Derivation of initial data warehouse structure by mapping operational database on transaction patterns

A. Khalfe & S. I. Hyder
College of Compter Science,
PAF - Karachi Institute of Economics and Technology, Pakistan

Abstract

Data warehouses improve the quality of integrated information in the organization for decision-making. The data for the data warehouse comes from online transaction systems. Typically, an involved process of analysis precedes the actual design phase of a data warehouse [1]. The analysis process becomes more difficult because of the costs involved in hiring experienced staff and the privacy issues arising from the use of external consultants [2]. Peter Coad's transaction pattern [3] is a higher-level description of a generic business process (or a template) that has helped in the analysis and design of a wide range of business domains. Thesis of this paper is that the mapping of the operational databases on the transaction pattern facilitates the derivation of initial data warehouse structure. During the mapping process, the relationships, roles and attributes of the players defined by the transaction pattern help us in identifying the instances of the pattern in the database. Through these instances, we can then derive the initial data warehouse structure i.e. the attributes of the fact and dimension table(s). The data warehouse structure thus derived reduces the need for an extensive information analysis of the needs of the user and the dependency on the experienced personnel for data warehouse development**.** This paper proposes a three-step derivation methodology that is illustrated using a case study of an organization's operational database.

Keywords: transaction patterns, mapping, operational database, data warehouse structure, transaction, transaction players, fact table, dimension table.

WIT Transactions on Information and Communication Technologies, Vol 36, © 2006 WIT Press
www.witpress.com, ISSN 1743-3517 (on-line)
doi:10.2495/IS060371

1 Introduction

A data warehouse is a subject-oriented, integrated collection of data specifically designed for organizational decision-making. Data in business organization originates in the form of transactions such as orders, invoices, issues and receipts and are stored in operational databases. The online transaction systems or operational databases generate vast quantities of products, customer and sales data that is not well suited for decision-support queries. The results of other studies [4, 5, 6, 7] show that a large number of organizations that have no experience of systems for decision support have to rely on an experienced development team for their data warehouse development task.

Han Schouten [1] found that typically, an involved process of analysis must precede the actual design phase of a data warehouse. According to Frendi *et al.* [8], there is a lack of guidance for the requirements engineering part of the data warehouse's development process. In addition, Jones and Song [9] research shows data warehousing projects are complex, large, and difficult to design. Pereira and Becker [2] found that for a number of reasons such as availability of experienced staff, the costs involved in hiring experienced staff and privacy, it is not always possible to count on external development teams or consultants. Other studies [2, 6] show inexperience as one of the main causes of failure in data warehouse projects.

A transaction pattern [3] defines business transactions in terms of commonly recurring arrangement of entities called *players* (further explained in Section 3). *Transaction patterns* have helped in the analysis and design of a wide range of business domains from sales, purchase, and production of physical products to travel, insurance and other professional services. Our methodology involves mapping entities of the operational database to the *player*s of the transaction pattern. During this process the relationships, roles and attributes of the *player*s defined by the *transaction pattern* help us in identifying the instances of the pattern in the database. Through these instances, we can then derive the initial data warehouse structure i.e. the attributes of the fact and dimension table(s).

In our opinion, the initial data warehouse structure thus derived is not domain specific but covers a wide range of business process as the transaction pattern used in this research is a tested template and has been successfully applied across a range of business processes as given in Coad *et al.* [3]. Using transaction pattern as a guide, staff with lesser experience is able to derive the initial data warehouse structure.

The work presented in this paper is motivated by a novel idea of maintaining the mapping of the operational database on the *transaction pattern* and using this mapping to facilitate the development of business software, in our case the initial data warehouse structure. This mapping has already been useful in the flexible development, maintenance, implementation and extension of a local proprietary ERP-like product through its various generations.

This work is part of an on-going research effort to use the higher-level abstraction of the *transaction pattern* as a guide for the auto generation of user

interfaces, reports and decision support systems. Existing work on such auto generation typically relies only on the relationships defined by the schema of the operational database. Our idea is to get additional information (in the form of a mapping) by supplementing the database schema and relationships with the roles, attributes and relationships defined by the *transaction pattern*.

The rest of the paper is organized as follows: Section 2 describes related work. Transaction pattern is briefly described in section 3. Derivation of the initial data warehouse structure is demonstrated with the help of a case study in section 4. Section 5 presents the conclusion and recommendations for future work.

2 Related work

Developing data warehouses has become a popular but exceedingly demanding and costly activity in information systems development and management [10]. The two most popular data modeling techniques for data warehousing are Entity-Relational modeling and Dimensional modeling. In addition to these techniques business process models, object modeling approach and patterns are also used.

The Entity-Relational modeling follows the standard OLTP database design process, starting with a conceptual entity-relationship (ER) design, translating the ER schema into a relational schema, and then normalizing the relational schema.

A dimensional model is composed of two types of tables; a fact table and a dimension table. A dimensional model is a specific discipline for modeling data that is an alternative to entity relationship modeling. Like an entity relationship model, a dimensional model reflects a data structure. Aggregation is an essential feature of data warehouses. Since the traditional conceptual data models (like ER diagrams) lack constructs for expressing aggregation, researchers have proposed extensions to ER diagrams for modeling aggregation over different dimensions. Some of these extensions are *ME/R, SERM, DFM, StarER and EVER*.

ME/R is an extension of Entity Relationship Model especially for multidimensional Modelling [11]. This approach uses an algorithm to automate the conceptual schema development and evaluation of data warehouse by using numeric fields and relationships between entities as the basis to create ME/R, Entity Relational model, and schemas. However, ME/R schema does not store semantic information like data source, data description, data type, constraints, etc. StarER model [12] combines the star structure, which is dominant in data warehouses, with the semantically rich constructs of the ER model. However, while analyzing the user requirements, starER model reveals a set of new modeling concepts that need to be handled with well-known models and schema resulting in the extension with new constructs. EVER [13] is an entity-based model language based on an event concept that is suited for multi-dimensional modeling because measurement data often represents events in multi-dimensional databases. However, this language does not support specification of filtering predicates and aggregation functions. In SERM [14], conceptual schemes of operational sources are used to derive the initial data warehouse structures by using a data modeling technique, where the data models are based

on the principle of dependency showing the visualization of the order of dependencies between data objects

Patterns are also used in data modeling for data warehouse. The warehouse pattern [15] is based on a time interval concatenated with the primary keys of the master entities, such as Customer, Product, and Salesperson. Jones and Song [9] recommend Dimensional Design Patterns (DDPs) and their applications in the designing of dimensional models. This is done by describing a meta-model of the DDPs and showing their integration into Kimball's dimensional modeling design process so that they can be applied to design problems using a known practice. DDPs have successfully shown reduction in time in generating the dimensional model; however, it is yet to be implemented in working/business environment.

The proposed methodology combines the dimensional model technique with patterns.

3 Transaction pattern

Transaction patterns define business transactions in terms of commonly recurring arrangement of entities called *players* [3]. These *players* are *transaction, transaction line-item, item, place, subsequent transaction, subsequent transaction line-item, participant* and *actor* as shown in Figure 1. Examples of *transactions* include purchase order, sales invoice, doctor's prescription, travel itinerary, etc. *Transaction line-item* is the detail specifying one or more items or products related to a transaction. *Item* is what the organization deals with. An *item* instance may either be a discrete product or service, which is kept track of individually or in an inventory. In either case, each *item* instance must be at a *place*. *Place* is the "area" where transaction takes place. It can be in an office, storage location, or department. *Subsequent transaction* is the next transaction in the workflow. *Subsequent transaction line-item* is the detail of the *subsequent transaction*.

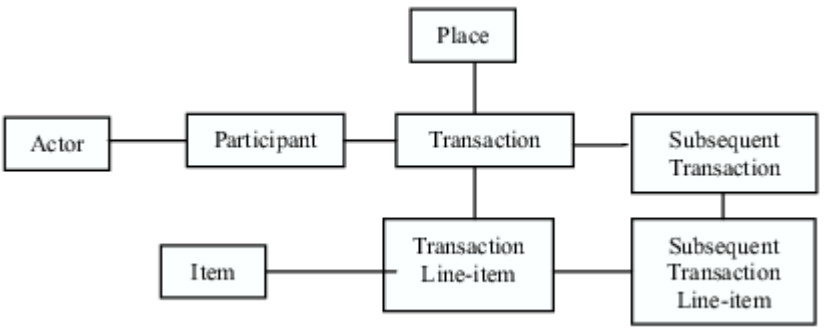

Figure 1: Transaction pattern and its *player*s [3].

An *actor* is an organization or person that plays a specific role in a transaction. An *actor* may play multiple roles in different transactions. Customer, transporter, booking clerk, or employees are *participants* that represent roles played by actors.

The dependency among the *player*s of the transaction pattern is a faceted classification that makes identification of a *player* easier given an earlier related identified *player*. That is to say, once a *transaction* is identified, the process of identifying *player*s related to it then becomes easier.

4 The derivation process

Our objective is to derive the initial data warehouse structure by mapping an operational database on to the transaction pattern. The operational database used in this case study is that of a live organization. The operational database consists of some 298 tables related to modules such as sales, accounting, purchase, inventory etc. The relevant entities (sales module) of the operational databases are mapped table-by-table on to the transaction pattern. In order to meet our objective, a three-step methodology is proposed.

The steps are as follows:

1) Identify the transaction *players* from the operational database
2) Identify the fact table from transaction *players*
3) Identify the dimensions from transaction *players*.

4.1 Identify the transaction *players* from operational database

Business activities for the operational database under study are Purchase, Sales and Inventory. In this paper, we illustrate the derivation of the data warehouse structure using the sales module. A typical sales module consists of placement of a sales order followed by the delivery of goods and its corresponding invoice. If the customer does not like the delivered goods, he may return the goods. Figure 2 shows the sale module entities from the operational database.

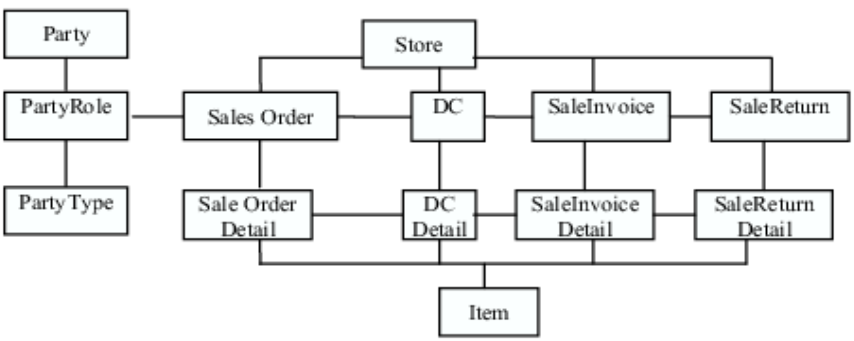

Figure 2: Sales workflow entities in the database.

In the first step, transaction pattern is used as a "stencil" to identify the instances of *players* in the database. Mapping between part of sales module from the operational database (Figure 2) and transaction pattern (Figure1 1) is shown in Figure 3.

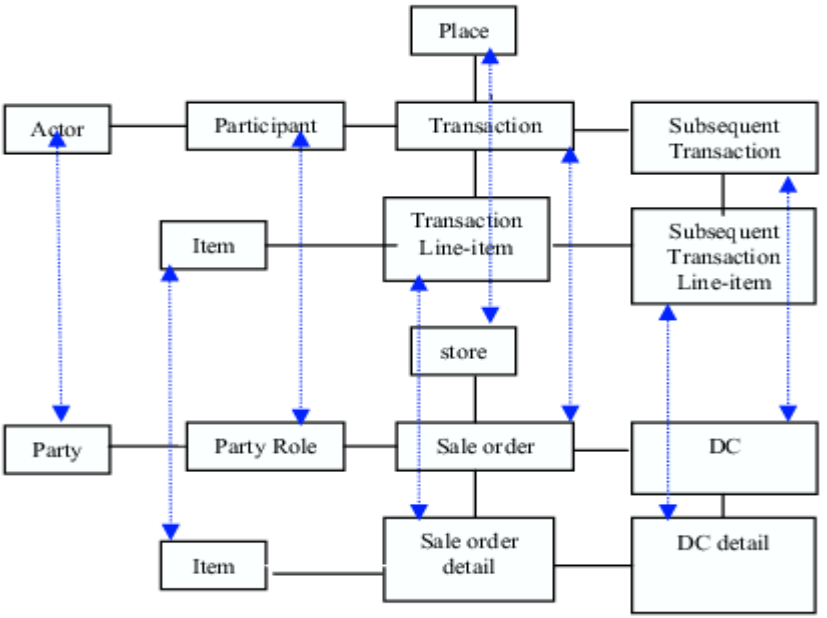

Figure 3: Mapping of a segment of sales module on transaction pattern.

According to the mapping, *Transaction line-item* is Sales Order Detail. *Subsequent transaction* for Sales Order is Delivery Challan (DC). *Subsequent transaction line-item* for DC is DC details. *Participant* for Sales *transaction* is PartyRole. The database uses a party model where Customer, Supplier, Employees, etc are all various roles of a *party (actor)* distinguished by the party type. *Item* is linked to all the *transaction line-items*. Sales office and Warehouse/Store are instances of *place* and are linked to the *transactions*.

This "stencil" is then moved so that DC becomes the *transaction*, and Sales Invoice becomes the next *subsequent transaction*, resulting in the complete mapping of sales module with recurring patterns *players* as shown in table 1.

Table 1: Sales module mapping on transaction pattern.

Transaction	Transaction Line-item	Subsequent Transaction	Subsequent Line Transaction	Item	Actor	Participant	Place
Sales Order	Sales Order Detail	DC	DC detail	Item	Party	Customer Employee Sales clerk	• Sales office • Store
DC	DC Detail	Sales Invoice	Sales Invoice Detail + Adjustments	Item	Party	Transporter Customer	• Sales office • Store • Department
Sales Invoice	Sales Invoice Detail	Sales Return	Sales Return Detail	Item	Party	Customer	• Sales office • Department
Sales Return	Sales Return Detail	Sales Return Invoice	Sales return adjustments	Item	Party	Customer	• Sales office • Store

4.2 Identify fact table from transaction *players*

An analyst's view of the enterprise universe should be multi-dimensional in nature. This multi-dimensional view is possible if fact table of a transaction contains attributes of when, whom, where, how much. For Sale Order, the fact table requires, details related to sales such as the time of sales when (*transaction line-item*), to whom *(participant*) it was made, how much sale was made (*item, transaction line-item*) and where (*place*) it was made. The attributes of fact table are **keys** of *transaction, transaction line-item, place, participant* and *item* as shown in third column of Table 2.

Table 2: Fact table attributes from transaction players.

Transaction Player	Mapped DB Entity	Key of entity
Transaction	Sales Order	Sales Order id
Transaction Line-item	Sales Invoice Detail	Sales Order Detail id
Item	Item	Item Id
Participant	Party, PartyType	Party Id, PartyType Id
Place	Store	StoreId

In addition to attributes in table 2, measurable attributes and time attributes are also required in the fact table. Measurable attributes are numeric attributes from *transaction line-item*, in this case quantity sold, cost price and sale price. Fact table de-normalizes relationships between *transaction* and *transaction line item*.

4.3 Identify dimensions from transaction *players*

Facts are bounded by dimensions. This means that dimension tables should support the facts. Facts are analyzed through each of transaction *players* that are *Transaction line-Item, Place, Participants and Items* (Table 3).

Table 3: Transaction *players* leading to dimension table.

Transaction Player	Dimension Name	Attributes from database
Transaction line-item	Sale details such as Sale date	Sale order date, year, quarter, month, date, day
Place	Store	Store Id, Store type
Participant	Party	Party Id, PartyType Id, Address, City/Town, Country
Item	Item	Item id, Item name, description, RatePerUnit, ItemCatId, UOMId

The dimension attributes for DC, Sales invoice and Sales Return can be similarly derived using transaction *players* as shown in table 3. In figure 4, we

show the transformation of the identified data warehouse structure into the star scheme.

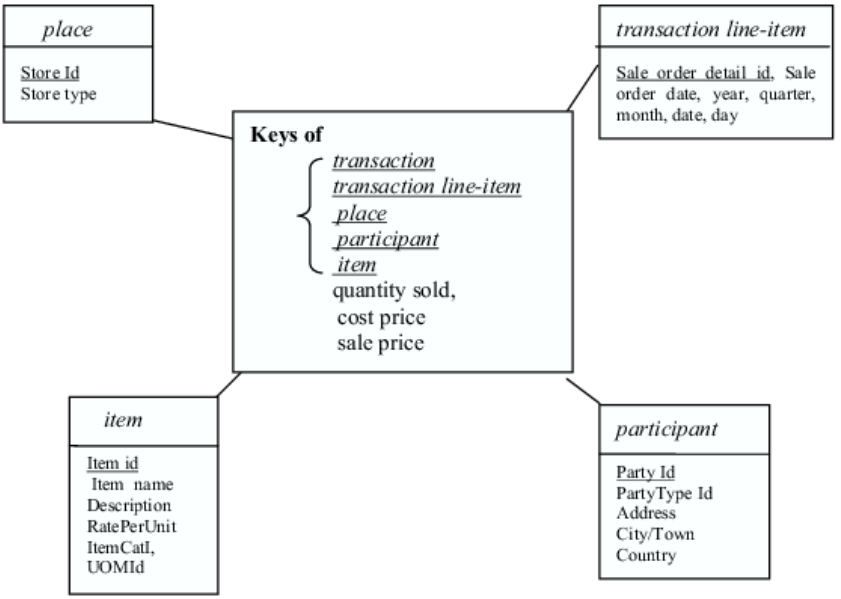

Figure 4: Star schema for sales module.

The fact table in the star scheme will contain keys of all *transaction players* as well as the numeric attributes from *transaction*. For sales module under study, it is Sale Order Id (*transaction)*, Sales Order Detail id *(transaction line-item*), Item Id (*item*), Party Id (*participant*) and Store Id *(place*).

The dimension table will be one each for *transaction players, transaction line-item*, *item*, *participant* and *place*. For each of the dimension, separate table is build. Dimension tables are Store (*place*), Sales Order Detail id *(transaction line-item*), Item (*item*) and Party (*participant*).

5 Conclusion and future work

In this paper, we presented a new approach to derive initial data warehouse structure by mapping a segment of operational database on transaction pattern. The efficiency of our approach was illustrated with the help of a case study. The methodology proposed in this paper eases the process of deriving the initial data warehouse structure. The entities of the operational database are mapped to the *players* of the *transaction pattern*; the relationships and attributes of the *players* defined by the transaction pattern help us in identifying the instances of the pattern in the database and thus developing the data warehouse structure.

Derivation process requires an initial understanding of the *transaction pattern* and the typical attributes and methods of the *players* before the mapping can proceed. It also requires the schema of the operational database and the role of the entities. Currently we have applied this approach to typical modules of business organizations such as sale and purchase module. Future studies include mapping the transaction patterns on businesses such as insurance companies, travel agencies, professional services such as health clinics or consulting services.

Research is underway to automate the derivation of data warehouse structure and automatically generate the scripts for populating the fact and dimension table of the data warehouse. If the mapping is maintained as a computer readable meta-description of the application, then we can make a system, which follows the proposed methodology, recognizes the key players of transaction patterns from the operational database, and then derives the data warehouse structure for any business domain.

References

[1] Han Schouten. Analysis and design of data warehouses. Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'99). Heidelberg, Germany, 14-15 June, 1999.

[2] Walter Pereira and Karin Becker. A Methodology Targeted at the Insertion of Data Warehouse Technology in Corporations. XV Simpósio Brasileiro de Banco de Dados, 2-4 Outubro 2000, João Pessoa, Paraíba, Brasil, pages 316-330

[3] Peter Coad, David North and Mark Mayfield . Object Models: Strategies, Patterns, and Applications. Prentice-Hall ECS Professional. Second Edition. 2000, Page 434

[4] Poe, V. Building a Data Warehouse for Decision Support, Prentice Hall, New Jersey, 1996.

[5] Dyché, Jill. Scoping Your Data Mart Implementation, DBMS. Aug. 1998.

[6] Gardner, Stephen R. Building the Data Warehouse. Communications of the ACM, 41 (9): 52-60. Sept; 1998.

[7] Kimball, Ralph; Reeves, Laura; Ross, Margy & Thornthwaite, Warren**.** The Data warehouse lifecycle toolkit: expert methods for designing, developing, and deploying data warehouses**.** New York, John Wiley & Sons, 1998.

[8] Mohamed Frendi, Camille Salinesi. Requirements Engineering for Data Warehousing. REFSQ'03. Ninth International Workshop on Requirements Engineering: Foundation for Software Quality. Klagenfurt/Velden, Austria.

[9] Mary Elizabeth "M.E." Jones and Il-Yeol Song. Dimensional Modeling: Identifying, Classifying & Applying Patterns. DOLAP'05, November 4–5, 2005, Bremen, Germany. Copyright 2005 ACM 1-59593-162-7/05/0011.

[10] Sen, Aru & Jacob, Varghese S. Industrial-Strength Data Warehousing. Communications of the ACM, 41 (9): 29-31. Sept; 1998.
[11] Cassandra Phipps, Karen C. Davis. Automating Data Warehouse Conceptual Schema Design and Evaluation. Design and Management of Data Warehouses 2002, Proceedings of the 4th Intl. Workshop DMDW'2002, Toronto, Canada: 23-32
[12] Nectaria Tryfona, Frank Busborg, and Jens G. Borch ChriStiansen. starER: A Conceptual Model for Data Warehouse Design. DOLAP '99 11/00 Kansas City, USA © 1999 ACM ISBN 1-58113-220-4/99/11
[13] Lars Baekgaard. Event-Entity-Relationship Modeling In Data Warehouse
[14] Environments. DOLAP '99 11/00 Kansas City Mo, USA © 1999 ACM
[15] ISBN 1-58113-220-4/99/11
[16] Michael Boehnlein & Achim Ulbrich-vom Ende. Deriving Initial Data Warehouse Structures from the Conceptual Data Models of the Underlying Operational Information Systems. Copyright ACM 1999 1-58113-220-4/99/11.
[17] Dinesh Batra. Conceptual Data Modeling Patterns: Representation and Validation. Journal of Database Management, 16(2), 84-106, April-June 2005 Page 85.