Modeling Data Warehouse Refreshment Process as a Workflow Application

Mokrane Bouzeghoub[(*)(**)], Françoise Fabret[(*)], Maja Matulovic-Broqué[(*)]

[(*)]INRIA Rocquencourt, France
[(**)] Laboratoire PRiSM, Université de Versailles, France
Mokrane.Bouzeghoub@prism.uvsq.fr

Abstract

This article is a position paper on the nature of the data warehouse refreshment which is often defined as a view maintenance problem or as a loading process. We will show that the refreshment process is more complex than the view maintenance problem, and different from the loading process. We conceptually define the refreshment process as a workflow whose activities depend on the available products for data extraction, cleaning and integration, and whose coordination events depend on the application domain and on the required quality in terms of data freshness. Implementation of this process is clearly distinguished from its conceptual modelling.

1. Introduction

Data warehousing is a new technology which provides software infrastructure for decision support systems and OLAP applications. Data warehouses collect data from heterogeneous and distributed sources. This data is aggregated and then customized with respect to organizational criteria defined by OLAP applications. The data warehouse can be defined as a hierarchy of data stores which goes from source data to the highly aggregated data (data marts). Between these two extreme data stores, we can find different other stores depending on the requirements of OLAP applications. One of these stores is the operational data store which reflects source data in a uniform and clean representation. The corporate data warehouse (CDW) contains highly aggregated data and can be organized into a multidimensional structure. Data extracted from each source can also be stored in intermediate data recipients. Obviously, this hierarchy of data stores is a logical way to represent the data flows which go from the sources to the data marts. All these stores are not necessarily materialized, and if they are, they can just constitute different layers of the same database.

Figure 1 shows a typical data warehouse architecture. This is a logical view whose operational implementation receives many different answers in the data warehousing products. Depending on each data source, extraction and cleaning can be done by the same wrapper or by distinct tools. Similarly data reconciliation (also called multi-source cleaning) can be separated from or merged with data integration (multi-sources operations). High level aggregation can be seen as a set of computation techniques ranging from simple statistical functions to advanced data mining algorithms. Customisation techniques may vary from one data mart to another, depending on the way decision makers want to see the elaborated data.

Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'99)
Heidelberg, Germany, 14. - 15. 6. 1999
(S. Gatziu, M. Jeusfeld, M. Staudt, Y. Vassiliou, eds.)
http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-19/

The research presented in this paper is supported by the European Commission under the Esprit Prgram LTR project 'DWQ: Foundations of Data Warehouse Quality'

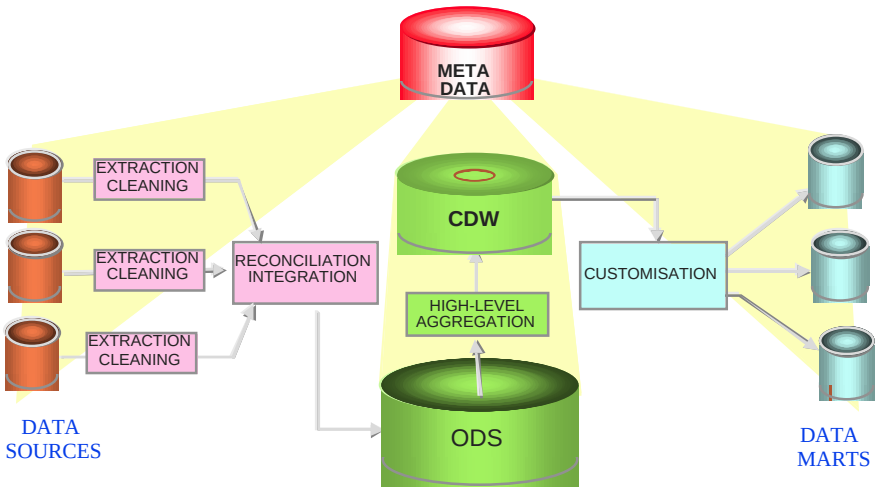

Figure 1: Data warehouse architecture

The refreshment of a data warehouse is an important process which determines the effective usability of the data collected and aggregated from the sources. Indeed, the quality of data provided to the decision makers depends on the capability of the data warehouse system to convey in a reasonable time, from the sources to the data marts, the changes made at the data sources. Most of the design decisions are then concerned by the choice of data structures and update techniques that optimise the refreshment of the data warehouse.

There is a quiet great confusion in the literature concerning data warehouse refreshment. Indeed, this process is often either reduced to view maintenance problem or confused with the data loading phase. Our purpose in this paper is to show that the data warehouse refreshment is a more complex than the view maintenance problem, and different from the loading process. We define the refreshment process as a workflow whose activities depend on the available products for data extraction, cleaning and integration, and whose triggering events of these activities depend on the application domain and on the required quality in terms of data freshness.

The objective of the following sections is to describe the refreshment process tasks and to demonstrate how they can be organised as a workflow. Section 2 arguments on differences between the refreshment process in one side and the data loading and view maintenance in the other side. Section 3 defines the generic workflow which logically represents the refreshment process, with examples of workflow scenarios. Section 4 defines the semantics of the refreshment process in terms of workflow design decisions. Section 5 concludes with the summary of the main ideas in this position paper and on some implementation issues.

2. View maintenance, data loading and data refreshment

Data refreshment in data warehouses is generally confused with data loading as done during the initial phase or with update propagation through a set of materialized views. Both analogies are wrong. The following paragraphs argument on the differences between data loading and data refreshment, and between view maintenance and data refreshment.

Data loading vs. data refreshment

The data warehouse *loading phase* consists in the initial data warehouse instantiation, that is the initial computation of the data warehouse content. This initial loading is globally a sequential process of four steps (Figure 2): (i) preparation, (ii) integration, (iii) high level aggregation and (iv) customisation. The

first step is done for each source and consists in data extraction, data cleaning and possibly data archiving before or after cleaning. Archiving data in a history can be used both for synchronisation purpose between sources having different access frequencies and for some specific temporal queries. The second step consists in data reconciliation and integration, that is cleaning multi-source cleaning of data originated from heterogeneous sources, and derivation of the base relations (or base views) of the operational data store (ODS). The third step consists in the computation of aggregated views from base views. While the data extracted from the sources and integrated in the ODS is considered as ground data with very low level aggregation, the data in the corporate data warehouse (CDW) is generally highly summarised using aggregation functions. The fourth step consists in the derivation and customisation of the user views which define the data marts. Customisation refers to various presentations needed by the users for multidimensional data.

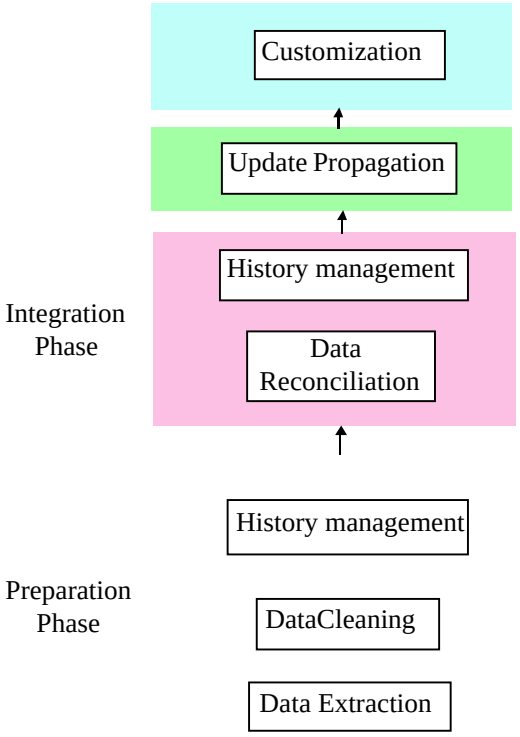

Figure 2: Data loading activities

The main feature of the loading phase is that it constitutes the latest stage of the data warehouse design project. Before the end of the data loading, the data warehouse does not yet exist for the users. Consequently, there is no constraint on the response time. But, in contrast, with respect to the data sources, the loading phase requires more availability.

The data flow which describes the loading phase can serve as a basis to define the refreshment process, but the corresponding workflows are different. The workflow of the refreshment process is dynamic and can evolve with users' needs and with source evolution, while the workflow of the initial loading process is static and defined with respect to current user requirements and current sources.

The difference between the refreshment process and the loading process is mainly in the following. First, the refreshment process may have a complete asynchronism between its different activities (preparation, integration, aggregation and customisation). Second, there may be a high level parallelism within the preparation activity itself, each data source having its own availability window and its own strategy of extraction. The synchronization is done by the integration activity. Another difference lies in the source availability. While the loading phase requires a long period of availability, the refreshment phase should not overload the operational applications which use the data sources. Then, each source provides a specific access frequency and a restricted availability duration. Finally, there are more constraints on response time for the refreshment process than for the loading process. Indeed, with respect to the users, the data warehouse does not exist before the initial loading, so the computation time is included within the design project duration. After the initial loading, the data becomes visible and should satisfy user requirements in terms of data availability, accessibility and freshness.

View maintenance vs. data refreshment

The propagation of changes during the refreshment process is done through a set of independent activities among which we find the maintenance of the views stored in the ODS and CDW levels. The view maintenance phase consists in propagating a certain change raised in a given source over a set of views stored at the ODS or CDW level. Such a phase is a classical materialized view maintenance problem except that, in data warehouses, the changes to propagate into the aggregated views are not exactly those occurred in the sources, but the result of pre-treatments

performed by other refreshment activities such as data cleaning and multi-source data reconciliation.

The view maintenance problem has been intensively studied in the database research community. Major work done in this area is synthesized in [BFG+99] and [ThBo 99]. Most of the references focus on the problems raised by the maintenance of a set of materialized (also called concrete) views derived from a set of base relations when the current state of the base relations is modified. The main results concern :

- *The self-maintainability* : Results concerning the self-maintainability are generalized for a set of views : a set of view V is self-maintainable with respect to the changes to the underlying base relations if the changes may be propagated in every views in V without querying the base relations (i.e. the information stored in the concrete views plus the instance of the changes are sufficient to maintain the views).
- *The coherent and efficient update propagation*: Various algorithms are provided to schedule updates propagation through each individual view, taking care of interdependencies between views, which may lead to possible inconsistencies. For this purpose, auxiliary views are often introduced to facilitate update propagation and to enforce self-maintainability.

Results over the self-maintainability of a set of views are of a great interest in the data warehouse context, and it is commonly admitted that the set of views stored in a data warehouse have to be globally self-maintainable. The rationale behind this recommendation is that the self-maintainability is a strong requirement imposed by the operational sources in order to not overload their regular activity.

As stated in the previous section, research on data warehouse refreshment has mainly focused on update propagation through materialized views. Many papers have been published on this topic, but a very few is devoted to the whole refreshment process as defined before. We consider view maintenance just as one step of the complete refreshment process. Other steps concern data cleaning, data reconciliation, data customisation, and if needed data archiving. In another hand, extraction and cleaning strategies may vary from one source to another, as well as update propagation which may vary from one user view to another, depending for example on the desired freshness for data. So the data warehouse refreshment process cannot be limited to a view maintenance process.

To summarize the previous discussion, we can say that a refreshment process is a complex system which may be composed of asynchronous and parallel activities that need a certain monitoring. The refreshment process is an event-driven system which evolves frequently, following the evolution of data sources and user requirements. Users, data warehouse administrators and data source administrators may impose specific constraints as, respectively, freshness of data, space limitation of the ODS or CDW, and access frequency to sources. There is no simple and unique refreshment strategy which is suitable for all data warehouse applications, for all data warehouse user, or for the whole data warehouse lifetime.

3. The Refreshment process is a workflow

A workflow is a set of coordinated activities which might be manual or automated activities performed by actors [Scha 98]. Workflow concepts have been used in various application domains such as business process modeling [HaCh 93], cooperative applications modeling [CSCW 96] [Lawr 97], and database transaction modeling [AAE+ 95] [Bern 98]. Depending on the application domain, activities and coordination are defined using appropriate specification languages such as state-chart diagrams and Petri nets [WoKr 93], or active rules [CCPP 95]. In spite of this diversity of applications and representation, most of the workflow users refer more or less to the concepts and terminology defined by the Workflow Coalition [WFMC 95]. Workflow systems are supposed to provide high level flexibility to recursively decompose and merge activities, and allow dynamic reorganization of the workflow process. These features are typically useful in the context of data warehouse refreshment as the activities are performed by market products whose functionalities and scope differ from one product to another.

In the following subsections, we show how the refreshment process can be defined as a workflow application. We illustrate the interest of this approach buy the ability to define different scenarios depending on user requirements, source constraints and data warehouse constraints. We show that these scenarios may evolve through the time to fulfill

evolution of any of the previous requirements and constraints.

3.1. The workflow of the refreshment process

The refreshment process aims to propagate changes raised in the data sources to the data warehouse stores. This propagation is done through a set of independent activities (extraction, cleaning, integration, ...) that can be organized in different ways, depending on the semantics one wants to assign to the refreshment process and on the quality he wants to achieve. The ordering of these activities and the context in which they are executed define this semantics and influence this quality. Ordering and context result from the analysis of view definitions, data source constraints and user requirement in terms of quality factors. In the following subsections, we will describe the refreshment activities and their organization as a workflow. Then we give examples of different workflow scenarios to show how refreshment may be a dynamic and evolving process. Finally, we summarize the different perspectives through which a given refreshment scenario should be considered.

Refreshment activities

The refreshment process is similar to the loading process in its data flow but, while the loading process is a massive feeding of the data warehouse, the refreshment process captures the differential changes hold in the sources and propagates them through the hierarchy of data stores in the data warehouse. The preparation step extracts from each source the data that characterises the changes that have occurred in this source since the last extraction. As for loading, this data is cleaned and possibly archived before its integration. The integration step reconciliates the source changes coming from multiple sources and adds them to the ODS. The aggregation step recomputes incrementally the hierarchy of aggregated views using these changes. The customisation step propagates the summarized data to the data marts. As well as for the loading phase, this is a logical decomposition whose operational implementation receives many different answers in the data warehouse products. This logical view allows a certain traceability of the refreshment process. Figure 3 shows the activities of the refreshment process as well as a sample of the coordinating events.

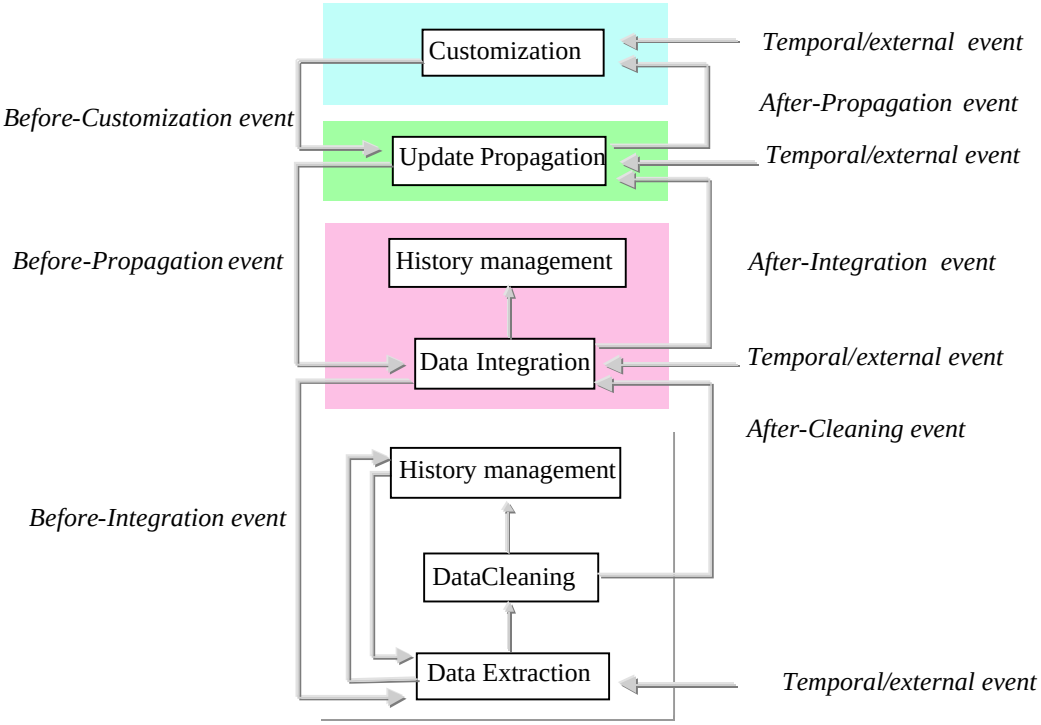

Figure 3: The generic workflow for the refreshment process

Coordination of activities

In workflow systems, activities are coordinated by control flows which may be notification of process commitment, emails issued by agents, temporal events, or any other trigger events. In the refreshment process, coordination is done through a wide range of event types.

We can distinguish several event types which may trigger and synchronize the refreshment activities. They might be temporal events, termination events (dashed lines in figure 3) or any other user-defined event. Depending on the refreshment scenario, one can choose an appropriate set of event types which allows to achieve the correct level of synchronization.

Activities of the refreshment workflow are not executed as soon as they are triggered, they may depend on the current state of the input data stores. For example, if the extraction is triggered periodically, it is actually executed only when there are effective changes in the source log file. If the cleaning process is triggered immediately after the extraction process, it is actually executed only if the extraction process has gathered some source changes. Consequently, we can consider that the state of the input data store of each activity may be considered as a condition to effectively execute this activity.

Within the workflow which represents the refreshment process, activities may be of different origins and different semantics, the refreshment strategy is logically considered as independent of what the activities actually do. However, at the operational level, some activities can be merged (e.g., extraction and cleaning), and some others decomposed (e.g. integration). The flexibility claimed for workflow systems should allow to dynamically tailor the refreshment activities and the coordinating events.

There may be another way to represent the workflow and its triggering strategies. Indeed, instead of considering external events such as temporal events or termination events of the different activities, we can consider data changes as events. Hence, each input data store of the refreshment workflow is considered as an event queue that triggers the corresponding activity. However, to be able to represent different refreshment strategies, this approach needs a parametric synchronization mechanism which allows to trigger the activities at the right moment. This can be done by introducing composite events which combine, for example, data change events and temporal events. Another alternative is to put locks on data stores and remove them after an activity or a set of activities decide to commit. In the case of a long term synchronization policy, as it may sometimes happen in some data warehouses, this latter approach is not sufficient.

The workflow agents

Two main agent types are involved in the refreshment workflow: human agents which define requirements, constraints and strategies, and computer agents which process activities. Among human agents we can distinguish users, the data warehouse administrator, source administrators. Among computer agents, we can mention source management systems, database systems used for the data warehouse and data marts, wrappers and mediators. For simplicity, agents are not represented in the refreshment workflow which concentrates on the activities and their coordination.

3.2. Defining refreshment scenarios

To illustrate different workflow scenarios, we consider the following example which concern three national Telecom billing sources represented by three relations S1, S2, and S3. Each relation has the same (simplified) schema: (#PC, date, duration, cost). An aggregated view V with schema (avg-duration, avg-cost, country) is defined in a data warehouse from these sources as the average duration and cost of a phone call in each of the three country associated with the sources, during the last 6 months. We assume that the construction of the view follows the steps as explained before. During the preparation step, the data of the last six months contained in each source is cleaned (e.g., all cost units are translated in Euros). Then, during the integration phase, a base relation R with schema (date, duration, cost country) is constructed by unioning the data coming from each source and generating an extra attribute (country). Finally, the view is computed using aggregates (figure 4).

We can define another refreshment scenario with the same sources and similar views. This scenario mirrors the average duration and cost for each day instead of for the last six months. This leads to change the frequency of extraction, cleaning, integration and propagation. Figure 5 gives such a

possible scenario. Frequencies of source extractions are those which are allowed by source administrators. Source 3 is permanently available.

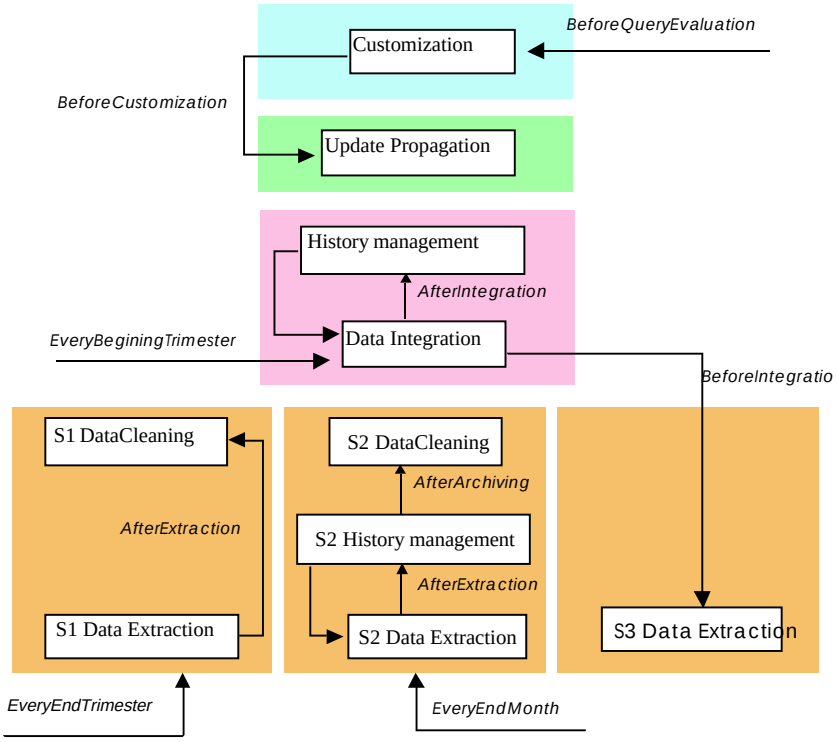

Figure 4: First example of refreshment scenario

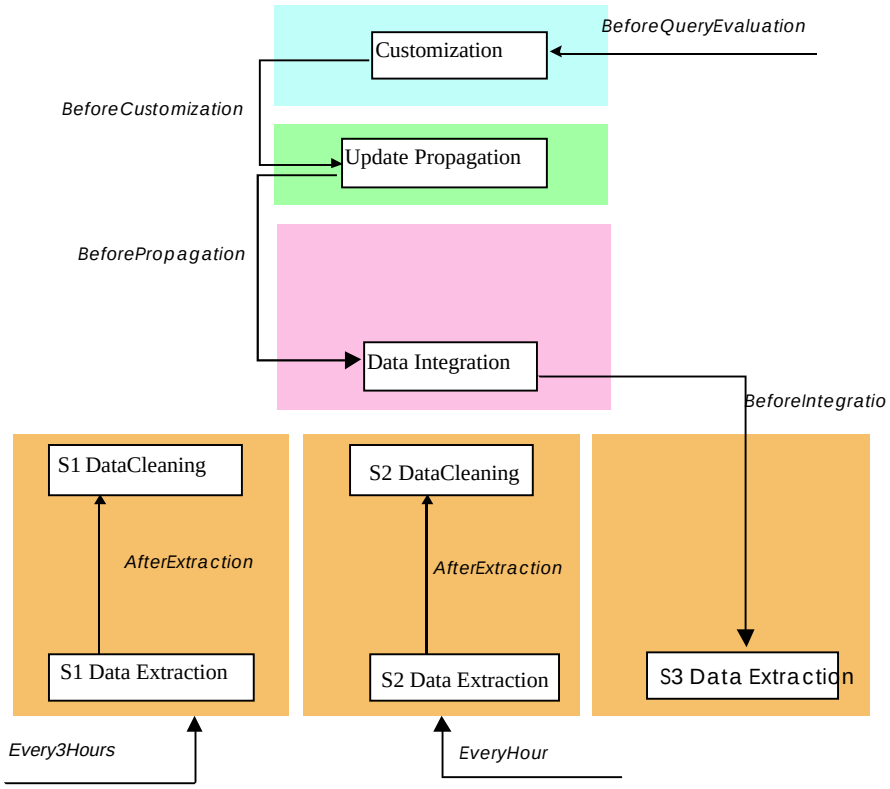

Figure 5: Second example of refreshment scenario

When the refreshment activities are long term activities or when the DWA wants to apply validation procedures between activities, temporal events or activity terminations can be used to synchronize all the refreshment process. In general, the quality requirements may impose a certain synchronization strategy. For example, if users desire high freshness for data, this means that each update in a source should be mirrored as soon as possible to the views. Consequently, this determines the strategy of synchronization: trigger the extraction after each change in a source, trigger the integration, when semantically relevant, after the commit of each data source, propagate changes through views immediately after integration, and customize the user views in data marts.

Refreshment scheduling

The refreshment process can be viewed through different perspectives :

- *Client-driven refreshment* which describes part of the process which is triggered on demand by the users. This part mainly concern update propagation from the ODS to the aggregated views. The on-demand strategy can be defined for all aggregated views or only for those for which the freshness of data is related to the date of querying.

- *Source-driven refreshment* which defines part of the process which is triggered by changes made in the sources. This part concerns the preparation phase. The independence between sources can be used as a way to define different preparation strategies, depending on the sources. Some sources may be associated with cleaning procedures, others not. Some sources need a history of the extracted data, others not. For some sources, the cleaning is done on the fly during the extraction, for some others after the extraction or on the history of these changes. The triggering of the extraction may be also different from one source to another. Different events can be defined, such as temporal events (periodic or fixed absolute time), after each change detected on the source, on demand from the integration process.

- *ODS-driven refreshment* which defines part of the process which is automatically monitored by the data warehouse system. This part concerns the integration phase. It may be triggered at a synchronization point defined with respect to the ending of the preparation phase. Integration can be considered as a whole and concerns all the source changes at the same time. In this case, it can be triggered by an external event which might be a temporal event or the ending of the preparation phase of the last source. The integration can also be sequenced with respect to the termination of the preparation phase of each source, that is extraction is integrated as soon as its cleaning is finished. The ODS can also monitor the preparation phase and the aggregation phase by generation the relevant events that triggers activities of these phases.

In the very simple case, one of the two first approaches is used as a single strategy. In a more complex case, there may be as much strategies as the number of sources or high level aggregated views. In between, there may be, for example, four different strategies corresponding to the previous four phases. For some given user views, one can apply the client driven strategy (pull strategy), while for other views one can apply the ODS-driven strategy (push strategy). Similarly, some sources are solicited through a pull strategy while other apply a push strategy.

The strategy to choose depends on the semantic parameters but also on the tools available to perform the refreshment activities (extraction, cleaning, integration). Some extraction tools do also the cleaning in the fly while some integrators propagate immediately changes until the high level views. Then, the generic workflow in Figure 3 is a logical view of the refreshment process. It shows the main identified activities and the potential event types which can trigger them.

4. Semantics of the refreshment process

As we have seen in the previous examples of scenarios, the view definition is not sufficient to fix the semantics of the refreshment process. Indeed, the query which defines a view does not specify whether this view operates on a history or not, how this history is sampled, whether the changes of a given source should be integrated each hour or each week, and which data timestamp should be taken when integrating changes of different sources. The view definition does not include specific filters defined in the cleaning process, such as choosing the same measure for certain attributes, rounding the values of some attributes, or eliminating some confidential data. Consequently, based on the same view definitions, the refreshment process may produce different results depending on all these extra-parameters which have to be fixed independently, outside the queries which define the views.

The result of a query against view V occurring at time t depends on two main parameters associated with the refreshment strategy implemented by the data warehouse. First, it depends on the change extraction capabilities of each source. For instance, changes in source S1 can be extracted as soon as they occurred, while changes in source S2 can be captured only during the last night of the month. This determines the availability of the changes from a source, and hence impacts the data freshness. It also impacts the data coherence because time discrepancies may occur in the view: the average may incorporate fresh data from S1 and old data from S2. Second, it depends on the time needed to compute the change to the view from the changes to the sources.

In fact, the two previous parameters may be repeated as many times as there are intermediate storages between the sources and the view. For instance, suppose that the result of the preparation step is stored. The availability parameter characterizes the moment at which the integration process is capable of accessing the result of a preparation step. Thus, if each result is only available at the end of the month then the integration can only be performed at that time and the view will consequently only reflect changes that occurred in the sources once per month.

Another parameter influences the result of a query against V. It characterizes the actualization of the data contained in each source. For instance, source S1 can be updated at the end of every week while source S2 is updated two days before the end of every month. If a query is posed against V at the end of the second week of a month, the effect of the phone calls that occurred since the beginning of the month in the country associated with source S2, will not be possibly reflected by V, and hence by the result of the query. Thus, the value of this parameter determines the difference that may exist between the state of the view reflected by the data warehouse and the state of the view in the real world. Because this

parameter is fixed and out of the control of the data warehouse application (it is actually part of the source operational applications), we do not consider it.

The previous discussion has shown how the refreshment process can depend on some parameters, independently of the choice of materialized views, and how these parameters impact on the semantics of the process. It also shows that building an efficient refreshment strategy with respect to application requirements (e.g. data freshness, computation time of queries and views, data accuracy) depends on various parameters related to:

- *source constraints* (e.g. availability windows, frequency of change),
- and *data warehouse system limits* (e.g. storage space limit, functional limits).

Finally the main lesson drawn from the previous examples and discussion is :

The operational semantics of the refreshment process can be defined as the set of all design decisions that contribute to provide to the users relevant data, fulfilling the quality requirements.

Some of these design decisions are inherited from the design of the initial loading, others are specific to the refreshment itself. The first design decisions inherited from the design of the initial loading may concern the view definition, the structure of the data flow which is between the sources and the data marts. The second design decisions inherited from the design of the initial loading are the semantics of the loading activities, that is cleaning rules, integration rules, etc.

The design decisions which are specific to the refreshment semantics are those that determine :

- the moment when each refreshment task takes place in the global process
- the way the different refreshment tasks are synchronized
- the way the shared data is made visible for the corresponding tasks

These design decisions are specified by defining :

- The decomposition of the refreshment process in elementary tasks (e.g. cleaning of some specific source, partial integration of two given changes originated from two different sources, detection and cleaning in a unique task for another source).
- Ordering of these tasks.
- The events initiating the tasks. The events put the rhythm into the refreshment process, and, depending on this rhythm, the freshness and the accuracy of the data may be quiet different.

5. Implementation issues

With respect to the implementation issues, different solutions can be considered. The conceptual definition of the refreshment process by means of a workflow, leads naturally to envision an implementation under the control of a common workflow system in the market, provided that this latter one supplies event types and all features needed by the refreshment scenario. Another solution we have preferred in [BFM+ 98] consists in using active rules which should be executed under a certain operational semantics. The rationale behind our choice is the flexibility and the evolutivity provided by active rules. Indeed the refreshment strategy is not defined once for all; it may evolve with the user needs, which may result in the change of the definition of materialized views or the change of desired quality factors. It may also evolve when the actual values of the quality factors slow down with the evolution of the data warehouse feeding or with the technology used to implement it. Consequently, in order to master the complexity and the evolutivity of the data warehouse, it is important to provide a flexible technology which allows to accommodate this complexity and evolutivity. This is what active rules meant to provide. A prototype has been developed and demonstrated in the context of the DWQ european research project on Data warehouses. However, active rules cannot be considered as an alternative to workflow representation. Workflow is a conceptual view of the refreshment process, while actives rules are operational implementation of the workflow.

6. Concluding remarks

This paper has presented an analysis of the refreshment process in data warehouse applications. We have demonstrated, that the refreshment process cannot be limited neither to a view maintenance process nor to a loading process. We have shown through a simple example, that the refreshment of a data warehouse can be conceptually viewed as a workflow process. We have identified the different tasks of the workflow and shown how they can be organized in different refreshment scenarios, leading

to different refreshment semantics. We have highlighted design decisions impacting over the refreshment semantics and we have shown how the decisions may be related to some quality factors such as data freshness and to some constraints such as source availability and accessibility.

References

[AAE+ 95] Alonso, G., Agrawal, D., El Abadi, A., Kamath, M., Gunther, R., Mohan, C., Advanced Transaction Models in Workflow Context, IBM Research Report, RJ 9970, IBM Research Division, 1995

[AGW 97] B.Adelberg, H. Garcia-Mollina, and J. Widom. The STRIP Rule System For Efficiently Maintaing Derived Data. *In Proc. of ACM SIGMOD International Conference on Management of Data*. Tucson, Arizona, USA, 1997.

[BeNe 98] Berstein, P., Newcomer, E., Principles of Transaction Processing, Morgan Kaufmann Publ., 1998.

[BFM+ 98] Bouzeghoub, M., Fabret, F., Matulovic, M., Simon, E., "A toolkit Approach for developing efficient and customizable active rule systems", DWQ Technical report, October 1998.

[BFM+ 99] Bouzeghoub, M., Fabret, F., Galhardas, H., Matulovic, M., Pereira, J., Simon, E., "Data Warehouse Refreshment", in Fundamentals in Data Warehouses, Chapter 4, M. Jarker et al (edts), Springer, 1999.

[BMF+ 97] Bouzeghoub, M., Fabret, F., Llirbat, L., Matulovic, M., Simon, E., "Designing data warehouse refreshment system", DWQ Technical report, October 1997.

[CGL+ 96] L.Colby, T. Griffin, L. Libkin, I. S. Mumick, and H. Trickey. Algorithms for Deferred View Maintenance. In *Proceedings of SIGMOD*, Montreal, Canada, 1996.

[ChD 97] S. Chaudhuri, U. Dayal. *An Overview of Data Warehousing and OLAP Technology*. SIGMOD Record, Vol. 26, No. 1, March 1997

[CCPP 95] Casati, F., Ceri, S., Pernici, B., Possi, G., Conceptual Modeling of Workflows, Proceed. Of the Internat. Conf. On Object Oriented and Entity-Relationship Approach, Austr. Springer, 1995.

[CSCW 96] ACM 1996 Conference on Computer-Supported Cooperative Work – Cooperating Communities (Proceedings). Ackerman, M.S. (ed.), ACM, Boston, MA, November 1996.

[HaC 93] Hammer, M., Champy, J., Reengineering the Corporation, a Manifesto for Business Revolution, Harper, New-York, 1993.

[JaV 97] M. Jarke, M. Vassiliou: Foundations of data warehouse quality: an overview of the DWQ project. *Proceedings 2nd International Conference on Information Quality*, Cambridge, Mass, 1997

[Lawr 97] Lawrence, P. (ed.), Workflow Handbook, Wiley and WfMC, 1997.

[Sch 98] Schael, T., Workflow Management Systems for Process Organisations, Second edition, Springer, 1998.

[ThBo 99] Theodoratos, D., Bouzeghoub, M., Data Currency Quality Factors in Data Warehouse Design, Proceed. of the International Workshop on Design and Management of Data Warehouses (DMDW'99), Heidelberg, Germany, June 1999.

[WAN 98] Wang, R. Y., "A product perspective on total data quality management", Com. [ZGJ+ 95] Yue Zhuge, Hector Garcia-Molina, Joachim Hammer, and Jennifer Widom. View maintenance in a warehousing environment. In Proc. of the ACM SIGMOD Int. Conf. on Management of Data, pages 316-327, 1995.

[WoKr 93] Woetzel, G., Kreifelts, T., The use of Petri nets for modeling workflow with the Domino system. Proceed. of the Workshop on Computer Supported Cooperative Work, Petri nets and Related Formalisms. Chicago, 1993.

[WFMC 95] WFMC-TC-1003 (The Workflow Management Coalition), The Workflow Reference Model, version 1.1, January 1995.

[ZGJ+ 95] Zhuge, Y., Garcia-Molina, H., Hammer, J., Widom, J., View Maintenance in a warehousing environment, Proceedings of the ACM SIGMOD Int. Conf. On Management of Data, P 316-327, 1995.