INTEGRATION OF MODAL LOGIC AND
THE FUNCTIONAL DATA MODEL

D.R. Sutton and P.J.H. King.
Dept. of Computer Science,
Birkbeck College,
University of London.
Malet Street,
London WC1E 7HX.

Abstract

This paper argues that the treatment of incomplete information in conventional
database systems is inadequate. It is not possible for users to formulate queries which clearly
state what it is they want to know and the answers to queries are often misleading. These
problems can be remedied by the use of modal logic which allows questions to be asked
about what might be true and what must be true, rather than simply about what is true.

This paper presents a functional database language which seamlessly
integrates propositional modal logic with the computationally complete functional data
model of FDL, thus allowing incomplete information to be handled more effectively than in
SQL or in more recent developments.

1. Introduction

The computationally complete functional data model of FDL [Poul90, Pou188]
is an important recent development. However in the area of the handling of incomplete
information it does not greatly improve upon relational database systems using SQL, the
shortcomings of which we discuss in section 3. Our conclusion - that these shortcomings are a
consequence of the use of truth functional logic - leads to a consideration of how proposi-
tional modal logic can be adapted to the functional database context.

The work we then describe could be viewed as a further development of
FDL. However in addition to the incorporation of modal logic we have made a number
of other, albeit relatively minor, changes to that language. We have therefore given
the language described here a distinct name, Fudal. Moreover the implementation of
Fudal is new and not a modification of the implementation of FDL.

In section 2 we survey the FDL development from which Fudal derives.
Section 3 discusses the problems of incomplete information we seek to address. Section 4
discusses propositional modal logic and its adaptation to the functional database context. In
section 5 we give a necessarily brief description of the syntax and semantics of Fudal, with
examples, only going into detail when considering those points where Fudal differs from
other functional languages and from FDL. Section 6 contains our conclusions.

2. The functional database language, FDL

FDL extends to computational completeness the functional data model, a
development from the binary relational model. In the functional data model information is
represented as entities and functions between them. Entities may be scalar meaning they have
a value that can be written down, e.g. as a number or a string, or else abstract meaning that
they do not. For instance a person's name is represented as a scalar entity but the person
himself is represented as an abstract entity. The distinction between scalar and abstract entities is the same as that between lexical and non-lexical entities in NIAM [Verh82].

From a programming language viewpoint FDL is functional, polymorphic, strongly typed and persistent, with a novel pattern matching algorithm, facilities for defaults and integrity constraints, and for transaction management. All functions are treated uniformly whether defined extensionally, intensionally, or of mixed definition. From a database viewpoint the function definitions and declarations are the database, which is updated by their modification.

The motivation for the development of FDL, descriptions of its syntax and semantics, and examples of its use are given by King and Poulavassilis in [Poul90, Poul88]. The implementation of FDL is over a virtual memory content addressed triple store which itself is implemented using a modified form of Nievergelt’s Grid File [Niev84] and conventional operating systems techniques for page and buffer management. A description of the implementations of FDL over this platform is given by Poulavassilis in [Poul92] and of the triple store itself in the PhD thesis of Derakhshan [Dera89]. An overview of the context of this work and its results is given by King et al in [King90].

A different development of FDL, termed PFL was described by Poulavassilis and Small in [Poul91]. This development abandons the functional data model and adopts a nested n-ary relational model whilst retaining the persistent functional programming approach. PFL does not, however, address the matter of incomplete information and suffers from the same defects as SQL and FDL as discussed in the next section.

3. Incomplete Information

Database systems such as SQL or FDL store imprecise information using null values. Different interpretations of null values are possible, but we restrict our attention here to the "as yet unknown" interpretation. Thus if a relational database contains the table:

<table>
<thead>
<tr>
<th>Staffdata</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Salary</td>
</tr>
<tr>
<td>John</td>
<td>15000</td>
</tr>
<tr>
<td>Michael</td>
<td>25000</td>
</tr>
<tr>
<td>Joanna</td>
<td>null</td>
</tr>
</tbody>
</table>

we interpret it as stating that John’s salary is 15000, Michael’s is 25000 and that, although Joanna has a salary, we do not know what it is.

3.1. Three valued logic

In SQL queries are interpreted using a three-valued logic in which the operators and, or, and not have the truth tables:

<table>
<thead>
<tr>
<th>p</th>
<th>not p</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>p</th>
<th>M</th>
<th>M</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>q</th>
<th>T</th>
<th>M</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>q</th>
<th>T</th>
<th>M</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>M</td>
</tr>
</tbody>
</table>

p and q

p or q