1 Introduction

The incumbent <name> project uses the classic OO/ORM monolithic model. The absence of an Open Architecture, with a Relational database, has been identified as the main error in the project, and its replacement is planned. However, the Team Leader of the project has proposed that the project should instead be further funded. The central issue in the proposal is that the OO/ORM model is sound and has theoretical support, and two theoretical papers that validate the model, are referenced. This article is the formal response requested by the bank. It addresses the issues in the proposal, and deals with <one of> the theoretical papers in detail.

1.1 Subject Paper

This section identifies and introduces the subject theoretical paper. The full paper is available on the shared drive <directory>.

The paper is typical of the many such papers that are used to support the OO/ORM monolithic model. While this response treats one typical paper in detail, the theme and finding of the response applies to all papers that support the OO/ORM model.

- The subject author is a lecturer in database theory. As is typical of such, he has little understanding of the *Relational Model* (the "relational" they discuss is a small fraction of the *RM*, plus non-relational additions that suit their purpose), or of implementation imperatives.
- He is also well-known for a number of papers in the object modelling space. The Team Leader suggests there are four that he uses as reference.
- He is a proponent of *Foundations of Databases* (commonly called the "Alice Book" by Abitebul, Hull & Vianu), which has previously been reviewed, as presenting nothing of the sort, as well as installing the a number of fictions re the *Relational Model* and OO in the readers mind. Thus the *Relational Model* is diminished, and the OO componentry is artificially elevated.
- Note the **References** (next page):
  - The paper is clearly based on, and supports, the OO/ORM monolithic model. The Team Leader's positioning of the paper is not disputed.
  - Some of the authors cited are mentors of the subject author (keeping my review of that book in mind).

The paper makes five proposals (right), with ③ being central, thus the main point we address. For our purposes, ④ and ⑤ are a subset of ③, and also solved.

Identity

Proceedings of the Fifth International Workshop on **Database Programming Languages**, Italy, 1995

Union-Types in Object-Oriented Schemas

Jan Hidders, Dept of Math & Computer Science, Eindhoven University of Technology, Eindhoven, NL

Abstract

In this paper we investigate union-types in object oriented IQL-like schemas. These types can be used to model null values, variant types and generalization classes. They make, however, deciding equivalence and subtyping more difficult. We will show that the complexity of these two problems is co-NP-complete and present complete sets of rules for deciding both problems. The combination of union-types and multiple inheritance makes it also harder to detect typing-conflicts in a schema. We will give an algorithm for deciding this and discuss its complexity. Furthermore, we will present an algorithm for detecting schemas that define types with a bounded number of values. Finally, an algorithm will be presented that verifies whether in a schema the type of a subclass specifies options that are forbidden by its superclasses.

Introduction

The introduction of union-types in object-oriented schemas makes them more expressive. Usually, however, they are limited to disjoint unions [9] or labeled unions such as variant records [7]. Because of this limitation the reasoning about these types remains simple [6, 13]. We argue that *general* union-types such as used in [10, 2, 5] are a useful extension and even arise naturally in data models without union-types.

We can use them, for instance, to model optional integer fields by specifying their type as $(\texttt{null} \vee \texttt{int})$ where `null` is a special basic type with only one value viz. *null*. Moreover, we could define several kinds of nulls and let the type be $(\texttt{null}_{unkn} \vee \texttt{null}_{undef} \vee \texttt{int})$. If an object has two fields that are both optional we might want to specify that one of them has to be defined but not both. This can be done by giving the object's class the tuple-type $([a : \texttt{int}, b : \texttt{null}] \vee [a : \texttt{null}, b : \texttt{str}])$[1]. This is an example of how in general it is possible to represent *variant records*.

① **Null as Type**

② **One of Two Required**

In data models that do not have union-types such as in [1] they arise naturally in the context of multiple inheritance. They can, for instance, be used to denote generalization classes without explicitly adding them to the schema. This can be demonstrated by the schema depicted in Figure 1. Here we see the two classes `Angler` and `Sea−Fisherman` both with the field *catch* which is, respectively, a set of `Fish` and a set of `Sea−Animals`. Since `Sea−Angler` is a subclass of `Angler` and `Sea−Fisherman` it inherits the field *catch* that must be both a set of `Fish` and a set of `Sea−Animals`. The type of the *catch* of a `Sea−Angler` is a set of $(\texttt{Mackerel} \vee \texttt{Tuna})$, or with intersection-types as a set of $(\texttt{Fish} \wedge \texttt{Sea−Animal})$ which is actually the same type. Without union-types or intersection-types this type cannot be denoted unless an extra class `Sea−Fish` is added that is the generalization of `Mackerel` and `Tuna`, and a subclass of `Fish` and `Sea−Animal`. Although this class might seem quite natural here, this approach can lead to the addition of more unnatural generalization classes.

③ **Sea-Angler**
unions or intersects required

Although union-types give us more expressiveness they also make the reasoning about types more difficult. Different type expressions can now denote the same type. For instance, the type $([a : \texttt{int}] \wedge ([b : \{\texttt{bool}\}] \vee [b : \{\texttt{str}\}]))$ is equivalent with $[a : \texttt{int}, b : \{(\texttt{bool} \vee \texttt{int})\}]$ and in the schema of Figure 1 it holds that $(\texttt{Mackerel} \vee \texttt{Fish})$ is equal to `Fish`. The same problem occurs with the rules for subtyping. Although the rules for types without union-types remain

④ **Type Loss**
⑤ **Mackerel | Fish = Fish**

Extract

- This document is not an academic review. It is a response from the implementation universe, where the subject paper has already been used, to support implementation decisions.
- Due to customer confidentiality, the full response (which addresses a second theoretical paper, and includes project details) cannot be provided. This Extract is the portion of the response that the customer has kindly permitted for release into the public domain. It is released on the basis that the subject paper is public domain, thus the findings and the damage it causes, should be as well.

Expansion

Although this document started out as an Extract (23 Jan 15), it is being used as a reference, regarding the nature of this most common problem, and its solution. That includes an audience who are not necessarily tertiary educated. Hence the detail has been expanded since the Extract was created (06 Jun 15).

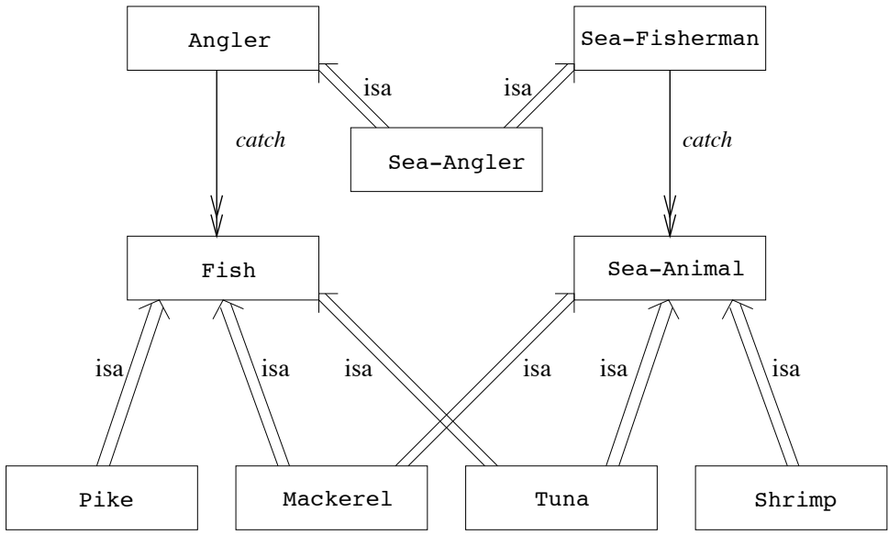


Figure 1: A schema with multiple inheritance

[1] It might seem more obvious to use the type $([a : \texttt{int}] \vee [b : \texttt{str}])$ but due to the subtyping semantics the two fields would not be mutually exclusive.

Reference

[1] S. Abiteboul and R. Hull. IFO: A formal semantic database model.[1] *ACM Transactions on Database Systems*, 12(4):525–565, December 1987.

[2] S. Abiteboul and P.C. Kanellakis. Object identity as a query language primitive. In J. Clifford, B. Lindsay, and D. Maier, editors, *Proc. of the 1989 ACM SIGMOD Int'l Conf. on Management of Data*, number 18:2 in SIGMOD Record, pages 159–173. ACM Press, 1989.

[3] F. Barbanera, M. Dezani-Ciancaglini, and U. de'Liguoro. Intersection and union types: Syntax and semantics. *Information and Computation*, 119(2):202–230, 1995.

[4] K. Brathwaite. *Object-Oriented Database Design: Concepts and Application*. Academic Press, Inc., 1993.

[5] D. Calvanese and M. Lenzerini. Making object-oriented schemas more expressive. In *Proc. of the 13th ACM Symp. on Principles of Database Systems*, pages 243–254, 1994.

[6] L. Cardelli. A semantics of multiple inheritance. *Information and Computation*, 76:138–164, 1988.

[7] V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl. From structured documents to novel query facilities. In *Proc. of the 1994 ACM SIGMOD Int'l Conf. on Management of Data*, pages 313–324, 1994.

[8] M.R. Garey and D.S. Johnson. *Computers and Intractability – A guide to NP-completeness*. W.H. Freeman and Company, San Francisco, 1979.

[9] G.M. Kuper and M.Y. Vardi. The logical data model. *ACM Transactions on Database Systems*, 18(3):379–413, September 1993.

[10] C. Lecluse and P. Richard. Modeling complex structures in object-oriented databases. In *Proc. of the 8th ACM Symp. on Principles of Database Systems*, pages 362–369, 1989.

[11] T. W. Ling and P. K. Teo. Inheritance conflicts in object-oriented systems. In *Proc. of the 4th Int'l Conf. on Database and Expert Systems Applications*, number 720 in Lecture Notes in Computer Science, pages 189–200. Springer-Verlag, 1993.

[12] B.C. Pierce. A decision procedure for the subtype relation on intersection types with bounded variables. Technical Report CMU-CS-89-169, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213-3890, August 1989. from: `http://www.cl.cam.ac.uk/users/bcp1000/ftp/index.html`.

[13] D. Remy. Typechecking records and variants in a natural extension of ML. In *Proc. of ACM symp. on Principles of Programming*, pages 242–249, 1989.

[14] K.D. Schewe, B. Thalheim, and I. Wetzel. Foundations of object-oriented database concepts. Technical report, University of Hamburg, 1992.

1 Yet another "semantic" semantic model, in denial of the then extant and established semantic models and Standards.

2 Issue

2.1 Error of Principle

Although this identifies the main errors, with both the paper, and the OO/ORM monolithic model that is employed in the current project, it is by no means exhaustive. Collectively, it is the mindset of a narrow focus on the OO space, and ignorance of the long-established requirements for Relational databases.

1 The subject author makes the classic, and now infamous, error that both OO/ORM theoreticians and practitioners make, that of perceiving Data through the lens of OO [1]. The database is treated merely as a storage location, to afford persistence for the singly-important objects. Thus the entire exercise of analysing and modelling the data, and doing so within the Relational context, is eliminated, and none of the benefits of that exercise is realised.

- Conversely, all the constraints and controls on the data, which should be deployed in the database, are deployed in the object layers. Not surprisingly, it does not work, due to the myriad problems that ensue from invalid deployment.

2 The principle of separating **Data vs Process**, and using the separately applicable methods for analysis, design, and implementation, is a long-established principle in all applied sciences, no less in the IT sphere. This has been ignored [2].

2.1 Here, the subject author has determined issues (problems ③ ④ ⑤) that are directly related to the integrity of data, but he has failed to identify the cause.

- Failure to recognise the problem for what it is, and to determine the cause
- Failure to address the problem in the causative location (the database),
- Instead he addresses it in the symptomatic location (the OO monolith, the object classifiers)
- He proposes yet more complexity, in the layers that are already complex, and that fail.

2.2 Note that the subject author's mentor and cited reference [1], S Abiteboul, has introduced a 'semantic model' in 1994 that combines *Data* and *Process*. (Of course, by definition, all models are semantic.) This in ignorance, or subversion, of standards:

- IDEF1X, the standard for [semantically] modelling Relational databases (the *Data*) 1985
- SSADM or IDEF0, the established standards for [semantically] modelling *Processes* and Systems 1980

2.3 This 'semantic model' is the basis for the OO/ORM Monolithic model, where object notation is introduced into an otherwise data-only model. It is primitive, and has only rudiments of the capability of IDEF1X, which existed for a decade before it was invented.

2.4 While the OO/ORM theoreticians and implementers use this 'semantic model', generally the more capable ones express it using UML [3].

3 Like many researchers these days, the subject author is totally ignorant re (i) Data Modelling, and (ii) Relational Databases, their purpose and their capabilities, etc. Thus, he constructs an apparatus within the OO arena, in an attempt to supply some fragment of Database and Relational capability. The notion of novelty is intact only within this arena, the problem was solved in the Relational Database arena in the 1970's.

- Due to [2], the subject author is unaware of the fact that the attempt is weak, misplaced, and that it will never be a complete solution.

4 Note that the OO/ORM monolith is not an architecture, it is a non-architecture. Something that its a single vertical stack of several dozen layers, with no differentiation between components (eg. vis-a-vis *Data* vs *Process*) cannot be construed to be an architecture. It is an absence of architecture.

- Further, in the context of **Open Architecture** databases, it is closed. The database can only be accessed via the application [4].

For the above, the following references are given. These papers are essential reading for all professionals in the IT space, theoreticians as well as practitioners:

- ***Unskilled and Unaware of It**: How Difficulties in Recognising One's Own Incompetence Lead to Inflated Self-Assessments* 1999
  Justin Kruger, David Dunning
- ***Why the Unskilled are Unaware:** Further Explorations of (Absent) Self-insight among the Incompetent* 2008
  Joyce Erhlinger, Kerri Johnson, Matthew Banner, David Dunning, Justin Kruger
- Abraham Maslow's ***Law of the Instrument***, commonly known as *Maslow's Hammer* 1966.
  The danger is, to view all problems from the perspective of one's profession, "if all you have is a hammer, everything looks like a nail".

In consideration of the above Errors of Principle, the premise of the paper is invalid, without any scientific basis. Thus the propositions therein are all invalid.

That being established, auditors and managers need read no further. The rest of this response provides (a) detail to support the finding, and (b) the correct implementation method, including both data and process models.

---

1 Which lens is very small, single-purpose. The result, myopia. The construction, a monolith. It is a **joke**, even among the OO developers.

2 While neither defending the Team Leader's use of the OO/ORM monolithic model, nor his failure to separate *Data* vs *Process*, it must be noted that he is following the theoreticians in the OO/ORM space, and the theoreticians who serve that space are bankrupt. Despite the numerous academic papers that are used to prop up the model, the space is devoid of theory that passes the litmus test. An examination of any such paper reveals its bankruptcy, its denial of related science. It is the same in the Relational database space, there has been no progress in the theory since the great Dr E F Codd.

3 UML is neither a standard nor a modelling facility. It has no capacity for decomposition, and makes no distinction between *Data* and *Process*. At best, it is a documentation method, one that has one single symbol, and a morass of notation that most modellers do not understand. The result, as evidenced in the project, is a free-for-all diagram, labelled "UML", one that is accepted without understanding by the very people who need to understand it.

4 Although it must be mentioned, the architectural issues that cause the problems in the incumbent project are not addressed here (refer to the Technical Audit).

ø A word about the *interpretation* of Null.

2.2 Denial of the Hierarchy

In the post-Codd era, there has been a propagation and marketing of books which allege to champion and further the *Relational Model*. However, such books expose only a fraction of the *RM*, and champion the authors' own ideas, which as evidenced, is limited to pre-1970 Record Filing Systems. Such books, and such authors, are in fact anti-relational.

A consequence of this is a pervasive denial of Hierarchies, they are unaware (or worse):

- that the *Relational Model* is founded on the Hierarchical Model
- that hierarchies exist in the data, and that they are specifically supported in the *RM*
- that the Keys that result from the **Relational Normalisation** given by Codd, are hierarchical

An implementation of a database without these hierarchies, robs it of the integrity, power, and speed that is available in *RM*-compliant databases. Here, the Integrity is at issue.

In addition to [*2.1 Error of Principle*], the following causative errors are noted.

1 The author of the subject paper is a typical lecturer, a student of these post-Codd authors. As such, he demonstrates the typical blindness re hierarchies. This blindness leads to a failure to recognise that an hierarchy exists naturally in the data, and to model and implement it as such.

2 The second item is also typical of these students, and of the OO/ORM implementers. Instead of focussing on analysing and *classifying* the data, the author focuses on data *content,* on *instances* of the data.

3 Further, the data is evaluated *without meaning*, the meaning having been lost due to [*2.1 Error of Principle.1*] and the consequential absence of formal Data Analysis.
   - The subject author focuses on the *catch*, the target, without understanding the data; where the *catch* came from, its provenance.

4 Stated otherwise, he makes the mistake, typical of theoreticians, of failing to distinguish base vs derived relations. *Catch* is a derived relation, but he focuses on it as if it were a base relation.

4 Further, while noting that the model is already complex, and blind to the fact that it has failed, universally, he addresses it by *adding more* complexity.

2.2.1 Datagram

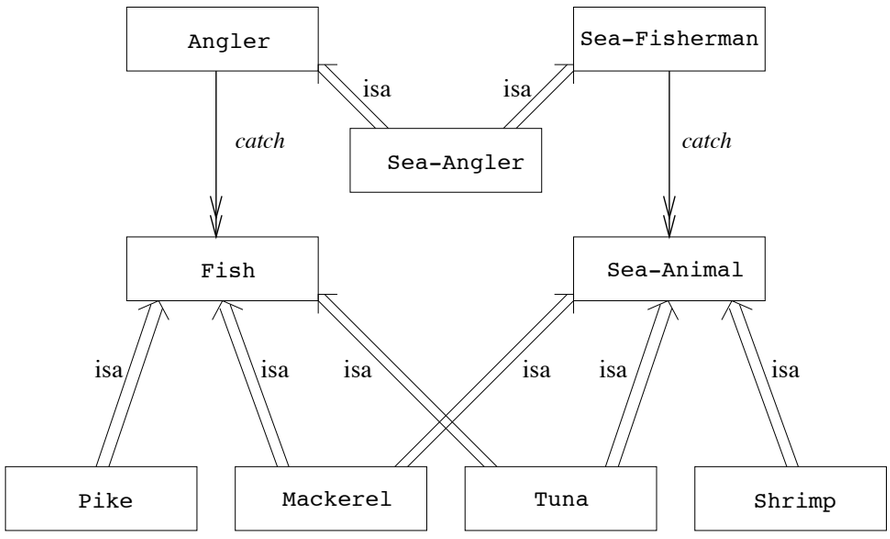


Figure 1: A schema with multiple inheritance

a Even though the diagram clearly *depicts* an Hierarchy, and the inheritance problem mentioned by the subject author *relies upon* an Hierarchy, the evidence is, he is in a state of denial regarding the Hierarchies that exist naturally in the data [1].

b Scientific work cannot be executed while one is in a state of denial of reality, or of related science. Such a state has a crippling effect on the tasks being performed.

c One example of the crippling effect of denial: it has the result that his classifications (refer text and diagram) are each limited to a single level. Thus the exercise of Classifying, and Identifying, Data, is severely hampered, the result is a primitive classification.

d A second example of the crippling effect: while he does mention the relevance of preventing "types that are forbidden in their superclasses", which intimates an intuition that Hierarchies do exist in the data, he fails completely in:
   i determining that those Hierarchies exist
   ii Identifying and Classifying them, and
   iii arranging them in some Order
   iv such that the content of higher-order types constrain the content of lower-order types.

1 Noting that the subject author has written another paper titled *How to recognise different kinds of tree patterns from quite a long way away*, this denial or blindness of hierarchies (tree patterns, visually, as depicted) is particularly ironic.

3 Solution

This section provides the details for the solution to the declared problem. This is not a basetype:subtype problem; it is not a class:subclass issue; and it is not a union-type issue, all of which are allopathic approaches, treating the problem where it appears, rather than determining and treating the cause. Even the multiple inheritance mentioned, is incidental, it is not a part of the *actual* problem.

3.1 Issue Address

Referring to the errors identified in [*2 Issue*], the solution consists of:

1 Observing the arhcitectural principle of separation of *Data* vs *Process*. Once they have been separated, it is easy to recognise that the problem lies in the data, its integrity (or lack of it), and to fix the problem where it exists. This is a reversal of *Maslow's Hammer*. Which consequently eliminates the use of the OO/ORM monolithic model, and opens the avenue to treat data, as data.

2 Application of the *Relational Model*, and Modelling the data. The analysis is detailed in section [*4 Data Analysis*], with [*4.B*] demonstrating the analysis and modelling together, in a single exercise. Here, the issue of data integrity is addressed in the correct location, the database, where all controls on data are required to be deployed, by virtue of standards.

- Indeed, the simple act of placing the data in the Relational context causes the declared problem to disappear [1].
- The entire issue, including the consequence in the object classifiers, of Typing, and of Union Types is eliminated [2].
- Relational-isation exposes, and eliminates, superfluous theory [1].

3 Observing the Hierarchies that exist in the data, and implementing them squarely, without impediment.

4 Regarding (a) the OO/ORM monolithic model, and (b) the notion of controlling *Data* in the *Process* space, in the subject paper (as well as the hundreds of similar papers supporting the notion, and specifically the four papers referenced by the Team Leader); the current project; and all implementations that employ that model, it must be said that:

- Any and all attempts to fix such problems where they do *not* exist, will fail, as enumerated in the current project.
- To the extent that they appear to work, they are (i) transient, and (ii) incomplete. In time, the temprorarily succesful attempt is exposed as a failure, and the control of data integrity (in the incorrect location) is exposed as grossly incomplete and fractional.

5 The specific problems referenced in the subject paper that apply to the project (ie. the proposal to increase funding and increase complexity, rather than to replace it) are solved.

③ Sea-Angler	Unions or intersects **not required**
④ Type Loss	**Prevented**
⑤ Mackerel | Fish = Fish	**Prevented**

They fall into two categories (although not determined as such in the subject paper): Data Integrity and Typing.

- Data Integrity errors are entirely eliminated by the solution given here. Erroneous data will simply not appear in any application object.
- Typing errors are prevented. All proposed complexity is eliminated. Classifiers and Determination of Types remain simple, thus less prone to error.

6 Finally, the subject paper illustrates *one* form of Data Integrity problem, *one* form of Typing error. Note that the architectural solution provided here eliminates *all* forms of Data Integrity errors, and prevents the great majority of Typing errors [3].

3.2 Architecture

The solution involves a classic implementation of the **Open Architecture** model:

1 A single, completely independent Relational database, providing maximal Data Independence, Integrity, and Transactions. Refer [*5 Data Model*].

- The data is fully **Normalised** (3NF as Codd intended, to the fullness of the *RM*, far more than "5NF").
- All data values are **constrained** by Domain and Key, as Codd intended it (as opposed to the fragmented definition in "domain key normal form", which the theoreticians state as being impossible)
- Modelled and documented according to the IDEF1X Standard.
- All writes to the database are via ACID Transactions only [4]. This is the **Database API**.
- All reads are directly from the tables, or from Views constructed for a purpose, ie. a single `SELECT` command.
- Further, the data persists beyond the life of any application, and it is extended independently (independent of the lot, in concert with one).

2 One or more applications of any type, interacting directly with the database. Refer [*6 Application*].

The rest of this response provides examples of relevant portions of the architecture, for discussion purposes. A Relational database and a classic OO application is treated. The first activity is [*4 Data Analysis*].

---

1 Although many theoretical papers these days are based on the erection of a *Straw Man* problem, and subsequently propose a solution to burn it down, for some nefarious academic purpose, it is unlikely that that is the case with the subject author. My considered opinion, after reading five of his papers, and interacting with him in open fora, is that he is simply ignorant of the relevant issues, that he is a product of education system in which each researcher operates in staggering isolation of reality, of related science.

2 That does not mean the paper has no value. Outside the Relational context, such as the OO/ORM monolith arena, and the "relational" theoretician circus, Record Fling Systems with no Keys are the norm. The paper may well have value there.

3 On the application side, in poor quality applications, where standards are not observed, it remains possible that Typing errors (not Data Integrity errors) can nevertheless be created.

4 Direct writes to the tables, or via Views, is prevented. Education re OLTP Standards and ACID Transactions for the developers is included in the project.

4 Data Analysis

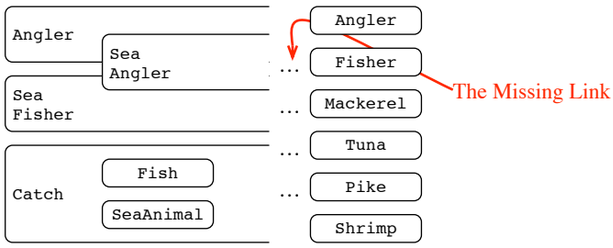
Although the analysis of the data in this case is exceedingly simple, it is given if only for the purpose of being complete, of avoiding the production of a second document in the event such detail was requested.

- the architectural principle that *Data* and *Process* must be separated, and treated separately, is observed.
- given that in 1995 (the date of the subject paper), the *Relational Model* was firmly established as The Method for understanding and modelling data, that context is employed.

Two alternate methods are given, the first for those who are unfamiliar with the process of Analysis, who perceive the data as a featureless mass, the second for those with some experience with the process.

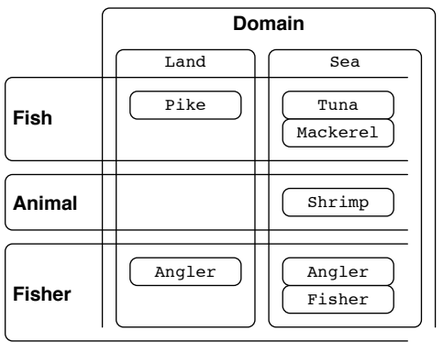
4.A Mass Approach

1 Amass all the given Data, as is. Tabulate it, in order to expose its features, to classify it. The problem shows it face, even at this early stage

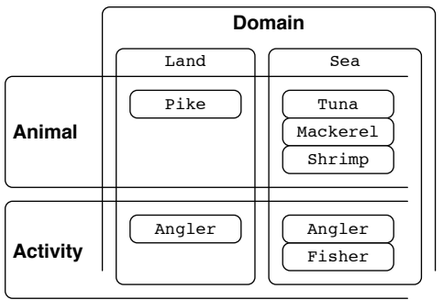


2 Catch is a result, or derived data, not base data, not relevant at this stage.

3 The obvious fact is, both the Actors and the possible Catches operate in, and are limited to, a Domain. That needs to be formally Identified, and the data re-classified. (The Classifications are open rectangles; the data items being classified are closed.)



4 Next, we formalise the Classifications, and name them (Fish is a form of Animal; Fisher and Angler are forms of Activity). There are now two separate classifications per Animal and Activity, across one of Domain.



5 Next, we determine the **Identifiers** for the Classifications. That is a simple matter because we already know that Land- and Sea-Fisher needs to be discriminated, and so does Land- and Sea-Animal. Its discriminator (or Type in OO terms) is:

```
Domain   ( Domain )
```

6 Now for the remaining Classifications:

```
Activity ( Domain, Activity )
Animal   ( Domain, Animal )
```

7 The natural Hierarchy is both obvious and visible. Notice that Activity and Animal are each dependent on, and exist within, the Domain; they are not Independent.

8 And now for the result, a derived relation, Catch, which is dependent on both the Activity and the Animal. The Identifier is, again, obvious:

```
Catch ( Domain, Activity, Animal )
```

In case it needs to be stated, `Catch.Domain` *is* ***both*** *the* `Domain` *of* `Activity`, ***and*** *the* `Domain` *of* `Animal` [1]. *The third purpose of* `Catch.Domain` *is to form the Identifier, or Relational Key, which as usual, is Hierarchical.*

1 In Relational Databases, each column serves more than one purpose, that is the basis of Identifiers, of Relational reduction.

4 Data Analysis

4.B Set Approach

1 Stop focussing on the data content, determine the type of data, construct the classifications.

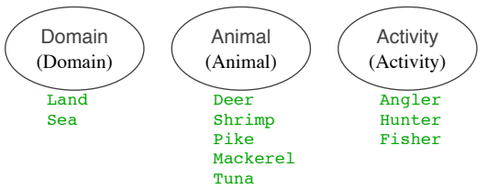
2 Catch is a result or derived data, not base data, not relevant at this stage.

3 Determine the entities. We have here, two Things:
Actors (anglers, fishers and hunters are all Actors)
Animals (fish, shrimp and deer are all Animals)

4 These are the **Relational Sets** or Entities (quite differents to the data-content sets in [*4.A*])

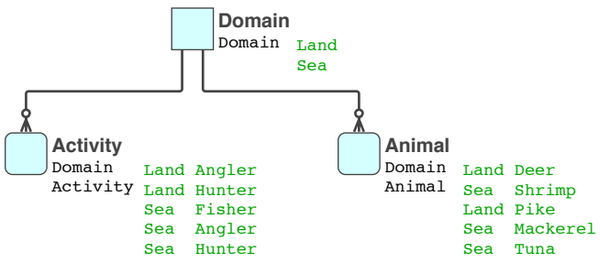
5 The item that is not identified, that must be exposed, is that both Actors and Animals exist within, and are limited to, a Domain. That is the third Thing. If this is not clear, revert to [*4.A Mass Approach*].

6 At this stage, **Key Determination**, the Entities, the Sets, are defined by their Keys, only. Do not concern yourself with attributes. The data content is provided for descriptive purposes, only.

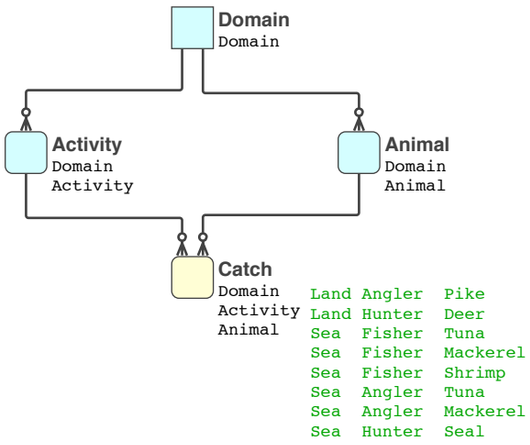


7 Now determine the organisation of the Sets, and model it. Choose the Keys (Identifiers) to suit that organisation.

8 Of course the foundation Set is Independent, the rest are Dependent. (The methodology is IDEF1X, unchanged: Codd and Brown, based on Chen's earlier work. However an extended *notation* that emphasises the Key is used; it should not require explanation.)



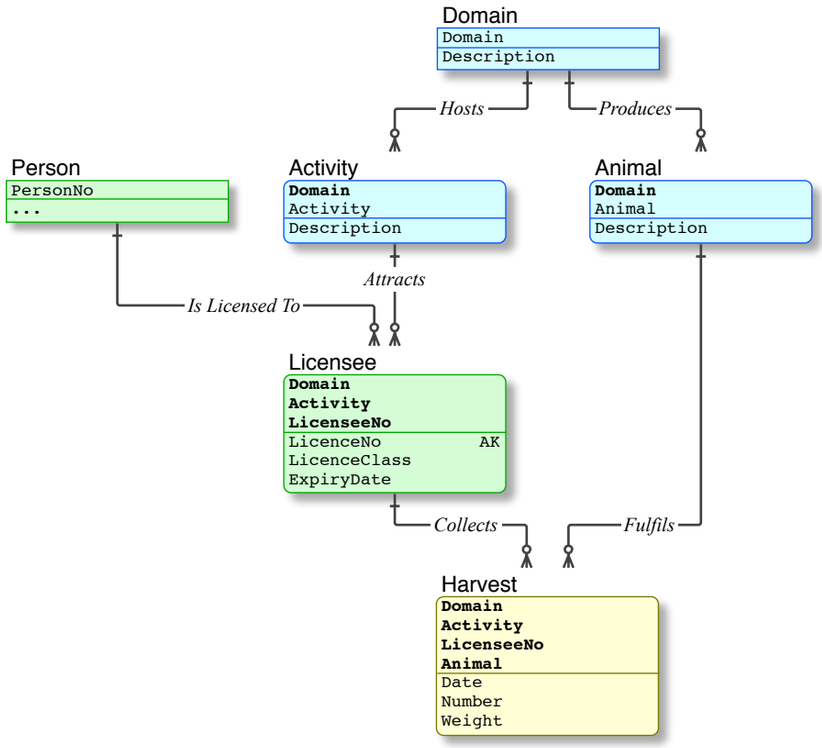
9 Now determine Catch, where it belongs, and its Identifier. It is a combination of Activity and Animal, ie. it has the Identifying characteristics of both.



10 The Catch is already limited to a Domain, by virtue of its Identifier, and that Domain is limited to *both* the Domain of Activity *and* the Domain of Animal. Thus a land actor can never catch a sea animal, etc.

5 Data Model

The exercise [*4 Data Analysis*] addresses all the data given in the subject paper, it is complete in itself. Nevertheless, a full Data Model has been requested by the bank, it is provided here. That however, exposes the issue that the elements given in the subject paper are incomplete, without context [1]. In order for the Data Model to be complete, such that each object can be contemplated in its full context, all relevant tables are given. The sample data excludes LicenseeNo, it matches [*4.B Set Approach*].



Domain
Land
Sea

Activity
Land Angler
Sea Sea-Fisher
Land Hunter
Sea Whaler

Animal
Land Deer
Sea Shrimp
Land Pike
Sea Mackerel
Sea Tuna

Harvest

Land	Angler	Pike
Land	Hunter	Deer
Sea	Sea-Fisher	Tuna
Sea	Sea-Fisher	Mackerel
Sea	Sea-Fisher	Shrimp

Entity Type
Reference
Identifying Entity
Transaction

IDEF1X Notation

Example Transaction
T License_Grant/Revoke
T Harvest_Add/Mod/Drop

- Notice the result is an Hierarchy, in both:
  - the layout of the tables, and
  - the components of the Keys.
- Per exercise [*4 Data Analysis*], the Integrity of the Harvest (which the subject paper is missing) is furnished by the Keys, which must be present anyway. If the Licensee Primary Key and Alternate Key are switched, as is typical of 'good' Record Filing Systems, or if there are no Primary Keys (all surrogates), as is typical of 'poor' Record Filing Systems, the Integrity of Harvest is lost.
- 'Catch' is merely the Harvest of one particular Date. Likewise 'Trip' is a particular Date range. Harvest is the name of the row, and of the set.

1 There is no suggestion that the subject paper itself is incomplete. The data elements given are quite adequate for its stated purpose. That purpose is devoid of context.

6 Application

6.1 Classifier (Suggested)

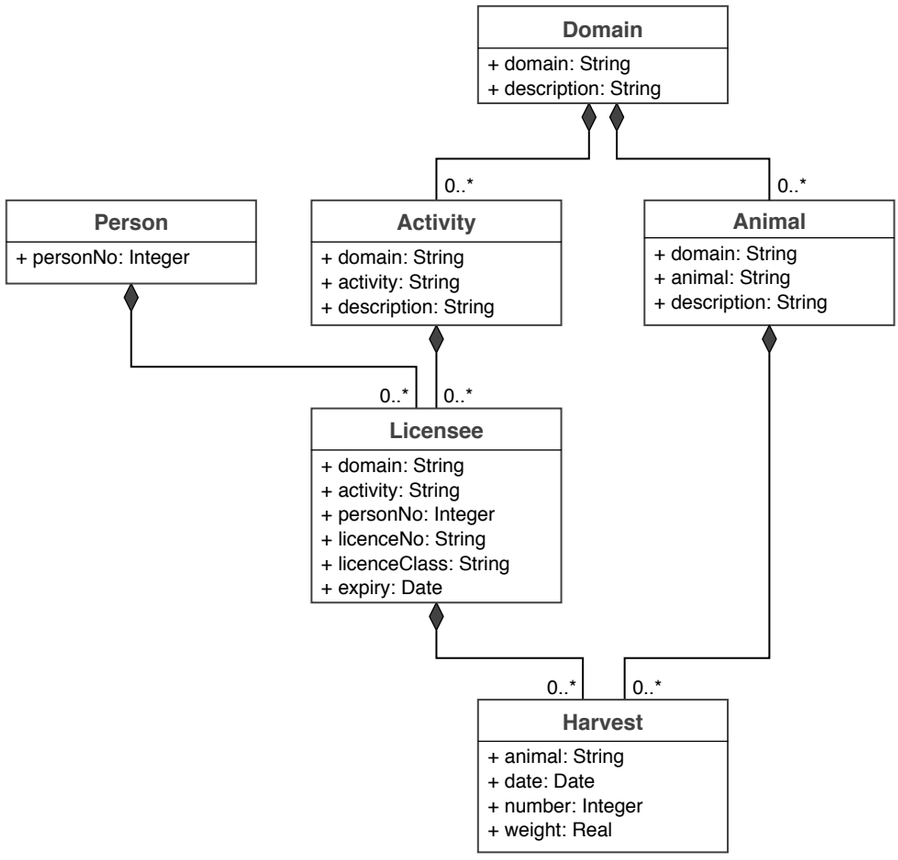
The implementation of an Open Architecture Relational database poses no limitations to the design of the application, or the classifiers within it. Indeed, it eliminates the great bulk of "Object Relational Mapping", which is desirable, since that "mapping" is fraught with problems.

1 That the data content of *all* objects will enjoy **Data Integrity** is guaranteed by, and within, the Relational Database, it has nothing to do with the classifiers or the application.

2 Therefore *any* collection of classes may be defined in an application that uses such a database. A suggestion of classes is given.

3 UML is intended for the OO monolith model, where everything is assumed to be in a single package (it has just one object, with a mass of notation), in particular it is inadequate for defining data elements, specific types of dependencies,constraints, etc. Thus:

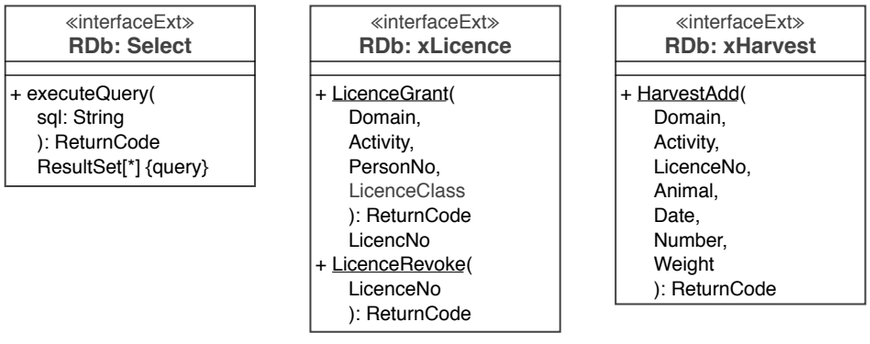
- This should not be viewed as equivalent to the Data model, which is rich in specific detail (particularly re the data hierarchy; details of the relationships), even though the overall structure appears to match it. These are object classifiers, and it is merely a one-to-one mapping.
- These classifiers are for definition purposes, to fill windows, drop-downs, etc, thus they 'match' the tables (or views) in the database.
- There are no formal rules for data objects, let alone Relational objects, let alone specific dependency types. It is a free-for-all. Thus the notation used is simply that which is most understood by the project team. The aggregators may just as well be simple dependencies.
- The cardinality at the aggregator end is always 1.



6.2 Method

Questions have been raised re how the Relational database will be updated, in the absence of OO/ORM direct updates. Although this will be covered in detail in the education delivery, an example is provided here, in order for this response to be complete. Again, the **Open Architecture** document provides an overview. The method that is used to update the database consists of a set of Methods, it is the **Database API**.

- Within the scope of "Object Relational mapping", the most problematic area concerns the direct updates to database tables. That entire set of problems is eliminated by the implementation of a set of ACID Transactions, which can be executed by any application.
- Each Method is in fact a Transaction stored procedure, resident in the database. The Method is merely a wrapper on the application side to call it from the OO framework.
- UML, in its monolithic myopia, does not provide symbols or notation for an external system, thus the notation is stretched here. Again, in the notation that the project team understands. ReturnCode values are documented once, elsewhere.



6.3 Instance/Monolith

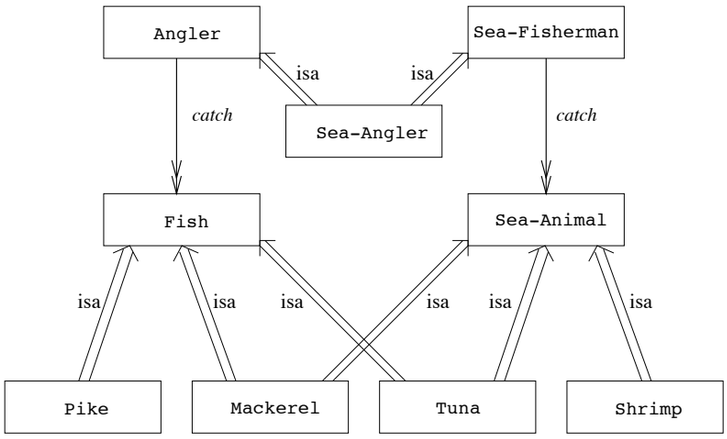
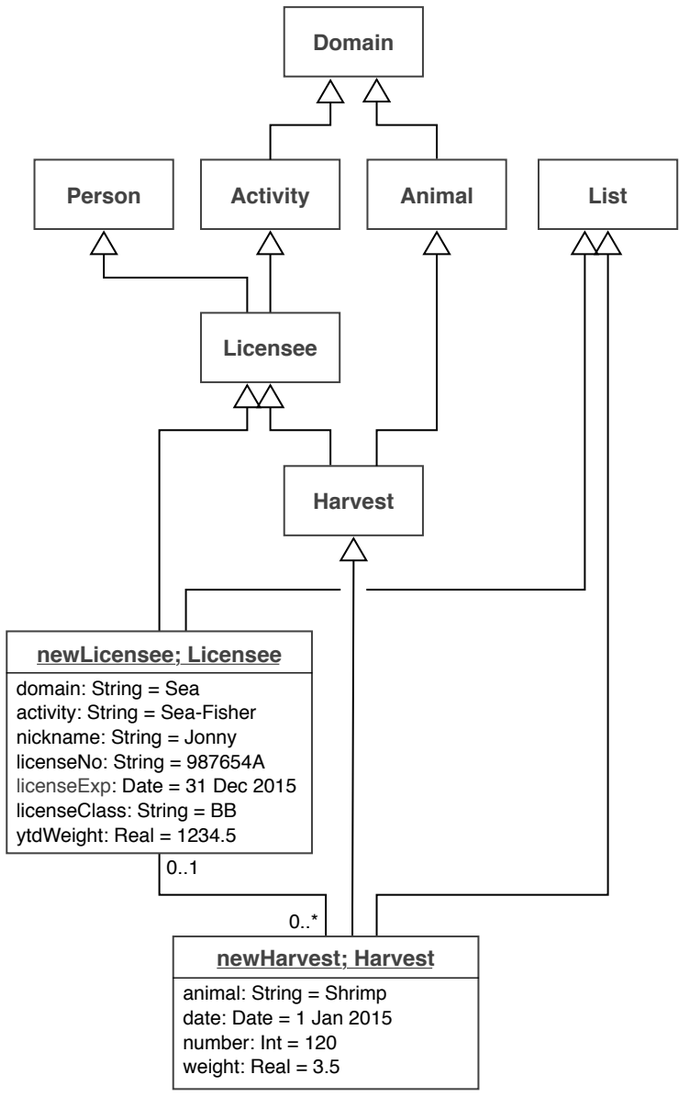


Figure 1: A schema with multiple inheritance

1 The notations used are not recognisable, thus I cannot produce the equivalent in the solution below. Classes are mentioned but not defined. However, as discussed in the text, I understand the subject author intends to show the lack of integrity in the data that occurs under multiple inheritance.

- Which, notably, occurs only where architecture, standards and the Relational context, are absent, where the monolith is constructed in isolation.
- Notice the absence of context, of Domain

6.4 Instance/Open Architecture



1 In order to afford a comparison with the datagram, some instances are given, without detailing the instantiation of classes.

- This is done reluctantly because we do not wish to validate the focus on data content, thus we assert, the classification and typing [*4 Data Analysis*] is relevant, the data content (this page) is not.

2 Multiple inheritance may be designed and implemented without Data Integrity being a constraining factor, or even a consideration.

- This further confirms that the implementation of Open Architecture Relational database eliminates a variety of problems, and that it provides more freedom: here in application design space.

3 Again, the classes are merely a suggestion. The Key values may be carried (not shown) or not carried (shown), depending on the extent of dependence on the inherited classifiers.