RELATÓRIO TÉCNICO

PROCEDURAL DATA MANIPULATION OPERATIONS FOR THE E-R MODEL

Pedro Manoel da Silveira

NCE-19/90
Agosto/90

Universidade Federal do Rio de Janeiro
Núcleo de Computação Eletrônica
Caixa Postal 2324
20001 - Rio de Janeiro - RJ
BRASIL

Este artigo foi aceito na XVI Conferência Latino Americana de Informatica, 1990, Assunção/Paraguai.

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
NÚCLEO DE COMPUTAÇÃO ELETRÔNICA

Procedural Data Manipulation Operations for the E-R Model

PEDRO MANOEL SILVEIRA

NÚCLEO DE COMPUTAÇÃO ELETRÔNICA — UFRJ

CAIXA POSTAL 2324, CEP 20001, RIO DE JANEIRO - RJ

E-mail: PEDRO@UFRJ.BITNET

ABSTRACT

A basic set of data manipulation operations for the E-R Model is defined. Ordering relations are formally introduced to support sequential ordered access to the instances of E-R schemata. These operations configurate a data manipulation language for application in E-R oriented database interfaces.

Operações de Manipulação de Dados para o Modelo E-R

RESUMO

Um conjunto básico de operações de manipulação de dados para o Modelo E-R é apresentado. Relações de ordenação são formalmente definidas e utilizadas para suportar acesso ordenado às instâncias dos esquemas E-R. Estas operações configuram uma linguagem de manipulação de dados para interfaces operando em bancos de dados orientados para o Modelo E-R.

1. INTRODUCTION

Usually, the E-R model[2,6,7] is employed for database modeling at the conceptual level, while implementation is carried out through data management systems that may rely on different modeling frameworks. For a few implementations, however, the E-R model remains as the reference guide even at the physical level, allowing a simpler and more consistent view of the database at different levels.[3,5]

In the UniversiData Project,[9] the E-R Model has been used both as modeling and database organization framework. Application programmers have a common and unique vision of the data bank through the E-R schema, and manipulate its instances by using a specially designed DML. The essence of this manipulation discipline is the range of operations provided along further concepts that allow a better understanding of the implementation at the physical level.

This paper presents the formal aspects of these operations, having as scenario a formal framework for E-R schemata and their interpretations. Section 2 defines the basis of E-R schemata and interpretations; Section 3 introduces further formal elements such as ordering functions, used to conduct sequential ordered access. Section 4 discusses the operations themselves, while implementation aspects are left to Section 5. Section 6 concludes the article.

2. A FORMAL BASIS FOR THE E-R MODEL

Although the E-R model has been widely explained and analised in the literature, its formal basis is not yet satisfactorily clear, and authors present different views of the logical underlying framework. This section establishes some basic points in this direction, and contains material heavily borrowed from a previous paper[8] where we lay our contribution for establishing this formal basis. The interested reader is referred to that work for further details on the E-R model formalization.

E-R SCHEMATA

The basic elements for building E-R schemata are:

Constants: c_1, $c_2, \ldots$ and ϕ, the null value.

Entity Sets: E_1, E_2

Relationships: R_1 $R_2, \ldots$ Each relationship R has an integer $dg(R)$, said the

degree of R, $dg(R) \geq 2$, and a list of entity sets of the form $(E_1, \ldots, E_{dg(R)})$ associated with it. These are said the participating entities of R.

Role functions: ρ_1, $\rho_2, \ldots$ For each relationship R there is a set of role functions of the form $\{\rho_1 : R \rightarrow E_1, \ldots, \rho_{dg(R)} : R \rightarrow E_{dg(R)}\}$, where each E_i is a participating entity of R.

Value Sets: V_1, $V_2, \ldots$ To each value set V is associated a set of constants $\{c_1, \ldots, c_n\}$.

Attribute symbols: α_1, $\alpha_2, \ldots$ Each attribute α_i is a function of the form $\alpha_i : X, I \rightarrow V$, where X is either an entity set or a relationship, I is an integer and V is a value set. α_i has an integer $maxoc(\alpha_i)$ associated with it, $1 \leq I \leq maxoc(\alpha_i)$.

Integrity constraints: θ_1, $\theta_2, \ldots$ where θ_i is a well-formed formula.

SCHEMA PROPERTIES

Although integrity constraints are freely allowed as basic elements of the model, real implementations of database are plagued by the difficulties in implementing them efficiently. However, there are some classes of restrictions that are intrinsically linked to the E-R model and must be reflected in the physical implementation of the derived databases. These are: *keys*, *type hierarchies*, *cardinalities of relationships* and *weak entities*.

Keys

Keys are sets of attributes that uniquely identify the instances of a given entity set or relationship. There is a set of functions $\kappa_1, \ldots, \kappa_m$, whose general form is

$$\kappa_i : \{E_1, \ldots, E_n, R_1, \ldots, R_k\} \rightarrow \{\alpha_1, \ldots, \alpha_n, \eta_0, \rho_1, \ldots, \rho_{dg(X)}\}^* \cup \phi$$

where $\alpha_1, \ldots, \alpha_n$ are the attribute symbols defined over the argument and $\rho_1, \ldots, \rho_{dg(X)}$ its role functions if the argument is a relationship. If the argument is a weak entity, then η_0 is a key-path for X. For each X in a schema S, such that X is either an entity or a relationship, the functions $\kappa_1, \ldots, \kappa_{n\kappa(X)}$ are defined, and the application $\kappa_i(X)$ returns the set of attribute symbols and/or role function symbols and/or key-path symbols that form the i-th key for X. The function $n\kappa(X)$ returns the number of keys defined for a given entity/relationship X.

The functions above defined return the list of attributes that form a given key. For entities, one can use only attributes defined for that entity, even

inherited ones. For relationships, the list of attributes forming some key can include attributes of the participating entities. For weak-entities, attributes of the reference entity, i. e. the one they are attached to, can be in the list too.

Notice that in the general case, attributes can have multiple occurrences and an indication of which occurrence is part of the key should be provided. For real implementations, however, multi-valued attributes are not normally used as keys. For this reason, we have decided to assume that if an attribute α is part of a key, then $maxoc(\alpha) = 1$ and so it is not necessary to especify which occurrence should be used.

Type hierarchies

Generalization is the term used when two or more entities can have their differences abstracted and, at a higher level, represented as a common entity, which holds the common attributes of its different sorts. The inverse way is called specialization, where one can go from the generic to the specific level. Subtype hierarchy is simply the assumption that a given entity type is always part of a more general entity type, though not all general entities have to be in that subtype entity.

In this work, we deal with simple cases of generalization, where a given entity type can have one and only one entity type as its super type. No recursion is allowed. If the net of generalization was represented by a graph, there could be no cycles. In fact, it would have to be a tree, since a given entity can have as its supertype at most one other entity. Notice that further levels of complexity can be achieved by the introduction of integrity constraints.

The function $\beta : \{E_1, \ldots, E_n\} \cup \phi$ when applied to an entity set symbol returns its basic entity set symbol. For example, if $\beta(E_i) = E_j$, it means that E_j is a generalization of the entity set E_i. For E_i being a primitive entity set, $\beta(E_i)$ would return ϕ.

Cardinalities

The functions $Cmax$ and $Cmin$, when applied to sets of role function symbols associated with relationships, return integers that mean the maximum and minimum cardinalities allowed for the possible combinations of the corresponding entities in the relationships instances. This construction was inspired in an article by Lanzerini & Santucci.[4] We work with the notion of role functions while those authors use the participating entities directly.

To exemplify, let us consider the simple case depicted in Figure 1. Consider

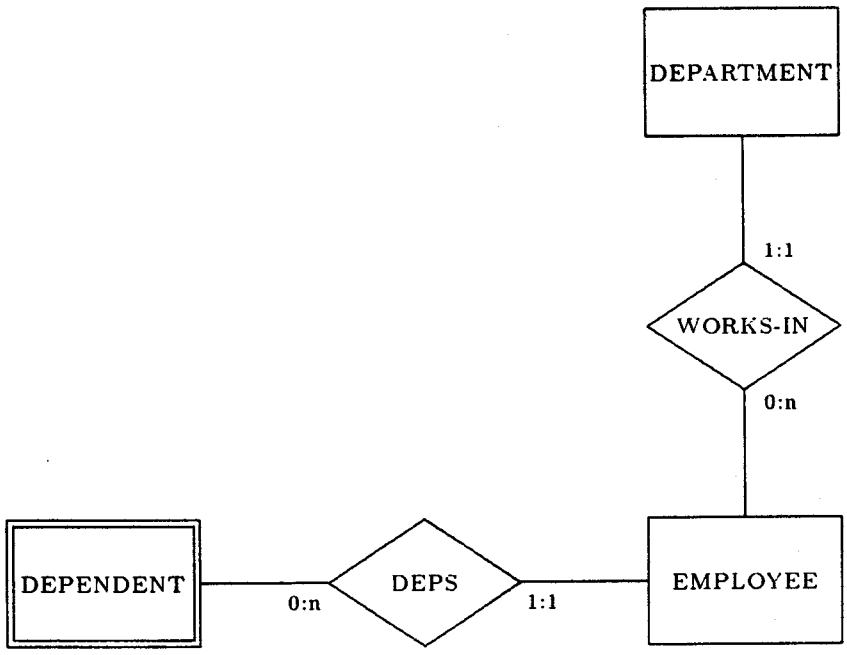

Figure 1. A simple E-R example schema

the relationship WORKS-IN between entities DEPARTMENT and EMPLOYEE, where the role functions are named after their range entities.

Entities: DEPARTMENT, EMPLOYEE

Relationships: WORKS-IN, $dg(\text{WORKS-IN}) = 2$,
with participating entities DEPARTMENT, EMPLOYEE

Role functions: employee: WORKS-IN $\rightarrow$ EMPLOYEE,
department: WORKS-IN $\rightarrow$ DEPARTMENT

Thus, the application

$$Cmax(\{\text{department}\}, \text{WORKSIN}, \{\text{employee}\})$$

would return, say, n, meaning that a given instance of DEPARTMENT might be related to at most n occurrences of EMPLOYEE. Similarly,

$$Cmin(\{\text{employee}\}, \text{WORKS-IN}, \{\text{department}\})$$

and

$$Cmax(\{\text{employee}\}, \text{WORKS-IN}, \{\text{department}\})$$

might return 1 and 1, saying that at least and at most one instance of DEPARTMENT should be related to a given instance of EMPLOYEE through the relationship WORKS-IN.

In fact, binary relationships pose no problem for cardinality constraints and would not require such gothic expressions. However, if we admit multiple entity relationships, such that $dg(R) \geq 2$, it becomes a bit more difficult to express them in a neat framework.

Weak entities

There is a special kind of entity sets, called *weak entities*, such that the existence and identification of its components depend upon the instances of other entity types, through a given relationship. The classical example is the entity DEPENDENT, related somehow to the entity EMPLOYEE. In this model, we might consider that the occurrences of DEPENDENT only make sense and have meaning from the existence of EMPLOYEE, and that dependents can only be identified by using some key of EMPLOYEE. See Figure 1.

The function $Weak$, when applied to weak entity set symbols, returns the navigation path to the key in the associated entity. Let E_r and E_w be entities, and R a relationship with participating entities E_r and E_w, and role functions ρ_r and ρ_w respectively, such that $Cmax(\{\rho_w\}, R, \{\rho_r\}) = 1$ and $Cmin(\{\rho_w\}, R, \{\rho_r\}) = 1$. Notice that this guarantees that there is one element in the instance of E_r related to some element in the instance of E_w through R. Thus, any key for E_w should include some reference to E_r. For an object o_w in the instance of E_w, the corresponding instance o_r could be reached through the path

$$\rho_r(\rho_w^{-1}(o_w)) = o_r$$

where ρ_w^{-1} is the inverse function of ρ_w. Notice that ρ_w^{-1}, by the $Cmax$ expression above, is always a function.

E-R SCHEMATA AND INTERPRETATIONS

A E-R schema S is a quintuple (**E**, **R**, **A**, **V**, Θ), where **E** is the set of entity sets, **R** is the set of relationships, **A** is the set of attribute symbols, **V** is the set of value sets and Θ is the set of integrity constraints in S.

An interpretation[1] **I** for an E-R Schema **S** must characterize the following:

(i) the instance of the set $O = \{o_1, \ldots, o_n\}$, which is the set of abstract objects in **I**;

(ii) the object of the interpretation, which is a set of assertions of the form $X(o_i)$, where X is either an entity set or a relationship;

(iii) for each value set V, defined as $V = \{c_1, \ldots, c_n\}$, the instance of V is the set $\{\delta(c_1), \ldots, \delta(c_n)\}$. δ is defined below.

(iv) for each entity set E, the instance of E;

(v) for each relationship R, the instance of R. The instance of an entity set/relationship X is the set of abstract objects $\{o_1, \ldots, o_n\}$, such that o_i is in that set iff the assertion $X(o_i)$ is in $\mathbf{I}$;

(vi) the denotation of a term t, here expressed as $\delta(t)$, which is an object in the domain of the referred term. According with the sort of the term, its denotation is:

- for a constant c, the term denotes some object in the domain of the interpretation.
- for an attribute defined as $\alpha_i : X, I \rightarrow V$, the term $\alpha_i(o, I)$ denotes the object in the instance of V assigned by α_i to its sequence of arguments. If o is not in the instance of X, α_i assigns the null value;
- for a role function defined as $\rho_i : R \rightarrow E$, the term $\rho_i(o)$ denotes the object in the instance of E assigned by ρ_i to its sequence of arguments;

A term t is said to be an E-term if it always evaluates to some abstract object in O. t is said to be a C-term otherwise.

Formulae

In an interpretation $\mathbf{I}$, formulae evaluate to 1 (truth) or 0 (falsity) in conformity with their sorts and usual rules for logical expressions.

Queries

A query Q is an expression of the form

$$t_1, \ldots, t_n \mid v_1 \in X_1, \ldots, v_k \in X_k \mid \theta(v_1, \ldots, v_k)$$

where t_i is either a C-term or an E-term, X_i is either an entity set or a relationship and θ is a wff. The only free variables occurring in $t_1, \ldots, t_n$ and θ are $v_1, \ldots, v_k$. n is the number of components of Q.

3. ORDERING

Ordered access is an important feature in most database implementations. In this section, we lay the basic formulation for a general treatment of ordering, that will be applied later in Section 4.

Obtaining ordered access to database elements generally involves

a) a domain, which is a set of entities/relationships whose elements are to be ordered;

b) a selection expression, which is a well-formed formula that selects elements from the domain above;

c) the determinants of the ordering, which are terms over the elements of the domain that determinate their final ordering criteria.

For example, suppose the following ordering:

Employees in ascending alphabetical order

over some entity EMPLOYEE with attribute *Name* (see Figure 1). We would then have

a) Domain: EMPLOYEE;

b) Selection expression: $\{e|e \in \text{EMPLOYEE}\}$;

c) Determinants: $Name\Uparrow$.

Here, $\Uparrow$ denotes *ascending* and $\Downarrow$ denotes *descending*. As a second example, suppose now the ordering

Dependents of employee e in ascending alphabetical order

The entities in EMPLOYEE are linked to their dependents through the relationship DEPS. We have then

a) Domain: DEPENDENT;

b) Selection expression:

$$\{\ d \mid d \in \text{DEPENDENT} \mid \exists r \in \text{DEPS}\ \ employee(r) = e\ \ \wedge\ \ dependent(r) = d\ \}$$

c) Determinants: $Name\Uparrow$

Notice that, this time, e appears as a free variable in the selection expression, cattering for the employee's identification, since the dependents should be ordered according with their corresponding element e in EMPLOYEE. In order to keep the example simple, we admit just one component in the target list of the selection expression. This way, the terms in the determinant list refer always to the first and only component of the domain of the ordering. To generalize this concept, we would need only to admit a notation for identifying the i-th component of the target list when defining the determinant terms.

We can now state the general form for the treatment of ordered accesses in our approach. We shall treat orderings as the result of functions that, applied to ER-Schema interpretations, return ordering relations that can be accessed by the data manipulation operations we define later in Section 4.

The application of the general ordering function f is of the form

$$f : (Q(v_1, ..., v_k), L, (t_1, ..., t_n)) \rightarrow \text{ORD}$$

where

a) Q is a query that defines the domain of the ordering and contains the selection expression over that domain. The only free variables in Q are $v_1, ..., v_k$;

b) L is the list of determinant terms, of the form $t_1\nabla_1, ..., t_n\nabla_n$, whose only free variables have the instance of Q as domain. $\nabla_1, ..., \nabla_n$ indicate the direction of the ordering, whether $\Uparrow$ (*ascending*) or $\Downarrow$ (*descending*);

c) $t_1, ..., t_k$ are the arguments of the ordering, replacing the occurrences of $v_1, ..., v_k$ in Q for its evaluation.

d) ORD is an ordering relation. The instance of an ordering relation is a set of tuples of the form $[o_p, o_n]$, such that o_p immediately precedes o_n in the ordering. For the first and last elements in the ordering, o_p and o_n should be ϕ respectively.

Going back to the second example above, we would have

$$f(Q(v), [Name \Uparrow], (e))$$

When we replace e in Q, we have the expression that returns the dependents related to e through R.

Database management systems, in general, do not offer such wide possibilities for ordering instances. So, we can simplify the expression by creating ordering functions $f_1, ..., f_n$, such that Q and L are implicit and characteristic to each function f_i. Only the arguments of the ordering would be kept external to the ordering functions. Thus, for the second example above, we might define a function *Dependents.Alphabetical*, with argument EMPLOYEE, such that the application

$$Dependents.Alphabetical(e)$$

would return the ordering relation for the set of elements of DEPENDENT related to the occureence e in EMPLOYEE. The expression

$$\{\ d\ |d \in \text{DEPENDENT} \mid \exists r \in \text{DEPS}\ \ employee(r) = e\ \ \wedge\ \ dependent(r) = d\ \}$$

would be implicit in the *Dependents.Alphabetical* definition, as well as the ordering criteria expressed as $Name \Uparrow$, corresponding to the L element for the general ordering function f.

4. DATA MANIPULATION OPERATIONS

After formally introducing E-R schemata, interpretations and orderings, we have finally come to the data manipulation operations themselves. The operations over E-R instances can be grouped into five main classes:

a) expression evaluation, which refers to the evaluation of logical expressions;

b) C-terms denotation, comprehending attribute evaluation and the manipulation of constants;

c) E-terms denotation, comprehending role functions and the manipulation of entity and relationship instances;

d) find operations, including all the various kinds of entity/relationship location operations;

e) update operations, regarding inclusion and removal of assertions in the object of the interpretation, as introduced in Section 2.

The next subsections detail each of the operations. In all cases, operations refer to the interpretation **I** of an E-R schema **S**. X represents either an entity set or a relationship.

TERM EVALUATION

Recalling Section 2, the denotation of a term t, expressed by $\delta(t)$, is an object in the domain of the referred term.

Definition 1. *The operator $\mathcal{T}$, when applied to a term t, returns $\delta(t)$.*

EXPRESSION EVALUATION

Expression evaluation is needed for query evaluation and other condition tests that might be required. The most usual case is the containment test, that is, $X(t)$, where it is necessary to know whether t belongs to the instance of X or not. The formal treatment employeed here, however, allows for a generic well-formed formula θ.

Definition 2. *Let $\theta(v)$ be a wff whose only free variable is v. The operator $\mathcal{B}$ when applied to an expression of the form $\theta(t/v)$ returns the value 1 (truth)*

iff θ evaluates to 1 when $T(t)$ substitutes all the occurrences of v in θ. The application returns 0 (falsity) otherwise.

FIND OPERATIONS

The $\mathcal{F}$ operator when applied to an entity set or relationship returns either an abstract object or the null value, either denoting the wanted element or indicating the failure of the operation.

There are two basic kinds of search: keyed and ordered. In the first, a key is referenced, along its denotation, and the corresponding object is returned. In the second case, an ordering relation is considered, together with the anchoring element, and the next or previous element in the ordering is returned.

Definition 3. *Let κ be a key for X, such that $\kappa(X) = \{\eta_1, ..., \eta_\ell\}$, where η_i is each of the components of κ. η_i can be an attribute, role function or key-path for X. The application of the operator $\mathcal{F}$ to an expression of the form $(\kappa, t_1, ..., t_\ell)$ returns o, such that $\delta(\eta_i(o)) = \delta(t_i)$, $i = 1, \ell$, where o is in the instance of X.*

By the definition above, the operator $\mathcal{F}$ locates the element o such that the evaluation of its κ key components is equal to the evaluation of its arguments $t_1, ..., t_\ell$. By the schema constraints derived from κ, there is one and only one o that satisfies the condition.

Definiton 4. *Let* ORD *be an ordering relation resulting from the application of an ordering function. The application of the operator $\mathcal{F}$ to an expression of the form* (ORD, o, Δ), *where* $\Delta \in \{\text{PREVIOUS}, \text{NEXT}\}$, *returns o_p, if* $\Delta = \text{PREVIOUS}$ *and $o = o_n$, or o_n if* $\Delta = \text{NEXT}$ *and $o_p = o$, iff there is a tuple $[o_p, o_n]$ in the instance of* ORD.

The definition above caters for the cases FIRST and LAST as well, since o_p and o_n can take the null value, indicating the beginning and the end of an ordering sequence. Notice that ORD results from the application of some ordering function f_i. Since the f_i functions incorporate the Q and L components of the original ordering function f, we can state a more general form for the $\mathcal{F}$ operator, which is

$$\mathcal{F}(f_i(t_1, ..., t_n), o, \Delta)$$

where $t_1, ..., t_n$ are the arguments of the ordering function f_i.

UPDATE OPERATIONS

There are basically three kinds of updating operations: insertion and deletion of entity/relationship occurrences and assignment of attribute values.

Insertion

The operator $\mathcal{U}_i$ is used for inserting new occurrences of entities/relationships. These appear in the form of assertions in the interpretation **I** of a schema **S**. For relationships, the inclusion of a new occurrence implies the assignment of its role function values.

Definition 5. *The operator $\mathcal{U}_i$ applied to an entity set E results in the inclusion of the assertion $E(o)$ in the object of the interpretation* **I**, *such that $o \in O$. If $\beta(E) = E_i$, then $\mathcal{U}_i(E_i)$ also applies, recursively.*

Definiton 6. *The operator $\mathcal{U}_i$ applied to an expression $(R, o_1, ..., o_{dg(R)})$, results in the inclusion of the assertion $R(o)$ in the object of the interpretation* **I**, *such that $o \in O$ and for no other X, $X(o)$ is in* **I**. *For each role function ρ_i associated to R, $\mathcal{T}(\rho_i(o)) = o_i$, $i = 1, dg(R)$.*

Deletion

Deleting an occurrence of an element of an entity/relationship is to remove the corresponding assertions from **I**. For relationships, the operator results also in the de-assignment of the role functions denotations.

Definition 7. *The operator $\mathcal{U}_d$ applied to an expression (o, X) results in the removal of the assertion $X(o)$ from the object of the interpretation* **I**.

Assignment

Attributes can have their denotations re-assigned by the operator $\mathcal{U}_a$.

Definiton 8. *The application of the operator $\mathcal{U}_a$ to an expression of the form (α_i, I, o, c_i) makes $\mathcal{T}(\alpha_i(o, I)) = c_i$.*

5. IMPLEMENTATION ASPECTS

The sections above treat the operations in a generic framework, not always efficiently implementable. This section comments on some restrictions that the author suggests in order to achieve feasible implementations.

The basis of entities and relationships instances are the abstract objects. which can be easily implemented using the concept of internally generated surrogates. Any instance identification should be carried out through those surrogates, rather than the defined keys. For the representation of generalizations, surrogates can also be helpful, providing a way for the correspondence between the instances in the type and its supertype.

Cardinalities must be simplified mainly because in real cases they are rarely used at the full extent. We think its sufficient to allow for expressions of the form

$$Cmax(\{\rho_i\}, R, \{\ .\rho_j.\ \}), \quad j \quad 1, dg(R); j \neq i$$

That is, we relate each of the participating entities of a relationship R, through its role function, with all the other participating entities, also through their role functions. As a result, each connection of R in the E-R diagram is labelled with just one value for $Cmax$. The same should apply to $Cmin$.

Schema constraints fall in a rather difficult problem if any kind of formula is allowed. However, at least the schema properties such as cardinalities, generalizations, keys and weak entities should be fully implemented. Other classes of constraints, especially the ones involving just one variable could also be considered.

Another question regards keys and attributes with more than one occurrence. Although there is no serious problem in their association, this is not common practice and perhaps keys involving multi-valued attributes should be avoided whenever possible.

A further point regards orderings. If a general query evaluator is not present, the database designer should restrict the determinants and expressions of the queries of the orderings in a way that they could be easily pre-stored in the database and be treated as indexes. For example, for an entity, only its attributes could be allowed as determinants and the query should always consider all the instances of the entity. For a relationship, only its own attributes plus the attributes of the participating entities. That sort of restriction permits that the ordering relations be kept by the data management system quite simply during update operations.

6. CONCLUSIONS

In this paper we presented a basic set of data manipulation operations for the E-R model. The operations were formally defined and configurate a sort of basic kit for implementing data access to an E-R based system.

This work is part of a project intended to provide a complete E-R environment for database users, extending from a physical implementation to a graphical query interface. For this reason, the main approach here is simplicity. The E-R model was taken in its simpler form, although an extension for generalization abstraction was included. The idea is to keep the model with features that are easily implementable, while further extensions should be brought later.

7. REFERENCES

1. BOOLOS, G. S. and JEFFREY, R. C., *Computability and Logic*, Cambridge University Press, 1980.
2. CHEN, P., The Entity-relationship Model - Toward a Unified View of Data, *ACM Transactions on Database Systems 1*, 1(1976).
3. JUNET, M., FALQUET, G. and LEONARD, M., ECRINS/86: An Extended Entity-Relationship Data Base Management System and its Semantic Query Language, *Proceedings of the Twelfth Internacional Conference on Very Large Databases*, Kyoto, Japan, 1986.
4. LENZERINI, M. and SANTUCCI, G., Cardinality Constraints in the Entity-Relationship Model, *Proceedings of the Third International Conference on Entity-Relationship Approach*, California, USA, 1983.
5. MALHOTRA, A. et al, An Entity-Relationship Programming Language, *IEEE Transactions on Software Engineering 15*, 9(1989).
6. PECKHAM, M. and MARYANSKI, F., Semantic Data Models, *ACM Computing Surveys 20*, 3(September 1988).
7. SETZER, V. W., *Projeto Lógico e Projeto Físico de Bancos de Dados*, V Escola de Computação, Belo Horizonte, 1986.
8. SILVEIRA, P. M., A Formalization of The E-R Model, *Proceedings of the IX Conferencia Internacional de la Sociedad Chilena de la Ciencia de la Computation*, Santiago, Chile, 1989.
9. SILVEIRA, P. M., Definindo e Utilizando Bancos de Dados com o Modelo Entidade-Relacionamento, *Anais do XXIII Congresso Nacional de Informática*, Rio de Janeiro, 1990.