Warehousing Workflow Data: Challenges and Opportunities

Angela Bonifati

Politecnico di Milano
Via Ponzio 34/5
20133 Milano
Italy
bonifati@elet.polimi.it

Fabio Casati, Umesh Dayal, Ming-Chien Shan

HP Labs
1501 Page Mill Road
Palo Alto, CA, 94304
USA
[casati,dayal,shan]@hpl.hp.com

Abstract

Workflow management systems (WfMSs) are software platforms that allow the definition, execution, monitoring, and management of business processes. WfMSs log every event that occurs during process execution. Therefore, workflow logs include a significant amount of information that can be used to analyze process executions, understand the causes of high- and low-quality process executions, and rate the performance of internal resources and business partners. In this paper we present a packaged data warehousing solution, coupled with HP Process Manager, for collecting and analyzing workflow execution data. We first present the main challenges involved in this effort, and then detail the proposed approach.

1. Introduction and motivations

Workflow Management Systems (WfMSs) are being increasingly used by many companies to improve the efficiencies of their processes and reduce costs. WfMSs log many events that occur during process executions, including the start and completion time of each activity, its input and output data, and the resource that executed it. The analysis of such precious information could allow business and IT manager to detect problems and inefficiencies, and to identify solutions. However, most WfMSs only offer basic log analysis functionality, such as the ability of retrieving the number of process instances completed in a given time period and their average execution time. To get more comprehensive reports, users have to configure commercial reporting tools and write queries on the logs to retrieve data of interest.

While this approach does provide basic reporting functionality, it requires a considerable configuration effort (it is very difficult to write the "right" queries and to extract the desired information). In addition, WfMSs logs were not designed for OLAP applications, may contain incorrect information that must be checked and cleaned[1], and do not provide support for aggregating data from multiple data sources.

In order to overcome the above limitations, we designed and implemented a warehouse of workflow execution data (called Workflow Data Warehouse, or WDW in the following). The goal of our work was to develop a packaged solution, optimized for HP Process Manager (HPPM) but generally applicable to any WfMS. Hence, the WDW must be easy to install and use, and must perform adequately under different conditions, (e.g., different log sizes, or different data loading and aggregation requirements). In addition, the problem of warehousing workflow data present several challenges:

- *Multiple related fact types.* Workflow executions may generate different kinds of *facts* about workflow activities, resources, and instances. These facts are related among each other. For example, activities are executed in the context of a specific workflow instance. The presence of multiple, related types of facts affects both the design of the warehouse schema and the data loading process, due to the need of ensuring semantic correctness, avoiding information loss, and guaranteeing an acceptable performance.
- *Conceptually complex aggregations:* the definition of summary tables is a complex problem in itself. As an example, the problem of generating aggregate data that allow rating workflow resources has been the subject of an entire research internship at HP.

Proceedings of the 27th VLDB Conference,
Roma, Italy, 2001

[1] For example, some WfMS may write special codes in the activity completion time (e.g., Jan 1, 1970) to denote failures. The presence of a single occurrence of this value, if not appropriately handled, would invalidate any aggregate data concerning the activity execution time.

– *Diversity and evolution management*: Workflow models are in continuous evolution, and commercial products often add new features to the model they support. While it is possible to focus the design on current models, the drawback is that even simple modifications to the workflow models may require a major warehouse redesign effort. In addition, the WDW should ideally be able to host data from any WfMS, in addition to HP Process Manager

In this paper we present the design of the WDW and detail how we addressed these challenges. Due to space limitations, we omit a discussion on workflow models and workflow execution logs. The interested reader is referred to [Leymann00] for an introduction to workflows, to [HPPM-PD] for the specific of the HPPM process model, and to [HPPM-TR] for the HPPM logs.

The development of a workflow data warehouse is part of a larger effort, aiming at developing a *Business Process Intelligence* (BPI) solution. The goals of BPI are to enable business and IT users to extract knowledge hidden in the WfMS logs and to be alerted of critical situations or foreseen quality of service degradations. An additional, long-term goal is the ability of dynamically optimizing workflow definitions and executions. Details on the BPI effort are provided in [Casati01].

2. Warehouse Design

Following "traditional" data warehousing techniques, we structure the WDW design according to a (relational) star schema, where data are described in terms of "*facts*" to be analyzed and "*dimensions*", i.e., perspectives under which the facts are analyzed. A design based on a star schema enables multidimensional analysis and allows the use of many query optimization techniques.

We have worked with customers and consultants to identify the structure and relationships of facts and dimensions in order to optimize performance for many typical analysis needs. The WDW design is summarized in Figure 1. Workflow dimensions are shown in the Figure and not detailed further due to space limitations. They are relatively stable across workflow models and do not present particular issues. The only exception is represented by the *behavior* dimension, which is very useful for analysis and will be detailed later in the paper. Instead, we next examine workflow facts.

2.1 Process and Node facts

In a business process, happenings of interest are changes in the process and node execution states. Hence, we consider state changes as the facts of the warehouse. WDW only includes facts about completed process instances, to simplify data archival and loading and to provide a simple framework in which to analyze data.

The definition of the structure and relationship among facts is complicated by the variety of node types present in most workflow models. For instance, the HPPM process model includes a *work* and a *route* node, to model service invocation and routing decisions, respectively. These nodes have different attributes that need to be described in the WDW schema. For example, a work node execution may be related to the service invoked or to the resource that executed the service, while a route node execution may be characterized by the set of arcs fired. In addition, different workflow models (or different versions of the same model) may have different types of nodes, as well as different attributes for a process. Hence, we are faced with the problem of designing fact tables so that all process and node facts can be represented, while still enabling easy maintenance and satisfactory performance.

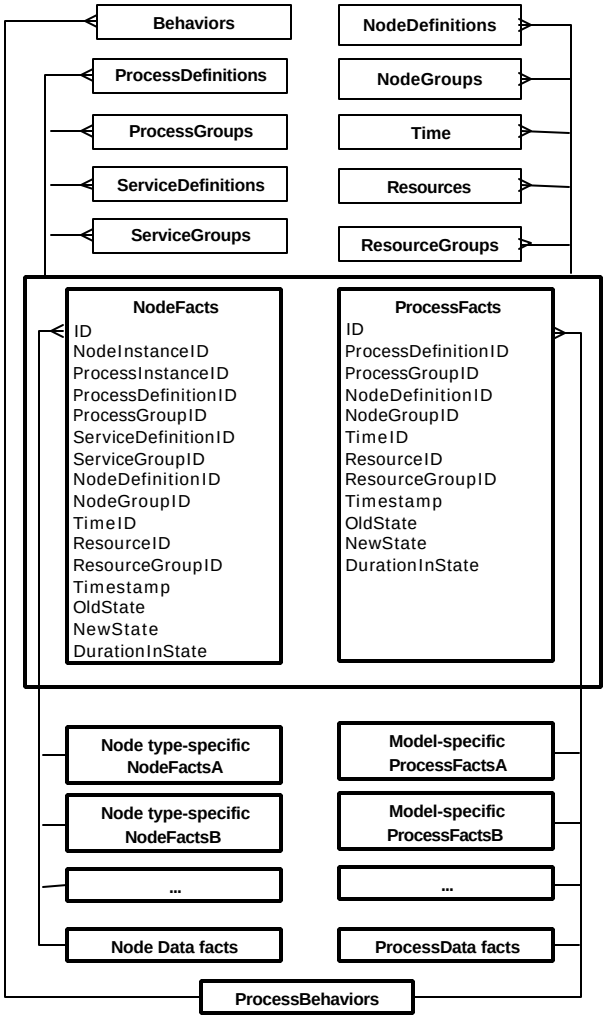

Figure 1 - WDW schema. Facts are depicted with a thicker border, while dimensions have a thin border

In order to address this issue, we designed *generic* (node type- and model-independent) fact tables that include not only attributes common to most models or node types, but also attributes often needed by typical queries (although they may have null values for some

workflow models or node types). In this way, most data aggregation and reports can be computed based on this table, thereby simplifying view definitions and avoiding the need of joining several tables to compute the results (see Figure 1). Facts tables that are specific to a node type or to a process model can also be added to the warehouse, to allow storing attributes that are not included in the generic tables but that can be occasionally required for specific reporting needs.

An additional problem to be handled is modeling data modifications. In fact, changes to process instance data could be of interest to the analyst. For instance, analysts may want to observe facts related to purchases of cars above 20,000$. Data modifications are not included in the process and node fact tables. In fact, different processes and nodes modify different data. Hence, it is unfeasible to store them horizontally in the relation. Instead, data modifications are stored in the *ProcessDataFacts* and *NodeDataFacts* relations, that include tuples detailing the process or node state change in correspondence of which the data modification occurred (e.g., a node completion), the data item name, and the old and new value.

Note that in general there is no need to log *all* data items. The analyst may only be interested in a fraction of them, and complete data logging could be a very heavy burden for the warehouse. Hence, WDW administrators can specify which data modifications are to be loaded, based on the data item name, and on the process and work node in the context of which the modifications take place.

2.2 Behaviors

The availability of process and node information in the WDW enables many data analysis functionalities. However, more interesting information can be extracted by including in the WDW data at a higher level of abstraction. Indeed, very frequently, analysts are not only interested in analyzing node activations and data modifications. Instead, they want to analyze process instances that exhibit specific *behaviors*, and to understand the *causes* of such behaviors. Examples of behaviors of interest are supply chain process instances that last more than 20 days, *Expense Approval* process instances that include more than 3 approval cycles, *Claim Management* instances in which node "Examine expenses" was executed by a manager, or processes instances related to order for goods over 20,000$.

The WDW approach is agnostic about the behaviors that users may be interested in analyzing. Indeed, we allow users to define the behaviors to be monitored. The system will then take care of identifying which processes exhibit a specific behavior and of analyzing them.

Behavior types are (conceptually) defined by Boolean conditions over process and node execution data available in the warehouse. Behavior conditions are parametric, and can be configured to monitor a specific *behavior* for a specific process definition. Completed process instances for which the condition is true are considered as being affected by the behavior. For instance, a *ProcessDuration* behavior type may be defined as "instance of a process definition *PD* that lasts more than *D* days". In this case, *PD* and *D* are parameters that can be defined to detail which processes should be monitored and the exact behavior to be detected. Multiple specific behaviors to be monitored (on different processes) can be defined for each behavior type, and a process can be analyzed for multiple behaviors. In the prototype implementation, conditions are defined by SQL scripts that label process instances affected by the exceptional behavior under consideration.

Processes to be monitored and behavior configuration parameters are stored in a behavior type-independent table. The table includes many attributes, to allow the configuration of a variety of behavior types. If a parameter is meaningless for a behavior type, then it is left unspecified. Labeling information are detected by executing the scripts on warehouse data and by storing the results in a *ProcessBeahvior* table that couples process instance identifiers with behavior identifiers.

WDW includes a wide set of built-in behavior types, such as processes lasting less (more) than a specified duration, being in the slowest (fastest) *x*%, including more (less) than *n* activations of a specific work node *WN*, or in which work node WN has (not) been executed by a resource in group *G*. Users can define new behavior types by simply providing the corresponding script.

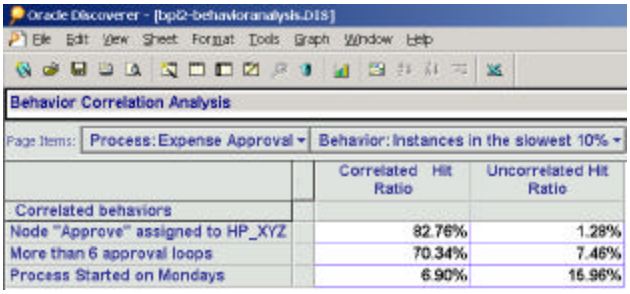

Figure 2 – Analysis of correlations among behaviors.

By detecting behaviors of interest, analysts can perform multidimensional analysis to understand the causes of "good" and "bad" process executions, and take actions accordingly. In particular, a very useful analysis consists in examining *correlations* among behaviors. The WDW includes a set of views that supports users in this analysis. As an example, Figure 2 shows a report build with Oracle Discoverer through a simple projection query on a WDW view. The report shows the analysis for a behavior B of *ProcessDuration* type, configured to detect instances in the slowest 10% for the *Expense Approval* process. The report shows which other behaviors were detected in instances affected by behavior B, and what are the correlated[2] and uncorrelated behavior hit rates.

[2] I.e., the percentage of instances affected by a behavior b_1 among those also affected by behavior b_2.

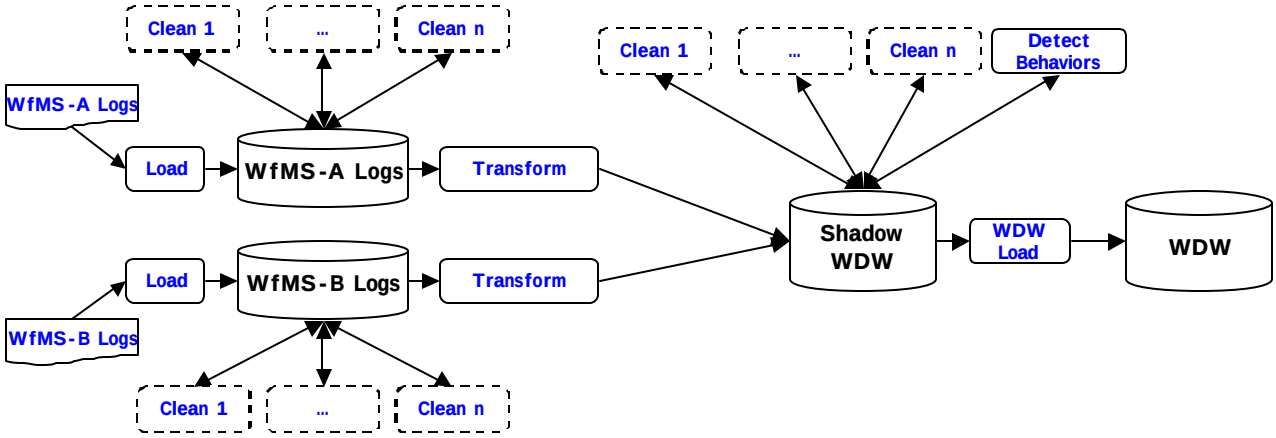

Figure 3 – WDW load process. Optional components are depicted with a dashed border

This kind of analysis is very useful in showing cause-effect relationships among behaviors, and therefore in identifying the causes of behaviors. An extension of this approach is presented in [Casati01], where business intelligence techniques are used to provide a more automated discovery of the causes of behaviors, although at the price of increased complexity and cost, due to the need for data mining tools and techniques.

2.3 Views

One of our design requirements was to make it very easy for users to obtain reports from the WDW. For this reason, the WDW includes a large set of views (providing partially overlapping and redundant information), to account for the large majority of reporting needs. Analysts only need to write simple projection and selection queries to get the report they need. In addition, we provide configuration files for the most common reporting tools, such as Oracle Discoverer or MS Excel, so that using the WDW requires minimum configuration effort.

In order to provide acceptable performance while at the same time avoid explosion in disk space usage, reporting views are not materialized. Instead, a smaller set of materialized views is defined, joining and aggregating data in different ways to support the needs of several reporting views. Through the query rewrite mechanisms, Oracle automatically uses materialized views (where possible) to speed up queries on the reporting views, thereby reducing the execution time.

3. Loading the Workflow Data Warehouse

We now describe the process of extracting data from workflow logs and loading the WDW, shown in Figure 3. This process is performed by a set of ETL scripts provided with the WDW. We assume that log data are available in the form of log files, extracted from WfMS logs (typically stored in relational databases). The first step consists in extracting data from the files and restoring the content in relational format. Then, a sequence of cleaning operations can be applied. WDW provides a set of data checking/cleaning modules that process data *without* changing the structure. The advantage of this approach is that cleaning modules can be plugged in and out depending on the cleaning needs of the users. The addition of each cleaning module causes delays in the load process, but can guarantee data consistency, absence of duplicates, and other properties that are crucial for WDW integrity.

Data are then inserted into WDW *shadow* database, i.e., a database that has the same schema of WDW. Preparing shadow table instead of directly loading the warehouse has several motivations:

- Once data are in the shadow table, then the WDW can be quickly loaded by means of simple inserts or partition exchanges, reducing the WDW downtime.
- Shadow tables have a WfMS-independent schema, and can be used to execute, WfMS-independent cleaning operations
- Dimensions as well as relationships between facts and dimensions can be computed from the shadow tables. For example, shadow tables can be used to collect timestamps of facts, and load the *Time* dimension table, to extract load statistics and to perform higher-level operations, such as detecting *behaviors*.

Once cleaning and transformation operations have been completed, data are loaded into the WDW.

4. References

[Casati01] F. Casati, U. Dayal, D. Grigori, M.C. Shan. Improving Business Process Quality through Exception Understanding, Prediction, and Prevention. *Procs. of VLDB'01*, Rome, Italy. Sept. 2001

[HPPM-PD] Hewlett-Packard. HP Changengine Process Design Guide. Edition 4.4. 2000

[HPPM-TR] Hewlett-Packard. HP Changengine Technical Reference Guide. Edition 4.4. 2000

[Leymann00] F. Leymann, D. Roller: *Production Workflow*. Prentice-Hall, 2000.