A CATEGORICAL VIEW OF DATABASES

Maria Zamfir Bleyberg

Computing and Information Sciences Department

Kansas State University, Manhattan, KS 66506-2302, USA

Abstract

It is extremely difficult to compare and combine the modeling capabilities of the various database models in use. The differences in the structure of data are perhaps the hardest to be reconciled. Only a database model at a high level of abstraction can bring together different database models and offer a common structure for describing data. This paper presents the *categorical entity-relationship* (CER) model, which is a highly abstract model for databases. This model (i) provides a structure for describing data with a clear separation between *syntax* and *semantics*; (ii) specifies a set of operations to manipulate the data, i.e.,a CER algebra, that supports a flexible representation of a database (a *view* facility) and a *declarative* query language; and (iii) incorporate the most important features of many existing database models, such as the relational, entity-relationship, and object-oriented models, creating a new perspective for these models.

Keywords: schemata, databases, natural sources, natural transformations

1 Introduction

Several data models have been created to describe databases at different levels of abstraction. These models include various logic-based [10] and object-oriented models [1, 9], the *entity-relationship* [2] and the *semantic* [6] models, and the models which more closely pattern a database designers' final product such as the *relational* and *network* [10] models. The emphasis on relational systems reflects the fact that they have provided good answers for business applications. The emphasis on the semantic and entity-relationship models comes from the need to accurately model data relationships in the design of conceptual schemata. The recent interest in object-oriented database systems is a consequence of the requirements arising new application areas of databases, such as office information systems and computer-aided design.

The formal foundations of the various database models reflect the specific needs of each model. Therefore, it is difficult to compare and combine their modeling capabilities. There is a consensus that a database "meta-model" with a mathematical foundation may bridge the gap between different database models and may provide a better understanding of these models. This is the purpose of the present paper; we use category theory to define a highly abstract model for databases, the *categorical entity-relationship* (CER) model. This model (i) provides a structure for describing data with a clear separation between *syntax* and *semantics*; (ii) specifies a set of operations to manipulate that data, i.e., a CER algebra, that supports a flexible representation of a database and a *declarative* query language; and (iii) incorporate the most important features of many existing database models, such as the relational, entity-relationship, and object-oriented models, creating a new perspective for these models. (Categories originally arose in mathematics out of the need of a formalism to describe the passage from one type of mathematical structure to another with an appropriate degree of generality.) Perhaps the main benefit of the CER model is in the common framework it creates. This framework allows the reconciliation of the differences in the structures of data of various data models.

Our model is based on the natural view of a database as a collection of *entities* (data items) connected by various *relationships*; entities have properties, called *attributes*, that are in a *dependency relation* R. In searching for a formal representation for entities and relationships at a high level of abstraction, we have considered the fact that many database models use graphs in the design of a conceptual database scheme: the entity-relationship models use entity-relationship diagrams; the functional dependencies in the relational models can be viewed as directed graphs; directed graphs are also used in object-oriented databases to represent inheritance. One might hope that graphs would be sufficient to represent the underlying structures of entities and relationships; unfortunately they are not. Although graphs could be used to represent the syntactic features of entities and relationships at an intuitive level, they cannot support a formal definition of the syntax. Goguen, Thatcher, Wagner, and Wright [8] in their work on initial algebra semantics propose that syntax should be an initial object in a category of algebraic models of the syntax. (An interpretation of the syntax is given by the unique homomorphism from the initial algebra to another algebra in the category.) Therefore, the claim that a particular graph gives the syntactic structure of an object must be substantiated by showing that the graph is initial in a certain category. Graphs present another problem: it is difficult to define operations for composing them. All these problems can be solved if we consider categories instead of graphs. (For every graph there is a free category over that graph.) In a category, there are many natural object constructors, such as products and pullbacks. Certain categories have initial objects. The objects of our particular category are certain comma categories, which represent entities and *isa* relationships. We choose comma categories because keys are inherent in these particular categories. Our particular category has an initial object. The extension of graphs to categories also allows us to introduce normal forms as natural transformations.

More precisely, in our model, an entity is represented as an ER-*instance* of an ER-*class*. ER-instances are constructed as comma categories from a given dependency relation R; R represents the *sort* (*type*) of an ER-instance. An ER-class is a category, which represents a collection of similar entities, i.e., entities that have the same structure (created by R). The *identity* of an ER-instance is established through a (*diagram*) functor; this identity is a globally unique reference, which is not visible to the user. The user identifies an ER-instance through a *key*. A key is a string of attribute names whose values uniquely identify each ER-instance. A relevant feature of the CER model is the distinction between entities and *isa* relationships: an ER-instance with an atomic key represents an entity and an ER-instance with a composite key represents an *isa* relationship. Other kinds of relationships can be represented as functors among ER-categories.

We show in the CER model that an ER-class has an initial object, which is called an ER-*schema*. In an ER-class, the *structure* of an ER-instance is given by the ER-schema of that category. The unique arrow from the initial object (schema) to the ER-instance represents its *semantics*. Therefore, the CER model offers a clear separation between syntax and semantics.

We use the object constructors offered by a category to define a many-sorted algebra, called a CER *algebra*. This algebra contains a complete set of operations to manipulate ER-classes. We introduce the ER-*addition* and ER-*deletion* operations to *update* ER-classes and the operations ER-*create*, ER-*join*, ER-*projection*, and ER-*selection* to *create* new ER-classes. Rigorous sort composition rules underlie the definitions of these operations. The operations of the CER algebra support (i) a flexible representation of the database and (ii) a *declarative* query language. In this framework, a given *database* can be represented as a particular expression in a CER algebra.

In the CER model, the ER-classes may be represented in various *normal forms*; the representation of an ER-class in a certain *normal form* is a natural transformation. The composition of ER-classes under ER-*join* preserves their normal forms. (In the standard relational model, relations are decomposed to satisfy normal forms requirements [10].)

This paper is organized as follows. Section 2 introduces ER-instances and ER-classes to represent database entities and entity-relationships. Section 3 introduces the operations of a CER algebra. Section 4 presents normal forms as natural transformations.

The concepts of category theory used in the present work are not repeated here; a complete introduction to category theory can be found in [5].

The work presented in this paper is a full development of the ideas reported in [12, 11].

2 Entities and Entity-Relationships

Our first objective is to provide representations for database *entities* and *relationships* in the CER model. In Subsection 2.1 we consider entities; they can be identified by single attributes, i.e., by *atomic keys*. Then in Subsection 2.2 we represent entity-relationships; they are identified by strings of two or more attributes, i.e., by *composite keys*. We assume that the attributes of an entity are in a *dependency* *relation* and that *keys* always exist.

2.1 Entities

Briefly, a database is a collection of *entities* connected by various *relationships*. A group consisting of all "similar" entities forms an *entity set*. Entity sets have properties, called *attributes*, which associate with each entity a *value* from a *semantic domain* for that attribute. A semantic domain may have a different structure than a set. In this paper, we will make use of *primitive domains* such as the set of integers, real numbers, or character strings, and the following *compound domains*, which are built from existing domains D and D': Cartesian product domains $D \times D'$; disjoint sum domains $D + D'$; lifted domains $D_\perp$, where $D_\perp = D \bigcup \perp$; symbol $\perp$ denotes "no value". We assume that the functions on lifted domains are *strict* [8].

The selection of the relevant attributes for an entity set is a critical step in the design of a conceptual database scheme. In principle, each entity set has a *key*. But, if a collection of attributes that induces a key for an entity set is not chosen, then one entity in the set cannot be distinguished from another. The key for an entity set can be expressed in terms of a *functional dependency* relation on its attributes. The functional dependency relation is intended to capture the relevant semantic information about the entities in an entity set. We introduce formally these issues in the definitions that follow.

Let $Att = \{a_1,a_2,...,a_n\}$ be a (finite) set of *attribute names* and let $\{D_{a_i} \mid a_i \in Att,\ i = 1,..,n\}$ be an *Att*-indexed family of *lifted semantic domains*. (We consider only lifted semantic domains; therefore, we can omit the symbol $\perp$ in the domain notation.) An element $d \in D_{a_i}$ is a *value* of the attribute name a_i. We denote by D_{Att} the disjoint union $D_{a_1}+...+D_{a_n}$; D_{Att} is a *lifted semantic domain* of *Att*. We call a function θ: $Att \rightarrow D_{Att}$ that assigns a value $d \in D_{a_i}$ to each attribute name a_i in *Att* an *assignment* function. An infinite number of assignment functions may exist; let $\{\theta_p \mid p = 1,2,..,m\}$ be a finite family of such functions (it is generally pointless to consider infinite families). We write $d^p_{a_i}$ for $\theta_p(a_i) = d$.

We use a preorder to represent the functional dependency relation on the attributes of an entity set. Let *att* be a subset of *Att* and let R_{att} be a preorder on *att*. A *preorder* on *att* is a transitive and reflexive relation $R_{att} \subseteq att \times att$. Any relation can be extended to a preorder by considering its reflexive and transitive closure. Intuitively, $\langle a_i,a_j \rangle \in R_{att}$ (also written $a_i \rightarrow a_j$) means that a_j *functionally depends on* a_i. $a^\bullet = \{a_j \in att \mid \langle a,a_j \rangle \in R_{att}\}$ is a useful notational aid that captures the *dependency extend* of an attribute name $a \in att$ in a preorder R_{att}.

Definition 2.1 An attribute name $a \in att$ for which $a^\bullet = att$ is called an *attribute key* of R_{att}. R_{att} is an ER-*preorder* if there exists an attribute name $a \in att$, such that $a^\bullet = att$.

Remark. There is always an ER-preorder on $att = \{a\}$. The empty relation $\emptyset$ is an ER-preorder on $att = \emptyset$.

Let att be a subset of Att for which there exists an ER-preorder; let R^a_{att} denote this ER-preorder, which has a as an attribute key. We define the category **att** to be the category whose objects are the elements of the set att and whose arrows are all the pairs $\langle a_i, a_j \rangle$ such that $\langle a_i, a_j \rangle \in R^a_{att}$. The composition of $\langle a_i, a_j \rangle$ followed by $\langle a_j, a_k \rangle$ is $\langle a_i, a_k \rangle$ (R^a_{att} is transitive) and the identity arrow for an element a_i is $\langle a_i, a_i \rangle$ (R^a_{att} is reflexive).

Let R_D be a preorder on D_{att}. (R_D is not an ER-preorder.) We define the category **Datt** to be the category with $Ob(\mathbf{Datt}) = D_{att}$ and $Mor(\mathbf{Datt}) = R_D$. (We use $Ob(\mathbf{Datt})$ to denote the class of the **Datt**-objects and $Mor(\mathbf{Datt})$ to denote the class of the **Datt**-arrows.)

We can extend an assignment function θ_p: $att \to D_{att}$ to an *assignment* functor θ_p: $\mathbf{att} \to \mathbf{Datt}$ if there exists a pair $\langle d^p_{a_i}, d^p_{a_j} \rangle$ in R_D, for each pair $\langle a_i, a_j \rangle \in R^a_{att}$. (Such functors are often referred to as *diagrams*.)

Definition 2.2 Given a family of assignment functors $\theta = \{\theta_p \colon \mathbf{att} \to \mathbf{Datt} \mid p = 1,2,..,m\}$, the ER-*preorder image under* θ of the ER-preorder R^a_{att}, denoted $\theta[R^a_{att}] = \bigcup_{p=1}^{m} \theta_p[R^a_{att}]$, is a θ-image of R^a_{att} in **Datt** that satisfies the following conditions:

(i) if $d^p_a = \perp$ then $d^p_{a_j} = \perp$ for each $a_j \in a^\bullet$;

(ii) $d^p_a \neq d^{p'}_a$ for $p \neq p'$;

(iii) each object d^p_a together with an arrow $\langle d^p_a, d^p_{a_j} \rangle$ for each $a_j \in a^\bullet$ is a *limit* for θ_p in **Datt**;
every triangle (b), which is a θ_p-image of the triangle (a), commutes:

$$\begin{tikzcd}
 & a \arrow[dl] \arrow[dr] & & & & & d^p_a \arrow[dl] \arrow[dr] & \\
a_j & & a_k \arrow[ll] & {} \arrow[r, "\theta_p"] & {} & d^p_{a_j} & & d^p_{a_k} \arrow[ll]
\end{tikzcd}$$

(a) (b)

Now let us return to the category **att**. We consider the comma category $(a,\mathbf{att})$, where the object a in $Ob(\mathbf{att})$ is the attribute key of R^a_{att}. We use $\mathbf{R^a_{att}}$ to denote this category, because R^a_{att} and $(a,\mathbf{att})$ have identical graphs. We also consider the comma category $(d^p_a,\mathbf{Datt})$ for each object $d^p_a = \theta_p(a_i)$ in $Ob(\mathbf{Datt})$, $p = 1,2,..,m$, such that $\theta_p[R^a_{att}]$ is an ER-preorder image. We use $\theta_p[\mathbf{R^a_{att}}]$ to denote such a category, to emphasize the role of the diagram θ_p.

The next step in our development is to assemble the categories $\mathbf{R^a_{att}}$ and $\theta_p[\mathbf{R^a_{att}}]$, $p = 1,2,..,m$; they have similar structures induced by the ER-preorder R^a_{att}.

Definition 2.3 Let $\mathbf{E}(R^a_{att})$ be the following category: $\mathbf{R^a_{att}}$ is an object of $\mathbf{E}(R^a_{att})$, and each $\theta_p[\mathbf{R^a_{att}}]$, $p = 1,2,..,m$, is also an object of $\mathbf{E}(R^a_{att})$. No other object is in $\mathbf{E}(R^a_{att})$. The arrows in $\mathbf{E}(R^a_{att})$ are precisely the assignment diagrams θ_p that exist between $\mathbf{R^a_{att}}$ and $\theta_p[\mathbf{R^a_{att}}]$, the isomorphism functors that exist between $\theta_p[\mathbf{R^a_{att}}]$, $p = 1,2,..,m$, and the identity functors for each object in $\mathbf{E}(R^a_{att})$. We call $\mathbf{E}(R^a_{att})$ an ER-*class* of *attributes att*, *sort* R^a_{att}, and *interpretation* $\theta[R^a_{att}]$; we call $\mathbf{R^a_{att}}$ the ER-*schema* of $\mathbf{E}(R^a_{att})$; we call $\theta_p[\mathbf{R^a_{att}}]$ an *entity-relationship* (ER) *instance*. An ER-class of one attribute is called *atomic*. We call the number m of the ER-instances of $\mathbf{E}(R^a_{att})$ the *size* of $\mathbf{E}(R^a_{att})$; an ER-class of size 0 is called an ER-class *skeleton*.

Remarks. Intuitively, an ER-class corresponds to an object-oriented class together with its instances. Definition 2.2 establishes the fact that a value d^p_a uniquely identifies an ER-instance. To keep the presentation tractable, we have included the sort of an ER-class in its notation.
The concepts introduced in Definition 2.3 are illustrated in the example below.

Example 2.4 We show in Fig. 1 a sample of the suppliers-and-parts relational database [3]. We will use this database for most of our examples in the paper. Suppliers (S) and parts (P) are uniquely identified by supplier number (S#) and part number (P#), respectively. Shipments (SH) are uniquely identified by S# and P# together. We have:

S#	NAME	STATUS	CITY
1	Smith	20	London
2	Jones	10	Paris
3	Clark	20	London
4	Adams	30	Athens

P#	NAME	COLOR	WEIGHT	CITY
1	Nut	Red	12	London
2	Bolt	Green	17	Paris
3	Screw	Blue	17	Rome
4	Screw	Red	14	London
5	Cam	Blue	12	Paris
6	Cog	Red	19	London

S#	P#	QTY
1	1	300
1	2	200
1	3	400
1	4	200
1	5	100
1	6	100
2	1	300
2	2	400
3	2	200
4	2	200
4	4	300
4	5	400

Figure 1: The relations S, P, and SH

attribute name set:
att = {S#, NAME (NM), STATUS (ST), CITY (CTY)}
lifted semantic domains:
$D_{S\#} = \{\perp, 1, 2, 3, 4\}$, $D_{NAME} = \{\perp, \text{Smith, Jones, Clark, Adams}\}$
$D_{STATUS} = \{\perp, 10, 20, 30\}$, $D_{CITY} = \{\perp, \text{London, Paris,Athens}\}$
$D_{att} = D_{S\#} + D_{NAME} + D_{STATUS} + D_{CITY}$

ER-preorder relation:

$R_{att}^{S\#} = \{$S# $\rightarrow$ NAME, S# $\rightarrow$ STATUS, S# $\rightarrow$ CITY, CITY $\rightarrow$ STATUS,
S# $\rightarrow$ S#, NAME $\rightarrow$ NAME, STATUS $\rightarrow$ STATUS, CITY $\rightarrow$ CITY$\}$,

which can be viewed as a graph

$$\begin{tikzcd} & \mathrm{S\#} \arrow[dl] \arrow[d] \arrow[dr] & \\ \mathrm{NAME} & \mathrm{STATUS} & \mathrm{CITY} \arrow[l] \end{tikzcd}$$

$\theta[R_{att}^{S\#}] = \bigcup_{p=1}^{4} \theta_p[R_{att}^{S\#}]$ is

$\{$S#.1 $\rightarrow$ NM.Smith, S#.1 $\rightarrow$ ST.20, S#.1 $\rightarrow$ CTY.London, CTY.London $\rightarrow$ ST.20,
S#.2 $\rightarrow$ NM.Jones, S#.2 $\rightarrow$ ST.10, S#.2 $\rightarrow$ CTY.Paris, CTY.Paris $\rightarrow$ ST.10,....$\}$

$R_{att}^{S\#}$ represents the *functional dependency* relation on the attribute names of relation S for which S# is an attribute key. The value of an attribute name under an assignment function θ_p: $att \rightarrow D_{att}$ such as 1 for S# (under θ_1) is written S#.1 in D_{att}. (We may assume that $\theta_p[R_{att}^{S\#}] = \{\langle \perp, \perp \rangle\}$ for $p > 4$.)

We use the above sets to construct the categories **att** and **Datt**: $Ob(\mathbf{att}) = att$, $Mor(\mathbf{att}) = R_{att}$, and $Ob(\mathbf{Datt}) = D_{att}$, $Mor(\mathbf{Datt}) = \theta[R_{att}^{S\#}]$.

$\mathbf{R_{att}^{S\#}}$ is the comma category of S# over **att** with elements

$$\begin{tikzcd} \mathrm{S\#} \arrow[d] & \mathrm{S\#} \arrow[d] & \mathrm{S\#} \arrow[d] & \mathrm{S\#} \arrow[d] \\ \mathrm{NAME} & \mathrm{STATUS} & \mathrm{CITY} & \mathrm{S\#} \end{tikzcd}$$

and arrow (we do not display the identity arrow)

$$\begin{tikzcd} & \mathrm{S\#} \arrow[dl] \arrow[dr] & \\ \mathrm{STATUS} & & \mathrm{CITY} \arrow[ll] \end{tikzcd}$$

$\mathbf{R_{att}^{S\#}}$, which is an ER-schema, can be viewed as a graph

$$\begin{tikzcd} & \mathrm{S\#} \arrow[dl] \arrow[d] \arrow[dr] & \\ \mathrm{NAME} & \mathrm{STATUS} & \mathrm{CITY} \arrow[l] \end{tikzcd}$$

Notice that $R_{att}^{S\#}$ and $\mathbf{R}_{\mathbf{att}}^{\mathbf{S\#}}$ have identical graphs.

$\theta_1[\mathbf{R}_{\mathbf{att}}^{\mathbf{S\#}}]$ is the comma category of S#.1 over **Datt** with elements

$$\begin{tikzcd}
\text{S\#.1} \arrow[d] & \text{S\#.1} \arrow[d] & \text{S\#.1} \arrow[d] & \text{S\#.1} \arrow[d] \\
\text{NM.Smith} & \text{ST.20} & \text{CTY.London} & \text{S\#.1}
\end{tikzcd}$$

and arrow (we do not display the identity arrow)

$$\begin{tikzcd}
& \text{S\#.1} \arrow[dl] \arrow[dr] & \\
\text{ST.20} & & \text{CTY.London} \arrow[ll]
\end{tikzcd}$$

$\theta_1[\mathbf{R}_{\mathbf{att}}^{\mathbf{S\#}}]$ is an ER-instance; its graph is

$$\begin{tikzcd}
& \text{S\#.1} \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{NAME.Smith} & \text{STATUS.20} & \text{CITY.London} \arrow[l]
\end{tikzcd}$$

Let $\mathbf{E}(R_{att}^{S\#})$ denote the ER-class containing ER-schema $\mathbf{R}_{\mathbf{att}}^{\mathbf{S\#}}$ and the ER-instances $\theta_i[\mathbf{R}_{\mathbf{att}}^{\mathbf{S\#}}]$, $i = 1{,}2{,}3{,}4$. $\mathbf{E}(R_{att}^{S\#})$ is the representation of relation S in the CER model.

Following the work of Goguen, Thatcher, Wagner, and Wright [8] on initial algebra semantics, in Proposition 2.5 we show that ER-schema $\mathbf{R}_{\mathbf{att}}^{\mathbf{a}}$ of an ER-class $\mathbf{E}(R_{att}^{a})$ represents the *abstract syntax structure* of an ER-instance $\theta_p[\mathbf{R}_{\mathbf{att}}^{\mathbf{a}}]$ of the ER-class. An ER-instance $\theta_p[\mathbf{R}_{\mathbf{att}}^{\mathbf{a}}]$ is an *interpretation* (*model*) of the ER-schema $\mathbf{R}_{\mathbf{att}}^{\mathbf{a}}$. Therefore, our model offers a clear separation between syntax and semantics.

Proposition 2.5 ER-schema $\mathbf{R}_{\mathbf{att}}^{\mathbf{a}}$ is an *initial* object in $\mathbf{E}(R_{att}^{a})$; that is, there exists a unique arrow θ_p: $\mathbf{R}_{\mathbf{att}}^{\mathbf{a}} \to \theta_p[\mathbf{R}_{\mathbf{att}}^{\mathbf{a}}]$ for every ER-instance $\theta_p[\mathbf{R}_{\mathbf{att}}^{\mathbf{a}}]$ in $\mathbf{E}(R_{att}^{a})$. □

Proposition 2.6 Given an ER-class $\mathbf{E}(R_{att}^{a})$ of size m and the discrete category $\{1,2,...,m\}$, we have that the ER-schema $\mathbf{R}_{\mathbf{att}}^{\mathbf{a}}$ together with the arrows θ_p: $\mathbf{R}_{\mathbf{att}}^{\mathbf{a}} \to \theta_p[\mathbf{R}_{\mathbf{att}}^{\mathbf{a}}]$, for $p = 1,2,..,m$, is a *natural source* for the diagram H: $\{1,2,..,m\} \to \mathbf{E}(R_{att}^{a})$, defined by $\mathrm{H}(p) = \theta_p[\mathbf{R}_{\mathbf{att}}^{\mathbf{a}}]$. □

Corollary The ER-schema $\mathbf{R}_{\mathbf{att}}^{\mathbf{a}}$ together with the arrows θ_p: $\mathbf{R}_{\mathbf{att}}^{\mathbf{a}} \to \theta_p[\mathbf{R}_{\mathbf{att}}^{\mathbf{a}}]$, $p = 1,2,..,m$, is a *limit* for the diagram H: $\{1,2,..,m\} \to \mathbf{E}(R_{att}^{a})$, $p = 1,2,..,m$. □

Example 2.4 (continued) In Fig. 2 we show that the ER-schema $\mathbf{R}^{\mathbf{S\#}}_{\mathbf{att}}$ together with the arrows θ_p: $\mathbf{R}^{\mathbf{S\#}}_{\mathbf{att}} \rightarrow \theta_p[\mathbf{R}^{\mathbf{S\#}}_{\mathbf{att}}]$, p = 1,2,3,4, is a limit for the diagram H: $\{1,2,3,4\} \rightarrow \mathbf{E}(R^{S\#}_{att})$ defined by H(p) = $\theta_p[\mathbf{R}^{\mathbf{S\#}}_{\mathbf{att}}]$, p = 1,2,3,4. The objects of

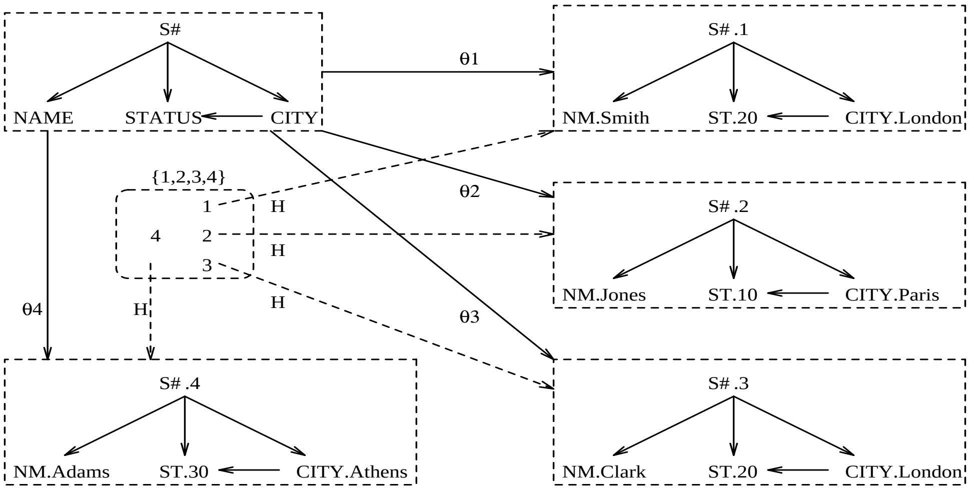


Figure 2: The ER-class $\mathbf{E}(R^{S\#}_{att})$

the discrete category {1,2,3,4} can be viewed as globally unique references to the ER-instances of $\mathbf{E}(R^{S\#}_{att})$; these references are not visible to the user. The user identifies the ER-instances through S#.1, S#.2, S#.3, S#.4, which are θ_p-images of the attribute key S#.

An important role in the definition of an ER-class is played by the attribute key. What happens when there are more keys available? We address this problem in Definition 2.10. The attribute names and the values of an attribute key have also an important role in the definition of an ER-class. Proposition 2.9 gives us some flexibility in the selection of the names for attributes and allows us to change the selected semantic domain of an attribute name.

Let $\mathbf{E}(R^a_{att})$ be an ER-class of interpretation $\theta[R^a_{att}] = \bigcup_{p=1}^{m} \theta_p[R^a_{att}]$. Function θ_p: $att \rightarrow D_{att}$ assigns a value $d^p_{a_i} \in D_{a_i}$ to each attribute name a_i in *att*. Let Z_{a_0} be another lifted semantic domain for a particular attribute a_0 in *att* and let $\psi : D_{a_0} \rightarrow Z_{a_0}$ be an injective function, which is strict. The $\psi \circ \theta_p$-image of R^a_{att}, denoted $\psi \circ \theta_p[R^a_{att}]$, is defined in the following way:

(i) $z^p_{a_j} = \psi(\theta_p(a_0))$ if $a_j = a_0$,

(ii) $z^p_{a_j} = \theta_p(a_j)$if $a_j \neq a_0$, and

(iii) there exists a tuple $\langle z^p_{a_i}, z^p_{a_j} \rangle$ in $\psi \circ \theta_p[R^a_{att}]$ for each tuple $\langle a_i, a_j \rangle$ in R^a_{att}; a_0, a_i, $a_j \in att$.

If ψ is injective, the $\psi\circ\theta_p$-image of R^a_{att} is an ER-preorder image.

Let *btt* be a subset of *Att* for which there exists a bijective function ϕ: $att \rightarrow btt$. Define $R^b_{btt} = \{\langle\phi(a),\phi(a_j)\rangle \mid \langle a,a_j\rangle \in R^a_{att}\} = \phi_p[R^a_{att}]$ as an ER-preorder on *btt* of attribute key $b = \phi(a)$. We will use these notations in the definitions that follow.

Definition 2.7 The *attribute renaming under* ϕ of the ER-class $\mathbf{E}(R^a_{att})$ of interpretation $\theta[R^a_{att}]$, denoted by $\mathbf{E}(R^a_{att})/\phi$, is the ER-class $\mathbf{E}(\phi[R^b_{btt}])$ of interpretation $\theta[R^b_{btt}]$, such that $\theta_p(b_j) = \theta_p(a_j)$ if $\phi(a_j) = b_j$.

Definition 2.8 The *semantic domain change under* ψ of the ER-class $\mathbf{E}(R^a_{att})$ of interpretation $\theta[R^a_{att}]$, denoted by $\mathbf{E}(R^a_{att})//\psi$, is the ER-class $\mathbf{E}(R^a_{att})$ of interpretation $\theta[R^a_{att}]//\psi = \bigcup_{p=1}^{m} \psi\circ\theta_p[R^a_{att}]$.

Proposition 2.9 $\mathbf{E}(R^a_{att})/\phi \approx \mathbf{E}(R^a_{att}) \approx \mathbf{E}(R^a_{att})//\psi$. $\square$

Definition 2.10 Let $\mathbf{E}(R^{a_i}_{att})$ be an ER-class of interpretation $\theta[R^{a_i}_{att}]$. Let a_j be another key in $R^{a_i}_{att}$ $(a_i \neq a_j)$. Then the *rekey* of $\mathbf{E}(R^{a_i}_{att})$ *from* a_i *to* a_j, denoted $\mathbf{E}(R^{a_i}_{att})[a_i \rightarrow a_j]$, is defined iff $\mathbf{E}(R^{a_i}_{att}) \approx \mathbf{E}(R^{a_j}_{att})$.
$\mathbf{E}(R^{a_i}_{att})[a_i \rightarrow a_j]$ is the ER-class $\mathbf{E}(R^{a_j}_{att})$ of interpretation $\theta[R^{a_j}_{att}] = \bigcup_{p=1}^{m} \theta_p[R^{a_i}_{att}]$.

2.2 Entity-Relationships

In general, more than one attribute is needed to uniquely identify an *isa* relationship. We introduce composite keys by extending the subset $att \subseteq Att$ of attribute names, which is used in Subsection 2.1, to a subset *att** of finite strings of attribute names. The subset *att** contains all the strings over *att* that satisfy the following conditions:

1. each attribute name in *att* occurs at most once in any string $s \in att^*$;
2. if $s,s' \in att^*$ have the same length, then there exists at least one attribute name in s that does not occur in s'.

It is clear that many sets *att** over *att* satisfy the above conditions; these sets are isomorphic [7]. We define on *att** a partial relation $\sqsubseteq$ in the following way: for s, $s' \in att^*$, $s' \sqsubseteq s$ iff all attribute names present in string s' are also present in s in the same relative order. Let R_{att*} be a preorder on *att**.

Definition 2.11 R_{att*} is an ER-*preorder* on *att** if the following conditions are satisfied:

(i) if s, $s' \in att^*$ and if $s' \sqsubseteq s$ then $\langle s,s'\rangle \in R_{att*}$;

(ii) if $s,s',s'' \in att^*$ and if $\langle s,s'\rangle \in R_{att*}$ then $\langle s\cdot s'',s'\cdot s''\rangle \in R_{att*}$ and $\langle s''\cdot s,s''\cdot s'\rangle \in R_{att*}$; (the symbol $\cdot$ denotes string concatenation)

(iii) there exists a string $s \in att^*$ such that $s^\bullet = att$.

Definition 2.12 A *composite attribute key* s is a string in att^* such that $s^\bullet = att$ and $s \sqsubseteq s'$ for any other string s' for which $s'^\bullet = att$.

Remarks In the relational model conditions (i) and (ii) in Definition 2.11 correspond to the Armstrong's axioms; Definition 2.12 contains the the minimality condition of a composite attribute key [10].

Let R^s_{att*} denote an ER-preorder on att^* of attribute key $s \in att^*$. R^a_{att} can be considered a particular case of R^s_{att*} by identifying a string with one element with the element. Let D_{att} be a lifted semantic domain for the set att (of n attribute names) under the assignment functions θ_p: $att \to D_{att}$, p = 1,2,..,m. We construct a lifted semantic domain for att^* by forming Cartesian products of finite families of D_{att} (of size not greater than n); let $\prod_{i \leq n} D^i_{att}$ ($D^i_{att} = D_{att}$, i = 1,2,..,n) denote the collection of these Cartesian products. Then we extend an assignment function θ_p: $att \to D_{att}$ to the function $\bar{\theta}_p$: $att^* \to \prod_{i \leq n} D^i_{att}$ in the following way:

1. $\bar{\theta}_p(s) = \theta_p(s)$ if $s \in att$;
2. $\bar{\theta}_p(s) = \langle \theta_p(s_1),..,\theta_p(s_n) \rangle$ if $s = s_1 \cdot s_2 \cdot .. \cdot s_n$.

We write d^p_s for $\bar{\theta}_p(s) = d \in \prod_{i \leq n} D^i_{att}$. Let R^s_{att*} be an ER-preorder and let R_{D*} be a preorder on $\prod_{i \leq n} D^i_{att}$. We use these preorders to define the categories **att*** and **Datt***, respectively, following the construction steps used in the definition of the categories **att** and **Datt** (given in Subsection 2.1). We can extend an assignment function $\bar{\theta}_p$: $att^* \to \prod_{i \leq n} D^i_{att}$ to an assignment functor $\bar{\theta}_p$: **att*** $\to$ **Datt*** if there exists a pair $\langle d^p_{s_i}, d^p_{s_j} \rangle$ in R_{D*}, for each pair $\langle s_i, s_j \rangle \in R^s_{att*}$.

Definition 2.13 Given a family of assignment functors $\bar{\theta} = \{\bar{\theta}_p$: **att*** $\to$ **Datt*** $\mid p$ = 1,2,..,$m\}$, the ER-*preorder image under* $\bar{\theta}$ of the ER-preorder R^s_{att*}, denoted $\bar{\theta}[R^s_{att*}] = \bigcup^m_{p=1} \bar{\theta}_p[R^s_{att*}]$, is a $\bar{\theta}$-image of R^s_{att*} that satisfies the following conditions:

(i) if $d^p_s = \perp$ then $d^p_{s_j} = \perp$ for each $s_j \in s^\bullet$;

(ii) $d^p_s \neq d^{p'}_s$ for $p \neq p'$;

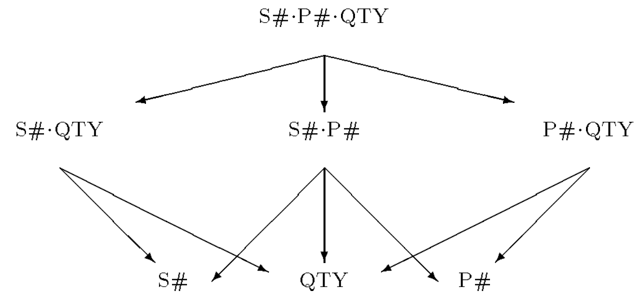
(i) each object d^p_s together with an arrow $\langle d^p_s, d^p_{s_j} \rangle$ for each $s_j \in s^\bullet$ is a *limit* for $\bar{\theta}_p$ in **Datt***.

Definition 2.14 Let $\mathbf{E}(R^s_{att*})$ denote an ER-class of sort R^s_{att*} of interpretation $\bar{\theta}[R^s_{att*}] = \bigcup^m_{p=1} \bar{\theta}_p[R^s_{att*}]$. The construction of an ER-class $\mathbf{E}(R^s_{att*})$ parallels the construction of an ER-class $\mathbf{E}(R^a_{att})$, which is given in Definition 2.3.

An ER-class represents a database entity when s is a string of length one and represents an *isa* relationship when s is a string of length greater than one. Therefore, our model offers a clear separation between entities and *isa* relationships. The construction of an *isa* relationship ER-class from given ER-classes uses an ER-*join* operation (Definition 3.8). In the sequel, we will consider all the definitions and propositions introduced in Subsection 2.1 extended to *att**.

Example 2.4 (continued) Let us consider the relation SH of the database in Fig. 1. We have:

attribute name sets:
  att $= \{$S#, P#, QTY$\}$ with
  att* $= \{$S#, P#, QTY, S#·P#, S#·QTY, P#·QTY, S#·P#·QTY$\}$
lifted semantic domains:
  $D_{S\#} = \{\perp, 1, 2, 3, 4\}$, $D_{P\#} = \{\perp, 1, 2, 3, 4, 5, 6\}$, $D_{QTY} = \mathbf{N}$
  $D_{att} = D_{S\#} + D_{P\#} + D_{QTY}$,
ER-preorder relation $R^{S\#\cdot P\#}_{att*}$ (given as a graph)



(S#·P# is the composite attribute key of $R^{S\#\cdot P\#}_{att*}$)
$\bar{\theta}[R^{S\#\cdot P\#}_{att*}] = \bigcup_{p=1}^{12} \bar{\theta}_p[R^{S\#\cdot P\#}_{att*}] = \{$S#.1·P#.1 $\rightarrow$ QTY.300, S#.1·P#.2 $\rightarrow$ QTY.200,
  S#.2·P#.2 $\rightarrow$ QTY.400, S#.3·P#.2 $\rightarrow$ QTY.200,...$\}$
  (we identify the tuples of one element with the element itself)

We use the above sets to construct the categories **att*** and **Datt***: $Ob(\mathbf{att*}) = att^*$, $Mor(\mathbf{att*}) = R_{att*}$, and $Ob(\mathbf{Datt*}) = \prod_{i \leq n} D^i_{att}$, $Mor(\mathbf{Datt*}) = \bar{\theta}[R^{S\#\cdot P\#}_{att*}]$.

$\mathbf{R^{S\#\cdot P\#}_{att*}}$ is the comma category of S#·P# over **att*** with objects

$$\begin{tikzcd}
S\#\cdot P\# \arrow[d] & S\#\cdot P\# \arrow[d] & S\#\cdot P\# \arrow[d] & S\#\cdot P\# \arrow[d] \\
QTY & S\# & P\# & S\#\cdot P\#
\end{tikzcd}$$

and only identity arrows.

$\mathbf{R^{S\#\cdot P\#}_{att*}}$, which is an ER-*schema*, can be viewed as a graph

$$\begin{tikzcd} & S\#\cdot P\# \arrow[dl] \arrow[d] \arrow[dr] & \\ S\# & P\# & QTY \end{tikzcd}$$

$\mathbf{R}^{\mathbf{S\#.1\cdot P\#.1}}_{\mathbf{att}*}$ is an ER-*instance*; it has the graph

$$\begin{tikzcd} & S\#.1\cdot P\#.1 \arrow[dl] \arrow[d] \arrow[dr] & \\ S\#.1 & P\#.1 & QTY.300 \end{tikzcd}$$

Let $\mathbf{SH}(R^{S\#\cdot P\#}_{att*})$ denote the ER-class containing ER-schema $\mathbf{R}^{\mathbf{S\#\cdot P\#}}_{\mathbf{att}*}$ and the ER-instances $\theta_i[\mathbf{R}^{\mathbf{S\#\cdot P\#}}_{\mathbf{att}*}]$, $i = 1,2,..,12$. $\mathbf{SH}(R^{S\#\cdot P\#}_{att*})$ is the representation of relation SH in the CER model.

Now we are ready to introduce the main concept of our model. Consider a set of attribute names *Att*, a lifted semantic domain D_{Att}, and a family of ER-preorders $\{R^s_{att*} \mid att \subseteq Att\}$.

Definition 2.15 A CER *database* is a category $\mathbf{DB_{Att}}$ whose objects are ER-classes $\mathbf{E}(R^s_{att*})$, for $att \subseteq Att$; the arrows of $\mathbf{DB_{Att}}$ represent various kinds of relationships among ER-classes.

Remarks: We cover in this paper only *isa* relationships (see Subsection 3). We assume that a *null* ER-class $\mathbf{0}$ exists in $\mathbf{DB_{Att}}$; $\mathbf{0}$ corresponds to $R_\emptyset = \emptyset$ and it is an initial object in $\mathbf{DB_{Att}}$. $\mathbf{DB_{Att}}$ is a small category. The objects of $\mathbf{DB_{Att}}$ are partitioned into equivalence classes of isomorphic objects (Proposition 2.9). Let $att \subseteq Att$. We denote by $[\mathbf{E}(R^s_{att*})]$ the equivalence class of all ER-classes isomorphic to $\mathbf{E}(R^s_{att*})$; we call $[\mathbf{E}(R^s_{att*})]$ an ER-*meta-class*. The equivalence class of the ER-schema and the equivalence class of an ER-instance of $[\mathbf{E}(R^s_{att*})]$ are denoted by $[\mathbf{R}^{\mathbf{s}}_{\mathbf{att}*}]$ and by $[\theta[\mathbf{R}^{\mathbf{s}}_{\mathbf{att}*}]]$, respectively.

3 CER Algebra

A database model, besides of a notation for data, must provide constructs for data manipulation. Particularly appealing are the models capable of supporting declarative query languages. We show in this section that the CER model supports a declarative query language.

In the CER model, entities and *isa* relationships are represented as ER-classes. We define operations on ER-classes following the style of the relational algebra [10]. One operation, called ER-*create* would correspond to the definition of a relation schema based on a functional dependency relation R. ER-*addition* would correspond to the addition of new tuples to a relation. ER-*deletion* would correspond to the deletion of tuples from a relation. ER-*join* and ER-*projection* on ER-classes would correspond to the Cartesian product and projection of relations, respectively. ER-*selection* would correspond to the selection of those tuples in a given relation that meet certain criteria.

Let $Att = \{a_1, a_2, \ldots, a_n\}$ be the set of attribute names of a given database. Let D_{Att} be the disjoint union $D_{a_1}+\ldots+D_{a_n}$ representing the lifted semantic domain of *Att*. A *trivial* ER-preorder $R^{a_i}_{\{a_i\}}$ is defined on each subset $\{a_i\}$ in *Att* of one element. The sort induced by a trivial ER-preorder is called an *atomic* sort; we use ambiguously a_i to denote the sort generated by $R^{a_i}_{\{a_i\}}$. An ER-preorder R^a_{att} of atomic attribute key is called a *basic* ER-preorder; it generates a *basic* sort. An ER-preorder R^s_{att*} of composite attribute key is called a *structured* ER-preorder; it generates a *structured* sort.

Let *att*, *btt*, and *ctt* ($att \neq btt \neq ctt$) be subsets of *Att* with two or more elements on which ER-preorders are given. Let R^a_{att*}, R^b_{btt*}, and R^c_{ctt*}, respectively, denote these ER-preorders; we use ambiguously the same names to represent the sorts they generate. Let $D_{Att/att}$, $D_{Att/btt}$, and $D_{Att/ctt}$ denote the restrictions of D_{Att} to *att*, *btt*, and *ctt*, respectively. We consider the following finite families of assignment functions:

- θ_r: $att \rightarrow D_{Att/att}$, $r = 1,..,m$; let $\theta(a) = \bigcup_{r=1}^{m}\{\theta_r(a)\}\subseteq D_a$;
- $\theta 1_p$: $att \rightarrow D_{Att/att}$, $p = 1,..,m1$; let $\theta 1(a) = \bigcup_{p=1}^{m1}\{\theta 1_p(a)\}\subseteq D_a$;
- $\theta 2_q$: $att \rightarrow D_{Att/att}$, $q = 1,..,m2$; let $\theta 2(a) = \bigcup_{q=1}^{m2}\{\theta 2_q(a)\}\subseteq D_a$;
- ϕ_t: $btt \rightarrow D_{Att/btt}$, $t = 1,..,m'$; let $\phi(b) = \bigcup_{t=1}^{m'}\{\phi_t(b)\}\subseteq D_b$;
- ψ_u: $ctt \rightarrow D_{Att/ctt}$, $u = 1,..,m''$; let $\psi(c) = \bigcup_{u=1}^{m''}\{\psi_u(c)\}\subseteq D_c$.

In addition to ER-classes of one attribute, we consider the following ER-classes:

- $\mathbf{E}(R^a_{att*})$ of interpretation $\bar{\theta}[R^a_{att*}] = \bigcup_{r=1}^{m} \bar{\theta}_r[R^a_{att*}]$;
- $\mathbf{E1}(R^a_{att*})$ of interpretation $\overline{\theta 1}[R^a_{att*}] = \bigcup_{p=1}^{m1} \overline{\theta 1}_p[R^a_{att*}]$;
- $\mathbf{E2}(R^a_{att*})$ of interpretation $\overline{\theta 2}[R^a_{att*}] = \bigcup_{q=1}^{m2} \overline{\theta 2}_q[R^a_{att*}]$;
- $\mathbf{E}(R^b_{btt*})$ of interpretation $\bar{\phi}[R^b_{btt*}] = \bigcup_{t=1}^{m'} \bar{\phi}_t[R^b_{btt*}]$;
- $\mathbf{E}(R^c_{ctt*})$ of interpretation $\bar{\psi}[R^c_{ctt*}] = \bigcup_{u=1}^{m''} \bar{\psi}_u[R^c_{ctt*}]$.

Definition 3.1 Two ER-classes that have the same sorts are *compatible* if they have the same lifted semantic domains D_{a_i} for all attributes a_i in *att*.

We will use the above conventions to define operations on ER-classes. First, we introduce an ER-*create* operation, which creates an ER-class skeleton with one attribute. A particular ER-*create* operation is needed for each attribute name in *Att*.

Definition 3.2 The result of ER-*create*-a_i, for $a_i \in Att$, is the ER-class skeleton $\mathbf{E}(a_i)$. An ER-class skeleton has no interpretation.

Then we introduce an ER-*preorder* operation that composes two ER-class skeletons of one attribute, selecting one ER-class as a key class.

Definition 3.3 The ER-*preorder* of $\mathbf{E}(a_i)$ and $\mathbf{E}(a_j)$ *under* key a_i, denoted ER-*preorder*($\mathbf{E}(a_i)$,$\mathbf{E}(a_j)$), is the ER-class skeleton $\mathbf{E}(R^{a_i}_{\{a_i,a_j\}})$ of sort $R^{a_i}_{\{a_i,a_j\}} = \{\langle a_i, a_j\rangle$.

We also introduce an ER-*create-instance* operation that creates an ER-instance (the first one) for a given ER-class skeleton of two or more attributes and adds it to the skeleton. The reason for making the addition part of the creation of an ER-instance is that we do not want ER-instances to exist outside of ER-classes.

Definition 3.4 Given an ER-class skeleton $\mathbf{E}(R^a_{att*})$ of two or more attributes and an assignment function θ_r: $att \rightarrow D_{Att/att}$, the result of ER-*create-instance*($\mathbf{E}(R^a_{att*})$,θ_r) is the ER-class $\mathbf{E}(R^a_{att*})$ of interpretation $\theta_r[R^a_{att*}]$. In other words, an ER-instance $\theta_r[\mathbf{R^a_{att*}}]$, is created from the ER-schema $\mathbf{R^a_{att*}}$ and is added to the ER-class skeleton $\mathbf{E}(R^a_{att*})$.

Intuitively, an ER-*addition* operation adds new ER-instances to an ER-class. The same result can be obtained if two compatible ER-classes are added. This operation is not defined for ER-classes that are not compatible.

Definition 3.5 The ER-*addition* of $\mathbf{E1}(R^a_{att*})$ and $\mathbf{E2}(R^a_{att*})$, denoted $\mathbf{E1}(R^a_{att*})$ $\oplus$ $\mathbf{E2}(R^a_{att*})$, is the ER-class $\mathbf{E}(R^a_{att*})$ whose

- ER-schema is $\mathbf{R^a_{att*}}$;
- interpretation is $\bar{\theta}[R^a_{att*}] = \overline{\theta 1}[R^a_{att*}]//\nu 1_a \bigcup \overline{\theta 2}[R^a_{att*}]//\nu 2_a$, such that $\bar{\theta}_r[R^a_{att*}] = \overline{\theta 1}_r[R^a_{att*}]//\nu 1_a$ if r = 1,..,$m1$, and $\bar{\theta}_r[R^a_{att*}] = \overline{\theta 2}_r[R^a_{att*}]//\nu 2_a$ if r = $m1$+1,..,$m2$, for all $a_i \in att*$; νj_a: $\theta j(a) \rightarrow \theta 1(a)+\theta 2(a)$, j = 1,2, are the natural injective arrows of the set disjoint union. (The values of the attribute key a have to be unique in $\mathbf{E}(R^a_{att*})$; we get unique values for a by modifying the semantic domains of a in $\mathbf{E1}(R^a_{att*})$ and $\mathbf{E2}(R^a_{att*})$ through the semantic domain change operation $//\nu j_a$, j = 1,2 (see Definition 2.8).)
- arrows are $Mor(\mathbf{E1}(R^a_{att*})//\nu 1_a) \bigcup Mor(\mathbf{E2}(R^a_{att*})//\nu 2_a) \bigcup \delta$, where δ represents the set of isomorphism functors that exist between ER-instances of $\mathbf{E1}(R^a_{att*})$ and ER-instances of $\mathbf{E2}(R^a_{att*})$.

(The ER-schema of an ER-class can be derived from its sort and its ER-instances can be derived from its interpretation.)

Intuitively, an ER-*deletion* operation removes ER-instances from an ER-class. The same result can be obtained if one ER-class is deleted from another ER-class, which is compatible with it. This operation is not defined for ER-classes that are not compatible.

Definition 3.6 The ER-*deletion* of $\mathbf{E1}(R^a_{att*})$ from $\mathbf{E}(R^a_{att*})$, denoted $\mathbf{E}(R^a_{att*})$ - $\mathbf{E1}(R^a_{att*})$, is the ER-class $\mathbf{E2}(R^a_{att*})$ whose

- ER-schema is $\mathbf{R^a_{att*}}$;
- interpretation is $\overline{\theta 2}[R^a_{att*}] = \bigcup_{q \leq m} \overline{\theta 2}_q[R^a_{att*}]$ for all $\overline{\theta 2}_q[R^a_{att*}]$ such that $\overline{\theta 2}_q[R^a_{att*}] \subseteq \overline{\theta}[R^a_{att*}]$ and $\overline{\theta 2}_q[R^a_{att*}] \not\subseteq \overline{\theta 1}[R^a_{att*}]$;
- arrows are the arrows of $\mathbf{E}(R^a_{att*})$ restricted to the ER-instances of $\mathbf{E2}(R^a_{att*})$.

Intuitively, an ER-*join* operation creates an ER-class of a new sort out of two ER-classes of two or more attributes that have different sorts. The new sort is induced from a new ER-preorder, which is basically the set disjoint union of the ER-preorders of the composing ER-classes with an additional element. This element is the concatenation of the attribute keys of the ER-preorders of the composing classes and it is chosen as the attribute key for the new ER-preorder. The resulting ER-class can be viewed as an ER-subclass of the both composing ER-classes (see Definition 3.10).

More precisely, given the ER-classes $\mathbf{E}(R^a_{att*})$ and $\mathbf{E}(R^b_{btt*})$, let *ctt* be the disjoint union *att* + *btt*, with $\mu 1$: $att \rightarrow ctt$, and $\mu 2$: $btt \rightarrow ctt$ as natural injective arrows. Consider $R_{ctt*} = \{\langle \mu 1(a_i), \mu 1(a_j)\rangle \mid \langle a_i, a_j\rangle \in R_{att*}\} \bigcup \{\langle \mu 2(b_k), \mu 2(b_l)\rangle \mid \langle b_k, b_l\rangle \in R_{btt*}\} \bigcup \{\langle \mu 1(a)\cdot\mu 2(b), \mu 1(a)\rangle\} \bigcup \{\langle \mu 1(a)\cdot\mu 2(b), \mu 2(b)\rangle\}$, where $ctt^* = \mu 1(att) \bigcup \mu 2(btt) \bigcup \{\mu 1(a)\cdot\mu 2(b)\}$.

Proposition 3.7 R_{ctt*} is an ER-preorder. The attribute key of R_{ctt*} is $\mu 1(a)\cdot\mu 2(b)$, for which $(\mu 1(a)\cdot\mu 2(b))^\bullet = \mu 1(a)^\bullet \bigcup \mu 2(b)^\bullet$. □

Definition 3.8 The ER-*join* of $\mathbf{E}(R^a_{att*})$ and $\mathbf{E}(R^b_{btt*})$, for $a \not\approx b$, denoted $\mathbf{E}(R^a_{att*}) \otimes \mathbf{E}(R^b_{btt*})$, is the ER-class $\mathbf{E}(R^c_{ctt*})$

- ER-schema is $\mathbf{R^c_{ctt*}}$, for $c = \mu 1(a)\cdot\mu 2(b)$;
- interpretation is $\bar{\psi}[R^c_{ctt*}] = \bigcup_{u=1}^{m''} \bar{\psi}_u[R^c_{ctt*}]$, such that $\bar{\psi}_u(c) = \bar{\psi}_u(\mu 1(a)\cdot\mu 2(b)) = \langle \bar{\theta}_r(a), \bar{\phi}_t(b)\rangle$, $\bar{\psi}_u(c_i) = \bar{\theta}_r(a_j)$ if $\mu 1(a_j) = c_i$, and $\bar{\psi}_u(c_i) = \bar{\phi}_t(b_k)$ if $\mu 2(b_k) = c_i$, for $u = 1,..m''$, $m'' = m \times m'$, $a_j \in att^*$, $b_k \in btt^*$, $c_i \in ctt^*$;
- arrows are the functions $\bar{\psi}_u$ extended to functors, for $u = 1,..m''$.

S#	NAME	CITY
1	Smith	London
2	Jones	Paris
3	Clark	London
4	Adams	Athens

CITY	STATUS
London	20
Paris	10
Athens	30

Figure 3: The relations SC and CS

Example 2.4 (continued) Consider the relations SC and CS in Fig. 3.

We use the same names to denote their corresponding ER-classes in the CER model. Let $att = \{$S#, NAME, CITY$\}$, $btt = \{$CITY, STATUS$\}$, and $ctt = att + btt$.
Let SC: $att \to ctt$ and CS: $btt \to ctt$ be the natural injective arrows of the set disjoint union. Then CITY in att would be SC.CITY in ctt and CITY in btt would be CS.CITY in ctt. (It is convenient to choose the relations' names to mark the origins of the attributes in att and btt in ctt, if the attributes belong to more than one set; the attributes that belong to only one set may have their markings omitted.) Similarly, London, a value of the attribute name CITY, would become SC.London and CS.London, respectively, in D_{ctt}. SC $\otimes$ CS is an ER-class of sort $R_{ctt*}^{S\#\cdot CS.CITY}$, where $ctt^* = \{$S#, NAME, SC.CITY, CS.CITY, STATUS, S#$\cdot$CS.CITY$\}$ Its size is 12. The graphical representation of SC $\otimes$ CS is given in Fig. 4. (Here ST stands for STATUS, NM for NAME, SCY for SC.CITY, and CSY for CS.CITY.)

Intuitively, the purpose of applying an ER-*join* operation to two ER-classes (of equal sizes) that have the same attribute keys and identical value keys is to add new attributes to an ER-class.

Definition 3.9 The ER-*join* of $\mathbf{E}(R^a_{att*})$ and $\mathbf{E}(R^b_{btt*})$, for a=b, is defined only if $m{=}m'$, $att \bigcap btt = \{a\}$, and $\bar{\theta}(a) = \bar{\phi}(b)$. $\mathbf{E}(R^a_{att*}) \otimes \mathbf{E}(R^b_{btt*})$ is the ER-class $\mathbf{E}(R^c_{ctt*})$ with $ctt = att \bigcup btt$ and $c = a = b$, of interpretation $\bar{\psi}[R^c_{ctt*}] = \bigcup_{u=1}^m \bar{\psi}_u[R^c_{ctt*}]$, such that $\bar{\psi}_u(c) = \bar{\theta}_u(a)$ for $c = a$, $\bar{\psi}_u(c_i) = \bar{\theta}_u(a_j)$ for $c_i = a_j$, and $\bar{\psi}_u(c_i) = \bar{\phi}(b_k)$ for $c_i = b_k$, where $a_j \in att$, $b_k \in btt$, $c_i \in ctt$.

Definition 3.10 Consider the ER-class $\mathbf{E}(R^c_{ctt*})$. The ER-*projection* of $\mathbf{E}(R^c_{ctt*})$ on $att \subseteq ctt$, denoted $pr_{att}(\mathbf{E}(R^c_{ctt*}))$, is defined only if an ER-preorder R^a_{att*} exists; $pr_{att}(\mathbf{E}(R^c_{ctt*}))$ is an ER-class $\mathbf{E}(R^a_{att*})$, which results from the restriction of ctt^* and D_{ctt*} to att^* and D_{att*}, respectively. We call $\mathbf{E}(R^c_{ctt*})$ an ER-*subclass* of $\mathbf{E}(R^a_{att*})$, and write $\mathbf{E}(R^c_{ctt*}) \sqsubseteq \mathbf{E}(R^a_{att*})$.

Remarks We consider that an ER-class $\mathbf{E}(R^a_{att*}) \otimes \mathbf{E}(R^b_{btt*})$ represents a database *isa* relationship, because it is an ER-subclass of both $\mathbf{E}(R^a_{att*})$ and $\mathbf{E}(R^b_{btt*})$and inherits all their attributes. The inclusion functors are the arrows from ER-classes to ER-subclasses.

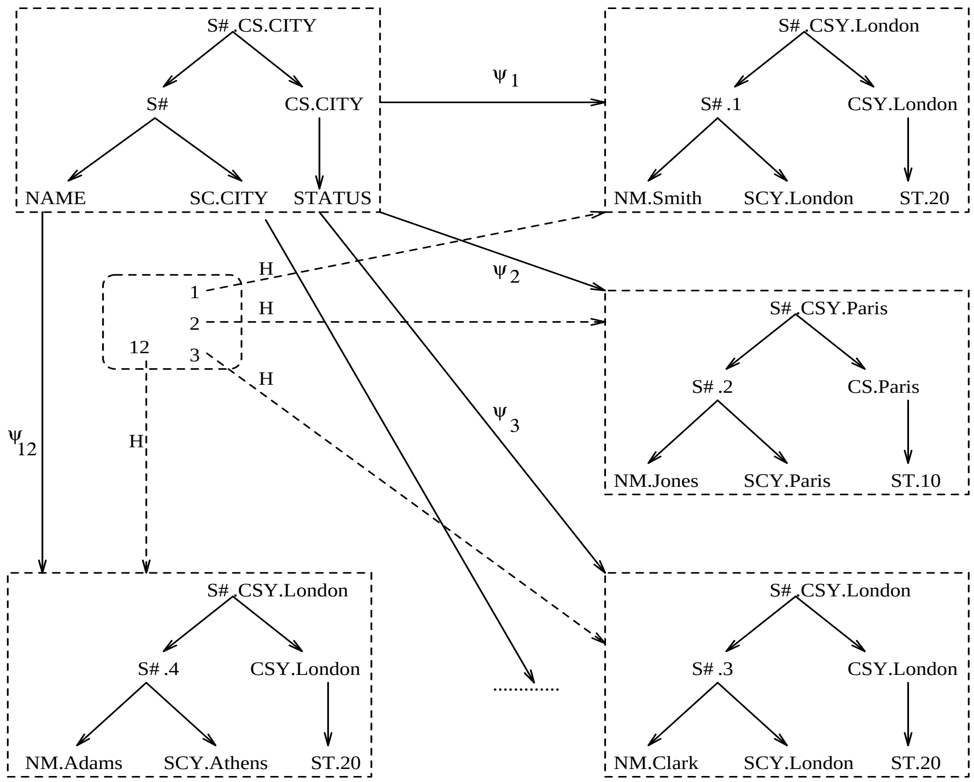


Figure 4: The ER-class SC $\otimes$ CS

The role of the selection operation is to support explicitly *semantic integrity constraints*.

Definition 3.11 The ER-*selection under* f of the ER-class $\mathbf{E}(R^a_{att*})$ of interpretation $\bar{\theta}[R^a_{att*}] = \bigcup_{r=1}^{m} \bar{\theta}_r[R^a_{att*}]$, for $\bar{\theta}_p$: $att* \to \prod_{i \leq n} D^i_{att}$, which is denoted $sel_f(\mathbf{E}(R^a_{att*}))$, is the ER-class $\mathbf{E}(R^a_{att*})$ whose

- ER-schema is $\mathbf{R^a_{att*}}$;
- interpretation is $\bar{\theta}'[R^a_{att*}] = \bigcup_{r=1}^{m'} \bar{\theta}'_r[R^a_{att*}]$, for θ'_p: $att \to f[D_{att}]$, where $f[D_{att}]$ is given by a pullback of the arrows f: $D_{att} \to \mathbf{Bool}$ and true: $\mathbf{1} \to \mathbf{Bool}$; f is a formula representing restrictions on the allowable values in D_{att} for the attribute names of att; $f[D_{att}]$ is shown in the following commutative diagram:
- arrows are the arrows of $\mathbf{E}(R^a_{att*})$ restricted to the ER-instances of $sel_f(\mathbf{E}(R^a_{att*}))$.

$$
\begin{tikzcd}
D' \arrow[dr, "h"] \arrow[drr, "q'"] \arrow[ddr, "p'"'] & & \\
& f[D_{att}] \arrow[r, "q"'] \arrow[d, "p"] & \mathbf{1} \arrow[d, "\mathbf{true}"] \\
& D_{att} \arrow[r, "f"'] & \mathbf{Bool}
\end{tikzcd}
$$

4 Normal Forms

In this section we discuss "normal form criteria" as introduced in the relational model. A relation is said to be in a particular normal form if it satisfies a certain set of constraints [10]. The normalization rules are designed to prevent update anomalies and data inconsistencies. In the relational model relations in a certain normal form are obtained by *decomposing* relations into subrelations that satisfy certain criteria. In our approach, an ER-class in a certain normal form can be obtained through *composition* using Algorithm 3.15. Relationships among various normal forms are natural transformations.

Definition 4.1 An ER-class $\mathbf{E}(R^a_{att*}))$ of interpretation $\bar{\theta}[R^a_{att*}]$ is in the *second normal form* (written also *2nf*) iff its ER-schema $\mathbf{R^a_{att*}}$ has no arrow

$$
\begin{tikzcd}
& a \arrow[dl] \arrow[dr] & \\
a_r & & a_j \arrow[ll]
\end{tikzcd}
$$

with $a_j \sqsubseteq a$, a_r, $a_j \in att$*.

Definition 4.2 An ER-class $\mathbf{E}(R^a_{att*}))$ of interpretation $\bar{\theta}[R^a_{att*}]$ is in the *third normal form* (written also *3nf*) iff all the arrows of its ER-schema $\mathbf{R^a_{att*}}$ are identity arrows.

Let DB be a particular database, for which we know the set *Att* of attribute names, its lifted semantic domain D_{Att}, and the family of ER-preorders (dependency relations) $\mathcal{R} = \{R_{att*} \mid att \subseteq Att\}$. We consider two categories, $\mathbf{W}_{Att}$ and $\mathbf{DB_{Att}}$. Category $\mathbf{W}_{Att}$ reflects the beginning of the designing phase of the database DB. $\mathbf{W}_{Att}$ is intended to represent the data in the database as a collection of sets;

the ER-preorders, i.e., the functional dependencies between its attributes, are not known at this time. Therefore, any subset of attributes is a potential entity. On the other hand, the construction of category $\mathbf{DB_{Att}}$ (see Definition 2.15) reflects the result of the analysis phase of the database when $\mathcal{R}$ is known.

Category $\mathbf{W}_{Att}$ has $Pow(Att) \bigcup Pow(D_{Att})$ (Pow denotes the power set) as objects; the functions θ: A $\rightarrow$ D, for A $= \{a_j,..,a_k\} \in Pow(Att)$ and D $= \{d_{a_j},..,d_{a_k}\} \in Pow(D_{Att})$, such that $\theta(a_j) = d_{a_j}$, are the only nontrivial arrows of $\mathbf{W}_{Att}$. Composition in $\mathbf{W}_{Att}$ is function composition.

We define the following functors between $\mathbf{W}_{Att}$ and $\mathbf{DB_{Att}}$:

1. *1NF*: $\mathbf{W}_{Att} \rightarrow \mathbf{DB_{Att}}$
2. *2NF*: $\mathbf{W}_{Att} \rightarrow \mathbf{DB_{Att}}$
3. *3NF*: $\mathbf{W}_{Att} \rightarrow \mathbf{DB_{Att}}$

Intuitively, functor *1NF* "selects" from the power set of attributes those subsets for which a dependency relation can be found, i.e., it defines the basic sorts and selects ER-classes of these sorts. More than one dependency relation may be found in a subset. Therefore, *1NF* associates an equivalence class of isomorphic ER-classes (see Proposition 2.9)to a given sort. Functor *2NF* "selects" from the power set of attributes those subsets in 1-1 correspondence with the ER-classes in *2nf*. Functor *3NF* "selects" from the power sets of attributes those subsets in 1-1 correspondence with the ER-classes in *3nf*. Formally,

1. *1NF*: $\mathbf{W}_{Att} \rightarrow \mathbf{DB_{Att}}$ such that:

   for $\{a_j,..,a_k\} = att \subseteq Att$,

   $$1NF(att) = \begin{cases} [\mathbf{R^a_{att*}}] & \text{if there exists } a \in att^* \text{ such that } a^\bullet = \{a_j,..,a_k\} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

   for $\{d_{a_j},..,d_{a_k}\} = D_{att} \subseteq D_{Att}$,

   $$1NF(D_{att}) = \begin{cases} \mathbf{0} & \text{if } 1NF(att) = \mathbf{0} \\ [\bar{\theta}[\mathbf{R^a_{att*}}]] & \text{if } a^\bullet = \{a_j,..,a_k\},\ a \in att^* \end{cases}$$

   $$1NF(\theta) = \begin{cases} \text{functors } \bar{\theta} & \text{from } [\mathbf{R^a_{att*}}] \text{ to } [\bar{\theta}[\mathbf{R^a_{att*}}]] \\ \mathbf{0} & \text{from } \mathbf{0} \text{ to } [(a,\mathbf{att})] \end{cases}$$

2. *2NF*: $\mathbf{W}_{Att} \rightarrow \mathbf{DB_{Att}}$ such that:

   for $\{a_j,..,a_k\} = att \subseteq Att$,

   $$2NF(att) = \begin{cases} [\mathbf{R^a_{att*}}] & \text{if it is in } 2nf \\ \mathbf{0} & \text{otherwise} \end{cases}$$

for $d_{a_j},..,d_{a_k} = D_{att} \subseteq D_{Att}$,

$$\mathit{2NF}(D_{att}) = \begin{cases} [\bar{\theta}[\mathbf{R}^{\mathbf{a}}_{\mathbf{att}*}]] & \text{if } [\mathbf{R}^{\mathbf{a}}_{\mathbf{att}*}] \text{ is in } \mathit{2nf} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

3. *3NF*: $\mathbf{W}_{Att} \rightarrow \mathbf{DB}_{\mathbf{Att}}$ such that:

for $\{a_j,..,a_k\} = att \subseteq Att$,

$$\mathit{3NF}(att) = \begin{cases} [\mathbf{R}^{\mathbf{a}}_{\mathbf{att}*}] & \text{if it is in } \mathit{3nf} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

for $d_{a_j},..,d_{a_k} = D_{att} \subseteq D_{Att}$,

$$\mathit{3NF}(D_{att}) = \begin{cases} [\bar{\theta}[\mathbf{R}^{\mathbf{a}}_{\mathbf{att}*}]] & \text{if } [\mathbf{R}^{\mathbf{a}}_{\mathbf{att}*}] \text{ is in } \mathit{3nf} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

Proposition 4.3 The functors *1NF*, *2NF*, and *3NF* between $\mathbf{W}_{Att}$ and $\mathbf{DB}_{\mathbf{Att}}$ are related to each other by the natural transformations ϕ, ψ, and η given in Fig. 5.

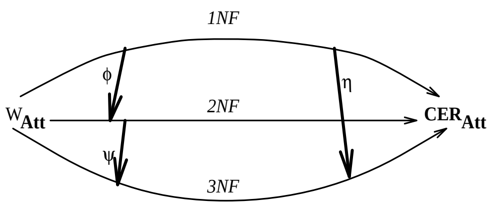


Figure 5: The natural transformations ϕ, ψ, and η

□

Proposition 4.4 The normal forms in which ER-classes are in are preserved by the operations ER-*addition*, ER-*deletion*, ER-*join*, and ER-*selection*; the *3nf* normal form is also preserved by ER-*projection*. □

5 Concluding Remarks

We have presented in this paper a highly abstract model of databases using category theory and initial algebra semantics. The model we have developed has a unifying power; it can be shown that the essential features of the relational, entity-relationship, and object-oriented models are captured by our model.

References

[1] C. Beeri. A formal approach to object-oriented databases. *Data and Knowledge Engineering*, 5, 1990.

[2] P.P. Chen. The entity-relationship model: toward a unified view of data. *ACM Trans. on Database Systems*, 1(1), March 1976.

[3] C.J. Date. *An Introduction to Database Systems*. Addison-Wesley, Reading Mass, 1981.

[4] R. Goldblatt. *Topoi: The Categorical Analysis of Logic*, volume 98 of *Studies in Logic and The Foundations of Mathematics*. North-Holland, 1979.

[5] H. Herrlich and G.E. Strecker. *Category Theory*. Heldermann Verlag, Berlin, 1979.

[6] R. Hull and R. King. Semantic database modeling: Survey, applications, and research issues. *ACM Computing Survey*, June 1987.

[7] N. Husberg. Categorical heterogeneous algebraic models of programming languages. Technical report, Aarhus University.

[8] E.G. Wagner J.A. Goguen, J.W. Thatcher and J.B. Wright. Initial algebra semantics and continuous algebras. *Journal of the ACM*, 24, Jan. 1977.

[9] W. Kim. Object-oriented databases: Definition and research directions. *IEEE Transactions on Knowledge and Data Engineering*, 2(3), 1990.

[10] J.D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume I. Computer Science Press, 1988.

[11] M. Zamfir-Bleyberg. A categorical entity-relationship model of databases. In *Proceedings, The 1991 Symposium on Applied Computing, Kansas City, MO*, April 1991.

[12] M. Zamfir-Bleyberg and A. Melton. Database systems: A categorical perspective. Technical report, KSU.