMS that selects a specific LP/MIP solver, applies the solving algorithm, and records a solution for the optimization problem by updating the hypothetical columns of the tables in the HDB. While in the case of *predictive DBMSes*, it was not possible to solve optimization problems, but only to find which items would most probably be sold in the future (as illustrated in Listing 2), Tiresias disregards the prediction phase and targets only the optimization task.

LogicBlox proposes another platform for integrating DA, PDA, and PSA in the same DBMS architecture. The intention is to expand the notion of database systems to include features found in programming languages, statistical systems, and mathematical optimization. LogicBlox extends DBMSes in a similar way as Tiresias, also introducing a new Datalog-based language, LogiQL, aimed at describing how-to queries. However, LogicBlox also supports forecasting techniques natively in the system. These are implemented as a collection of built-in machine learning algorithms and can be accessed via the creation of *statistical relational models*. These models are obtained by extensions of LogiQL supporting the modeling of Markov logic networks [86] and probabilistic soft logic (PSL) [34]. PSL models are specified via the use of *soft constraints*, rules similar to regular optimization constraints but holding continuous values instead of a binary acceptance condition.

Considering again the storage optimization problem, an example of the solution obtained with LogicBlox is shown in Listing 4. Here, the first part of the code (lines 1–9) specifies the problem of maximizing the profit under the storage space limit. LogiQL uses Datalog-like constraints, where in lines [1–5] the user defines the predicates that will be used in the optimization problem, such as how to calculate the profit for an item (line 1) or for all the items (line 5). Line 6 specifies the constraint of the storage space. Finally, line 8 describes the prescription problem by defining `Stock` as a free variable, which the system is responsible for populating while respecting the constraints in line 6 and the objective function in line 9.

Additionally, as LogicBlox allows for predictive modeling, it is possible to forecast the probability with which the items will be sold in the future, in order to optimize the storage in advance. In this case, the store could consider that a user will buy a promoted item (w_1), or an item in the same category of what she has already purchased (w_2), and will not buy an item too similar to what has already been purchased (w_3). Under this formalism, maximum a posteriori (MAP) inference can be used for finding the most likely possible world under the specified constraints. After having populated the tables with the predicted data, the optimization problem is again solvable as a how-to query similar to the one described in lines 1–9.

Listing 4 Example of a LogiQL program for an optimization storage problem (adapted from the original paper [4])

```
// BASE PREDICATES
1 profitItem[i]=v → Item(i),float(v)
3 Stock[i]=v → Item(i), float(v)
4 totalShelf[]+=Stock[i]
5 totalProfit[]+=profitItem[i]*Stock[i]
// RULES
6 totalShelf[] = u → u ≤ 70

// PRESCRIPTION CONSTRAINTS
8 lang:solve:variable(Stock)
9 lang:solve:max(totalProfit)

// PREDICTION CONSTRAINTS
w1:Customer(c),Promoted(i) → Purchase(c,i)
w2:Customer(c),Purchased(j),SameCategory(i,j)→
                                  Purchase(c,i)
w1:Customer(c),Purchased(i),Similar(i,j)→
                                 !Purchase(c,i)
```

The Datalog-based *optimization DBMSes* we have described focus on integrating *in-DBMS* optimization solving capabilities within a relational DBMS. Even though LogicBlox also allows for soft constraint programming to enable the user to specify prediction tasks, compared to the *predictive DBMSes* the support for PDA is limited by the number of problem classes that can be specified in the LogiQL language. Although the Datalog extensions offer a *declarative* and *task-oriented* approach to the user, they fail, however, at *unifying* data processing and data analytics, as they require the combination of SQL and a Datalog-based language. Finally, the systems do not yet support *extensibility* of the tools implemented in the architecture.

On the other branch of the tree, SQL-based *optimization DBMSes* propose to unify both data management and analytics layers under the same language, by extending standard SQL with additional constructs. Among these, PaQL [16] proposes a system for solving integer linear programming problems by adopting so-called *package queries*. Standard database queries follow the principle that each result tuple must satisfy a given set of constraints. However, PaQL advocates that, as many problems require a collection of result tuples to be evaluated over the constraints, rather than individual tuples, it is more efficient to handle the result set collectively as *packages*, i.e., a set of result tuples that describe the possible *worlds* which could solve the problem. PaQL thus offers a *declarative language*, based on SQL, to specify package queries, and a DBMS integrated system to solve such queries.

An example is shown in Listing 5, where we reuse the storage optimization example. Intuitively, the PACKAGE keyword describes that the result of the query will be the set of tuples from the schema `itemFacts` that collectively satisfy the constraint defined in `SUCH THAT` and according to the objective in `MINIMIZE`. As in Tiresias, PaQL does not directly support the prediction task; thus, the code in Listing 5 only solves the optimization part of the problem. While it is shown how PaQL query approximation techniques can scale to large datasets [16], the class of problems that can be handled is limited to integer linear programming.

Listing 5 Example of PaQL program for a storage problem

```
1 select PACKAGE(I) as P
2 from itemFacts r
3 SUCH THAT sum(stock) <= 70 and
4 MINIMIZE sum(profit*stock)
```

In the same category of SQL-based *optimization DBMSes*, SolveDB [94,103] offers a more general framework for solving optimization problems using an extensible infrastructure for integrating solvers for different classes of problems directly within a relational DBMS. SolveDB sees every prediction and decision problem as an instance of a special optimization problem, solvable within the framework. This is achieved with an interface for the use and extension of optimization problem solver modules, similar to what is provided by the analytics framework described in Sect. 5.2, and an SQL-based syntax for defining optimization problems.

Solver modules can be accessed by the user via a special SQL clause `SOLVESELECT`, which can potentially be embedded into more complex nested `SELECT` SQL queries. These solvers include pre-implemented optimization solvers from libraries such as GLPK [66] or CBC [21], or user-defined solvers (UDSs), installed in the DBMS as extensions. The extension interface can also be used to add predictive algorithms as UDSs, making use of the optimization machinery already provided in SolveDB to train the models' parameters. The specification of a PSA application thus reduces to a combination of optimization problems for which the user defines the objectives, constraints, and which of the installed solvers to apply.

As an example, to apply SolveDB to the same storage problem described above, we present a possible solution in Listing 6. The optimization query is specified by the clause `SOLVESELECT x IN` that defines `x` as a database table column with free variables that has to be populated according to the constraints in `SUBJECTTO` and the objective function in `MAXIMIZE`. The constraint specification consists of a series of `SELECT` statements in which the allowed values are specified. In the example, the query will use `stock` as a free variable, whereby the quantity of items will be chosen by the optimization solver. The only constraint is specified in line 5, where the sum of the items in `stock` must not exceed the maximum value of 70. The objective function is given in line 4, where the aggregation function `SUM(profit*stock)` calculates the profit of the items selected to be in storage. Finally, line 6 defines which solver module should be used for solving the resulting optimization problem, in this case `solverlp`.

Listing 6 Example of SolveDB program for solving the storage problem

```
SOLVESELECT stock in r as
(select itemID,profit,null::integer as stock
 from ItemFacts)
MAXIMIZE (select sum(profit*stock) from r)
SUBJECTTO (select sum(stock)<=70 from r)
USING solverlp()
```

The same query can also be specified as in Listing 7, where the logic is instead hard-coded in the `storageSolver` extension that can be installed by the user in the DBMS. This approach can be useful for more experienced users who want to use custom algorithms, to add domain knowledge for specific cases, and to encode predictive algorithms directly in the solver.

Listing 7 Example of use of a SolveDB user-defined solver for the storage problem

```
SOLVESELECT stock IN r AS
(select itemID,stock from ItemFacts)
USING storageSolver()
```

To conclude, all the *optimization DBMSes* we have presented exploit the natural declarative characteristics of relational DBMSes for analytics purposes, thus satisfying both *data independence* and *implementation independence*. These systems offer a *unified* and *declarative* approach to the query language by extending the standard SQL for solving optimization problems. The language syntax and the underlying optimization techniques enable the user to follow a *task-oriented* approach to the specification of optimization problem workflows. SolveDB also offers a generic interface for extending various types of solvers, which enables the specification of a wide range of optimization tasks and, by the use of UDSs, also predictive tasks. Nevertheless, *optimization DBMSes* have been designed mainly for optimization problem solving. While they excel in this scenario, PSA applications require the solution of more intermediate tasks currently lacking in the available *optimization DBMSes*, thus not providing the user with support for carrying out the full PSA process.

5.6 Discussion

In our evaluation, we have first identified traditional (classical) systems that have been used for many years for developing DA, PDA, and PSA applications. We have found that there exist many *BA tools* that typically target one (or a few) tasks within the full workflow of a typical PSA application. *BA suites* integrate functionalities of such individual BA tools into a single eco-environment, offering better end-to-end support for user applications, in particular DA and PDA. Despite the growing importance of DBMSes in decision-making and business applications [3], the *BA Tools* and *BA Suites* we have reviewed do not yet natively integrate data management technologies, forcing the user to utilize multiple software systems and multiple (often procedural, closed) languages. This causes user errors, poor developer productivity, reduced overall execution performance, and lack of user guidance throughout the PSA workflow. These traditional software systems use DBMSes only as a back-end data server, missing the advantages of a tight coupling between data and analytics for an improved overall performance and usability. Some commercial DBMSes (Oracle [99], SQL server [105], DB2 [47]) include simple analytics extensions, but they provide limited or no support for PSA.

There have, however, been attempts at providing much richer support for user PSA (and PDA) applications. We have provided a comparison of such PSA$^+$ systems that aims at providing support for user PSA applications by focusing on the aforementioned limitations of the classical systems. These are evaluated based on the criteria given in Sect. 4.1. As seen in our survey, the database community has already proposed new architectures to support solutions with integrated DBMS and analytics framework functionalities. Among these, analytical DBMSes and analytical frameworks improve user PSA applications by providing easier access to the required tools through a unified high-level languages aimed at increasing *developer productivity*. We have identified some important trends in this field, such as declarative approaches for PDA and PSA. Further, we have found that a few systems, described as *analytical DBMSes*, have combined a number of analytics tools inside the DBMS back-end itself, to be able to optimize mixed data management and analytics workloads while offering overall improved performance and a unified language for data management and analytics. However, as these systems were not originally designed for the totality of PSA, they are still far from being easily applied in PSA scenarios. Thus, a more integrated approach for PSA is needed.

Furthermore, we have seen how the analytics process has undergone a paradigm shift, where *declarative languages* have taken the place of traditional procedural approaches. All the emerging PSA$^+$ systems we have described in Table 3 support a declarative paradigm, as the difficulty encountered by the end users in designing algorithms and applications via traditional procedural languages has already been recognized as one of the major issues in the diffusion of PSA systems [11]. Nevertheless, while the different authors agree on having a declarative paradigm, there is no consensus about the concrete use of declarative approaches.

The first choice regards which type of declarative language best supports PSA applications. Among the selected systems, we have identified two main approaches, Datalog-based and SQL-based systems. Specifically, SQL-based *optimization DBMSes* attempt to integrate advanced analytics with traditional query processing, and at the same time to leverage the well-known SQL syntax for data operations. We see the argument for language unification as a compelling idea toward the simplification of PSA applications.

Second, the scope of PSA declarative paradigms is yet to be fully defined. Some of the analytical frameworks and analytical DBMSes we have reviewed propose the argument of *task-oriented* approaches. Others also allow the users to *extend* the system/frameworks with UDFs, providing ways to solve analytics applications as collections of domain-specific tools. Nevertheless, we find that the *task-oriented* declarative paradigms provided by most of the systems reviewed do not yet match the needs of PSA applications. When targeting specific predictive or optimization tasks, the systems successfully provide the user with declarative methods to specify the application workflow. However, in the case of full PSA applications, when multiple tasks from different BA phases have to be combined, the programmer lacks PSA *process-oriented* support for specific processes. In this case, the user has to fall back to procedurally defining each of the phases of the workflow. We will discuss the specific challenges and opportunities for developing the next-generation PSA$^+$ systems more extensively in the next section.

6 Challenges and opportunities

In this section, we summarize our findings and, based on the PSA system problems presented in Sect. 3, identify the three major challenges in developing the next generation of systems for PSA applications. For these challenges, we also describe opportunities available for PSA researchers and system engineers.

6.1 PSA language challenge

In general, declarative languages have been a huge success in data management and analytics, ranging from simple SQL/MDX (multidimensional expressions used for OLAP) to declarative data mining and machine learning languages [11,12]). We thus believe that a possible solution to some of the current PSA limitations consists in making PSA more *declarative*, especially for data engineers and PSA application developers; these groups of advanced users should, however, also be offered procedural and/or imperative constructs for specifying computations. The difference between procedural and declarative approaches is easily exemplified by comparing data retrieval before and after the introduction of SQL. In the pre-SQL DBMS era, developers had to program the complex data access procedures themselves, e.g., in CODASYL or hierarchical databases. With SQL, this has been both simplified and highly optimized. In comparison, most PSA tasks still need to be defined in a *pre-SQL* fashion [72].

While yet another analytics language is not a goal in itself, the advantages are imminent. For example, a PSA analytics language based on SQL could easily be integrated into existing DBMSes. It will thus find a large user base, without the need for BA developers to learn a completely new syntax. The PSA$^+$ systems (analytical DBMSes) we have reviewed demonstrate these advantages.

An equally important gap between an ideal PSA language and currently employed analytics languages is discussed in Sect. 5: Although the languages of current systems may follow a declarative paradigm, the full PSA workflow is not supported. For example, predictive DBMSes give the possibility of easily defining forecasting tasks, while optimization DBMSes can effectively solve optimization problems. However, to combine the two phases, which is needed in a full PSA application, the systems force users to follow a *procedural* approach, severely diminishing the advantages of the declarative languages. A powerful declarative unified language for the entire PSA workflow would enable a more effective, faster, and easier development of PSA applications. We thus define this as our first challenge:

– **Challenge 1 - How to develop effective languages for PSA applications?** *PSA systems need prescriptive-oriented languages with enough generality and expressivity to support the full variety of PSA tasks.*

A candidate language addressing this challenge should ensure an appropriate balance of language constructs as well as declarative and imperative primitives to cater the full range of users, ranging from analysts to PSA application and algorithm developers. At the same time, the language should support a wide range of relevant PSA application domains, while also offering *data and implementation independence* so that performance optimization is possible across the full PSA workflow. Furthermore, a number of advanced PSA language features should also be available to ensure an appropriate degree of support in the development process: First, *hybrid data* (structured, semi-, and unstructured) have been identified as one of the key pillars for PSA success [5]; hence, native language support for this type of data should be considered. The language should thus offer effective specialized primitives for accessing, manipulating, analyzing/mining, fusing, and integrating into PSA workflows a wide range of data types (e.g., documents, images, videos, JSON, graphs), similar to what *document stores* [18] do for data management alone.

Second, native support and treatment of (AI/prediction/simulation/optimization) *models* should be considered. Such models need to be managed as first-class citizens like the data itself. The language should offer effective model specification, manipulation (composition, decomposition), analysis, and processing primitives for a variety of model types supporting, e.g., prediction and optimization. These primitives could be intermixed with standard data management operations, giving the user increased flexibility when dealing with these models. Some of the reviewed systems, including MATLAB [71], R/SystemML [35], and SolveDB [103], offer such primitives to some degree.

Third, direct language support for *what-if scenarios* and/or *time travel* capabilities is crucial. What-if primitives would offer analysts an effective way of creating, analyzing, and comparing analytical results in case of hypothetical changes in input data and/or models without having to redefine the complete workflow. One reviewed system, Tiresias [73], offers limited support for such hypothetical scenarios. Further, time traveling primitives [55] enable effective evaluation of PSA workflows in the context of both historical data and predicted states/observations. Thus, users could conveniently travel forward and backward in time, while comparing DA, PDA, and PSA query results using both historical data and predicted/expected observations.

Lastly, new advanced types of queries that benefit from integrated *descriptive*, *predictive*, and *prescriptive* functionality, such as *package queries* [16] or advanced *exploratory queries* [52], should also be considered.

Of the above four contributions to increased language support, the first and third appear as the lowest hanging fruits from a conceptual point of view. There already exists previous work on how to deal with hybrid data, what-if scenarios, and time travel in more restricted data management-only scenarios. It thus appears likely that one could make progress by initially investigating how to best integrate these proposals within the concepts and constructs provided by the emerging PSA$^+$ systems. On the other hand, support for more advanced queries and native model support have seen less earlier work and represent bigger conceptual steps.

We have here considered the challenge of language functionality alone, but will look at the implementation and optimization of such languages in Sect. 6.2.

6.2 PSA system optimization challenges

Traditional BA applications have evolved around a myriad of technologies, growing into a complex software stack composed of many distinct tools [38]. While these technologies are highly optimized for their individual purposes, the structure and the nature of the complete PSA workflow are typically not exploited. This often leads to labor-intensive, cumbersome, poor performing ad hoc solutions, which are typically based on a single DBMS manually coupled with one or multiple analytical packages. As discussed in Sect. 5.6, to address these problems recent developments aim at marrying traditional data management and analytics to optimize and execute the whole PSA workflow in a single common runtime (back-end) system. This raises a challenge as well as opportunities:

– **Challenge 2 - How to optimize PSA workflows in a unified (PSA$^+$) data management and analytics system?** The aim is to offer the best result quality in the shortest execution time, where performance and result quality are complementary objectives.

In the context of this challenge, there are a number of opportunities for database researchers and practitioners. First, techniques for optimizing user-specified targets, while offering the most effective use of computational and network resources, are important, especially in a distributed setting with parallel execution of PSA workloads. In this setting, the aim is to find the most effective distribution and placement of data and analytics algorithms for an arbitrary user-given PSA workflow. When performance is desired, query execution/optimization techniques based on automatic analytical task partitioning, parameter tuning, data sampling, and progressive execution with fail-safe state snapshotting can significantly reduce execution time of CPU-intensive workloads. For input–output (IO)-intensive workloads, more traditional optimization techniques based on pipelining and streaming of analytical task input/output become relevant. When the accuracy of the results is key, optimization techniques involving more elaborate auto-selection of algorithms, test runs, and/or ensemble processing become prominent. Furthermore, in continuous online PSA applications, additional optimization based on result caching and warm-starting is possible.

Second, native support for models (treated as *white boxes*/first-class citizens) offers a number of optimization possibilities. For instance, automatic on-the-fly synthesis (compilation) of model management algorithms becomes possible. Such algorithms make the processing of a specific model instance much faster on a given hardware platform. An example of this optimization is the solver synthesis for symbolic nonlinear optimization models. Further, optimization techniques based on automatic model partitioning [103], composition/decomposition [29], aggregation [104,106], or approximation [102] become possible.

Lastly, additional PSA workflow optimization techniques based on analytical query rewriting, indexing, materialization, and use of new hardware (main memory, NVM, multicore, GPUs) are potentially feasible.

In terms of difficulty, this challenge is conceptually relatively easy, since the (optimization) problem is both well-specified and measurable. Thus, the challenge lies more in designing methods and techniques for these more complex workflows and queries. Optimization based on native model support is probably the hardest, but also the most interesting.

6.3 PSA user productivity challenge

Decades of research in PSA-related fields, e.g., statistical analysis, data mining, and machine learning, have led to a multitude of methods and tools for BA, many of which are based on sophisticated algorithms and complex mathematics [36,84,88]. The focus of these disciplines has mostly been on efficient and scalable algorithms, rather than studying the inner relationship between these methods and how to make them available to the users in the most accessible way [11].

Having access to more techniques does not necessarily translate into better applications for the end users, but often only increases complexity and confusion among the developers. In a typical PSA scenario, developers will end up choosing either well-known off-the-shelf algorithms that are not tailored for their specific tasks, leading to sub-optimal solutions, or more recent techniques for which the risk of misuse is higher. Furthermore, PSA applications often involve analytical tasks that require picking the right tools, inputs, and parameters for improving performance. Again, if not performed in a rigorous and well-informed manner, these practices can lead to misinterpretations of required inputs, parameters, and results [25,27]. While picking and fine-tuning the right techniques is complex in itself, things get even worse when different tools for different PSA phases need to be integrated, often resulting in long development cycles and low programmer productivity. A challenge is thus how to achieve the best user productivity in PSA systems while ensuring efficient and correct use of techniques. We identify this as our third challenge:

– **Challenge 3 - How to achieve high user productivity when developing PSA applications?** *PSA systems should offer effective tools and user support for all levels and tasks in the PSA development process, while ensuring that the best techniques and practices are chosen and used correctly.*

In the context of this challenge, there are a number of opportunities for system architects, developers, and system usability experts (UX). First, end-to-end PSA ecosystems with comprehensive tool packages ranging from scalable (big)data stores, over-advanced query processing and AI engines (supporting the aforementioned language features), to flexible integrated development environments (IDEs) and dashboards, need to be developed and customized to support the most typical PSA scenarios and application domains. For the most common PSA (sub-)tasks, models, and queries, *templates* and *wizards* need to be prepared and exposed to the users, e.g., via GUI-based process-, model-, and query builders. Where possible, problem and data specific (GUI-based) model and algorithm advisors/recommenders should be provided. Furthermore, such PSA systems should also offer support for developing *online* PSA applications (e.g., for energy flexibility management [31]), which are becoming quite common. In these applications, process measurements (and other data) are collected automatically, continuously, and in (near) real time (e.g., via sensors) and then immediately used in the next decision-making cycle. Among the reviewed systems, only BI suites such as MATLAB [71] and SAS [89] offer similar capabilities, but they lack a tight data management integration.

This challenge is perhaps the hardest of the three, since it does not concern (relatively) simple language constructs or (objectively measurable) system performance. Instead, it concerns the aspect of user productivity which is both quite fuzzy, inherently subjective with different users having different preferences, and very hard to measure. Thus, solving this challenge will require many trial-and-error iterations of designing tool support and having diverse users applying them in different scenarios.

7 Conclusion and future work

In this paper, we surveyed developments and trends in an emerging subfield of BA, called prescriptive analytics. We have presented an overview of the evolution of BA, from the traditional *descriptive analytics* (DA) and *predictive analytics* (PDA), to the more recent *prescriptive analytics* (PSA). As part of the survey, we described the typical decision-making workflow used in BA applications and identified tasks that are relevant for a particular type of analytics along with the technology requirements. We provided an overview of both established and emerging technologies that offer user support in the different phases of the PSA development process. Three major limitations of the existing established systems were identified (*limited language support*, *lack of high-productivity features*, and *lack of PSA workflow optimizations*), together with a number of criteria for evaluating more recent emerging systems (denote as PSA$^+$): *Workflow Support*, *System Extensibility*, *Language Integration* and on the properties of *Distributed Computation*, *Data Independence*, *Implementation Independence*, and *Descriptive, Predictive* and *Optimization* primitives. Finally, we surveyed, evaluated, and compared a number of recent PSA$^+$ systems in the areas of *analytical frameworks* and *analytical DBMSes* (including *prediction DBMSes* and *Optimization DBMSes*).

In general, the emerging PSA$^+$ systems we have surveyed attempt to solve the aforementioned limitations by combining specialized analytics tools with generic data management tools. These integrated systems often demonstrate the ability to outperform more ad hoc implementations based on a number of highly specialized analytics tools. We argue that, for successful PSA applications, the focus of the research in the coming years should be on a continued effort at combining analytics and data management tools, while offering new languages and language primitives, user productivity features, as well as mixed workflow optimizations encompassing the full PSA process. These observations have been condensed and presented as three distinct challenges: *How to develop effective languages for PSA applications*?, *How to optimize PSA workflows in a unified data management and analytics system?*, and *How to achieve high user productivity when developing PSA applications?*.

To conclude, PSA is not yet an established field. While the tasks and methods that characterize PSA applications have already been used in BA and decision-making, the discipline is, compared to the more established DA and PDA, still only affirming its separate identity. However, if future research is done along the presented directions, we will soon experience a wider adoption and use of PSA applications, together with a more well-understood and established PSA field.

Acknowledgements This research was supported in part by the MADE-AAU Project, the DiCyPS Project funded by Innovation Fund Denmark, and the GOFLEX Project funded by the EC under the Horizon 2020 Program.

References

1. Aalst, W.M.P.V.D.: Process Mining—Discovery, Conformance and Enhancement of Business Processes. Springer, Berlin (2011)
2. Abbena, E., Salamon, S., Gray, A.: Modern Differential Geometry of Curves and Surfaces with Mathematica. Chapman and Hall/CRC, Boca Raton (2017)
3. Akdere, M., Çetintemel, U., Riondato, M., Upfal, E., Zdonik, S.B.: The case for predictive database systems: opportunities and challenges. CIDR **2011**, 167–174 (2011)
4. Aref, M., ten Cate, B., Green, T.J., Kimelfeld, B., Olteanu, D., Pasalic, E., Veldhuizen, T.L., Washburn, G.: Design and implementation of the logicblox system. In: Proceedings of SIGMOD, pp. 1371–1382 (2015)
5. Basu, A.: Five pillars of prescriptive analytics success. Analyt. Mag. March-April (2013). http://analytics-magazine.org/executive-edge-five-pillars-of-prescriptiveanalytics-success/. Accessed 27 May 2019
6. Bertsimas, D., Kallus, N.: From predictive to prescriptive analytics. ArXiv e-prints (2014)
7. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: a fresh approach to numerical computing. SIAM Rev. **59**(1), 65–98 (2017)
8. Bihis, M., Roychowdhury, S.: A generalized flow for multiclass and binary classification tasks: an azure ml approach. In: 2015 IEEE International Conference on Big Data, pp. 1728–1737 (2015)
9. Birge, J.R., Louveaux, F.: Introduction to Stochastic Programming. Springer, Berlin (2011)
10. Bixby, R.E.: Solving real-world linear programs: a decade and more of progress. Oper. Res. **50**(1), 3–15 (2002)
11. Blockeel, H.: Data mining: from procedural to declarative approaches. New Gener. Comput. **33**(2), 115–135 (2015)
12. Boehm, M., Evfimievski, A.V., Pansare, N., Reinwald, B.: Declarative machine learning—a classification of basic properties and types. CoRR arXiv:abs/1605.05826 (2016)
13. Bonczek, R.H., Holsapple, C.W., Whinston, A.B.: Foundations of Decision Support Systems. Academic Press, London (2014)
14. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
15. Brown, P.G.: Overview of scidb: large scale array storage, processing and analysis. In: Proceedings of SIGMOD, pp. 963–968 (2010)
16. Brucato, M., Beltran, J.F., Abouzied, A., Meliou, A.: Scalable package queries in relational database systems. PVLDB **9**(7), 576–587 (2016)
17. Burstein, F., Holsapple, C.: Handbook on Decision Support Systems 2: Variations. Springer, Berlin (2008)
18. Chasseur, C., Li, Y., Patel, J.M.: Enabling JSON document stores in relational systems. WebDB **13**, 14–15 (2013)
19. Chaudhuri, S., Dayal, U.: An overview of data warehousing and OLAP technology. SIGMOD Rec. **26**(1), 65–74 (1997)
20. Chen, D.S., Batson, R.G., Dang, Y.: Applied Integer Programming: Modeling and Solution. Wiley, New York (2010)
21. COIN-OR: COIN-OR: Computational infrastructure for operations research—open-source software for the operations research community. https://www.coin-or.org/ (2018). Accessed 22 Mar 2018
22. Crotty, A., Galakatos, A., Dursun, K., Kraska, T., Binnig, C., Çetintemel, U., Zdonik, S.: An architecture for compiling udf-centric workflows. PVLDB **8**(12), 1466–1477 (2015)
23. Crotty, A., Galakatos, A., Dursun, K., Kraska, T., Çetintemel, U., Zdonik, S.B.: Tupleware: "big" data, big analytics, small clusters. In: CIDR 2015 (2015)
24. De Gooijer, J.G., Hyndman, R.J.: 25 years of time series forecasting. Int. J. Forecast. **22**(3), 443–473 (2006)
25. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
26. Desanctis, G., Gallupe, R.B.: A foundation for the study of group decision support systems. Manag. Sci. **33**(5), 589–609 (1987)
27. Dietterich, T.G.: Approximate statistical tests for comparing supervised classification learning algorithms. Neural Comput. **10**(7), 1895–1923 (1998)
28. Feng, X., Kumar, A., Recht, B., Ré, C.: Towards a unified architecture for in-RDBMS analytics. In: Proceedings of SIGMOD, pp. 325–336 (2012)
29. Fischer, U., Dannecker, L., Siksnys, L., Rosenthal, F., Böhm, M., Lehner, W.: Towards integrated data analytics: time series forecasting in DBMS. Datenbank-Spektrum **13**(1), 45–53 (2013)
30. Fischer, U., Rosenthal, F., Lehner, W.: F2DB: the flash-forward database system. In: IEEE 28th ICDE 2012, pp. 1245–1248 (2012)
31. Frazzetto, D., Neupane, B., Pedersen, T.B., Nielsen, T.D.: Adaptive user-oriented direct load-control of residential flexible devices. In: Proceedings of e-Energy, pp. 1–11 (2018)
32. Gartner: Flipping to Digital Leadership, Insights from the 2015 Gartner CIO Agenda Report (2015). https://www.gartner.com/imagesrv/cio/pdf/cio_agenda_insights2015.pdf. Accessed 21 Aug 2018
33. Gartner: Gartner's 2016 hype cycle for emerging technologies identifies three key trends that organizations must track to gain competitive advantage. https://www.gartner.com/newsroom/id/3412017 (2016). Accessed 22 Mar 2018
34. Getoor, L.: Introduction to Statistical Relational Learning. MIT Press, Cambridge (2007)
35. Ghoting, A., Krishnamurthy, R., Pednault, E.P.D., Reinwald, B., Sindhwani, V., Tatikonda, S., Tian, Y., Vaithyanathan, S.: SystemML: declarative machine learning on MapReduce. In: Proceedings of ICDE, pp. 231–242 (2011)
36. Gorunescu, F.: Data Mining—Concepts, Models and Techniques, Intelligent Systems Reference Library, vol. 12. Springer, Berlin (2011)

37. Goyal, A., Aprilia, E., Janssen, G., Kim, Y., Kumar, T., Mueller, R., Phan, D., Raman, A., Schuddebeurs, J.D., Xiong, J., Zhang, R.: Asset health management using predictive and prescriptive analytics for the electric power grid. IBM J. Res. Dev. **60**(1), 1–4 (2016)
38. Green, T.J., Aref, M., Karvounarakis, G.: Logicblox, platform and language: a tutorial. In: Proceedings of Datalog, pp. 1–8 (2012)
39. Gröger, C., Schwarz, H., Mitschang, B.: Prescriptive analytics for recommendation-based business process optimization. In: International Conference on Business Information Systems, pp. 25–37 (2014)
40. Gurobi Optimization LLC: Gurobi Optimizer (2014). http://www.gurobi.com/products/gurobi-optimizer. Accessed 7 May 2019
41. Haas, P.J., Maglio, P.P., Selinger, P.G., Tan, W.C.: Data is dead... without what-if models. PVLDB **4**(12), 1486–1489 (2011)
42. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. ACM SIGKDD Explor. Newslett. **11**(1), 10–18 (2009)
43. Hellerstein, J.M., Ré, C., Schoppmann, F., Wang, D.Z., Fratkin, E., Gorajek, A., Ng, K.S., Welton, C., Feng, X., Li, K., Kumar, A.: The madlib analytics library or MAD skills, the SQL. PVLDB **5**(12), 1700–1711 (2012)
44. High, R.: The Era of Cognitive Systems: An Inside Look at IBM Watson and How It Works. IBM Corporation, Redbooks (2012)
45. Holsapple, C.W., Lee-Post, A., Pakath, R.: A unified foundation for business analytics. Decis. Support Syst. **64**, 130–141 (2014)
46. Hupfeld, D., Maccioni, R., Sesemann, R., Ravazzolo, D.: Fleet asset capacity analysis and revenue management optimization using advanced prescriptive analytics. J. Revenue Pricing Manag. **15**(6), 516–522 (2016)
47. IBM: IBM DB2 database—database software: IBM analytics. https://www.ibm.com/analytics/us/en/db2/ (2018). Accessed 22 Mar 2018
48. IBM: Prescriptive analytics—IBM analytics. https://www.ibm.com/analytics/data-science/prescriptive-analytics (2018). Accessed 22 Mar 2018
49. Inmon, W.H.: Building the Data Warehouse. Wiley, New York (2005)
50. Jardine, D.A.: The ANSI/SPARC DBMS Model; Proceedings of the Second Share Working Conference on Data Base Management Systems, Montreal, Canada, April 26–30, 1976. Elsevier Science Inc., Amsterdam (1977)
51. Jarke, M., Lenzerini, M., Vassiliou, Y., Vassiliadis, P.: Fundamentals of Data Warehouses. Springer, Berlin (2013)
52. Kalinin, A., Cetintemel, U., Zdonik, S.: Searchlight: enabling integrated search and exploration over large multidimensional data. Proc. VLDB Endow. **8**(10), 1094–1105 (2015)
53. Kaur, J., Mann, K.S.: AI based healthcare platform for real time, predictive and prescriptive analytics using reactive programming. J. Phys. Conf. Ser. **933**, 012010 (2018)
54. Keen, P.G., Morton, M.S.S.: Decision Support Systems: An Organizational Perspective, vol. 35. Addison-Wesley, Reading (1978)
55. Khalefa, M.E., Fischer, U., Pedersen, T.B., Lehner, W.: Model-based integration of past and future in timetravel. Proc. VLDB Endow. **5**(12), 1974–1977 (2012)
56. Kimball, R., Ross, M.: The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling. Wiley, New York (2011)
57. Kraska, T., Talwalkar, A., Duchi, J.C., Griffith, R., Franklin, M.J., Jordan, M.I.: Mlbase: a distributed machine-learning system. In: Proceedings of CIDR (2013)
58. Kumar, A., McCann, R., Naughton, J., Patel, J.M., Babros, T.E., Hunt, R.J., Koski, K., Strikwerda, J.C., Wade, B.A., Arnold, R.B., et al.: A survey of the existing landscape of ml systems. UW-Madison CS Tech. Rep. TR1827 (2015)
59. Kumar, A., McCann, R., Naughton, J.F., Patel, J.M.: Model selection management systems: the next frontier of advanced analytics. SIGMOD Rec. **44**(4), 17–22 (2015)
60. Laborie, P., Rogerie, J., Shaw, P., Vilím, P.: IBM ILOG CP optimizer for scheduling. Constraints **23**(2), 210–250 (2018)
61. Lattner, C., Adve, V.S.: LLVM: a compilation framework for lifelong program analysis and transformation. In: 2nd IEEE ACM CGO, pp. 75–88 (2004)
62. Linoff, G.S., Berry, M.J.: Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management. Wiley, New York (2011)
63. Luhn, H.P.: A business intelligence system. IBM J. Res. Dev. **2**(4), 314–319 (1958)
64. Lustig, I., Dietrich, B., Johnson, C., Dziekan, C.: The analytics journey. Analyt. Mag. **3**(6), 11–13 (2010)
65. Madsen, A.L., Jensen, F., Kjærulff, U., Lang, M.: The hugin tool for probabilistic graphical models. Int. J. Artif. Intell. Tools **14**(3), 507–544 (2005)
66. Makhorin, A.: The GNU linear programming kit (GLPK). GNU Software Foundation (2015). https://www.gnu.org/software/glpk/. Accessed 7 May 2019
67. Makridakis, S., Wheelwright, S.C., Hyndman, R.J.: Forecasting Methods and Applications. Wiley, New York (2008)
68. Malinowski, E., Zimányi, E.: Advanced Data Warehouse Design—From Conventional to Spatial and Temporal Applications. Data-Centric Systems and Applications. Springer, Berlin (2008)
69. Mansinghka, V.K., Tibbetts, R., Baxter, J., Shafto, P., Eaves, B.: Bayesdb: a probabilistic programming system for querying the probable implications of data. CoRR arXiv:abs/1512.05006 (2015)
70. Markl, V.: Breaking the chains: on declarative data analysis and data independence in the big data era. PVLDB **7**(13), 1730–1733 (2014)
71. MathWorks: Matlab—mathworks. https://www.mathworks.com/products/matlab.html (2018). Accessed 22 Mar 2018
72. Meliou, A., Gatterbauer, W., Suciu, D.: Reverse data management. PVLDB **4**(12), 1490–1493 (2011)
73. Meliou, A., Suciu, D.: Tiresias: the database oracle for how-to queries. In: Proceedings of SIGMOD, pp. 337–348 (2012)
74. Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., et al.: Mllib: machine learning in apache spark. J. Mach. Learn. Res. **17**(1), 1235–1241 (2016)
75. Microsoft: Microsoft excel 2016, spreadsheet software, excel free trial. https://products.office.com/en-us/excel (2018). Accessed on 22 Mar 2018
76. Nagabhushana, S.: Data Warehousing OLAP and Data Mining. New Age International, Chennai (2006)
77. Nechifor, S., Puiu, D., Tarnauca, B., Moldoveanu, F.: Prescriptive analytics based autonomic networking for urban streams services provisioning. In: 2015 IEEE 81st Vehicular Technology Conference (VTC Spring), pp. 1–5 (2015)
78. Neupane, B., Pedersen, T.B., Thiesson, B.: Utilizing device-level demand forecasting for flexibility markets. In: Proceedings of e-Energy, pp. 108–118 (2018)
79. Neupane, B., Šikšnys, L., Pedersen, T.B.: Generation and evaluation of flex-offers from flexible electrical devices. In: Proceedings of e-Energy, pp. 143–156 (2017)
80. Owen, S., Anil, R., Dunning, T., Friedman, E.: Mahout in action. Manning Publications Co, Shelter Island, NY (2011)
81. Power, D.J., Sharda, R., Burstein, F.: Decision Support Systems. Wiley, New York (2015)
82. Powers, C.A., Meyer, C.M., Roebuck, M.C., Vaziri, B.: Predictive modeling of total healthcare costs using pharmacy claims data: a

comparison of alternative econometric cost modeling techniques. Med. Care **43**(11), 1065–1072 (2005)
83. Pritchard, P.J., Pritchard, R.: MathCAD: A Tool for Engineering Problem Solving (BEST Series). McGraw-Hill Higher Education, New York (1998)
84. Ramakrishnan, R., Gehrke, J.: Database Management Systems, 3rd edn. McGraw-Hill, New York (2003)
85. Recht, B., Re, C., Wright, S., Niu, F.: Hogwild: a lock-free approach to parallelizing stochastic gradient descent. In: Proceedings of the 25th Annual Conference on Neural Information Processing Systems, pp. 693–701 (2011)
86. Richardson, M., Domingos, P.M.: Markov logic networks. Mach. Learn. **62**(1–2), 107–136 (2006)
87. Rusitschka, S., Doblander, C., Goebel, C., Jacobsen, H.A.: Adaptive middleware for real-time prescriptive analytics in large scale power systems. In: Proceedings of Middleware, p. 5 (2013)
88. Russell, S.J., Norvig, P., Canny, J.F., Malik, J.M., Edwards, D.D.: Artificial Intelligence: A Modern Approach, vol. 2. Prentice Hall, Upper Saddle River (2003)
89. SAS: SAS business analytics—SAS. https://www.sas.com/en_us/solutions/business-analytics.html (2018). Accessed 22 Mar 2018
90. Sauter, V.L.: Decision Support Systems for Business Intelligence. Wiley, New York (2014)
91. Shim, J.P., Warkentin, M., Courtney, J.F., Power, D.J., Sharda, R., Carlsson, C.: Past, present, and future of decision support technology. Decis. Support Syst. **33**(2), 111–126 (2002)
92. Siegel, E.: Predictive Analytics: The Power to Predict Who Will Click, Buy, Lie, or Die. Wiley, New York (2013)
93. Šikšnys, L., Pedersen, T.B.: Prescriptive analytics. In: Encyclopedia of Database Systems, 2nd ed. Springer, New York, NY (2018). https://doi.org/10.1007/978-1-4614-8265-9_80624
94. Šikšnys, L., Pedersen, T.B.: Demonstrating solveDB: an SQL-based DBMS for optimization applications. In: Proceedings of ICDE, pp. 1367–1368 (2017)
95. Smet, G.D.: A decade of optaplanner. https://www.optaplanner.org/blog/2016/08/07/ADecadeOfOptaPlanner.html (2016). Accessed 01 Sept 2018
96. Soltanpoor, R., Sellis, T.: Prescriptive analytics for big data. In: Databases Theory and Applications—27th Australasian Database Conference, pp. 245–256 (2016)
97. Song, S., Kim, D.J., Hwang, M., Kim, J., Jeong, D., Lee, S., Jung, H., Sung, W.: Prescriptive analytics system for improving research power. In: 16th IEEE CSE, pp. 1144–1145 (2013)
98. Souza, G.C.: Supply chain analytics. Bus. Horiz. **57**(5), 595–605 (2014)
99. Stackowiak, R., Rayman, J., Greenwald, R.: Oracle Data Warehousing and Business Intelligence SO. Wiley, New York (2007)
100. Steinhaus, S.: Comparison of mathematical programs for data analysis. http://www.cybertester.com/data/ncrunch4.pdf (2008). Accessed 24 Aug 2018
101. Šikšnys, L.: Towards prescriptive analytics in cyber-physical systems. Ph.D. thesis, Aalborg University and Dresden University of Technology (2015)
102. Šikšnys, L., Pedersen, T.B.: Dependency-based flexoffers: scalable management of flexible loads with dependencies. In: Proceedings of e-Energy, pp. 11:1–11:13 (2016)
103. Šikšnys, L., Pedersen, T.B.: Solvedb: integrating optimization problem solvers into SQL databases. In: Proceedings of SSDBM, pp. 14:1–14:12 (2016)
104. Šikšnys, L., Valsomatzis, E., Hose, K., Pedersen, T.B.: Aggregating and disaggregating flexibility objects. TKDE **27**(11), 2893–2906 (2015)
105. Tang, Z., Maclennan, J.: Data Mining with SQL Server 2005. Wiley, New York (2005)
106. Valsomatzis, E., Pedersen, T.B., Abell, A., Hose, K.: Aggregating energy flexibilities under constraints. In: Proceedings of SmartGridComm, pp. 484–490 (2016)
107. Van Poucke, S., Thomeer, M., Heath, J., Vukicevic, M.: Are randomized controlled trials the (g) old standard? From clinical intelligence to prescriptive analytics. J. Med. Internet Res. **18**(7), e185 (2016)
108. Vanderbei, R.J.: Linear Programming. Springer, Berlin (2014)
109. Waller, M.A., Fawcett, S.E.: Data science, predictive analytics, and big data: a revolution that will transform supply chain design and management. J. Bus. Logist. **34**(2), 77–84 (2013)
110. Watkins, E.R.: Principles of the business rule approach: Ronald G. Ross, Addison-Wesley information technology series, february 2003, 256pp., price £30.99, ISBN 0-201-78893-4. Int. J. Inf. Manag. **24**(2), 196–197 (2004)
111. Winston, W.L., Goldberg, J.B.: Operations Research: Applications and Algorithms, vol. 3. Thomson/Brooks/Cole, Belmont (2004)
112. Wu, P.J., Yang, C.K.: The green fleet optimization model for a low-carbon economy: a prescriptive analytics. ICASI **2017**, 107–110 (2017)