1. INTRODUCTION

Databases are very easy to describe in terms of category theory, if we assume that every table has a designated primary key column. In the following paragraphs we will give a rough outline of how this is done; the details are in Section ??

Suppose one has a database $D$. Write down the directed graph $G(D)$ whose vertices are the tables of $D$ and whose edges are foreign key constraints. That is if $T$ is a table in $D$ and $T$ has a column which links to another table $T'$, then we draw an edge from $T$ to $T'$. One can account for referential integrity by making $G(D)$ a category rather than a graph; i.e. two paths may be forced to be the same.

In the category $G(D)$, every arrow is called a key. The primary keys are the identity arrows; the foreign keys are the non-identity arrows (note that one may have a foreign key from a table to itself).

For each table we need a set of records, and foreign keys induce functions between these sets of records. In other words, the data in the database is given by a functor from $G(D)$ to the category of sets.

We are only missing one thing – attributes. Of course, this is not a problem. We in fact take $G(D)$ not just to be the tables in $D$ but also the set of attribute columns. This is not really more complicated since attributes can be considered as one-column tables if necessary, in which case the above discussion goes through as written.

Left to discuss are morphisms of databases, queries, etc. This will be covered below. We also explain how to turn a database into an RDF triple store.

2. DATABASES

Definition 2.1. A attribute declaration, denoted $(\mathbf{DT}, \pi)$ consists of a functor $\pi : \mathbf{DT} \to \mathbf{Sets}$ for some small category $\mathbf{DT}$.

A morphism of attribute declarations, denoted $f : (\mathbf{DT}, \pi) \to (\mathbf{DT}', \pi')$ consists of a functor $f : \mathbf{DT} \to \mathbf{DT}'$ such that $\pi' \circ f = \pi$.

Example 2.2. We are thinking of the attribute declaration as containing all standardized fields contained in the database. For example one might have an object ‘Date’ in $\mathbf{DT}$ where $\pi(\text{‘Date’})$ is the set of dates written in some format.

Definition 2.3. A database schema, denoted $\mathcal{S} = (\mathbf{DT}, \pi, S, i)$, consists of a attribute declaration $(\mathbf{DT}, \pi)$, a small category $S$, and a functor $i : \mathbf{DT} \to S$. 

This project was supported in part by a grant from the Office of Naval Research: N000140910466.
Suppose that $S' = (D^{T'}, \pi', S', i')$ is another database schema. A morphism of database schema, denoted $(f, g): S' \to S$, consists of a morphism of attribute declarations $f: (D^{T'}, \pi') \to (D^T, \pi)$ and a functor $g: S' \to S$ such that the diagram

\[
\begin{array}{ccc}
D^{T'} & \xrightarrow{f} & D^T \\
\downarrow{\pi'} & & \downarrow{\pi} \\
S' & \xrightarrow{g} & S
\end{array}
\]

commutes.

**Definition 2.4.** Let $S = (D^T, \pi, S, i)$ be a database schema. An object of $S$ is called an attribute of $S$ if it is in the image of $i$; otherwise it is called a table of $S$.

Let $f: X \to Y$ be a morphism in $S$. If $X$ and $Y$ are tables then $f$ is called a foreign key constraint unless $X = Y$ and $f = \text{id}_X$, in which case $f$ is called the primary key of $X$. If $X$ is a table and $Y$ is an attribute then $f$ is called a data column. If $X$ and $Y$ are both attributes then $f$ is called a known relationship of attributes. If $X$ is an attribute and $Y$ is a table then $f$ is called ???.

**Definition 2.5.** A database, denoted $(S, \delta)$ consists of a database schema $S = (D^T, \pi, S, i)$ and a functor $\delta: S \to \text{Sets}$ such that the diagram

\[
\begin{array}{ccc}
D^T & \xrightarrow{i} & S \\
\downarrow{\pi} & & \downarrow{\delta} \\
\text{Sets}
\end{array}
\]

commutes.

Let $(S', \delta')$ denote another database. A morphism of databases, denoted $(f, g, h): (S, \delta) \to (S', \delta')$

consists of a morphism $(f, g): S \to S'$ of database schema and a natural transformation $h: \delta \circ g \to \delta'$ of functors $S' \to \text{Sets}$, as in the diagram

**Remark 2.6.** The above definition is greatly simplified if the schema remains constant. Given a schema $S = (D^T, \pi, S, i)$, a database on $S$ is, as above, a functor $\delta: S \to \text{Sets}$ such that $\delta \circ i = \pi$. A morphism of databases, denoted $h: \delta \to \delta'$, is just a natural transformation of functors.
3. Manipulations and queries

As in simplicial databases [?, a morphism from one schema to another allows us to transfer data between them.

**Definition 3.1.** Let $g: S' \to S$ denote a functor. Given a functor $\delta: S \to \text{Sets}$, define $g' \delta: S' \to \text{Sets}$ to simply be $\delta \circ g$. Given a functor $s': S' \to \text{Sets}$ define $g_+ s': S \to \text{Sets}$ to be the right Kan extension of $\delta$ along $g$. We can also define $g_! \delta: S \to \text{Sets}$ to be the left Kan extension of $\delta$ along $g$.

**Remark 3.2.** Given $h: S' \to S$ and $\delta: S' \to \text{Sets}$, the right Kan extension $g_+ \delta: S \to \text{Sets}$ can be computed as follows. Given an object $s \in S$ consider the category $(h \downarrow s)$, in which an object is a pair $(s', \alpha)$ where $s' \in \text{Ob}(S')$ and $\alpha: h(s') \to s$ and morphisms are defined in the usual way. We define $g_+(s)$ to be the set

$$\lim_{(h(a) \to b) \in (h \downarrow s)} \delta(a).$$

This construction is functorial in $a$.

**Construction 3.3.** Taking the union of databases with the same schema is simply a matter of taking a colimit in the diagram of databases. Adding a new row to a table is an example of such a union of databases (except that in this case one of the two databases will consist of just one record in some tables and none in others).

**Construction 3.4.** Limits in the category of databases correspond to joins and selects. For details, see [?] or encourage the author to write up the details.

4. Conversion to RDF

The conversion of a database in this model to RDF graph is simple. Namely, given a database $(\text{DT}, \pi, S, i, \delta)$ one takes the Grothendieck construction $\text{Gr}(\delta)$. This is a category whose objects are pairs $(T, x)$, where either $T$ is a table of the database and $x$ is the primary-identifier of a record in it, or where $T$ is an attribute and $x$ is an element in its domain. An arrow in $\text{Gr}(\delta)$ is a foreign key constraint (or attribute). The resulting category may be viewed as a graph (called the RDF graph), though it really should be viewed as a category to ensure referential integrity.

References