Relational Databases and Indexed Categories

Robert Rosebrugh and R. J. Wood

ABSTRACT. A description of relational databases in categorical terminology given here has as intended application the study of database dynamics, in particular we view (i) updates as database objects in a suitable category indexed by a topos; (ii) L-fuzzy databases as database objects in sheaves. Indexed categories are constructed to model the databases on a fixed family of domains and also all databases for a varying family of domains. Further, we show that the process of constructing the relational completion of a relational database is a monad in a 2-category of functors.

Introduction

We use the term *relation* for a subobject of a finite product of objects in a category. Following the relational database literature, we use the term *domain* for an object of the ambient category (and warn readers that these are *not* the ordered objects which go by the name "domain" elsewhere in theoretical Computer Science.) A *relational database*, as defined by E. F. Codd [3], is first of all a family of relations (or tables) on a family of domains. A heavily used example of domain is the set of character strings over an alphabet. Thus domains should be logically permitted to be infinite, though in practice they are always finite sets, e.g. character strings up to a fixed maximum length. The theory of databases as families of relations views domains simply as discrete objects. We adopt that point of view for this paper though the domains of practice usually have at least an order structure.

A very brief example will serve to illustrate the concepts mentioned so far. We introduce three domains, `name, address, phone`, which can each be viewed as sets of character strings satisfying appropriate constraints. An example of a database on this family of domains is the family of two relations, `address-book, phone-book`, where `address-book` is a subobject of `name` $\times$ `address`, and `phone-book` is a subobject of `name` $\times$ `phone`. Clearly, the storage and manipulation of databases is an important part of computing practise.

The theory of relational databases is well-developed, and the relational model for databases is now the most widely implemented. Earlier database paradigms ("network" and "hierarchical") are still found in many older systems. They are not as amenable to theoretical treatment, do not provide a portable conceptual structure and are of decreasing interest. Moreover, there is active current research on enhancements and extensions of the relational model. Current editions of the texts by Date [4] or Ullman [18] contain pointers to this work.

Research partially supported by grants from NSERC Canada. Diagrams typeset using Catmac.

This paper is in final form and no verison of it will be submitted for publication elsewhere.

The theory of families we use is the theory of *indexed categories* as studied by Paré and Schumacher [13]. Indexed categories are a widely used categorical tool, but have only begun to be explicitly used in theoretical computer science relatively recently [6,7,12,16]. The "relational algebra" of relational database theory involves operations which are set-theoretic and other operations which can be defined by a language involving only constants, variables of domain or relation type and equality. An objective of this article is to construct the relational completion of a database as the action of a monad, so that relationally complete databases are algebras for this monad. Section 1 gives some examples, and then the construction of a required family. In Section 2 we describe databases as families of relations in an **S**-indexed category $\boldsymbol{A}$ and construct an indexed category of databases for a fixed family of domains. We then return to examples, including updates and fuzzy databases. Section 3 considers the effect of varying domains and attributes and finds an indexed category of all databases in an indexed category. In Section 4 we construct the relational completion monad. We find that the endo-functor part of the monad is an endo-functor on the fibration which arises from the indexed category of databases. Finally, we observe that relationally complete databases are 2-categories of relations.

1. The Setting

We will freely use the notion of indexed category, so we first describe the basic language of indexed categories. We begin with a base category, **S**, which is required to have finite limits. Moreover, for our description of database objects, **S** must allow construction of free monoids. It suffices to assume that **S** is an elementary topos with natural numbers object N. Appropriate examples of **S** include the category of sets and functions, **set**, any topos of diagrams (or presheaves), or any Grothendieck topos. A topos which will interest us below is $\mathbf{set}^{\mathbf{2}}$, the topos whose objects are functions in **set** and whose arrows are commutative squares.

An **S**-*indexed category* $\boldsymbol{A}$ is given by a category $\mathbf{A}^I$ for each object I in **S** and a functor $\alpha^* : \mathbf{A}^I \longrightarrow \mathbf{A}^J$ for each arrow $\alpha : J \longrightarrow I$ in **S**. These *substitution functors* α^* are subject to isomorphisms making them compatible with identities and composition in **S**, and coherent with associativity. For example, if $\beta : K \longrightarrow J$ is also in **S**, then there is a canonical isomorphism $(\alpha\beta)^* \cong \beta^*\alpha^*$. For a complete description see [13].

We will often want $\boldsymbol{A}$ to be just **S** with a canonical indexed structure. We denote it by $\boldsymbol{S}$ with $\mathbf{S}^I$ defined to be the slice category $\mathbf{S}/I$, and the required substitutions defined by pullbacks. We detail two examples of $\boldsymbol{S}$ now.

Example 1. When we take **S** to be the category **set**, we find that the **set**-indexed category ***set*** has, for any set I, ordinary I-indexed families of sets as its I-indexed families. This follows since $\mathbf{set}/I$ has functions with codomain I as objects. Such a function, $x : X \longrightarrow I$ say, may be identified with a family of sets $< X_i >_{i \in I}$ defined by $X_i = x^{-1}(i)$, and conversely. In fact *any* category is **set**-indexed, again taking I-indexed families to be just ordinary families of objects.

Example 2. When $\mathbf{S}$ is $\mathbf{set}^{\mathbf{2}}$ we get a more interesting indexing. The indexing objects are now functions in $\mathbf{set}$, e.g. $I : I_0 \longrightarrow I_1$, and an I-indexed family X, being an arrow of $\mathbf{set}^{\mathbf{2}}$, is a pair of functions $x_0 : X_0 \longrightarrow I_0$, $x_1 : X_1 \longrightarrow I_1$ making a commutative square in $\mathbf{set}$, $x_1 X = I x_0$. Substitutions are defined by pullback which are computed point-wise.

Example 3. Another example of a $\mathbf{set}^{\mathbf{2}}$-indexed category arises when we allow the object X above to be replaced by a *partial* function, which we will denote $X : X_0 \leftarrowtail X_d \xrightarrow{X} X_1$. Thus, when $I = 1$ we obtain the category whose objects are partial functions and whose morphisms, from X to Y say, are pairs of functions $f_0 : X_0 \longrightarrow Y_0$ and $f_1 : X_1 \longrightarrow Y_1$ such that the restriction of f_0 to X_d factors through Y_d by f_d and $Y f_d = f_1 X$. An I-indexed family is a pair $X_0 \xrightarrow{x_0} I_0$, $X_1 \xrightarrow{x_1} I_1$ so that $x_1 X = I x_d$ — with x_d the restriction of x_0 to X_d. A morphism in I-indexed families is a pair (f_0, f_1) of functions so that in

$$\begin{tikzcd}
X_0 \arrow[dd, "x_0"'] \arrow[drr, "f_0"] & X_d \arrow[l, tail] \arrow[rrr, "X"] & & & X_1 \arrow[dd] \arrow[dr, "f_1"] & \\
& & Y_0 \arrow[dll, "y_0"] & Y_d \arrow[l, tail] \arrow[rr, "Y"] & & Y_1 \arrow[dl, "y_1"] \\
I_0 \arrow[rrrr, "I"] & & & & I_1 &
\end{tikzcd}$$

f_0 restricts to $f_d : X_d \longrightarrow Y_d$ and $f_1 X = Y f_d$. Substitution is still accomplished by pointwise pullback, including on the domain of full definition, X_d. We denote the resulting indexed category by $\boldsymbol{set}^{pf}$.

We want to define a relational database to be a J-indexed family of relations in $\boldsymbol{A}$ on some I-indexed family of domains, say A. A central feature of indexed category theory is that it identifies a J-indexed family of structures as a structure in the category of J-families, e.g. a J-indexed family of groups is a group in J-families. Similarly, a J-indexed family of relations is a *single* relation in the category $\boldsymbol{A}^J$ of J-indexed families. To define relational database in $\boldsymbol{A}$ we need to be able to say when a J-indexed family of relations is a family of subobjects of finite products of domains in A. In order to make this requirement precise, we will need some notation and some hypotheses on $\boldsymbol{A}$. The remainder of this section provides this background.

For an object I of $\mathbf{S}$ we denote the free monoid on I by $M(I)$. Henceforth *we assume that $M(I)$ exists in* $\mathbf{S}$. It is well known that $M(I)$ exists in any topos with a natural numbers object [10,13]. We also need to assume, and do so for the remainder of this paper that $\boldsymbol{A}$ *has finite products.* This requires that each $\mathbf{A}^I$ has finite products preserved by the α^*. If A is an object of $\mathbf{A}^I$, we will need the $M(I)$-indexed family of "finite products of members of A", denoted $P_0(A)$. When $\mathbf{S}$ is $\mathbf{set}$ the family desired has as fibre over a word $w = i_1 i_2 \ldots i_k \in M(I)$, the (finite) product whose description is $A^w = A_{i_1} \times A_{i_2} \times \ldots \times A_{i_k}$. We conclude this section by finding sufficient conditions for the existence of $P_0(A)$.

Under suitable hypotheses, the required family of finite products can be constructed as a solution to a "recursion problem" [15] for the indexed functor "crossing with A". We recall that, for **S**-indexed categories $\boldsymbol{A}$ and $\boldsymbol{B}$, an *indexed functor* $F : \boldsymbol{A} \longrightarrow \boldsymbol{B}$ is a family of functors $F^I : \mathbf{A}^I \longrightarrow \mathbf{B}^I$, one for each I in **S**. Further, for any arrow $\alpha : J \longrightarrow I$ in **S**, we must have the squares

$$\begin{tikzcd} \mathbf{A}^I \arrow[r, "F^I"] \arrow[d, "\alpha^*"'] & \mathbf{B}^I \arrow[d, "\alpha^*"] \\ \mathbf{A}^J \arrow[r, "F^J"'] & \mathbf{B}^J \end{tikzcd}$$

commuting up to coherent isomorphism. A *recursion problem* on $\boldsymbol{A}$ is a pair (Φ, C_0) with Φ an indexed endofunctor of $\boldsymbol{A}$ and C_0 in $\mathbf{A}^1$. The recursion problem (Φ, C_0) *has a solution* if there is an object C in $\mathbf{A}^N$ such that $0^*C = C_0$ and $s^*C = \Phi^N C$.

An hypothesis we shall need on $\boldsymbol{A}$ is that it has an indexed functor $E : \boldsymbol{A} \longrightarrow \boldsymbol{S}$ with small fibres. An indexed functor E has *small fibres,* when the objects of $\mathbf{A}^I$ whose image under E^I is a given object of $\mathbf{S}^I$ form a family indexed by an object of $\mathbf{S}/I$. An indexed functor with codomain $\boldsymbol{S}$ and small fibres was called an "e-functor" in [15]. The name refers to "elements" since the idea is that E^I gives a (very rough) idea of the cardinality of an object in $\mathbf{A}^I$. Examples include the identity functor on $\boldsymbol{S}$ and forgetful functors from (the **S**-indexed) category of groups (in **S**.) More interesting, any category of sheaves has an e-functor to **set** given by taking the union of all sections.

In the proposition which follows, we use a basic technique of indexed category theory "localizing" at $M(I)$. That is, we use the $\mathbf{S}/M(I)$-indexed category $\boldsymbol{A}^{M(I)}$, whose σ-indexed families, for $\sigma : J \longrightarrow M(I)$, are defined by $(\mathbf{A}^{M(I)})^\sigma = \mathbf{A}^J$. The hypotheses on the indexed category $\boldsymbol{A}$ remain true for $\boldsymbol{A}^{M(I)}$, and allow us to construct the required object there. We denote the inclusion of generators I in the monoid $M(I)$ by η. The internal sum of A in $\mathbf{A}^I$ along η is denoted $\Sigma_\eta A$. When **S** is **set**, $\Sigma_\eta A$ is the $M(I)$ family with fibre A_i over i and empty elsewhere.

Proposition 1. *Let* **S** *be a topos with NNO and* $\boldsymbol{A}$ *an* **S**-*indexed category with an e-functor* $E : \boldsymbol{A} \longrightarrow \mathbf{S}$. *Suppose that* $\boldsymbol{A}$ *has finite products and that the canonical arrows* $E^I(A \times B) \longrightarrow E^I(A) \times E^I(B)$ *are all monic. Suppose that* A *is in* $\mathbf{A}^I$ *and* $\Sigma_\eta A$ *exists. Then the recursion problem* $(\Phi, 1)$, *where* $\Phi : \boldsymbol{A}^{M(I)} \longrightarrow \boldsymbol{A}^{M(I)}$ *is defined by* $\Phi(C) = C \times \Sigma_\eta A$, *has a solution.*

Proof. We first localize E to $E^{M(I)} : \boldsymbol{A}^{M(I)} \longrightarrow \boldsymbol{S}^{M(I)}$, and note that the canonical arrows are still monic. Now we can invoke Proposition 2.2 of [15] as soon as we show that Φ is mono-bounded, i. e. for all C in $\mathbf{A}^{M(I)}$ there is a B in $\boldsymbol{S}/M(I)$ such that

(i) $E^{M(I)}(C) \rightarrowtail B$;

(ii) for all C' in $(A^{M(I)})^\sigma$, if $E^\sigma(C') \rightarrowtail B$ then $E^\sigma(\Phi^\sigma C') \rightarrowtail \sigma^* B$.

We let $B = \Omega^{(E(C)+E(\Sigma_\eta A)+2)^N}$ where Ω is the subobject classifier in $\mathbf{S}/M(I)$. Then using the coproduct injection and singleton, $E(C) \rightarrowtail B$ is clear and

$$E(\Phi C') = E(C' \times \Sigma_\eta A) \rightarrowtail E(C') \times E(\Sigma_\eta A) \rightarrowtail B \times B \rightarrowtail B.$$

The last monomorphism, whose existence is guaranteed by Lemma 2.2.3, Chapter 5 of [13], localizes to $(\mathbf{S}/M(I))^\sigma$ giving (ii). ∎

2. A Categorical Model for Databases

For this section we assume that $\mathbf{S}$ is a topos with NNO and $\boldsymbol{A}$ is $\mathbf{S}$-indexed and has finite products and an e-functor to $\mathbf{S}$. The *scheme* of a relation in a database is the list of attributes (= names of domains) appearing in the relation, that is, if A in $\mathbf{A}^I$ is an I-indexed family viewed as a family of domains, the scheme of a relation is in $M(I)$. A database with J relations and (database) signature $\sigma : J \longrightarrow M(I)$ is as follows:

Definition 1. *Let A in $\mathbf{A}^I$ be a family of domains. An A-*database *with* database signature σ, *where $\sigma : J \longrightarrow M(I)$, is a relation $R : R_0 \rightarrowtail \sigma^* P_0(A)$ in $\mathbf{A}^J$.*

Remarks. The signature σ of a database should not be confused with the scheme of a relation in the database. When $\boldsymbol{A} = \boldsymbol{S} = \mathbf{set}$, the schemes of the relations in the database, R_j for $j \in J$, are elements of $M(I)$ given by the signature, $\sigma : J \longrightarrow M(I)$.

As J and σ vary we obtain databases with varying families of relations. From this variation we will find a category of databases below.

The category $\mathbf{S}/M(I)$ is the base for indexing below, but we wish to also note that the full subcategory of $\mathbf{set}/M(I)$ whose objects are those $J \longrightarrow M(I)$ with J finite is of important practical interest, for these objects are signatures of databases in $\mathbf{set}$ with finitely many relations. However, this subcategory fails to have a terminal object. (In fact the terminal object in $\mathbf{S}/M(I)$ is $1_{M(I)} : M(I) \longrightarrow M(I)$, whose domain is essentially never finite.) The subcategory is thus unsuitable as a base category for indexing as arguments by localization become impossible. ∎

For any object $\sigma : J \longrightarrow M(I)$ of $\mathbf{S}/M(I)$, we define $\mathcal{D}(A)^\sigma$ to be the partial order, hence category, of subobjects of $\sigma^* P_0(A)$. If $\alpha : \sigma' \longrightarrow \sigma$ is an arrow of $\mathbf{S}/M(I)$, i.e. $\sigma\alpha = \sigma' : J' \longrightarrow M(I)$, define $\alpha^* : \mathcal{D}(A)^\sigma \longrightarrow \mathcal{D}(A)^{\sigma'}$ using substitution in $\boldsymbol{A}$.

Proposition 2. *Assume that all the functors α^* preserve monomorphisms. The $\mathcal{D}(A)^\sigma$ determine an $\mathbf{S}/M(I)$-indexed category denoted $\mathcal{D}(A)$.*

PROOF. We merely note that to $R : R_0 \rightarrowtail \sigma^* P_0(A)$, we have $\alpha^* R$ defined by

$$\alpha^* R_0 \rightarrowtail \alpha^* \sigma^* P_0(A) \cong \sigma'^* P_0(A).$$

Checking functorality of α^* and the equations is routine. ∎

REMARKS. The indexed category of databases which we denote $\mathcal{D}(A)$ above is locally a poset; in fact, it is the indexed category $\boldsymbol{sub}(P_0(A))$ described in [13], and when $\boldsymbol{A} = \boldsymbol{S}$ it is internally complete and cocomplete. Below we will consider subcategories arising by restricting the morphisms of $\mathcal{D}(A)$. ∎

Except for Example 3, where we will comment in more detail, we may take the identity functor as e-functor in the examples below. Thus all conditions of Proposition 1 are easily met. The substitution functors below are all defined by pulling back, so the conditions of Proposition 2 are met in all of the examples.

EXAMPLE 1 - CONTINUED. Once again, let $\mathbf{S} = \mathbf{set}$ and $\boldsymbol{A} = \boldsymbol{set}$. We note that for any family $< A_i >_{i \in I}$ of sets, $P_0(A)$ is the family of finite products of the A_i. When I, all of the A_i and J are finite, a database according to Definition 1 coincides exactly with the usual definition of a relational database with I domains $< A_i >_{i \in I}$ and J relations. That is, we have not lost anything by considering the setting proposed. We have gained a definition of morphism of databases — here just an inclusion, and further a definition of substitution along a function $\phi : K \longrightarrow J$, say. This substitution with ϕ monomorphic, for example, defines a database from a subset of the current relations. It might define an authorization class for example.

EXAMPLE 2 - CONTINUED. For an object A in $(\mathbf{set}^{\mathbf{2}})^I$, the object $P_0(A)$ is constructed "pointwise" i.e.

$$\begin{tikzcd}
P_0(A_0) \arrow[r, "P_0(A)"] \arrow[d, "p_0"'] & P_0(A_1) \arrow[d, "p_1"] \\
M(I_0) \arrow[r, "M(I)"'] & M(I_1)
\end{tikzcd}$$

where to $(a_{i_1}, \ldots, a_{i_k}) \in p_0^{-1}(i_1 \ldots i_k)$ we have

$$P_0(A)(a_{i_1}, \ldots, a_{i_k}) = (A(a_{i_1}), \ldots, A(a_{i_k})) \in p_1^{-1}(M(I)(i_1 \ldots i_k))$$

Thus a database object with $J : J_0 \longrightarrow J_1$ relations arises from a diagram such as the following:

$$\begin{tikzcd}
R_0 \arrow[rr, "R"] \arrow[dr, tail] \arrow[dddr] & & R_1 \arrow[dr, tail] \arrow[dddr] & & \\
& \sigma^* P_0(A_0) \arrow[rr] \arrow[dr] \arrow[dd] & & \sigma^* P_1(A_1) \arrow[dr] \arrow[dd] & \\
& & P_0(A_0) \arrow[rr] \arrow[dd] & & P_1(A_1) \arrow[dd] \\
& J_0 \arrow[rr] \arrow[dr] & & J_1 \arrow[dr] & \\
& & M(I_0) \arrow[rr, "M(I)"'] & & M(I_1)
\end{tikzcd}$$

That is, it consists of an A_0 database, R_0, in ***set*** with J_0 relations, an A_1 database, R_1, in ***set*** with J_1 relations and a function $R : R_0 \longrightarrow R_1$ so that a tuple $\mathbf{a} = (a_{l_1}, \ldots, a_{l_k})$ in the j'th relation of R_0 has $R(\mathbf{a})$ in the $J(j)$'th relation of R_1. Thus R itself is a rather general morphism of databases in that the domains, relations and attributes are allowed to be fully variable. This notion of database morphism will be encountered again in the next section. Restricting A and I to be identity functions, and similarly J, provides a simpler notion of database morphism. We exploit such restrictions in the next example.

Example 3 - continued. Again letting $\mathbf{S} = \mathbf{set^2}$, but now with $\boldsymbol{A} = \boldsymbol{set}^{pf}$, we obtain rather complicated objects similar to those in Example 2 as databases, but restricting attention to special cases will provide a description of the notion of "update". First, in order to guarantee that $P_0(A)$ exists we define an e-functor $E : \boldsymbol{A} \longrightarrow \boldsymbol{S}$ as in Proposition 1. If $a_0 : A_0 \longrightarrow I_0, a_1 : A_1 \longrightarrow I_1$ is an object of $\mathbf{A}^I$ with domain $A_0 \leftarrowtail A_d \longrightarrow A_1$, we define $E^I(a_0, a_1)$ by

$$\begin{tikzcd}
A_0 \arrow[r, "A^+"] \arrow[d, "a_0"'] & A_1 + I_1 \arrow[d, "a_1^+"] \\
I_0 \arrow[r, "I"'] & I_1
\end{tikzcd}$$

where A^+ is the canonical extension of $A : A_d \longrightarrow A_1$ given by viewing A as an I_1-indexed family of functions in **set** viz,

$$\langle \Sigma_{I(i_0)=i_1}(A_d)_{i_0} \longrightarrow (A_1)_{i_1} \rangle_{i_1 \in I_1}$$

and since $(A_d)_{i_0} \rightarrowtail (A_0)_{i_0}$ we may extend the I_1-family canonically to

$$\langle \Sigma_{I(i_0)=i_1}(A_0)_{i_0} \longrightarrow (A_1)_{i_1} + 1 \rangle_{i_1 \in I_1}.$$

Then a_1^+ is defined on $A_1 + I_1$ using a_1 and the identity on I_1. The E^I define an indexed functor $E : \boldsymbol{A} \longrightarrow \boldsymbol{S}$ with small fibres. We omit the technicalities of proving this. With this definition of E it is easy to see that

$E^I(A \times B) \rightarrowtail E^I(A) \times E^I(B)$. Moreover, $\Sigma_\eta A$ exists as internal sums are inherited from $\mathbf{set^2}$. So the requirements of Proposition 1 are met. The experienced reader will have noted that $P_0(A)$ could have been constructed directly in the case at hand. The point of the construction given is that it can be carried out for $\boldsymbol{S}^{pf}$ for any topos $\mathbf{S}$ with NNO. In that case, $a_1^+ : A_1 + I_1 \longrightarrow I_1$ above is replaced with the partial morphism classifier [8] for a_1 in $\mathbf{S}/I_1$.

An update of a database which does not change the current family of relations is one or several of three possible actions on elements (called "tuples") of the current relations:
(i) delete a tuple from a relation
(ii) add a tuple to a relation
(iii) change the values of some components of a tuple.
In fact, the third action can clearly be accomplished by a combination of the first two actions, namely delete the tuple in its current state and add it in the new state. Thus, an update can be accomplished by specifying a subset of current tuples which will be unchanged by the update (i.e. deleting the tuples to be changed or deleted) and specifying tuples to be added. In short, it is specified by a partial monomorphism whose domain is the current set of tuples, whose subset of total definition is the set of unchanged tuples, and whose function part is a monomorphism on that subset. Now let us describe the database objects in $\boldsymbol{set}^{pf}$ which express updates. For simplicity, we assume that the family of domains and the database signature are constant. That is, we assume that I and J in $\mathbf{set^2}$ are identity functions, and A in $(\mathbf{set}^{pf})^I$ has a fully defined identity function as its domain. In this special case, consider the diagram below for a database r with signature $\sigma : J \longrightarrow M(I)$. In the diagram following we denote the domain of $\sigma^* P_0(A)$ by P.

$$\begin{tikzcd}
R_0 \arrow[dd] \arrow[rd, tail, "r_0"] & R_d \arrow[l, tail, "i_R"'] \arrow[rr, "R"] \arrow[rd, "r_d"] & & R_1 \arrow[dd] \arrow[rd, tail, "r_1"] & \\
& P \arrow[ld] & P \arrow[l, tail, "="'] \arrow[rr, "="] & & P \arrow[ld] \\
J \arrow[rrr, "="'] & & & J &
\end{tikzcd}$$

We see that since r_0 and i_R are monic, r_d is monic and then R is too as $r_1 R = r_d$. The conclusion is:

a database object in $\boldsymbol{set}^{pf}$ *is an update of a database object in* $\mathbf{set}$.

It follows from this conclusion that we can reason about updates by reasoning about mere database objects in $\boldsymbol{set}^{pf}$.

EXAMPLE 4. Let us denote by ω the category freely generated by the graph

$$\mathbf{0} \longrightarrow \mathbf{1} \longrightarrow \mathbf{2} \longrightarrow \ldots .$$

$\mathbf{set}^{\omega}$ is a topos whose objects are diagrams

$$X : X_0 \xrightarrow{x_0} X_1 \xrightarrow{x_1} X_2 \xrightarrow{x_2} \ldots$$

with X_i sets and x_i functions, and which can be thought of as X varying in discrete time steps encoded in the x_i. An analysis similar to that in Example 2 shows that a database object in $\mathbf{set}^{\omega}$ is a sequence $R_0, R_1, \ldots$ of databases in $\mathbf{set}$, and the general morphisms of them discussed in Example 2. Restrictions to constant attribute families and relation signatures (the σ's of Definition 1) again provide a description of a database's variation through discrete time steps.

An important class of elementary topos is the categories of sheaves on a topological space, or more generally a locale. All base categories $\mathbf{S}$ above are of this form. Categories of fuzzy sets, after appropriate completion for fuzzy equality, are categories of sheaves on a locale, as Barr [1] has explained.

EXAMPLE 5. Let L be a locale. Following Barr, we will denote by L^{+} the locale L with a new bottom element adjoined. Then (the completion for fuzzy equality of) L-fuzzy sets is the category of sheaves on L^{+}, $\mathbf{sh}(L^{+})$. It is thus appropriate to define an *L-fuzzy database* to be a database object in $\mathbf{sh}(L^{+})$. With this definition, we note that Examples 1 and 2 above are special cases; Example 4 can be modified to $\mathbf{set}^{\omega+1}$, where $\omega + 1$ is $\mathbf{0} \longrightarrow \mathbf{1} \longrightarrow \ldots \omega$.The topos $\mathbf{set}^{(\omega+1)^{+}}$ is of the form $\mathbf{sh}(L^{+})$. It is worth emphasizing that, with the definition we give here, an L-fuzzy database has not only fuzzy relations, but also fuzzy domains and fuzzy index objects for its family of relations. Compare [14].

3. More indexed categories of databases

We have concentrated on a fixed (I-indexed) object A in the discussion above. It is clear that variation of A within $\mathbf{A}^{I}$, and further, variation of I, will define more indexed categories of databases. We have already seen such morphisms for the case $\mathbf{S} = \mathbf{set}$. They appeared as objects with variable attribute families and indexes for them, together with their associated databases in $\mathbf{set}^{\mathbf{2}}$. We indicate a few definitions:

Let $f : B \longrightarrow A$ be a morphism of $\mathbf{A}^{I}$. The arrow f determines an arrow of $\mathbf{A}^{M(I)}$ which we denote $P_0(f) : P_0(B) \longrightarrow P_0(A)$. For signatures $\sigma : J \longrightarrow M(I)$ and $\tau : K \longrightarrow M(I)$ we have a morphism between them in $\mathbf{S}/M(I)$ when there is $\nu : J \longrightarrow K$ satisfying $\tau\nu = \sigma$. Now we define a category $\mathcal{D}(\mathbf{A})^{I}$ as follows. Objects are triples (A, σ, R) with R an A-database with signature σ. An arrow from (A, σ, R) to (B, τ, R') is a pair $(f : A \longrightarrow B, \nu : \sigma \longrightarrow \tau)$ such that $\sigma^{*}P_0(f)$ restricted to R factors through $\nu^{*}R'$ as in the following (recall that we assume substitution preserves monics):

$$\begin{tikzcd}
\sigma^*P_0(A) \arrow[r, "\sigma^*P_0(f)"] & \nu^*\tau^*P_0(B) = \sigma^*P_0(B) \\
R \arrow[u, tail] \arrow[r, "\phi"'] & \nu^*R' \arrow[u, tail]
\end{tikzcd} .$$

Thus, when $\mathbf{S} = \mathbf{set}$ for example, we are requiring that for any $r \in R, \phi_j(r) = (\sigma^*P_0(f))_j(r)$ lies in $(\nu^*R')_j = R'_{\nu(j)}$. Since f determines ϕ, our description of arrows is appropriate.

Now suppose $\alpha : J \longrightarrow I$ is an arrow of $\mathbf{S}$. We obtain a functor $\alpha^* : \mathcal{D}(\mathbf{A})^I \longrightarrow \mathcal{D}(\mathbf{A})^J$ when we define

$$\alpha^*(A, \sigma, R) = (\alpha^*(A), \sigma', R')$$

where $\alpha^*(A)$ refers to the indexed structure on $\mathbf{A}$, $R' = (M(\alpha)')^*R$, and the following is a pullback:

$$\begin{tikzcd}
J' \arrow[r, "M(\alpha)'"] \arrow[d, "\sigma'"'] & J \arrow[d, "\sigma"] \\
M(J) \arrow[r, "M(\alpha)"'] & M(I)
\end{tikzcd}$$

Proposition 3. *The categories $\mathcal{D}(\mathbf{A})^I$ and functors α^* determine an* $\mathbf{S}$*-indexed category, denoted $\mathcal{D}(\mathbf{A})$.*

Proof. All that remains after the construction above is to verify that the $\mathcal{D}(\mathbf{A})^I$ are categories and that the α^* are functorial and compatible with identities and composition. This is a good exercise in the definitions of indexed category. ∎

4. Relational Completion is a Monad

The information obtained by querying a relational database is presented as relations. The relations which present the information derived are obtained by operating on the "base tables" or stored relations. The operations are collectively known as the "relational algebra" and depend on the schemes of stored relations as well as on the constants in the family of domains for the database. Simple examples of the operations are "list tuples in either of two relations" (union), "list the cartesian product of two relations", "list tuples whose value in one attribute equals their value in another" (selection.) The reader will have noted that combining the latter two sorts of operations allows formation of pullbacks. We will say that a database is *relationally complete* if it is closed under the operations of relational algebra. Of course, a relationally complete database can never be physically stored, having at least countably many relations. Thus, the ability to present any relation in the relational completion of a physically stored database is an important objective for a database system. Our objective in this section is to show that constructing the relational completion of database objects on a family of domains is a monad in a suitable 2-category, and that relationally complete databases are precisely algebras for this monad. Until further notice, *we assume that our databases are in* **set**, though much of what we say holds for databases in indexed categories satisfying the hypotheses of Proposition 1 by having $\boldsymbol{A} = \boldsymbol{S}$. Various families of binary and unary operations on the relations in a database have been called the "relational algebra". We describe a simple family of them namely two boolean operations, cartesian product, projection and selection. We remark that the other common operations, with the exception of difference, may be derived from those mentioned below [11]. We are unable to handle the binary difference of relations operation in our setup. Unlike the other operations of relational algebra we use here, it is not monotonic. This spoils functoriality of relational completion and that is too high a price to pay. We make some further remarks about this difficulty below.

The boolean operations are the simplest to describe. To do so, suppose that $R : R_0 \rightarrowtail \sigma^* P_0(A)$ is an A-database with signature $\sigma : J \longrightarrow M(I)$ and r, s are relations in R with the same scheme i.e. for some $j_1, j_2 : 1 \longrightarrow J$ we have $r = j_1^*(R_0)$, $s = j_2^*(R_0)$ and $\sigma j_1 = \sigma j_2$ so $r, s \rightarrowtail (\sigma j_1)^* P_0(A)$. Then $r \cup s, r \cap s \rightarrowtail (\sigma j_1)^* P_0(A)$ are relations with scheme σj_1. Next we consider cartesian product. If $r \rightarrowtail (\sigma j_1)^* P_0(A)$ and $s \rightarrowtail (\sigma j_2)^* P_0(A)$ then $r \times s \rightarrowtail (\sigma j_1 \cdot \sigma j_2)^* P_0(A)$ where $\sigma j_1 \cdot \sigma j_2$ denotes product in M(I) of σj_1, and σj_2.

Projection and selection require a little more care. Suppose that $j : 1 \longrightarrow J$ and that $r \rightarrowtail (\sigma j)^* P_0(A)$. Since σj is a word of length $n : 1 \longrightarrow N$, say, we may view the word as $\sigma j : [n] \longrightarrow I$. Now let $\varphi : [n'] \rightarrowtail [n]$ be one-one. The *projection of r along φ*, denoted $\pi_\varphi r$ is the (n'-ary) relation obtained by "projecting on the n' columns specified by φ." Finally, for j and r as above, and viewing σj as $\sigma j : [n] \longrightarrow I$, suppose that $\sigma j(i_1) = \sigma j(i_2) \in I$ and $c \in A_{\sigma j(i_1)}$. The selection operators $S_{i_1 = i_2}(r)$ and $S_{i_1 = c}(r)$ are defined by equalizers with codomain r as follows:

$$\begin{tikzcd}
S_{i_1=i_2}(r) \arrow[r, tail] & r \arrow[r] & (\sigma j)^* P_0(A) \arrow[r, shift left, "\pi_{i_1}"] \arrow[r, shift right, "\pi_{i_2}"'] & A_{\sigma j(i_1)}
\end{tikzcd}$$

and

$$\begin{tikzcd}
S_{i_1=c}(r) \arrow[r, tail] & r \arrow[r, tail] \arrow[dr] & (\sigma j)^* P_0(A) \arrow[r, "\pi_{i_1}"] & A_{\sigma j(i_1)} \\
& & 1 \arrow[ur, "c"'] &
\end{tikzcd}$$

Before proceeding, to establish notation, we give a brief description of 2-categories. The standard reference is [9]. A *2-category* **B** consists of a class of objects, denoted $\mathbf{B}_0$, and for each pair B, C of objects, a *hom category,* denoted $\mathbf{B}(B, C)$, whose objects are called 1-cells, and whose arrows are called 2-cells. We denote a 2-cell by $\sigma : f \Longrightarrow g$ in $\mathbf{B}(B, C)$. Composition of 2-cells in $\mathbf{B}(B, C)$

is called *vertical composition.* There is also a *horizontal composition* of 2-cells defined by a functor $\mathbf{B}(C,D) \times \mathbf{B}(B,C) \longrightarrow \mathbf{B}(B,D)$ for each triple B, C, D of objects of $\mathbf{B}$. An identity 1-cell in $\mathbf{B}(B,B)$ for each object B completes the necessary data. The data are required to satisfy associativity of horizontal composition, neutrality of identity arrows for horizontal composition and an interchange law linking horizontal and vertical composition. We generally denote vertical composition by juxtaposition or $\cdot$, and horizontal composition by $\circ$. The interchange law, for example, states that for

$$\begin{tikzcd}
B \arrow[r, bend left=50, ""{name=U1}] \arrow[r, ""{name=M1}] \arrow[r, bend right=50, ""{name=D1}] & C \arrow[r, bend left=50, ""{name=U2}] \arrow[r, ""{name=M2}] \arrow[r, bend right=50, ""{name=D2}] & D
\arrow[Rightarrow, from=U1, to=M1, "\sigma"] \arrow[Rightarrow, from=M1, to=D1, "\sigma'"] \arrow[Rightarrow, from=U2, to=M2, "\tau"] \arrow[Rightarrow, from=M2, to=D2, "\tau'"]
\end{tikzcd}$$

we have $(\tau' \cdot \tau) \circ (\sigma' \cdot \sigma) = (\tau' \circ \sigma') \cdot (\tau \circ \sigma)$. The motivating example of a 2-category is **cat**, the 2-category with categories as objects, functors as 1-cells and natural transformations as 2-cells. Another important example is **rel**, the 2-category of relations. The objects of **rel** are sets. If B and C are sets, then the category $\mathbf{rel}(B,C)$ is the partial order, qua category, of relations from B to C, so vertical composition is trivial and horizontal composition is the usual composition of relations. It is defined on 2-cells since if $R \subseteq R' : B \longrightarrow C$ and $S \subseteq S' : C \longrightarrow D$ are relations then $SR \subseteq S'R'$. The interchange law follows since all equations among 2-cells in **rel** are trivial: inclusion either holds or it doesn't.

We next describe the 2-category $\mathbf{cat}^{\mathbf{2}}$ in which we will work below. The objects of $\mathbf{cat}^{\mathbf{2}}$ are functors. If $A : \mathbf{A}_0 \to \mathbf{A}_1$ and $B : \mathbf{B}_0 \to \mathbf{B}_1$ are functors a 1-cell $F : \mathbf{A} \to \mathbf{B}$ is a triple $\mathbf{F} = (F_0, F_1, \Phi)$ where $F_0 : \mathbf{A}_0 \to \mathbf{B}_0$ and $F_1 : \mathbf{A}_1 \to \mathbf{B}_1$ are functors and $\Phi : BF_0 \cong F_1A$ is a natural isomorphism. A 2-cell $\tau : \mathbf{F} \Longrightarrow \mathbf{G} : \mathbf{A} \to \mathbf{B}$, with $\mathbf{G} = (G_0, G_1, \Gamma)$, is a pair of transformations $\tau_0 : F_0 \Longrightarrow G_0$ and $\tau_1 : F_1 \Longrightarrow G_1$ such that $\Gamma \cdot (\tau_1 \circ A) = (B \circ \tau_0) \cdot \Phi$. Verification that the obvious compositions of 1- and 2- cells provide a 2-category is left to the reader.

For any $\mathbf{S}$-indexed category $\mathbf{A}$ there is a functor with codomain $\mathbf{S}$ called the Grothendieck construction for $\mathbf{A}$. We denote this functor $G_A : gr(\mathbf{A}) \to \mathbf{S}$ and briefly describe it. The objects of $gr(\mathbf{A})$ are pairs (I, A) with A an object of $\mathbf{A}^I$. The hom-sets are defined by the formula $gr(\mathbf{A})\,((I,A),(J,A')) = \left\{(\alpha, \varphi) \,\middle|\, I \xrightarrow{\alpha} J, A \xrightarrow{\varphi} \alpha^* A' \right\}$. The functor G_A is given on objects and arrows by projecting on I and α. In fact, G_A is a fibration [5].

We can now proceed to describe the relational completion monad. We recall that a monad in a 2-category is a 1-cell with a monoid structure given by "unit" and "multiplication" 2-cells [17]. Thus in $\mathbf{cat}^{\mathbf{2}}$ a monad is an arrow of $\mathbf{cat}^{\mathbf{2}}$ together with unit and multiplication 2-cells. We note that a monad on $A : \mathbf{A}_0 \longrightarrow \mathbf{A}_1$ in $\mathbf{cat}^{\mathbf{2}}$ is specified by a monad M_0 in **cat** on $\mathbf{A}_0$, a monad M_1 in **cat** on $\mathbf{A}_1$ and a 2-cell $M : M_1A \cong AM_0$. These must satisfy equations involving the structures of the given monads in **cat** and M.

The domain of our monad $\mathbf{C}$ is the Grothendieck construction for the $\mathbf{S}/M(I)$-indexed category $\mathcal{D}(A)$, which is denoted $G_{\mathcal{D}(A)} : gr\mathcal{D}(A) \longrightarrow \mathbf{S}/M(I)$.

Thus, the monad $\mathbf{C} = (C^0, C^1, \Gamma)$ requires functors $C^0 : gr\mathcal{D}(A) \longrightarrow gr\mathcal{D}(A)$, $C^1 : \mathbf{S}/M(I) \longrightarrow \mathbf{S}/M(I)$ and a transformation $\Gamma : C^1 G_{\mathcal{D}(A)} \Longrightarrow G_{\mathcal{D}(A)} C^0$.

The second functor C^1 is inductively defined and we describe it first. Its action on a database signature $\sigma : J \longrightarrow M(I)$ produces the database signature we denote $C^1(\sigma) : \sigma^1 \longrightarrow M(I)$. This is generated, beginning from the set of relation names J, by freely adjoining to σ^1 all expressions of relational algebra as described above for the family of domains A. These expressions have appropriate schemes given by $C^1(\sigma)$. The inductive definition of σ^1 (and simultaneously $C^1(\sigma)$) follows:
σ^1 is the smallest set satisfying clauses 1. – 5. below subject to identifications generated by equations 1. – 7.:

1. $\forall j \in J, j \in \sigma^1$ (with scheme $\sigma(j)$)
2. if s_1, $s_2 \in \sigma^1$ (with scheme $C^1(\sigma)(s_1) = C^1(\sigma)(s_2)$) then $(s_1 \cup s_2)$, $(s_1 \cap s_2) \in \sigma^1$ (with scheme $C^1(\sigma)(s_1)$)
3. if s_1, s_2 in σ^1 (with schemes $C^1(\sigma)(s_1)$ and $C^1(\sigma)(s_2)$ then $(s_1 \times s_2) \in \sigma^1$ (with scheme $C^1(\sigma)(s_1) \cdot C^1(\sigma)(s_2)$)
4. if $s \in \sigma^1$ (with scheme $C^1(\sigma)(s) : [n] \longrightarrow I$) and $\varphi : [n'] \longrightarrow [n]$, then $(\pi_\varphi s) \in \sigma^1$ (with scheme $C^1(\sigma)(s) \cdot \varphi$)
5. if $s \in \sigma^1$, $C^1(\sigma)(s)(i_1) = C^1(\sigma)(s)(i_2)$ and if $c \in A_{C^1(\sigma)(s)(i_1)}$, then $(S_{i_1=i_2}(s))$ and $(S_{i_1=c}(s))$ are in σ^1 (with schemes $C^1(\sigma)(s)$.)

The following equations (which have no effect on schemes) apply to the "words" generated above. Let $s, s_1, s_2, s_3 \in \sigma^1$ be such that the operations mentioned below are defined, then

1. $(s_1 \times s_2) \times s_3 = s_1 \times (s_2 \times s_3)$
2. $(s_1 \cup s_2) \cup s_3 = s_1 \cup (s_2 \cup s_3)$ $\quad (s_1 \cup s_2) = (s_2 \cup s_1)$ $\quad (s_1 \cup s_1) = s_1$
   $(s_1 \cup s_2) \cap s_3 = (s_1 \cap s_3) \cup (s_2 \cap s_3)$ $\quad (s_1 \cup s_2) \cap s_2 = s_1$
   and the dual equations
3. if $\varphi : [n'] \rightarrowtail [n]$ and $\psi : [n''] \rightarrowtail [n']$ then $\pi_\psi \pi_\varphi s = \pi_{\varphi\psi} s$
4. $S_{i=j}(S_{i=j} s) = S_{i=j} s = S_{j=i} s$ $\quad S_{i=i} s = s$ $\quad S_{i=j}(S_{k=l} s) = S_{k=l}(S_{i=j} s)$
   $S_{i=c}(S_{i=j} s) = S_{i=j}(S_{i=c} s)$ $\quad S_{i=c}(S_{i=c} s) = S_{i=c} s$
5. $s_1 \times (s_2 \cup s_3) = (s_1 \times s_2) \cup (s_1 \times s_3)$ $\quad s_1 \times (s_2 \cap s_3) = (s_1 \times s_2) \cap (s_1 \times s_3)$
6. $\pi_\varphi(s_1 \cup s_2) = \pi_\varphi s_1 \cup \pi_\varphi s_2$ $\quad \pi_\varphi(s_1 \cap s_2) = \pi_\varphi s_1 \cap \pi_\varphi s_2$
7. $S_{i=j}(s_1 \cup s_2) = S_{i=j} s_1 \cup S_{i=j} s_2$ $\quad S_{i=j}(s_1 \cap s_2) = S_{i=j} s_1 \cap S_{i=j} s_2$

The definition of C^1 on an arrow $\tau : \sigma_1 \longrightarrow \sigma_2$ of $\mathbf{S}/M(I)$ is that induced by the action of τ from generators of σ_1^1 to those of σ_2^1. The transformations making C^1 a monad on $\mathbf{S}/M(I)$ are easily described. The unit is simply inclusion of the generators, while the multiplication simply rewrites a "word of words" in

$(C^1(\sigma))^1$ as that element of σ^1 obtained by treating operations on elements of σ^1 as operations defining an element of σ^1.

Next we decribe C^0. The image under C^0 of an object (σ, R) of $gr\mathcal{D}(A)$ is $(C^1(\sigma), R^0)$ where R^0 is the σ^1-indexed family of relations where, for $s \in \sigma^1$, R^0_s is the relation obtained by applying the relational operations in the construction of s to the corresponding relations of R. Extending C^0 to arrows of $gr\mathcal{D}(A)$ presents no difficulties. Once again, the monad unit is inclusion of generators and the multiplication arrow on the family of relations component merely applies the appropriate relational operations.

The non-monotonicity of binary difference of relations would prevent the definition of C^0 indicated above. One way to get around this problem is to use updates rather than simply inclusions as arrows in a modified $\mathcal{D}(A)$. The relationally complete databases are then those algebras for the resulting monad which have fully defined structural arrows. We have chosen to leave out difference in order that a "relationally complete database" is precisely an algebra for the monad $\mathbf{C}$ we have described.

Now the definition of Γ is straightforward. For any object (σ, R) of $gr\mathcal{D}(A)$, the natural isomorphism required has components the identity on $C^1(\sigma)$. The reader will have no trouble verifying that $\mathbf{C}$ is a monad on $G_{\mathcal{D}(A)}$ in $\mathbf{cat}^{\mathbf{2}}$.

The category $gr\mathcal{D}(A)^{\mathbf{C}^0}$ of Eilenberg-Moore algebras [17] for the monad $\mathbf{C}^0$ has objects pairs (σ, R) where σ is the database signature of a relationally complete database with relations R. The Eilenberg-Moore algebras $\mathbf{set}/M(I)^{\mathbf{C}^1}$ for $\mathbf{C}^1$ has objects database signatures with interpretations for the operations of relational algebra. There is a functor between these categories $G^{\mathbf{C}}_{\mathcal{D}(A)} : gr\mathcal{D}(A)^{C^0} \longrightarrow \mathbf{set}/M(I)^{C^1}$ which forgets the relations.

Proposition 4. *The functor $G^{\mathbf{C}}_{\mathcal{D}(A)}$ is the Eilenberg-Moore object for the monad $\mathbf{C}$ on $G_{\mathcal{D}(A)}$ in $\mathbf{cat}^{\mathbf{2}}$.*

Proof. This is an instance of a more general result which asserts that Eilenberg-Moore algebras for monads in $\mathbf{cat}^{\mathbf{2}}$ are computed, as here, by taking algebras for the domain and codomain monads and the induced functor between these categories of algebras. ∎

For the balance of this section we will assume that the indexing category $\mathbf{S}$ is an elementary topos with NNO and that $\boldsymbol{A} = \boldsymbol{S}$. Before describing the 2-category arising from a relationally complete database, we note that to any word $w : 1 \longrightarrow M(I)$ of $M(I)$ we can associate a "diagonal" relation $\delta_w : \Delta_w \rightarrowtail (ww)^*P_0(A) \cong w^*P_0(A) \times w^*P_0(A)$. We take Δ_w to be $w^*P_0(A)$, and define δ_w by projections to $w^*P_0(A)$ which are both the identity.

Now let $R : R_0 \rightarrowtail \sigma^*P_0(A)$ be a datbase with database signature $\sigma : J \longrightarrow M(I)$. Provided that R is closed under projection, selection and cartesian product, we can construct a 2-category which we will denote $\mathbf{R}$. We define the objects of $\mathbf{R}$ to be $M(I)$. Letting $v, w : 1 \longrightarrow M(I)$ be objects of $\mathbf{R}$, we define

1-cells by the formula

$$\mathbf{R}(v,w) = \begin{cases} \{(v,R_j,w) \in R \mid \sigma(j) = vw\} & \text{if } v \neq w \\ \{(v,R_j,w) \in R \mid \sigma(j) = vw\} \bigcup \{\Delta_w\} & \text{if } v = w \end{cases}$$

The two-cells are given by inclusion.

Proposition 5. *Let ρ be a database closed under selection, projection and cartesian product. The structure $\mathbf{R}$ is a 2-category with composition of 1-cells given by relational composition.*

Proof. We need only observe that if $r \in \mathbf{R}(v,w)$ and $s \in \mathbf{R}(u,v)$ then their relational composite (with a slight abuse of our notation for projection and selection) is

$$rs = \pi_{uw}(s \bowtie r) := \pi_{uw}(S_{r.v=s.v}(s \times r))$$

so that rs is in R. Now, with identities given by the Δ's, we have that R is a sub-2-category of $\mathbf{rel}(\mathbf{S})$ and the required equations all follow from the latter's structure. ∎

Corollary 1. *If ρ is a relationally complete database, then $\mathbf{R}$ is a 2-category of relations.* ∎

We end this section with two remarks:
(i) the results raise the question whether a relationally closed database is a "2-category of relations" in the sense of Carboni and Walters [2] (= the 2-category $\mathbf{rel}(\mathbf{E})$ for a regular category $\mathbf{E}$) – the answer appears to be no;
(ii) the setting we have constructed will be useful for the expression of functional dependencies which are important in database design – the statement that there is a functional dependency becomes the categorical statement that a 1-cell in $\mathbf{R}(v,w)$ always has a right adjoint.

Conclusion

We conclude this article with some directions for further work. While the theory of indexed categories gives a satisfactory account of the theory of relational databases, viewing databases as 2-categories of relations begs the investigation of databases on a different data structure which appears in a bicategories closely related to 2-categories of relations, namely bicategories of profunctors. A (binary) relation from set A_1 to set A_2 is determined by stating whether or not an element of A_1 is related to an element of A_2. A profunctor generalizes this in two ways. First, A_1 and A_2 are categories and there is a set (not just a boolean truth value) relating an object of A_1 to an object of A_2. Second, this doubly indexed family of sets is acted on by arrows of the categories A_1 and A_2. Replacing the discrete objects used as domains in the relational theory by categories, and hence considering the profunctor data structure, may provide a suitable theoretical foundation for object-oriented databases. This observation is currently under investigation.

References

1. M. Barr, Fuzzy set theory and topos theory. Bull. Can. Math. Soc. **29** (1986), 501–508.

2. Aurelio Carboni and R. F. C. Walters, Cartesian bicategories 1. Journal of Pure and Applied Algebra **49** (1987), 11–32.

3. E. F. Codd, A relational model for large shared data banks. Comm. ACM **13** (6) (1970), 377–387.

4. C. J. Date, *Introduction to Database Systems, Fifth edition.* Addison-Wesley, 1990.

5. J. W. Gray, Fibered and Cofibered Categories, *Proceedings of the Conference on Categorical Algebra, La Jolla*, pages 21–83, Springer, 1966.

6. B. Hilken and D. E. Rydeheard, Indexed categories for program development. Cahiers de topologie et géometrie différentielle catégoriques, XXXII (1991), 165–185.

7. M. Hyland and A. M. Pitts. The Theory of Constructions: Categorical Semantics and Topos-Theoretical Models, *Categories in Computer Science and Logic,* Contemporary Math No. 92, American Mathematical Society, 1989.

8. Peter T. Johnstone, *Topos Theory*, Academic Press, 1977.

9. G. M. Kelly and R. Street, Review of the elements of 2-categories, *Lecture Notes in Math. 420,* pages 75–103. Springer-Verlag, 1974.

10. B. LeSaffre, *Structures algebriques dans les topos elementaires.* PhD thesis, U. de Paris 7, 1974.

11. D. Maier, *The Theory of Relational Databases*, Computer Science Press, 1983.

12. Eugenio Moggi, A category-theoretic account of program modules, Mathematical Structures in Computer Science **1** (1991), 103–139.

13. R. Paré and D. Schumacher, Abstract Families and the Adjoint Functor Theorems, *Indexed Categories and their Applications,* pages 1–125. Volume 661 of *Lecture Notes in Math.*, Springer-Verlag, 1978.

14. K. Raju and A. Majumdar, Fuzzy functional dependencies and losslessjoin decomposition of fuzzy relational database systems. ACM Transactions on Database Systems, **13** (1988), 129–166.

15. Robert Rosebrugh, On defining objects by recursion in a topos, Journal of Pure and Applied Algebra, **20** (1981), 325–335.

16. R. A. G. Seely, Categorical semantics for higher order polymorphic lambda calculus, Journal of Symbolic Logic, **52** (1987), 4.

17. R. Street, The formal theory of monads, Journal of Pure and Applied Algebra, **2** (1972), 149-168.

18. J. D. Ullman, *Principles of Database and Knowledge Base Systems*, Volumes 1 and 2, Computer Science Press, 1988.

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE
MOUNT ALLISON UNIVERSITY
SACKVILLE, N.B. CANADA
RROSEBRUGH@MTA.CA

and

DEPARTMENT OF MATHEMATICS, STATISTICS AND COMPUTING SCIENCES
DALHOUSIE UNIVERSITY
HALIFAX, N.S. CANADA
RJWOOD@CS.DAL.CA

References

[1] M. Barr. Fuzzy set theory and topos theory. *Bull. Can. Math Soc.*, 29:501–508, 1986.

[2] Aurelio Carboni and R. F. C. Walters. Cartesian bicategories 1. *Journal of Pure and Applied Algebra*, 49:11–32, 1987.

[3] E. F. Codd. A relational model for large shared data banks. *Comm. ACM*, 13(6):377–387, 1970.

[4] C. J. Date. *Introduction to Database Systems, Fifth Edition.* Addison-Wesley, 1990.

[5] J. W. Gray. *Proceedings of the Conference on Categorical Algebra, La Jolla*, chapter Fibered and Cofibered Categories, pages 21–83. Springer, 1966.

[6] B. Hilken and D.E. Rydeheard. Indexed categories for program development. *Cahiers de topologie et géometrie différentielle catégoriques*, XXXII:165–185, 1991.

[7] M. Hyland and A. M. Pitts. *Categories in Computer Science and Logic*, chapter The Theory of Constructions: Categorical Semantics and Topos-Theoretic Models. *Contemporay Math No. 92*, American Mathematical Society, 1989.

[8] Peter T. Johnstone. *Topos Theory.* Academic Press, 1977.

[9] G. M. Kelly and R. Street. *Lecture Notes in Math. 420*, chapter Review of the elements of 2-categories, pages 75–103. Springer-Verlag, 1974.

[10] B. LeSaffre. *Structures algebriques dans les topos elementaires.* PhD thesis, U. de Paris 7, 1974.

[11] D. Maier. *The Theory of Relational Databases.* Computer Science Press, 1983.

[12] Eugenio Moggi. A category-theoretic account of program modules. *Mathematical Structures in Computer Science*, 1:103–139, 1991.

[13] R. Paré and D. Schumacher. *Indexed Categories and their Applications*, chapter Abstract Families and the Adjoint Functor Theorems, pages 1–125. Volume 661 of *Lecture Notes in Math.*, Springer-Verlag, 1978.

[14] K. Raju and A. Majumdar. Fuzzy functional dependencies and losslessjoin decomposition of fuzzy relational database systems. *ACM Transactions on Database Systems*, 13:129–166, 1988.

[15] Robert Rosebrugh. On defining objects by recursion in a topos. *Journal of Pure and Applied Algebra*, 20:325–335, 1981.

[16] R.A.G. Seely. Categorical semantics for higher order polymorphic lambda calculus. *Journal of Symbolic Logic*, 52:4, 1987.

[17] R. Street. The formal theory of monads. *Journal of Pure and Applied Algebra*, 2:149–168, 1972.

[18] J. D. Ullmann. *Principles of database and knowledge base systems, Vol. 1 and 2.* Computer Science Press, 1988.