Title

Abbreviated Query Interpretation in Extended Entity-Relationship Oriented Databases

Authors

Markowitz, V M

Shoshani, A

Publication Date

1989-08-01

LBL-25337

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA, BERKELEY

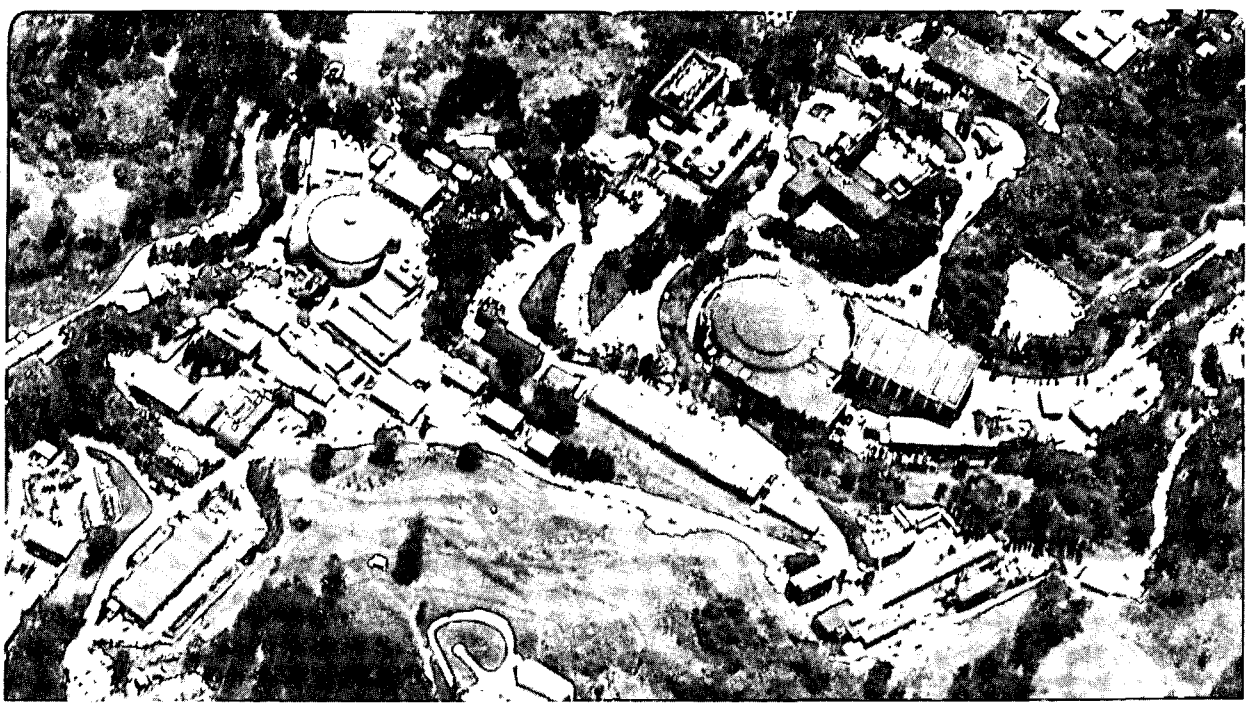
Information and Computing Sciences Division

To be presented at the Eighth International Conference on Entity-Relationship Approach, Toronto, Canada, October 18–20, 1989, and to be published in the Proceedings

Abbreviated Query Interpretation in Extended Entity-Relationship Oriented Databases

V.M. Markowitz and A. Shoshani

August 1989



Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098.

DISCLAIMER

LBL-25337

ABBREVIATED QUERY INTERPRETATION IN EXTENDED ENTITY-RELATIONSHIP ORIENTED DATABASES

Victor M. Markowitz and Arie Shoshani

Computing Science Research & Development
Information & Computing Sciences Division
Lawrence Berkeley Laboratory
1 Cyclotron Road
Berkeley, California 94720

August 1989

To appear in Proceedings of the 8th International Conference on Entity-Relationship Approach, Toronto, Canada, October 18-20, 1989.

This work was supported by the Office of Health and Environmental Research Program and the Director, Office of Energy Research, Applied Mathematical Sciences Research Program, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

ABBREVIATED QUERY INTERPRETATION IN EXTENDED ENTITY–RELATIONSHIP ORIENTED DATABASES *

Victor M. Markowitz and Arie Shoshani

Computer Science Research Department
Information and Computing Sciences Division
Lawrence Berkeley Laboratory
1 Cyclotron Road, Berkeley, CA 94720

In order to express database queries, users are often required to understand large, complex database structures. It is important to relax this requirement by allowing users to express concise (*abbreviated*) queries, so that they can manage with partial, or even no knowledge of the database structure. Abbreviated queries involve only the specification of the objects that are relevant to the users. The main problem with abbreviated queries is to derive the corresponding full queries. In this paper we present a methodology to support the expression of abbreviated queries in *Extended Entity-Relationship* (EER) oriented relational databases, that is, relational databases whose schemas are translations of EER schemas. In EER-oriented databases, abbreviated queries can be represented by disconnected subgraphs of the corresponding EER diagrams. We propose a criterion for determining the connected subgraphs corresponding to such abbreviated queries. Our criterion is based on the *robustness* of the relational *view* corresponding to a connected subgraph of an EER diagram. View robustness expresses the capability of performing *side-effect free* updates via the view. We also investigate techniques that take advantage of specific properties of EER diagrams in order to reduce the complexity of finding the subgraphs that correspond to a given abbreviated query.

1. INTRODUCTION

As database systems grow in complexity, it becomes increasingly difficult for users to understand and remember the database schema, and to formulate queries. The capability to write concise queries is a desirable goal, and is usually supported by special-purpose interfaces, such as *Universal–Relation* (UR) interfaces. Such interfaces attempt to free the user from specifying unnecessary details: users pose *abbreviated* queries, mentioning only the objects they are interested in, and the interface is able to *interpret* the query, that is, to find the necessary

* Presented at the *8th International Conference on Entity-Relationship Approach*, 18-20 October 1989, Toronto, Canada. This work was supported by the Office of Health and Environmental Research Program and the Applied Mathematical Sciences Research Program, of the Office of Energy Research, U.S. Department of Energy, under Contract DE-AC03-76SF00098.

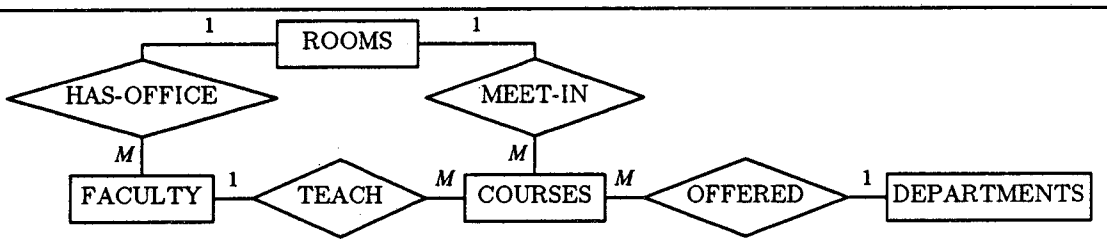
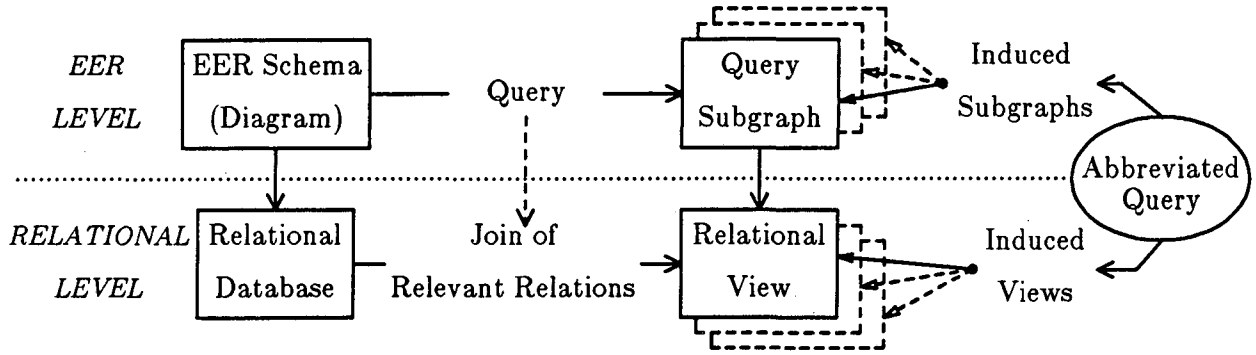


Figure 1. An Entity-Relationship Structure.

connections between the objects. Consider the following example. Suppose that DEPARTMENTS offer COURSES, COURSES are taught by FACULTY members and meet in certain ROOMS, and FACULTY members have offices in certain ROOMS, as represented by the entity-relationship structure of figure 1. A user who does not know, or remember, the details of such a structure may want to ask the query "how many FACULTY members over 40 work in the Computer Science DEPARTMENT". One would like an intelligent system to generate an equivalent full query that includes all the missing intermediate elements (for this example, TEACH, COURSES, and OFFERED) as well as the necessary expressions connecting them.

In section 2 we outline a methodology to support the expression of abbreviated queries in relational databases. Responding to relational queries usually requires combining information from multiple relations by *joins*. Determining what relations should be joined in response to an abbreviated relational query can be based on various forms of structural information, such as dependencies, relation connectivity implied by attribute names, etc. Our approach is based on the association of relational databases with *Extended Entity-Relationship* (EER) schemas, which requires the use of EER schemas in the design of relational databases. The structural richness and clarity of EER schemas makes them more appropriate to use as a source of information on various connections in the database. We take advantage of the graph representation of EER schemas, the EER diagrams. Thus, queries that are expressed over EER diagrams can be represented by connected subgraphs of EER diagrams, while abbreviated queries can be represented by disconnected subgraphs. The interpretation of abbreviated queries reduces to the problem of finding the most desirable connected subgraph (out of multiple possible subgraphs) that covers (spans) a given disconnected subgraph. Our approach can be summarized by the following diagram:



Given an EER schema (diagram), a mapping associates it with an EER-oriented relational database. Every query is represented by a subgraph of the EER diagram, called *query—subgraph*, which is associated with a relational view generated by joining the relevant relations in the corresponding relational database. For each abbreviated query there are one or more induced query-subgraphs and their associated views. We need to find the appropriate induced query-subgraphs and rank their plausibility in order to choose the query-subgraph with the highest plausibility. Our criterion for the most desirable query-subgraph is the *robustness* of the corresponding view, which is based on the capability of translating *view—updates* (i.e. updates expressed over views) into *base—updates* (i.e. updates expressed over base relations) so that the base-updates have no effect on views beyond the specified view-updates (are *side-effect free*). It is important to point out that there is no way of actually determining the query-subgraph that a user may prefer, because each subgraph has its own semantics, and the semantic intentions of the user are unknown to the system. What we provide is information for the user as to the robustness of his view; without any guidance from the user concerning the desired query-subgraph, it is reasonable to choose the query-subgraph with the highest robustness. In general, a system that supports abbreviated queries should not be limited to retrievals only, but should support updates as well. Accordingly, when an interpretation to an abbreviated query is selected the view robustness criterion indicates to the user what are the consequences of such a selection relative to the capability of performing view-updates.

In section 2, we also contrast our approach with the *Universal—Relation* (*UR*) approach. We discuss the restrictions imposed by the UR approach which make it impractical in the context of abbreviated query interpretation. Section 3 contains a brief review of the EER model and of the association of EER schemas with relational databases. In section 4 we introduce the concept of query-subgraph and develop the view robustness criterion employed by our methodology. In section 5 we examine techniques that take advantage of specific properties of EER diagrams in order to reduce the complexity of finding the query-subgraphs that correspond to a given

abbreviated query. We close the paper by summarizing the results and drawing some conclusions. We use in this paper some graph-theoretical concepts. Any textbook on graph theory (e.g. [5]) can provide the necessary background. Details concerning the mapping of EER diagrams into relational schemas are given in appendix A, and the formal aspects of the view robustness criterion are condensed in appendix B.

2. ABBREVIATED QUERY INTERPRETATION

We begin this section by discussing the inadequacy of the Universal-Relation (UR) approach to support the expression of abbreviated queries in relational databases. Then we explain our reason for choosing the Extended Entity-Relationship framework for abbreviated queries. Finally, we outline the criterion we use for evaluating the plausibility of the possible interpretations for abbreviated queries.

2.1 Inadequacy of the Universal Relation Approach.

The best known methodologies for interpreting abbreviated *relational* queries are based on the *Universal–Relation* (*UR*) approach to the relational model (a survey is provided by [12]). UR schemas consist of unnamed sets of attributes, and the user can express queries by specifying the attributes of interest only, leaving the *interpretation* of the query, that is, the task of finding the relations to be joined, to the UR system. The UR approach generated many works that suggest ways of interpreting such *abbreviated* queries. The main problem of this approach is that it imposes certain design restrictions that result either in a limited modeling capability or in embedding the access paths into the attribute names. These restrictions are discussed below.

The UR approach is based on several assumptions which restrict the association of attributes with relation-schemes. In particular, the *Unique Role Assumption* (URA) [12] requires that any set of attributes can be associated by *at most one relationship*. In practical terms, this means that any two, or more, attributes must represent the same set of objects in every relation-scheme in which they appear together. For example, the relational schema of figure 2(ii) cannot represent the ER schema of figure 2(i) because the association of attribute set {CN,DN} with relations OFFER and TEACH implies under URA that the corresponding relationship-sets are not independent as represented by the ER schema of figure 2(i) (see [16] for details). The URA restriction can be overcome by renaming attributes, but attribute renaming leads to the proliferation of attribute names and the loss of semantic information [11]. The loss of semantic information can be resolved by keeping track of the attribute renamings [11]. However, as noted in

(ii) FACULTY (FN)
DEPARTMENT (DN)
COURSE (CN)
TEACH (FN, CN, DN)
OFFER (CN, DN)

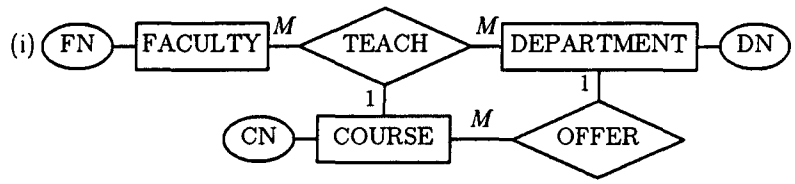


Figure 2. Relational Representation for an Entity-Relationship Schema.

[1], attribute renaming implies the embedding of access paths names, which the user supposedly does not know, into the attribute names. Thus, the structural semantics of the database is loaded onto the attributes alone. Consequently, the attribute naming becomes a critical problem, and the requirement to know and understand the semantic structure is replaced, for the user, by the requirement to understand and manage a names structure. This make UR databases very difficult to design and comprehend. Moreover, the original goal of providing the users with logical (i.e. structural) independence [12] cannot be achieved when renaming is necessary. A possible solution to this problem is discussed below in section 2.2.

2.2 Extended Entity–Relationship Query Interfaces.

A more appropriate framework for expressing queries is offered by the *Entity–Relationship* (ER) model [2]. The main reason is that the ER model, like any other semantic model, provides a higher degree of logical independence than the relational model [6]. Thus, since in ER schemas the associations between entities are represented explicitly by relationships, queries based on the ER model can avoid the explicit expression of connections by joins [6]. We refer in this paper to a version of the *extended ER (EER)* model [18] which includes, in addition to the constructs of the basic ER model of [2], the generalization and full aggregation (allowing the involvement of relationships in other relationships) constructs.

Queries over an EER schema, which we shall call *EER queries*, can be viewed as selecting a subgraph of the EER diagram representing the EER schema [9]. Thus, an *abbreviated* EER query can be represented by disconnected, rather than connected, subgraphs. This can be viewed as a shortcut to the specification of a full query, which obviously simplifies the user's task of expressing queries. Thus, in the example mentioned in the introduction, namely "how many FACULTY members over 40 work in the Computer Science DEPARTMENT", one could take advantage of the information in the EER diagram (see figure 1) and generate the subgraph that includes the

intermediate entity-set COURSES as well as the relationships connecting them. We use the term *query—subgraph* to denote a connected subgraph of the EER diagram that represents a full (not abbreviated) EER query.

Relational databases associated with EER schemas are said to be *EER—oriented*. In this context a query-subgraph of the EER diagram represents all the joins necessary to evaluate the query in the underlying EER-oriented database. Conversely, given an abbreviated query over an EER-oriented database, the query can be interpreted by finding the corresponding subgraph in the EER diagram. In general, every abbreviated EER query may be associated with several query-subgraphs, depending on the complexity of the EER diagram. Naturally, we would like to select the *most plausible* query-subgraph. The criteria for plausibility we are proposing here is discussed in section 2.3 below.

Putting ER interfaces on top of relational databases for the purpose of abbreviated query expression and interpretation has been considered by several authors (e.g. [17], [21], [22]). Their goal, however, was to apply the UR approach for interpreting abbreviated queries expressed over ER interfaces. Obviously, what is needed in such a case is a translation of ER schemas into relational schemas that would take into account the UR assumptions. We have examined the implications of such a translation in [16] and found that most translations of ER schemas into relational schemas proposed in the literature do not satisfy the UR assumptions. Interestingly, [17] and [22] are not concerned with the details of such a translation. However, following the formalism developed in [16] it is not difficult to see that the translations they use do not satisfy the UR assumptions and therefore their approaches are imprecise. For example, following [22] (which employs the translation of [20]), the relational schema of figure 3(ii) represents the ER diagram of figure 3(i). Assuming that this representation is correct, an ER abbreviated query that refers to entity-sets DEPARTMENT and COURSE corresponds to a relational abbreviated query that refers to attributes DN and CN. Following [13] (the UR approach adopted by [22]), the interpretation of this relational query is the projection on DN and CN of the union of two joins, of relations DEPARTMENT, WORK, FACULTY, TEACH, and COURSE, and of relations DEPARTMENT, ENROLL, STUDENT, TAKE, and COURSE, respectively. The union here is used because the two joins are supposed to produce tuples that represent relationships that have the same meaning, or in the words of [14], tuples of the *same flavor*. This condition, which the relational schema of figure 3(ii) satisfies, is expressed by the *One Flavor Assumption (OFA)* [14]. However, since distinct paths between two object-sets represent associations which have distinct meanings [9], the relational schema of figure 3(ii) under OFA distorts the semantics of the ER structure represented in figure

(ii)

DEPARTMENT	(DN)
FACULTY	(FN)
COURSE	(CN)
STUDENT	(SN)
WORK	(FN, DN)
TEACH	(CN, FN)
TAKE	(CN, SN)
ENROLL	(SN, DN)

(i)

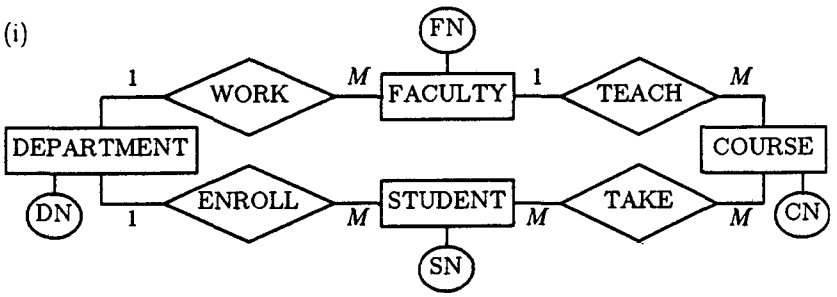


Figure 3. Relational Representation for an Entity-Relationship Schema (*keys are underlined*).

3(i). Consequently, the interpretation of the abbreviated relational query above is not valid for the ER query above. A methodology for applying the UR approach in the context of EER-oriented relational databases is proposed in [15].

2.3 A Plausibility Criterion for Interpreting Abbreviated Queries.

The criterion used by most UR methodologies for query interpretation is to select relations that can be joined *losslessly* [12]. This approach is based on the conjecture that a join makes sense only if it is lossless [14]. Losslessness ensures that the view that is constructed by joining the selected relations does not contain tuples that cannot be in the *universal relation* which is presumed to be (the view) referred by the user. We propose below a criterion which, like losslessness, is based on an assumption concerning the view referred by the user. However, unlike losslessness, our criterion is not based on UR assumptions, so that we can avoid dealing with their complex implications.

When a user with no knowledge of the database structure poses an abbreviated query, he refers to some *view* that he assumes to be represented in the database. Accordingly, he would expect his view to behave as any base relation in the database. The major problem with views is the difficulty of mapping *view–updates* (that is, updates expressed over views) into updates of base relations. This problem has been studied extensively (e.g. [4], [7]). *Side-effect freeness*, meaning that the underlying updates must produce no effects beyond the intended update on the view, is generally accepted as ensuring the correctness of the view-update mapping. We propose the existence of a side-effect free view-update mapping as the selection criterion for query-subgraphs. We grade the *view–robustness* of the query-subgraphs as follows: the query-subgraph

is *robust* (rank 2) if it corresponds to a view that has a side-effect free update (delete and insert) mapping; *weakly robust* (rank 1) if it corresponds to a view that has a side-effect free delete mapping but no side-effect free insert mapping; and *non robust* (rank 0) if it corresponds to a view that has no side-effect free update mapping. Interestingly, it can be shown that for UR databases involving only functional dependencies, join losslessness is equivalent to the existence of a side-effect free *delete* mapping, that is, to view-robustness of rank 1. By using several ranks of view-robustness, our methodology determines a desirability factor for the possible subgraphs which can be associated with an abbreviated query. Consider the ER abbreviated query mentioned in the introduction, which refers to entity-sets FACULTY and DEPARTMENT of the ER diagram in figure 1. This query can be associated with two query-subgraphs, the subgraph induced by vertices FACULTY, TEACH, COURSES, OFFERED, and DEPARTMENTS, and the subgraph induced by vertices FACULTY, HAS–OFFICE, ROOM, MEET–IN, COURSES, OFFERED, and DEPARTMENTS, respectively. Following our selection criterion, the first subgraph has view-robustness 1, while the second subgraph has view-robustness 0.

In general, there is no way of selecting the *most reasonable* query-subgraph for a user whose semantic intentions are unknown. Our criterion only offers guidance by finding the subgraph corresponding to the most robust view, since in general one would expect his view to behave exactly as a base relation. Moreover, it stands to reason that if abbreviated retrievals are useful to a user, then abbreviated updates are useful too. It is in this context that the view-robustness criterion is particularly attractive. When an interpretation to a query is selected, the user is made aware of the consequences of such a selection relative to the capability of performing update operations. Consequently, even if we wish to use other heuristic criteria which may reflect the user's preference, we should include view-robustness as part of any strategy for selecting query-subgraphs. Such strategies are further discussed in section 4.

3. EXTENDED ENTITY–RELATIONSHIP ORIENTED DATABASES

3.1 The Extended Entity–Relationship Model.

The concepts of the basic *Entity-Relationship (ER)* model, (*entity*, *relationship*, *entity–set*, *relationship–set*, *value–set*, *attribute*, *entity–identifier*, *weak* entity-set, relationship *cardinality*, *role*) have been defined originally in [2] and have been repeatedly reviewed since then (e.g. see [18]). Entities and relationships are commonly called *objects*. For the sake of brevity, we omit the definitions of these concepts. Unlike the basic ER model of [2] the *extended ER* (*EER*) model we consider in this paper has two additional abstraction capabilities, generalization and full

aggregation, which are briefly reviewed below.

Generalization is an abstraction mechanism that views a set of entity-sets as a single *generic* entity-set. The inverse of generalization is called *specialization*. A specialization entity-set *inherits* all the attributes of any of its generic entity-sets, including the entity-identifier. An entity-set which is not specified as the specialization of any other entity-set is called a *generalization—source*. For the sake of simplicity we do not distinguish in this paper between different kinds of generalization such as those discussed in [6] and [18].

In the basic ER model the *aggregation* construct takes three forms: (i) the aggregation of a collection of attributes into an entity-set; (ii) the aggregation of a collection of attributes and the entity-identifiers of several existing entity-sets into a weak entity-set; and (iii) the aggregation of two or more entity-sets into a relationship-set. The basic ER model falls short of providing the full capability of aggregation by disallowing relationship-sets to be aggregated further. What is needed in order to provide this capability is simply to allow relationship-sets to associate any object-set, rather than only entity-sets.

An EER schema can be represented in a diagrammatic form called an *EER diagram* (EERD). An EER diagram is defined as a *directed graph* of the form $G_{ER} = (V, H)$, where V denotes the set of vertices and H denotes the set of directed edges. Our reason for preferring a directed, rather than the usual undirected, graph notation for EER diagrams is outlined below. The vertices of an EER diagram represent entity-sets, relationship-sets, and attributes. Entity-sets, relationship-sets, and attributes, are represented graphically by rectangles, diamonds, and ellipses, respectively. Every vertex is labeled by the name of the represented object-set or attribute. The directionality of edges allows the explicit representation not only of the interaction of the various object-sets, but also of their mutual *existence dependencies*. Thus, in an EER diagram there are directed edges (i) from relationship-sets to the object-sets they associate,

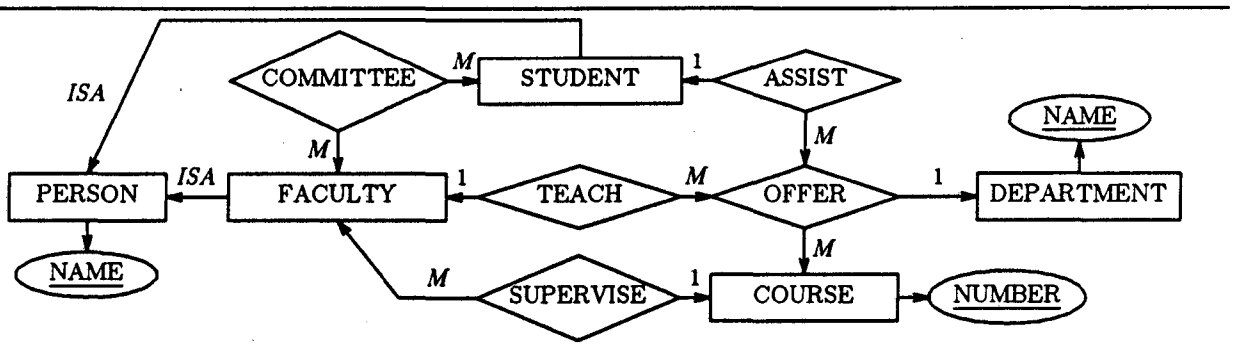


Figure 4. Extended Entity-Relationship Diagram Example (*identifiers are underlined*).

labeled by the corresponding cardinality, which is either 1 (*one*) or M (*many*), (ii) from weak entity-sets to the entity-sets on which they depend for identification, labeled *ID* ; (iii) from specialization entity-sets to the corresponding generic entity-sets, labeled *ISA* ; and (iv) from object-sets to their associated attributes. Note that the directionality of the edges is essential for the unambiguous representation of the generalization and aggregation constructs. An example of an EER diagram is shown in figure 4.

3.2 EER–Oriented Relational Databases.

Relational databases associated with EER schemas are called *EER–oriented* databases. Relational schemas of EER-oriented databases are of the form $(R, F \cup I)$, where R denotes a set of relation-schemes, and F and I denote sets of functional and inclusion dependencies, respectively. Informally, object-sets are represented by relation-schemes, connections of object-sets are represented by inclusion dependencies, and entity-identifiers and relationship cardinalities are represented by functional (key) dependencies. As an example, we present in figure 5 the relational schema corresponding to the EER schema of figure 4. The translation of EER schemas into relational schemas and the necessary relational concepts are outlined in appendix A. Details concerning the relational representation of EER structures can be found in [16].

Relation-Schemes (*underlined keys*)		**Inclusion Dependencies**	
PERSON	(NAME)		
DEPARTMENT	(NAME)		
COURSE	(NUMBER)		
FACULTY	(NAME)	FACULTY [NAME]	⊆ PERSON [NAME]
STUDENT	(NAME)	STUDENT [NAME]	⊆ PERSON [NAME]
OFFER	(C.NUMBER, D.NAME)	OFFER [C.NUMBER]	⊆ COURSE [NUMBER]
		OFFER [D.NAME]	⊆ DEPARTMENT [NAME]
SUPERVISE	(F.NAME, C.NUMBER)	SUPERVISE [C.NUMBER]	⊆ COURSE [NUMBER]
		SUPERVISE [F.NAME]	⊆ FACULTY [NAME]
COMMITTEE	(F.NAME, S.NAME)	COMMITTEE [S.NAME]	⊆ STUDENT [NAME]
		COMMITTEE [F.NAME]	⊆ FACULTY [NAME]
TEACH	(C.NUMBER, F.NAME)	TEACH [C.NUMBER]	⊆ OFFER [C.NUMBER]
		TEACH [F.NAME]	⊆ FACULTY [NAME]
ASSIST	(C.NUMBER, S.NAME)	ASSIST [C.NUMBER]	⊆ OFFER [C.NUMBER]
		ASSIST [S.NAME]	⊆ STUDENT [NAME]

Abbreviations: C–COURSE, D–DEPARTMENT, F–FACULTY, S–STUDENT

Figure 5. Relational Schema Representing the EER Schema of Figure 4.

4. PLAUSIBILITY CRITERIA FOR QUERY SUBGRAPHS

The capability to express abbreviated relational queries has the goal of relieving users from knowing the logical structure of the database, that is, its partition into relations. Thus, abbreviated relational queries are phrased in terms of attributes only, without the specification of the joins connecting the relations in which they appear [12]. In EER-oriented relational databases abbreviated relational queries correspond to the specification of abbreviated EER queries, as discussed below.

4.1 Query–Subgraphs.

An *EER–query* can be seen as selecting a connected subgraph of the EER diagram and associating predicate conditions with the diagram vertices [9]. We call such a subgraph a *query–subgraph*. Then, an *abbreviated EER-query* consists of the specification of (unconnected) EER diagram vertices and (possibly) edges, where an edge can be referenced by the role associated with it. An abbreviated EER-query implies a query-subgraph that spans all the vertices and edges referenced by the abbreviated query. For example, consider the EER schema represented in figure 4. If one wishes to know the faculty members in a certain department, one could just specify the entity-sets FACULTY and DEPARTMENT in the EER diagram of figure 4. In general there are multiple query-subgraphs that can be associated with some abbreviated query. For example, the subgraphs of figures 6(i), 6(ii) and 6(iii), are all valid query-subgraphs for the abbreviated query above. The obvious question is how would the system know to select the *most*

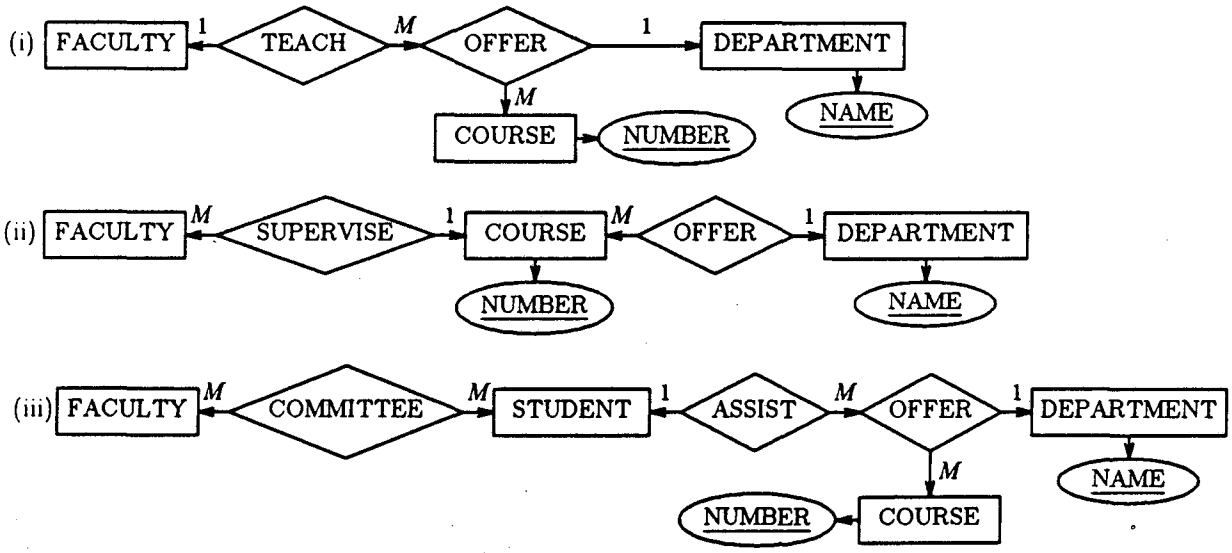


Figure 6. Query-Subgraphs of the Extended Entity-Relationship Diagram of Figure 4.

plausible query-subgraph. The issue of determining the plausibility of query-subgraphs is discussed in the following two subsections, while the complexity of finding the query-subgraphs for an abbreviated EER-query is examined in the next section.

4.2 The View–Robustness Criterion.

Every query-subgraph is associated with a *view* which is constructed by joining all the relations corresponding to the object-sets represented in the query-subgraph, as defined below. Informally, the joins involved in the definition of this view follow the adjacency † of object-set vertices in the corresponding query-subgraph. The precise form of the join-expressions is given in appendix B.

Definition 4.1 (*Query–View*). Let G_{ER} be an EER diagram and let $(R, F \cup I)$ be the relational schema corresponding to G_{ER}. Let $G'_{ER}(V', H')$ be a query-subgraph of G_{ER}, and let R' be the subset of relation-schemes of R that correspond to object-set vertices of V'. Then G'_{ER} is associated with a *query–view*, defined as the join of the relations associated with the relation-schemes of R'. □

Our criterion for the plausibility of a query-subgraph is the robustness of its associated query-view, where robustness is based on the capability of performing updates via the query-view, that is, the capability of translating update requests over the query-view into unambiguous updates on the base relations.

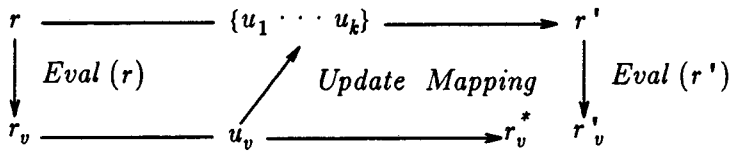
Definition 4.2 (*View–Robustness*). Let $(R, F \cup I)$ be the relational schema corresponding to EER diagram G_{ER}, and G'_{ER} a query-subgraph of G_{ER}. Then the *view–robustness* of G'_{ER} is defined as the robustness of the query-view associated with G'_{ER}. □

The *view–update* problem, that is, how to map updates expressed over views into updates of base relations, has been studied extensively (e.g. [4], [7]). The following definition, adapted from [4], characterizes the basic correctness criteria for view-update mappings.

Definition 4.3 (*View–Update Mapping*). Let r_v be a view relation derived from some database state r by an evaluation function *Eval*. Let u_v be an elementary update over r_v, and let $\{u_1 \cdots u_k\}$ be the underlying updates to which u_v is mapped.

Let $r_v^* = u_v(r_v)$, $r' = \{u_1 \cdots u_k\}(r)$, and $r'_v = Eval(r')$, as shown below:

† Two vertices are said to be adjacent if they are connected by an edge.



Then $\{u_1 \cdots u_k\}$ is said to

(i) *perform* $u_v = delete$ iff $r'_v \subseteq r^*_v$; and *perform* $u_v = insert$ iff $r^*_v \subseteq r'_v$;

(ii) *exactly perform* u_v (is *side–effect free*) iff $r'_v = r^*_v$. □

In other words, (i) states that the underlying updates produce the effect of the update on the view, and (ii) states that the underlying updates produce no side-effects beyond the update on the view. We propose to rank the robustness of a query-view by the existence of a side-effect free view-update mapping for this query-view. Thus, the *robustness* of a query-view is defined as follows:

(2) *robust* — the query-view has a side-effect free update (delete and insert) mapping;

(1) *weakly robust* — the query-view has a side-effect free delete mapping, but no side-effect free insert mapping; and

(0) *non robust* — the query-view has no side-effect free update mapping.

Proposition 4.1. Let $(R, F \cup I)$ be the relational schema corresponding to an EER diagram G_{ER}. Let $G'_{ER}(V', H')$ be a query-subgraph of G_{ER}, and let r_v be the query-view associated with G'_{ER} and having relation-scheme $R_v(X_v)$. Let $\overline{V}'$ be the subset of vertices of V' that have indegree 0‡, and let $\overline{R}'$ be the subset of relation-schemes of R that correspond to the vertices of $\overline{V}'$. Then

(i) r_v has a *side-effect free* insert mapping if for *every* relation-scheme $R_i(X_i)$ of $\overline{R}'$ the functional dependency $R_v : X_i \rightarrow X_v$ holds in r_v ; and

(ii) r_v has a *side-effect free* delete mapping if for *at least one* relation-scheme $R_i(X_i)$ of $\overline{R}'$ the functional dependency $R_v : X_i \rightarrow X_v$ holds in r_v. □

The proof of proposition 4.1, outlined in appendix B, is based on results from [7] and [4]. Proposition 4.1 shows that the insert side-effect-freeness condition is stronger than (i.e. includes) the delete side-effect-freeness condition.

Using the formalism developed in this section, one can show that the query-subgraphs of figure 6 have different robustness ranks: 2 (highest) for the query-subgraph of figure 6(i), 1 for the

‡ The indegree of a vertex is the number of edges incident to, and directed *from*, that vertex.

query-subgraph of figure 6(ii), and 0 (lowest) for the query-subgraph of figure 6(iii). For example, let r_1 denote the query-view associated with the query-subgraph of figure 6(i); r_1 is defined as the join of the relations associated with relation-schemes TEACH, FACULTY, OFFER, COURSE, and DEPARTMENT (see figure 5). Then subset $\overline{R}'$ of proposition 4.1 consists of relation-scheme TEACH; both conditions (i) and (ii) of proposition 4.1 are satisfied because the key-based inclusion dependencies involving TEACH ensure that each tuple in r_1 corresponds to a unique tuple of any relation associated with TEACH. Consequently, r_1 has a side-effect free update mapping. Similarly, let r_2 denote the query-view associated with the query-subgraph of figure 6(ii); r_2 is defined as the join of the relations associated with relation-schemes SUPERVISE, FACULTY, COURSE, OFFER, and DEPARTMENT (see figure 5). Then subset $\overline{R}'$ of proposition 4.1 consists of relation-schemes SUPERVISE and OFFER; condition (ii) of proposition 4.1 is satisfied because the join of the relations associated with SUPERVISE and OFFER, is on the key of OFFER, thus ensuring that each tuple in r_2 corresponds to a unique tuple of any relation associated with SUPERVISE; condition (i) of proposition 4.1, however, is not satisfied because a tuple in r_2 can correspond to more than one tuple of some relation associated with OFFER. Consequently, r_2 has only a side-effect free delete mapping.

Note that in this case it turns out that the most reasonable interpretation is indeed the one with the highest robustness (i.e. faculty members who teach courses offered by departments). The query-subgraph with robustness 1 is less reasonable (i.e. faculty members who supervise courses offered by departments). Finally, the query-subgraph with the lowest robustness is the least reasonable (i.e. faculty members who are on committees of graduate students who assist on courses offered by departments).

4.3 Combining Plausibility Criteria.

Selection criteria are necessary only when multiple query-subgraphs can be associated with a given abbreviated EER-query. However, our view-robustness criterion does not guarantee the choice of a unique query-subgraph for a given abbreviated EER-query. Moreover, our criterion, and any other similar criterion as well, does not necessarily lead to the query-subgraph that a user has actually in mind. Consequently, it is important to combine the view-robustness criterion with other preference criteria. There are many heuristic criteria that can be used. An obvious one is to prefer the query-subgraph with the fewest number of object-set vertices. The rationale is that simpler structures are less confusing for the user to imagine. Another is some affinity criterion. An affinity criterion is by its very nature imprecise, since it is based on some semantic intuition as to what a preferred query-subgraph would be for a given abbreviated query.

One such criterion may be based, for example, on the usage frequency of the different parts of the database. Thus, suppose that we associate a counter with each edge of the EER diagram, and that we increment these counters each time the edge belongs to the query-subgraph associated with some (abbreviated) query. Over time these counters indicate the preference for certain paths in the EER diagram, namely the paths whose edges have higher counters. Then the affinity criterion will rank the possible query-subgraphs according to their overall usage frequency. We feel that while such an affinity criterion still cannot guarantee the user's preference, it is a reasonable criterion because it is based on the accumulated usage of the database.

For a strategy that employs different plausibility criteria, the generation of multiple query-subgraphs for a given abbreviated query, is essential. Suppose, for example, that the view-robustness criterion is used jointly with some affinity criterion. One strategy may assume that the robustness of the view associated with the query is the dominant criterion for the user. Thus, the type of the query (retrieval, update) should determine which subgraphs to consider. If the query is of type *delete*, for example, then only query-subgraphs of view-robustness 1 and 2 would be considered. These query-subgraphs will then be further analyzed according to the affinity criterion. Yet another strategy may assume that the affinity is well specified for the application, and should be the dominant criterion. Accordingly, the query-subgraphs will be selected on the basis of the affinity criterion, and then ranked according to their robustness.

5. FINDING QUERY SUBGRAPHS FOR ABBREVIATED QUERIES

We discuss below the graph-theoretic aspects of finding query-subgraphs. First, we briefly review the graph-theoretic concepts needed in this section. Any textbook on graph theory, such as [5], can provide the necessary details.

5.1 Graph Concepts.

Let $G = (V, H)$ be a directed graph (digraph) with set of vertices V and set of directed edges H. The *underlying* undirected graph of a digraph results by ignoring the edge directions in the digraph. Let h denote a directed edge *incident* to vertices v_i and v_j. An undirected *path* from vertex v_{i_0} to vertex v_{i_m} is a sequence of alternating vertices and edges, $v_{i_0} h_{j_1} v_{i_1} \ldots h_{j_m} v_{i_m}$, such that h_{j_k} is incident to $v_{i_{k-1}}$ and v_{i_k}, $1 \leq k \leq m$; if $v_{i_0} \equiv v_{i_m}$ then the path is called a *cycle*. An undirected graph is called a *tree* iff it has no cycles and any edge added to it forms a cycle.

The subgraph *induced* by a subset of V, V', is denoted $G(V')$ and is defined as follows:

$G(V') = (V', H')$, where $H' = \{h \mid h \in H$ and h is incident to vertices of $V'\}$.

A *spanning tree* of a graph G is a subgraph of G that contains all the vertices of G and is a tree. If the edges of the graph are associated with *lengths* (*weights*) then the *minimum* spanning tree is the spanning tree with the minimum sum of edge lengths. Given a subset of the vertices of a graph G, W, a *Steiner tree* is a subgraph of G that contains all the vertices of W, is a tree, and whose sum of edge lengths is minimal.

Let $G' = (V', H')$ and $G'' = (V'', H'')$ denote two subgraphs of G, induced by V' and V'', respectively. Then G' and G'' are said to be *adjacent* iff $V' \cap V'' \neq \emptyset$; and G is said to be the *edge-disjoint union* (ed-union) of G' and G'' iff $V = V' \cup V''$, $H = H' \cup H''$, and $H' \cap H'' = \emptyset$.

A connected undirected graph can be partitioned into a ed-union of *nonseparable components* which are subgraphs that have only one vertex in common, called the *separation vertex* (for the underlying graph of the EER diagram in figure 7, for instance, G_a and G_b are two nonseparable components and vertex 4 is their separation vertex). All the paths between the vertices of two adjacent nonseparable components pass through their separation vertex. Let $NS_1, \ldots, NS_m$, be the nonseparable components of $G = (V, H)$, and let $s_1, \ldots, s_p$, be its separating vertices. The *superstructure* of G, G^*, is an undirected graph constructed as follows: the vertices of G^* are $s_1, \ldots, s_p$, and $ns_1, \ldots, ns_m$, where vertex ns_j represents the nonseparable component NS_j; and if s_i is a vertex of NS_j in G then there is an edge incident to s_i and ns_j in G^*.

5.2 Finding Query–Subgraphs.

Given a set of EER elements, Q, we want to find the most plausible query-subgraph spanning Q. Solutions to this problem in the context of the basic ER model are proposed in [10] and [21]. Both [10] and [21] associate a *weight* with every edge of an ER diagram: in [10] the weight is an affinity probability for the vertices connected by the edge, and in [21] the weight is an estimated cost for traversing the edge. Then the most plausible query-subgraph is considered to be the Steiner tree corresponding to Q. It is worth noting that the main concern of [10] and [21] is to cope with the complexity of the Steiner tree problem, known to be NP-complete [5]. Thus, [21] proposes a linear time algorithm for a class of restricted ER diagrams, while [10] proposes an approximation algorithm for unrestricted ER diagrams. These approaches depend on subjective justifications for assigning weights and always produce a single query-subgraph.

The more appealing approach to the problem of finding the most plausible query-subgraph

spanning a set of EER elements Q, is to generate all the relevant query-subgraphs, evaluate their plausibilities, and select the query-subgraph with the highest plausibility. This approach does not depend on a specific plausibility criterion, and therefore could employ any criterion or a combination of different criteria, as discussed in section 4.3. Moreover, if the user does not accept the interpretation (query-subgraph) selected for a given abbreviated query, the other possible interpretations are already available for the user's inspection. Note that for acyclic graphs, the query-subgraph implied by an abbreviated query is unique and its generation is trivial: simply remove all vertices that are not in Q while preserving the connectivity of the subgraph. However, for arbitrary graphs the generation of all query-subgraphs can be impractical since it would require time exponential to the size of the graph (worst case of a variation of the Steiner tree problem).

For practical EER diagrams, however, the generation of query-subgraphs can be significantly simpler. First, we do not need to look at the entire EER diagram, but only at the subgraph induced by the subset of vertices representing object-sets; typically, the number of object-sets in an EER schema is smaller than the number of attributes by an order of magnitude. Furthermore, it is sufficient to look for query-subgraphs which have acyclic (tree) underlying graphs, since any (cyclic) query-subgraph can be constructed as a graph union of such acyclic subgraphs. Moreover, we observe that EER diagrams tend to have few undirected cycles and can be partitioned into several nonseparable components. For example, the EER diagram shown in figure 7, G_{ER}, can be partitioned into the ed-union of subgraphs G_a, G_b, G_c, G_d, G_e, and G_f; vertices 4, 5, 7, 8, 9, 12, and 16 are separation vertices; and subgraphs G_a, G_b, G_d, and G_e correspond to nonseparable components of the underlying graph of G_{ER}, while subgraphs G_c and G_f are acyclic and can be further partitioned into nonseparable components.

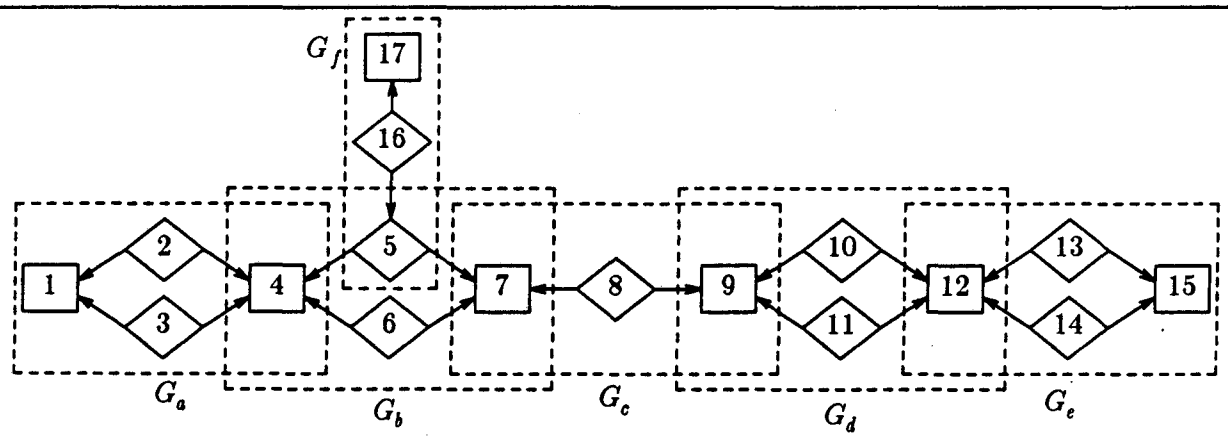


Figure 7. Nonseparable Components in an Extended Entity-Relationship Diagram.

The nonseparable components of an EER diagram can be determined by using a *Depth-First Search* oriented algorithm (see e.g. [5]). As noted in [5], the superstructure of an undirected graph is acyclic. This means that the undirected cycles in an EER diagram are local to (lie entirely in) the single nonseparable components (e.g. see the cycles of the EER diagram in figure 7). Then, given the set of object-sets referenced by an abbreviated query, Q', a query-subgraph G' can be decomposed into a ed-union of subgraphs, G'_i, where every subgraph G'_i is a subgraph of a nonseparable component. Consequently, (i) the acyclicity of the superstructure allows the removal of all the nonseparable components which intersect Q' in at most one separating vertex and which are not placed between nonseparable components that contain vertices of Q'; and (ii) the query-subgraph generation reduces to the generation of subgraphs within single nonseparable components of an EER diagram. Furthermore, for acyclic components, such as G_c and G_f in the EER diagram of figure 7, the query-subgraph generation is trivial.

Consider, for example, an abbreviated EER-query over the EER diagram of figure 7 that refers to vertices 4, 12, and 17. First, the subgraphs G_a and G_e can be removed since they satisfy condition (i) above. Next, observe that the query-subgraphs that are generated for this abbreviated query consist of subgraphs of G_b, G_c, G_d, and G_f, where every subgraph includes the separation vertices of the corresponding nonseparable component. Accordingly, vertices 4, 5, 7, 8, 9, 12, 16, and 17, must be included in any query-subgraph. Vertices 4, 5, and 7, can be covered by two different spanning trees of the underlying graph of G_b, and vertices 9 and 12 are connected by two different paths in the underlying graph of G_d. Consequently, there are four different query-subgraphs that can be associated with the abbreviated query above, namely the four possible ed-union combinations of the two spanning trees of G_b, the single spanning tree of G_c, the two spanning trees of G_d, and the single spanning tree of G_f.

The approach discussed above takes advantage of the characteristic structure of EER diagrams (few and localized cycles) in order to reduce the complexity of generating query-subgraphs. Further examination of real (large) EER diagrams is warranted to strengthen the observation of such properties.

6. CONCLUSION

We have proposed a methodology for abbreviated query interpretation in EER-oriented relational databases. We have introduced a new criterion to rank the plausibility of the interpretations that can be associated with abbreviated queries, namely the existence of a side-effect free update mapping for the view implied by the query. We have discussed techniques that simplify the generation of the query-subgraphs for a given abbreviated query by taking advantage of certain properties of EER diagrams.

Although having a natural intuition, our methodology, like analogous UR methodologies, does not guarantee the selection of a single query-subgraph for a given abbreviated query. Furthermore, even when a single query-subgraph is found, we cannot guarantee that it agrees with the user's intended meaning. Therefore some strategy is needed for verifying whether the user's intentions have been satisfied, and for selecting from several query-subgraphs. Most UR methodologies simply union all the derived relations corresponding to the various query-subgraphs [12]. In EER-oriented databases such a strategy cannot be applied because multiple paths in these databases have distinct meanings (*flavors*) and therefore cannot be mixed without consulting the user. Moreover, while union is sometimes desirable, it is by no means the only reasonable choice. Indeed, selecting a single query-subgraph is more likely to be desired in most cases.

Another obvious strategy is to consult the user. In the extreme, the user can be shown all the alternative query-graphs, and chose the desirable one. If such a strategy is employed, then our methodology can order the choices according to the view-robustness. We have outlined some heuristic criteria that can be used in order to approximate the user's intentions. We have argued that any such criterion should be used in combination with our robustness criterion, because view-robustness is essential for ensuring the correctness of abbreviated update queries.

REFERENCES

[1] P. Atzeni and D.S. Parker, "Assumptions in Relational Database Theory",*ACM Symposium on Principles of Database Systems*, 1982, pp. 1-9.

[2] P.P. Chen, "The Entity-Relationship Model- Towards a Unified View of Data", *ACM Trans. on Database Systems* 1,1 (March 1976), pp. 9-36.

[3] S.S. Cosmadakis and P.C. Kanellakis, "Equational Theories and Database Constraints", in *Proc 17th ACM Symposium on Theory of Computing,* 1985, pp. 273-284.

[4] U. Dayal and P.A. Bernstein, "On the Correct Translation of Update Operations on Relational Views", *ACM Trans. on Database Systems* **8**,3 (Sep 1982), pp. 381-416.

[5] S. Even, *Graph Algorithms* , Computer Science Press, 1979.

[6] R. Hull and R. King, "Semantic Database Modeling: Survey, Applications, and Research Issues", *Computing Surveys* **19**,3 (September 1987), pp. 201-260.

[7] A.M. Keller, "Algorithms for Translating View Updates to Database Updates for Views Involving Selections, Projections, and Joins", *ACM Symposium on Principles of Database Systems* , 1985, pp. 154-163.

[8] A. Klug, "Calculating Constraints on Relational Expressions", *ACM Trans. on Database Systems* **5**,3 (Sep. 1980), pp. 260-290.

[9] Y.E. Lien, "On the Semantics of the Entity-Relationship Data Model", in *Entity-Relationship Approach to System Analysis and Design*, P.P. Chen (ed), North-Holland, Amsterdam, 1980, pp. 155-167.

[10] D. Lin, "Automatic Logical Navigation among Relations Using Steiner Trees", Proc. of the *5th International Conference on Data Engineering*, 1989, pp. 582-588.

[11] D. Maier, D. Rozenshtein, and J. Stein, "Representing Roles in Universal Scheme Interfaces", *IEEE Trans. on Software Engineering* , vol SE-11,7, July 1985, pp. 644-652.

[12] D. Maier, D. Rozenshtein, and D.S. Warren, "Window Functions", *Advances in Computing Research* , vol.3, JAI Press, 1986, pp. 213-246.

[13] D. Maier and J.D. Ullman, "Maximal Objects and the Semantics of Universal Relation Databases", *ACM Trans. on Database Systems* **8**,1 (March 1983), pp. 1-14.

[14] D. Maier, J.D. Ullman, and M. Vardi, "On the Foundations of the Universal Relation Model", *ACM Trans. on Database Systems* **9**,2 (June 1984), pp. 283-308.

[15] V.M. Markowitz and A. Shoshani, "Name Assignment Techniques for Relational Schemas Representing Extended Entity-Relationship Schemas", Proc. of the *8th International Conference on Entity-Relationship Approach*, F.H. Lochovsky (ed), North-Holland, 1989 (also published as LBL TR-27070, Lawrence Berkeley Laboratory, August 1989).

[16] V.M. Markowitz and A. Shoshani, "On the Correctness of Representing Extended Entity-Relationship Structures in the Relational Model", Proc. of *1989 SIGMOD Conference*, SIGMOD Record **18**, 2, June 1989, pp. 430-439.

[17] A. Pahwa and A.K. Arora, "Automatic Database Navigation: Towards a High Level User

Interface", Proc. of *The 4th International Conference on Entity-Relationship Approach* , P.P. Chen (ed), IEEE Computer Society Press, 1985, pp. 36-43.

[18] T.J. Teorey, D. Yang, and J.P. Fry, "A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model", *Computing Surveys* **18**,2 (June 1986), pp. 197-222.

[19] J.D. Ullman, *Principles of Database Systems*, Computer Science Press, 1982.

[20] M.S.D. Wilkie and A.O. Mendelzon, "Analysis and Translation of Entity-Relationship Database Schemas", *Information Processing 83*, R.E. Mason (ed), North-Holland, 1983, pp.609-614.

[21] J.A. Wald and P.G. Sorenson, "Resolving the Query Inference Problem Using Steiner Trees", *ACM Trans. on Database Systems* **9**,3 (Sep. 1984), pp. 348-368.

[22] Z.Q. Zhanq and A.O. Mendelzon, "A Graphical Query Language for Entity-Relationship Databases", *Entity-Relationship Approach to Software Engineering* , G.C. Davis and al (eds), North-Holland, 1983, pp. 441-448.

APPENDIX A: MAPPING EER SCHEMAS INTO RELATIONAL SCHEMAS

In this appendix we present the mapping of EER schemas into relational schemas. First, we briefly review the relational concepts we use. Details can be found in any textbook (e.g. [19]) for the basic concepts and in [3] for the theory of inclusion dependencies. We use letters from the beginning of the alphabet to denote attributes and letters from the end of the alphabet to denote sets of attributes. A sequence of attributes (e.g. ABC) denotes the set containing these attributes and a sequence of sets (e.g. XY) denotes the union of these sets. We denote by t a tuple, and by $t[W]$ the sub-tuple of t corresponding to the attributes of W.

A *relational schema* is a pair (R, Δ) where R is a set of relation-schemes and Δ is a set of dependencies over R. We consider relational schemas which are associated with set of dependencies $\Delta = F \cup I$, where F and I denote sets of functional and inclusion dependencies, respectively. A *relation–scheme* is a named set of attributes, $R_i(X_i)$, where R_i is the relation-scheme name and X_i denotes the associated set of attributes. Every attribute is assigned a *domain*, and every relation-scheme, $R_i(X_i)$, is assigned a *relation* (value), r_i. Two attributes are said to be *compatible* if they are associated with the same domain, and two sets of attributes, X and Y, are said to be *compatible* iff there exists a one-to-one correspondence of compatible attributes between X and Y.

Let $R_i(X_i)$ be a relation-scheme associated with relation r_i; the *projection* of r_i on $W \subseteq X_i$ is denoted $r_i[W]$, and is equal to $\{ t[W_i] \mid t \in r_i \}$. Let $R_i(X_i)$ and $R_j(X_j)$ be two relation-schemes associated with relations r_i and r_j, respectively; the *natural join* of r_i and r_j is denoted $r_i \bowtie r_j$, and is equal to $\{t \mid t[X_i] \in r_i, t[X_j] \in r_j\}$; the *equi–join* of r_i and r_j on $(Y = Z)$, where Y and Z are compatible and disjoint subsets of X_i and X_j, respectively, is denoted $r_i \underset{Y=Z}{\bowtie} r_j$, and is equal to $\{t \mid t[X_i] \in r_i, t[X_j] \in r_j$, and $t[Y] = t[Z]\}$.

Let $R_i(X_i)$ be a relation-scheme associated with relation r_i. A *functional dependency* over R_i is a statement of the form $R_i: Y \rightarrow Z$, where Y and and Z are subsets of X_i; $R_i: Y \rightarrow Z$ is *satisfied* by r_i iff for any two tuples of r_i, t and t', $t[Y] = t'[Y]$ implies $t[Z] = t'[Z]$. Let $R_i(X_i)$ and $R_j(X_j)$ be two relation-schemes associated with relations r_i and r_j, respectively. An *inclusion dependency* is a statement of the form $R_i[Y] \subseteq R_j[Z]$, where Y and Z are compatible subsets of X_i and X_j, respectively; $R_i[Y] \subseteq R_j[Z]$ is *satisfied* by r_i and r_j iff $r_i[Y] \subseteq r_j[Z]$.

A *key* associated with R_i is a subset of X_i, K_i, such that $R_i : K_i \rightarrow X_i$ is satisfied by any r_i associated with R_i and there does not exist any proper subset of K_i which has this

property. A relation-scheme can be associated with several *candidate keys* from which one *primary key* is chosen. If $R_i[Y] \subseteq R_j[Z]$ is an inclusion dependency and Z is the primary key of R_j then $R_i[Y] \subseteq R_j[Z]$ is said to be *key–based*, and Y is called a *foreign key* of R_i *referencing* R_j, denoted FK_{i_j}.

The mapping of EER schemas into relational schemas, called **REL**, is specified below. **REL** is based on certain *well–definedness* properties of EER diagrams, such as, lack of directed cycles, uniqueness of generalization-sources for specializations, etc., which are discussed in [16]. The correctness of mappings such as **REL**, is also examined in [16]. Note that the relational attributes generated by **REL** are assigned names that are guaranteed to be unique only with respect to their relation-schemes.

Definition A.1. - REL : *Mapping EER Schemas Into Relational Schemas.*

Input: an EER schema associated with EER Diagram $G_{ER} = (V, H)$.

Output: a relational schema of the form $(R, F \cup I)$.

(1) *Value–Sets*. Every value-set is mapped into a relational domain.

(2) *Independent Entity–Sets*. Every independent entity-set, E_i, is mapped into a relation-scheme, $R_i(X_i)$, such that: (i) X_i is in a one-to-one correspondence with the EER attributes of E_i, (ii) every attribute A_{i_j} of X_i (a) is assigned the name of the EER attribute of E_i corresponding to A_i, and (b) is associated with the domain corresponding to the value-set of the EER attribute of E_i corresponding to A_i.

The subset of X_i, K_i, which is in a one-to-one correspondence with the identifier of E_i is the (unique) key of R_i, and the unique functional dependency added to F is the key dependency $R_i : K_i \rightarrow X_i$.

(3) *Specialization Entity–Sets*. Let entity-set E_i be the specialization of entity-sets E_{i_j}, $1 \leq j \leq m$, and E_s be the (unique) generalization-source of E_i. Let E_s correspond to relation-scheme $R_s(Y_s)$ and entity-sets E_{i_j} correspond to relation-schemes $R_{i_j}(Y_{i_j})$, $1 \leq j \leq m$, respectively. Then entity-set E_i is mapped into relation-scheme $R_i(X_i)$ and inclusion dependencies $R_i[FK_{i_j}] \subseteq R_{i_j}[K_{i_j}]$, $1 \leq j \leq m$, where X_i is the union of two disjoint sets of attributes, X'_i and K_i, such that: (i) X'_i is in a one-to-one correspondence with the EER attributes of E_i, where the correspondence is specified as in (2.ii) above; (ii) the (unique) key of R_i, K_i, is equal to the key of R_s, and (iii) every foreign key FK_{i_j}, $1 \leq j \leq m$, is equal to K_i.

The unique functional dependency added to F is the key dependency, $R_i : K_i \rightarrow X_i$.

(4) *Aggregation Object–Sets.* Let object-set O_i be the aggregation of object-sets O_{i_j}, $1 \leq j \leq m$, and let object-sets O_{i_j} correspond to relation-schemes $R_{i_j}(Y_{i_j})$, $1 \leq j \leq m$, respectively. Then object-set O_i is mapped into relation-scheme $R_i(X_i)$ and inclusion dependencies $R_i[FK_{i_j}] \subseteq R_{i_j}[K_{i_j}]$, $1 \leq j \leq m$, where X_i is the union of two disjoint sets of attributes, X'_i and X''_i, such that: (i) X'_i is in a one-to-one correspondence with the EER attributes of O_i, where the correspondence is specified as in (2.ii) above;

(ii) $X''_i = \bigcup_{j=1}^{m} FK_{i_j}$, is a set of foreign-key attributes, where every foreign-key FK_{i_j} is in a one-to-one correspondence with K_{i_j}, $1 \leq j \leq m$, such that (a) every attribute of FK_{i_j} is assigned the name† of the corresponding attribute of K_{i_j}; and (b) the domain associated with an attribute of FK_{i_j} is the domain of the corresponding attribute of K_{i_j}.

Let O_i be a *weak entity-set* and Z_i be the subset of X_i which is in a one-to-one correspondence with the identifier of E_i. Then the key of R_i, K_i, is equal to $Z_i\, X''_i$, and the unique functional dependency added to F is the key dependency $R_i : K_i \rightarrow X_i$.

Let O_i be a *relationship–set*. If all the cardinalities of the object-sets involved in O_i are *many*, then the primary key of R_i, K_i, is equal to X''_i, and the unique functional dependency added to F is the key dependency $R_i : K_i \rightarrow X_i$; otherwise the primary key of R_i, K_i, is equal to $(X''_i - FK_{i_k})$, where FK_{i_k} is the *foreign–key* that references the relation-scheme corresponding to an object-set that has cardinality *one* in O_i, and for every object-set O_{i_j} which has cardinality *one* in O_i, the functional dependency $R_i : (X''_i - FK_{i_j}) \rightarrow FK_{i_j}$ is added to F. □

† For certain EER structures, it is necessary to prefix the names of such foreign key attributes either by the role of O_{i_j} in O_i, or (when the role is not specified) by the name of O_{i_j}, $1 \leq j \leq m$. For details see [15].

APPENDIX B : SIDE–EFFECT FREE VIEW–UPDATES FOR JOIN–VIEWS

We present in this appendix the formal details concerning the conditions that guarantee side-effect free view-update mappings for query-views associated with query-subgraphs. An elementary update in a relational database consists of either deleting a tuple from a relation or inserting a tuple into a relation. Let r_i be a relation. We denote the deletion and insertion of a tuple t, from/into r_i, as $delete(t, r_i)$ and $insert(t, r_i)$, respectively, and commonly as $update(t, r_i)$.

We use below the additional graph-theoretic concepts of *reachability* and *root*. Let $G = (V, H)$ be a digraph. A vertex v_j of V is said to be *reachable* from another vertex v_i of V iff there exists a directed path from v_i to v_j in G. A vertex v_i is said to be a *root* in G iff for every v_j of V, v_j is reachable from v_i.

The correspondence of attributes for the joins of definitions B.1 and B.2 below is based on the adjacency of vertices in the EER diagram. We assume below that all the relational attributes have globally unique names in the relational schema, for instance by prefixing every attribute name by the associated relation name.

Definition B.1 - (*Rooted–Subgraph View*). Let G_{ER} be an EER diagram and let $G'_{ER} = (V', H')$ be a connected subgraph of G_{ER} with root O_i. Let $(R, F \cup I)$ be the relational schema corresponding to G_{ER}, R' the subset of relation-schemes of R corresponding to the vertices of V', I' the subset of inclusion dependencies of I corresponding to the edges of H', and $R_i(X_i)$ the relation-scheme of R corresponding to O_i.

Let S be a set of relation-schemes consisting initially of $R_i(X_i)$.

(i) The *rooted-subgraph view* associated with G'_{ER} is denoted r_i^* and is defined as follows:

$r_i^* := r_i$, $X_i^* := X_i$; and

While $(S \neq R')$ *Do*

Choose $R_j(X_j) \in (R' - S)$ such that $R_k[FK_{k_j}] \subseteq R_j[K_j] \in I'$ and $R_k(X_k) \in S$;

$r_i^* := r_i^* \underset{FK_{k_j} = K_j}{\bowtie} r_j$, $X_i^* := X_i^* \cup X_j$, and $S := S \cup \{R_j\}$.

EndDo

(ii) The view-update mapping for the *rooted-subgraph view* associated with G'_{ER} is defined by mapping T'_{upd}, as follows:

$$T'_{upd}(\ update(t,\ r_i^*\ )\ ) \stackrel{\Delta}{=} update(t,\ r_i^*\ ) \cup \{update(\ t',\ r_j\ ) \mid R_j \in R',\ t' = t[X_j]\}.\ \square$$

Lemma B.1. Let r_i^* be a rooted-subgraph view defined as above, and t a tuple to be inserted/deleted into/from r_i^*. Then $T'_{upd}(\ update(t,\ r_i^*\ )\ )$ exactly performs $update(t,\ r_i^*\ )$.

Proof : Apply theorem 5-2 from [7]. $\square$

Definition B.2 - (*Non—Rooted—Subgraph View*) . Let G_{ER} be an EER diagram and let $G'_{ER}\ (\ V',\ H'\ )$ be a connected subgraph of G_{ER} . Let $\overline{V}'$ be the subset of vertices of V' with indegree 0 in G'. Let $(\ R,\ F \cup I\ )$ be the relational schema corresponding to G_{ER} , and let $\overline{R}'$ be the subset of relation-schemes of R that correspond to the vertices of $\overline{V}'$. Every vertex of $\overline{V}'$, O_i , together with all the vertices of V' that are reachable from O_i induce a subgraph of G'_{ER} with root O_i which is associated with a rooted-subgraph view, r_i^* , specified as in definition B.1 above.

(i) The *non-rooted-subgraph view* associated with G'_{ER} is defined as follows:

(a) for some $R_i(X_i) \in \overline{R}'$: $\quad r_v := r_i^*\ ,\ X_v := X_i^*\ ;$ and

(b) for every $R_j(X_j) \in (\ \overline{R}' - R_i\ )$: $\quad r_v := r_v \bowtie r_j^*$ and $X_v := X_v \cup X_j^*$.

(ii) The view-update mapping for the *non-rooted-subgraph view* associated with G'_{ER} is defined by mapping T''_{upd} , as follows:

(a) $T''_{upd}(insert(t,\ r_v)\ ) \stackrel{\Delta}{=} insert(t,\ r_v\ ) \cup \{insert(t',\ r_j^*\ ) \mid R_j \in \overline{R}',\ t' = t[X_j^*\ ]\}$;

(b) $T''_{upd}(\ delete(t,\ r_v)\ ) \stackrel{\Delta}{=} delete(t,\ r_v)\ \cup \{delete(t',\ r_j^*\ ) \mid R_j \in Delete(r_v),\ t' = t[X_j^*\ ]\}$,

where $Delete\ (r_v)$ is a nonempty subset of $\overline{R}'$. $\square$

Lemma B.2. Let r_v be a non-rooted-subgraph view defined as above, and t any tuple to be inserted/deleted into/from r_v . Then

(i) $T''_{upd}(\ insert(t,\ r_v)\ )$ exactly performs $insert(t,\ r_v)$ if for every relation-scheme $R_i(X_i)$ of $\overline{R}'$ the functional dependency $R_v : X_i \rightarrow X_v$ holds in r_v ; and

(ii) $T''_{upd}(\ delete(t,\ r_v)\ )$ exactly performs $delete(t,\ r_v)$ if for every relation-scheme $R_i(X_i)$ of $Delete\ (r_v)$ the functional dependency $R_v : X_i \rightarrow X_v$ holds in r_v.

Proof : apply theorems 4 and 5 of [4]. $\square$

Proof of Proposition 4.1.

Every query-subgraph G'_{ER} is decomposed into a union of adjacent rooted subgraphs. Then the joins involved in the definition of r_v are ordered as follows: (i) for every subgraph of G'_{ER} which has a root, a rooted-subgraph view is defined; and (ii) all rooted-subgraph views are joined, which is equivalent to the definition of a non-rooted-subgraph view, r_v. The update mapping for r_v is defined as the composition of T'_{upd} and T''_{upd}. By lemma B.1 and B.2, the composition of T'_{upd} and T''_{upd} exactly performs $update(t\ ,\ r_v)$ when the conditions of proposition 4.1 are satisfied. □

Although we have applied results from [4] in order to prove proposition 4.1, the method proposed in [4] to verify the side-effect freeness of view-update mappings (constructing view graphs) is not necessary. Alternatively, the functional dependencies that hold in some query-view can be generated by using the well known derivation rules together with the following additional rule: $R_i : X = Y$ *implies* $R_i : X \rightarrow Y$ (for details and proofs see [8]).

LAWRENCE BERKELEY LABORATORY
TECHNICAL INFORMATION DEPARTMENT
1 CYCLOTRON ROAD
BERKELEY, CALIFORNIA 94720