A FORMAL TOOL THAT INTEGRATES RELATIONAL DATABASE SCHEMES AND PRESERVES THE ORIGINAL INFORMATION *

A. Mora, M. Enciso

E.T.S.I. Informática. Universidad de Málaga.
Campus de Teatinos. 29071 Málaga, Spain

Keywords: Functional dependencies, Logic and Information systems; Schema integration.

Abstract: In this work we face on with the main problem concerning the database design process in a collaborative environment: several users provide different models representing a part of the global model and we must integrate these database sub-shemes to render a unified database schema.
In this work we propose a technique to integrate relational database sub-schemes based on a formal language. The extraction, integration and generation tasks are carried out efficiently using the $\mathbf{SL}_{FD}$ logic (Substitution Logic for functional dependencies). We have selected this logic because it is appropriated to manage the functional dependencies in a automatic way.

1 INTRODUCTION

In (Chang and Moskowitz, 2000) the authors affirm that "the development of software to ease the integration and interoperation of existing information sources is one of the most significant challenges currently facing computer researchers and developers". This sentences emphasizes the needed of intelligent tools to automate the collaborative database design database when several designers participate in the process.

Thus, different sub-sets of information are provided and a unique and consistent set containing all the information must be built. The main difficulties arise from the matching of the different structures defined by the designers in their sub-schemes.

As *Arch-int and Batanov* says in (Arch-int and Batanov, 2003), the goal is "*to get integrate database in a global model without the existence of redundances and inconsistencies*". In this work we cover both goals by using the Functional Dependency (FD) notion as the center of our integration architecture.

FDs are powerful relational constraints introduced in the 70 by E.F. Cood (Codd, 1974) and consolidated by outstanding authors like W. Armstrong (Armstrong, 1974) and R. Fagin (Fagin, 1977).

A set of recent works have showed that there exists a set of classical FD problems which may be successfully treated with novel techniques and tools (Lee et al., 2002). This renewal of FDs are being used also in the design of databases. Thus, X. Linga (Ling et al., 1996) considers that "the objective of the logical design step is to eliminate redundancies and updating anomalies using the notion of data dependencies".

In (Enciso and Mora, 2002; Mora et al., 2004), we propose a *Functional Dependencies Data Dictionary* $FD3$, a tool to store the information provided by different sources in a unified way. $FD3$ gathers the information provided by the original sub-schemes, but it flattens this information removing the two dimensions of the E/R model. So, $FD3$ is a set of FDs generated over the set $\mathcal{A}$ of all the attributes defined by all the designers.

Now, we introduce an integration architecture which uses $FD3$ as a central element. The architecture provides a framework to cover all the stages of the collaborative database design with a full level of automatization.

In this work, we present a technique to integrate the information in a full automated way. It works as follows: the collaborative coordinator indicates the sub-schemes to be integrated. In the architecture, the structural functional dependencies (from the primary key and unique keys) are considered and the collaborative coordinator are asked to add extra FDs (named

*This work has been partially supported by the CICYT research project TIC2003-08687-C02-01.

Mora A. and Enciso M. (2006).
A FORMAL TOOL THAT INTEGRATES RELATIONAL DATABASE SCHEMES AND PRESERVES THE ORIGINAL INFORMATION.
In *Proceedings of the Eighth International Conference on Enterprise Information Systems - DISI*, pages 302-305
DOI: 10.5220/0002451603020305

environment FDs). The FDs belonging to different sub-shemes are uniformly stored in a FD Data Dictionary and the automated algorithms based on the $\mathbf{SL}_{FD}$ inference system reduce the redundancy and render a set of FDs that may be translated into a relational database.

The technique preservers the foreign key constraints and normalizes the output tables into Boyce Codd Normal Form. We have design an algorithm which uses the trace of the $\mathbf{SL}_{FD}$ algorithm execution and generates a set of INSERT sentences over the output tables to preserve the original data contained in the original tables.

2 AN ARCHITECTURE FOR SCHEME INTEGRATION

We present in this section, a formal tool appropriated for the integration of DBMS schemes.

We have selected Oracle as the target DBMS. An oracle scheme is defined as follows: a user (with password) are associated with a work area where he defines attributes, tables and the following constraints (*dependencies*): Primary, Unique and Foreign key.

The architecture (figure 1) have three levels:

- Back-end: the extraction of FDs of the scheme.
- Front-end: the user introduce additional FDs.
- Middle-end: the integration method (based on logic) renders a global scheme without redundancy.

The formal tool is directly based on this architecture and it has three modules:

- **Integration engine**, developed in PL/SQL, is responsible to make both structural integration (generates a scheme containing the elements of the schemes) and the integration of contents (cover the migration of the original data).
- The **FD Data Dictionary (ODD)**, implemented as an extra *Oracle* scheme, allows the integration engine to store, query and manipulate efficiently all the information that needs throughout the integration process.
- **User interface**, implemented by means of *portless* inside of Oracle Portal. This web interface acts as an assistant that simplifies the configuration of integration process and allows the user to examine in detail the output schema and the way in which it has been obtained.

2.1 Solving Structural Integration

Our approach use exhaustively the FDs in all the stages of the integration process. Thus, the functional

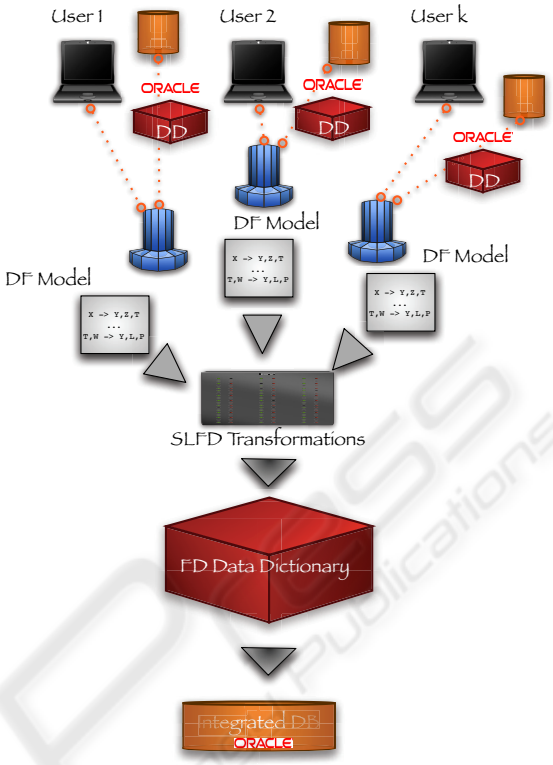

Figure 1: Integration architecture.

dependencies data dictionary $FD3$ (Enciso and Mora, 2002; Mora et al., 2004) is used to store the FDs collected from the original schemes.

These FDs are inferred from the original key constraints specified in the Oracle schemes and they are named *structural dependencies*. We enrich this reverse engineering process with an user aided process where an external agent with domain knowledge may specify additional FDs to provide additional semantic information (*the environment dependencies*).

When both kinds of FDs have been collected, they are depurated and integrated by the application of the *RemoveRedundancy* algorithm over the $FD3$ data dictionary.

The algorithm is based directly in the $\mathbf{SL}_{FD}$ inference system rules.

Definition 2.1 (The $\mathbf{SL}_{FD}$ language) *Let Ω be an infinite enumerable set of atoms (attributes) and let $\mapsto$ be a binary connective, we define the language*

$$\mathcal{L}_{FD} = \{X \mapsto Y \mid X, Y \in 2^{\Omega}\}$$

Definition 2.2 *The $\mathbf{SL}_{FD}$ logic is defined as the pair $(\mathcal{L}_{FD}, \mathcal{S}_{FDS})$ where $\mathcal{S}_{FDS}$ has one axiom scheme: $\lfloor Ax_{FDS} \rfloor : \ \vdash X \mapsto Y$, where $Y \subseteq X$.*

Particulary, $X \mapsto \top$ is an axiom scheme.

The inferences rules are the following:

$\lfloor Frag \rfloor$ **Fragmentation rule:**

$X \mapsto Y \vdash_{\mathcal{S}_{FDS}} X \mapsto Y'$ *if* $Y' \subseteq Y$

$\lfloor Comp \rfloor$ **Composition rule:**

$X \mapsto Y,\ U \mapsto V \vdash_{\mathcal{S}_{FDS}} XU \mapsto YV$

$\lfloor Subst \rfloor$ **Substitution rule:**

$X \mapsto Y,\ U \mapsto V \vdash_{\mathcal{S}_{FDS}} (U\text{-}Y) \mapsto (V\text{-}Y)$

if $X \subseteq U,\ X \cap Y = \varnothing$

Also, a novel derived rule is defined:

$\lfloor rSust \rfloor$ **r-Substitution Rule:**

$X \mapsto Y, U \mapsto V \vdash_{\mathcal{S}_{FDS}} U \mapsto (V\text{-}Y)$

if $X \subseteq UV, X \cap Y = \varnothing$

Finally, we will use the following derived rules:

$\lfloor Union \rfloor$ **Union Rule:**

$X \mapsto Y,\ X \mapsto Z \vdash_{\mathcal{S}_{FDS}} X \mapsto YZ$

$\lfloor Reduc \rfloor$ **Reduction Rule:**

$X \mapsto Y \vdash_{\mathcal{S}_{Par}} X \mapsto Y\text{-}X$, where $Y\text{-}X \neq \varnothing$

The algorithm *removeredundancy* is depicted in the following figure:

Algorithm 2.3

removeRedundancy:

Input: Γ (a FD set)
Output: Γ' (a FD set with less redundancy)
Begin

1. $\lfloor Reduc \rfloor$
2. $\lfloor Union \rfloor$

Repeat

3. Substitution:
   $\lfloor Sust \rfloor + \lfloor rSust \rfloor$

Until more substitution cannot be applied

End

We remark that this is the first approach to n-Aryan integration of relational data bases that uses directly logic as a tool of simplification and integration.

Our algorithms simplifies attributes and removes redundancy from the global data dictionary. This redundancy arises when the information of the subschemes are joined in a unique data dictionary.

From the $FD3$ Data Dictionary, we infer an Oracle unified scheme. We have designed an algorithm able to generate SQL sentences which creates the tables, as well as their foreign and primary key restrictions.

In addition to the conventional FDs, there are other dependencies stored in the data dictionary which we will call **foreign key dependencies (FKD)**.

These FKDs represent the foreign key constraints existing in the original schemes so that their information can be included later in the new integrated scheme.

2.2 Solving Contents Integration

We have developed a powerful and flexible algorithm that constructs a nested SELECT sentence, cradle in OUTER JOIN, for each table in the integrated scheme.

The migration of original data is guided by the result of the structural integration process.

Thus, the INSERT SQL sentences that store the original rows into the new tables are built using the guidelines provided by the *remove redundancy algorithm*.

The efficiency of this algorithm has been improved using the information stored in the FD^3. Whenever an inference rule of $\mathbf{SL}_{FD}$ logic is applied, we keep in the FD^3, a link between each attribute of the new unified scheme and the set of attributes of the original schemes that has produced it.

We remark that it is an incremental process.

3 CONCLUSIONS AND FUTURE WORK

In this paper we present a formal tool directly based in the $\mathbf{SL}_{FD}$ logic. It deduces the FDs (structural dependencies) contained in the key constraints of a set of schemes from an Oracle database.

The user may add others dependencies (environment dependencies) that represent a more deeper knowledge of the sub-systems.

The tool applies an optimization algorithm directly based on the $\mathbf{SL}_{FD}$ logic. Thus the functional dependencies data dictionary (FD3) is transformed and a global database may be inferred from its.

We use a generic algorithm which allows a migration of data to the output tables, avoiding the loss of the information.

In a medium-future, we will complete the integration software with a view generation algorithm. This extension will be used to join several information systems that was built in the past.

The views will be used to prevent the re-compilation of the applications when the new global database substitute the old set of databases.

The Functional Dependencies appear in other data models apart from the Relational Model. As a future work, we may extend the algorithm to generate an FD model from other source data model such hierarchic, network data model or even XML.

REFERENCES

Arch-int, S. and Batanov, D. (2003). Development of industrial information systems on the web using business components. *Computers in Industry*, 50 (2):231–250.

Armstrong, W. W. (1974). Dependency structures of data base relationships. *Proc. IFIP Congress.*, pages 580–583.

Chang, L. and Moskowitz, I. S. (2000). An integrated framework for database privacy protection. *IFIP Workshop on Database Security*, ISBN: 0-7923-8129-7:161–172.

Codd, E. F. (1974). Recent investigations into relational data base systems. *IFIP Congress, Estocolmo, Suecia*.

Enciso, M. and Mora, A. (2002). FD3: A functional dependencies data dictionary. *Proceedings of the Fourth Conference on Enterprise Information Systems - ICEIS*, 2:807–811.

Fagin, R. (1977). Functional dependencies in a relational database and propositional logic. *IBM. Journal of research and development*, 21 (6):534–544.

Lee, M. L., Ling, T. W., and Low, W. L. (2002). Designing functional dependencies for xml. *LNCS*, 2287:124–141.

Ling, T., Goh, C., and Lee, J. (1996). Extending classical functional dependencies for physical database design. *Information and Software Technology*, 38:601–608.

Mora, A., Enciso, M., Cordero, P., Guzmán, I. P. d., and Guerrero, J. (2004). A/d case: a new heart for FD3. *Proceedings of ICEIS 2004 - 6th International Conference on Enterprise Information Systems. Porto, PORTUGAL*.