Managing Big Data Analytics Workflows with a Database System

Carlos Ordonez
University of Houston
USA

Javier García-García
UNAM University
Mexico

***Abstract*—A big data analytics workflow is long and complex, with many programs, tools and scripts interacting together. In general, in modern organizations there is a significant amount of big data analytics processing performed outside a database system, which creates many issues to manage and process big data analytics workflows. In general, data preprocessing is the most time-consuming task in a big data analytics workflow. In this work, we defend the idea of preprocessing, computing models and scoring data sets inside a database system. In addition, we discuss recommendations and experiences to improve big data analytics workflows by pushing data preprocessing (i.e. data cleaning, aggregation and column transformation) into a database system. We present a discussion of practical issues and common solutions when transforming and preparing data sets to improve big data analytics workflows. As a case study validation, based on experience from real-life big data analytics projects, we compare pros and cons between running big data analytics workflows inside and outside the database system. We highlight which tasks in a big data analytics workflow are easier to manage and faster when processed by the database system, compared to external processing.**

I. Introduction

In a modern organization transaction processing [4] and data warehousing [6] are managed by database systems. On the other hand, big data analytics projects are a different story. Despite existing data mining [5], [6] functionality offered by most database systems many statistical tasks are generally performed outside the database system [7], [6], [15]. This is due to the existence of sophisticated statistical tools and libraries, the lack of expertise of statistical analysts to write correct and efficient SQL queries, a somewhat limited set of statistical algorithms and techniques in the database system (compared to statistical packages) and abundance of legacy code (generally well tuned and difficult to rewrite in a different language). In such scenario, users exploit the database system just to extract data with ad-hoc SQL queries. Once large tables are exported they are summarized and further transformed depending on the task at hand, but outside the database system. Finally, when the data set has the desired variables (features), statistical models are computed, interpreted and tuned. In general, preprocessing data for analysis is the most time consuming task [5], [16], [17] because it requires cleaning, joining, aggregating and transforming files, tables to obtain "analytic" data sets appropriate for cube or machine learning processing. Unfortunately, in such environments manipulating data sets outside the database system creates many data management issues: data sets must be recreated and re-exported every time there is a change, models need to be imported back into the database system and then deployed, different users may have inconsistent versions of the same data set in their own computers, security is compromised. Therefore, we defend the thesis that it is better to transform data sets and compute statistical models inside the database system. We provide evidence such approach improves management and processing of big data analytics workflows. We argue database system-centric big data analytics workflows are a promising processing approach for large organizations.

The experiences reported in this work are based on analytics projects where the first author participated as as developer and consultant in a major DBMS company. Based on experience from big data analytics projects we have become aware statistical analysts cannot write correct and efficient SQL code to extract data from the database system or to transform their data sets for the big data analytics task at hand. To overcome such limitations, statistical analysts use big data analytics tools, statistical software, data mining packages to manipulate and transform their data sets. Consequently, users end up creating a complicated collection of programs mixing data transformation and statistical modeling tasks together. In a typical big data analytics workflow most of the development (programming) effort is spent on transforming the data set: this is the main aspect studied in this article. The data set represents the core element in a big data analytics workflow: all data transformation and modeling tasks must work on the entire data set. We present evidence running workflows entirely inside a database system improves workflow management and accelerates workflow processing.

The article is organized as follows. Section II provides the specification of the most common big data analytics workflows. Section III discusses practical issues and solutions when preparing and transforming data sets inside a database system. We contrast processing workflows inside and outside the database system. Section IV compares advantages and time performance of processing big data analytics workflows inside the database system and outside exploiting with external data mining tools. Such comparisons are based on experience from real-life data mining projects. Section V discusses related work. Section VI presents conclusions and research issues for future work.

II. Big Data Analytics Workflows

We consider big data analytics workflows in which Task i must be finished before Task $i + 1$. Processing is mostly sequential from task to task, but it is feasible to have cycles from Task i back to Task 1 because data mining is an iterative process. We distinguish two complementary big data analytics workflows: (1) computing statistical models; (2) scoring data sets based on a model. In general, statistical models are computed on a new data set, whereas scoring takes place after the model is well understood and an acceptable data mining has been produced.

A typical big data analytics workflow to compute a model consists of the following major tasks (this basic workflow can vary depending on users knowledge and focus):

1) Data preprocessing: Building data sets for analysis (selecting records, denormalization and aggregation)
2) Exploratory analysis (descriptive statistics, OLAP)
3) Computing statistical models
4) Tuning models: add, modify data set attributes, feature/variable selection, train/test models
5) Create scoring scripts based on selected features and best model

This modeling workflow tends to be linear (Task i executed before Task $i+1$), but in general it has cycles. This is because there are many iterations building data sets, analyzing variable behavior and computing models before and acceptable can be obtained. Initially the bottleneck of the workflow is data preprocessing. Later, when the data mining project evolves most workflow processing is testing and tuning the desired model. In general, the database system performs some denormalization and aggregation, but it is common to perform most transformations outside the database system. When features need to be added or modified queries need to be recomputed, going back to Task 1. Therefore, only a part of the first task of the workflow is computed inside the database system; the remaining stages of the workflow are computed with external data mining tools.

The second workflow we consider is actually deploying a model with arriving (new) data. This task is called scoring [7], [12]. Since the best features and the best model are already known this is generally processed inside the database system. A scoring workflow is linear and non-iterative: Processing starts with Task 1, Task i is executed before Task $i + 1$ and processing ends with the last task. In a scoring workflow tasks are as follows:

1) Data preprocessing: Building data set for scoring (select records, compute best features).
2) Deploy model on test data set (e.g predict class or target value), store model output per record as a new table or view in the database system.
3) Evaluate queries and produce summary reports on scored data sets joined with source tables. Explain models by tracing data back to source tables in the database system.

Building the data set tends to be most important task in both workflows since it requires gathering data from many source tables. In this work, we defend the idea of preprocessing big data inside a database system. This approach makes sense only when the raw data originates from a data warehouse. That is, we do not tackle the problem of migrating processing of external data sets into the database system.

III. Tasks to Prepare Data Sets

We discuss issues and recommendations to improve the data processing stage in a big data analytics workflow. Our main motivation is to bypass the use of external tools to perform data preprocessing, using the SQL language instead.

A. Reasons for Processing big data analytics workflows Outside the database system

In general, the main reason is of a practical nature: users do not want to translate existing code working on external tools. Commonly such code has existed for a long time (legacy programs), it is extensive (there exist many programs) and it has been debugged and tuned. Therefore, users are reluctant to rewrite it in a different language, given associated risks. A second complaint is that, in general, the database system provides elementary statistical functionality, compared to sophisticated statistical packages. Nevertheless, such gap has been shrinking over the years. As explained before, in a data mining environment most user time is spent on preparing data sets for analytic purposes. We discuss some of the most important database issues when tables are manipulated and transformed to prepare a data set for data mining or statistical analysis.

B. Main Data Preprocessing Tasks

We consider three major tasks:

1) selecting records for analysis (filtering)
2) denormalization (including math transformation)
3) aggregation

Selecting Records for Analysis: Selection of rows can be done at several stages, in different tables. Such filtering is done to discard outliers, to discard records with a significant missing information content (including referential integrity [14]), to discard records whose potential contribution to the model provides no insight or sets of records whose characteristics deserve separate analysis. It is well known that pushing selection is the basic strategy to accelerate SPJ (select-project-join) queries, but it is not straightforward to apply into multiple queries. A common solution we have used is to perform as much filtering as possible on one data set. This makes code maintenance easier and the query optimizer is able to exploit filtering predicates as much as possible.

In general, for a data mining task it is necessary to select a set of records from one of the largest tables in the database based on a date range. In general, this selection requires a scan on the entire table, which is slow. When there is a secondary index based on date it is more efficient to select rows, but it is not always available. The basic issue is that such transaction table is much larger compared to other tables in the database. For instance, this time window defines a set of active customers or bank accounts that have recent activity. Common solutions to this problem include creating materialized views (avoiding join recomputation on large tables) and lean tables with primary keys of the object being analyzed (record, product, etc) to act as filter in further data preparation tasks.

Denormalization: In general, it is necessary to gather data from many tables and store data elements in one place. It is well-known that on-line transaction processing (OLTP) database systems update normalized tables. Normalization makes transaction processing faster and ACID [4] semantics are easier to ensure. Queries that retrieve a few records from normalized tables are relatively fast. On the other hand, analysis on the database requires precisely the opposite: a large set of records is required and such records gather information from many tables. Such processing typically involves complex queries involving joins and aggregations. Therefore, normalization works against efficiently building data sets for analytic purposes. One solution is to keep a few key denormalized tables from which specialized tables can be built. In general, such tables cannot be dynamically maintained because they involve join computation with large tables. Therefore, they are periodically recreated as a batch process. The query shown below builds a denormalized table from which several aggregations can be computed (e.g. similar to cube queries).

```
SELECT
  customer_id
 ,customer_name
 ,product.product_id
 ,product_name
 ,department.department_id
 ,department_name
FROM sales
   JOIN product
     ON sales.product_id=product.product_id
   JOIN department
     ON product.department_id
        =department.department_id;
```

For analytic purposes it is always best to use as much data as possible. There are strong reasons for this. Statistical models are more reliable, it is easier to deal with missing information, skewed distributions, discover outliers and so on, when there is a large data set at hand. In a large database with tables coming from a normalized database being joined with tables used in the past for analytic purposes may involve joins with records whose foreign keys may not be found in some table. That is, natural joins may discard potentially useful records. The net effect of this issue is that the resulting data set does not include all potential objects (e.g. records, products). The solution is define a universe data set containing all objects gathered with union from all tables and then use such table as the fundamental table to perform outer joins. For simplicity and elegance, left outer joins are preferred. Then left outer joins are propagated everywhere in data preparation and completeness of records is guaranteed. In general such left outer joins have a "star" form on the joining conditions, where the primary key of the master table is left joined with the primary keys of the other tables, instead of joining them with chained conditions (FK of table T1 is joined with PK of table T2, FK of table T2 is joined with PK of T3, and so on). The query below computes a global data set built from individual data sets. Notice data sets may or may not overlap each other.

```
SELECT
 ,record_id
 ,T1.A1
 ,T2.A2
 ..
 ,Tk.Ak
FROM T_0
   LEFT JOIN T1 ON T_0.record_id= T1.record_id
   LEFT JOIN T2 ON T_0.record_id= T2.record_id
   ..
   LEFT JOIN Tk ON T_0.record_id= Tk.record_id;
```

Aggregation: Unfortunately, most data mining tasks require dimensions (variables) that are not readily available from the database. Such dimensions typically require computing aggregations at several granularity levels. This is because most columns required by statistical or machine learning techniques require measures (or metrics), which translate as sums or counts computed with SQL. Unfortunately, granularity levels are not hierarchical (like cubes or OLAP [4]) making the use of separate summary tables necessary (e.g. summarization by product or by customer, in a retail database). A straightforward optimization is to compute as many dimensions in the same statement exploiting the same group-by clause, when possible. In general, for a statistical analyst it is best to create as many variables (dimensions) as possible in order to isolate those that can help build a more accurate model. Then summarization tends to create tables with hundreds of columns, which make query processing slower. However, most state-of-the-art statistical and machine learning techniques are designed to perform variable (feature) selection [7], [10] and many of those columns end up being discarded.

A typical query to derive dimensions from a transaction table is as follows:

```
SELECT
  customer_id
 ,count(*) AS cntItems
 ,sum(salesAmt) AS totalSales
 ,sum(case when salesAmt<0 then 1 end)
  AS cntReturns
FROM sales
GROUP BY customer_id;
```

Transaction tables generally have two or even more levels of detail, sharing some columns in their primary key. The typical example is store transaction table with individual items and the total count of items and total amount paid. This means that many times it is not possible to perform a statistical analysis only from one table. There may be unique pieces of information at each level. Therefore, such large tables need to be joined with each other and then aggregated at the appropriate granularity level, depending on the data mining task at hand. In general, such queries are optimized by indexing both tables on their common columns so that hash-joins can be used.

Combining Denormalization and Aggregation: Multiple primary keys: A last aspect is considering aggregation and denormalization interacting together when there are multiple primary keys. In a large database different subsets of tables have different primary keys. In other words, such tables are not compatible with each other to perform further aggregation. The key issue is that at some point large tables with different primary keys must be joined and summarized. Join operations will be slow because indexing involves foreign keys with large cardinalities. Two solutions are common: creating a secondary index on the alternative primary key of the largest table, or creating a denormalized table having both primary keys in order to enable fast join processing.

For instance, consider a data mining project in a bank that requires analysis by *customer id*, but also *account id*. One customer may have multiple accounts. An account may have multiple account holders. Joining and manipulating such tables is challenging given their sizes.

C. Workflow Processing

Dependent SQL statements: A data transformation script is a long sequence of SELECT statements. Their dependencies are complex, although there exists a partial order defined by the order in which temporary tables and data sets for analysis are created. To debug SQL code it is a bad idea to create a single query with multiple query blocks. In other words, such SQL statements are not amenable to the query optimizer because they are separate, unless it can keep track of historic usage patterns of queries. A common solution is to create intermediate tables that can be shared by several statements. Those intermediate tables commonly have columns that can later be aggregated at the appropriate granularity levels.

Computer resource usage: This aspect includes both disk and CPU usage, with the second one being a more valuable resource. This problem gets compounded by the fact that most data mining tasks work on the entire data set or large subsets from it. In an active database environment running data preparation tasks during peak usage hours can degrade performance since, generally speaking, large tables are read and large tables are created. Therefore, it is necessary to use workload management tools to optimize queries from several users together. In general, the solution is to give data preparation tasks a lower priority than the priority for queries from interactive users. On a longer term strategy, it is best to organize data mining projects around common data sets, but such goal is difficult to reach given the mathematical nature of analysis and the ever changing nature of variables (dimensions) in the data sets.

Comparing views and temporary tables: Views provide limited control on storage and indexing. It may be better to create temporary tables, especially when there are many primary keys used in summarization. Nevertheless, disk space usage grows fast and such tables/views need to be refreshed when new records are inserted or new variables (dimensions) are created.

Scoring: Even though many models are built outside the database system with statistical packages and data mining tools, in the end the model must be applied inside the database system [6]. When volumes of data are not large it is feasible to perform model deployment outside: exporting data sets, applying the model and building reports can be done in no more than a few minutes. However, as data volume increases exporting data from the database system becomes a bottleneck. This problem gets compounded with results interpretation when it is necessary to relate statistical numbers back to the original tables in the database. Therefore, it is common to build models outside, frequently based on samples, and then once an acceptable model is obtained, then it is imported back into the database system. Nowadays, model deployment basically happens in two ways: using SQL queries if the mathematical computations are relatively simple or with UDFs [2], [12] if the computations are more sophisticated. In most cases, such scoring process can work in a single table scan, providing good performance.

IV. Comparing Processing of Workflows Inside and Outside a database system

In this section we present a qualitative and quantitative comparison between running big data analytics workflows inside and outside the database system. This discussion is a summary of representative successful projects. We first discuss a typical database environment. Second, we present a summary of the data mining projects presenting their practical application and the statistical and data mining techniques used. Third, we discuss advantages and accomplishments for each workflow running inside the database system, as well as the main objections or concerns against such approach (i.e. migrating external transformation code into the database system). Fourth, we present time measurements taken from actual projects at each organization, running big data analytics workflows completely inside and partially outside the database system.

A. Data Warehousing Environment

The environment was a data warehouse, where several databases were already integrated into a large enterprise-wide database. The database server was surrounded by specialized servers performing OLAP and statistical analysis. One of those servers was a statistical server with a fast network connection to the database server.

First of all, an entire set of statistical language programs were translated into SQL using Teradata data mining program, the translator tool and customized SQL code. Second, in every case the data sets were verified to have the same contents in the statistical language and SQL. In most cases, the numeric output from statistical and machine learning models was the same, but sometimes there were slight numeric differences, given variations in algorithmic improvements and advanced parameters (e.g. epsilon for convergence, step-wise regression procedures, pruning method in decision tree and so on).

B. Organizations: statistical models and business application

We now give a brief discussion about the organizations where the statistical code migration took place. We also discuss the specific type of data mining techniques used in each case. Due to privacy concerns we omit discussion of specific information about each organization, their databases and the hardware configuration of their database system servers. The big data analytics workflows were executed on the organizations database servers, concurrently with other users (analysts, managers, DBAs, and so on).

We now describe the computers processing the workflow in more detail. The database system server was, in general, a parallel multiprocessor computer with a large number of CPUs, ample memory per CPU and several terabytes of parallel disk storage in high performance RAID configurations. On the other hand, the statistical server was generally a smaller computer with less than 500 GB of disk space with ample memory space. Statistical and data mining analysis inside the database system was performed only with SQL. In general, a workstation connected to each server with appropriate client utilities. The connection to the database system was done with ODBC. All time measurements discussed herein were taken on 32-bit CPUs over the course of several years. Therefore, they cannot be compared with each other and they should only be used to understand performance gains within the same organization.

The first organization was an insurance company. The data mining goal involved segmenting customers into tiers according to their profitability based on demographic data, billing information and claims. The statistical techniques used to determine segments involved histograms and clustering. The final data set had about $n = 300$k records and $d = 25$ variables. There were four segments, categorizing customers from best to worst.

The second organization was a cellular telephone service provider. The data mining task involved predicting which customers were likely to upgrade their call service package or purchase a new handset. The default technique was logistic regression [7] with stepwise procedure for variable selection. The data set used for scoring had about $n = 10$M records and $d = 120$ variables. The predicted variable was binary.

The third organization was an Internet Service Provider (ISP). The predictive task was to detect which customers were likely to disconnect service within a time window of a few months, based on their demographic data, billing information and service usage. The statistical techniques used in this case were decision trees and logistic regression and the predicted variable was binary. The final data set had $n = 3.5$M records and $d = 50$ variables.

C. Database system-centric big data analytics workflows: Users Opinion

We summarize pros and cons about running big data analytics workflows inside and outside the database system. Table I contains a summary of outcomes. As we can see performance to score data sets and transforming data sets are positive outcomes in every case. Building the models faster turned out not be as important because users relied on sampling to build models and several samples were collected to tune and test models. Since all databases and servers were within the same firewall security was not a major concern. In general, improving data management was not seen as major concern because there existed a data warehouse, but users acknowledge a "distributed" analytic environment could be a potential management issue. We now summarize the main objections, despite the advantages discussed above. We exclude cost as a decisive factor to preserve anonymity of users opinion and give an unbiased discussion. First, many users preferred a traditional programming language like Java or C++ instead of a set-oriented language like SQL. Second, some specialized techniques are not available in the database system due to their mathematical complexity; relevant examples include Support Vector Machines, Non-linear regression and time series models. Finally, sampling is a standard mechanism to analyze large data sets.

D. Workflow Processing Time Comparison

We compare workflow processing time inside the database system using SQL and UDFs and outside the database system, using an external server transforming exported data sets and computing models. In general, the external server ran existing data transformation programs developed by each organization. On the other hand, each organization had diverse data mining tools that analyzed flat files. We must mention the comparison is not fair because the database system server was in general a powerful parallel computer and the external server was a smaller computer. Nevertheless, such setting represents a common IT environment where the fastest computer is precisely the database system server.

We discuss tables from the database in more detail. There were several input tables coming from a large normalized database that were transformed and denormalized to build data sets used by statistical or machine learning techniques. In short, the input were tables and the output were tables as well. No data sets were exported in this case: all processing happened inside the database system. On the other hand, analysis on the external server relied on SQL queries to extract data from the database system, transform the data to produce data sets in the statistical server and then building models or scoring data sets based on a model. In general, data extraction from the database system was performed using bulk utilities which exported data records in blocks. Clearly, there was an export bottleneck from the database system to the external server.

Table II compares performance between both workflow processing alternatives: inside and outside the database system. As

TABLE I
ADVANTAGES AND DISADVANTAGES ON RUNNING BIG DATA ANALYTICS WORKFLOWS INSIDE A DATABASE SYSTEM.

	Insur.	Phone	ISP
Advantages:			
Improve Workflow Management	X	X	X
Accelerate Workflow Execution	X		
Prepare data sets more easily	X	X	X
Compute models faster without sampling	X		
Score data sets faster	X	X	X
Enforce database security	X	X	
Disadvantages:			
Workflow output independent		X	X
Workflow outside database system OK		X	X
Prefer program. lang. over SQL		X	X
Samples accelerate modeling	X	X	
database system lacks statistical models		X	
Legacy code		X	

TABLE II
WORKFLOW PROCESSING TIME INSIDE AND OUTSIDE A DATABASE SYSTEM (TIME IN MINUTES).

Task	outside	inside
Build model:		
Segmentation	2	1
Predict propensity	38	8
Predict churn	120	20
Score data set:		
Segmentation	5	1
Predict propensity	150	2
Predict churn	10	1

TABLE III
TIME TO COMPUTE MODELS INSIDE THE DATABASE SYSTEM AND TIME TO EXPORT DATA SET (SECS).

$n \times 1000$	d	SQL/UDF	BULK	ODBC
100	8	4	17	168
100	16	5	32	311
100	32	6	63	615
100	64	8	121	1204
1000	8	40	164	1690
1000	16	51	319	3112
1000	32	62	622	6160
1000	64	78	1188	12010

introduced in Section II we distinguish two big data analytics workflows: computing the model and scoring the model on large data sets. The times shown in Table II include the time to transform the data set with joins and aggregations the time to compute the model and the time to score applying the best model. As we can see the database system is significantly faster. We must mention that to build the predictive models both approaches exploited samples from a large data set. Then the models were tuned with further samples. To score data sets the gap is wider, highlighting the efficiency of SQL to compute joins and aggregations to build the data set and then computing mathematical equations on the data set. In general, the main reason the external server was slower was the time to export data from the database system. A secondary reason was its more limited computing power.

Table III compares processing time to compute a model inside the database system and the time to export the data set (with a bulk utility and ODBC). In this case the database system runs on a relatively small computer with 3.2 GHz, 4 GB on memory and 1 TB on disk. The models include PCA, Naive Bayes and linear regression, which can be derived from the correlation matrix [7] of the data set in a single table scan using SQL queries and UDFs. Exporting the data set is a bottleneck to perform data mining processing outside the database system, regardless of how fast the external server is. However, exporting a sample from the data set can be done fast, but analyzing a large data set without sampling is much faster to do inside the database system. Also, sampling can also be exploited inside the database system.

V. RELATED WORK

There exist many proposals that extend SQL with data mining functionality. Most proposals add syntax to SQL and optimize queries using the proposed extensions. Several techniques to execute aggregate UDFs in parallel are studied in [9]; these ideas are currently used by modern parallel relational database systems such as Teradata. SQL extensions to define, query and deploy data mining models are proposed in [11]; such extensions provide a friendly language interface to manage data mining models. This proposal focuses on managing models rather than computing them and therefore such extensions are complementary to UDFs. Query optimization techniques and a simple SQL syntax extension to compute multidimensional histograms are proposed in [8], where a multiple grouping clause is optimized.

Some related work on exploiting SQL for data manipulation tasks includes the following. Data mining primitive operators are proposed in [1], including an operator to pivot a table and another one for sampling, useful to build data sets. The pivot/unpivot operators are extremely useful to transpose and transform data sets for data mining and OLAP tasks [3], but they have not been standardized. Horizontal aggregations were proposed to create tabular data sets [13], as required by statistical and machine learning techniques, combining pivoting and aggregation in one function. For the most part research work on preparing data sets for analytic purposes in a relational database system remains scarce. Data quality is a fundamental aspect in a data mining application. Referential integrity quality metrics are proposed in [14], where users can isolate tables and columns with invalid foreign keys. Such referential problems must be solved in order to build a data set without missing information.

Mining workflow logs, a different problem, has received attention [18]; the basic idea is to discover patterns in workflow process logs. To the best of our knowledge, there is scarce research work dealing with migrating data preprocessing into a database system to improve management and processing of big data analytics workflows.

VI. Conclusions

We presented practical issues and discussed common solutions to push data preprocessing into a database system to improve management and processing of big data analytics workflows. It is important to emphasize data preprocessing is generally the most time consuming and error-prone task in a big data analytics workflow. We identified specific data preprocessing issues. Summarization generally has to be done at different granularity levels and such levels are generally not hierarchical. Rows are selected based on a time window, which requires indexes on date columns. Row selection (filtering) with complex predicates happens on many tables, making code maintenance and query optimization difficult. In general, it is necessary to create a “universe” data set to define left outer joins. Model deployment requires importing models as SQL queries or UDFs to deploy a model on large data sets. Based on experience from real-life projects, we compared advantages and disadvantages when running big data analytics workflows entirely inside and outside a database system. Our general observations are the following. big data analytics workflows are easier to manage inside a database system. A big data analytics workflow is generally faster to run inside a database system assuming a data warehousing environment (i.e. data originates from the database). From a practical perspective workflows are easier to manage inside a database system because users can exploit the extensive capabilities of the database system (querying, recovery, security and concurrency control) and there is less data redundancy. However, external statistical tools may provide more flexibility than SQL and more statistical techniques. From an efficiency (processing time) perspective, transforming and scoring data sets and computing models are faster to develop and run inside the database system. Nevertheless, sampling represent a practical solution to accelerate big data analytics workflows running outside a database system.

Improving and optimizing the management and processing of big data analytics workflows provides many opportunities for future work. It is necessary to specify models which take into account processing with external data mining tools. The data set tends to be the bottleneck in a big data analytics workflow both from a programming and processing time perspectives. Therefore, it is necessary to specify workflow models which can serve as templates to prepare data sets. The role of the statistical model in a workflow needs to studied in the context of the source tables that were used to build the data set; very few organizations reach that stage.

References

[1] J. Clear, D. Dunn, B. Harvey, M.L. Heytens, and P. Lohman. Non-stop SQL/MX primitives for knowledge discovery. In *ACM KDD Conference*, pages 425–429, 1999.

[2] J. Cohen, B. Dolan, M. Dunlap, J. Hellerstein, and C. Welton. MAD skills: New analysis practices for big data. In *Proc. VLDB Conference*, pages 1481–1492, 2009.

[3] C. Cunningham, G. Graefe, and C.A. Galindo-Legaria. PIVOT and UNPIVOT: Optimization and execution strategies in an RDBMS. In *Proc. VLDB Conference*, pages 998–1009, 2004.

[4] R. Elmasri and S.B. Navathe. *Fundamentals of Database Systems*. Addison-Wesley, 4th edition, 2003.

[5] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, November 1996.

[6] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2006.

[7] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning*. Springer, New York, 1st edition, 2001.

[8] A. Hinneburg, D. Habich, and W. Lehner. Combi-operator-database support for data mining applications. In *Proc. VLDB Conference*, pages 429–439, 2003.

[9] M. Jaedicke and B. Mitschang. On parallel processing of aggregate and scalar functions in object-relational DBMS. In *ACM SIGMOD Conference*, pages 379–389, 1998.

[10] T.M. Mitchell. *Machine Learning*. Mac-Graw Hill, New York, 1997.

[11] A. Netz, S. Chaudhuri, U. Fayyad, and J. Berhardt. Integrating data mining with SQL databases: OLE DB for data mining. In *Proc. IEEE ICDE Conference*, pages 379–387, 2001.

[12] C. Ordonez. Statistical model computation with UDFs. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 22(12):1752–1765, 2010.

[13] C. Ordonez and Z. Chen. Horizontal aggregations in SQL to prepare data sets for data mining analysis. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 24(4):678–691, 2012.

[14] C. Ordonez and J. García-García. Referential integrity quality metrics. *Decision Support Systems Journal*, 44(2):495–508, 2008.

[15] C. Ordonez and J. García-García. Database systems research on data mining. In *Proc. ACM SIGMOD Conference*, pages 1253–1254, 2010.

[16] D. Pyle. *Data preparation for data mining*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[17] E. Rahm and D. Hong-Hai. Data cleaning: Problems and current approaches. *IEEE Bulletin of the Technical Committee on Data Engineering*, 23(4), 2000.

[18] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters. Workflow mining: a survey of issues and approaches. *Data & Knowledge Engineering*, 47(2):237–267, 2003.