A fact approach on Data Migration

An algorithm for migrating data from one database system to another

Elton Manoku, Guido Bakema
Research and Competence Group Data Architectures & Metadata Management
Academy of Communication and Information Technology
HAN University of Applied Sciences, The Netherlands
June 2006

Abstract

The problem of migrating data is present in almost every application development process, such as datawarehousing and application integration. The process of migrating data involves firstly the mapping between the structures of the source and target databases and secondly the migration of the data from the source to the target. This paper presents an algorithm to implement this process. The algorithm is based on the idea that databases store meaningful facts, rather then instances of atomic data. The fact approach simplifies the algorithm and generalizes the problem of mapping and migrating data into the problem of mapping and migrating two kinds of facts: independent facts and dependent facts. The algorithm has been implemented in a prototype tool to prove the idea and for further research.

1 Introduction

This paper presents an algorithm for covering two processes: the mapping between two database structures (source and target) and migrating data between these two databases.

The paper is organized in 3 parts:
- The first part covers the idea of looking at the Relational Model from a fact perspective.
- The second part explains the mapping process between the two structures of the source and destination database and the algorithm of moving the data from the source to the destination based on this mapping.
- The third part briefly discusses briefly the implementation of the algorithm.

2 A fact approach on the Relational Model

Although in practice data are stored in other kinds of formats as well, here we consider only relational storage. It is after all possible to map other kinds of formats of storing data into a relational database, because any kind of data can be represented in the form of tables. The fact approach assumes that data in databases are to be conceived as facts [1]. Considering data as facts simplifies the validation of the correctness of the data, because the domain experts recognize the facts behind the data easier than understand the data in the table in other ways.

A fact can be found in one or more atomic values, which are the intersections of a column or several columns and one row in a relational table. Such atomic values combined (i.e. completed) with the appropriate soft semantics represent the facts in the database.

Bringing the discussion up to the type level, this means that every relational data model can be projected onto a fact type data model. There cannot be an atomic value in a database that is not part of an elementary fact. We will consider Fully Communication Oriented Information Modeling FCO-IM [1] as the standard way of looking into the fact based modeling methods.

In this paper we will consider only elementary facts and - at the type level - elementary fact types. Figure 1 shows - in the form of an example - how relational data can be represented in a fact based model.

Student				
First Name	**Surname**	**Mentor**	**Project**	**Project Description**
Nick	Smith	DRS	DBP	Database Project
Peter	Johnson	TRS		

Add soft semantics
And get composite facts for the rows.

Fact 1: "Student Nick Smith exists and is supervised by DRS during his project named DBP with description Database Project."
Fact 2: "Student Peter Johnson exists and is supervised by TRS."

Split the composite facts to get the most elementary level.

Fact 1.1: "There is a student Nick Smith."
Fact 1.2: "Student Nick Smith is supervised by DRS."
Fact 1.3: "Student Nick Smith is assigned project DBP."
Fact 1.4: "Project DBP is about Database Project."

Fact 2.1: "There is a student Peter Johnson."
Fact 2.2: "Student Peter Johnson is supervised by TRS."

Qualify facts into fact types.

Student: There is a student **<First Name> <Surname>**.
Mentor: Student **<First Name> <Surname>** is supervised by **<Mentor>**.
Project Assignment: Student **<First Name> <Surname>** is assigned project **<Project code>**.
Project Description: Project **<Project code>** is about **<Project Description>**.

Diagramming the elementary fact model (Information Grammar).

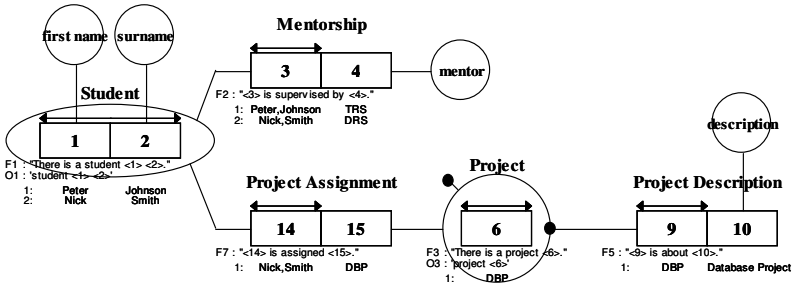


Figure 1: Relational data can be understood in terms of a fact model

From the structure of the relational data model we obtained an Information Grammar [1], in which the tables and columns are transformed into elementary fact types. The population of the tables, which consists of the atomic data values, is transformed into populations of fact

types consisting of atomic facts. When a relational data model is transformed into an elementary fact type model, despite the complexity of the relational model, we have only two kinds of fact types: *independent fact types* and *dependent fact types*; at the instance level: *independent facts* and *dependent facts*.

Independent facts:
Independent facts can stand alone in the database. In the relational model, these facts are stored in the columns under the primary key. Figure 2 shows - again in the form of an example - how an independent FCO-IM fact type is retrieved from the relational representation.

Relational representation:

Student	
First Name	**Surname**
Nick	Smith
Peter	Johnson

Soft semantics added:

Student		
There is a student	**<First Name>**	**<Surname>**
There is a student	Nick	Smith
There is a student	Peter	Johnson

FCO-IM diagram:

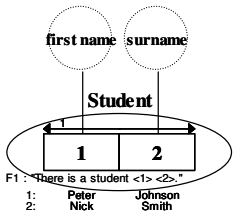


Figure 2: How an independent fact type is retrieved from the relational representation

Dependent facts:
Dependent facts cannot stand alone. They need an independent fact for their definition. In the relational model, the atomic values of these facts are the combination of atomic values that are in the primary key columns and other atomic values that are in columns that are functionally dependent on the primary key columns. If the database is not normalized, the dependent facts can be dependent on still other dependent facts.

Figure 3 shows an example of how a dependent fact type**Mentorship** is retrieved that is connected with the independent fact type **Student**: role 3 of the depenent fact type **Mentorship** is played by the independent objectified (nominalized) fact type **Student**, which means that fact tyupe **Mentorship** depends on fact type **Student**.

Relational representation:

Student		
First Name	**Surname**	**Mentor**
Nick	Smith	DRS
Peter	Johnson	TRS

Soft semantics added:

Mentorship				
Student	**<First Name>**	**<Surname>**	**is supervised by**	**<Mentor>**
Student	Nick	Smith	Is supervised by	DRS
Student	Peter	Johnson	Is supervised by	TRS

FCO-IM diagram:

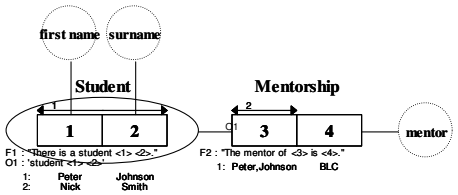


Figure 3: How an independent fact type is retrieved from the relational presentation

3 The algorithm

The aim of the algorithm is to migrate data from one database to another database that has a different structure without changing the meaning of the data or their correctness. Considering the fact approach on data, we reformulate the aim of the algorithm into: migrating the facts from one database to another database with a different structure without altering the facts. In order to assure the correctness of the data during the migration, we consider moving a single fact as the most elementary data migration operation in the algorithm. If after the mifgration has been done every elementary fact found in the source database is present in the target database from a domain perspective point of view, then the migration was carried out correctly.

The algorithm goes through two main phases (both with different substeps):

- *Mapping the source database structure to the target database structure.*
- *Migrating facts.*

Mapping the source database structure to the target database structure:
Mapping between two database structures is only possible when they belong to the same Universe of Discourse [1]. The mapping process is based on *identifying the elementary fact types in the relational source and target databases* and making a *mapping between the elementary fact types of the source database and the target database*. See figure 4.

Suppose there is no fact in the target database that cannot be found in or derived from facts in the source database. Then this means that although the structure of the target and the source databases can be different, the target and source databases belong to the same elementary fact model. Let's consider a simple example to illustrate this. Table **Student** that was used as an example in figure 1 is not normalized. In figure 5 the one table database **Student** is first represented as an elementary fact diagram from which a normalized relational representation is generated, for storing the data. The target database is named **StudentNew**.

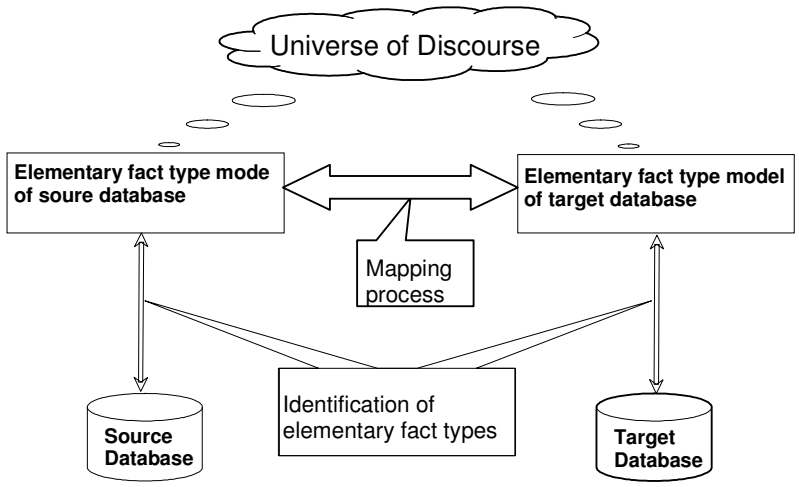
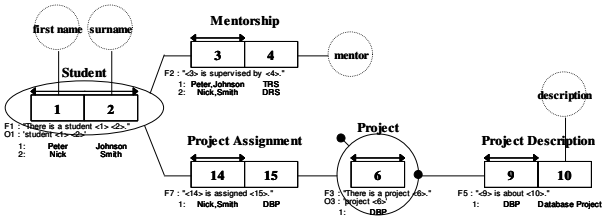


Figure 4: The mapping of the source database structure to the target database structure

Student

Student				
First Name	**Surname**	**Mentor**	**Project**	**Project Description**
Nick	Smith	DRS	DBP	Database Project
Peter	Johnson	TRS		



StudentNew

Student			
First Name	**Surname**	**Mentor**	**Project**
Nick	Smith	DRS	DBP
Peter	Johnson	TRS	

Project	
Project	**Project Description**
DBP	Database Project

Figure 5: From a not-normalized source database to a normalized target database

Determining elementary fact types in the relational model:
During this step a fact type representation of the relational model has to be created, in which every column in the tables of the relational model will belong to a fact type and every atomic data item in the table to a fact. There are many ways to achieve this:

- If the database is generated by an FCO-IM modeling tool such as CaseTalk [3], then the representation is already present.
- If the relational model has a good structure, then we can reverse-engineer [4] the database. This can be done with CaseTalk as well.
- If the database structure does not give us enough information to use the reverse-engineering algorithm [4], then we have to model it first with an FCO-IM modeling tool.

After this step, we will have an elementary model in which the elementary fact types are present. We also still have the relational model and the proper mapping between elementary fact types and the relational model is present as well.

The mapping between the elementary fact types in the source and the target databases:
Such mapping between elementary models can be done in different ways. Depending on the reason why we need the database, we can have different structures that are based on the same elementary model.

If, for example, we want to build an OLTP system, the database must be normalized. If we like to define an OLAP system for the same model as well, then a redundancy-free structure is most likely not not very well suited and as a consequence we will transform the normalized structure into a redundant structure that is optimized for retrieving information with an OLAP tool. In this case, when we transform the model we keep track of its references to the elementary model by automatically preserving the information about the mapping between the two systems.

It is also possible that source and target are generated by different ways. In such case, again the set of fact types found in the target database corresponds to the set of fact types found in the source database, but they are not expressed in the same way. The names of elementary fact types might be different, although expressing the same fact type. In such case, the process of mapping is to be done manually. Suppose for example that from the source database is to be generated / constructed an elementary fact type model by interfering manually, because the source is not redundancy-free. Then from the elementary model we generate the relational model for the target database.

Migrating facts:

The process of migrating facts will use the information introduced during the mapping process. There are two main sub-processes in this phase:

- *Identify and move the independent facts.*
- *Identify and move the dependent facts.*

Identify and move the independent facts:
It is obvious that the independent facts have to be moved first, since these facts can stand alone in the target database. See figure 6 for the different subsequent steps of this sub-process:

- *Recognize the independent fact types in the relational model of the target.*
- *Find out the corresponding fact types in the source.*
- *Move the independent facts.*

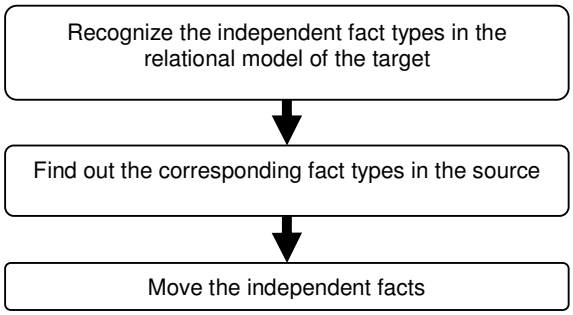


Figure 6: Recognizing and moving the indepenent facts

Recognize the independent fact types in the target:
We consider a subset of the independent fact types, namely those that fulfill the condition that the corresponding columns in the relational data model belong to the primary key of a table. In the elementary fact type model there might be independent fact types that are not independent in the structure of the relational data model (because they are not a primary key in any table). If this is the case, then the relational data model is not redundancy-free. To find such fact types we use the information generated during the mapping of the source relational model to the elementary fact type model and the target relational model.

Find out the corresponding fact types in the source:
Using the information from the mapping between the source and target information grammars, we find the corresponding fact types in the source information grammar and their representation in the relational model of the source. It might be possible that an independent fact type in the target corresponds to a dependent fact type in the source. In this case the relational model of the source is not redundancy-free. For an example, see again figure 5: the fact type **Student** in the source is found in table **Student (First Name, Surname),** which is independent, but the the fact type **Project** that is indepenent in the taget, is to be found in table **Student (Project),** which is not independent in table **Student**.

Move the independent facts:
We know where these facts reside in the source and where they must go in the target. This information is enough to move the facts from the source to the target. There must be the same number of atomic values in the relational model of the source and in the target for each fact. If this is not the case, then the target structure is a derivation of the source structure which means that the facts in the destination system are derived from the facts in the source system. In this case there must be a derivation rule in the mapping between the elementary fact type structures of the source and destination. For an example, see figure 7, where the independent fact types of the database **StudentNew** are already populated.

Student			
First Name	**Surname**	**Mentor**	**Project**
Nick	Smith		
Peter	Johnson		

←

Project	
Project	**Project Description**
DBP	

Figure 7: Populating the independent fact types only

Identify and move the dependent facts:
The process of identifying and moving the dependent facts is a recursive process if the target database is not redundancy-free. Figure 8 shows the algorithm.

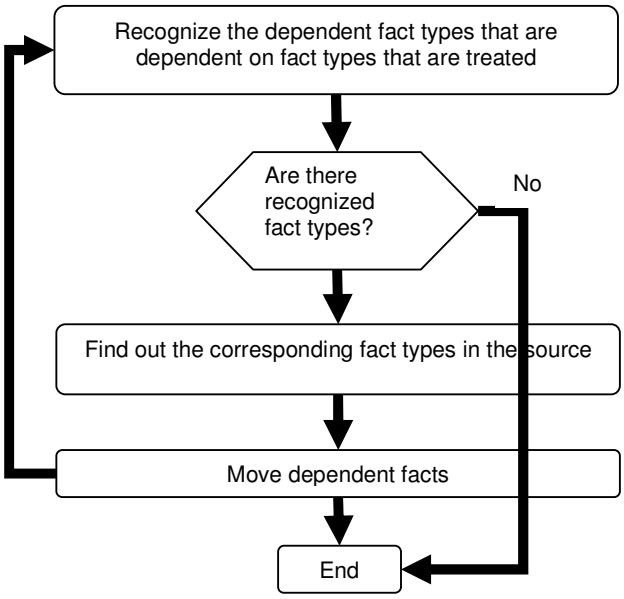


Figure 8: Recognizing and moving the depenent facts

Until all fact types have been treated, the indicated substeps are to be repeated:

Identify dependent fact types that are dependent on fact types that are already treated:
During this step, the algorithm finds fact types that depend on the fact types that have already been treated. Being treated means that the facts related to these fact types are already present in the target. For every fact type treated, find all dependent fact types that are dependent on them in the relational model. In the first occurrence of this step, only the independent fact types are treated (during the step 'Moving the independent facts'). If there are no more fact types depending on the already treated fact types, then the fact transferring process has finished.

See again figure 5. The fact types that are already treated are **Student** and **Project**. Fact types that are dependent on these are now to be identified. For **Student** we have the dependent fact types **Mentorship** and **Project Assignment** and for **Project** there is one dependent fact type **Project Description**.

Find out the corresponding fact types in the source:
Using the mapping information between the source and target fact type models, we find the corresponding fact types in the source information grammar and their representation in the relational model of the source. A dependent fact type in the target will always map to a dependent fact type in the source.

See figure 5. The depenent fact type **Mentorship** is found in the source as **Student (First Name, Surname, Mentor)** and the fact type **Project Assignment** is found as **Student (First Name, Surname, Project)**. For these two depenent fact types the independent fact type is **Student**. The fact type **Project Description** is found in the source as **Student (Project, Project Description)**. For this last fact type the independent fact type is **Project**.

Move dependent facts:
To find out where the dependent facts reside in the source, we need also the fact on which these dependent facts depend. Now we have all the information to move the dependent facts from the source to the target. As in the case of moving independent facts, there must be the same number of atomic values in the relational model of the source and in the target for each fact. If this is not the case, then the target structure is a derivation of the source structure which means that the facts in the destination system are derived from the facts in the source system. In this case there must be a derivation rule in the mapping between the elementary fact type structures of the source and destination.

See again figure 5. In the target database **StudentNew** for every fact present of fact type **Student,** on which the fact type **Mentorship** depends, the algorithm looks into the source. From there it retrieves the corresponding **Student (Mentor)** for the **Student (First Name, Surname)** and puts it in the target.The same will happen for **Project Assignment** and **Project Description**. The result is shown in figure 9.

Student			
First Name	Surname	Mentor	Project
Nick	Smith	DRS	DBP
Peter	Johnson	TRS	

←

Project	
Project	Project Description
DBP	Database Project

Figure 7: Populating the dependent fact types as well

4 Implementation

In chapter 3 has been explained that for running the described algorithm, a relationship between the structure of a relational model and a fact type model must be established. This relationship can be established through a repository that 'considers' data as facts stored in a relational model and that 'sees' these facts as instances of fact types. The implementation of the described algorithm has proved to work properly in a metadata based prototype [2].

5 Conclusion

The fact approach substantially simplifies the complexity of data migration for relational models. It helps for automating many technical aspects of this process.

This paper touches only the basics of data migration. It assumes that the databases are free from unnecessary information and it assumes that the data in the source database is already correct. It does not yet consider row-to-column transformations (which from a fact approach we would call role-to-object transformations) and the other way around. Neither does it consider meaningless keys, which occur often in practice.

References

1. Guido Bakema, Jan Pieter Zwart, Harm van der Lek, 'Fully Communication Oriented Information Modeling (FCO-IM)', HAN University, Arnhem/Nijmegen, The Netherlands 2000. This is an English translation of the Dutch FCO-IM book.
To be downloaded from http://www.FCO-IM.com and/or http://www.CaseTalk.com

2. Mahmoud Abdullah, 'The Automation of Data Mapping', HAN University, Arnhem/Nijmegen, master thesis 2005.

3. FCO-IM modeling tool CaseTalk, BCP Software, Utrecht, The Netherlands 2002-2006. See http://www.CaseTalk.com

4. Elton Manoku, 'RDBMS to FCO-IM', HAN University, Arnhem/Nijmegen, The Netherlands 2003.