WanD: A CASE Tool for Workload-Based Design of a Data Mart

Matteo Golfarelli, Stefano Rizzi, and Ettore Saltarelli

DEIS - University of Bologna
Viale Risorgimento, 2
40136 Bologna - Italy
{mgolfarelli, srizzi, esaltarelli}@deis.unibo.it

Abstract. The goal of this demonstration is to present the main features of WanD, the prototype CASE tool we have implemented to support our methodology for data warehouse design on ROLAP architectures. WanD assists the designer in building a data mart: it carries out conceptual design in a semi-automatic fashion from relational operational sources, allows for a core workload to be defined on the conceptual scheme, acquires the data volume, and carries out logical and physical design to produce the data mart scheme. In output, the SQL statements for creating tables and indexes and for feeding them from the operational sources are generated.

1 Motivation and overview

There is substantial agreement on the fact that, when planning a *data warehouse* (DW) , a bottom-up approach should be followed: the *data mart* playing the most strategic role for the enterprise should be identified and prototyped first in order to convince the final users of the potential benefits; other data marts are built progressively, to be finally integrated bottom-up into the global warehouse. We believe that a methodological framework for data mart design is an essential requirement to ensure the success of complex projects; in this direction, our past research was aimed at defining the basic steps required for a correct design of ROLAP architectures.

It is well-known among software engineers that devising a design methodology is almost useless, if no CASE tool to support it is provided. The goal of this demonstration is to present the main features of WanD (Warehouse Integrated Designer), the prototype CASE tool we have implemented to support our methodology. WanD assists the designer in building a data mart: it carries out conceptual design in a semi-automatic fashion starting from the relational scheme of the operational database (read via ODBC), allows for a core workload to be defined on the conceptual scheme, acquires the data volume, and carries out logical and physical design to produce the data mart scheme. The conceptual model adopted is the *Dimensional Fact Model* (DFM), described in [4]. Star schemes are used on the logical level, while tid-list and bitmap indexes are created on the physical level.

The demonstration will be based on a sample operational database describing a supply chain, and on the standard TPC-H database.

2 The methodological framework

According to our methodology, outlined in [6], each data mart is mainly created in a top-down fashion by building a fact scheme for each fact of interest. The design steps are summarized in Table 1; those actively supported by WanD are steps 3, 4, 5, and 7. As a matter of fact, our on-the-field experience showed that also step 2 (requirement specification) can be fruitfully carried out by relying on WanD to quickly sketch different conceptual scenarios to be submitted to the users and interactively tuned to meet their expectations. Besides, step 6 is partially supported by generating the loading queries for the data mart.

Table 1. The seven phases in data mart design.

Step	*Input*	*Output*	*Involves*
Analysis of the operational database	existing documentation	(reconciled) database scheme	designer; managers of the information system
Requirement specification	database scheme	facts; preliminary workload	designer; final users
Conceptual design	database scheme; facts; preliminary workload	conceptual scheme	designer; final users
Workload refinement; conc. scheme validation	conceptual scheme; preliminary workload	validated conceptual scheme; workload	designer; final users
Logical design	conceptual scheme; data volume; workload	logical scheme	designer
ETL design	database scheme; logical scheme	ETL scripts	designer; managers of the information system
Physical design	logical scheme; target DBMS; workload	physical scheme	designer

3 The WanD architecture

The functional architecture of WanD is sketched in Figure 1. The main functions are shown, together with the actors who carry them out and the main archives used. Each functions is briefly described in the following.

- *Acquisition of the relational scheme describing the operational database.* The scheme is acquired by querying, via ODBC, the DBMS hosting the database. If necessary, the designer can then edit the scheme (for instance, by defining primary and foreign keys if they are not defined in the source scheme or if the ODBC driver used does not support them). It should be noted that WanD is not an ETL or scheme integration tool, thus we assume that source schemes do not present any conflicts or, more pragmatically, that an integrated scheme is already available. Scheme integration may have been carried out beforehand in a semi-automatic fashion using one of the tools proposed in the literature (see [2] for instance).

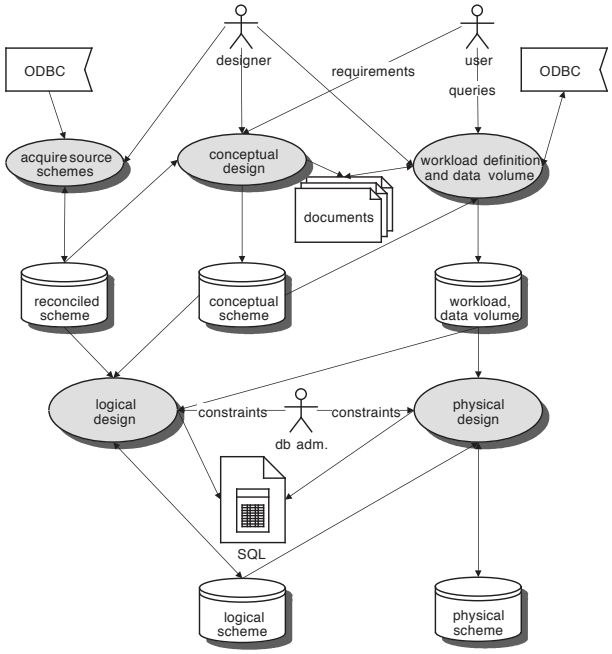

Fig. 1. The functional architecture of WAND.

- *Conceptual design.* Conceptual design is carried out semi-automatically from the scheme of the operational database, according to the algorithm proposed in [4]. The output is a set of fact schemes, one for each fact of interest (see Figure 2).
- *Workload definition.* The designer defines the most frequent/interesting queries on fact schemes through a simple query language. The queries express aggregations over the star join between a fact table and a set of dimension tables; selection predicates may be formulated on both attributes in dimension tables and measures in the fact table. The workload expressed is checked against the conceptual scheme, thus allowing its validation. The workload is then used to drive both logical and physical design.
- *Data volume acquisition.* WAND determines the current data volume, to be used for logical and physical design, by querying the operational database via ODBC. A relevant problem is that of estimating the cardinalities of aggregate views, which is necessary for correctly applying view materialization techniques but is made complex by the sparsity of data. The usual techniques proposed in the literature are based only on probabilistic criteria and often lead to unreliable results. In order to obtain better estimates, in WAND the designer can define a set of *cardinality constraints*, suggested by the experts of the application domain, used to compute satisfactory bounds to the cardinalities [3].

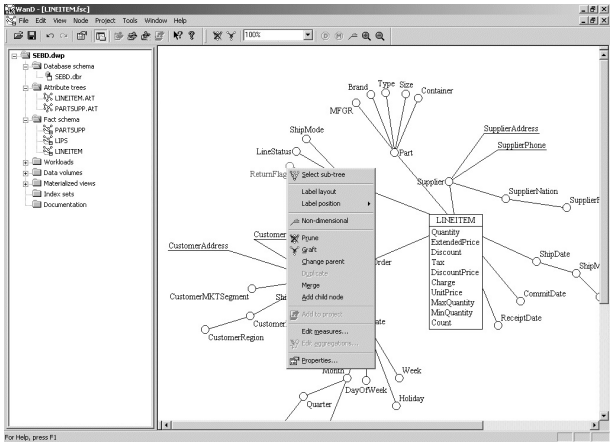

Fig. 2. Editing the fact scheme.

- *Logical design*. During this phase, the conceptual schemes previously defined are translated into relational tables according to the well-know star-scheme. For each fact scheme defined, and taking both the workload and the data volume into account, WAND determines the best set of aggregations to be materialized in order to minimize the query response time. Data are materialized in *vertical fragments* which may store any subset of the measures in the source fact scheme. Materialization of fragments is a more sophisticated technique than the classical all-or-nothing-approach to view materialization (see [1] for example), and it was shown to determine better performances [8]. The algorithm adopted is based on an approach in two steps: first it determines all the potentially useful fragments (*candidate fragments*); then, using a linear programming technique [5], it selects the best subset fitting the system constraints. Since the time complexity for determining the best solution in real cases may be too high, also a heuristic approach is implemented. In output, WAND automatically generates the SQL statements defining fact and dimension tables according to the star scheme, as well as the SQL queries allowing these tables to be populated from the operational database (Figure 3).
- *Physical design*. Given the logical scheme (including materialized fragments), the workload, the data volume and a constraint defining the disk space devoted to indexes, this function determines the index set that minimizes the workload execution cost respecting the space constraint. It is based on a rule-based optimizer model capable of determining an execution plan for each query, and on a cost model to compare different plans. The indexes considered are tid-list and bitmap indexes, both accessed via B^+-trees. A set of potentially useful *candidate indexes* is preliminarily determined considering the workload; then, a greedy algorithm progressively chooses, from

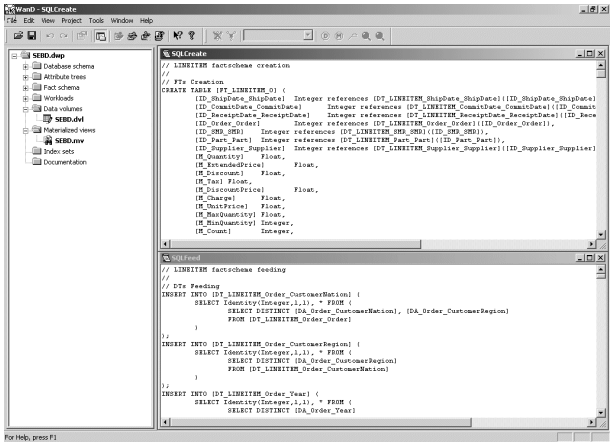

Fig. 3. The SQL statements for creating and feeding the data mart.

the set of candidate indexes, the most beneficial ones while satisfying the space constraint [7].

– *Production of the documentation.* The documentation produced includes the fact schemes designed, the attribute and measure glossaries, the workload and the data volume, and the SQL statements for creating and populating the data mart.

References

1. E. Baralis, S. Paraboschi, and E. Teniente. Materialized view selection in multidimensional database. In *Proc. 23rd VLDB*, pages 156–165, Athens, 1997.
2. D. Beneventano, S. Bergamaschi, S. Castano, A. Corni, R. Guidetti, G. Malvezzi, M. Melchiori, and M. Vincini. Information Integration: the MOMIS Project Demonstration. In *Proc. 26th VLDB*, Cairo, Egypt, 2000.
3. P. Ciaccia, M. Golfarelli, and S. Rizzi. On estimating the cardinality of aggregate views. In *Proc. DMDW*, pages 12.–12.10, Interlaken, Switzerland, 2001.
4. M. Golfarelli, D. Maio, and S. Rizzi. The dimensional fact model: a conceptual model for data warehouses. *Int. Journal of Cooperative Information Systems*, 7(2-3):215–247, 1998.
5. M. Golfarelli, D. Maio, and S. Rizzi. Applying Vertical Fragmentation Techniques in Logical Design of Multidimensional Databases. In *Proc. DaWaK*, pages 11–23, Greenwich, 2000.
6. M. Golfarelli and S. Rizzi. Designing the data warehouse: key steps and crucial issues. *Journal of Computer Science and Information Management*, 2(3), 1999.
7. M. Golfarelli, S. Rizzi, and E. Saltarelli. Index selection for data warehousing. In *Proc. DMDW*, Toronto, Canada, 2002.
8. V. Maniezzo, A. Carbonaro, M. Golfarelli, and S. Rizzi. ANTS for Data Warehouse Logical Design. In *Proc. 4th Metaheuristics Int. Conf.*, pages 249–254, Porto, 2001.