Proc. 5th Int. Conf. and Workshop on Database and Expert Systems
Applications (DEXA'95), N. Revell (Ed.), ONMIPRESS, San Mateo, CA (1995)

Towards Schema Queries for Semantic Data Models*

Martin Gogolla
Bremen University
Department for Mathematics and Informatics
Database Systems Group
Postfach 330440, D-28334 Bremen, GERMANY
e-mail: gogolla@informatik.uni-bremen.de

Abstract

We contribute to metadata management by giving a specification of extended Entity-Relationship schemas with the extended Entity-Relationship model itself. We formulate a number of integrity constraints describing the context-sensitive requirements an extended Entity-Relationship schema has to fulfill. By doing this, a certain class of schema queries can be formulated.

Key words: *Semantic data model, formalization of data models, metadata specification, schema queries.*

1 Introduction

Information systems and databases are designed to store *data* items representing some slice of reality, the so-called universe of discourse. Within the information system community the notion of *metadata* usually refers to something which is not part of the represented universe of discourse but which describes the structure of the stored data items. Approaches to metadata management have been studied in the context of the (extended) relational [AN86, MR86] and Entity-Relationship data model [Alt85], within the area of knowledge representation [AZG88, Hsu91] and re-engineering [JJ94], and also in connection with object-oriented databases [McL88, HM90, TS92]. However, as far as we know, there is no rigorous approach to metadata descriptions employing a semantic data model [STW84, HK87, PM88] in order to specify the data model with its own language features and in order to formulate schema queries. But, from the very beginning of computer science there was an enormous interest in describing languages by the languages themselves. One of the first examples for this was the description of LISP within LISP [McC60].

Here, we employ an extended Entity-Relationship model [HNSE87, EGH$^+$92] as a typical representative of a semantic data model. The model offers means for (1) user-defined data types, (2) specialization and generalization abstractions, (3) object- and multi-valued attributes, and (4) representation of null values. The model comes along with a language [HG88, GH91] to formulate integrity constraints in order to restrict database states to consistent ones. The same language can also be used to query the database. Both, the model and the constraint and query language, have a precise mathematical semantics [HG88, GH91, Hoh93, Gog94].

*Work reported here has been partially supported by the CEC under Grant No. 6112 (COMPASS).

Our contribution to metadata management is a specification of this Entity-Relationship model with the Entity-Relationship model itself, i.e., we specify the metadata of a semantic data model schema. By doing this, a certain class of schema queries can be formulated against Entity-Relationship schemas. A formulation of another Entity-Relationship model within itself can be found in [TNCK91]. A similar approach like ours but one which specifies the concepts of a data model completely in a formal way has appeared in [LM91].

The structure of our paper is as follows. In Sect. 2 we give a short introduction to the data model we use. In Sect. 3 we discuss an Entity-Relationship schema with appropriate integrity constraints for the data model introduced before. The paper closes in Sect. 4 with some concluding remarks.

2 Sketching the Entity-Relationship Model

Here, we can only give a glimpse of our extended Entity-Relationship model and therefore just mention the essential features by means of an example. More elaborate descriptions of our model can be found for instance in [GH91, EGH$^+$92, Hoh93, Gog94]. Consider the rather abstract extended Entity-Relationship diagram in Fig. 1 and have a closer look at the non-dotted parts.

Entity Types: We identify the basic concept of an entity type depicted as a rectangle, for example the entity type E6.

Data-valued Attributes: In order to condense the presentation we have not depicted any data-valued attributes for entity types, but one can imagine that there are attributes like name, color, age, or geometry which either belong to standard data types like integer or string or to user-defined, application dependent data types, for instance, geographical data types like point, circle, line, or polygon.

Object-valued Attributes: This diagram does not involve a classical relationship type, but offers more advanced features connecting entity types, for instance the object-valued attribute A5 belonging to entity type E6 which yields entities of type E7. The attribute A5 is therefore regarded as a function $\mathsf{A5} : \mathsf{E6} \rightarrow \mathsf{E7}$.

Type Constructions: The depicted triangle between entity types E7, E9, and E10 is a type construction called C2; E9 and E10 are the input types (the types on the base line), and E7 is the output type (the type on the opposite side of the base line). This type construction expresses that entities of type E9 and E10 are generalized to entities of type E7. Thus each E7 entity is either an E9 or an E10 entity.

Multi-valued Attributes: Between E9 and E10 we find a multi-valued, in this case list-valued attribute A8; multi-valued attributes may additionally be set- or bag-valued. Thus attribute A8 can be regarded as a function which, when applied to an E10 entity, gives a list of E9 entities $\mathsf{A8} : \mathsf{E10} \rightarrow \mathsf{list(E9)}$.

Entity-Relationship Calculus: Our Entity-Relationship model comes along with a language for specifying integrity constraints and queries for ER database schemas [HG88, GH91]. This calculus provides an orthogonal SQL-like query facility which again can be used in constraints. For example, the requirement that every E9-entity must occur in an A8-list could be expressed as:[1]

```
BTS [ SELECT e9 FROM (e9:E9) ]
```

[1] Queries in this calculus yield multi-sets (or bags) as results. Duplicates are removed by applying the standard function BTS.

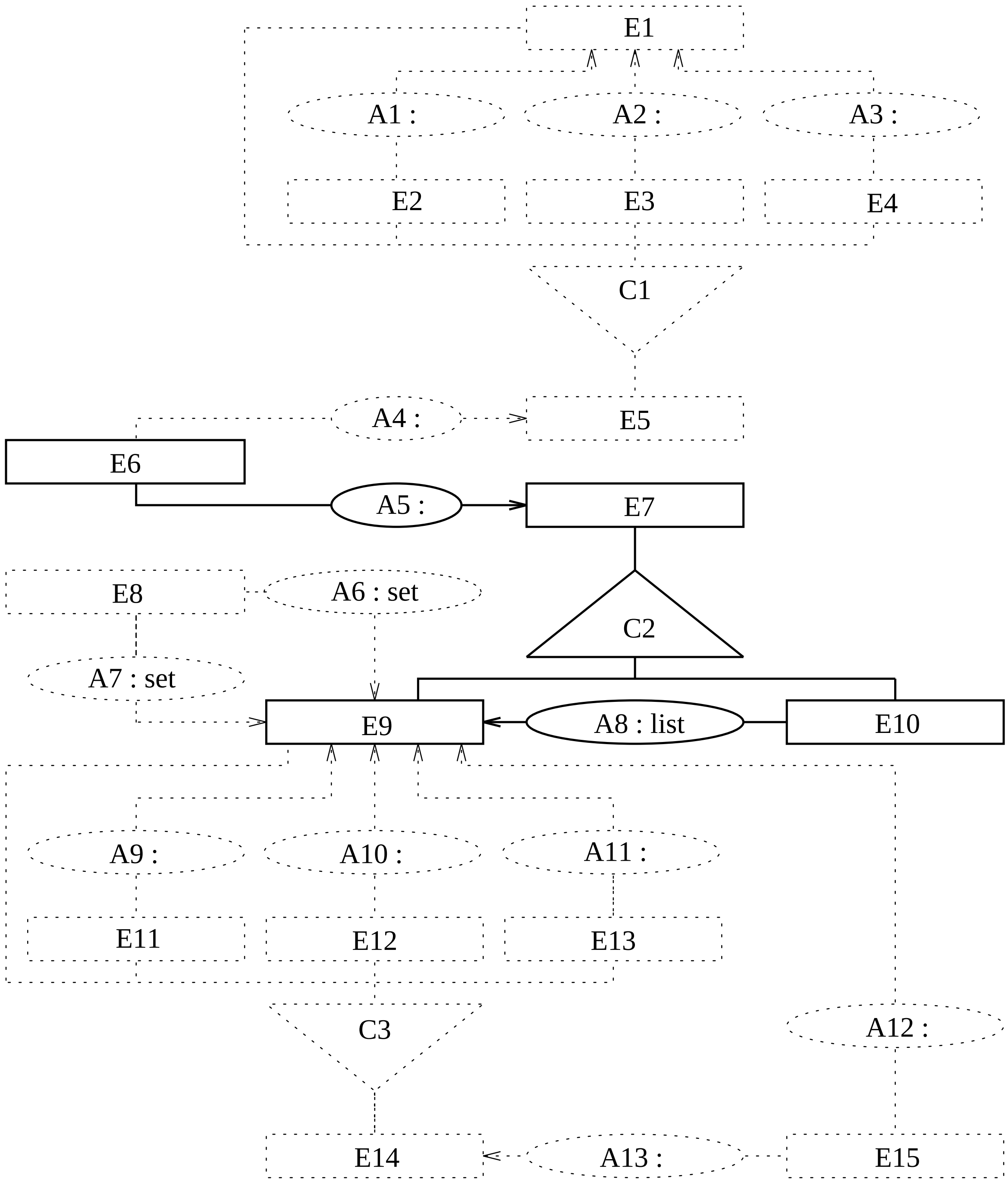

Figure 1: Sample Extended Entity-Relationship Schema.

```
=
BTS [ SELECT e9
      FROM (e10:E10), (e9:A8(e10)) ]
```

This very rough and abstract description of the extended Entity-Relationship model should give a first impression of the features offered. For a more comprehensive introduction we refer to [HNSE87, HG88, GH91, EGH+92, Hoh93, Gog94].

3 A Meta-Schema for the Data Model

We now specify our Entity-Relationship schemas by means of the Entity-Relationship model. In Fig. 2 entity types are introduced for names of entity and relationship types, for data sorts, attributes (data-valued), components (object-valued attributes), and type constructions. Participants of relationship types, source and destination of attributes, etc. are described by object-valued attributes.

In Fig. 2 only entity types, relationship types, type constructions and components are specified. The attributes are not mentioned. The entity types DATA, ENTTYPE, ATTRIBUTE, RELTYPE, CONSTRUCTION, and COMPONENT all have an attribute name of data type string.

A number of integrity constraints describing the conditions of the formal definition of extended Entity-Relationship schemas [HG88, GH91] can be formulated. These conditions constitute the context-sensitive requirements an extended Entity-Relationship schema has to fulfill and are depicted in Fig. 3.

1. The attribute name uniquely determines data types, entity types, attributes, relationship types, type constructions, and components.

   Analogous conditions have to be formulated for attributes, relationship types, type constructions, and components.

2. Names of data types, entity types, attributes, relationship types, type constructions, and components must also be globally unique, i.e., it is, for example, not allowed for an entity type and an attribute to have the same name.

   Analogous conditions have to be formulated for attributes, relationship types, and type constructions.

3. Input and output sets of type constructions are not empty.

4. A constructed entity type is an output type for exactly one type construction.

5. Type constructions are acyclic.

   This constraint refers to a derived relationship connection with participants <ENTTYPE, ENTTYPE> which is defined by

   connection(e_{in},e_{out}) :⇔
   EXISTS (con : CONSTRUCTION)
     (e_{in} IN input(con) AND
       e_{out} IN output(con))

   The notation connection$^+$ refers to the transitive closure of the derived relationship connection. For details see [Gog94].

6. Entities in DATA# or TYPE are really used for attributes.

   This constraint is a technical requirement. It is necessary because of the existence of the type constructions cons-data and cons-type. It has no counterpart in the formal definition of extended Entity-Relationship schema.

We will now describe in Fig. 4 one concrete database state for the above extended Entity-Relationship schema, namely the database state corresponding to the diagram in Fig. 1.

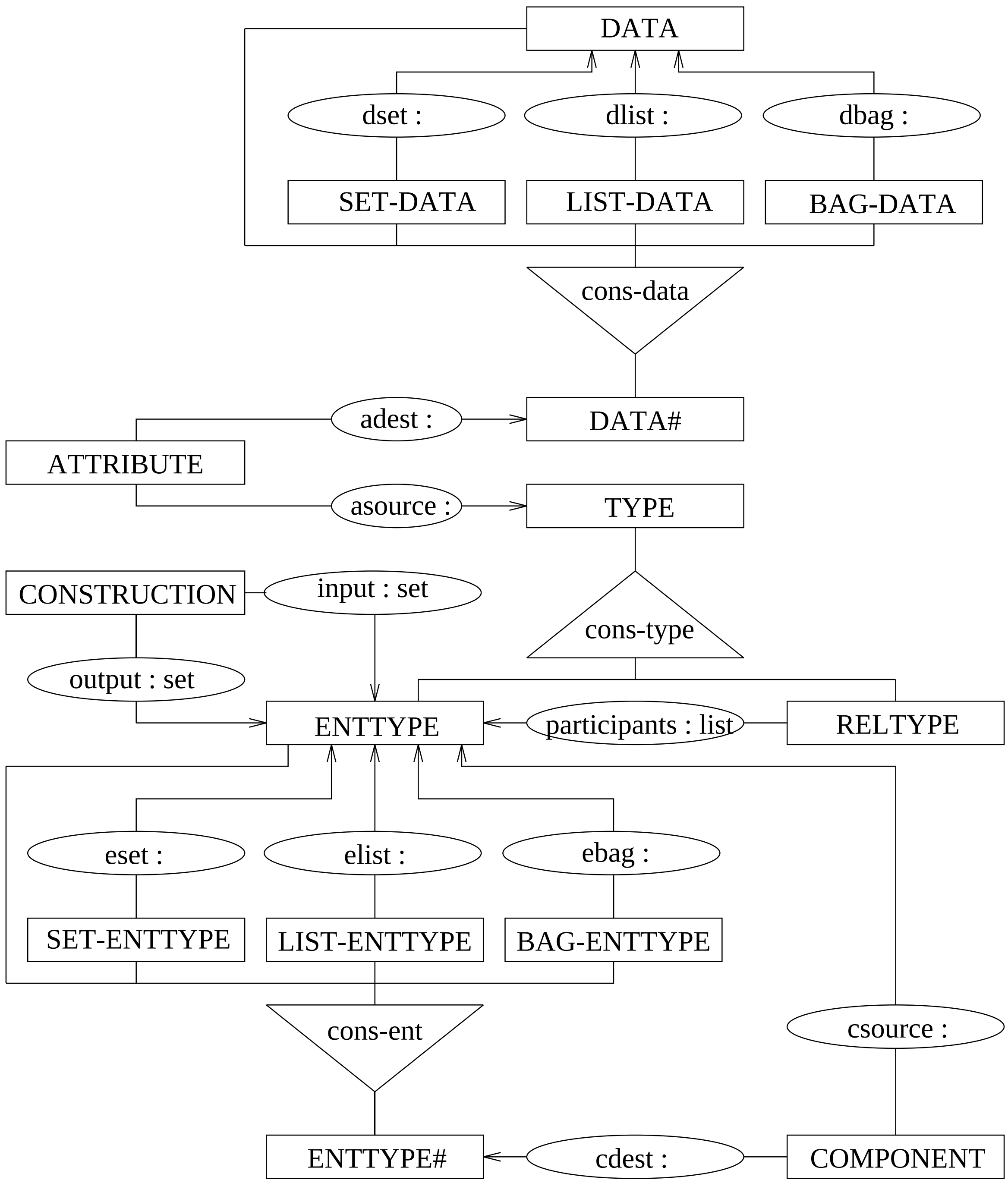

Figure 2: Extended Entity-Relationship Meta-Schema.

1. FORALL (d_1, d_2 : DATA) (name(d_1)=name(d_2) IMPLIES d_1=d_2)
   FORALL (e_1, e_2 : ENTITY) (name(e_1)=name(e_2) IMPLIES e_1=e_2)
   ...

2. FORALL (d : DATA) (e : ENTTYPE) (a : ATTRIBUTE)
   FORALL (r : RELTYPE) (con : CONSTRUCTION) (com : COMPONENT)
   name(d) $\neq$ name(e) AND name(d) $\neq$ name(a) AND
   name(d) $\neq$ name(r) AND name(d) $\neq$ name(con) AND
   name(d) $\neq$ name(com)

   FORALL (e : ENTTYPE) (a : ATTRIBUTE) (r : RELTYPE)
   FORALL (con : CONSTRUCTION) (com : COMPONENT)
   name(e) $\neq$ name(a) AND name(e) $\neq$ name(r) AND
   name(e) $\neq$ name(con) AND name(e) $\neq$ name(com)
   ...

3. FORALL (con_1, con_2 : CONSTRUCTION)
   (EXISTS (e : ENTITY) (e IN output(con_1) AND e IN output(con_2)))
   IMPLIES (con_1=con_2)

4. FORALL (con_1, con_2 : CONSTRUCTION)
   (EXISTS (e : ENTITY) (e IN output(con_1) AND e IN output(con_2)))
   IMPLIES (con_1=con_2)

5. FORALL (e : ENTITY) (NOT $connection^+$(e,e))

6. FORALL (d# : DATA#) EXISTS (a : ATTRIBUTE) adest(a)=d#
   FORALL (t : TYPE) EXISTS (a : ATTRIBUTE) asource(a)=t

Figure 3: Constraints for Extended Entity-Relationship Schemas.

We will employ the following sets and functions where the Greek letter μ in the descriptions presented hereafter denotes the semantics (or meaning) of an entity type, relationship type, etc. Here, it refers to one concrete database state which describes the Entity-Relationship diagram of Fig. 2.

The interpretation of the other parts should be clear from the diagram in Fig. 1. You have observed that the Entity-Relationship diagrams in Fig. 1 and Fig. 2 are isomorphic. Hopefully, we have used the abstract version of the diagram (Fig. 1) for the concrete database state in order to avoid confusion. But of course we could have also used the other diagram (Fig. 2) by changing some identifiers only. For example, the above "abstract" constraint

BTS [SELECT e9 FROM (e9:E9)]
=
BTS [SELECT e9
FROM (e10:E10), (e9:A8(e10))]

would be translated into the following "concrete" constraint demanding that all entity types must participate in at least one relationship type.[2]

BTS [SELECT e FROM (e:ENTTYPE)]
=

[2] We include this constraint for demonstration reasons only and do not think of it as a proper requirement for our approach.

$$
\begin{array}{l}
\mu(\text{ENTTYPE}) = \{\ \text{e}_{\text{E1}}, \text{e}_{\text{E2}}, \text{e}_{\text{E3}}, ..., \text{e}_{\text{E13}}, \text{e}_{\text{E14}}, \text{e}_{\text{E15}}\ \} \\
\mu(\text{RELTYPE}) = \emptyset \\
\mu(\text{ATTRIBUTE}) = \{\ \text{name}, ...\ \}
\end{array}
$$

$$
\begin{array}{llcl}
\mu(\text{name}) : & \mu(\text{ENTTYPE}) & \to & \mu(\text{string}) \\
 & \text{e}_{\text{E1}} & \mapsto & \text{`E1'} \\
 & \text{e}_{\text{E2}} & \mapsto & \text{`E2'} \\
 & \text{e}_{\text{E3}} & \mapsto & \text{`E3'} \\
 & & ... &
\end{array}
$$

$$
\mu(\text{COMPONENT}) = \{\ \text{e}_{\text{A1}}, \text{e}_{\text{A2}}, \text{e}_{\text{A3}}, ..., \text{e}_{\text{A11}}, \text{e}_{\text{A12}}, \text{e}_{\text{A13}}\ \}
$$

$$
\begin{array}{llcl}
\mu(\text{csource}) : & \mu(\text{COMPONENT}) & \to & \mu(\text{ENTTYPE}) \\
 & \text{e}_{\text{A1}} & \mapsto & \text{e}_{\text{E2}} \\
 & \text{e}_{\text{A2}} & \mapsto & \text{e}_{\text{E3}} \\
 & \text{e}_{\text{A3}} & \mapsto & \text{e}_{\text{E4}} \\
 & \text{e}_{\text{A4}} & \mapsto & \text{e}_{\text{E6}} \\
 & \text{e}_{\text{A5}} & \mapsto & \text{e}_{\text{E6}} \\
 & & ... &
\end{array}
$$

$$
\mu(\text{CONSTRUCTION}) = \{\ \text{e}_{\text{C1}}, \text{e}_{\text{C2}}, \text{e}_{\text{C3}}\ \}
$$

$$
\begin{array}{llcl}
\mu(\text{input}) : & \mu(\text{CONSTRUCTION}) & \to & \mu(\text{set}(\text{ENTTYPE})) \\
 & \text{e}_{\text{C1}} & \mapsto & \{\ \text{e}_{\text{E1}}, \text{e}_{\text{E2}}, \text{e}_{\text{E3}}, \text{e}_{\text{E4}}\ \} \\
 & \text{e}_{\text{C2}} & \mapsto & \{\ \text{e}_{\text{E9}}, \text{e}_{\text{E10}}\ \} \\
 & \text{e}_{\text{C3}} & \mapsto & \{\ \text{e}_{\text{E9}}, \text{e}_{\text{E11}}, \text{e}_{\text{E12}}, \text{e}_{\text{E13}}\ \} \\
\mu(\text{output}) : & \mu(\text{CONSTRUCTION}) & \to & \mu(\text{set}(\text{ENTTYPE})) \\
 & \text{e}_{\text{C1}} & \mapsto & \{\ \text{e}_{\text{E5}}\ \} \\
 & \text{e}_{\text{C2}} & \mapsto & \{\ \text{e}_{\text{E7}}\ \} \\
 & \text{e}_{\text{C3}} & \mapsto & \{\ \text{e}_{\text{E14}}\ \}
\end{array}
$$

Figure 4: Sample Database State.

```
BTS [ SELECT e
      FROM   (r:RELTYPE),
             (e:participants(r)) ]
```

We now come to the most interesting point in connection with the representation of the extended Entity-Relationship approach within itself. It is now possible to query the above database state corresponding to the Entity-Relationship schema of our example. By so doing, a certain class of schema queries depicted in Fig. 5 can be formulated.

1. "Names of all entity and relationship types"

2. "Names of those relationship types participating with entity type X"

3. "Names of all entity types together with their attributes"

   This query can be formulated in at least two different ways. First, we structure the result by giving the name of the entity type with the collection of names of its attributes.

   Second, we can give nearly the same result but this time in a flat way, i.e., we yield a collection of tuples where each tuple consists of a name of an entity type and a name of a corresponding attribute.

4. "Names and relationships of non-constructed entity types."

1. [SELECT name(e) FROM (e : ENTTYPE)]
   UNION
   [SELECT name(r) FROM (r : RELTYPE))]

2. [SELECT name(r)
   FROM (r : RELTYPE)
   WHERE EXISTS (e : ENTTYPE) (name(e)='X' AND e IN participants(r))]

3. [SELECT name(e),
   [SELECT name(a)
   FROM (a : ATTRIBUTE)
   WHERE ENTTYPE(asource(a))=e]
   FROM (e : ENTTYPE)]

   [SELECT name(e), name(a)
   FROM (e : ENTTYPE), (a : ATTRIBUTE)
   WHERE ENTTYPE(asource(a))=e]

4. [SELECT name(e),
   [SELECT name(r)
   FROM (r : RELTYPE)
   WHERE e IN participants(r)]
   FROM (e : ENTTYPE)
   WHERE NOT(EXISTS (con : CONSTRUCTION) e IN output(con))]

Figure 5: Schema Queries.

4 Conclusion

We have tried to present the very basic ideas of an extended Entity-Relationship model. We employed this data model in order to give a schema for the semantic data model behind this extended Entity-Relationship approach. We gave integrity constraints which, roughly speaking, specified the context-sensitive conditions an extended Entity-Relationship schema has to fulfill. By formulating the data model as a "normal" Entity-Relationship schema we were able express schema queries, a feature which is usually beyond the scope of data models.

References

[Alt85] D.E. Altenkrueger. FAIM, The Formal Association Inference Model, A Means for the Formal Definition of Conceptual Meta Schemas. In IEEE, editor, *Proc. 4th Int. Conf. on Entity-Relationship Approach (ER'85)*, pages 10–17. IEEE, Silver Spring (MD), 1985.

[AN86] M. Adiba and G.T. Nguyen. Handling Constraints and Meta-Data on a Generalized Data Management System. In L. Kerschberg, editor, *Expert Database Systems: Proc. 1st International Workshop*, pages 487–505, 1986.

[AZG88] A. Al-Zobaidie and J. Grimson. Use of Metadata to Drive the Interaction between Database and Expert Systems. *Journal of Infor-*

mation and Software Technology, 30(8):484–496, 1988.

[EGH+92] G. Engels, M. Gogolla, U. Hohenstein, K. Hülsmann, P. Löhr-Richter, G. Saake, and H.-D. Ehrich. Conceptual modelling of database applications using an extended ER model. *Data & Knowledge Engineering, North-Holland*, 9(2):157–204, 1992.

[GH91] M. Gogolla and U. Hohenstein. Towards a Semantic View of an Extended Entity-Relationship Model. *ACM Transactions on Database Systems*, 16(3):369–416, 1991.

[Gog94] M. Gogolla. *An Extended Entity-Relationship Model – Fundamentals and Pragmatics.* Springer, Berlin, LNCS 767, 1994.

[HG88] U. Hohenstein and M. Gogolla. A Calculus for an Extended Entity-Relationship Model Incorporating Arbitrary Data Operations and Aggregate Functions. In C. Batini, editor, *Proc. 7th Int. Conf. on the Entity-Relationship Approach*, pages 129–148. North-Holland, Amsterdam, 1988.

[HK87] R. Hull and R. King. Semantic Database Modelling: Survey, Applications, and Research Issues. *ACM Computing Surveys*, 19(3):201–260, 1987.

[HM90] S. Hong and F. Maryanski. Using a Meta Model to Represent Object-Oriented Data Models. In *Proc. 6th Int. Conf. on Data Engineering (ICDE'90)*, pages 11–19. IEEE Computer Society, 1990.

[HNSE87] U. Hohenstein, L. Neugebauer, G. Saake, and H.-D. Ehrich. Three-Level Specification of Databases Using an Extended Entity-Relationship Model. In R. Wagner, R. Traunmüller, and H. C. Mayr, editors, *Proc. GI-Fachtagung "Informationsermittlung und -analyse für den Entwurf von Informationssystemen"*, pages 58–88, Linz, 1987. Informatik-Fachbericht 143, Springer, Berlin, 1987.

[Hoh93] U. Hohenstein. *Formale Semantik eines erweiterten Entity-Relationship-Modells.* Teubner, Stuttgart, 1993.

[Hsu91] C. Hsu. The Metadatabase Project at Rensselaer. *ACM SIGMOD Record*, 20(4):83–90, 1991.

[JJ94] M.A. Jeusfeld and U.A. Johnen. An Executable Meta Model for Re-Engineering of Database Schemas. In P. Loucopoulos, editor, *Proc. 13th Int. Conf. on Entity-Relationship Approach (ER'94)*, pages 533–547. Springer, Berlin, LNCS 881, 1994.

[LM91] P.C. Lockemann and G. Moerkotte. On the Notion of Concept. In T.J. Teorey, editor, *Proc. 10th Int. Conf. on Entity-Relationship Approach*, pages 349–370. ER Institute, Pittsburgh (CA), 1991.

[McC60] J. McCarthy. Recursive Functions of Symbolic Expressions and Their Computation by Machine. *Communication of the ACM*, 3(4):184–195, 1960.

[McL88] D. McLeod. A Learning-Based Approach to Meta-Data Evolution in an Object-Oriented Database. In K.R. Dittrich, editor, *2nd International Workshop on Object-Oriented Database Systems*, pages 219–224, 1988.

[MR86] L. Mark and N. Roussopoulos. Metadata Management. *IEEE Computer*, 19(12):26–36, 1986.

[PM88] J. Peckham and F. Maryanski. Semantic Data Models. *ACM Computing Surveys*, 20(3):153–189, 1988.

[STW84] M. Schrefl, A M. Tjoa, and R.R. Wagner. Comparison Criteria for Semantic Data Models. In C.V. Ramamoorthy, editor, *Proc. Int. Conf. on Data Engineering*, pages 120–125. IEEE, Silver Spring (MD), 1984.

[TNCK91] A.K. Tanaka, S.B. Navathe, S. Chakravarthy, and K. Karlapalem. ER-R: An Enhanced ER Model with Situation-Action Rules to Capture Application Semantics. In T.J. Teorey, editor, *Proc. 10th Int. Conf. on Entity-Relationship Approach*, pages 59–75. ER Institute, Pittsburgh (CA), 1991.

[TS92] M. Tresch and M.H. Scholl. Meta Object Management and its Application to Database Evolution. In G. Pernul and A.M. Tjoa, editors, *Proc. 11th Int. Conf. on Entity-Relationship Approach (ER'92)*, pages 299–321. Springer, Berlin, LNCS 645, 1992.