

A fact approach on data migration

An algorithm for migrating data from one database system to another

Elton Manoku, June 2006

Freelance IS consultant

emanoku@gmail.com. Tel: +355-69-20-81-566, Tirana, Albania

Research and Competence Group Data Architectures & Metadata Management

Informatics and Communication Academy, HAN University of Applied Science

Beverweerdlaan 3, 6825AE Arnhem, The Netherlands

Abstract: The problem of migrating data is present in almost every application development process, such as data warehousing and application generation. 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[1] rather than 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 [2].

1 Introduction

This material 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 into a 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 the implementation of the algorithm.

2 A fact approach to the relational model

Although in practice data are stored in other kinds of formats as well, here we consider only the relational model. It is after all possible to map other kinds of formats of storing data into a relational model, because any kind of data can be represented in the form of tables.

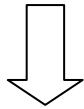
The fact approach assumes that data in databases are facts. Considering data as facts simplifies the validation of the correctness of the data, because the domain experts recognize facts easier than data in the table.

A fact can be found in one or more atomic values, which are the intersections of a column and a row in a relational table. The atomic values combined with soft semantics represent a fact in the database. There cannot be an atomic value that is not part of a fact. Bringing the discussion up to the type level, this means that every relational model can be projected onto a fact type model. Below, we will consider only elementary facts and - at the type level - elementary fact types.

We will consider Fully Communication Oriented Information Modeling FCO-IM [1] as the standard way of looking into the fact based modeling methods.

Example: this example shows how a relational model is represented in a fact based model.

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



Add soft semantics and get a composite fact for a whole row.

Fact 1: Student **Nick Smith** exists and is supervised by **DRS** during his project, which is named **DBP** and has the description **Database Project**.
Fact 2: Student **Peter Johnson** exists and is supervised by **TRS**.



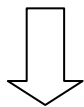
Split the composite facts to reach the most elementary level

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**.

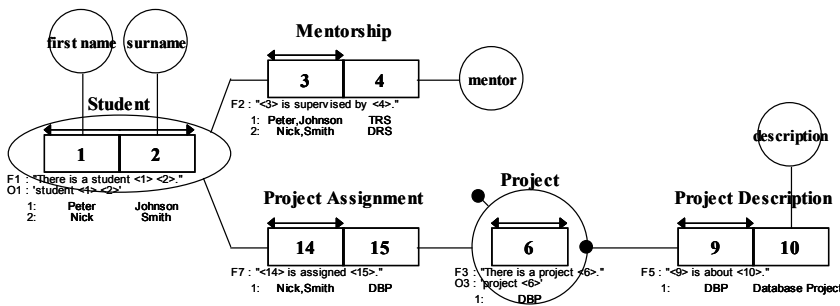


Qualify facts into fact types

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 **<ProjectDescription>**.



Diagramming the elementary fact model (Information grammar)



From the structure of the relational 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. Despite the complexity of the relational model, when it is transformed into an elementary fact type 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.

Example: This example shows how an independent fact type is retrieved from the relational model.

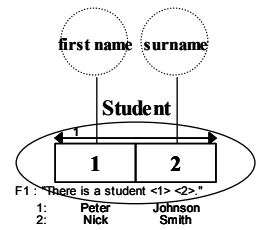
In relational model

<i>Student</i>	
First Name	Surname
Nick	Smith
Peter	Johnson

In elementary fact model

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

FCO-IM Presentation



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.

Example: This example shows a dependent fact type in the relational model. We can see how it is connected with the independent fact type Student.

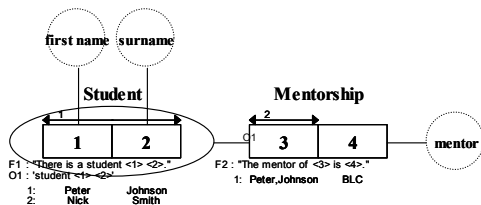
In relational model

<i>Student</i>		
First Name	Surname	Mentor
Nick	Smith	DRS
Peter	Johnson	TRS

In elementary fact model

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

FCO-IM presentation



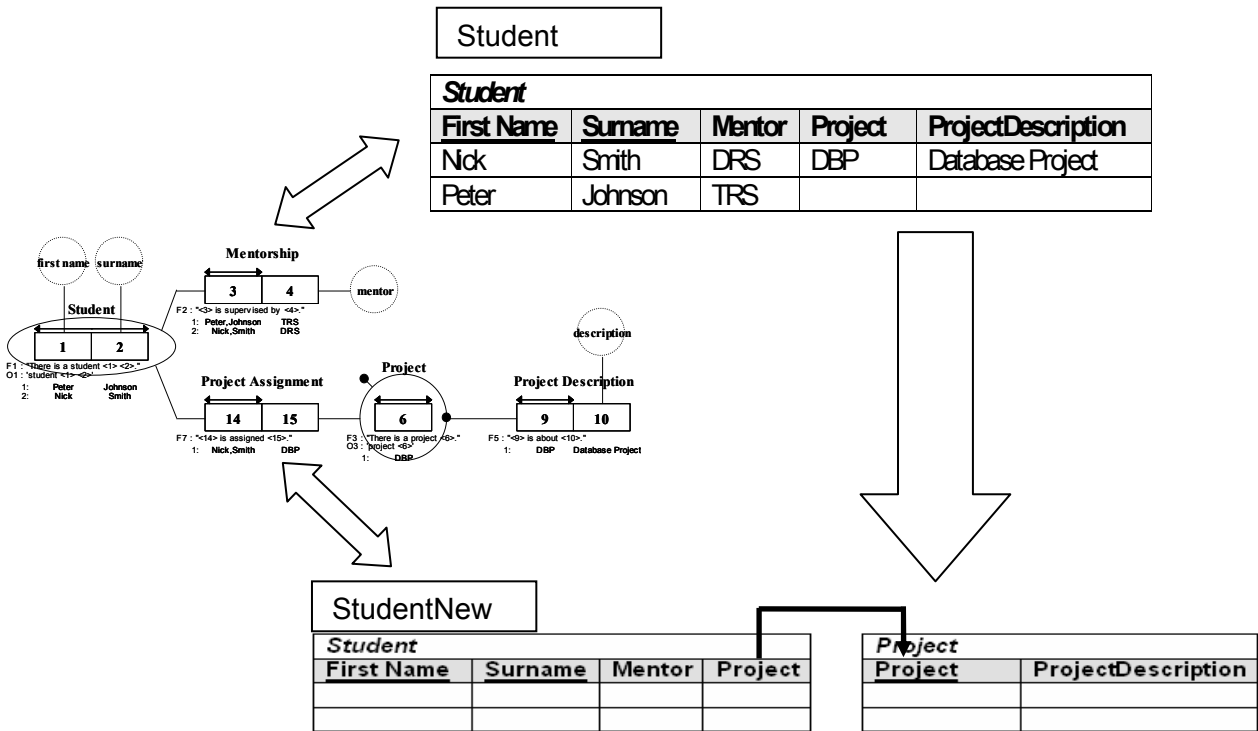
Role 3 is played by the independent fact type Student on which Mentorship depends.

3 The algorithm

The aim of the algorithm is to migrate data from one database to another 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 that has a different structure without altering the facts. In order to assure the correctness of the data during the migration (facts must be unchangeable), we consider moving a single fact as the most elementary data migration operation in the algorithm. If every fact found in the source database is still the same in the target database from a domain perspective point of view, then the migration was carried out correctly. The algorithm goes through different phases:

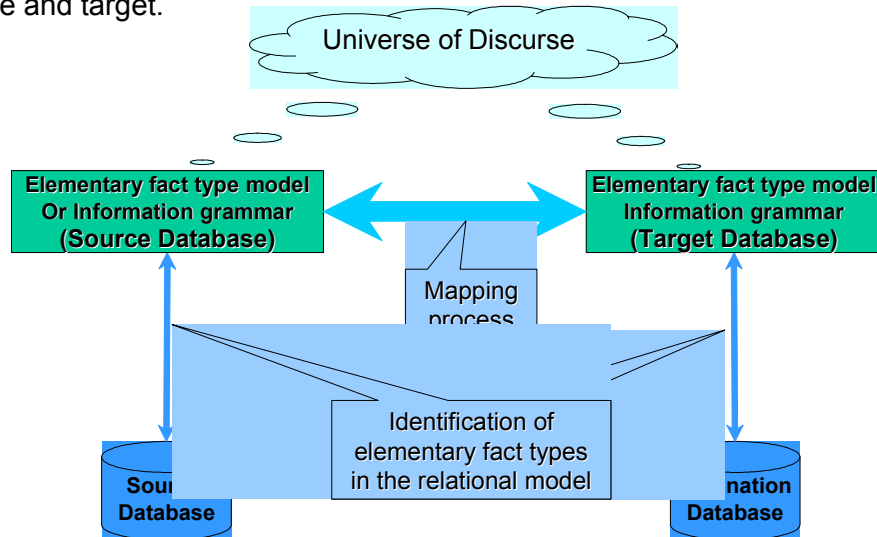
- Mapping the source database structure to the target database structure
- Migrating facts

Let's take a simple example to explain the algorithm: Note that the table Student used as an example above is not normalized. We represented it in an elementary model and generated a normalized relational model to store the data. The new database is the target database named StudentNew.



Mapping the source database structure and destination database structure

Mapping between two database structures is possible only if they belong to the same Universe of Discourse [1]. Suppose that 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, they belong to the same elementary model. Mapping consists of two steps: determining the elementary fact types in the relational model and mapping the elementary fact types of the source and target.



Identifying elementary fact types in the relational model

This step in other words will create a fact type representation of the relational model 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 of achieving 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. If the database structure does not

give us enough information to use the reverse-engineering algorithm [4], then we have to do it manually. 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 elementary fact type structures of the source and the target database

The mapping between elementary models can be done in different ways:

Source and Target are automatically derived from the same elementary model

Depending on the reason why we need the database, we can have different structures that are based on the same elementary model.

Example: If we want to build an OLTP system, the database must be normalized. If for the same model we like to define also an OLAP system, then a redundancy-free structure is not very well suited and we transform it to a redundant structure optimized for retrieving information. In this case, as 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.

Source and Target are generated by different ways

In this 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 the elementary fact types might be different although they express the same fact type. In this case, the process of mapping is done manually.

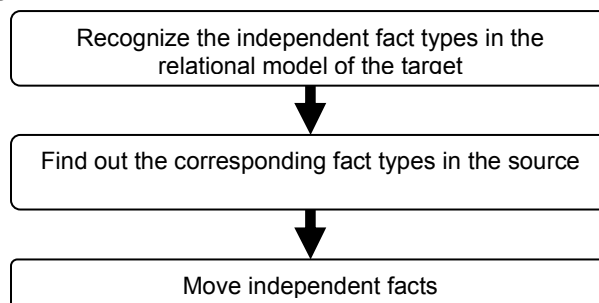
Example: In our example, from the source database is generated the elementary fact type model by interfering manually because the source is not redundancy-free. Then from the elementary model we generate the relational model of the target database.

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: Moving the independent facts and moving the dependent facts.

Moving the independent facts

It is obvious to move the independent facts first, since these facts can stand alone in the target database. Below are given different steps of this sub-process.



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 model belong to the primary key of a table. In the elementary model there might be independent fact types that are not independent in the structure of the relational model (because they are not a primary key in any table). If this is the case, then the relational 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 Information grammar and the target relational model.

Example: There are fact types **Student** and **Project**, and their corresponding relational structure is *Student* (First Name, Surname) and *Project* (Project).

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.

Example: The **Student** fact type in the source is found in **Student(First Name, Surname)**, which is independent as well, but the **Project** fact type is found in **Student(Project)**, which is not independent.

Move independent facts

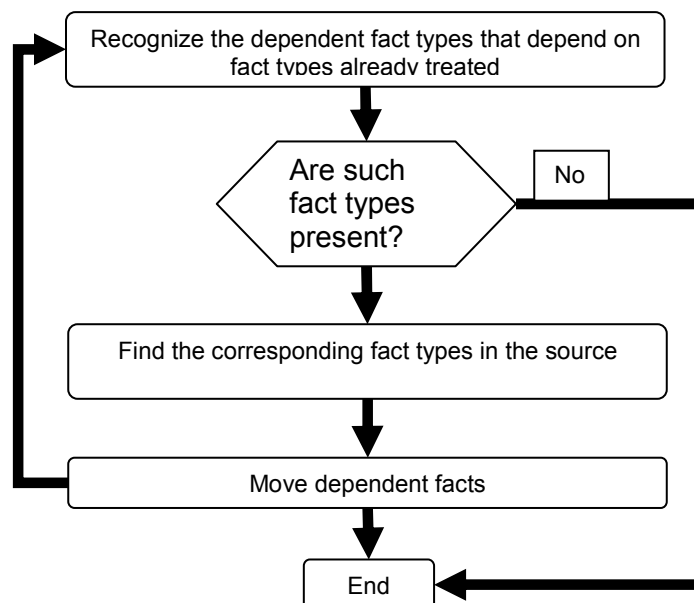
We know where these facts reside in the source and where they must go in the target. This information is enough to move the facts from the source to the target. There must be the same number of atomic values in the relational model of the source and in the target for each fact. If this is not the case, 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.

Example: The independent fact types of the **StudentNew** are populated.

<i>Student</i>				<i>Project</i>	
First name	Surname	Mentor	Project	Project	ProjectDescription
Nick	Smith			DBP	
Peter	Johnson				

Moving the dependent facts

The process of moving the dependent facts is a recursive process if the target database is not redundancy-free. The schema below gives the algorithm for moving the dependent facts.



The steps explained below are repeated until all fact types have been treated.

Recognize the dependent fact types that depend on fact types 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.

Example: The fact types that are treated are: Student and Project. Fact types that are dependent on them are: for Student: Mentorship, Project Assignment and for Project: ProjectDescription.

Find the corresponding fact types in the source

Using the mapping information 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. Always a dependent fact type in the target will map to a dependent fact type in the source.

*Example: The **Mentorship** fact type in the source is found in **Student(First Name, Surname, Mentor)** and the **Project Assignment** fact type is found in **Student(First Name, Surname, Project)**. For these two fact types the independent fact type is **Student**.*

*The **ProjectDescription** fact type in the source is found in **Student(Project, ProjectDescription)**. For this 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.

*Example: The case of **Mentorship** fact type:*

*For every fact present in **StudentNew** 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 **ProjectDescription**.*

<i>Student</i>				<i>Project</i>	
First name	Surname	Mentor	Project	Project	ProjectDescription
Nick	Smith	DRS	DBP	DBP	Database Project
Peter	Johnson	TRS			

4 Implementation

We see that for running this algorithm, we must establish a relationship between the structure of a relational model and a fact type model. This relationship can be established through a repository that considers data as facts stored in a relational model and sees these facts as instances of fact types. The implementation of this algorithm has proved to work in a metadata based prototype[2].

5 Conclusion

The fact approach in data transfer simplifies the complexity that comes from the size of relational models. It helps in automating many technical aspects in this process by increasing the party of people who can participate.

This paper touches only the basics of the transferring data problem. It assumes that the databases are free from unnecessary information and it assumes that the data are already correct in the source database. 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 however occur often in practice.

Acknowledgement

Jan Pieter Zwart is gratefully acknowledged for reviewing the phrasing in this paper.

References

1. Bakema Guido, Zwart Jan Pieter, van der Lek Harm: Fully Communication Oriented Information Modeling FCO-IM (2002). The book can be downloaded from <http://www.CaseTalk.com>.
2. Mahmoud Abdalah: Dact Oriented Data Mapping and transformation (2005). Major thesis for Master of Science in Information Development, Arnhem, the Netherlands.
3. FCO-IM CASE-tool CaseTalk, see <http://www.CaseTalk.com>.
4. Elton Manoku: RDBMS to FCO-IM (2003). HAN University, Arnhem, the Netherlands.