Building a data warehouse using heterogeneous data sources

Sana Hamdoun and Faouzi Boufares
LIPN-UMR 7030 of C.N.R.S - Paris 13 University, Galilee Institute
99, avenue J. B. Clément, F-93430 Villetaneuse FRANCE.
Sh@lipn.univ-paris13.fr
boufares@lipn.univ-paris13.fr

Abstract: This work describes the construction of a data warehouse by the integration of heterogeneous relational and object-relational data. In fact, developing intelligent tools for the integration of information extracted from multiple heterogeneous sources is a challenging issue to effectively exploit the numerous sources available in global information systems. Due to the heterogeneity of the sources, various languages of interrogation and different data models are used for the warehouses. Thus the construction of the latter can be made in several manners. Our work is based on the extraction of the inter-schema relationships between the sources. Related to this, a global schema is generated and the views of the data warehouse are constructed. All these stages proposed in this work are implemented by the use of a functional prototype.

Key words: Database, data warehouse, heterogeneous structures and data, integration.

1 Introduction

The goal of information integration techniques is to construct synthesized information coming from multiple sources. In recent years, there have been many research projects focusing on heterogeneous information integration [JG04]. In fact, several contributions appeared in the recent literature, including methods, techniques and tools for integrating and querying heterogeneous databases [PV01], [DB02], [DU02], [PN02], [PV03], [BN03]. The heterogeneity not only concerns the semantic of data but it can also concerns the data structures. In this paper, we propose a framework of integration of heterogeneous structured databases (relational and/or object-relational). This work is based on the extraction of different relationships between sources schemas. According to this relationships, we generate the global schema of the integration result (the data warehouse). The construction step is then realized by the generation of different interrogation requests of the data sources.

Our contribution treats not only the heterogeneity of data but also that of structures. Moreover, we can note that we do not simply compare attributes names, to extract the inter-schema relationships, but we also compare attributes domains and constraints.

This paper is organized as follows. Section 2 describes the inters-schema relationships extraction. Section 3 is devoted to the global schema generation. Section 4 describes the methodology of construction. Our prototype architecture is presented in section 5, and then, we give our concluding remarks. Finally, in the appendix, we present some parts of the prototype implemented.

2 Interschema relationships extraction

The first step of our work is the extraction of relationships between different components of sources schemas. Currently, these relationships are proposed by the data warehouse administrator, our future goal is to automate or partially automate this step.

We define three types of relationships between attributes: the synonymy (SYN), the inclusion (INC) and the disjunction (DISJ). To illustrate these relationships we consider two relational and/or object-relational databases: DB_1 and DB_2.

DB_1 is made of m relations DB_1={R_i} 1 ≤ i ≤ m, each relation R_i is composed of n_i attributes R_i={ A_j } 1 ≤ j ≤ n_i.

DB_2 is made of m’ relations DB_2={R’_p} 1 ≤ p ≤ m’, each relation R’_p is made of n’_p attributes R’_p={A’_q} 1 ≤ q ≤ n’_p.

Each attribute A is defined by a domain Dom(A)
which is the set of valid instances of A. Dom(A) can be either a predefined domain (varchar, number, date) or a user defined domain (type).

We note DB.R.A the attribute of the relation R belonging to the database DB.

**Definition 1 : The Synonymy**
An attribute A is a synonym of an attribute A’ if they have the same domains.
We can validate this relationship by verifying that the set of constraints defined on Aⱼ and A’ᵣ are either identical or, at least, do not present contradiction.

**Definition 2 : The Inclusion**
We can validate this relationship by verifying that the constraints defined on A’ᵣ are a subset or the same of the constraints defined on Aⱼ.

**Definition 3 : The Disjunction**
Note that in the case of object-relational database, attributes can be composed ones. Relationships between composed attributes can be validated using the following rules:
Let’s consider two composed attributes : DB₁.Rᵢ.Aⱼ={c₁,…,cᵣ} and DB₂.R’ₚ.A’ᵣ={c’₁,…,c’ᵣ}.

**Rule 1** : DB₁.Rᵢ.Aⱼ SYN DB₂.R’ₚ.A’ᵣ if r=r’ and ∀ s / 1 ≤ s ≤ r, cₛ SYN c’ₛ.


**Example 1** : Two databases

DB₁ represents the information system of a company of production and DB₂ represents the information system of a service company.

**Example 2** : A set of synonyms
Let’s take into account the relationships given below to integrate DB₁ and DB₂.

DB₁.client.id_client SYN DB₂.company.id_co  
DB₁.client.address SYN DB₂.company.address  
DB₁.sale.amount SYN DB₂.operation.amount  
DB₁.product.id_productSYN DB₂.service.id_service  
DB₁.product.label SYN DB₂.service.label  
DB₁.sale.id_sale SYN DB₂.operation.id_operation  
DB₁.sale.id_client SYN DB₂.operation.id_co  
DB₁.sale.id_product SYN DB₂.operation.id_service  
DB₁.sale.date_of_sale SYN DB₂.operation.start_date

Once the different schemas and relationships between their attributes are given, we can move to the presentation of our methodology of the global schema generation.

### 3 Global schema generation
A data warehouse (DW) is composed of a set of views. DW={Vi}/ i ∈ [1..n]. Each view is constructed by interrogation and integration of data from different sources.

In our work, we propose to generate the global schema while considering the set L of sources attributes which must appear in the DW. L={DBk.Rᵢ.Aⱼ} / k ∈ [1..d]; i ∈ [1..rk] and j ∈ [1..ai]. L is proposed by the DW administrator according to decision needs.
We note \( d = \) the number of DBs; \( rk = \) number of relations of DB\(_k\); \( ai = \) number of attributes of Ri

The global schema generation is done by the following way (six steps):

1- The first step of the global schema generation is to filter the set \( L \) to form the set \( L' \) by eliminating from \( L \) one attribute from each couple of synonymous ones. \( L' \) does not contain synonyms and \( L' \) is included in \( L \).

2- \( L' = \bigcup_{k \in [1..d]} (L'k) \) such as each \( L'k \) contains only the DB\(_k\) attributes. \( L'k = \{DBk.Ri.Aj\} i \in [1..rk] \) and \( j \in [1..ai] \).

3- For each \( L'k \) a non oriented graph \( G_k \) is built. Each node corresponds to a relation \( Ri \) from DB\(_k\) and each arc denotes reference constraints between two relations. \( d \) graphs are obtained.

4- Each graph \( G_k \) may be composed of subgraphs strongly connex \( G_k = \{G'ko\} / o \in [1..tk] \); with \( tk = \) number of subgraphs strongly connex of \( G_k \). A graph is strongly connex if there is at least one path between each two nodes of the graph [LZ03].

5- Considering the \( d \) graphs \( \{G_k\} / k \in [1..d] \), divided on subgraphs, the different possible combinations between the subgraphs are realized by taking one subgraph from each graph \( G_k \). The set \( C \) of combinations is obtained : \( C = \{G'ko\} / k \in [1..d] \) and \( o \in [1..tk] \).

6- A view \( Vu \) is associated to each combination \( Cu \) from \( C \). \( Vu \) contains all the attributes belonging to \( L' \) and having their respective relations represented by nodes in the subgraphs forming \( Cu \). \( T \) views are then obtained. The schemas of the \( T \) views obtained form the global schema of the warehouse.

Example 3: A list of attributes, the set \( L \)

Let us now generate the global schema based on the given set of attributes \( L \) using Example 1:

\[
L = \{DB1.client.(id_client, first_name, last_name, address, type); DB1.product.(id_product, label); DB1.sale.(id_sale, id_client, id_product, quantity, amount, date_of_sale); DB1.employee.(id_emp, first_name, last_name, address); DB1.Company.(id_co, name, address); DB2.service.(id_service, label); DB2.operation.(id_op, id_co, id_service, duration, amount, start_date)\}.
\]

The global schema generation is done as follows:

1- The first step is:

\[
L' = \{ DB1.client.(id_client, first_name, last_name, address, type); DB1.product.(id_product, label); DB1.sale.(id_sale, id_client, id_product, quantity, amount, date_of_sale); DB1.employee.(id_emp, first_name, last_name, address); DB1.Company.(id_co, name, address); DB2.service.(id_service, label); DB2.operation.(id_op, id_co, id_service, duration, amount, start_date)\}
\]

\( L' \) with out synonyms is then:

\[
L' = \{ DB1.client.(id_client, first_name, last_name, address, type); DB1.product.(id_product, label); DB1.sale.(id_sale, id_client, id_product, quantity, amount, date_of_sale); DB1.employee.(id_emp, first_name, last_name, address); DB2.operation.duration\}
\]

2- The second step is:

\[
L'1 = \{ DB1.client.(id_client, first_name, last_name, address, type); DB1.product.(id_product, label); DB1.sale.(id_sale, id_client, id_product, quantity, amount, date_of_sale); DB1.employee.(id_emp, first_name, last_name, address); DB2.operation.duration\}
\]

3- Several graphs are constructed.

The graph \( G1 \):

The graph \( G2 \):

4- Strongly connex graphs

\( G1 = \{G'11, G'12\} \) and \( G2 = \{G'21\} \)
Steps 5 and 6 - the views are then calculated:
T = the number of the views.
t1 = 2 and t2 = 1. T = t1*t2 = 2.
Two views V1 and V2 are then obtained:
V1 = { DB1.client.(id_client, first_name, last_name, address, type) ; DB1.product.(id_product, label) ;
DB1.sale.(id_sale, id_client, id_product, DB1.sale.(quantity, amount, date_of_sale) ; DB1.operation.duration
}
V2 = { DB1.employee.(id_emp, first_name, last_name, address) ; DB2.operation.duration

4 Data warehouse/Views Construction

The global schema generation allows the determination of the number of views composing the DW and their contents. In fact, L can describe one or more views.

For each view V from the GS, V = {DBk.Ri.Aj}/ k ∈ [1..d]; i ∈ [1..rk] and j ∈ [1..ai].
The construction of the DW consists in the construction of all the views of the global schema.
To illustrate our approach, let’s build a view V where V = {DB1.RiAj, DB2.R’p.A’q}. We proceed in the same way in the case of several attributes.

Four steps are needed for the construction of the DW. First the join conditions in the databases are established. Second, the sets of synonyms are found. Then the sets of inclusions are made up. At last requests are generated.

The construction is done by the following way:

1- Join conditions
C1 ← the set of all the join conditions in DB1.
C2 ← the set of all the join conditions in DB2.

2- Synonyms
We note: NS(A) = the number of synonyms (SYN) of A.
S(DB1.RiAj) ← the set of synonyms of DB1.RiAj which belong to DB2:
S(DB1.RiAj) = {DB2.R’k.Sx(Aj)}/ k ∈ [1..r2], x ∈ [1..NS(Aj)].
S(DB2.R’.p.A’q) ← the set of synonyms of DB2.R’.p.A’q which belong to DB1:
S(DB2.R’.p.A’q) = {DB1.Rk.Sz(A’q)}/ k ∈ [1..r1], z ∈ [1..NS(A’q)].

3- Inclusions
We note: Ni(A) = number of attributes which are include (INC) in A.
I(DB1.RiAj) ← the set of attributes which are include in DB1.RiAj and belong to DB2:
I(DB1.RiAj) = {DB2.R’k.Iy(Aj)}/ k ∈ [1..r2], y ∈ [1..Ni(Aj)].
I(DB2.R’.p.A’q) ← the set of attributes which are included in DB2. R’.p.A’q and belong to DB1:
I(DB2.R’.p.A’q) = {DB1.Rk.Iw(A’q)}/ k ∈ [1..r1], w ∈ [1..Ni(A’q)].

4- Generation of requests

DB1 interrogations:
FOR each DB1.Rk.Sz(A’q) IN S(DB2.R’.p.A’q) generate the request Q(Sz(A’q)) END;
Q(Sz(A’q)) can be defined by the SQL language:
Select Aj, Sz(A’q) from Ri, Rk where Cond1;
With Cond1 the join condition between Ri and Rk, extracted from C1.
We note Q(S(A’q)) the set of all the requests Q(Sz(A’q)) generated, z ∈ [1..NS(A’q)].
FOR each DB1.Rk.lw(A’q) IN I(DB2.R’.p.A’q) generate the request Q(lw(A’q)) END;
Q(lw(A’q)) can be defined by the SQL language:
Select Aj, lw(A’q) from Ri, Rk where Cond1;
We note Q(I(A’q)) the set of all the requests Q(lw(A’q)) generated, w ∈ [1..Ni(A’q)].

DB2 interrogations:
FOR each DB2.R’.k.Sx(Aj) IN S(DB1.Ri.Aj) generate the request Q(Sx(Aj)) END;
Q(Sx(Aj)) can be defined by the SQL language:
Select Sx(Aj), A’q from R’k, R’p where Cond2;
With Cond2 the join condition between R’k and R’p, extracted from C2.
We note Q(S(Aj)) the set of all the requests Q(Sx(Aj)) generated, x ∈ [1..NS(Aj)].
FOR each DB2.R’.k.Iy(Aj) IN I(DB1.Ri.Aj) generate the request Q(Iy(Aj)) END;
Q(Iy(Aj)) can be defined by the SQL language:
Select Iy(Aj), A’q from R’k, R’p where Cond2;
We note Q(I(Aj)) the set of all the requests Q(Iy(Aj)) generated, y ∈ [1..Ni(Aj)].
The construction of the view V = {DB1, Ri.Aj, DB2.R’.p.A’q} is realized by the execution of the SQL Union statement of all the requests in Q = {Q(S(Aj)) ∪ Q(I(Aj)) ∪ Q(S(S’A’q)) ∪ Q(I(I’A’q))}

Example 4 : Views building
Let’s now build the view V = (DB1.client.id_client, DB2.operation.amount) using Example 1.

1- Join conditions
C1 = {sale.id_client = client.id_client, sale.id_product =product.id_product; sale.id_service = service.id_service}
C2 = {operation.id_co = company.id_co, operation.id_service = service.id_service}

2- Synonyms
S(DB1.client.id_client)= {DB2.company.id_co}
S(DB2.operation.amount)= {DB1.sale.amount}

3- Inclusions
I(DB1.client.id_client)= {} I(DB2.operation.amount)= {}
6 Conclusion

In this paper, we have described an approach of integration of heterogeneous relational and object-relational databases. This approach is implemented by the mean of a prototype using PL/SQL and the Oracle Data Dictionary. We have treated, in this work, structured heterogeneous data using extraction of relationships between attributes. Currently, we try to automate this step using graph theory [LZ01]. We project in the future to extend our work by using semi-structured (XML data) [GM02], [HB03] and unstructured Data (Multimedia data).

REFERENCES

[LZ01]: S.K. Lellahi, A. Zamulin; « Object oriented database as a dynamic system with implicit state », proceedings of the Fifth East-European Conference on Advances in Database and Information System (ADBBIS’01), Vilnius, Lithuania, September 01.


[LZ03]: S.K. Lando and A.K. Zvonkin, « Graphs on...


[OR04] : «Oracle Database 10g»
http://otn.oracle.com/software/products/database/oracle10g/index.html
APPENDIX

procedure Graph_division is
j,i,max number;
varbase liste1.ref_base%type;
varrel liste1.rel%type;
varatt liste1.att%type;
varbase2 groupes1.ref_base%type;
varrel2 groupes1.rel%type;

begin
j:=1;

while 0 < (SELECT COUNT(*) from liste1)

loop
select ref_base into varbase from (select ref_base, rel, att, rank() over(order by ref_base,rel,att)) r from liste1) where r=1; select rel into varrel from (select ref_base, rel, att, rank() over(order by ref_base,rel,att)) r from liste1) where r=1; select att into varatt from (select ref_base, rel, att, rank() over(order by ref_base,rel,att)) r from liste1) where r=1; insert into groupes1 values (j, varbase, varrel, varatt); delete * from (select ref_base, rel, att, rank() over(order by ref_base,rel,att)) r from liste1) where r=1;

i:=1;
select count(*) into max from liste1;
select ref_base into varbase2 from groupes1 where grp=j; select rel into varrel2 from groupes1 where grp=j; while i<= max
loop
select ref_base into varbase from (select ref_base, rel, att, rank() over(order by ref_base,rel,att)) r from liste1) where r=1; select rel into varrel from (select ref_base, rel, att, rank() over(order by ref_base,rel,att)) r from liste1) where r=1; select att into varatt from (select ref_base, rel, att, rank() over(order by ref_base,rel,att)) r from liste1) where r=1;
if exists ( select * from join_table where (rel1=varrel and rel2=varrel2) or (rel1=varrel2 and rel2=varrel) then insert into groupes1 values (j, varbase, varrel, varatt); delete * from (select ref_base, rel, att, rank() over(order by ref_base,rel,att)) r from liste1) where r=1; max:=max+1; else
i:=i+1;
end if;
end loop;
if i>j
end loop;
end loop;
end;

procedure GS_Generation1 is
nbgrp1 number;
bgrp2 number;
k number;
begin
k:=1;
select max(grp) into nbgrp1 from groupes1;
select max(grp) into nbgrp2 from groupes2;
for i in 1..nbgrp1 loop
for j in 1..nbgrp2 loop
for e1 in (select * from groupes1 where grp=i)
insert into vues values (k, e1.ref_base, e1.rel, e1.att);
end loop;
for e2 in (select * from groupes2 where grp=j)
insert into vues values (k, e2.ref_base, e2.rel, e2.att);
end loop;
k:=k+1;
end loop;
end loop;
end;

procedure GS_Generation2 is
nbrv number;
begin
select max(numv) into nbrv from vues;
for i in 1..nbrv loop
for e1 in (select * from vues where numv=i)
for e2 in (select * from vues where numv=i)
if exists (select * from liens where (ref_base1=e1.ref_base and rel1=e1.rel and att1=e1.att and ref_base2=e2.ref_base and rel2=e2.rel and att2=e2.att and lien='SYN') or (ref_base1=e2.ref_base and rel1=e2.rel and att1=e2.att and ref_base2=e1.ref_base and rel2=e1.rel and att2=e1.att and lien='SYN')) then
insert into vues values (k, e1.ref_base, e1.rel, e1.att);
delete * from vues where (ref_base=e2.ref_base and rel=e2.rel and att=e2.att and numv=i);
end if;
end if;
end loop;
end loop;
end loop;
end;

The procedure « graph_division » uses the list of attributes, given by the administrator, and allows the representation of the graphs relating to data sources and the division of these graphs on strongly connex subgraphs.

The procedures « GS_Generation1 » and « GS_Generation2 » use the strongly connex subgraphs to generate the views of the DW and their attributes. The global schema generated is stored in the table Vues=(numv:number; ref_base:varchar; rel:varchar; att:varchar) where «numv» refers to the the number of the view, «ref_base» refers to a data source, «rel» refers to a relation and «att» refers to an attribute.