ELSEVIER

Computers & Geosciences 31 (2005) 780–785

COMPUTERS & GEOSCIENCES

www.elsevier.com/locate/cageo

The man who wasn't there: The problem of partially missing data

Stephen Henley*

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

Received 8 September 2004; received in revised form 13 January 2005; accepted 13 January 2005

Abstract

Existing commercial database management systems offer little or no functionality to handle the complexity of geoscience data—and other environmental science data—particularly in respect of missing and partially missing (incomplete or imprecise) data items. The emphasis of both the relational theorists (Codd, Date, and others) and the developers of database systems is on commercial applications where only rudimentary treatment of missing data is required, in the form of NULLs, and even these are not handled properly by the SQL language.

Keywords: Database; RDBMS; Missing data; Null; SQL; Logic; Relational; Fuzzy logic

Yesterday upon the stair,
I met a man who wasn't there
He wasn't there again today:
I wish that man would go away. – Children's nonsense rhyme

1. Introduction

Although one of the earliest relational database management systems (G-EXEC—Jeffery and Gill, 1976a–c) was developed in the 1970s to support applications in the geosciences, in recent years there has been progressively more reliance on general-purpose relational systems developed for 'business' users. This has the unfortunate consequence that little or no thought has been given to the complexities of managing real scientific data, and the resulting mismatch causes problems which have rarely been recognised despite the potentially severe consequences for the integrity of scientific databases.

2. Database management systems and data models

The closest that many geoscientists come (or want to come) to database management systems (DBMS) is the Microsoft Access that comes bundled with the Office suite, or a packaged ODBC-compliant system sitting underneath an applications software product. Yet effective management of their geological data is vital for all exploration and mining projects.

From the 1970s onwards, database management has been and remains an intensely fought-over battlefield. The original protagonists were hierarchical and network DBMSs following international CODASYL standards, and relational systems following (more or less) the principles first articulated by Codd (1970). During the 1980s the relational systems came to dominate the marketplace, largely by default as the older

*Tel.: +44 629 581454; fax: +44 1629 581471.
E-mail address: stephen.henley@btconnect.com.

0098-3004/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cageo.2005.01.010

COBOL-based hierarchical systems became obsolete with the mainframe computers which hosted them.

The most widely used database management systems, such as Oracle, Access, mySQL, SQLserver, Paradox, Ingres, and others, are all claimed to be relational. Certainly they all use SQL (Structured Query Language) which itself is often assumed to be an indicator of a relational database system. Unfortunately, SQL itself violates some of the relational principles, and fails to support others, so this is not a good criterion.

3. The missing data problem

Every geologist knows the problem: there is an incomplete data set—a gold assay has been missed out by the laboratory, or there is a strange-looking co-ordinate value, a stratigraphic interval cut out by an unconformity, or a rock-type description that has been forgotten. If the data must be stored and processed, something must be done to indicate that these values are missing (whether temporarily or for good). In the bad old (good old) days each application program would have its own way of dealing with missing data and its own requirement for coding it. Quite often the solution consisted of inserting a '-99' or some such value in place of the absent data item. There are two big problems with this type of solution: first, the difficulty of ensuring that the missing-data code could not be confused with a legitimate data value, and second, the certainty that different application programs would require different missing-data codes.

With the development of database management systems, the handling of missing-data codes became more systematic. For example, in G-EXEC (a relational data handling system for geoscience developed in the 1970s) each data file (relational table) contained a data description (sub-schema) in which a missing-data code was defined for each column. This missing-data code was carried with the data wherever it was copied, and was recognised and acted upon by all applications programs within G-EXEC. For new columns created by G-EXEC application programs, a missing-data code was set up that was very unlikely to be a valid data value—the highest negative real number which could be represented in the computer concerned. This was perfectly adequate as long as the data were not transferred to a different computer which allowed a different range of real numbers. A similar, though simplified, approach was adopted in developing Datamine (a relational database/applications system for the mining industry, developed in the 1980s—Henley and Stokes, 1983; Henley, 1992), in that -1.0×10^{30} was adopted as a universal numeric missing-data value, while a blank string was used for a character data null value.

In commercial database management systems, the question of nulls (shorthand for missing data values) became a central issue. In most early systems, and in many database systems to the present day, a hard-coded null-value solution was adopted—most commonly an empty character string. As Codd, the originator of the relational data model, pointed out, however, any column in any table is drawn from a 'domain' or extended data type which could be defined to include any ranges of values—including any special value which is chosen to be the 'null'—and so no 'null' representation can be assumed not to be identical with some real data value. Although his arguments were made in the context of the relational model, they indeed apply equally to any other type of database management system.

Conventional logic allows for just two truth values, *true* and *false*, leaving no room for uncertainty and no provision for missing information. It was recognised very early that this was inadequate for database management systems, and the 'null' concept was introduced. SQL provides a three-valued logic (3VL) solution with most logical operations involving nulls leading to a new logic value *unknown*. Unfortunately, as Date (1995, Chapter 9) demonstrates, standard SQL offers incomplete support even for this simple 3VL model and as a result can lead to serious database integrity problems. His preferred solution is to use only 2VL and a 'default value' approach: the database designer or application developer is responsible for defining one or more special values in each column, each with a set of operations that are allowed (and which must be coded by the designer or developer).

Codd (1990, pp. 203–204) argues convincingly that it is an abdication of the responsibility of the database management system to maintain integrity, if such a crucial role is left to the user or the application to define on a case-by-case basis. One solution to this problem, which he proposed, lies in attaching an extra one-byte column to each data column. This extra column would contain a flag or 'mark' for each missing data value in the column, and the data item in the column would simply be ignored whenever a mark was encountered. This is the method which was adopted in DB2 and some other IBM database management systems. It has the merit that it guarantees there cannot be any confusion between nulls and any legitimate values.

If nulls are used, then however they are coded, both Codd and Date assume that they imply a system of three-valued logic: when comparing values (for example in retrieval or table–join operations) there is either a match (True) or a mismatch (False)—or a 'Maybe' (truth value 'unknown') when one of the operands is a missing value. In his second version of the relational database model, Codd (1990) takes the argument one step further. He identifies two kinds of missing data. The ordinary 'missing but applicable' value—which might be supplied later (the missing gold assay)—is simply unknown, while a much stronger form of 'missing and