ONE-TO-MANY DATA TRANSFORMATIONS
As Relational Operations

Paulo Carreira
Faculty of Sciences, University of Lisbon, C6 - Piso 3, 1749-016 Lisboa, Portugal
paulo.carreira@xldb.di.fc.ul.pt

Keywords: Data Warehousing, Data Cleaning, Data Integration, ETL, Query optimization

Abstract: Transforming data is a fundamental operation in data management activities like *data integration*, *legacy data migration*, *data cleaning*, and *extract-transform-load processes* for data warehousing. Since data often resides on relational databases, data transformations are often implemented as relational queries that aim at leveraging the optimization capabilities of most RDBMSs.
However, due to the limited expressive power of Relational Algebra, several important classes of data transformations cannot be specified as SQL queries. In particular, SQL is unable to express data transformations that require the dynamic creation of several tuples for each tuple of the source relation.
This paper proposes to address this class of data transformations, common in data management activities, by extending Relational Algebra with a new relational operator named *data mapper.* A starting contribution of this work consists of studying the formal aspects of the *mapper* operator focusing on its formal semantics and expressiveness. A further contribution consists of supporting a cost-based optimization of data transformations expressions combining mappers with standard relational operators. To that end, a set of algebraic rewriting rules and different physical execution algorithms are being developed.

1 INTRODUCTION

Data transformation is a fundamental step of data management activities such as integration, cleaning, migration and warehousing of data. In these activities, data represented by a fixed source schema must be transformed into a fixed target data schema.

A frequent problem in this context is the existence of *data heterogeneities*, i.e., the use of different representations of the same data in source and target schemas (Rahm and Do, 2000). For example: the use of different units of measurement or the use of different representations for compound data (e.g. multiple attributes representing day, month and year information *vs* a single date attribute) ocur frequently. Another important source of data heterogeneities is the representation of data according to different aggregation levels (e.g. hourly *vs* daily). When the source data represents an aggregation of the target data (e.g., yearly aggregated data in the source and monthly data in the target), the data transformation that has to take place needs to generate several tuples for each source tuple. Let us henceforth designate this class of transformations as *one-to-many* data transformations.

Consider a relation `LOANS[ACCT, AM]` (represented in Figure 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 attribute `ACCT`.
2. The target system does not support pay-

Relation LOANS

ACCT	AM
12	20.00
3456	140.00
901	250.00

Relation PAYMENTS

ACCTNO	AMOUNT	SEQNO
0012	20.00	1
3456	100.00	1
3456	40.00	2
0901	100.00	1
0901	100.00	2
0901	50.00	3

Figure 1: Illustration of an unbounded data-transformation. *(a)* The source relation LOANS on the left, and *(b)* the target relation PAYMENTS on the right.

ment amounts greater than 100. The attribute AMOUNT is obtained by breaking down the value of attribute AM into multiple installments with a maximum value of 100, in such a way that the sum of amounts for the same ACCTNO is equal to the source amount for the same account. Furthermore, the target field SEQNO is a sequence number for the installment. This sequence number starts at 1 for each sequence of installments of a given account.

The implementation of data transformations similar to those requested for producing the target relation PAYMENTS is challenging, since solutions to the problem involve the dynamic creation of tuples based on the value of attribute AM.

The remainder of the paper is organized as follows: Next, we present the problem statement and enumerate the main contributions of the paper. The research field of data transformations is reviewed in Section 2. Section 3 briefly presents the mapper operator, introducing the logical and physical optimization issues. The current status of the paper work is detailed in Section 4 and Section 5 concludes.

1.1 Problem statement

To minimize development effort and maximize performance, data transformations must be written in a language that is *declarative*, *optimizable*, *expressive*. Data transformations are often expressed as Relational algebra (RA) expressions, which is a language that meets the two former requirements. In fact, many usefull data transformations can be naturally expressed as RA queries. However, due to the limitations in the expressive power of RA, relational queries are insufficient for expressing many interesting data transformations (Lakshmanan et al., 1996; Miller, 1998). In particular, RA is not capable of deriving new items (Paredaens, 1978) and thus, cannot represent the class of one-to-many data transformations.

Currently, to develop one-to-many data transformations, one has to resort, either to a general purpose programming language, to some flavor of proprietary scripting of an ETL tool, or to a stored procedure written in the DBMS proprietary programming language. In any case, besides the inadequacy of these solutions to express one-to-many data transformations concisely, there is little possibility of leveraging the dynamic optimization capabilities of the DBMS.

1.2 Contribution

This paper proposes a new operator named *data mapper* which extends RA for expressing one-to-many data transformations.

Since data transformations are often performed by RDBMSs, or by tools and languages that are also to based on RA to various extents, the new operator is a general solution to express one-to-many data transformations in these systems. In particular, by incorporating the mapper operator, RDBMSs will be capable of efficiently handling a new class of data transformations, enabling their use in a greater variety of data management activities that require data transformations.

An advantage of adressing the problem of one-to-many data transformations through an operator is that it can be embedded in expressions involving standard relational operators and also be logically and physically optimized. To this end, we envision a cost-based optimization of data transformations expressed as a combination of standard relational operators and mappers. We propose *(i)* to formalize the mapper operator, *(ii)* to study the formal properties of the mapper operator focusing on its expressiveness and algebraic properties, *(iii)* to develop alternative physical execution algorithms, and *(iv)* to adapt current cost-based query optimization techniques to handle mappers.

2 DATA TRANSFORMATIONS

Data transformation is an old problem and the idea of using a declarative language to specify such transformations has been proposed back in the 1970's with two prototypes, Convert (Shu et al., 1975) and Express (Shu et al., 1977), both aiming at data conversion.

To support the growing span of applications of RDBMSs, several extensions to RA have been proposed since its inception, mainly in the form of new operators. Applications requiring data transformations bring a new requirement to RA as their focus is no more limited to the initial idea of selecting information, but also involves the production of new data items (Paredaens, 1978).

In the context of data cleaning, Potter's Wheel fold (Raman and Hellerstein, 2001) operator and Ajax (Galhardas et al., 2000) map operator were proposed for expressing one-to-many data transformations. The Data Fusion tool (Carreira and Galhardas, 2004) implements one-to-many data transformations in the context of legacy-data migrations. None of these, however, proposes an extension of the relational algebra or addresses logical and physical optimization issues.

Data transformations are also required in ETL processes. To the best of our knowledge, in most ETL tools, to express one-to-many data-transformations, the user has to resort to some form of ad-hoc scripting. Furthermore, the optimization of ETL data transformations is a recent effort (Simitsis et al., 2005).

When performing data integration, data has to be transformed from the data sources to the integrated database or vice-versa. TSIMMIS MSL (Papakonstantinou et al., 1996) and Squirrel ISL (Zhou et al., 1996) are data integration languages whose main goal is to fusion data from several sources. These languages, like others for restructuring semi-structured data, e.g, YAT (Cluet et al., 1998), and TransScm (Milo and Zhoar, 1998), have their expressiveness restricted to avoid potentially dangerous specifications (that may result in diverging computations). As a result, they cannot express the class of one-to-many data transformations.

3 THE MAPPER OPERATOR

The mapper operator can be formalized as a unary operator μ_F that takes a relation instance of the source relation schema as input and produces a relation instance of the target relation schema as output. The operator is parameterized by a set F of special functions, which we designate as *mapper functions*. The intuition is that each mapper function f_{A_i} expresses a part of the envisaged data transformation, focused on a non-empty set A_i of attributes of the target schema. A key insight is that, when applied to a tuple, a mapper function can produce a set of values in the domain of its target attributes $Dom(A_i)$, rather than a single value.

Consider relation schemas X and Y. Furthermore, let $Y = A_1 \cdot ... \cdot A_k$, where each A_i is a set of schema attributes. Given a tuple u of a source relation $s(X)$, the expression $\mu_F(\{u\})$ denotes the tuples t in $Dom(Y)$ such that, for every set of attributes A_i, associate the values given by $f_{A_i}(s)$. Further details can be found in (Carreira et al., 2005a). The mapper operator is formally defined as follows: Given a set of mapper functions $F = \{f_{A_1}, ..., f_{A_k}\}$, the *mapper* of s with respect to F, denoted by $\mu_F(s)$, is the relation instance of the target relation schema defined by

$$\mu_F(s) \overset{\text{def}}{=} \{t \in Dom(Y) \mid \exists u \in s \text{ s.t. } t[A_i] \in f_{A_i}(u), \\ \forall 1 \leq i \leq k\}$$

The data transformation of the introductory example can be expressed by means of a mapper $\mu_{acct,amt}$, with two mapper functions. The function *acct* is the ACCT-mapper function that returns a singleton with the account number ACCT properly left padded with zeroes. The function *amt* is the [AMOUNT,SEQNO]-mapper function s.t., $amt(am)$ is given by $\{(100, i) \mid 1 \leq i \leq (am/100)\} \cup \{(am\%100, (am/100)+1) \mid am\%100 \neq 0\}$, where % represents the modulus operation. For instance, if v is the source tuple (901,250.00), the result of evaluating $amt(v)$ is the set $\{$(100,1),(100,2),(50,3)$\}$. Given a source relation s including v, the result of the expression $\mu_{acct,amt}(s)$ is another relation that contains the set of tuples $\{\langle$'0901',100,1$\rangle$, $\langle$'0901',100,2$\rangle$, $\langle$'0901',50,3$\rangle\}$.

3.1 Logical optimization

Better plans for queries involving mappers can be achieved through the systematic application of a new set of algebraic rewriting rules. One such simple algebraic rewriting rule is $\mu_F(r \bowtie s) = \mu_F(r) \bowtie \mu_F(s)$, if none of the mapper functions in F produces duplicate values.

An important property that influences the choice of a particular plan for binary operators, is the expected number of tuples of each of its sub-plans. Since mappers can generate several output tuples for each input tuple, estimating the number of output tuples of a mapper is an interesting problem. One way to approach the problem consists of estimating the *mapper fan-out* factor. If a mapper was never evaluated before, an interesting question is how to find a good initial estimate for its fan-out. We believe that in many situations the fan-out factors of mapper functions can be combined to produce an initial answer. Another interesting observation is that when mapper functions return empty sets, no output tuples are produced. Thus, the mapper in some situations may act as a filter, which turns *selectivity* into another relevant factor. The already non-trivial problem of optimizing queries with mappers can be taken one step further. Mapper functions can be expensive and due to data skewness, its cost is subject to change at query-execution time.

3.2 Physical optimization

The formal semantics of the mapper equips it with a simple iterator-based execution model as follows: For each input tuple, perform the evaluation of each mapper function and then compute the Cartesian product of the results. The output relation is obtained by unioning all the tuples so obtained.

This simple model favors the integration of our mapper operator in the query execution mechanisms of an RDBMS. However, it turns out that in the presence of *expensive functions*, like, e.g., string matching or check-digit computations, this naïve execution of the mapper operator can be very inefficient.

The total cost of evaluating a mapper can be minimized by avoiding superfluous function evaluations. First, columns often have duplicate values. This suggests the use of caching techniques. In the presence of potentially many functions and tables with multi-million tuples, the choice of the functions is an optimization problem in itself. Second, some functions return empty sets. When an empty set is found, no output tuples are produced for a given input tuple. Thus, there is no need to evaluate the remaining functions. This observation suggests an interesting strategy that consists of evaluating the functions that are more *selective* first.

4 CURRENT STATUS

Defining a new operator is a significant research effort as it requires both theoretical and practical insight. In such effort, two issues need to be addressed forefront. The usefulness of the operator needs to be validated and the class of problems being solved has to be formally defined.

To address the first issue, we pursued a commercial venture that resulted in the inclusion of native support for one-to-many data transformations in a commercial tool (Carreira and Galhardas, 2004). The tool is being used in several real-world legacy-data migration projects that corroborate the need for supporting one-to-many data transformations.

Up to this moment, we have been able to put forward a formal semantics for the new operator that enabled us to perform a formal study of the expressiveness of the operator. We developed the formal demonstration that the *mapper-extended RA* (MRA) is strictly more expressive than standard RA.

A formal definition of the class of one-to-many data transformations is underway. We conjecture that two sub-classes of one-to-many data transformations exist: One comprising data transformations expressible through RA and another comprising those expressible only through MRA.

A set of algebraic rewriting rules for generating logical query plans involving mappers and some standard relational operators have been developed together with their formal proofs of correctness (Carreira et al., 2005a). A first set of rewriting rules for expressions involving mappers and joins has also emerged. Currently, a set of experiments is being conducted to determine the factors that influence the effectiveness of the proposed rewritings (Carreira et al., 2005b). Prototypical implementations for physical mapper operator algorithms are being developed in Java using the XXL framework (van den Bercken et al., 2000). These algorithms adapt ideas of *memoization* and *hybrid hashing* proposed by (Hellerstein and Naughton, 1996) to multiple functions.

5 CONCLUSIONS

In this work, we address the problem of specifying one-to-many data transformations that are frequently required in data integration, data cleaning, legacy-data migration, and ETL scenarios. Since one-to-many data transformations are not expressible through standard RA queries, we proposed the mapper operator. The new operator allows to naturally express one-to-many data transformations, while extending the expressive power of RA at the same time.

Up to now some operators have been proposed for addressing the problem of expressing one to many data-transformations (Cunningham et al., 2004; Galhardas et al., 2001; Raman and Hellerstein, 2001; Amer-Yahia and Cluet, 2004). Although these operators show similarities with mappers, most of them are only capable of expressing a subset of one-to-many transformations.

As data often resides in RDBMSs, data transformations specified as relational expressions can take direct advantage of their optimization capabilities. In this trend, several RDBMSs, like e.g., Microsoft SQL Server, already include additional software packages specific for ETL tasks. However, as far as we know, none of these extensions is supported by the corresponding theoretical background in terms of existing database theory. Therefore, the capabilities of relational engines, in terms of optimization opportunities are not fully exploited in activities involving data transformations, like ETL or data-cleaning.

REFERENCES

Amer-Yahia, S. and Cluet, S. (2004). A declarative approach to optimize bulk loading into databases. *ACM Transactions of Database Systems*, 29(2):233–281.

Carreira, P. and Galhardas, H. (2004). Efficient development of data migration transformations. In *ACM SIGMOD Int'l Conf. on the Managt. of Data*.

Carreira, P., Galhardas, H., Lopes, A., and Pereira, J. (2005a). Extending relational algebra to express one-to-many data transformations. In *20th Brasillian Symposium on Databases SBBD'05*.

Carreira, P., Galhardas, H., Pereira, J., and Lopes, A. (2005b). Data mapper: An operator for expressiong one-to-many data transformations. In *7th Int'l Conf. on Data Warehousing and Knowledge Discovery, DaWaK '05*, volume 3589 of *LNCS*. Springer-Verlag.

Cluet, S., Delobel, C., Siméon, J., and Smaga, K. (1998). Your mediators need data conversion! In *ACM SIGMOD Int'l Conf. on the Managt. of Data*.

Cunningham, C., Graefe, G., and Galindo-Legaria, C. A. (2004). PIVOT and UNPIVOT: Optimization and Execution Strategies in an RDBMS. In *Proceedings of the International Conference on Very Large Data Bases (VLDB'04)*, pages 998–1009. Morgan Kaufmann.

Galhardas, H., Florescu, D., Shasha, D., and Simon, E. (2000). Ajax: An extensible data cleaning tool. *ACM SIGMOD Int'l Conf. on Managt. of Data*, 2(29).

Galhardas, H., Florescu, D., Shasha, D., Simon, E., and Saita, C. A. (2001). Declarative data cleaning: Language, model, and algorithms. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB'01)*.

Hellerstein, J. M. and Naughton, J. F. (1996). Query execution techniques for caching expensive methods. *ACM SIGMOD Int'l Conf. on Managt. of Data*.

Lakshmanan, L. V. S., Sadri, F., and Subramanian, I. N. (1996). SchemaSQL - A Language for Querying and Restructuring Database Systems. In *Proc. Int'l Conf. on Very Large Databases (VLDB'96)*, pages 239–250.

Miller, R. J. (1998). Using schematically heterogeneous structures. *Proc. of ACM SIGMOD Int'l Conf. on the Managt. of Data*, 2(22):189–200.

Milo, T. and Zhoar, S. (1998). Using schema matching to simplify heterogeneous data translation. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB'98)*.

Papakonstantinou, Y., Garcia-Molina, H., and Ullman, J. (1996). MedMaker: A Mediator System Based on Declarative Specifications. In *Proc. Int'l. Conf. on Data Engineering*.

Paredaens, J. (1978). On the expressive power of the relational algebra. *Information Processing Letters*, 7(2):107–111.

Rahm, E. and Do, H.-H. (2000). Data Cleaning: Problems and current approaches. *IEEE Bulletin of the Technical Comittee on Data Engineering*, 24(4).

Raman, V. and Hellerstein, J. M. (2001). Potter's Wheel: An Interactive Data Cleaning System. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB'01)*.

Shu, N. C., Housel, B. C., and Lum, V. Y. (1975). CONVERT: A High Level Translation Definition Language for Data Conversion. *Communic. of the ACM*, 18(10):557–567.

Shu, N. C., Housel, B. C., Taylor, R. W., Ghosh, S. P., and Lum, V. Y. (1977). EXPRESS: A Data EXtraction, Processing and REStructuring System. *ACM Transactions on Database Systems*, 2(2):134–174.

Simitsis, A., Vassiliadis, P., and Sellis, T. K. (2005). Optimizing etl processes in data warehouses. In *Proc. of the 21st Int'l Conf. on Data Engineering, ICDE 2005*.

van den Bercken, J., Dittrich, J. P., and Seeger, B. (2000). XXL: A prototype for a library of query processing algorithms. In *Proc. of the ACM SIGMOD Int'l Conf. on Managt. of Data*. ACM Press.

Zhou, G., Hull, R., and King, R. (1996). Generating Data Integration Mediators That Use Materialization. *Journal of Intelligent Information Systems*, 6(2/3):199–221.