A Decision Support System that Reverse Engineers Abstract Database Transactions - The Conceptual Model

A. Sulaiman [1,2,3] & J. Souza [1]
[1] *Coordenação dos Programas de Pós-graduação em Engenharia, COPPE/UFRJ.* [2] *Laboratório Nacional de Computação Científica - LNCC/CNPq.* [3] *Universidade Santa Úrsula - USU.* [1] *Cidade Universitária, Centro de Tecnologia, bloco H, sala 319, Ilha do Fundão, Rio de Janeiro, RJ. Caixa Postal: 68513.*
EMail: sulaiman@lncc.br, jano@cos.ufrj.br

Abstract

Traditional methods of database integration don't solve applications integration issues. These methods work on database schema. Beyond the complexity of integrating names and meanings between data and metadata of heterogeneous databases, is the understanding of the business that originated applications. Database transactions should have been projected to reflect the business activities. In general, differences between abstractions levels don't let database transactions mirror business activities. Business process consists of separate activities. Sometimes database transactions solve part of a defined activity. Otherwise more than one activity can be embedded in a database transaction. Database log records all occurrences of database transactions; it has their complete histories. Database log is an endogenous datawarehouse: In this datawarehouse one can mine rules, which will help understanding the nature of the business task and the adequacy of the related database logical model. This paper describes a conceptual model for a Decision Support System that will help database administrators on reverse engineering legacy system applications from its database logs, generating rules about transactions, to retrieve business process. That approach will complement Conceptual Schema Integration.

1 Introduction

The database log is a kind of datawarehouse. In this datawarehouse one can mine rules, as described in [1], which will help understanding the nature of the business process and its adequacy to the database logical model.

As defined in [2], a business process is a set of activities. But differently from [2], where a process execution with its activities occurrences is completely represented in a workflow log record, there is no assurance that a database log record will represent completely an activity.

There are important differences between workflow software and database software, among others: in a workflow software, the kind of diagrams used symbolizes business process and its activities; in a database software the emphasis isn't the business process, but the information transaction (On Line Transaction Processing - OLTP).

Using Zachman Framework [3][4] one can understand these important differences related to the abstraction axes (workflow software is well suited to the functional axe and database software to the data axe) and to the abstraction level (workflow software is to represent knowledge and database software represents information and data). So the granularity of information is distinct in each case.

In general database log transactions grouped together stands for an activity. Sometimes database transactions solve part of a defined activity. Otherwise more than one activity can be embedded in a database transaction. Abstract transaction is the name we are using to represent an activity implemented in a database.

Mining database log transaction one can discover knowledge about functional and control abstraction axe. The rules that represent that knowledge can be combined with the rules inferred from the metadata (data abstraction axe), in the database integration.

In general, there is no documentation about business process in legacy systems, and a software to retrieving it (even partially) should be desirable, for economical reasons.

Next versions of the model will work on federated databases (network axe).

The rest of this paper is organized as follows. In section 2, we compare the nature of a database log and a datawarehouse and formulate the hypothesis of an endogenous datawarehouse. In section 3, we formulate the hypothesis of an abstract transaction as an computational interpretation of a business activity. In section 4, we review the concept of a production rule server. In section 5, we review the concept of mining

The rest of this paper is organized as follows. In section 2, we compare the nature of a database log and a datawarehouse and formulate the hypothesis of an endogenous datawarehouse. In section 3, we formulate the hypothesis of an abstract transaction as an computational interpretation of a business activity. In section 4, we review the concept of a production rule server. In section 5, we review the concept of mining association rules. In section 6 we model a decision support system that reverse engineers abstract database transactions. In section 7 we conclude with a summary.

2 An Endogenous Datawarehouse

Is possible to develop some taxonomies to computer applications. One of them can be based on the necessity of modeling one or more abstraction axes. Another possibility is to classify them by the exogenous or endogenous nature of the application.

Exogenous applications in relation to databases, are generic programs that belong to the users and specialists universe. Stock control and bank transactions are related to users. Bank loan and medical advice are related to specialists.

Endogenous applications in relation to databases, are system administration programs. Typical example is the integrity constraint control in a DBMS. Other examples are: periodic file reorganization, and backup.

A Datawarehouse is "a subject oriented, integrated, time-variant, and non-volatile collection of data in support of management's decision making process" [5].

A Database log is "the temporal database. The online sessions, tables, contexts, queues and other durable objects are just their current versions. The log has their complete histories." [6]. And "the log manager provides read and write access to the log table for all other resource managers and for the transaction manager" [6]].

Our Hypothesis is that database logs can be viewed as datawarehouses:

- Database logs are subject oriented. The subject is the record of a database transaction;
- Database logs are integrated. Once the log is the history of the transactions, the information obtained is semantically related to the application;

- Database logs are time-variant. They record each modification in each object at each time;
- Database logs are non-volatile. They have to be preserved to prevent crashes, but also to let audit checking.

Our point of view is that the log manager software is an endogenous application and the database log is an endogenous datawarehouse where one can mine rules to the database administrator's decision support.

3 Abstract Transactions

Observing the Zachman framework [3] one can realize that there are important differences between the business point of view and the computational point of view about a task to be performed.

One of the most important differences is the semantical one. In modeling business one can define business process as a set of separate activities [2]. And an activity can be defined as an action that is a semantical unit at some level, or as a function that modifies the state of a process [2].

On the other hand, from the computational point of view, a transaction is the execution of a program that access or changes the contents of the database, and an atomic transaction is an atomic unit of work, or a logical unit of database processing [7].

When building a system, one adequates business models to computational models. A semantical gap then is found: none all the meaning stated for the business can be built by a piece of software program. And the part that can be built doesn't necessarily obey the business logic, but the computational logic.

In general, database transactions grouped together stands for an activity. Sometimes database transactions solve part of a defined activity. Otherwise more than one activity can be embedded in a database transaction.

Abstract transaction is the name we are using to represent an activity (or the computational part of the activity) implemented in a database.

4 A Production Rule Server

One of our goals is to record facts and rules automatically generated. We will use our production rule server [8][9][10][11], to use the DBMS to store facts and rules homogeneously.

With this rule server we can perform forward and backward chaining. If we choose a relational environment, the methods can be implemented through a third generation language that supports recursion and embedded SQL call.

With this model, the rules can be stored with the facts in the same DBMS. If an external event happens, it can change the state of a clause, firing a rule. The rule server can be implemented in any DBMS.

The Venn diagram represents all of our transactions and the possible states of each transaction. From [12] the assertion: "generalization is an OR relationship", one can deduce that the set of the clauses is made by *hypothesis U facts U premises U conclusions*. But the point is that a clause can be simultaneously in two or even three of these states. There are eight possible states represented below [9]:

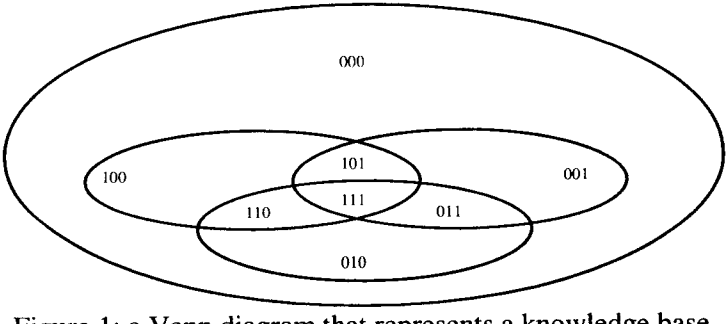


Figure 1: a Venn diagram that represents a knowledge base.

CLAUSE STATE TABLE	
000	Hypothesis in its initial state. When the knowledge base receives a message this state must change or the inferencing cannot happen. There is no knowledge.
001	Exclusive premise. In this case this premise will never be fired, if the clause does not change its state.
010	Exclusive conclusion. In this case this conclusion will never be reached, if the clause does not change its state.
100	Exclusive fact. This is a truth. If this is an initial hypothesis, it is proved.
011	Premise and conclusion. It must change its state, or the clause can't be proved.
101	Fact and premise. The clause is proved. The knowledge base must be reorganized.
110	Fact and conclusion. The clause is proved. The knowledge base must be reorganized.
111	Fact and premise and conclusion. The clause is proved. The knowledge base must be reorganized.

Figure 2: a clause state table.

5. Mining Association Rules

With the fact base (the Database log) and the rule server, the next step is to generate the rules in the rule server to represent the knowledge about database transactions. Borrowing the idea from [1], originally used for exogenous applications, one can use new functionalities, endogenous by nature, and search over the log datawarehouse as follows:

- Find all rules that have "delete record from table A" as a consequent. These rules may help to built a CRUD matrix [13][14], from a legacy system for future data planning and distribution;

- Find all rules that have "insert record into table B" in the antecedent. These rules may help determine the impact of that operation on other transactions;
- Find all rules that have "insert record into table C" in the antecedent AND "delete record from table D". This rules can help in finding same patterns of behavior in different transactions;
- Find all rules relating operations over tables X and Y in the schema Z. These rules may help Information System Planning, by determining if the operations over table X from transaction Q are related to the operations over table Y from transaction W.

Reading [1] one will find the analogy that has been done here between store shelves and the database conceptual and logic schema, between a selling transaction and database transaction, and between item sold and insert/delete operations.

6 The Conceptual Model

The definition of the granularity of a datawarehouse, is one of the first steps recommended in the literature [immon92][4]. Our grain is the atomic transaction recorded in a transaction database log.

To do so, one of the desired properties of database transactions is the "A" of ACID properties: Atomicity. "A transaction is an atomic unit of processing; it is either performed in its entirety or not performed at all" [7].

There are long transactions that, if implemented, don't obey the atomicity principle. In general nested transactions are necessary to implement them. However using traditional relational DBMS and hence under the atomicity principle, one must break a long transaction in small (atomic) pieces.

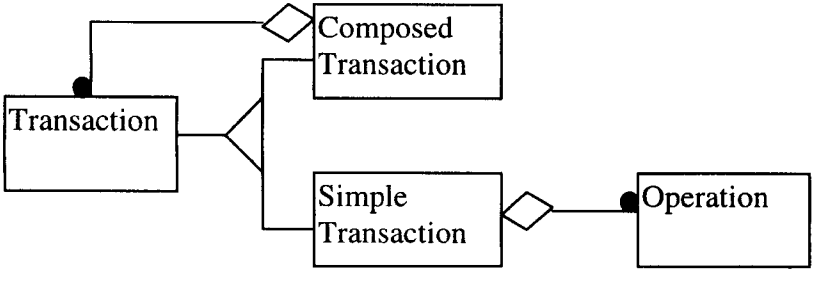


Figure 3: nested transaction

Other principle, Consistency, says "A correct execution of the transaction must take the database from one consistent state to another"[7]. If the business address a group of atomic transactions to perform a task, there is no warranty at all: the transaction will be atomic and consistent, but can happen that the task doesn't.
In this work we are calling a group of atomic transactions aggregated by a business reason as "abstract transactions":

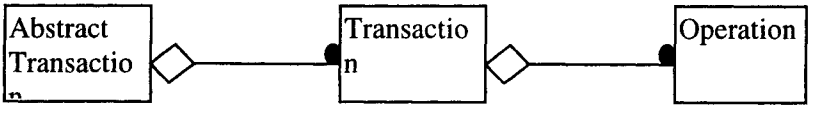


Figure 4: abstract transaction

On the other hand there are atomic transactions contained in more than one abstract transaction, to be performed in other place and/or time:

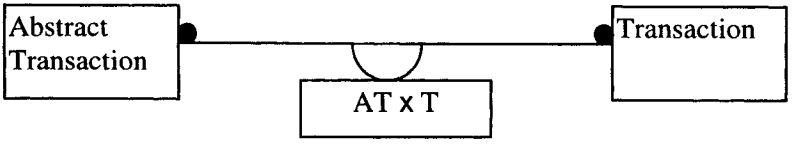


Figure 5: abstract transactions versus simple transactions

To generate the rules automatically using the production rule server one have to see the transaction as a clause (fact or premise or conclusion [12]):

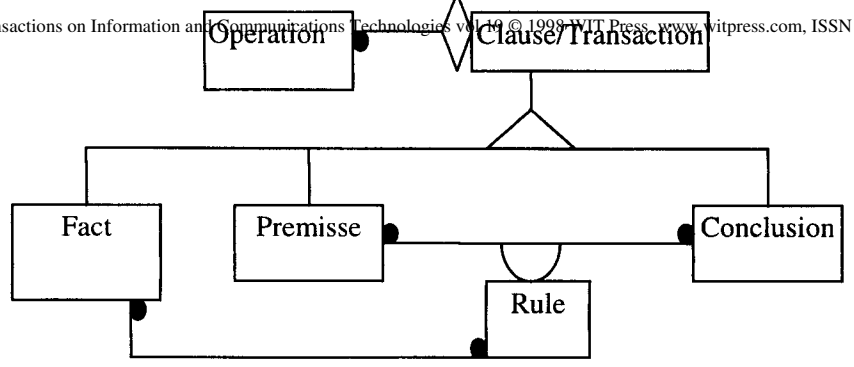
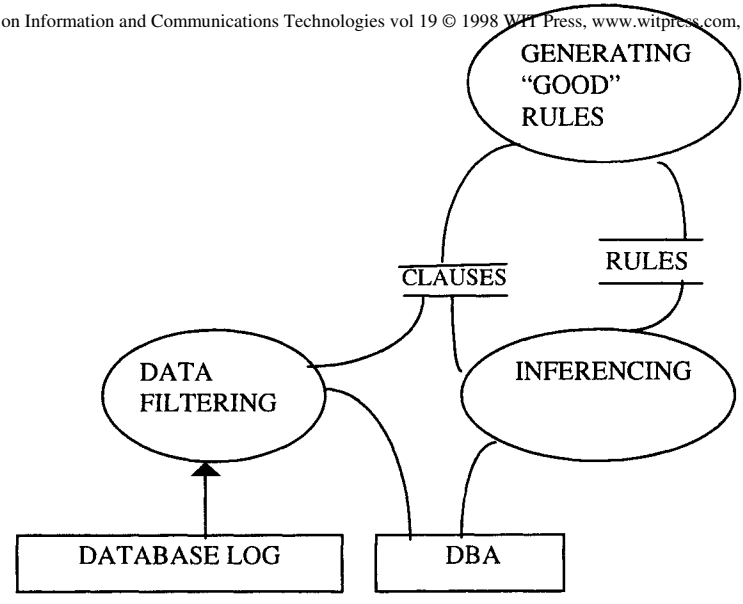


Figure 6: An object model to the Decision Support System

This conceptual model is coherent with the Knowledge Discovery in Databases process [KDD95][KDD96];

1. The application domain is an endogenous datawarehouse: the database log;
2. To choose which group of logs will be searched, by period of time or by business unit, for example;
3. To do the cleaning and pre-processing;
4. To generate the desired rules, searching the significant rules, using the algorithm from [1];
5. To do inference over the patterns obtained (by the specialist, the DBA);
6. To interpret the results (by the specialist, the DBA);
7. To reconstruct hypothetical business process.



High abstraction level Data Flow Diagram

7. Summary

In this work we presented the Conceptual Model of A Decision Support System that Reverse Engineers Abstract Database Transactions. Related work was cited. We introduced the concepts of an Endogenous Datawarehouse and Abstract Transactions.

References

[1] R. Agrawal, T. Imielinski, A Swami - Mining Rules between Sets of Items in Large Databases - Proceedings of the ACM SIGMOD, 1993.

[2] Agrawal R., Gunopulos D, Leymann F. - Mining Process Models from Workflow Logs, 1998.

[3] Zachman J. - A Framework for Information Systems Architecture, IBM Systems Journal, vol. 26, no. 3, 1987, IBM publication G3212-5298.
[4] Immon W., Zachman J. & Geiger J. - Data Stores Data Warehousing and the Zachman Framework - McGraw-Hill, 1997.
[5] Immon W. H. - Building the Data Warehouse, QED Technical Publishing Group, 1992.
[6] J. Gray & A Reuter - Transaction Processing: Concepts and Techniques - Morgan Kaufmann, 1993.
[7] R. Elmasri, S. Navathe - Fundamentals of Database Systens - Benjamin/Cummings, 1994.
[8] A Sulaiman - An Inteligent Decision Support Environment for Demographic and Statistics Applications. Ms Dissertation (in portuguese), Engineering Militar Institute, IME-RJ, 1992.
[9] E. Passos, A Sulaiman, C. Garcez, A Tanaka - A Conceptual Model for a Knowledge Base Homogeneously Stored in a Database Environment - Lecture Notes in Artificial Intelligence 991, 12th Brazilian Symposium on Artificial Intelligence, SBIA'95 -, October 1995.
[10] A Sulaiman, M. Mattoso, J. Souza - An Expert System Shell built using O2, USE O2! (in Portuguese) - II Unconventional Database Workshop - UFF - December 1995.
[11]A Sulaiman & G. Xexéo - Developers Magazine, Mai. 97 - A Production Rule Server (in portuguese).
[12] J. Rumbaugh et alii. - Object-Oriented Modeling and Design - Prentice Hall Inc, 1991.
[13] Business System Planning. Information Systems Planning Guide, 2nd ed. IBM, 1978.
[14] J. Martin - Strategic Data Planning Methodologies - Prentice-Hall, 1982.