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

A. Khalfe & S. I. Hyder
College of Computer 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.
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 rely 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 players of the transaction pattern. During this 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).

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 players [3].](image-url)

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 players 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 players 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.

![Sales workflow entities in the database.](image)

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 (Figure 1) is shown in Figure 3.
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.

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Transaction Line-item</th>
<th>Subsequent Transaction</th>
<th>Subsequent Line Transaction</th>
<th>Item</th>
<th>Actor</th>
<th>Participant</th>
<th>Place</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales Order</td>
<td>Sales Order Detail</td>
<td>DC</td>
<td>DC detail</td>
<td>Item</td>
<td>Party</td>
<td>Customer</td>
<td>Sales office</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Employee</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sales clerk</td>
<td>Store</td>
</tr>
<tr>
<td>DC</td>
<td>DC Detail</td>
<td>Sales Invoice</td>
<td>Sales Invoice Detail + Adjustments</td>
<td>Item</td>
<td>Party</td>
<td>Customer</td>
<td>Sales office</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Transporter</td>
<td>Store</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Department</td>
</tr>
<tr>
<td>Sales Invoice</td>
<td>Sales Invoice Detail</td>
<td>Sales Return</td>
<td>Sales Return Detail</td>
<td>Item</td>
<td>Party</td>
<td>Customer</td>
<td>Sales office</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Department</td>
</tr>
<tr>
<td>Sales Return</td>
<td>Sales Return Detail</td>
<td>Sales Return Invoice</td>
<td>Sales return adjustments</td>
<td>Item</td>
<td>Party</td>
<td>Customer</td>
<td>Sales office</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Store</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Transaction Player</th>
<th>Mapped DB Entity</th>
<th>Key of entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction</td>
<td>Sales Order</td>
<td>Sales Order id</td>
</tr>
<tr>
<td>Transaction Line-item</td>
<td>Sales Invoice Detail</td>
<td>Sales Order Detail id</td>
</tr>
<tr>
<td>Item</td>
<td>Item</td>
<td>Item Id</td>
</tr>
<tr>
<td>Participant</td>
<td>Party, PartyType</td>
<td>Party Id, PartyType Id</td>
</tr>
<tr>
<td>Place</td>
<td>Store</td>
<td>StoreId</td>
</tr>
</tbody>
</table>

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

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.

![Diagram of Star Schema](image)

**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**


