2VL and 3VL - how many operators are really needed ?

S. Henley - 27 Aug. 2006

Of the 16 dyadic operators of 2VL, only a small number are of practical use in database management systems. Most are of no importance. Those which are needed for data manipulation are AND and OR, which together with monadic operator NOT give all the functionality necessary. Why are these the only operators needed ? The reason is not that all other operators can be derived from them (as Date asserts) but that these are the operators which the user understands and finds useful, and which provide all the necessary capabilities.

In the same way, there is no reason to expect the user to want more than the same set of operators in a 3VL - however many thousands might in theory be available. A strange operator that yields weird results (which is what most of the 19xxx 3VL operators must be) is really of no interest - and Date's use of this large number to frighten people away from 3VL does a disservice to rational discussion of the need for a way to represent incompleteness of their data sets: a thing which is very hard (indeed I would assert impossible) with 2VL.

What may be needed is ultimately a more intelligent approach which allows representation of data with varying degrees of confidence - ranging from exact known values (as in the narrowly defined CWA 'relational' database of Date, Darwen, and Pascal), to completely missing data elements represented by placeholders (such as the SQL 'NULL') or missing tuples assumed to represent absence of information rather than falsehood; with in between, some way to represent data that have some degree of uncertainty or error. This was discussed in Henley (2005) but needs much further study. There have been a few attempts at developing 'fuzzy database systems' using the methods of fuzzy logic, but these have not been particularly successful and have not led to any mainstream products.

One of the principal difficulties with the 'fuzzy' approach is that, while the data manipulation and the computation of probability estimates are straightforward, the method is always dependent upon subjective selection by the user (or in a dbms perhaps by the database administrator) of probability distribution function and all of its parameters, for each attribute in each tuple. This imposes a major subjective input which could subvert the database by making the returned probability values unacceptable to users of the data.

An alternative approach which is possible with some data - such as commonly in geology - is to encode any known 'hard' constrainbts such as maxima and/or minima, but to leave the value as completely unknown subject to these constraints. This could be implemented as a modification of Codd's 'mark' concept (Codd, 1990) as suggested by Henley (2005). However, it does not help in any way to avoid the use of 3VL. Such 'partially missing' or incomplete data would require special versions of numerical operators such as > that would return either known (true or false) or unknown truth values depending on the numeric values in the query.

For example, in a drillhole database there might be an attribute "Top_Jurassic" for the "depth to the top of the Jurassic". If, in a given tuple, this depth is below the bottom of

the hole at 350m, the attribute would be recorded as ">350m." For this tuple, the following queries would produce different results:-

- WHERE Top_Jurassic > 300 would return TRUE unequivocally
- WHERE Top_Jurassic < 200 would return FALSE unequivocally
- WHERE Top_Jurassic > 400 would have to return UNKNOWN because the real value, if known, could be either in the range 350-400m, or greater than 400m.

Such semi-quantitative data are commonplace in geology and in other observational sciences, but are also more common than is generally thought in other fields. For example, the population of a city is unlikely ever to be known exactly (except in such cases as "Deadwood City, Colorado, pop.93"). Similarly, until the end of an accounting period an employee's expenses are unlikely to be known exactly if he'she has not submitted all claim forms - the value will be something like "≥£328" (being tht total of claims submitted to date, but with an unknown value of outstanding claims). There may of course also be an upper limit on this figure if the employer has an expense account policy which limits its employees' total expenses claims. So the value then might have to be recorded as "≥£328 and ≤£5000".

Nevertheless, however complex the bounding parameters, such incomplete data will entail only the same 'unknown' truth value as with a simple ('NULL' style) missing data placeholder - so only a 3VL is needed.

Another form of uncertain data is exemplified by laboratory analyses, where the reported value has a (usually implicit) associated uncertainty which can be represented numerically as a standard error. The standard error for geochemical data is the composition of sampling errors at different stages and the actual instrumental error. Commonly such errors have a gaussian distribution. Handling uncertain data of this sort in a database is problematic. For example, given an analysed value for Cu in a sample, recorded as 100ppm with a standard error of 10ppm, what is the correct truth value for the query

WHERE Cu > 105ppm ?

Clearly it is possible that the true value is >105ppm, but also possible (and rather more likely) that it is <105ppm - but since both possibilities exist perhaps the answer should be returned as UNKNOWN. However, the reported value is 100ppm, so the answer conventionally would be FALSE.

A better solution would be to return a probability estimate rather than TRUE, FALSE, or UNKNOWN. This uses similar numerical methods to the fuzzy databases, but the situation ought to be much better, because the error distribution can be determined objectively: there is a considerable body of theory to help estimate sampling error, and analytical instrument error is normally monitored routinely by laboratories. This may not be of course be the case for all uncertain data of this type - error distributions and their parameters may be unknown. This again is an area which requires much more work - and the database representation and manipulation are probably the easier parts of this question to resolve.