EQUIVALENCE AND MAPPING OF DATABASE SCHEMES

A.D'Atri

University of Rome
Italy

D.SACCA'

CRAI, Rende
Italy

ABSTRACT

We investigate the problem of database equivalence which arises in database design process. We introduce a graph formalism for the treatment of this problem. More precisely, we represent Entity-Relationship schemes by a special kind of graphs (called JFD-graphs) and we give a simple and efficient algorithm for testing the equivalence of two schemes. In addition, we present a set of elementary operators (preserving equivalence) for modifying an Entity-Relationship scheme and we prove that all equivalent schemes can be obtained by repeatedly applying such operators. Finally, we propose a methodology for mapping Entity-Relationship schemes into both relational and network schemes.

1. INTRODUCTION

In database design process four phases may be used to derive a DBMS-processable database scheme from an informal description of the real world (for an introduction to these topics see [12, 14]). First of all, in the user requirement analysis phase all information about the realm of interest is gathered; then, during the conceptual design phase, the information collected in the previous phase is manipulated and described in terms of a formal conceptual data model (conceptual scheme). This scheme is "entity-oriented", in the sense that entities of the real world are represented in terms of data model tokens by means of abstraction and aggregation primitives. In this framework, semantic properties of data are expressed by means of a set of integrity constraints. In the implementation phase the conceptual scheme is transformed in an equivalent DBMS-processable scheme by using a data model (network, relational or hierarchycal). The DBMS-processable scheme is, in general, "record-oriented" in the sense that elementary data items are clustered in records taking into account some database performance criteria (e.g., minimality of the number of logical record accesses, redundancy, reduction of updating anomalies). Finally, the physical design phase improves the overall performance by defining access paths, storage allocation of records, device media allocation, etc.

In this paper we are concerned with the database scheme equivalence problem which is important during the phases of conceptual design and of implementation. Our goal is to give an algorithm for testing whether two database schemes are equivalent (i.e., they represent the same information) and to provide a set of elementary operators which allow to modify a conceptual scheme in all possible ways without changing the semantic properties of it (i.e., by preserving the equivalence). Such formal tools may be used both to improve the expressive power of a conceptual scheme and to eventually map it into a DBMS-processable scheme by redefining entities in terms of records according to some performance criteria.

We use the Entity-Relationship Model [5] at the conceptual level and the Network Model [2,6] and the relational model [7] at the implementation level. We introduce a graph formalism (JFD-graphs) to represent Entity-Relationship schemes, and we define the equivalence of two Entity-Relationship schemes under the universal relation assumption

This work has been supported by CASMEZ, CNR and MPI.

[8], by assuming that every database scheme can be described by a universal relation, a full join dependency and a set of functional dependencies. This assumption and the graph formalism provide a suitable environment in which the equivalence of database schemes can be formally treated.

In Section 2 we give the basic definitions on the Entity-Relationship Model and in Section 3 we introduce the graph formalism of JFD-graphs to represent Entity-Relationship schemes. In Section 4 we study the problem of deciding whether two Entity-Relationship schemes are equivalent by introducing the concept of JFD-graph closure and we present an efficient algorithm to test the equivalence. In Section 5 we provide a set of manipulation operators on JFD-graphs and we prove that this set of operators is sound (i.e., preserve the equivalence), complete (i.e., all equivalent representations can be obtained by repeatedly using the operators) and independent (i.e., no proper subset of it is complete). Finally, in Section 6, we sketch a methodology to map JFD-graphs into relational and network schemes.

2.THE ENTITY-RELATIONSHIP CONCEPTUAL SCHEME

The past few years have seen a proliferation of data models as possible candidates for the conceptual description of a database [13,15]. Among them the Entity-Relationship model incorporates several suitable features such as an easy formalism to describe the semantics of the real world, based on a coincise diagrammatic technique.

Two levels of aggregate data exist in such a model: entity sets (or simply entities) composed by instances of the same type, and relationships-sets (or simply relationships) which describe meaningful associations among entities, that perform different roles in the relationship. Attributes are also defined as functions that associate to every entity (relationship)-instances a value belonging to a corresponding domain. Since attributes are properties which only hold within the objects in which they are defined, without loss of generality, we shall omit attributes in the model.

Furthermore, an additional set of integrity constraints may be specified on entities and relationships such as the minimal and the maximal number of instances of a relationship that may contain the same instance of an entity, or more general cardinality constraints like functional dependencies [16]. We point out that functional dependencies are defined in the relational model, and their interpretation in the Entity-Relationship model may be obtained by considering a relationship as a relation whose tuples correspond to the relationship instances and whose attributes correspond to role names in the relationship.

Formally an Entity-Relationship scheme is described by a bipartite graph and a set of functional dependencies as follows.

Definition 1. An ER-scheme is a pair $\langle G,F \rangle$, where:

a- $G = \langle E,R,A,ent,rel \rangle$ is a bipartite multigraph with set of nodes $E \cup R$ and set of arcs A, such that:

i) E is the set of entities and R is the set of relationships

ii) A is the set of roles

iii) $rel:A \rightarrow R$ and $ent:A \rightarrow E$ are two functions which specify, respectively, the relationship of every role and the entity playing this role in the relationship. We denote by $e(R_i)$ the set of all entities which have a role in the relationship R_i.

b- F is a set of "intra-relationship" functional dependencies over A, i.e., $X \rightarrow Y$ in F is an ordered pair of non-empty subsets of A such that there exists a relationship R_i in R for which for all k in $X \cup Y$, we have $rel(k)=R_i$. ■

Example 1. In Fig. 1 we show a simple ER-scheme. ■

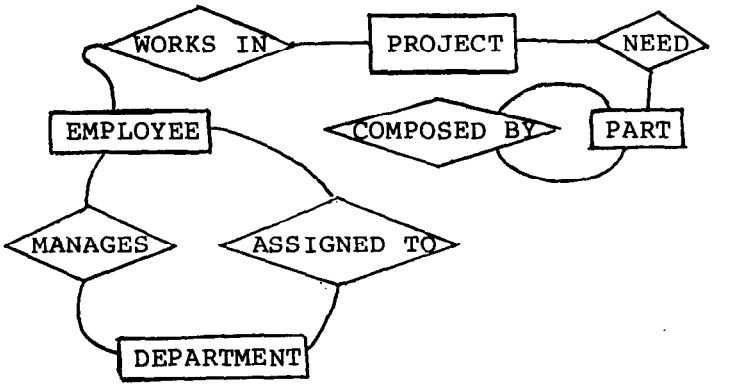

Figure 1. An Entity-Relationship scheme.

3. GRAPH REPRESENTATION OF ENTITY-RELATIONSHIP SCHEMES

Several authors have used graph or hypergraph formalisms to model the set of data dependencies contained in a relational database scheme. In [1] a particular kind of directed graphs (called FD-graphs) has been introduced to represent and manipulate a set of functional dependencies. In this paper, we extend this formalism to represent an ER-scheme by introducing JFD-graphs. JFD-graphs are an extension of the usual concept of graph such that properties of ER-schemes can be formally characterized in terms of graph properties.

Definition 2. A JFD-graph is a directed graph $G=<N_S,N_C,N_J,A_F,A_D>$ where:

i) $N = N_S \cup N_C$ is the set of nodes, where N_S and N_C are the disjoint sets of simple and compound nodes, respectively.

ii) $A = A_F \cup A_D$ is the set of arcs, where A_F contained in $N \times N_S$ is the set of full arcs and A_D contained in $N_C \times N_S$ is the disjoint set of dotted arcs

iii) there exists a function $c:N \rightarrow P(N)$ such that

a) for each i in N_C, $c(i)=\{m \mid (i,m)$ is in $A_D\}$ and $|c(i)| \geq 2$, and

b) for each i in N_S, $c(i) = \{i\}$.

The nodes c(i) are called component nodes of i.

iv) the subset N_J of N is the set of join nodes such that:

a) for each (i,j) in A_F, there exists a node k in N_J for which $c(i) \cup \{j\}$ is contained in c(k) (independency property) and

b) for each i in N_S, there exists a node j in N_J such that i is in c(j). ■

Example 2. The ER-scheme of Fig.2a is represented by the JFD-graph $G=<N_S,N_C,N_J,A_F,A_D>$ in Fig.2b, where:

$N_S=\{A,B,C,D\}$, $N_C=\{AB,BC,BCD\}$,

$N_J=\{AB,BCD\}$,

$A_F=\{(A,B),(BC,D)\}$, $A_D=\{(AB,A),(AB,B),(BCD,B),(BCD,C),(BCD,D),(BC,B),(BC,C)\}$;
and circles denotes join nodes.

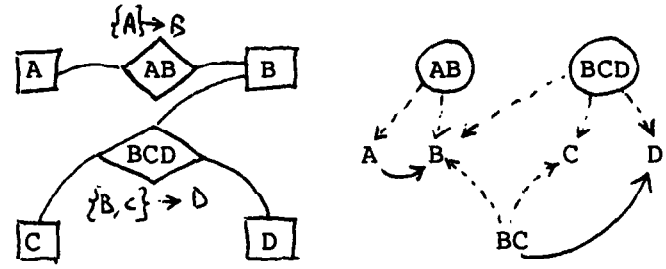

Figure 2. a) An ER-scheme and
b) its JFD-graph.

It is easy to see that JFD-graphs can be used to represent ER-schemes. In fact, each simple node corresponds to a distinct entity, each compound join node i corresponds to a distinct relationships and the component nodes in c(i) correspond to entities involved in the relationship. Since a JFD-graph is not a multigraph, entities may have at most one role in a relationship. Finally, given a full arc (i,j) and any join node k such that both c(i) and c(j) is contained in c(k), the arc (i,j) corresponds to a functional dependency X->Y in the relationship corresponding to k where X are the roles of the entities corresponding to the nodes in c(i) and Y is the role of the entity corresponding to the node j. It follows that a full arc generates a functional dependency which may hold in several distinct relationships.

From the above interpretation, it follows that an ER-scheme S can be represented by a JFD-graph if and only if the following conditions are satisfied:

a- there is no recursive relationship in S, that is a relationship involving an entity with more than one role (e.g., the relationship COMPOSED BY over the entity PART in the scheme of Fig.1 is recursive)

b- there is no local functional dependency, that is a functional dependency X->Y such that X->Y holds in a relationship R_i and there exists

another relationship R_j for which $ent(X \cup Y)$ is contained in $e(R_j)$ and no functional dependency $X' \rightarrow Y'$ holds in R_j, where $ent(X)=ent(X')$ and $ent(Y)=ent(Y')$.

In addition, we give a particular interpretation of JFD-graphs in which simple nodes correspond to the attributes of a universal relation U, join nodes correspond to the relations of a full join dependency on U and full arcs correspond to functional dependencies on U (for an introduction to data dependency theory, see [16]).

Definition 3. Given a JFD-graph $G=<N_S,N_C,N_J,A_F,A_D>$, we associate to G a relational database scheme $D=<U,J,F>$, where U is a universal relation scheme whose attributes are the nodes in N_S, J is a join dependency $*[X_1,\ldots,X_{|N_J|}]$ (under this assumption , we shall not consider JFD-graphs having two or more distinct compound nodes with exactly the same set of component nodes) such that for each i in N_J, $X_i=c(i)$, and F is a set of functional dependencies such that $F=\{X \rightarrow Y |$ there exists (i,j) in A_F such that $X=c(i)$ and $Y=c(j)\}$. A legal instance of D is a relation u over U which satisfies the join dependency J and the set of functional dependencies F. ■

This interpretation of JFD-graphs is based on the conjecture that every plausible real world can be described by a universal relation, a set of functional dependencies and a full join dependency [8]. Even though this assumption has been sometime criticized as unrealistic, our belief is that the assumption is indeed reasonable in most cases; besides, it provides a formal environment where the database equivalence and mapping problems can be approached.

From now on, we shall consider ER-schemes which can be represented by JFD-graphs having the above simplifying interpretation. This means that we shall not deal with ER-schemes which, for example, have local functional dependencies, recursive relationships or, more in general, relationships sharing the same set of entities with different meaning (e.g., the relationships ASSIGNED_TO and MANAGES between the entities EMPLOYEE and DEPARTMENT).

Despite the above restrictions, we conjecture that any ER-scheme can be "approximated" by JFD-graphs by performing simple modifications (e.g., by duplicating entities). We note that a similar approximation is used in [11].

Example 2. the relational database scheme associated with the JFD-graph in Fig.2a is $D=<U,J,F>$, where $U=\{A,B,C,D\}$, $J=*[\{A,B\}, \{B,C,D\}]$ and $F=\{\{A\}\rightarrow\{B\}, \{B,C\}\rightarrow\{D\}\}$. ■

4. EQUIVALENCE OF JFD-GRAPHS

We now give a definition of equivalence between two JFD-graphs, based on the equivalence of the corresponding sets of data dependencies (join dependency and functional dependencies).

Definition 4. Let G_1 and G_2 be two JFD-graphs with the same set of simple nodes and let $D_1=<U,J_1,F_1>$ and $D_2=<U,J_2,F_2>$ be the relational database schemes corresponding to G_1 and G_2, respectively. We say that G_1 is contained in G_2 ($G_1 \sqsubseteq G_2$) if the set of data dependencies J_1 and F_1 implies J_2 and F_2, i.e., every legal instance of D_1 is a legal instance of D_2. Furthermore G_1 is equivalent to G_2 ($G_1 \equiv G_2$) if both $G_1 \sqsubseteq G_2$ and $G_2 \sqsubseteq G_1$. ■

In order to check whether two JFD-graphs G_1 and G_2 are equivalent, we can use the following algorithm. By Definition 4 we only need to check whether $<J_1,F_1>$ implies $<J_2,F_2>$,and conversely. By definition of JFD-graph, all functional dependencies F_1 and F_2 are embedded in the components of J_1 and J_2, respectively. This means that all functional dependencies implied by $<J_1,F_1>$ ($<J_2,F_2>$) are those implied only by F_1 (F_2). Hence, testing the implication of functional dependencies can be done by the membership algorithm given in [2]. In addition, since J_1 preserves functional dependencies in the sense of [4], testing whether $<J_1,F_1>$ implies J_2 can be done in polynomial time by computing the chase C of the tableau associated to J_2, denoted $T(J_2)$, under F_1

and by checking whether there is a containment mapping from $T(J_1)$ to the chase C (see [4] for more details about this algorithm). We note that this approach has been used in [10] for testing the equivalence of ER-schemes.

In this section we shall present a simpler and more efficient algorithm for testing JFD-graph equivalence based on the concept of JFD-graph closure that is an extension of the FD-graph closure introduced in [1]. An other definition of closure of a hypergraph representing a relational scheme, that also includes inter-relational functional dependencies, is given in [13].

Let us first of all give some intermediate definitions and results.

Definition 5. Let $G=<N_S,N_C,N_J,A_F,A_D>$ and $G'=<N_S,N_C',N_J',A_F',A_D'>$ be two JFD-graphs with the same set of simple nodes. $G''=G+G'$ is the JFD-graph $G''=<N_S,N_C \cup N_C', N_J \cup N_J', A_F \cup A_F', A_D \cup A_D'>$. ■

Definition 6. Let G and G' be two JFD-graphs. We say that $G \leq G'$ if $G \equiv G'$ and G is a subgraph of G' (i.e., there exists a JFD-graph G'' such that $G'=G+G''$). ■

Theorem 1. Let G and G' be two JFD-graphs. If $G \equiv G'$, then $G'' \equiv G'$, where $G''=G+G'$.

Proof.(Sketch) By definition of JFD-graph equivalence, we have to prove that $G \sqsubseteq G''$ and $G'' \sqsubseteq G$, i.e., <J,F> implies <J'',F''> and <J'',F''> implies <J,F>, where <J,F> and <J'',F''> are the data dependencies corresponding to G and G'', respectively. We have that F'' implies F since F is contained in F'', and F implies F'' since F''=F F' and F implies F' by hypothesis. In addition, J implies J'' since, by costruction of G'', for each component R_i in J, there exists a component R_j'' in J'' such that $R_i=R_j''$.

Let us now prove that <J'',F''> implies J. To this end, since the functional dependencies F'' are embedded in J'', and, then, J'' preserves F'', we have to show that there exists a containement mapping from T(J'') to the chase C'' of T(J) under F'' (see Theorem 4 in [4]). Since obviously there is a containement mapping from T(J) to C'' (by definition of C'') and from T(J') to C'' (because <J',F'> implies J and C'' is also the chase of T(J) under F'), it follows that there is a containement mapping from T(J'') to C'', because T(J'') is the union of T(J) and T(J'). Hence, <J'',F''> implies J and this concludes the proof. ■

Let us now give the definition of JFD-closure.

Definition 7. Given a JFD-graph G, a JFD-graph G^+ is the <u>closure</u> of G if $G \leq G^+$ and for each G' such that $G \leq G'$, we have $G' \leq G^+$. ■

Remark. By Definition 6, G^+ is equivalent to G. ■

Proposition 1. The closure of a JFD-graph G always exists and is unique.

Proof. Let us consider the JFD-graph G^+ obtained by repeatedly adding to G all JFD-graphs G' such that $G \leq G'$. We have that G^+ is equivalent to G by Theorem 1, and for each G' such that $G \leq G'$, $G' \leq G^+$ by construction. Hence G^+ is the closure of G. The uniqueness of G^+ derives directly from the existence. ■

We now prove that all equivalent JFD-graphs have the same closure.

Theorem 2. Let G and G' be to JFD-graphs. G is equivalent to G' if and only if $G^+=G'^+$.

Proof. <u>If part</u>. If $G^+=G'^+$ then $G^+ \equiv G'^+$. Hence, since $G^+ \equiv G$ and $G'^+ \equiv G'$, also G is equivalent to G'.
<u>Only if part</u>. Since G is equivalent to G', $G''=G+G'$ is equivalent to G and G' by Theorem 1. Hence, since $G \leq G''$ and $G' \leq G''$, we have $G''^+=G^+$ and $G''^+=G'^+$ by definition of closure and, then, $G^+=G'^+$. ■

Since the size of the closure of a JFD-graph G is in general exponential in the size of G, we show that the equivalence of two JFD-graphs G and G' can be checked by considering two suitable subgraphs of G^+ and G'^+.

Definition 8. Let G be a JFD-graph. A JFD-graph G' is a <u>covering</u> of G if $G' \leq G$ and for each join node i in G there exists a join node k in G' such that c(i) is contained in c(k). ■

We shall show that, given a JFD-graph G, we can find a covering of G^+ whose size is polynomially bounded in the size of G. To this end, we shall give the definition of JFD-path and node closure in a JFD-graph.

Definition 9. Given a JFD-graph $G=<N_S,N_C,N_J,A_F,A_D>$ and two nodes i,j in N, a (directed) JFD-path $<i,j>$ from i to j is a minimal subgraph $G'=<N_S',N_C',N_J',A_F',A_D'>$ such that i,j are in N' and either (i,j) is in $A_F' \cup A_D'$ or one of the following possibilities holds:

i) j is a simple node and there exists a node k such (k,j) is in $A_F' \cup A_D'$ and there is a JFD-path $<i,k>$ in G' (graph transitivity);

ii) j is a compound node with component nodes $m_1,..,m_r$ and there are r JFD-paths $<i,m_1>,..,<i,m_r>$ included in G' (graph union). ■

Fact 1.[1] There exists a JFD-path $<i,j>$ in a JFD-graph G if and only if I --> J is implied by the set of functional dependencies represented by G, where I and J are the sets of attributes represented respectively by i and j. ■

Definition 10. Let $G = <N_S,N_C,N_J,A_F,A_D>$ be a JFD-graph and i be a node of G. We call (node)-closure of i the set of simple nodes $i^+ = \{j \mid$ there is a JFD-path $<i,j>$ in G$\}$. ■

Fact 2.[1] The closure of a node i of a JFD-graph G can be computed in O(m) time, where m is the number of arcs in G (both dotted and full). ■

There is a strong relationship between the closure of a JFD-graph and the closure of its join nodes.

Proposition 3. Let G be a JFD-graph and G^+ be its closure. Then for each join node i in G there is a join node j in G^+ such that i^+ is contained in $c^+(j)$, and for each join node i in G^+, there exists a joinnode j in G such that $c^+(i)$ is contained in j^+.

Proof. The proof can be done by using the same arguments used in the proof of Theorem 1. ■

The concept of node closure can be used to contruct a covering of G^+ which is polynomially bounded in the size of G.

Theorem 3. Let G be a JFD-graph an G^+ be the closure of G. A covering of G^+ can be computed in $O(n_J \times m)$, where n_J and m are the number of join nodes and the number of arcs in G, respectivelly.

Proof. Let $G=<N_S,N_C,N_J,A_F,A_D>$ and let us consider the JFD-graph $G'=<N_S,N_C',N_J',A_F',A_D'>$ obtained from G by adding for each join node i in N_J a join node k (and the corresponding dotted arcs) such that $c(k)=i^+$ (if it does not already exists). By Proposition 3, G' is a covering of G^+. As far as the complexity of finding G' is concerned, we note that we only need to compute the closure of all join nodes in G. Hence, by Fact 2, G' can be found in $O(n_J \times m)$ time. ■

Theorem 4. Let G and G' be two JFD-graphs. The equivalence of G and G' can be checked in time $O((n+n')x(m+m'))$, where n and m (n' and m') are respectively the number of nodes and of arcs in G (G'), and n_j ($n_{j'}$) is the number of join nodes in G (G').

Proof. Let us consider the following algorithm. Let $\tilde{G}$ ($\tilde{G}'$) be the covering of G^+ (G'^+) obtained as in the proof of Theorem 3. A necessary condition for G and G' to be equivalent is that for each join node i in $\tilde{G}$, there exists a join node j in $\tilde{G}'$ such that c(i) is contained in c(j) and conversely (see Theorem 2 and the definition of covering). If this condition is satisfied, we can add to $\tilde{G}$ ($\tilde{G}'$) all compound nodes in G' (G) which are not already in $\tilde{G}$ ($\tilde{G}'$). Let $\hat{G}$ ($\hat{G}'$) be the JFD-graphs so obtained. Also $\hat{G}$ and $\hat{G}'$ are coverings of G^+ and G'^+, respectively, and they have the same set

of nodes. It follows that $G^+=G'^+$ if and only if for each node i, the node closure of i in $\hat{G}$ is equal to the node closure of i in $\hat{G}'$. Finding $\tilde{G}$ and $\tilde{G}'$ can be done in $O(n_j \times m + n_{j'} \times m')$ time (see Theorem 3), and the closure of all nodes in $\hat{G}$ and $\hat{G}'$ can be computed in $O((n + n') \times (m + m'))$. This concludes the proof. ■

5. MANIPULATION OF JFD-GRAPHS

In database design process, it is very important not only to check whether two schemes are equivalent but also to have simple rules which allow to modify a scheme in order to improve its expressive power (during the conceptual design phase) or its efficiency (during the implementation phase).

To this end, we now introduce a set of elementary operators for the manipulation of a JFD-graph $G=<N_S,N_C,N_J, A_F,A_D>$, which preserve the equivalence.

OPERATORS FOR JFD-GRAPH MODIFICATION

O1. add a full arc (i,j) if

a) j is a simple node, and

b) there exists a join node k such that $c(i) \cup \{j\}$ is contained in c(k), and

c) the closure of i does not change

O2. cancel a full arc (i,j) if the closure of i does not change

O3. add a non-join compound node i and all its outgoing dotted arcs to component nodes if there exists a join node j in G for which c(i) is contained in c(j)

O4. cancel a non-join compound node i and all its outgoing dotted arcs if i has no outgoing full arcs in G.

O5. add a join node i and its outgoing dotted arcs if there exists a join node j for which c(i) is contained in j^+.

O6. cancel a join node i and its outgoing dotted arcs if

- there exists a join nodej for which c(i) is contained in j^+, and
- for all non-join node k, there exists a join node $j \neq i$ for which c(k) is contained in c(j), and
- for all full arcs (k,m), there exists a join node $j \neq i$ for which $c(k) \cup \{m\}$ is contained in c(j)

We prove that the operators for JFD-graph modification enjoy the following properties:

a- soundness, i.e., all JFD-graphs obtained by repeatedly applying the operators are equivalent to the initial JFD-graph G;

b- completness, i.e, for each JFD-graphs G' equivalent to G, there exists a sequence of application of the operators which modifies G in G';

c- independency, i.e., no proper subset of the set of operators is complete.

Theorem 5. The set of operators O1-O6 is a sound, complete and independent set of modification rules for JFD-graphs.

Proof. Soundness and independence. Straightforward.

Completeness. (Sketch) Let G and G' be two equivalent JFD-graphs. We have to show that there exists a sequence of applications of the operators which modifies G in G'. Let us consider the JFD-graph $\tilde{G}$= G+G'. By Theorem 1, $G \leq \tilde{G}$ and $G' \leq \tilde{G}$. Hence $\tilde{G}$ is equivalent to both G and G' and both G and G' are subgraphs of $\tilde{G}$. It is easy to see that by repeatedly using the add operators (O1, O3 and O5) G can be trasformed in $\tilde{G}$, and $\tilde{G}$ can be trasformed to G' by repeatedlyapplying the cancel operators (O2, O4 and O6). ■

6. MAPPING JFD-GRAPHS TO DBMS-PROCESSABLE SCHEMATA

Given a JFD-graph G, there are simple rules which allow to obtain a DBMS-processable (either relational or network) database scheme which is equivalent to the scheme represented by G. The approach we propose here is straithforward in the sense that there is a direct correspondence between the objects in the JFD-graph and the aggregation structures (relations or records and sets) in the target model.

In the case that the target model is relational, the mapping rules are trivial, since we only need to associate a relation to every join node and to enforce the same data dependencies represented in G on the target relations. Notice that, in this case, simple join nodes in the JFD-graph are relevant since they specify relations corresponding to entities of the conceptual scheme.

Let us now consider the case of the network model. The problem of obtaining a network scheme starting from a join dependency (without functional dependencies) has been approached in [11,17]. We use the same approach starting from a JFD-graph $G=<N_S,N_C,N_J,A_F,A_D>$. Let us denote by R_i a record of the target network scheme and by $Attr(R_i)$ the set of attributes of R_i. We construct the network scheme as follows:

- for each node i in $N_S \cup N_J$, we introduce a record R_i such that $Attr(R_i)=c(i)$
- for each subset X of N_J such that $I\neq\emptyset$, where $I=_k \bigcap_{in\ X} c(k)$, we introduce a record R_i such that $Attr(R_i)=I$ (if there is no other record with the same set of attributes).
- we introduce a set between two records R_1 (owner) and R_2 (member) if both $Attr(R_1)$ is contained in $Attr(R_2)$ and there exists no record R_3 such that $Attr(R_1)$ is properly contained in $Attr(R_3)$ that is properly contained in $Attr(R_2)$.

In addition, we enforce functional dependencies within the records corresponding to join nodes.

Let us now give a methodology for mapping an Entity-Relationship conceptual scheme into a relational or network scheme:

- represent the real world of interest by means of an ER-scheme
- represent the ER-scheme by a JFD-graph (possibly by restructuring the scheme in order to meet the basic assumptions of JFD-graphs)
- repeat
  - map the JFD-graph into the target relational or network scheme (according to the above mapping rules)
  - evaluate the performance of the so-obtained scheme (on the basis of required criteria by considering quantitative data on transactions running against the scheme)
  - if the performances are not satisfactory then
    - modify the JFD-graph using the modification operators introduced in the previous section on the basis of a suitable heuristic strategy for searching in the space of possible alternate solutions
- until the performance requirements are satisfied.

REFERENCES

[1] Ausiello, G., A.D'Atri and D.Sacca', "Graph Algorithms for Functional Dependency Manipulation", Journal of ACM 30:4, (Oct. 1983), 752-766.

[2] Bachman, C.W., "Data Structure Diagrams", Data Base 1:2, (1969), 4-10.

[3] Beeri, C. and P. A. Bernstein, "Computational Problems Related to the Design of Normal Form Relational Schemas", ACM Transaction on Database Systems 4:1, (March 1979), 30-59.

[4] Beeri, C., A.O.Mendelzon, Y.Sagiv and J. D. Ullman, "Equivalence of Relational Database Schemes", SIAM Journal on Computing 10:2, (May 1981), 352-370.

[5] Chen, P.P.,"The Entity-Relationship Model: Toward a Unified View of Data", ACM Transaction on Database Systems 1:1, (1979), 9-36.

[6] CODASYL Data Base Task Group,"April 1971 Report"; ACM, New York, (1971)

[7] Codd, E.F.,"A Relational Model for Large Shared Data Banks", Communication of ACM 13:6, (June 1970), 377-387.

[8] Fagin, R., A. O. Mendelzon and J.D.Ullman, "A Simplified Universal Relation Assumption and its Properties", ACM Transactions on Database Systems 7:3, (Sept.1982), 343-360.

[9] "ISO TC97/SC5/W43, report on concepts and terminology for the conceptual schema and the information base",van Griethuyesen J.J. (ed.),ISO/TC97/SC5 n. 695, (March 1982).

[10] Jajodia, S., P. A. Ng and F.N.Springsteel, "The Problem of Equivalence for Entity-Relationship Diagrams", IEEE Transaction on Software Engineering SE-9:5, (Sept.1983), 617-629.

[11] Lien, Y.E.,"On the Equivalence of Database Models", Journal of ACM 29:2, (April 1982), 333-363.

[12] Lum, V.Y. et al., "1978 New Orleans Database Design Workshop Report", Proc. 5th Conf. on Very Large Data Bases, Rio de Janeiro, Brazil, (Oct.1979), 328-339.

[13] Sacca', D., "Closures of Database Hypergraphs", IBM Research Rept. RJ3723, San Jose, California, (Dec.1982).

[14] Teory, T. and J.Fry, Design of Database Structures, Prentice Hall, Englewood Cliffs, N.J., (1982).

[15] Tsichritzis, D.C. and F.H.Lochovsky, Data Models, Prentice Hall, Englewood Cliffs, N.J., (1982).

[16] Ullman, J.D., Principles of Database Systems, second edition, Computer Science Press, Potomac, Maryland, (1982).

[17] Yannakakis, M., "Algorithms for Acyclic Database Schemes", Proceedings 7th Conference on Very Large Data Bases, Cannes, France, (Sept.1981), 82-94.