QBX: A CASE Tool for Data Mart Design

Antonino Battaglia[1], Matteo Golfarelli[2], and Stefano Rizzi[2]

[1] Theorematica S.p.A., Rome, Italy
a.battaglia@theorematica.it
[2] DEIS - University of Bologna, Italy
{matteo.golfarelli,stefano.rizzi}@unibo.it

Abstract. QBX is a CASE tool for data mart design resulting from a close collaboration between academy and industry. It supports designers during conceptual design, logical design, and deployment of ROLAP data marts in the form of star/snowflake schemata, and it can also be used by business users to interactively explore project-related knowledge at different levels of abstraction. We will demonstrate QBX functionalities focusing on both forward and reverse engineering scenarios.

1 Introduction and Motivation

The continuous market evolution and the increasing competition among companies solicit organizations to improve their ability to foresee customer demand and create new business opportunities. In this direction, data warehouses have become an essential element for strategic analyses. However, data warehouse systems are characterized by a long and expensive development process that hardly meets the ambitious requirements of today's market. This is one of the main causes behind the low penetration of data warehouse systems in small-medium firms, and even behind the failure of whole projects [4].

A data warehouse is incrementally built by designing and implementing one data mart at a time; so, one of the directions to increase the efficiency of the data warehouse development process is to automate design of single data marts. Several techniques for automating some phases of data mart design have been proposed in the literature (e.g., [2] for conceptual design, [5] for logical design, [3] for physical design, [6] for designing the ETL process), and some research prototypes of CASE tools have been developed (e.g., [1]). On the other hand, commercial tools such as Oracle Warehouse Builder are oriented to a single platform and should be considered as design wizards rather than CASE tools.

In this paper we introduce *QBX*, a CASE tool resulting from a close collaboration between academy and industry; in particular, industrial partners took care of the executive design and implementation of the tool. QBX includes two separate components: *QB-Xpose* (read *cube-expose*) and *QB-Xplore* (read *cube-explore*). The first is used by designers for conceptual design, logical design, and deployment of ROLAP data marts in the form of star/snowflake schemata; the design process is further streamlined by letting QBX read from and write to the

O. De Troyer et al. (Eds.): ER 2011 Workshops, LNCS 6999, pp. 358–363, 2011.

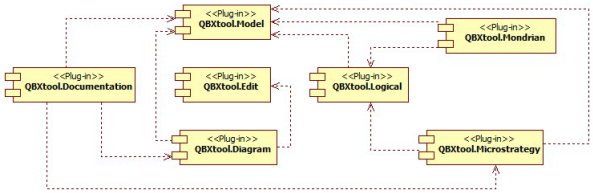

Fig. 1. Simplified component diagram for QB-Xpose

metadata repositories of the Mondrian and Microstrategy multidimensional engines. The second is accessed via browser by business users and technical experts to interactively explore project-related knowledge at different levels of abstraction, ranging from technical documentation to glossaries of terms and concepts based on the business users vocabulary.

2 Architecture

As already mentioned, QBX includes two integrated software tools.

QB-Xpose gives designers an effective support by automating conceptual and logical design of data marts, and by making deployment on ROLAP platforms easier. It was implemented based on *Eclipse* [7], an open source project providing an extensible development platform. The QB-Xpose components were developed in accordance with the Eclipse plug-in contract using three complementary frameworks: Eclipse Modeling Framework (EMF), Graphical Editing Framework (GEF), and Graphical Modeling Framework (GMF). Figure 1 shows the main components of QB-Xpose and their dependencies. Three components implement the model-view-controller design pattern: QBXtool.Model uses the EMF and is responsible for managing the QBX model; QBXtool.Edit and QBXtool.Diagram use the EMF and the GMF, and play the roles of the controller and of the viewer, respectively. QBXtool.Logical supports logical design, while QBXtool.Mondrian and QBXtool.Microstrategy manage the conversion of the QBX meta-model to/from the Mondrian and Microstrategy meta-models.

QB-Xplore enables business users to interactively browse and annotate the project documentation, both at business and technical levels. QB-Xplore is a web application implemented using the Google Web Toolkit. The underlying model was built using the EMF to achieve higher efficiency when exchanging information with QB-Xpose.

3 Functional Overview

From a methodological point of view, QBX supports both classical scenarios of data mart design [8]:

– *Demand-driven approach*, where designers draw their data mart starting from user requirements, possibly by composing existing hierarchies and

reusing conformed dimensions. Establishing a mapping with the source operational schema is postponed to the ETL design phase.
- *Supply-driven approach*, where a data mart schema is semi-automatically derived from the schema of a source operational database. User requirements help designers choose facts, dimensions, and measures. Mappings between the data mart schema and the source schema make ETL design simpler.[1]

A basic feature of QBX is that of using conceptual schemata for multidimensional design. Conceptual modeling provides a high level of abstraction in describing the multidimensional repository, aimed at achieving independence of implementation issues. It is widely recognized to be the necessary foundation for building a database that is well-documented and fully satisfies user requirements; usually, it relies on a graphical notation that facilitates writing, understanding, and managing conceptual schemata by both designers and business users. The conceptual model adopted by QBX is the Dimensional Fact Model (DFM) [2]. The DFM was born in the academic context and it has been widely experimented in the industrial world; its main goals are to lend effective support to conceptual design, to make communication possible between designers and end-users with the goal of formalizing requirement specifications, to build a stable platform for logical design, and to provide clear and expressive design documentation. The DFM gives a graphical and intuitive representation of facts, measures, and dimensions; dimensional, descriptive, and cross-dimensional attributes; optional and multiple arcs; convergences; shared, incomplete, and recursive hierarchies; additivity; temporal scenarios.

The adoption of a conceptual model breaks design into two distinct but interrelated phases, that are largely independent of the features of the OLAP engine chosen for deployment:

1. *Conceptual Design.* This phase maps user requirements into a conceptual schema of the data mart, that is, a platform-independent, non-ambiguous, comprehensive representation of the facts for decision support that gives a multidimensional picture of the data mart content to both designers and business users. In a supply-driven approach, conceptual design is automated by choosing relevant facts on a source operational database schema and letting QBX draw hierarchies using the approach in [2]. In a demand-driven approach, a conceptual schema is created from scratch by manually drawing hierarchies and reusing conformed dimensions.
2. *Logical Design.* Starting from a conceptual schema, QBX implements a large set of best practices to create an optimized logical schema for the data mart, that is, a set of star/snowflake schemata. The resulting logical schema can be fine-tuned based on the expected data volume and on the designer's preferences.

[1] This function will be made available starting from release 2.0. In case two or more databases are used to feed the data mart, QBX derives the data mart schema from the schema of the *operational data store* (ODS) that integrates the source databases. Database integration is out of the scope of QBX.

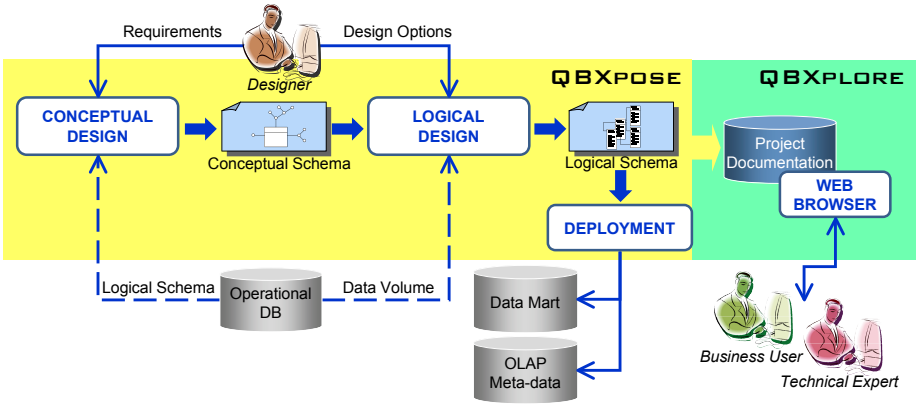

Fig. 2. Forward engineering with QBX

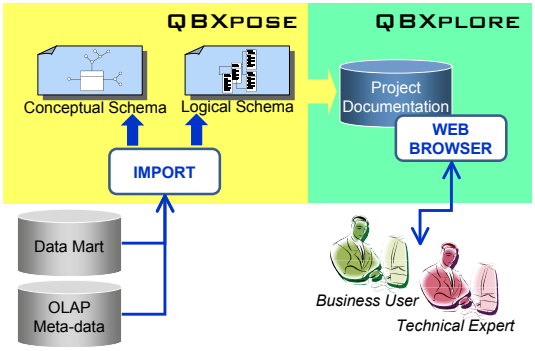

Fig. 3. Reverse engineering with QBX

In this forward engineering scenario (sketched in Figure 2), logical design is followed by a third phase:

3. *Deployment.* Based on both a conceptual and a logical schema, QBX generates SQL code for the underlying relational DBMS and writes the corresponding meta-data on the chosen OLAP engine (either Microstrategy or Mondrian in this release). Since the expressiveness of the DFM (in terms of multidimensional constructs) is wider than the ones of the meta-models of both Microstrategy and Mondrian, the actual structure of the deployed schemata depends on the capabilities of the OLAP engine selected. Similarly, QBX adapts the syntax of the generated SQL statements to the one of the adopted DBMS.

Further QBX functionalities include:

– *Reverse engineering.* To enable designers and business users to enjoy a more expressive view of an existing data mart, QBX can also be employed to

acquire metadata from an OLAP engine and translate them into a DFM conceptual schema. This scenario is sketched in Figure 3.

- *Logical design preferences.* To enable a finer tuning of logical schemata, designers can express a set of preferences about logical design, including e.g. how to deal with degenerate dimensions, shared hierarchies, and cross-dimensional attributes.
- *Data volume.* QBX enables designers to specify the data volume for a data mart, in terms of expected cardinalities for both facts and attributes. This is done manually in a demand-driven scenario, automatically in a supply-driven scenario. When a data volume has been specified, designers can ask QBX to optimize ROLAP schemata from the point of view of their storage space.

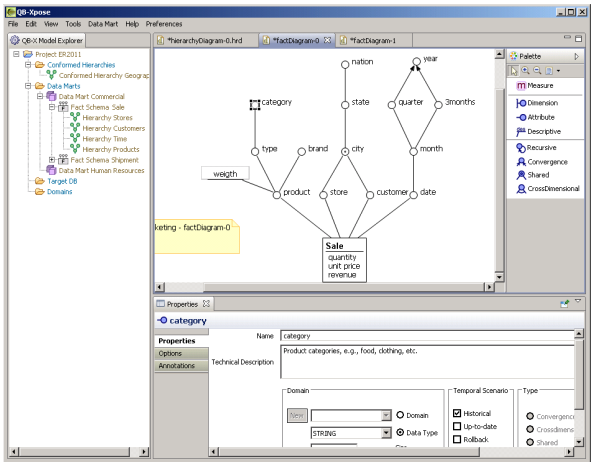

Fig. 4. Conceptual design with QBX

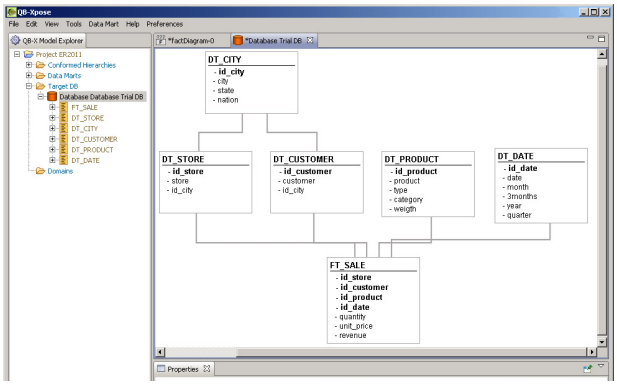

Fig. 5. Logical design with QBX

– *Project-related knowledge.* QB-Xpose automatically creates the documentation for data marts, including fact schemata, conformed dimensions, glossaries, and data volumes. This technical documentation, that can be published on the web and easily browsed via QB-Xplore, is only a part of the project knowledge, that also includes the business descriptions of terms and concepts appearing in reports and analyses. This precious information can be collected from domain experts during requirement analysis using QB-Xpose, or it can be progressively acquired in the form of annotations made by business users using QB-Xplore.

4 Demonstration Scenarios

The demonstration will focus on both forward and reverse engineering scenarios. In the forward engineering scenario, we will adopt a demand-driven approach to interactively draw a conceptual schema (Figure 4), showing how QB-Xpose automatically checks for hierarchy consistency in presence of advanced constructs of the DFM. Then, we will let QB-Xpose generate alternative logical schemata using different design preferences, and critically compare the results (Figure 5). Finally, we will let QB-Xpose create a comprehensive documentation for the project.

In the reverse engineering scenario, the relational schema and metadata of an existing data mart will be imported from the Mondrian engine, and the effectiveness of their translation to the DFM will be discussed.

References

1. Golfarelli, M., Rizzi, S.: WAND: A CASE tool for data warehouse design. In: Proc. ICDE, pp. 7–9 (2001)
2. Golfarelli, M., Rizzi, S.: Data warehouse design: Modern principles and methodologies. McGraw-Hill, New York (2009)
3. Golfarelli, M., Rizzi, S., Saltarelli, E.: Index selection for data warehousing. In: Proc. DMDW, pp. 33–42 (2002)
4. Ramamurthy, K., Sen, A., Sinha, A.P.: An empirical investigation of the key determinants of data warehouse adoption. Decision Support Systems 44(4), 817–841 (2008)
5. Theodoratos, D., Sellis, T.: Designing data warehouses. Data & Knowledge Engineering 31(3), 279–301 (1999)
6. Vassiliadis, P., Simitsis, A., Georgantas, P., Terrovitis, M., Skiadopoulos, S.: A generic and customizable framework for the design of ETL scenarios. Information Systems 30(7), 492–525 (2005)
7. Vv. Aa.: Eclipse. http://www.eclipse.org/platform/ (2011)
8. Winter, R., Strauch, B.: A method for demand-driven information requirements analysis in data warehousing projects. In: Proc. HICSS, pp. 1359–1365 (2003)