A Relational Model of Incomplete Data without NULLs

Michael Johnson[1] **Stefano Kasangian**[2]

[1] Mathematics and Computer Science
Macquarie University,
Sydney, Australia 2109,
Email: mike@ics.mq.edu.au

[2] Dipartimento di Matematica
Università degli Studi di Milano,
Milan, Italy,
Email: stefano.kasangian@unimi.it

Abstract

The theoretical study of the relational model of data is ongoing and highly developed. Yet the vast majority of real databases include incomplete data, and the incomplete data is widely modelled using special flags called *nulls*. As noted many times by Date and others, the inclusion of nulls is not compatible with the relational model and invalidates many of the theoretical results as well as requiring a three-valued logic for query support. In category theoretic applications to computer science, partial functions are frequently modelled by using a special value approach (the *partial map classifier*), or by explicit reference to the *domain of definition subobject*. In a former edition of the CATS conference the first author and his colleague Rosebrugh proved a Morita equivalence theorem showing that for database modelling the two approaches are equivalent, *provided* the domain of definition subobject is complemented. In this paper we study the uncomplemented domain of definition approach (which is *not* equivalent to using special values). Our main results show that using uncomplemented domains of definition to model incomplete data is entirely compatible with the relational model and so leaves the well-developed theory applicable to real databases that use this approach. Furthermore, using uncomplemented domains of definition supports in-place updating, in stark contrast to special values, and, in a wide variety of circumstances, ensures the existence of cartesian and op-cartesian models which, as shown in a recent TCS article, are important for solving view update problems.

Keywords: Category theory, relational model, partiality, database special values

1 Introduction

One of the more remarkable mismatches between theory and practice is the distinction between the theory of the relational data model, and real world databases. The relational model (Codd 1970, 1990) has been extensively developed (for a survey see Date (2004)), but consistently assumes that complete data is available. In contrast real systems are replete with missing data, typically handled by adding a NULL which is used in place of the missing values. The theory of the relational model has been vital to the development of modern database systems. But the ubiquity of missing data, and the lack of applicability of the theory in the presence of NULLs, leave a significant mismatch which is often glossed over.

Missing data is difficult to handle in the relational model. As noted many times over the years, a tuple in which one coordinate has a missing value is not a tuple at all, and much of the theory of relations, and particularly any attempts to use them to calculate queries, becomes invalid. Nor does the NULL value resolve the issue. The null is either a member of all types, leading to positively absurd query results, or it is in some sense a meta-value in which case once again our tuples are not tuples. As noted long ago (Maier 1983) "It all makes sense if you squint a little and don't think too hard."

It is probably best to view many modern database systems as theoretically well-founded on the relational model provided that the data that they store is complete. Then, the notion of NULL is retro-fitted for practical purposes to deal with the unpleasant fact that in the real world we rarely have complete data to store. The result is a beautiful theory about reliable systems that are not built, since the systems that are needed include the retrofitted NULLs, and the theory then no longer applies.

The authors and others (Johnson & Rosebrugh 2007, Kasangian, Kelly & Vighi 2000) including Diskin, Piessens, Buneman, Phoa, and White have over a number of years been using insights from the branches of mathematics known as category theory and universal algebra to study database design and specification. Partial functions may be viewed as one example of "missing data", and category theory uses two common techniques to deal with partial functions (Johnstone 1977, page 28): The representability of partial maps by a partial map classifier and the explicit representation of the domain of definition as a subobject of the domain of the partial function. The two approaches are generally thought of as being equivalent. Indeed, aspects of such an equivalence are fundamental to the definition of partial map classifiers. However, in universal algebra, when one approach or the other is used for *specification*, there is a little studied difference as was demonstrated in the conditions required for a Morita equivalence theorem (Johnson & Rosebrugh 2003). The fundamental difference arises because complements of the domains of definition are not basic types — since they are not part of the specification, a simple update can insert a new instance in the domain of definition, effectively extending the partial function. In contrast, if complements of domains of definition were base types, then inserting into the domain of definition would require simultaneously deleting from its complement, and a single natural transformation can't do that.

In this paper we study the umcomplemented domain of definition approach, and explore how it can be used

The authors acknowledge gratefully financial support from the Australian Research Council and the Università degli Studi di Milano, and the helpful advice of anonymous referees.

at specification time to design systems that will support partial data *without* using NULLs. Our main results show that such an approach is compatible with the relational model. In particular we work through many of the fundamental relation algebra operations and show how they can be calculated category theoretically in the classical case, and we develop appropriate analogues of those operations for models which include possibly partial attributes, that is which support missing data.

The plan of the paper is as follows. In the next section we review the category theoretic representations of partial data and establish notation that we will use throughout the remainder of the paper. Section 3 considers the treatment of possibly partial attributes using the explicit domain of definition approach. In Section 4 we review how composition of apparently not composable morphisms is calculated using pullbacks, and observe that this is really just the standard composition of relations. Then in Sections 5-8 we consider in turn each of the operations of select, restrict, project and join and some of their interactions treating them both for the classical case with categorical techniques and for models which support missing data. Finally Sections 9 and 10 briefly review related work and draw conclusions including outlining some very recent findings adding further weight to the benefits of supporting missing data using the techniques presented here.

2 Category theoretic representation of partial data

For elementary category theoretic notions readers are referred to the usual texts (Barr & Wells 1990, Mac Lane 1971, Walters 1991) although there are now many others too. In this paper, in many cases, a reader's intuition from diagrams of sets and functions will suffice, so we will not take the space to review the formal notions. Nevertheless, it is important to remember that all of what follows has a fully rigorous mathematical treatment, and that the abstraction from functions to arbitrary categories is precisely what is required to properly support database specification.

Databases are frequently specified by indicating those real world *entities* (for example employees, products, equipment, etc) about which we might wish to store known *attributes* (addresses, salaries; locations product-numbers; owners, capacity; etc). In addition relationships between entities can also be stored (an employee might be responsible for selling particular products, an employee might be assigned the exclusive use of particular equipment, etc). The database itself might be thought of as a collection of functions with domains given by the entities about which information is stored, and codomains being the sets of possible values for their corresponding attributes, along with other functions which encode the relationships between the entities. As an example consider Figure 1 which can be thought of as a diagram of sets and functions to be stored. Figure 1 is a fragment of a flight control system database taken from the first author's work with his colleague Rosebrugh. In this case we have for simplicity not shown attributes which add a number of arrows starting from each entity. For example, a Runway has a direction, length, width, category, etc, and each of these attributes would appear as an arrow with domain Runway and various appropriate codomains (direction measured in compass degrees, length measured in metres, etc).

The most common kind of missing information is an unknown attribute value. For example, if an employee moves house without notifying us we may discover that his or her address is no longer valid, but not have any knowledge of the new home address to replace it with. The great bulk of incomplete information is like this, and can be modelled by assuming that the attribute value functions are partial functions. So, we will briefly review the two most common representations of partial functions in

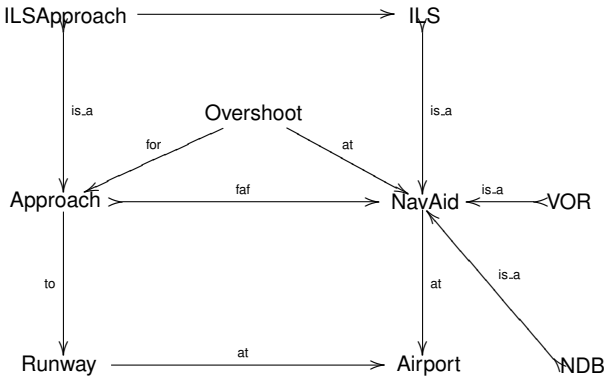


Figure 1: Part of a flight control system

category theory.

In this paper we will frequently indicate a partial function between an entity X and an attribute A by a barred arrow

$$f : X \nrightarrow A$$

which should be read as "f is a possibly partial function from X to A". Such a function can be represented using two total functions, one the function f restricted to its domain of definition, and the other the inclusion of that domain of definition, say X', into X. In other words

$$\frac{X \nrightarrow A}{X \hookleftarrow X' \rightarrow A}$$

where the horizontal line can, as usual, be read as indicating that it is equivalent to give either the data above the line (a partial function) or the data below the line (two total functions, one of which is monic as shown).

Notice the arrow with the extra tail? This is the standard notation for a *monic* arrow — in terms of sets and functions it represents an injective function. We will frequently call the domain of an injective function a *subobject* and treat the injective function as the inclusion of the subobject.

Incidentally, a span of functions is a standard representation for a relation. So, a possibly partial function $X \nrightarrow A$ is a relation X' from X to A which because of the monic arrow is either 0-valued or 1-valued, i.e. for each $x \in X$ there are precisely 0 or 1 elements $a \in A$ such that $xX'a$.

Briefly we outline the alternative approach. In the category of sets the partial map classifier for functions into a set A is the set $A + 1$, so

$$\frac{X \nrightarrow A}{X \rightarrow A + 1}$$

The extra element of the set $A + 1$ is analagous to the NULL used in many database systems — we replace the partial function f into A by a total function into $A + 1$ by sending all the elements of X for which f is undefined to the 'special' element 1.

In the remainder of this paper we will use the explicit subobject of definition approach, the span of two arrows one of which is monic, to represent possibly partial functions. And we repeat: despite common misconceptions, earlier work has shown that in Morita terms, that is in terms of the model theory and logic that they support, the two approaches are *not* equivalent.

3 Database specification with partially defined attributes

In a range of applications, the authors and their colleagues (particularly Rosebrugh and Dampney) have used cate-

gory theoretic specification of database systems to analyse and improve real systems, often as part of industrial consultancies. To systematise some of our techniques Johnson & Rosebrugh (2002) showed how to translate between category theoretic representation techniques based on universal algebra and traditional relational database design. Those techniques can be applied directly to convert categorical database specifications which include subobject of definition representations of possibly partial attributes into traditional relational databases that support missing data but don't use NULLs. This section will consider briefly guidelines for incorporating possibly partial attributes and identify issues of concern that will then be addressed in the remainder of the paper.

Firstly, in specifying a database we can decide which attributes will be required to be total, and which may be permitted to be partial. To distinguish between the two cases we simply have a single arrow $X \to A$ for attributes A of X which are required to be total, and a span

$$\frac{X \nrightarrow A}{X \leftarrowtail X' \to A}$$

for those attributes A of X which are permitted to be partial.

It is important that we have the choice. There are attributes which must never include missing data. The most obvious such attributes are *keys* — those values which are used to identify distinct instances of an entity. In addition, there may be other attributes which while not formally recorded as keys are for semantic reasons essential for a particular entity. In both cases we simply specify that the attribute is total.

On the other hand, very many, indeed most, attributes can be partial, even if this may be thought to be undesirable. Usually we don't plan how missing data might occur. We know that some missing data (for example missing key data) invalidates an entity instance's existence in the database, but other missing data arises simply through exigencies in the world, and if it can be supported by the database then it should be.

This suggests that to properly support missing data most attributes in most databases, each of which would normally be modelled by a single arrow, should be replaced by spans of arrows allowing an explicit subobject of definition for the attribute.

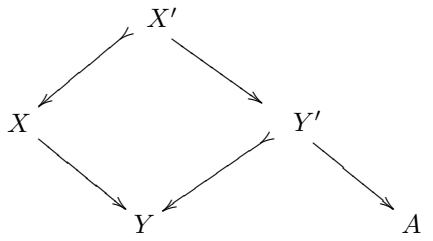
Now, replacing each possibly partial arrow $X \nrightarrow A$ with a span $X \leftarrowtail X' \to A$ might be expected to complicate an entity-relationship representation enormously, and would also spoil composition (composable $X \to Y \nrightarrow A$ becomes $X \to Y \leftarrowtail Y' \to A$, a not composable zig-zag of arrows). Furthermore the standard relational operations (select, project, join and so on) will need to be redefined, if possible, to take account of partiality.

The remainder of this paper will carefully consider each of these issues. As usual, we will conduct our analysis in category theoretic terms and the implications for databases can then be read off via the translations of Johnson & Rosebrugh (2002).

4 Composition involving partial attributes

The first concern is easily dealt with. In all our category theoretic specifications we use pullbacks (see for example Barr & Wells (1990) for a definition and tutorial treatment). Indeed, even specifying that an arrow is monic, as we need to do to indicate possible partiality using an explicit subobject of definition, is done with pullbacks. We can calculate pullbacks whenever they might be required. In the category of sets and functions, a pullback like X' in the diagram below can be calculated as the subset of the product of X and Y' determined by requiring that the two components agree at Y. It is easy to see that special cases of pullbacks include selections as shown in Section 5, and the intersections that are used in Sections 6–8.

Now the composition above can be calculated using a pullback. The zig-zag of arrows identified in the last section is the lower half of the following diagram and the square is to be a pullback.



Let us interpret that answer and check its validity. Why is there an X' generated by the pullback and what does it mean? Well, it only makes sense to compose the arrow $X \to Y \nrightarrow A$ for those $x \in X$ which are sent by the first arrow to a Y value for which the partial arrow is defined. So the result is a partial arrow $X \nrightarrow A$, and its domain of definition is X'. Furthermore, the pullback calculates as X' precisely those X values for which composition can be defined. Lastly, X' really is a subobject of X using the easy result that a pullback of a monic is monic.

It's easy to modify this calculation to calculate composites of partial and total functions in the other order, or indeed of two partial functions.

Of course many readers will recognise the above calculation as a special case of classical relational composition: Two relations R and S are composed by forming the composite relation RS with the property that $xRSy$ if an only if there is a z with xRz and zSy. Such compositions can always be calculated by pullback, and in the case we've looked at in detail the first relation is in fact functional, allowing it to be represented as a single arrow rather than a span.

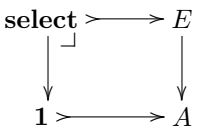
From now on we will calculate compositions of zig-zags using pullbacks without further comment.

Next we consider the relational operations and how to calculate each one in turn in normal circumstances (when all attributes are total) and how to modify that calculation in order to take account of possible partiality, that is, of possibly missing data.

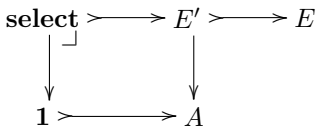
5 Select with and without partiality

The basic case of a database select operation is, given an entity E and one of its attributes $E \to A$, along with an attribute value $a \in A$, to identify those instances of E which have associated attribute value a. In category theoretic terms the specification of a single attribute value a is given by an arrow $1 \to A$ from the terminal object into A. That arrow "picks-out" a as its image in A.

The select operation is again a pullback as shown (note that arrows out of a terminal object are automatically monic, and that pullbacks of monics are monic, and so the object labelled select is guaranteed to be a subobject of E):



But now, how might that operation be changed if $E \to A$ is replaced by a possibly partial attribute $E \nrightarrow A$? The selection can after all only take place from among those entity instances which have the attribute defined, so we can calculate the selection in the presence of a possibly partial attribute as follows.

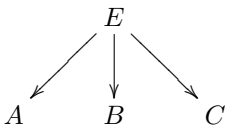


Of course, the result is a subobject of the domain of definition object E', but since being a subobject is transitive, or equivalently since monics compose to give monics, the result is, as we would want, a subobject of E.

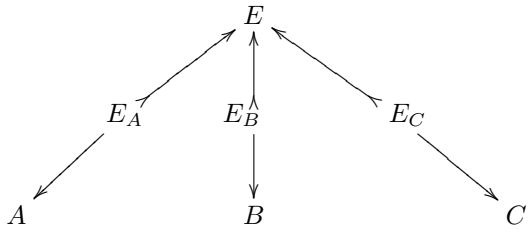
Easy extensions of the select operation using category theoretic techniques support the restriction operations (`where` clauses) and these are not changed by the introduction of possibly partial attributes so we will not treat them in more detail.

6 Project with and without partiality

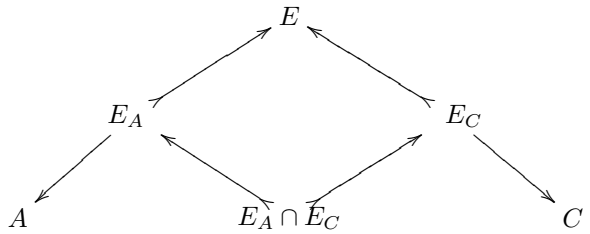
The project operator amounts to choosing to ignore, or "project off" certain attributes. For example, an entity E with attributes A, B, C and D might, via the project operator, yield the same entity with only attributes A, B, and C.



If some or all of the attributes are possibly partial a similar result arises as shown.



As an alternative, we might choose to project onto the attributes A and C, but ask that the projection result in a relation in which for each instance of the entity E the two attributes either both have defined values together, or are not defined. In that case we use a pullback to determine the common domain of definition of both A and C as a subobject of E, as shown.

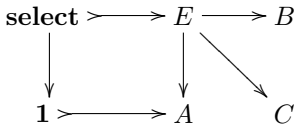


Notice that the "intersection" subobject is a subobject, using again the transitivity of subobjects, of E, and on it values of attributes A and C are defined by composing the two lower leftward arrows and the two lower rightward arrows respectively. Thus A and C appear as possibly partial attributes with a common domain of definition.

It's very important that in the presence of partiality there are analogues of the standard relational algebra operations. This section shows how, in the presence of partiality, there are richer choices and there may be several different but useful analogues available.

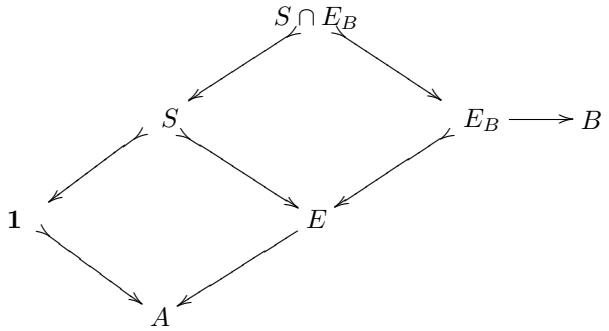
7 Select and project together with partiality

In the fully defined context select and project work together without any further ado.

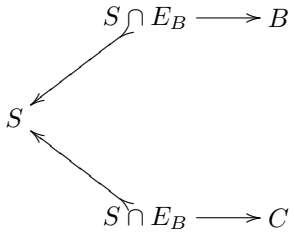


Notice that the composition with the inclusion of the select into the entity E ensures that the attributes B and C, assuming that we choose to keep them in our projection, are still attributes of the selected subobject of E.

On the other hand, writing S for the selected subobject of E, if an attribute B say is possibly partial the selected subobject does *not* necessarily have the domain of definition of B, E_B as a subobject. So, a priori B is not inherited as a possibly partial attribute of S. Nevertheless, calculating a second pullback, the upper diamond in the following diagram, resolves the issue. The second pullback, being the intersection of the selected entities S and the domain of definition E_B of B, exhibits B as a possibly partial attribute of S.



Similarly if in addition our projection retains the attribute C, and if C is originally a possibly partial attribute of E, we obtain two possibly partial attributes of S as shown.

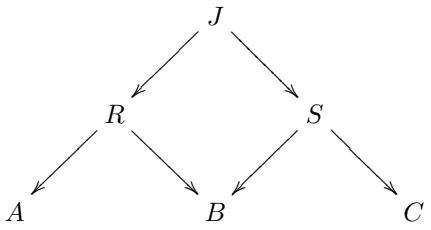


Once again, if our intention was to project onto the attributes B and C so as to obtain commonly defined attribute values (see the last example in the previous section) we would proceed by calculating yet another pullback in the diagram above to find the intersection of the two intersection subobjects, and that would be the common domain of definition for the possibly partial attribute B and C taken together.

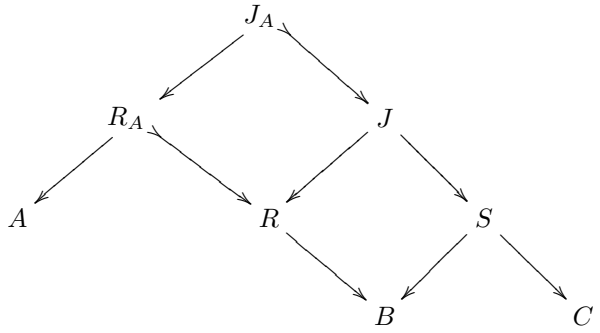
8 Join with and without partiality

We turn now to joins. Forming a join over the attribute value B say can be seen to be calculated by a pullback as shown. We have included also attributes A of the relation R and C of the relation S to illustrate the fact that in the classical case both A and C appear as attributes of the

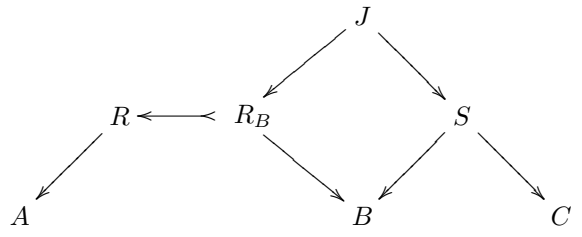
join J.



Now suppose that A is a possibly partial attribute of R with R_A the subobject of R giving the domain of definition of the attribute. How does the operation change? Again we can calculate the join J. Again C is an attribute of the join automatically. But a priori A is not an attribute of J. Instead we need to calculate a second pullback to find J_A the subobject of J on which attribute A is defined. This exhibits A as a possibly partial attribute of the join.



So we can see how to treat attributes that are inherited, whether possibly partial attributes or total attributes, by the join. But what if the attribute that we are joining over, B is possibly partial, say for relation R? Then rather than forming the pullback over R we can only calculate it over the domain of definition R_B of B as shown.



This is in fact what we would want. As you can see, the join has been calculated over the defined values only. (Indeed, joining over undefined values is one of the peculiarities that arises in the presence of NULLs that Date and others have analysed in detail and railed against.)

Notice that the join still inherits attributes C, as usual, and A via the relation R of which R_B is a subobject.

9 Related work

Three approaches to partiality in database systems were studied by Johnson & Rosebrugh (2003). Two of the approaches are those described in Section 3, one of which corresponds to the approach taken here and another of which corresponds to the use of nulls. The third approach was an attempt in the style of domain theory to incorporate missing value information in the value sets of attributes themselves. The main theorem was a Morita equivalence theorem for all three approaches provided that explicit subobjects of definition are required to be complemented.

It is only in recent work exploring the dynamics of database systems, especially in the light of view updating, that it has become apparent that subobjects of definition must not be complemented at specification time. Using such subobjects, as in this paper, is no longer Morita equivalent to the other two approaches, and this is the first paper to study in detail the resulting relational algebra operations.

A different version of the domain theoretic approach appeared in (Libkin 1991). Further work will be required to determine whether the Morita equivalence theorem applies to Libkin's formulation, or whether it too extends beyond the NULLs approach, and if the latter, whether it is Morita equivalent to the approach described here.

Many other authors have decried the mistreatment of the relational model forced upon practitioners by systems which use NULL values. Date is the foremost author in that group, and he reviews much of the other work (Date 2004). He also has proposals for distinguishing special values from NULL values, and for restructuring query logics, but those ideas remain Morita equivalent to models that incorporate NULL values.

Date has also suggested in an example a relational representation (but without explicit 0 or 1-valuedness) of an $Employee \twoheadrightarrow Dept$ attribute. In a sense we are here taking that suggestion seriously and pushing it to its limits.

Another approach to partiality is presented in a sequence of papers on *restriction categories*. See (Cockett & Lack 2009) and papers cited therein. As yet that approach has not been seriously applied to databases.

10 Conclusion

This paper has examined in detail some of the effects of permitting partiality, represented by explicitly modelling domains of definition, and hence of modelling partial functions as necessarily 0- or 1-valued relations. We have concentrated on the relation algebra operations that don't involve difference since these are the ones that have been demonstrated to be of practical importance in category theoretic modelling of real systems.

The main results that we note here are as follows. The first two are demonstrated in detail in the paper and the second two are recently completed findings which will be reported in detail elsewhere.

1. Such an approach does no violence to the relational model. In particular, select, restrict, project and join work as expected with only minor modifications (extensions) to take account of spans
2. The new approach is mathematically well-founded being based on elementary constructions from category theory
3. The new approach supports "in-place updating of (non key) attribute values" something category theoretic models of data were long supposed not to do
4. We have also proved that in a wide variety of circumstances cartesian and op-cartesian models (Barr & Wells 1990) exist when partiality via domain of definition relations is supported. This means that the category theoretic solutions to view update problems (Johnson & Rosebrugh 2007) are even more widely available.

In contrast, the use of NULLs or special values is extra relational (contrary to 1) and conflicts with category theoretic view updating (contrary to 4).

In summary: Earlier approaches to incomplete data have generally retrofitted NULLs or other special values when they are unavoidable even if they are contrary to the data model's theoretical foundations. Instead we've adopted the slogan

> If it's not a key, an attribute should be allowed to be partial unless there is a special semantic reason to require it to be total

and explained how to support that within a relational framework.

References

Barr, Michael & Wells, Charles (1990), *Category theory for computing science*, Prentice Hall.

Cockett, J.R.B. & Lack, S. (2009), 'Restriction categories III: colimits, partial limits and extensivity', *Mathematical Structures in Computer Science*, to appear.

Codd, E.F. (1970), 'A relational model of data for large shared data banks', *Communications of the ACM* **13**(6), 377–387.

Codd, E.F. (1990), *The relational model for database management version 2*, Addison-Wesley.

Date, C.J. (2004), *An introduction to database systems*, Addison-Wesley.

Johnson, Michael & Rosebrugh, Robert (2002), 'Sketch Data Models, Relational Schema and Data Specifications', *ENTCS* **61**, 1–13.

Johnson, Michael & Rosebrugh, Robert (2003), 'Three approaches to partiality in the sketch data model', *ENTCS* **78**, 1–18.

Johnson, Michael & Rosebrugh, Robert (2007), 'Fibrations and Universal View Updatability', *Theoretical Computer Science* **388**, 109–129.

Kasangian, Stefano, Kelly, G.M. & Vighi, Veronica (2000), 'A bicategorical approach to information flow and security', *Rendiconti di Circolo Matematico di Palermo* **64**, 99–122.

Johnstone, P.T. (1977), *Topos theory*, Academic Press.

Libkin, L. (1991), 'A relational algebra for complex objects based on partial information', *Springer LNCS* **495**, 29–43.

Mac Lane, Saunders (1971), *Categories for the working mathematician*, Springer.

Maier, D. (1983), *The theory of relational databases*, Computer Science Press.

Walters, R.F.C. (1991), *Categories and computer science*, Cambridge University Press.