OBJECT ORIENTATION IN DATABASE INTEROPERATION

CASE STUDY OF VERSION CHANGED RELATIONAL DATABASES

M Dixon*, System Development Group, Faculty of Information Systems, Beckett Park, Leeds Metropolitan University, LEEDS, LS6 3QS

and

John Kalmus, Data Engineering Group, Systems Engineering Division, Rutherford Appleton Laboratory, Chilton, DIDCOT, OX11 0QX

* also Visitor to System Engineering Division, RAL.

ABSTRACT

This paper focuses on semantic issues that arise during non update querying of interoperating relational databases; the semantics are classified according to their location in a five layer information expressivity architecture. The system used as a case study is an operational engineering fault logging system which has been significantly modified to meet user requirements; however for tactical planning purposes information from the old and new systems need to be analysed together. This paper discusses the possible use of object oriented techniques and data models for the interoperation of relational database systems.

1 INTRODUCTION

1.1 Theoretical Framework

Coulomb and Orlowska [CouOrl93] have shown that the assumption of design autonomy for database federations was flawed; it founders on semantic heterogeneity. Additional semantic information generally needs to be added to the schema to achieve interoperaton. They have identified version control as a key issue when schema definitions are updated for what they classify as public reporting systems. They also assumed that such systems were likely to be based on the relational model in the short to medium term.

Jeffery [Jef&al94] proposed that the semantics of interoperating systems could be classified in a five layer information expressivity model which allows us to segregate database model features from business rule features at appropriate layers.

Saltor [Sal&al91] has established criteria for a canonical data model for the case of several databases interoperating in a federation [SheLar90]. It was shown that one based upon an object data model would be superior to one based upon the relational model both in terms of its expressiveness and of its semantic relativism. An object database standard has now been published by the Object Database Management Group [Cattell94].

1.2 Strategy for Study

The system we are using for the case study is a real airport maintenance system, based upon a relational database management system. The system has been substantially modified to meet user requirements.[DixKal94]

Following on from ideas at the 6th EDRGW we are using the 5 layer information expressivity model to classify the new semantics arising from:

1) addition of a new business rule in the later version or by the removal of a business rule present in the older version

2) a change in the business rule between versions

3) the addition of semantics to the reconcile schema definitions in order to obtain interoperation

4) the addition of semantics to a query to enrich the interoperation

5) a restructuring of the schema changing the atomicity of the attributes

It is clear that many of the systems that will be interoperated will be based upon the relational data model running on relational database management systems [HurBriPak94]. We are examining the appropriateness of using an object models for interoperating this type of system. The issue is whether an object data model can give a significantly better description of the semantics for its use to be invoked in a purely relational environment or will the relational model give good enough modelling.

1.3 Case Study

1.3.1 The case description

In the Appendices we give the entity, attributes, and attribute definitions for the old (Appendix 1) and new (Appendix 2) versions of the fault logging system as implemented. Conceptual models are given in Appendix 3. Further details of the case study may be found elsewhere [DixKal94, Framp86]. In a previous paper we selected two examples from the case for study in order to show how the 5 layer information expressivity model allows us to focus at the appropriate information level [DixKal95]. One example has no change to the schema but there is a substantial change to the business rules. For the other example the schema are radically different in the different versions but represent the same business rules.

1.3.2 Reasons for wanting interoperation

Here it is important to recall that the systems were developed and installed to meet the operational requirements of the maintenance crew; the lifetime of most faults is measured in hours rather than weeks. Analysing the information across different versions is seen as an additional requirement used to provide
a) historical continuity within a Terminal
b) patterns of reliability across the airport
c) tracking of plant transfers between Terminals
d) comparison of performance on maintenance effectiveness

1.4 OO for Relational Legacy System

Suppose we want two relational databases to interoperate at the non update query level then does the OO data model provide a better canonical data model than relational? Here we are considering the case of a relational database that has been upversioned and then we want to query both databases together. We will consider the introduction of SQL3 with object identity and abstract data types elsewhere since these are hardly ‘legacy’ issues at the moment. It is important to explore precisely how the 5 layer information model fits Sheth and Larson’s federation architecture model especially with respect to the canonical data model. Topologically the cdm schema sits directly on top of the local rdbms schema. Klas distinguishes between a syntactically uniform schema and the subsequent semantically enriched versions of that uniform schema [Klas94]. By considering relational schema as the starting point we are excluding user defined datatypes so it is likely that the benefits of using an object model may focus around behaviour. However by choosing to restrict our investigation to non update queries we are excluding many of the areas that would have benefited from OO.

From a practical point of view the relational tool set provides operations capable of accessing the necessary data on the combined database tables. We are not concerned here with simple changes to the format of data eg changing field size or modifying the type. We will look at the things which the OO data model is considered to do well and in which the relational model is deficient: 1) complex objects 2) aggregate attributes 3) functions for object behaviour 4) versioning 5) inheritance

1.5 Querying

In the environment for which this problem is being considered we do not think it realistic to assume that if an object cdm is used as an intermediate schema then the external schema would revert to the relational model. Rather we assume that if a switch to an object schema is made then that would be maintained for the external schema. However we are still concerned to make a distinction between the use of an object programming language such as C++ as a superior programming paradigm and the use of an object cdm and query language.

2 5 LAYER MODEL OF INFORMATION EXPRESSIVITY

2.1 Introduction

In this section we indicate the way that we partition our description of the different versions of the information systems following the 5 layer model of information expressivity. [Jef&al94]. It is important to note that we are not attempting to describe the original information systems in terms of the 5 layer model; we are trying to focus only on those features which are of interest to the application of queries relevant to interoperation of version changed systems.

2.2 Semantic Layer

In the semantic layer we define the business rules that we are considering. The terms will be defined in the way that the business sees them using a structured English.

2.3 Conceptual Layer

The conceptual modelling process is one of expressing the business in terms of one specific data model. Different data modelling approachs use different tool boxes and hence yield different descriptions. We propose to use SSADM in constructing our data model since this methodology is often a central government requirement on its contracts in the UK. The data model involves an Entity Relationship model, Entity Life History models, and Data Flow models in which the processes are described in structured mini specifications.

2.4 Intensional Layer

The intensional layer of the information system expresses the constraints that apply to the data values of the attributes. Within databases the constraints can be represented by triggers which are invoked by the database management system; typically these are embedded in application code attached to tables or forms which are used for data manipulation. These constraints may reflect the domain of the attribute or depend on the values of other attributes. The satisfaction of a constraint may lead to the resetting of attribute values. Access to certain data values may be restricted so that only designated users could modify those data values.

2.5 Logical Layer

The logical layer expresses the conceptual model in terms of the underlying data model of the database management system. This level is expressed in terms of the schema of the database. For the system we are considering this was the relational data model; we therefore assume the tables can be mapped to a union compatible form across the version change and a query applied using the UNION operator. It is important to recognize that the RM is different from the (E)ERM [Sal&al91] although some authors consider the latter to be a thin skin over the former.

2.6 Physical Layer

The physical layer describes the way that the model is implemented physically on the platform; for performance reasons this may differ significantly from the logical schema.

3 ADDITION/REMOVAL OF A BUSINESS RULE

One of the commonest reasons that database systems change is that there is a change in the business requirements. Usually this is based on the experience gained in running a verion of the database. As a consequence a new business need is identified as a business rule and the database schema and associated applications are modified.

In our system it was decided that it would be advantageous to record the stock levels and the re-order levels for replacement parts. For the new system this was added to the system; examination of the Future Work field/ Fault Description field / Fault Clearance Field on the fault logs could be used to identify work which was not completed because of a lack of parts. This could be used to identify which parts deliveries might have been problematic and need higher stocks holding to deal with erratic delivery lead times. The stock re order level is not likely to be available retrospectively from stores. Because we are looking for deviant behaviour related to individual parts it would not be possible reconstruct the history by a rough approximation. In this case the new rule can only be examined in the light of data in the new system and interoperation on this feature may not be feasible.

Different arguments apply to the introduction of a database trigger which issues a warning when a particular maintenance crew seeks to exceed 10% of its fault logs being commited to further work required. The constraint is naturally described as being represented in the intensional layer of the 5 layer model. In this case both old and new systems are compatible at the logical and physical schema levels so no specific additional semantics need to be introduced for interoperation.

We believe that further study should be given to this area using a wider range of case studies: 1) to investigate a range of constraints 2) to assess implications of inter and intra row constraints 3) to see whether removing a constraint from a system can lead to an increase in complexity leading to semantic differences between systems

4 A CHANGE IN THE BUSINESS RULE BETWEEN VERSIONS

Here we will consider the example of the change in meaning of plant repair time. This statistic is used as a measure of maintenance effectiveness; it indicates the time that the plant was not available for use. For the OFLS the Repair Time was calculated as the difference between the time the fault was logged and the time the fault was cleared. However improved performance against this statistic may not lead to improved business performance. There is a period each night when plant is not required because aircraft movements are not permitted; we call this a silent window. Paying night shift rates for staff to have plant ready for use during the silent window may make the statistics look better but is a poorer business strategy. In the NFLS the silent window for each plant group is recorded and deducted from the repair time. This example has been discussed extensively in a recent paper since it shows no change in the Repair Time attribute at the database schema layer for the considerable difference at the business rule (semantic) layer [DixKal95].

The calculation is straightforward but non trivial and is well suited to a programmed function such as an object method. This is why it is stored as a calculated field rather than simply calculated at with relational functions at retrieval time. Interoperation may require recalculation of the OFLS Repair Time using semantics introduced at the query level.

5 ADDITION OF SEMANTICS FOR SCHEMA RECONCILIATION

Here we will consider the example of plant renaming between versions. This is essentially a classic Coulomb and Orlowska case in which an additional table and additional data are introduced at the logical layer. This can be done in a completely relational way.

6 ADDITION OF SEMANTICS FOR QUERY ENRICHMENT

6.1 Example 1

We first discuss an example where the user provides contextual information to add semantics across versions so there is no change in the combined schema just in the semantics. We consider the case of a tactical planner who wishes to calculate how the cost of parts has varied over time. The later version of the database contains a parts inventory which includes current unit pricing. For most practical purposes the user could decide that this represents a current price approximation and therefore appropriate for the analysis of cost variation. If this were the only semantic difficulty associated with the version change then the relational data model would be adequate for the purpose; see however the section below on restructuring the schema for why this is not the case for our demonstrator system.

6.2 Example 2

A tactical planner may well wish to consider alternative possibilities to the ones represented by the states of the databases. Here we will consider the example of additional silent windows for repair time. In the scenarios for this example there are four different types of semantic changes being considered in addition to that required for schema reconciliation:
o Firstly we are altering the business rule for the definition of Repair Time.
o Secondly we are altering the schema to define the extra data required.
o Thirdly we are introducing new data associated with the silent window.
o Consequent on this the code for the calculation of the Repair Time must accomodate the new business rule.

For new data and semantics to be introduced in this way rather than through the specific modification of the base schema we assume that the modification can be expressed in a very few terms at the time of querying; eg the additional silent window is the same for all the plant subject to the query in that local database.

6.2.1 Scenario 1

In one scenario we are considering there being more than one silent window of the same type. This could arise when say it was recognised that a particular Terminal handling long haul flights had an inactive period during part of the afternoon. In doing this we are moving the calculation of Repair Time from a data capture to a data analysis operation for the revised semantics.

Taking an object approach, the object features relevant here would be
1) the possibility of using collection attributes to allow more than one silent window; this would appear to be of minor importance compared to the relational alternative.
2) object operations in the cdm which returned a revised value for the Repair Time

depending on the context. This would be an extremely convenient feature as it locates the required functionality in one overloaded operation.
3) versioning so that the additional semantics introduced at this point did not compromise the integrity of the cdm. However this feature does not seem to be a standard feature of C++ or the object profile of ODMG.

6.2.2 Scenario 2

In another scenario we are considering there being more than one type of window; the second type of window being introduced for the lag between ordering and delivery of parts by a supplier. Here it is reasonable to assume that there would need to be specific capture of the order lead times for the replacement parts. Under these circumstances the considerations that are relevant are more appropriate to a change in business rule than to added semantics on querying enrichment.

7 RESTRUCTURING OF SCHEMA CHANGING ATOMICITY

Here we will consider example of Parts Used being seperated out from a free text window. This changing of atomicity of the attribute effectively means that the attribute becomes a structure type when additional rules for interpretation are added. We have already shown that although this leads to a significant restructuring of the schema the changes are located down at the logical layer using the terminology of the 5 layer information expressivity model [DixKal95].

There are three features that we need to consider here.
1) Complex object: The old version, OFLS, has an attribute which has a complex structure which would become Part_Number and Part_Quantity in the new version, NFLS. This is most naturally represented by a 2 dimensional array. It would appear that ODMG support single dimensional arrays with access by integer specifying position while Ontos/C++ supports a dictionary in which Part_Number could be used to access Part_Quantity; this would not be suitable for more attributes. The alternative is to define a structured object type consisting of Part_Number and Part_Quantity with each instance referencing the relevant fault log instance. However this is directly analogous to the way NFLS is implemented; there appears to be the implication that for this factor the relational model is adequate.
2) Behaviour: In Sheth and Larson’s 5 level architecture there is a transformation operator relating the local and cdm schema. The mapping rules of the complex attribute have been described elsewhere [DixKal95]. Here we are concerned with whether the replacement of transformation application code operators and a target relational schema by an object in which the transformation is part of the instance creator function is a significant improvement. For clean data there would appear to be little difference; however the OFLS attribute is likely to be quite dirty in terms of non standardised naming, absence of quantities, non standard or missing terminators, extraneous comments. A nice feature would be that querying could be defined on version changed objects with the same interface; the transformation rules being explicitly linked to the version object type.
3) Aggregate: - see 1) above

8 CONCLUSIONS

1. Semantics added in the query means that we need to allow for dynamic extension to the schema from the query language.

2. 5 layer information expressivity classification allows us to distinguish between different types of semantics introduced during interoperation:

- schema reconciliation semantics
- query reconciliation semantics

3. The application of new business rules which are constraints can be located in the intensional layer alone. However more work needs to be done to see whether this is truly representative of a wider set of constraints.

4. The main strength of object models for non update querying on relational systems appeared to be in the close linking of methods to calculated attributes. C++ is a superior form of C for programming although there is a high learning barrier. Unfortunately the advantages of versioning do not appear standard. In our case study the examples selected did not indicate inheritance would be an advantage. We wish to explore further the introduction of querying semantics.

REFERENCES

[Cattell94] R G G Cattell (Editor), The Object Database Standard: ODMG-93 Release 1.1, pub Morgan Kaufmann, ISBN 1-55860-302-6
[CouOrl93] R M Coulomb & M Orlowska, Interoperability in Information Systems, Technical Report 263, Department of Computer Science, University of Queensland, 1993.
[DixKal94] M Dixon & J Kalmus, Semantic Heterogeneity in Interoperating Databases. An Invetsigation based upon Engineering Maintenance. Proceedings of the EDRGW6 on Deductive and Interoperable Databases (Barcelona), 1994.
[DixKal95] M Dixon & J Kalmus, A 5 Layer Information Expressivity Model Applied to Semantic Heterogeneity in a Decentralised Organisation. Paper in preparation.
[Framp86] A Frampton Planned Maintenance System, Report BAAPMS.Z1, DM England & Partners, 1986
[HurBriPak94] A R Hurson, M W Bright, and S Pakzad, Multidatabase Systems: An Advanced Solution to Global Information Sharing, pub IEEE Computer Society Press, ISBN 0-8186-4422-2.
[Jef&al94] K G Jeffery, L Hutchinson, J Kalmus, M Wilson, W Behrendt, C Macnee, A Model for Heterogeneous Distributed Database Systems, BNCOD12, Springer Verlag Lecture Notes in Computer Science 826, p221, 1994.
[Klas94] W Klas Interoperable Databases. Proceedings of the EDRGW6 on Deductive and Interoperable Databases (Barcelona), 1994.
[Sal&al91] F Saltor, M Castellanos, M Garcia-Solaco Suitability of Data Models as Canonical Models for Federated Databases, SIGMOD Record 20(4), p44, 1991.
[SheLar90] A P Sheth & J A Larson, Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases, ACM Computing Surveys 22(3), 183, 1990.

ACRONYMS

BNCOD British National Conference on Databases
cdm canonical data model
ERCIM European Research Consortium for Informatics and Mathematics
EDRGW ERCIM Database Research Group Workshop
(E)ERM (Extended) Entity Relationship Model
NFLS New Fault Logging System
ODMG Object Database Management Group
OFLS Old Fault Logging System
OO Object Oriented
rdbms Relational Database Management System
RM Relational Model

APPENDIX 1

Definition of Old Fault Log Entity

Attribute	**Definition**
Fault Number	System issued unique number
Report Time	Date/Time when fault logged; autostamped
Plant Name	Unique name for plant item
Plant Group	Category of Plant
Location	Place where plant is located in Terminal
Area	Grid value
Cost Code	Head of charge for work
Fault Title	Summary of Description of Fault
Fault Descrip	Full description of fault
Crew	Maintenance Crew responsible for work
Report by	Person issuing initial fault report
Clearence	Action and comments taken to clear fault
Future Work	Engineers view of further work needed on it
Complete Time	Date/Time engineers report fault cleared
Repair Time	Time out of service
PartsUsed	A list of parts used in repair

APPENDIX 2

Defintion of Plant Entity

Attribute	**Definition**
Plant Number	Unique Nato code for plant
PlantDescript	Working name for plant; unique
Plant Group	Type of plant
Location	Place where plant is in terminal
Area	Grid reference
Cost code	Head of charge for work
DateInstalled	When plant was installed at airport
Manufacturer	Name/address of plant manufacturer
BeginNotNeed	Begin time when plant would not be needed
EndNotInNeed	Time when plant needed again

Definition of New Fault Log Entity

Attribute	**Definition**
Fault Number	System issued unique number within year
Fault Year	Code for year; unique fault year / number
Report Time	Date/Time when fault logged; autostamped
Plant Numer	Unique NATO code; foreign key
Fault Title	Summary of Description of Fault
Fault Descrip	Full description of fault
Crew	Maintenance Crew responsible for work
Report by	Person issuing initial fault report
Clearence	Action and comments taken to clear fault
Future Work	Engineers view of further work needed on it
Complete Time	Date/Time engineers report fault cleared
Repair Time	Time out of service
PartsUsed	A list of parts used in repair

Definition of Parts in Store Entity

Attribute	**Definition**
PartNumber	Nato code; unique
PartDescrip	Common name for part; non unique
StoreLocation	Which stores part is held
StockLeve l	Count of parts in stock
ReorderLevel	Count of parts when should reorder

Definition of Parts Used on Fault Entity

Attribute	**Definition**
PartNumber	Nato code; unique
Fault Number	Fault on which part was used; foreign key
Part Quantity	Count of this part used on this fault
WorkOrderNum	Identifies work order part used on; unique

APPENDIX 3

Figure 1: Entity Relationship Diagram of Conceptual Model for Repair Time allowing for the possibility of more than one out of service window per plant item

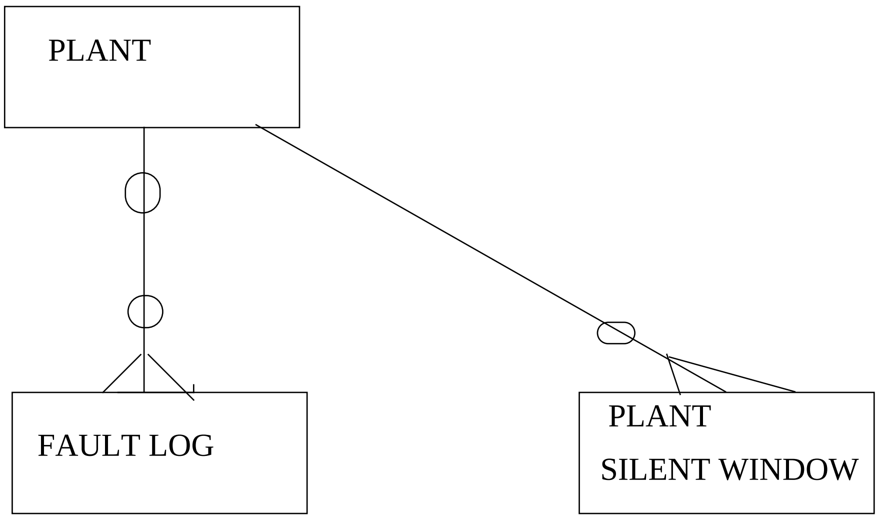
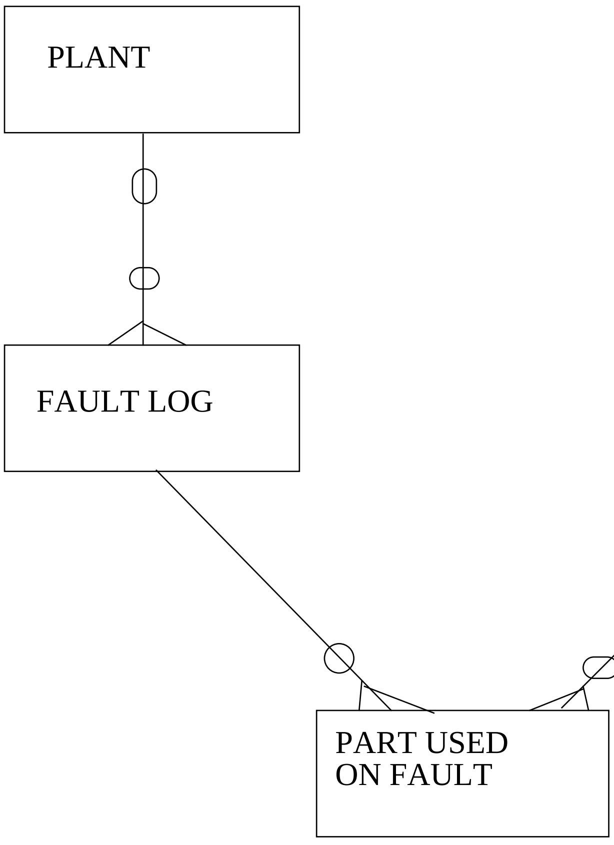


Figure 2: Entity Relationship Diagram of Conceptual Model for Parts Used in a fault repair.


PLANT
FAULT LOG
PART IN STORE
PART USED
ON FAULT