ELSEVIER

Computers & Geosciences 32 (2006) 1368–1377

COMPUTERS & GEOSCIENCES

www.elsevier.com/locate/cageo

The problem of missing data in geoscience databases

Stephen Henley*

Resources Computing International Ltd., 185 Starkholmes Road, Matlock, Derbyshire DE4 5JA, UK

Received 14 September 2005; received in revised form 5 December 2005; accepted 16 December 2005

Abstract

SQL is the (more or less) standardised language that is used by the majority of commercial database management systems. However, it is seriously flawed, as has been documented in detail by Date, Darwen, Pascal, and others. One of the most serious problems with SQL is the way it handles missing data. It uses a special value 'NULL' to represent data items whose value is not known. This can have a variety of meanings in different circumstances (such as 'inapplicable' or 'unknown'). The SQL language also allows an 'unknown' truth value in logical expressions. The resulting incomplete three-valued logic leads to inconsistencies in data handling within relational database management systems. Relational database theorists advocate that a strict two-valued logic (true/false) be used instead, with prohibition of the use of NULL, and justify this stance by assertion that it is a true representation of the 'real world'. Nevertheless, in real geoscience data there is a complete gradation between exact values and missing data: for example, geochemical analyses are inexact (and the uncertainty should be recorded); the precision of numeric or textual data may also be expressed qualitatively by terms such as 'approximately' or 'possibly'. Furthermore, some data are by their nature incomplete: for example, where samples could not be collected or measurements could not be taken because of inaccessibility.

It is proposed in this paper that the best way to handle such data sets is to replace the closed-world assumption and its concomitant strict two-valued logic, upon which the present relational database model is based, by the open-world assumption which allows for other logical values in addition to the extremes of 'true' and 'false'. Possible frameworks for such a system are explored, and could use Codd's 'marks', Darwen's approach (recording the status of information known about each data item), or other approaches such as fuzzy logic.

Keywords: Relational database; Open-world assumption; Closed-world assumption; Missing data; SQL; Logic; Fuzzy logic

1. Introduction

> Reports that say that something hasn't happened are always interesting to me, because as we know, there are known knowns; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns—the ones we don't know we don't know—Donald Rumsfeld, US Secretary of Defense

In the geosciences (as in other observational sciences, and as in military intelligence), it is very common for observational data sets to be incomplete. Data items may be missing altogether, or they may be imprecise in one way or another. There are many things we do not know—and many more that we do not know we do not know.

*Tel.: +44 1629 581454.
E-mail address: stephen.henley@resourcescomputing.com.

0098-3004/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cageo.2005.12.008

If the data are tabulated, then whether a database management system is used or not, some placeholder must be used to indicate the absence of a data item. In systems which are based on SQL, this placeholder is known as 'NULL' and conventionally represented by an empty character string. However, from the earliest days of relational databases it has been recognised that there can be logical problems in handling relations in which there are NULLs. Indeed, relational database theorists (for example, Date, 1995, 2005; Date and Darwen, 1998, 2000; Pascal[1]) have insisted for many years that a table which contains NULLs is not even a relation, because they assert that the predicate (the logical statement), corresponding to any tuple (row) in which there are NULLs, has no meaning.

In order to deal with missing data, Pascal (see footnote 1) advocates an absolutist approach which eliminates the use of 'NULL' altogether. His solution is to use a table containing the data (as a user I/O medium—not constituting any part of the database)—with nothing at all defined for data items which are missing, but rather a separate audit trail table listing the tuples and attributes which are missing. The main data table, as long as it contains 'holes' (i.e. NULLs) is not accepted as a relation but is merely a table, from which (using the audit trail data) a set of valid relations can be created by partitioning the table into a number of sub-tables—each of which is a valid relation because it does not contain any nulls. Unfortunately, the number of relations which can be required increases steeply (2^m) with the number m of attributes that can contain missing values. If it is required to carry out operations such as a relational join on two or more such tables, the number of relations needed can soon become astronomical. Pascal dismisses this complexity as something which can be hidden from the user and automated within the database management system implementation, although of course the problem still remains, even if hidden.

However, it seems that the complexities of this approach may be unnecessary if the predicate logic proposition for a tuple is modified a little. For example, the example proposition which he presents (p. 10):

> Employee uniquely identified by employee number (EMP#) has name (ENAME) works in department (DEPT#), was hired on (HIREDATE), earns salary (SALARY).

could be replaced by

> Employee uniquely identified by employee number (EMP#) has name (ENAME) *is reported to work* in department (DEPT#), *is reported to have been* hired on (HIREDATE), *is reported to earn* salary (SALARY).

As we shall see later, this re-formulation is very important. The predicate logic used in relational databases should be a set of statements about our knowledge of the world rather than statements of the 'absolute truth' which is usually unknowable. Indeed, if they were statements about 'absolute truth', then this would invalidate the use of relational databases for all scientific applications—as well as for military intelligence and many business applications.

In the above example, if (SALARY) is unknown, and represented by 'NULL', then the new formulation is still true and still records a valid and meaningful fact in its entirety: '…is reported to earn a salary the actual value of which we do not know at present'.

2. Darwen's proposal

Another way that has been proposed to circumvent the logical problems associated with representing missing data was suggested by Darwen.[2] In Darwen's example, although he reaches this by a roundabout route involving horizontal and vertical decompositions of the original relation, addition of extra attributes, and subsequent recombination, the effect is that a 'salary' attribute is replaced by an attribute of 'sal_info' containing *information about* salaries—which will be either the salary itself if known, or words such as 'salary unknown' if unknown or 'unsalaried' if inapplicable. The type of such an attribute must be defined to allow both the full range of possible actual values AND the full set of descriptors that must be used in the absence of an actual value. This is similar to the solution proposed above. Such an interpretation of the data in a table may allow it to retain its full relational credentials, but of course this is at the cost of requiring that either applications or the database

[1]Pascal, F: The Final Null in the Coffin, 2004, http://www.dbdebunk.com

[2]http://www.TheThirdManifesto.com: How to handle missing information without using nulls: presented at Warwick University, 9 May 2003.

management system itself recognise and have rules for handling the missing-data cases (such as 'salary unknown').

This solution suggested by Darwen implies the creation of more complex attribute types, which are internally heterogeneous, in that the action of operators defined for these types varies depending on the value of the operand—for example, simple arithmetic operators on known integer values, or one of a set of logical operators on 'null' or other non-integer (descriptive text) values within the same compound type.

It is of course possible (as Darwen himself envisaged in his proposal) to use Darwen's decomposed set of relations, in which there is one relation (SALARY_UNK) containing a list of all employees whose salary is unknown, and another (UNSALARIED) containing a list of all who do not receive a salary. This certainly answers the problem of *representation* of the missing data without needing any modification of type definitions and without violating the closed-world assumption or requiring more than two-valued logic. However, it is not clear whether and how it resolves the problem of *manipulating* such a data set, as can be demonstrated.

In a real, geoscientific, case—for example, a geochemical survey where a large number of samples have been analysed for say 50 chemical elements—it is very common for analyses to be temporarily or permanently missing for many elements in different samples. Were Darwen's decomposition solution to be adopted, this would lead to 50 'X-UNK' relations, each containing a list of samples for which the analysis of chemical element X is missing, and horizontal decomposition of the original relation into perhaps a very large number of relations, one for each different combination of attributes (chemical elements) in which there are missing data. Apart from the complexity of the data management (which might conceivably be automated), there remains the problem of defining the results of operators on the missing data.

In particular, it is unclear how such a solution would help in selection of all samples 'WHERE Mg > Ca' if the values for either (or both) of Mg and Ca are missing in some samples. Whatever the data organisation, and however the original relation might have been decomposed to hide the fact that data are missing, the correct result from such an operator is the truth value 'unknown'—prohibited in the two-valued logic of the relational model.

3. Closed world and open world

Relational database theory was developed mainly in the context of business applications, in which the universe of discourse can be pre-defined, and the absolute truth of all relevant facts is expected to be known. For example, it makes no sense to have a table of *some* employees, or even a table of all employees in which only *some* of the relevant facts are recorded for some of them. A table of employees should contain a complete pre-defined set of data for all employees in a department, or in a company (or in the universe of discourse, which must be specified). This is known as a *closed-world* model. A simple definition of a closed-world model is that anything not represented in the database is automatically defined as false. This is an integral part of the relational database model as currently defined by Date and Darwen (1998, 2000).

Date and Darwen (1998) make it clear in their statement (p. 136) that the relational database model is built upon the closed-world assumption:

- *If t appears in the body of r, then it is a true instantiation of the predicate (i.e. the corresponding proposition is considered to be true)*;
- *conversely, if t does not appear in the body of r, then it is a false instantiation (i.e. the corresponding proposition is considered to be false)*,

where *r* is any relation and *t* is any possible tuple that conforms to the heading of *r*.

This is made even more explicit in Date et al. (2003, Chapter 1):

- *Furthermore, we subscribe, noncontroversially, to the* Closed-World Assumption*, which says that if a given tuple conforms to the relation heading but does not in fact appear in the relation body, then the corresponding proposition is understood by convention to be one that evaluates to false. In other words, the body of the relation contains all and only the tuples that correspond to propositions that evaluate to true.*

Thus anything within the defined universe of discourse which is not represented in the database is deemed to be false. This may be a necessary presumption in, for example, a table of employees or a table of components held in a warehouse. In such cases the tables are expected to be complete and correct—anyone missing from the 'employees'

table is not an employee, and any component not in the 'components' table is not held in the warehouse. Within a limited closed world of clearly defined business applications, perhaps subscribing to the closed-world assumption might also be noncontroversial.

The closed-world assumption is expressed in terms of existence or non-existence of tuples and not explicitly to missing data placeholders. However, since the effect of decompositions such as those proposed by Pascal or Darwen is to represent individual missing data placeholders as omitted tuples within decomposed relations, the closed-world assumption applies to these too.

In contrast to the closed-world model, when considering observational scientific data, by definition we are dealing with an *open-world* model (unless, of course, the *universe of discourse* is defined in a very limited and limiting way as simply the set of observations already made—see below). Absence of information from the database does not imply its falseness. It is always possible to record information additional to that which is already in the database. Further geochemical samples may be collected and analysed, or gravity measurements may be taken at new locations, or additional spectra may be obtained from stars not previously recorded.

A table in a scientific database will typically contain a set of observations, where each tuple (each row in the table) represents one observation of a number of parameters at a defined location and time or measured on a defined sample. It is obviously incorrect to assert that all other possible tuples (relating to other possible locations, times, and samples) are false: the observations have simply not been made, or are not yet recorded. However, it is impracticable to re-define the universe of discourse every time a new observation is made. Scientific data are represented *naturally* by an 'open-world' model.

There are fundamental logical differences between a closed-world database and an open-world database. Although it can be shown that in the closed-world case a two-valued logic should always be sufficient, this cannot be demonstrated in an open-world database.

In an open-world database model, a relation can be properly constructed—say from a table of geochemical analyses with attributes *Sample_Number*, $SiO_2\%$, $Al_2O_3\%$, … The database may (indeed should) be designed before any data have even been collected. However, if an observation (say one of the $Al_2O_3\%$ values for a particular sample) is missed perhaps because of an instrument malfunction, it is unreasonable to demand that the database be restructured simply to avoid the use of a NULL—especially if it is expected that a later measurement will be inserted into the space. Pascal is correct in that such a table, with a hole, if interpreted using simple (two-valued) predicate logic, under the conditions of the closed-world assumption, does not represent physical reality. However, it does correctly represent the state of *reporting* of physical reality.

McGoveran's response[3] to a suggestion that scientific data might better be represented using the open-world assumption, rather than the closed-world assumption that the relational model requires, fails to address this question directly, but offers a philosophical interpretation. His suggestion is that within the context of any scientific hypothesis the universe of discourse is defined and closed. A strict interpretation of his position is that it must be re-defined each and every time that a new observation is added to the data set. He suggests that each time this is done, 'the impact on other hypotheses (axioms and therefore database design in the database world)' must be re-evaluated in order to avoid drawing erroneous conclusions. Clearly he is not a practising scientist. The concept of reviewing scientific hypotheses or database design after every single new observation—though it might be considered desirable—is not how science is done. If this is what is required in order for scientific data to be managed through the relational database model as narrowly defined by Date and Darwen, then it may be more reasonable to suggest that perhaps their database model is inappropriate.

4. Codd's 1979 definition

It is interesting that the original developer of the relational model, E.F. Codd, in a paper (Codd, 1979) in which he gives his definition of the relational model which he names RM/T, does not restrict it to follow the closed-world assumption but allows also the open-world assumption which he recognises as necessary to handle relations with missing data. He allows databases, and even individual relations within one database, to be defined as 'closed world' or 'open world', and

[3]http://www.dbdebunk.com, *More on the "Final Null in the Coffin"*, 4 January 2005.

devotes considerable attention to the three-valued logic which is necessary for proper handling of missing data. However, subsequent work by Date and others has concentrated on a more rigorous, but much narrower, definition of the relational model based solely on the closed-world assumption.

5. Two-valued logic

The discussion that follows will examine in more detail the cases for and against the use of NULL in a closed-world database, before moving to the open-world database structures which are needed for observational science data.

Pascal (2000), as also Date (1995) and Date and Darwen (1998,2000) note that SQL handles missing data incorrectly by using an incomplete and inconsistent three-valued logic (3VL). Additionally, both Pascal and Date interpret Codd's proposal (Codd, 1990) separately to identify 'missing' and 'inapplicable' versions of NULL as leading to an even more unworkable 4VL.

However, they have perhaps already identified a solution, in their recognition that NULL, as an entry in a relation, is not at all the same thing as an 'unknown' logic value that might result from an SQL query. Perhaps there is no need to go beyond 2VL, however, many different types of NULL that one wishes to use. All that may be required is abandonment of the flawed SQL language, and the definition of a set of rules on the results of operations that involve NULLs of each particular flavour.

It is of course essential to maintain the requirement that no primary keys may include any missing data items. Then if any relational operation were to yield a result that might, in SQL, be returned as 'unknown', it is clearly not TRUE and must therefore be returned as FALSE. Pascal is quite correct in his insistence that NULL is not a data value and cannot be treated as if it were, hence, if working within the constraints of the closed-world assumption and a two-valued logic system, (almost) any logical expression that includes an operand with value NULL should return a FALSE value. The only case where an expression containing a NULL operand would return a TRUE value is the special case of the 'IS NULL' monadic operator.

So, is the 'IS NULL' (monadic—single-operand) operator itself valid? The question is whether it is valid to frame queries which include such conditions as

WHERE [*lithology_code*] *IS NULL*

or

WHERE [*Au_assay*] *IS NOT NULL*

These will always have definite true or false values. There seems no particular reason to prohibit them. What is important is to establish consistency of treatment in dyadic (two-operand) operators—such that comparisons in which one of the operands has NULL value will return FALSE—since the presence of a NULL operand means that the comparison cannot be satisfied.

In the case of an equality test such as

WHERE [*start_salary*] = [*end_salary*]

if one or both of the operands is NULL, the unknown actual value of one operand may be equal to the actual value (known or unknown) of the other operand. However, it is unsafe in such circumstances to select this tuple, and therefore the test must be evaluated to FALSE, even if there is a possibility that, were the values of both operands to be known, it would be found actually to be TRUE. If, of course, the test is an inequality, then likewise it would have to be evaluated as FALSE, because the NULL operand(s) would mean that the test cannot be satisfied. At no time is a 3VL 'unknown' truth value required. What is important is that the meanings of TRUE and FALSE be understood.

Because the closed-world assumption and its associated 2VL does not allow the luxury of uncertainty, although TRUE means 'true' in the generally accepted sense, FALSE could mean either 'known to be false' or 'unknown'. This is analogous to the English system of criminal justice,[4] where a 'guilty' verdict corresponds to TRUE, while a 'not guilty' verdict, corresponding to FALSE, can mean innocent but can also mean 'not proven guilty beyond reasonable doubt'. Furthermore, if it is necessary in any particular case to distinguish between 'known to be false' and 'unknown', the IS NULL operator can be used.

6. More on the prohibition of NULL

The four principal protagonists in the debate over the prohibition of NULL have been Codd, Date,

[4]This is in contrast with the Scottish system, similar to a 3VL, with 'guilty', 'not guilty', and 'not proven'—the last corresponding to a 3VL 'unknown'.

Pascal, and Darwen. It is clear from Codd (1979, 1990) that the originator of the relational database concept not only saw no fundamental problem in using NULL (in the sense of a generic placeholder for missing data, rather than the specific SQL implementation of NULL), but on the contrary he understood that there were a number of different possible meanings of 'missing data'—though, if allowed, with the associated complication of using multi-valued logic.

Date and Darwen (1998, 2000), however, adopt the view that it is not legitimate to use NULL—and that by definition any table which includes NULLs is not a relation and forms no part of a relational database. Their proposed alternative—to provide a mechanism for representation of missing data—is to insert 'special values' and to require applications software to make sense of these as best it can. Unfortunately, Codd (1990) had already examined the special value ('default value' in Codd's terminology) approach to the problem and provided powerful arguments against it.

Pascal[5] goes further than Date and Darwen, and supports development of the 'transrelational (TM)' model which re-structures relations completely, effectively to a set of inverted indexes, one for each attribute, removing any need for representation of missing data. The status of this 'transrelational (TM)' approach, of which public knowledge at present is based solely on a patent application, is unknown at present. It is not yet commercially available due to unspecified legal problems. From what little has been published about it there seem to be serious performance issues,[6] apart from the likelihood that—if it really is nothing more than a set of inverted indexes—it is not new technology and hence not patentable.

Certainly, as Pascal identified in detail, the implementation of NULL in SQL-based systems is flawed (see Appendix A and discussion below), but the arguments against this particular query language are not necessarily valid criticisms of the use of one (or more) NULL markers in relational databases—whether built on closed-world or open-world assumptions.

[5]Pascal, F., The final null in the coffin? Outline of a relational solution to missing data, Practical Database Foundations #8, http://www.dbdebunk.com

[6]Hewitt, J., http://www.thetechtwo.com/detail-8879800.html and http://www.webservertalk.com/message480744.html

7. Prohibition of NULL in primary keys

In a relational database, each relation has a special attribute (or set of attributes) that is defined as the primary key. It is a pre-requisite that every tuple contains an unique primary key value. For this reason, the occurrence of NULL, or indeed any other placeholder for missing data, is necessarily forbidden in all primary keys. In scientific data, however, there are other types of 'partially missing' data or data which have some element of uncertainty. The extent to which these may be used in primary keys is something which needs to be examined in more detail. For example, one might wish to use sample location as the primary key. If this is a numeric co-ordinate vector (X, Y, Z), there will generally be some positional uncertainty in the recorded values. A relational operation on two such sets of data in different relations using equality of sample location (X, Y, Z) as the primary key cannot be guaranteed to give the correct links between samples, because of the likelihood that measurement discrepancies lead to slightly different recorded positions for samples actually collected at the same position—or to the same recorded positions for samples actually collected in two adjacent but different positions.

Hence, in order to maintain referential integrity of the database, the use of data containing any element of uncertainty *within a primary key* probably must be prohibited.

8. It is what you do with it that counts

Provided there are unambiguous and consistent logical rules for handling expressions which contain one or more NULLs, there should not be any need for a prohibition on using NULL, or indeed using any number of different 'flavours' of NULL with different meanings (like Codd's '*missing but applicable*' and '*missing and inapplicable*'). If working in a conventional relational database management system, under the closed-world assumption, the pre-requisite is that this system adheres strictly to two-valued logic.

A consequence of this would be that, if NULL is an acceptable placeholder for a missing data item, then the re-structuring of relations, as Pascal (see footnote 5) requires, is completely unnecessary.

However, Pascal also makes a series of criticisms of the way in which SQL handles NULL, and these criticisms must be considered insofar as they are

potentially relevant to the use of a generic placeholder 'NULL' itself (see Appendix A). It must be noted that SQL in itself is a language developed in a (flawed) attempt to implement the relational database model. Although it has been taken up very widely by the developers of database management systems, SQL must never be confused with relational database theory. As a corollary, criticisms of the handling of NULL in SQL, even if valid, are not necessarily criticisms of the use of a 'NULL' in relational databases. Pascal's paper[7] contains (page 9) a set of 10 specific criticisms of NULL as implemented in SQL and its incomplete and inconsistent 3VL. These are examined in detail in Appendix A. The conclusion from this examination is that none of Pascal's criticisms of SQL's NULL have any general validity as criticisms of the use of a generic missing data placeholder in relational databases—though they do contribute to a very strong case that SQL does not offer an appropriate environment for handling databases whose relations may contain NULLs. They are criticisms of SQL, not of the use of NULL.

Codd (1979, 1990) intended to develop a richer and more useful method of defining missing data placeholders, in the *I* and *A* 'marks' which he proposed. These are not in themselves NULL in the sense that NULL is used in SQL, but include all of the meaning of NULL (i.e. they are placeholders for data elements which are missing for one reason or another) while allowing more detailed representation of the reasons for absence of data. It can be and has been argued that Codd's *I* mark (for 'missing and inapplicable') should not be needed—because it can be removed simply by modifying the database design. Nevertheless, the argument made by Date (1995) that Codd's proposal requires an (unworkable) four-valued logic system is perhaps based on a misconception. However many varieties of missing-data representation might be defined, there may be no need to extend the logic by which they are processed beyond the normal 2VL—as has been shown above for the case of a single type of NULL—provided that one can accept that the result of any dyadic operator, any of whose operands is a NULL, a mark, or other missing-data placeholder, is false. This will yield a relational database model which has well-defined properties. One of these properties is that sometimes a comparison involving a missing data placeholder will return a false truth value when (if the data value had been known) the operator should actually have returned true.

All of this discussion, so far, concerns 'closed-world' databases. Here a 2VL can be defined to be always sufficient whether or not NULLs are allowed, though at the cost of failing to allow any distinction between 'false' and 'unknown'.

9. The universe of discourse

Closed-world relational database systems are necessarily defined to occupy and to fill a defined 'universe of discourse'. For example, the universe of discourse for a database of petroleum wells might be 'Production Wells of Acme Oil Ltd. on 31 July 2005'. This would then be defined as a complete set of the relevant information about the company's production as a snapshot on the specified date. Every well owned by the company on that date would be included, and the absence of a well would mean, by definition, that it was not a producing well owned by the company on that date. This is, in a very strict sense, a closed-world database. Absence of a tuple implies falseness of the corresponding proposition.

The universe of discourse could, however, be defined more broadly, for example, to allow the contents of relations to be time-varying (reflecting different states of ownership and production), as 'Wells of Acme Oil Ltd.' and this might then also include relations containing information about dry wells owned by the company, wells formerly producing but now exhausted, and wells formerly but no longer owned by the company. Absence of a tuple for a particular well should again imply that the specified well is not and was never owned by the company.

If Acme Oil Ltd. is a very old company, the contents of such a database may be incomplete. Records may have been lost, or data may be unreliable due to transcription errors. The absence of a tuple then does not imply falseness. This is no longer a closed-world database. Of course the universe of discourse could be redefined as 'All the data we can locate about wells of Acme Oil Ltd.', but this is a very imprecise definition which will change unpredictably. The absence of a tuple implies falseness only in the sense that the specified well is not in the set of 'wells known to have been

[7]Pascal, F., The final null in the coffin? Outline of a relational solution to missing data, Practical Database Foundations #8, http://www.dbdebunk.com

owned by the company at some time and whose data have been transcribed correctly'.

With databases in observational sciences such as geology the universe of discourse is commonly drawn even more widely: for example, 'Geochemistry of Tertiary volcanic rocks in north-west Scotland'. Clearly this is not a closed-world model, as it is always possible to add new tuples to relations in such a database, and indeed to add new relations representing different sample types, different sets of analysed elements, or different analytical methods.

10. Open-world database management

Under the open-world assumption, the absence of a tuple from the database does not imply its falsehood:

- *Any proposition which cannot be derived from the facts and axioms present in the system is held to be* unknown.[8] *Things which are known to be true or false must be stated explicitly, or else inferrable from facts and axioms.* (*Contrast this with the Closed-World assumption, which holds that all which cannot be proven true is false*).

The otherwise 'excluded middle' must be handled in some way. The Open-World assumption allows for real life uncertainty and imprecision in the data.

It has been proposed (Henley, 2005) that a variety of forms of placeholder for missing and partially missing data be established, based on the mark idea proposed by Codd (1990). Representation issues (i.e. how such placeholders are actually stored in any real database) can be addressed as implementation questions. For example, there is one particular issue which needs to be addressed—whether the relational model itself allows the use of 'marks' as flags on each data element. There is also, of course, a much larger issue, whether open-world databases can even be considered as relational—though Codd (1979) clearly intended that they should.

All that needs to be added, but it is quite a large 'all', is the processing intelligence needed to manipulate such data in order to generate correct and useful results. Quite clearly, from what has been written above, SQL or indeed any similar database query/manipulation language already in existence for business applications, will probably be inadequate. Scientific data tend to be numeric and to require extensive and intensive numerical processing.

What has to be provided as a database language therefore includes two essential elements:-

(1) a data manipulation language which recognises and can correctly operate upon relations which include various flavours of missing and incomplete data
(2) a numerical data processing capability which is fully integrated with the database management system to provide a means of extracting valid deductions.

This second aspect is crucial. There are a number of possible approaches to providing this, but one which appears to offer promise is that of fuzzy logic (Bardossy and Fodor, 2004). This has the advantage that it is grounded in rigorous logic theory and in principle it should be possible to include within the database model framework.

However, the generality of such an approach is limited, because the way in which fuzzy methods are used requires the application of subjective judgment, in choosing thresholds, distribution curves, etc. There may, of course, not be any option but to accept this limitation. In any event, it will be desirable to keep such subjective decisions visible, and any database implementation using such methods must of course keep such subjective information totally separate from the data, if for no other reason than the need to allow different scientists to draw different (but perhaps equally valid) conclusions from the same database, based on their own models and hypotheses.

There are other possible approaches to providing the richer logical and computational framework needed for open-world database management systems. One which has been (partially) implemented is to integrate general-purpose computational capabilities—numerical processing including statistics—with the database management system. This was done first with the G-Exec system developed for geoscience data handling applications (Jeffery and Gill, 1976a–c), and subsequently in Datamine, an integrated software system for mining industry applications (Henley & Stokes, 1983; Henley, 1992). In these two products, database management systems both formed the core of larger applications packages.

[8] http://c2.com/cgi/wiki?OpenWorldAssumption

11. Conclusions

There are severe problems in dealing with missing data in the relational model as commonly implemented in commercial products, especially those which are built around the SQL language, which is flawed in that it handles NULLs, or missing-value placeholders, inconsistently. However, even if the SQL problems are ignored, the relational database model, defined under the constraints of the closed-world assumption, may be unsuitable for handling real data from the observational sciences, which by their nature are incomplete and imprecise, with all gradations from exact ('crisp') numbers through imprecise and incomplete data to completely missing data items. It is, in general, not acceptable to insist on a re-design of the database to avoid the risk of missing data values in the middle of tables which would otherwise be perfectly good relations.

There are different possible solutions to this problem. The simplest is to ignore it, and treat any logical operation with a missing data operand as evaluating to false. Another way is to represent missing data by a series of 'special values' and pass all responsibility for their interpretation to the applications software. One might adopt a radical implementation solution such as the 'transrelational model (TM)' as advocated by Pascal. Another option might be to change the meaning of each attribute from the value of the attribute to a description of the state of knowledge of the attribute, which is the ultimate effect of the proposal by Darwen. However, there are problems with any of these approaches, as was pointed out by Codd (1990) and in the discussion above.

A more radical approach would be to abandon the total reliance upon the closed-world assumption which constrains the relational database model as now defined. The alternative open-world assumption allows for incompleteness and imprecision of the data. It will be necessary to define how such an open-world database management system might operate. The basic structure and functionality will necessarily be very similar to that of the conventional relational database model—because an open-world database in which there is actually no missing data ought to be identical, in practice, to a standard (closed-world) relational database. It is interesting that this may be close to Codd's (1979) RM/T definition. It will clearly need an extended query and manipulation language, which will have to include a range of types of numerical (and logical) functionality to handle imprecise and incomplete data (as Codd himself already envisaged in 1979). It may also include features drawn from fuzzy data handling methods, and will need to be sufficiently flexible—with user-definable functionality and operators—to handle an unlimited range of data types.

Acknowledgements

The author gratefully acknowledges useful discussions with and helpful suggestions by Professor George Bardossy, Dr. T. Victor Loudon, and Dr. Peter Robson.

Appendix A. Notes on Pascal's criticisms of NULL as implemented in SQL-based database management systems

Pascal's paper[1] contains (page 9) a set of 10 specific criticisms (*italicised* in the list below) of SQL and its incomplete and inconsistent 3VL. These are examined in detail in Appendix A. Each of Pascal's items will be examined below to identify any general points which might arise in relation to the use of NULL:

- *Aggregate functions, e.g. SUM( ), AVG( ) ignore NULLs, COUNT( ) does not.* However useful they may be, aggregate functions have nothing directly to do with relational database theory. Strictly speaking they are applications, and it should be possible to define whatever behaviour and rules are required into appropriate aggregate functions.
- *A scalar expression on a table without rows evaluates incorrectly to NULL, instead of 0.* This behaviour of SQL may indeed be wrong, but this says nothing about whether NULL should be acceptable or otherwise.
- *The expression NULL = NULL evaluates to NULL, yet ORDER BY treats NULLs as equal.* NULL = NULL correctly evaluates to NULL. The behaviour of ORDER BY is immaterial. In a relation the ordering of tuples is irrelevant—this is merely an export/presentation matter. In any case, the ordering of NULLs is by definition arbitrary and whatever ordering is chosen by SQL cannot be 'wrong'.
- *SQL's NOT is 'not' of natural language.* This is a problem with SQL, and not relevant to the NULL question.
- *It is unknown how SQL's EXISTS should behave,*

because no 3VL is defined for SQL (the definition is ambiguous in 3VL systems). Again this is an SQL problem because it attempts to use a 3VL, but is irrelevant to the question of the acceptability or otherwise of NULL in a relational database.

- *Expressions evaluating to NULL do not violate integrity constraints* (*NULL is treated as true*). NULL is of course not in itself a truth value, and expressions in a 2VL cannot evaluate to *unknown*. This is an SQL problem rather than a problem with NULL. The only expressions that can evaluate to NULL are those that are of a defined data type. Given a truth-value data type, it can indeed contain NULLs but these data items do not have any truth value at all—by definition—and they are certainly not of truth value 'unknown' in the SQL sense.
- *The truth-valued data type can take the values true, false, and the non-value NULL, but the literal NULL cannot appear in contexts in which any literals can appear*. This is a problem with SQL and SQL-based DBMSs. It violates the principle that NULL must not be capable of confusion with any valid data representation. This principle must be rigidly adhered to in any truly relational DBMS. Codd's marks, kept separately from the data, are one solution to this problem.
- *Since 3VL referencing rules are not defined, it is unknown which primitive operators out of the 27 monadic and 19,683 dyadic would suffice to express others and, therefore, whether all necessary operators are supported or not*. This is a possibly valid criticism of SQL's 3VL, but does not bear directly on the NULL question.
- *Highly complex operators*. This is a matter concerning SQL as an implementation language but irrelevant to the NULL question.

Thus none of Pascal's criticisms of SQL's NULLs have any general validity as criticism of the use of NULL in relational databases—though they do contribute to a very strong case that SQL does not offer an appropriate environment for handling databases whose relations may contain NULLs. They are criticisms of SQL, not of NULL.

References

Bardossy, G., Fodor, J., 2004. Evaluation of Uncertainties and Risks in Geology. Springer, Berlin (221pp).

Codd, E.F., 1979. Extending the database relational model to capture more meaning. ACM Transactions on Database Systems 4 (4), 397–434.

Codd, E.F., 1990. The Relational Model for Database Management: Version 2. Addison-Wesley, Reading, MA (538pp).

Date, C.J., 1991. Relational Database Writings, 1994–1995. Addison-Wesley, Reading, MA (542pp).

Date, C.J., 2005. Database in Depth: Relational Theory for Practitioners; O'Reilly. Sebastopol, CA, USA (208pp).

Date, C.J., Darwen, H., 1998. Foundation for Object/Relational Databases: The Third Manifesto. Addison-Wesley, Reading, MA (496pp).

Date, C.J., Darwen, H., 2000. Foundation for Future Database Systems: The Third Manifesto. Addison-Wesley, Reading, MA (608pp).

Date, C.J., Darwen, H., Lorentzos, N.A., 2003. Temporal Data and the Relational Model. Morgan Kaufmann, Amsterdam (422pp).

Henley, S., 1992. Orebody modelling in a relational database framework. In: Dowd, P.A., Royer, J.J. (Eds.), 2nd CODATA Conference on Geomathematics and Geostatistics, vol. 31. Sciences de la Terre, Ser.Inf., Nancy, pp. 477–484.

Henley, S., 2005. The man who wasn't there: the problem of partially missing data. Computers & Geosciences 31 (6), 780–785.

Henley, S., Stokes, W.P.C., 1983. Use of graphics computers in mine planning. In: Surface Mining and Quarrying. Institution of Mining and Metallurgy, London, pp. 323–328.

Jeffery, K.G., Gill, E.M., 1976a. The design philosophy of the G-EXEC system. Computers & Geosciences 2 (3), 345–346.

Jeffery, K.G., Gill, E.M., 1976b. The geological computer. Computers & Geosciences 2 (3), 347–349.

Jeffery, K.G., Gill, E.M., 1976c. The use of G-EXEC for resource analysis. Mathematical Geology 9 (3), 265–272.

Pascal, F., 2000. Practical Issues in Database Management: A Reference for the Thinking Practitioner. Addison-Wesley, Boston (256pp).