Available online at www.sciencedirect.com

ScienceDirect

Data & Knowledge Engineering 62 (2007) 483–503

DATA & KNOWLEDGE ENGINEERING

www.elsevier.com/locate/datak

One-to-many data transformations through data mappers

Paulo Carreira [a,b,*], Helena Galhardas [a], Antónia Lopes [b], João Pereira [a]

[a] INESC-ID and Technical University of Lisbon, Avenida Prof. Cavaco Silva, Tagus Park, 2780-990, Porto-Salvo, Portugal
[b] Faculty of Sciences, University of Lisbon, C6 – Piso 3, 1749-016 Lisboa, Portugal

Received 12 August 2006; accepted 12 August 2006
Available online 2 October 2006

Abstract

The optimization capabilities of RDBMSs make them attractive for executing data transformations. However, despite the fact that many useful data transformations can be expressed as relational queries, an important class of data transformations that produce several output tuples for a single input tuple cannot be expressed in that way.

To overcome this limitation, we propose to extend Relational Algebra with a new operator named *data mapper*. In this paper, we formalize the data mapper operator and investigate some of its properties. We then propose a set of algebraic rewriting rules that enable the logical optimization of expressions with mappers and prove their correctness. Finally, we experimentally study the proposed optimizations and identify the key factors that influence the optimization gains.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Data warehousing; ETL; Data mapper operator; Query optimization; Relational algebra

1. Introduction

The setup of modern information systems comprises a number of activities that rely, to a great extent, in the use of data transformations [21]. Well known cases are migrations of legacy-data, Extract-Transform-Load (ETL) processes that support data warehousing, cleansing of data and integration of data from multiple sources. This situation leads to the development of data transformation programs that must move data instances from a fixed source schema into a fixed target schema.

One natural way of expressing data transformations is to use a declarative query language and specifying the data transformations as queries (or views) over the source data. Because of the broad adoption of RDBMSs, such language is often SQL, a language based on Relational Algebra (RA). Unfortunately, due

* Corresponding author. Address: INESC-ID and Technical University of Lisbon, Avenida Prof. Cavaco Silva, Tagus Park, 2780-990, Porto-Salvo, Portugal.

E-mail addresses: paulo.carreira@tagus.ist.utl.pt (P. Carreira), hig@inesc-id.pt (H. Galhardas), mal@di.fc.ul.pt (A. Lopes), joao@inesc-id.pt (J. Pereira).

0169-023X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2006.08.011

to its limited expressive power [3], RA alone cannot be used to specify many important data transformations [22].

To overcome these limitations, several alternatives have been adopted: (i) the implementation of data transformation programs using a programming language, such as C or Java, (ii) the use of an RDBMS proprietary language like Oracle PL/SQL; or (iii) the development of data transformation scripts using a commercial ETL tool. However, transformations expressed in this way are often difficult to maintain, and more importantly, there is little room for optimization [6]. We remark that only recently an optimization technique for ETL processes was proposed [33].

The normalization theory underlying the relational model imposes the organization of data according to several relations in order to avoid redundancy and inconsistency of information. In Codd's original model, new relations are derived from the database by selecting, joining and unioning relations. Despite the fact that RA expressions denote transformations among relations, the notion that presided the design of RA (as noted by [3]) was that of retrieving data. This notion, however, is insufficient for reconciling the substantial differences in the representation of data that occur between fixed source and target schemas [24].

One such difference occurs when the source data is an aggregation of the target data. For example, source data may consist of salaries aggregated by year, while the target consists of salaries aggregated by month. The data transformation that has to take place needs to produce several tuples in the target relation to represent each tuple of the source relation. We designate these data transformations as *one-to-many data mappings*. As we will demonstrate in Section 3.3, this class of data transformations cannot be expressed by standard RA, even if we use the generalized projection operator [31].

Our experience with the Ajax [15] data cleaning tool and with the Data Fusion [5] legacy-data migration tool has shown that in the context of data transformation, there is a considerable number of data transformations that require one-to-many mappings. In fact, as recognized in [13], an important class of data transformations requires the inverse operation of the SQL group by/aggregates primitive so that, for each input tuple, they produce several output tuples.

In this paper, we propose an extension to RA to represent one-to-many data transformations. This extension is achieved through a new operator that, like the generalized projection operator, relies on the use of arbitrary, external functions.

There are two main reasons why we chose to extend RA. First, even though RA is not expressive enough to capture the semantics of one-to-many mappings, we want to make use of the available expressiveness for the remaining data transformations. Second, we intend to take advantage of the optimization strategies that are implemented by relational database engines [7]. Our decision to adopt database technology as the basis for data transformation is not new. Several RDBMSs, like Microsoft SQL Server, already include additional software packages specifically designed for ETL tasks. However, to the best of our knowledge, none of these extensions is grounded in database theory. Therefore, the capabilities of relational engines, for example, in terms of optimization opportunities are not fully exploited.

In the remainder of this section, we first present a motivating example to illustrate the usefulness of one-to-many data transformations. Then, in Section 1.2, we highlight the contributions of this paper.

1.1. Motivating example

Here, we present a simple example of a data transformation that is a one-to-many mapping. It is based on a real-world data migration scenario, that was intentionally simplified for illustration purposes.

Example 1.1. Consider the source relation `LOANS[ACCT, AM]` (represented in Fig. 1) that stores the details of loans requested per account. Suppose `LOANS` data must be transformed into `PAYMENTS[ACCTNO, AMOUNT, SEQNO]`, the target relation, according to the following requirements:

(1) In the target relation, all the account numbers are left padded with zeroes. Thus, the attribute `ACCTNO` is obtained by (left) concatenating zeroes to the value of `ACCT`.