

1.0 Introduction

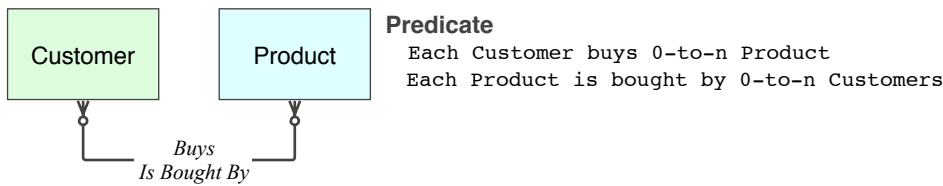
This document was initially produced to provide diagrammed support for a thread in a forum that is text-only.

"Theoreticians" are Clueless about Relational Data Modelling, Teach Anti-Relational Muddling

As the thread progressed, it became a response to a proposal made by Nicola Vitacolonna. Approaching closure, it has become a stand-alone article, defining Subtypes from first principles. The two initial sections are retained, in order to support the understanding of anyone navigating the thread: otherwise they can be skipped.

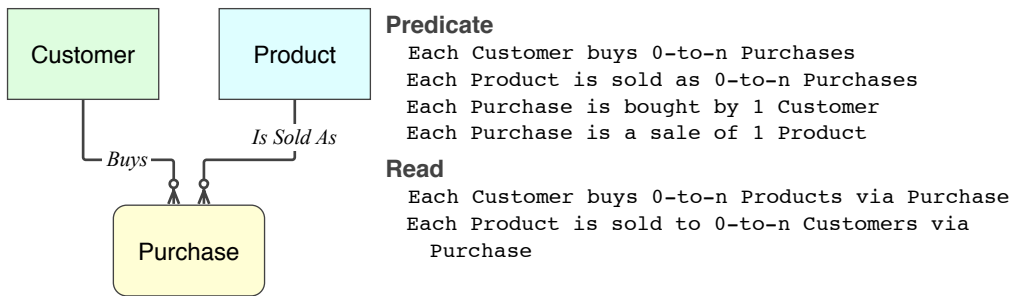
1.1 Associative, Logical • IDEF1X Entity Level

Unresolved: early Logical stage



1.2 Associative, Physical • IDEF1X Entity Level

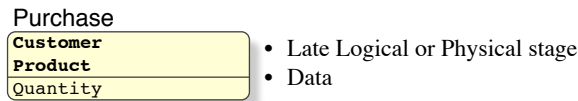
Resolved: late Logical or Physical stage



1.3 Binary, Associative Entity



1.4 Binary, Not Associative Entity



1.5 Rolename

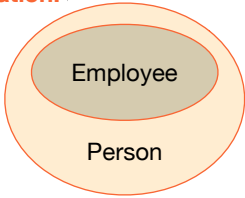
Roles are defined in the *Relational Model*. The *rendition* is in IDEF1X, the naming is a bit cumbersome. Nevertheless, at the physical stage, the role must have a normal SQL column name. Rather than switch back-and-forth between an IDEF1X Rolename and a resolved one, I use the resolved Rolename throughout the modelling exercise (it does not violate the *RM*). Examine:

- **Bill of Materials**: PersonId ... ParentId ... ChildId; ShortName ... Assembly ... Component
- **Order DM Advanced**: PartyNo ... CorporationNo ... CustomerNo ... VendorNo

1.6 Re "0. Definitions"

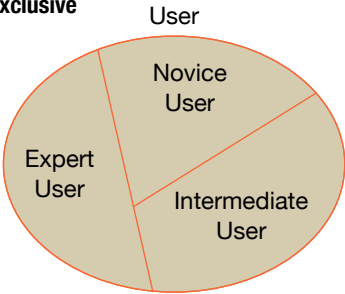
This addresses articles given in Nicola's V4 document, under **0. Definitions** (left) along with suggestions (right). The definitions themselves (accuracy; veracity; etc) are not addressed here.

Specialization:

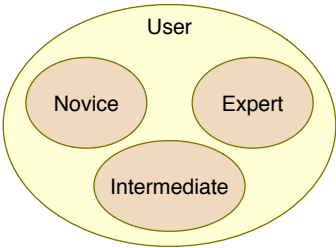


- The definition, and the example, are too contrived
- issues are too many to list here, detailed in the post
- rejected outright.

Total, Exclusive

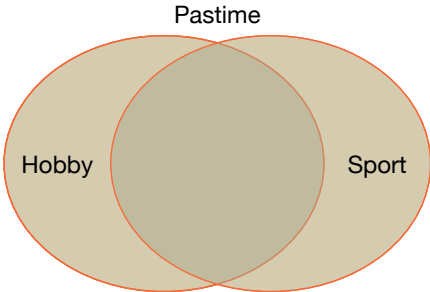


Correction:

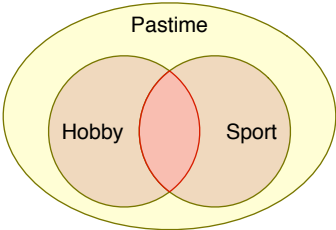


- The Venn diagram is inadequate for portraying Relational sets
- This class of error, the misunderstanding of Relational Sets, is common among OO/ORM developers and SQL novices. Especially so when Venn diagrams are used.*
- The Venn diagram can portray exclusivity adequately, without butchering
- There are four sets, not three:
 - a set for the Subtype **set**, which should not be hidden, accidentally or purposely
 - a set for each subtype.

Total, Non-Exclusive

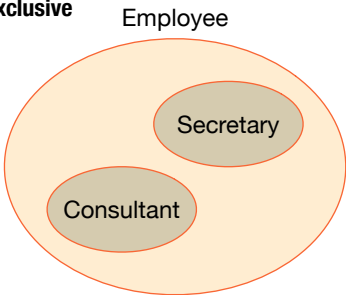


Correction:

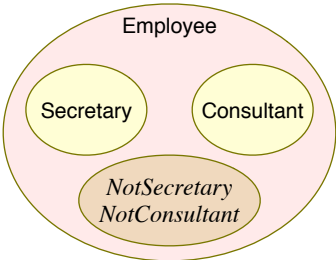


- There are four sets, not three:
 - a set for the Subtype **set**, which should not be hidden
 - a set for each subtype
 - a set for each intersection
- The Venn diagram is inadequate when the number of Subtypes is greater than two.*

Partial, Exclusive



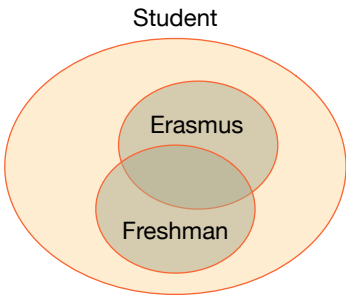
Correction:



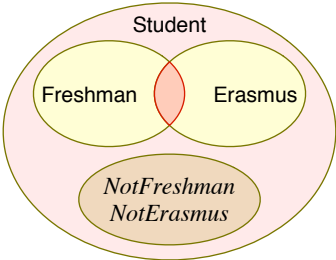
- I do not accept this classification, I am merely correcting your definition*
- The sets, made explicit

- Here the Specialisation **set** is confused with the **set** of Employee who is NotSecretary **and** NotConsultant is loose (orange minus brown)

Partial, Non-Exclusive



Correction:



- I do not accept this classification, I am merely correcting your definition*
- The sets, made explicit

- Here the Specialisation **set** is confused with the **set** of Employee who is NotSecretary **and** NotConsultant is loose (orange minus brown)

2.1 ERwin Methods Guide, p 48, **Annotated**

Relationship Cardinality

Up to this point, we have discussed one-to-many relationships in a logical model, without capturing any information on what we mean by the word “many.” The idea of “many” does not mean that there has to be more than one instance of the child connected to a given parent. Instead the “many” in one-to-many really means that there are zero, one or more instances of the child paired up to the parent.

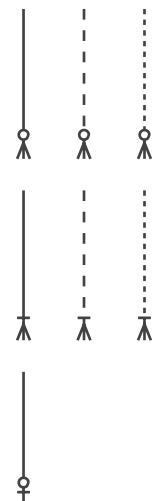
Cardinality is the relational property that defines exactly how many instances appear in a child table for each corresponding instance in the parent table. IDEF1X and IE differ in the symbols are used to specify cardinality. However, both methods provide symbols to denote one or more, zero or more, zero or one, or exactly N, as explained in the following table.

Cardinality Description	IDEF1X Notation		IE Notation	
	Identifying	Non-identifying	Identifying	Non-identifying
One to zero, one, or more				
One to one or more				
One to zero or one Unreferenced classification prohibited ("spinster" parent prohibited)				
Zero or one to zero, one, or more (non-identifying only) Parent is mandatory ("optional parent" is prohibited)				
Zero or one to zero or one (non-identifying only) Parent is mandatory ("optional parent" is prohibited)				

Derek Asirvadem Extension

IE Improved

Parent is always mandatory

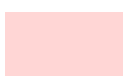
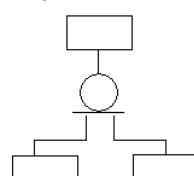
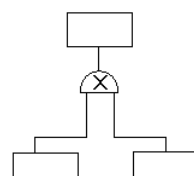

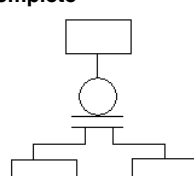
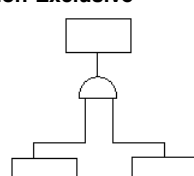


- Parent is present at insertion of child, but may be removed
- Thus a FK Constraint cannot be declared

2.2 ERwin Methods Guide, p 69, **Corrected**

IDEF1X and IE Subtype Notation

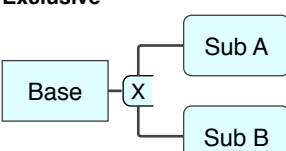
The following chart illustrates subtype notation in IDEF1X and IE.

	IDEF1X Subtype Notation Inferior to pre-existing IEEE & Invalid	IE Subtype Notation
	Incomplete 	Exclusive 
	Complete 	Non-Exclusive 

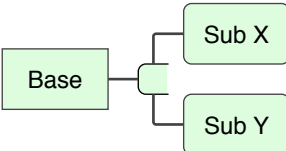
IE Improved

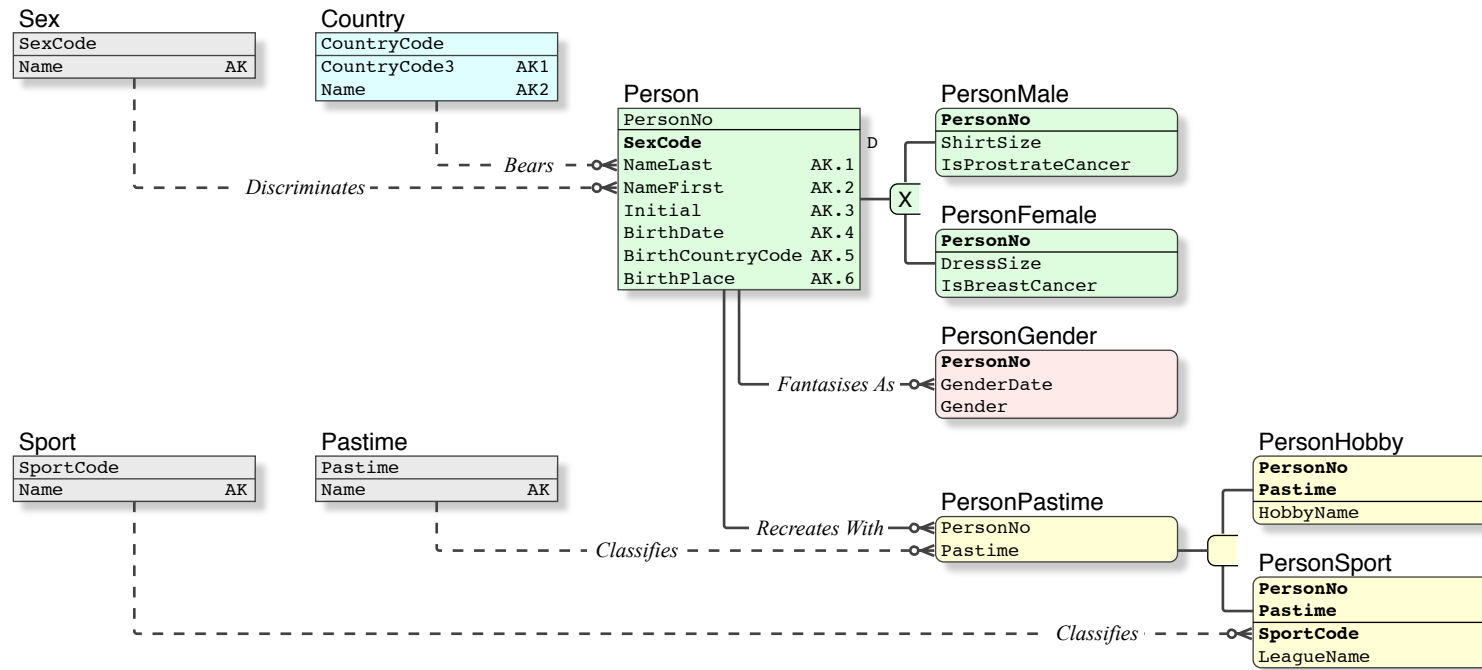
- Improved symbol denoting the basetype is incomplete
- Subtypes placed beside the basetype: desirable when the model is laid out in hierarchic order.

Exclusive



Non-Exclusive





Other Constraint

- Ⓒ **PersonMale_IsExclusive_ck**
CHECK Person[PersonNo].SexCode = "M"
- Ⓒ **PersonFemale_IsExclusive_ck**
CHECK Person[PersonNo].SexCode = "F"

Transaction

- Ⓓ **Person_Add_tr**
-- Validate ...
IF EXISTS @Sex
AND EXISTS @CountryCode
AND NOT EXISTS @Person.AK
BEGIN TRAN
INSERT @Person
IF @SexCode = "M"
INSERT @PersonMale
ELSE
INSERT @PersonFemale
COMMIT TRAN
- Ⓓ **PersonGender_Add_tr**
-- Validate ...
IF EXISTS @PersonNo
BEGIN TRAN
INSERT @PersonGender
COMMIT TRAN
- Ⓓ **PersonHobby_Add_tr**
-- Validate ...
IF EXISTS @PersonNo
AND NOT EXISTS PersonHobby
BEGIN TRAN
IF NOT EXISTS Pastime
INSERT @PersonPastime
INSERT @PersonHobby
COMMIT TRAN
- Ⓓ **PersonSport_Add_tr**
-- Validate ...
IF EXISTS @PersonNo
AND NOT EXISTS PersonSport
BEGIN TRAN
IF NOT EXISTS Pastime
INSERT @PersonPastime
INSERT @PersonSport
COMMIT TRAN

Predicate

All Predicates are established in an IDEF1X Data Model, in diagrammatic form. For those unfamiliar with Predicates or IDEF1X, they are duplicated here in text form.

Country

is independent
is primarily identified by (CountryCode) -- 2-char ISO
is alternately identified by (CountryCode3) -- 3-char ANSI
is alternately identified by (Name)
bears 0-to-n Persons

Sex

is independent
is primarily identified by (SexCode)
is alternately identified by (Name)
discriminates 0-to-n Persons

Pastime

is independent
is primarily identified by (Pastime)
is alternately identified by (Name)
classifies 0-to-n PersonPastimes

Sport

is independent
is primarily identified by (SportCode)
is alternately identified by (Name)
classifies 0-to-n PersonSports

Person

is independent -- exists independently
is primarily identified by (PersonNo) -- relational breach
is alternately identified by (NameLast, NameFirst, Initial, BirthDate, BirthCountryCode, BirthPlace)
is discriminated by 1 SexCode
is born in 1 Country
is an exclusive Basetype, one of { Male, Female }
identifies, and fantasises as 0-to-n Genders
identifies, and recreates with 0-to-n Pastimes

PersonMale

is an exclusive Subtype of, and identified by, 1 Person
is primarily identified by (PersonNo)
is described by (ShirtSize, IsProstrateCancer)

PersonFemale

is an exclusive Subtype of, and identified by, 1 Person
is primarily identified by (PersonNo)
is described by (DressSize, IsBreastCancer)

PersonGender

is dependent on, and identified by, and a fantasy of, 1 Person
is primarily identified by (PersonNo, GenderDate, Gender)

PersonPastime

is dependent on, and identified by, and a recreation of, 1 Person
is classified by 1 Pastime
is primarily identified by (PersonNo, Pastime)
is a non-exclusive Basetype, any of { Hobby, Sport }

PersonHobby

is a non-exclusive Subtype of, and identified by, 1 PersonPastime
is primarily identified by (PersonNo, Pastime)
is described by (HobbyName)

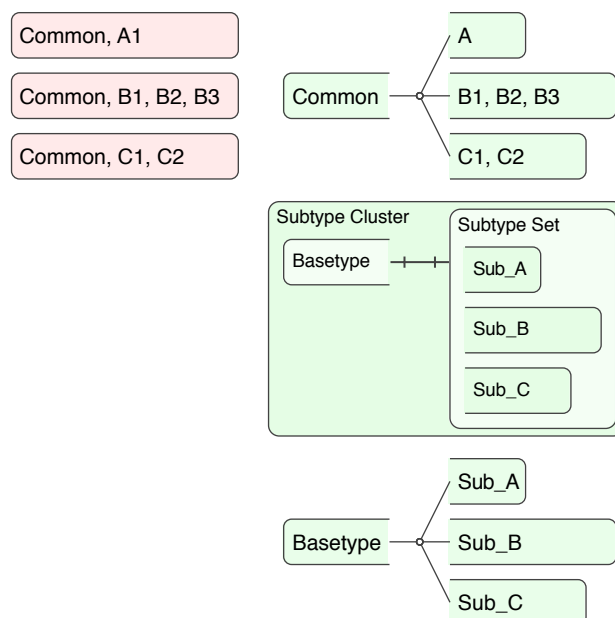
PersonSport

is a non-exclusive Subtype of, and identified by, 1 PersonPastime
is classified by 1 Sport
is primarily identified by (PersonNo, Pastime)
is described by (SportCode, LeagueName)

Subtypes are fully defined and treated in my [Subtype](#) document, in an implementation context. That document does not contain a theoretical definition of what a Subtype is; what its nature is. That is provided here.

3.1 Definition

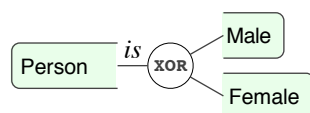
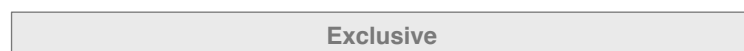
The concept of Subtypes existed long before the *Relational Model*. All the DBMS of the day permitted it, and some had explicit implementation methods. This section discusses the nature of Subtypes (logical), and does so without reference to a notation (physical), so that the concept (logical) can be understood before evaluating a notation.



- 1 First, to understand the need: the subject of attention is a set of rows, that have both common and discrete elements.
 - The common elements need to be Normalised into a common row (such that they are not duplicated), and the discrete elements into discrete rows.
- 2 A Subtype cluster consists of one Basetype and a **set** of Subtypes
 - the **set** of Subtypes is two or more members **not** one or more, refer to §3.1.1 below
 - the cardinality of the relation between the Basetype and **set** of Subtypes is 1::1 the set is mandatory, ie. there is no such thing as a Basetype without a Subtype
 - an entity may have more than one Subtype clusters, ie. it is a Basetype in more than one Subtype Set.
- 3 the cardinality of the relation between the Basetype and **each** Subtype is 1::0-to-1
 - whether the Basetype is one Subtype of the set, or more, depends on the classification (next)
 - each Basetype and Subtype pair should be perceived as a single logical unit.

4 The next step is to control the relation between the Basetype and the Subtypes.

- The established classification or type of control ¹ is IEEE{Exclusive|Non-Exclusive}
- the relation; the Predicate; the *VerbPhrase*, is always [*is*] in both directions. As such, it is not stated in the model
- the Predicates for each classification are given next.



- The Subtypes in the set are mutually exclusive
- the Basetype is related to one Subtype aka an XOR Gate
- it is controlled by a **Discriminator** in the Basetype, and an additional constraint.

Person

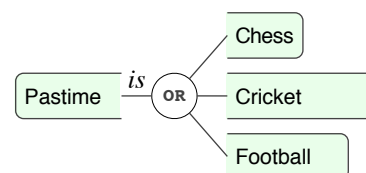
```
is { independent | dependent }
is primarily identified by ( PK )
is discriminated by 1 SexCode
is an exclusive Basetype, one of { Male, Female }
```

Male

```
is an exclusive Subtype of Person
is identified by 1 Person
is primarily identified by ( Person.PK )
```

Female

```
is an exclusive Subtype of Person
is identified by 1 Person
is primarily identified by ( Person.PK )
```



- The Subtypes in the set are not mutually exclusive
- the Basetype is related to one or more Subtypes aka an OR Gate
- a **Discriminator** is not required ²

Pastime

```
is { independent | dependent }
is primarily identified by ( PK )
is a non-exclusive Basetype, any of { Chess, Cricket, Football }
```

Chess

```
is a non-exclusive Subtype of Pastime
is identified by 1 Pastime
is primarily identified by ( Pastime.PK )
```

Cricket

```
is a non-exclusive Subtype of Pastime
is identified by 1 Pastime
is primarily identified by ( Pastime.PK )
```

Football

```
is a non-exclusive Subtype of Pastime
is identified by 1 Pastime
is primarily identified by ( Pastime.PK )
```

¹ Other classifications (types of control) existed, but they were minor, or product-specific.

² If there are rules regarding the members of the Subtype Set, such as some exclusivity, (a) the set should be Normalised correctly, and (b) the rules implemented as Constraints, a Discriminator is quite useless.

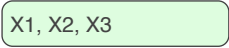
3.1.1 Exclusion

According to the Philosopher, a definition of a concept must declare what it is not, in order to be complete. There are two mistakes that are made by naïve developers.

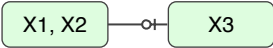
1 A Subtype cluster consisting of a single Subtype



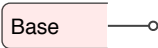
- It is unnormalised, an absurdity. There is nothing common, upon which a container (Basetype) to hold it can be asserted
- It fails the Relational Predicates (detailed in the section above)
- When Normalised, the Subtype columns become part of the Basetype, an ordinary row is formed:



- If it is optional, the set of optional attributes may be deployed to a separate table to host it. In both cases, the childless Basetype disappears.



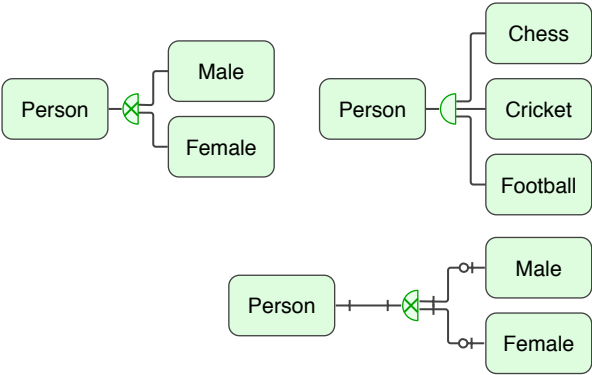
2 A Subtype set that is optional, ie. a "Basetype" that has no "Subtype"



- It is a gross error in classification (or falsely labelled). Therefore it is not a Basetype
- While the relation may be switched, the premise of a Subtype cluster is that the *set* of Subtypes is mandatory, the Basetype must relate to at least one Subtype
- Therefore the optional "Subtype" is merely an optional row. Again, potentially deployed in a separate table.

3.2 IEEE Notation

Now we consider the physical. In data models, the IEEE notation was used, it was the only notation available (minor others were incomplete, they were dismissed), and already established.



- **Exclusive:** a half circle with an X
 - it is controlled by a Discriminator in the Basetype, and an additional constraint.
- **Non-Exclusive:** a half-circle
 - a Discriminator is not required
- the orientation is as per the modeller's convenience
- the cardinality is known (previous section), it is not shown in the notation.

In the physical, of course each Subtype has a Foreign Key to the Basetype. In the logical, the switch is in the relation, not in the Basetype or Subtype, because:

- although each Basetype::Subtype pair is 1::0-to-1
- the cluster, the sum, the Basetype::SubtypeSet is 1::1 (mandatory)

Subtypes were easily implemented in the DBMS or ISAM of the day, because it is an intellectual, not an implementation issue. Some DBMS provided explicit methods; others needed no addition, just a "how to" guide. Of course the implementor who used ISAM did so by writing the DBMS constraints into his library code.

3.3 IDEF1X

When IDEF1X came along, it was very well received. Mainly due to the fact that had been no progress, no articulation of the *Relational Model* whatsoever (other than Codd's own diffusion in his attempts to obtain acceptance of the *RM* in academia which was hostile): not a single academic paper. The second reason was, in contrast to the *RM* theory which broke completely new ground, it contained practical steps and was useable by practitioners.

But there were problems:

- While IDEF1X was great in the arena of articulating some of the elements of the *Relational Model*,
 - it ignored other elements, and
 - it contradicted the *RM* regarding yet other elements (eg. the Record ID is prohibition is not observed)
- it ignored the established Subtype implementations and the IEEE notation that served it.
- it introduced a number of new concepts and terms, without complete definitions, that were neither tried nor tested
 - at worst, these had the effect of argumentation without resolution, which lead to incorrect implementations
 - at best, the new concepts were determined as errors, and dismissed
 - one central plank in that arena was Subtyping: that was particularly bad because Subtyping was well and truly established, and it ignored that real world evidence³, and sought to redefine it (again without trial or test)
- the writing is not precise, nor technical enough to be used readily, let alone to be published as a Standard
 - it contained ambiguities, beloved of academics, detested of scientists. That is, the "definitions" are not tight enough to qualify as technical definitions, the subordinate "theory" for such items is quite incorrect
 - it contained contradictions in its own "definitions", which in the normal case would disqualify the entire paper
- thus the experienced implementor qualified the

It needs to be understood at this point, that the academics suppressed the *Relational Model*, by (a) not producing a single paper that advanced or progressed the understanding of it, and (b) instead produced volumes of papers that presented 1960's Record Filing Systems as "Relational". So when IDEF1X came out, and in the years afterwards, the academics either ignored it completely or wrote up minor points of error, oblivious to the major errors described above.

Due to my absolute adherence to the *RM* and my own articulation of it⁴, and due to the errors in IDEF1X, I took the best concepts from it and ignored the rest. That was good enough for the experienced Relational Data Modeller. Other implementors did likewise, to different degrees of precision.

However, when large customers requested it, following its publication as a Standard in 1993, I wrote a paper that addressed all the issues in IDEF1X, and the resolution. The Software Gems' **IDEF1X Qualification** document, that is included in every Relational Database delivery, identifying the precise Standard that the delivery conformed to. It is not published stand-alone.

3.4 ERwin

Subsequently a Data Modelling tool, *ERwin* came out in the early 1990's. It was the first such tool, and it was the first to use and implement IDEF1X to some degree, and it produced SQL DDL, elevating database creation to a much higher level. Because IDEF1X was confused and unuseable as is, LogicWorks provided the [ERwin Methods Guide](#) which sought to provide useable definitions and methods. For this reason, this important document became the *de facto* IDEF1X manual, used by many, including those who were not *ERwin* customers.

- While it did correct some IDEF1X errors (unfortunately not all), it reinforced other errors by providing a "definition" or method for them.
- Eg. it propagated the crippling error that Record IDs were "Relational", by way of examples throughout the document
 - it failed to communicate Relational Keys as defined in the *Relational Model*
 - Eg. since IDEF1X Subtyping is hopeless,
 - it included the established Subtype concept, and the IEEE notation, in order to afford a data model that was coherent, that worked
 - but it included the IDEF1X Subtype {Complete|Incomplete} *titles* as well, failing to appreciate that it is was a hopeless error
 - and worse, it introduced a naïve "definition" of {Complete|Incomplete}⁵ under those *titles*, that directly contradicted the IDEF1X concept and "definition" of {Complete|Incomplete}, but which has now become the common understanding of IDEF1X Subtype {Complete|Incomplete}. That is, although it is incorrect, it is the commonly understood one.

IDEF1X "Definition"			ERwin Version of IDEF1X (Common)		
Categorisation (Incoherent)	Complete	All Basetypes have one Subtype	Subtype (Naïve)	Complete	The set of Subtypes is complete
	Incomplete	A Basetype without a Subtype exists		Incomplete	The set of Subtypes is not complete

- As can be seen, the *ERwin* notion of {Complete|Incomplete} in its naïveté, is quite irrelevant: virtually all Categories, except a few such as {Male|Female}, are Incomplete, regardless of the modellers declaration. In light of the pre-existing IEEE Subtype concept definition and notation [§3.1 & 3.2](#), it is doubly ridiculous.
- Thus capable practitioners dismiss the IDEF1X Categorisation concept & notation, and always use the IEEE concept & notation.
- A comparison of the discussed IEEE Subtypes vs the IDEF1X Categorisation is given in [§5.2](#).

Thus over time, the IDEF1X "definition" and treatment of Categorisation was buried, unfortunately not due to its incoherence, but due to a common misunderstanding.

The *ERwin* Methods Guide, too, was evidently written by a non-technical person, which was part of the appeal. Again, due to its exposition of IDEF1X, and its immediate useability, and despite its errors (unrecognised by all but the faithful) this important document became the *de facto* manual for IDEF1X. It still is today (for organisations other than my customers and those who follow my articles).

3 That error was evidently due to the writer having apprehended subtyping from an implementation in one or the other DBMS, which would of course be quite limited, rather than from the theory [\[§3.1 & 3.2\]](#) or from actual practice. This is a common error among academics, it is much worse today: *PissGrossNONsql* is paraded as as "SQL" is defiance of the ISO/IEEE Standard; as a "platform" in defiance of SQL Platforms that we have had since 1983, and in ignorance of Server Architecture.

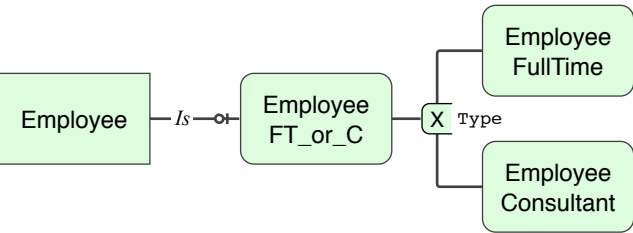
4 Notably, I do not suffer, and have never suffered the various alleged "problems" or "failures" in the *Relational Model*, that academics have written volumes about. Such papers are unscientific, because (a) they deny reality, and (b) rely on Straw Man arguments.

5 Either the original LogicWorks [ERwin Methods Guide](#) page 67, or the current online [ERwin/Complete Compared to Incomplete Subtype Structures](#).

4.1 Conceptual Requirement, Labelled "Concrete"

- "Concrete example:
- An Employee may be a Consultant;
 - An Employee may be a Full-Time Employee;
 - An Employee may be neither a Consultant nor a Full-Time Employee;
 - An Employee cannot be both a Consultant and a Full-Time Employee."

4.2 Conceptual Answer to Conceptual Requirement



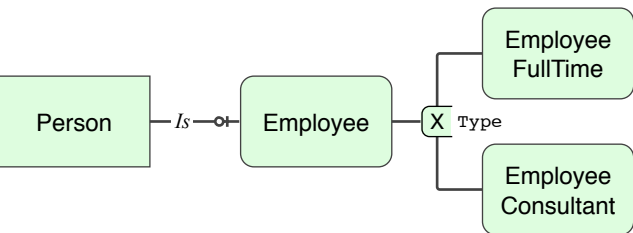
- 1 In consideration of the fact that the purpose of the [Subtype Discussion] is for discussion of Subtypes, using the Predicates given therein, validate this model
 - a correct model cannot be reduced: a reduction will cause a loss of definition (constraint)
- 2 Attempt validation of Nicola's "three-entity" model (please do not ask me to draw an error), considering the Predicates.
 - it requires *three Facts in one place*, which is illegal in the Relational paradigm
 - it requires the PK to address the Discriminator, to contrive a "fact" separate to the PK.

4.3 Academic Deems His Conceptual Model "Concrete"

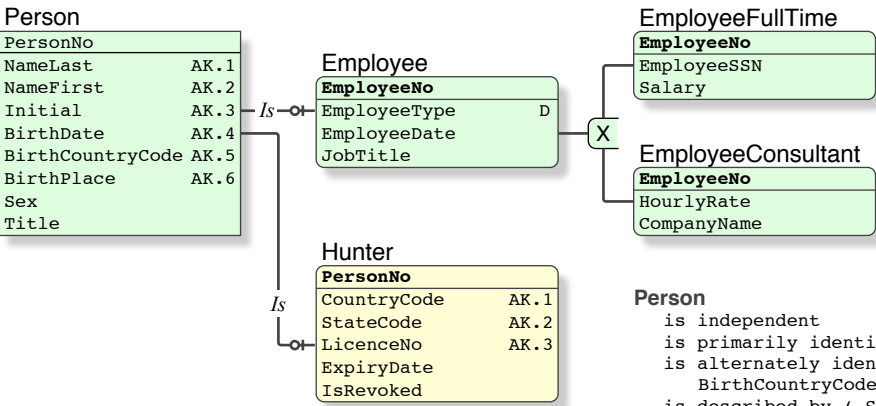
His model (not shown here), not mine [2]. Cuz he said so. Without the Logical. Without the Physical (is not concrete physical ?). Not possible. But then he makes it an Universal. Perhaps in the dark halls of what passes for "education" these days, with the lights turned off, and imaginary companions to snuggle with.

4.4 Practitioner Delivers Concrete

- I will skip the increments and go directly to a reasonable real world implementation.
- Here is an Employee cluster that models the given requirement.
- Much like, a n::m relation at the Logical Level is an Associative table at the physical level (concrete), wherein the real world practitioner dispenses with the additional work of hiding the essential Associative table at the Logical level:
 - to propose that something that exists "at the physical level" does not exist "at the Logical level" is insane.
- In [1][2] the Fact of Employee vs Employee_FT_or_C is exposed as unrealistic (not real world, no possibility of concrete), but a perfectly valid academic-only exercise.
- There are likewise other issues in the conceptual requirement, that classify it as incomplete, such as, *what is heavens name is an Employee who is neither FullTime nor a Consultant* which thus again deem it premature for a concrete evaluation, and again an academic-only exercise.



Person = Fact of existence of a person
Employee = Fact of employment of a Person
Employee{FT,C} = Fact of type of employment



- Employees must provide their Social Security Number SSN for taxation purposes
- Conversely, a Consultant does not, he manages his own taxation, through his own company
- The Fact of Employee, as distinct from that Discriminator EmployeeType, has attributes.
- The academic relies on EmployeeType (the fragment of an entity), as as entity.

- Ⓢ EmployeeFullTime_IsExclusive_ck
CHECK Employee[EmployeeNo].EmployeeType = "F"
- Ⓢ EmployeeConsultant_IsExclusive_ck
CHECK Employee[EmployeeNo].EmployeeType = "C"
- Ⓢ Employee_EmployeeType_ck
CHECK EmployeeType IN (F, C)

Person
is independent -- exists independently
is primarily identified by (PersonNo) -- relational breach
is alternately identified by (NameLast, NameFirst, Initial, BirthDate, BirthCountryCode, BirthPlace)
is described by (Sex, Title) [1]
is born in 1 Country
identifies, and is 0-to-1 Employee
identifies, and is 0-to-1 Hunter

Employee
is identified by, and dependent on, 1 Person
is primarily identified by (EmployeeNo)
is described by (EmployeeDate, JobTitle)
is an exclusive basetype, one of { FullTime, Consultant }

EmployeeFullTime
is an exclusive Subtype of, and identified by, 1 Employee
is primarily identified by (EmployeeNo)
is described by (EmployeeSSN, Salary)

EmployeeConsultant
is an exclusive Subtype of, and identified by, 1 Employee
is primarily identified by (EmployeeNo)
is described by (HourlyRate, CompanyName)

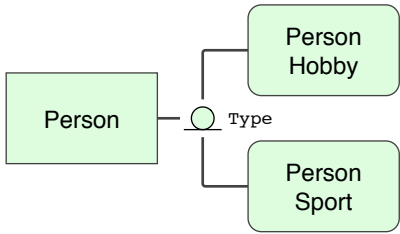
Hunter
is identified by, and dependent on, 1 Person
is primarily identified by (PersonNo)
is alternately identified by (CountryCode, StateCode, LicenceNo)
is described by (ExpiryDate, IsRevoked) [1]

1 The [strict] **Predicate Lexicon** is Normalised, such that only non-Key attributes are Descriptors.

As detailed, the original meaning of IDEF1X Categorisation {Complete|Incomplete} is lost ⁶, and the commonly understood and implemented meaning is used ⁶. Further, that meaning is irrelevant ⁶, thus the entire IDEF1X Categorisation is not used, it is defunct. Nevertheless, it is being evaluated now, in its original form, by one academic, forty years later. Thus a discussion is provided.

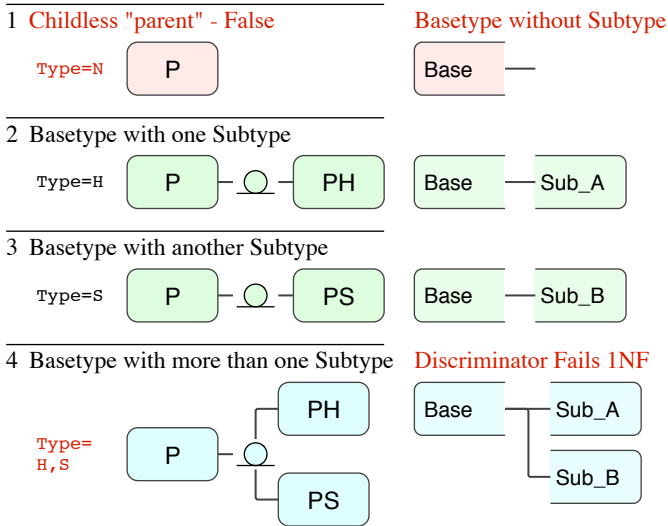
4.5 IDEF1X-Original[Incomplete] is Incoherent, Not Basetype

- According to the original "Definition" (as opposed to the commonly understood meaning):
FIPS 184 §3.6.1 para 5:
In an “incomplete category cluster”, an instance of the generic entity can exist without being associated with an instance of any of the category entities, i.e., some categories are omitted.
FIPS 184 §C1.4.2:
Note that axioms C1.4.1 and C1.4.2 do not preclude an incomplete categorization in which a generic has a discriminator value but no category, so long as no generic with that discriminator value has a category.
• For the "Incomplete" Subtype cluster modelled at below left, the permitted row combinations ("instances") are shown further below:



Person = Fact of existence of a person
Person = Fact of pastime of a person
Person = Fact of non-pastime of a person
PersonHobby = Fact of person is a hobby
PersonSport = Fact of person is a sport

- The notion of a Discriminator for "Incomplete" is hysterical: it constrains absolutely nothing
- Person stands for three Facts, two of which contradict each other
- The notion of Basetype without a Subtype [1] means it is not a Basetype or Base (common attributes, of a nothing) (A mother without a child is not a mother, she is an ordinary woman)
- Thus the IDEF1X notion of "Incomplete" is incoherent, false.
- The remaining notion of "Complete" is an ordinary requirement, pedestrian, it does not need to be stated
- Thus the entire classification is rejected outright.



Discriminator

1 IDEF1X calls for a Discriminator for both {Complete|Incomplete}, because both sets are mutually exclusive (it is not for supporting {Complete|Incomplete})

- Due to the poor definitions being misunderstood; the common understanding of {Complete|Incomplete} being quite different to the original definition; the absence of IEEE classifications and its strict definition, and the confusion that ensued, implementors often made mistakes in their implementations.
- one common error is shown here [4]: the implementor assumes that **Incomplete** means IEEE[Non-Exclusive], which is the most common need, and not expected to be absent in IDEF1X
- that combined with the demand for a Discriminator in both {Complete|Incomplete} leads to compounded errors
- he is ignorant, that a Discriminator is **not** required for IEEE[Non-Exclusive], and implements it as per the IDEF1X demand
- such a Discriminator breaks 1NF, and does nothing in terms of controlling the members of the set
- I do not suggest that IDEF1X *defines* this particular insanity, but that by virtue of the evidence, it is caused by the confusion and poor definitions therein.

2 The IDEF1X method given for implementation of the Discriminator requires a structure in both the Generic and in each Category

- the structure is massive, as it includes duplication and indexing of the Discriminator in each Category
- it is quite unnecessary, as the intention can be served by simple constraints
- but it is recognised, as one of the typical anti-relational and massively inefficient structures promoted by the detractors (Date; Darwen; Fagin; all their followers; all the compliant textbooks; all the professors that use the anti-theory; etc).

- Such a classification is so invalid that it cannot be corrected in place
- The IDEF1X concept and definition of {Complete|Incomplete} is rejected outright.

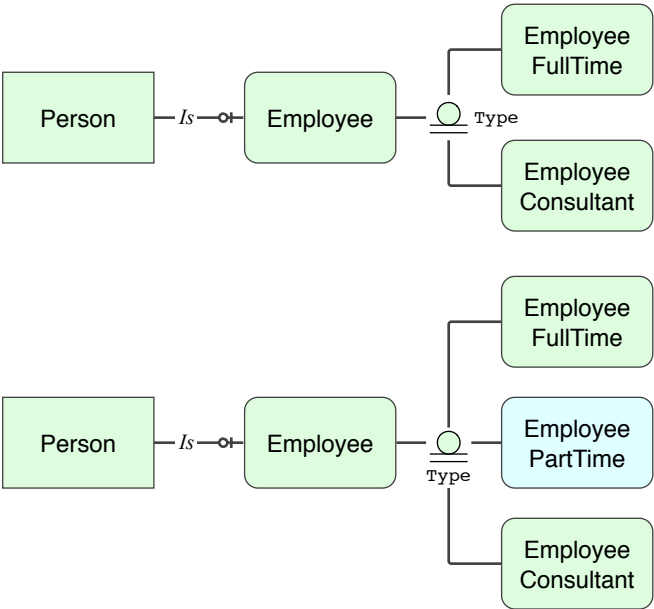
⁶ Refer to sections 3.1 to 3.4.

4.6 "IDEF1X-Original[Complete] = IEEE[Exclusive]" is False

- The original meaning of {Complete|Incomplete} is discussed here.
- It is suggested that IDEF1X{Complete} is "equal to" the IEEE{Exclusive} definition, which is one clasification of the previously established and commonly used IEEE Subtype concept and notation (instead of IDEF1X). The suggestion is refuted:
- 1 such a supposition is based on the fact that *both* IDEF1X{Complete|Incomplete} classifications are mutually exclusive
 - 2 IDEF1X fails to evaluate the set of Categories that is *not* mutually exclusive, let alone provide a definition and method, while IEEE does
 - as evidenced, the IDEF1X notion of [mutually exclusive] excludes the required consideration of its counterpart [not mutually exclusive]
 - in fact [not mutually exclusive] is not even mentioned, let alone defined, in IDEF1X
 - while [not mutually exclusive] is properly defined and served in IEEE, its predecessor
 - therefore the definitions of [mutually exclusive] and [not mutually exclusive] in IDEF1X vs IEEE are radically different
 - therefore the definitions of IDEF1X{mutually exclusivelnot mutually exclusive} cannot be said to be equivalent to IEEE{mutually exclusivelnot mutually exclusive}
 - therefore the definition of IDEF1X{mutually exclusive} cannot be said to be equivalent to IEEE{mutually exclusive}
 - 3 a property gained by conscious intent and design, is not equivalent to a property obtained by accident, by privation
a person who lives on a desert island with no animals to eat, is not a [vegetarian], because that classification is gained, by intent, and with the availability of meat, it is not obtained by accident
 - 4 a "default" or default position can only be postulated in denial of the above facts. Such a postulation is rejected outright.

4.7 IDEF1X-Common[Complete] is Not Complete

As detailed, the original meaning of {Complete|Incomplete} is lost ⁶, the commonly understood meaning is discussed here.



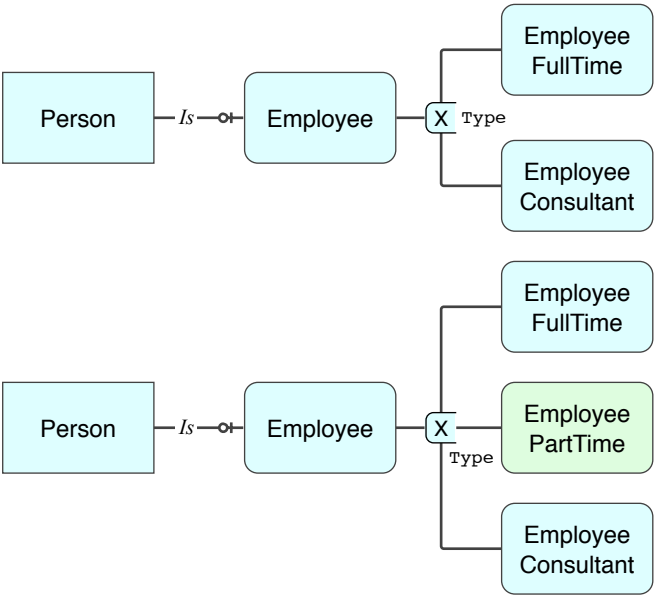
A "**Complete**" Subtype cluster.

Person = Fact of existence of a person
Employee = Fact of employment of a Person
Employee{FT,C} = Fact of type of employment

- Accountant said all Employees must be accounted. Oops.
 - Previous "Complete" designation is *de facto* false.
 - This one will be, as per the act of its creation.
 - The notion of "Complete" cannot be implemented (prevent data modeller from adding Subtypes).
 - Thus the IDEF1X notion of "Complete" is false.
 - Thus it is rejected outright.
- Person = Fact of existence of a person
Employee = Fact of employment of a Person
Employee{FT,PT,C} = Fact of type of employment

4.8 "IDEF1X-Common[Complete] = IEEE[Exclusive]" is False

As detailed, the original meaning of {Complete|Incomplete} is lost ⁶, the commonly understood meaning is discussed here.



An **Exclusive** Subtype cluster. An example in full context. with Predicates in text. is in §3.

Person = Fact of existence of a person
Employee = Fact of employment of a Person
Employee{FT,C} = Fact of type of employment

- Accountant said all Employees must be accounted. Sustained, without changing extant definitions.
 - IEEE[Exclusive] can be extended without deeming the past false, without the act of addition proving the notion of "Complete" false
 - IDEF1X[Complete] cannot be extended, such an act makes it absolutely "**Incomplete**"
 - Thus the notion that IDEF1X["Complete"] is equivalent to IEEE[Exclusive] is false.
- Person = Fact of existence of a person
Employee = Fact of employment of a Person
Employee{FT,PT,C} = Fact of type of employment

4.9 IEEE[Non-Exclusive] is Undefined in IDEF1X

The pre-existing IEEE Subtype classification defined both *Exclusive* and *Non-Exclusive* (incorrectly called *Inclusive* in the *ERwin* Methods Guide). The IEEE concept and notation were well-established long before IDEF1X was published. Its failure to define that well-known concept and notation is not a small error; it is a severe lack. But that lack is not in itself a singular error, it is one in th emidst of a large error, that of failing to define concepts for Subtyping; for the treatment of Sets, which are integral to the *Relational Model*.

Of course, most people continued to use the IEEE classification, which complies with the *RM* without change. And *ERwin* provided it.

4.10 IDEF1X-Original{ Complete | Incomplete } Cannot be Used

The original meaning of IDEF1X[Incomplete] is discussed here.

- Refer to §4.5 for a definition of this classification, before entering into this discussion.
- A simpler example is used in §4.1 to 4.4 to discuss a simpler issue, wherein it is given a full treatment, which needs to be understood before entering into this. The "three-entity" proposal has several errors, eg. requires *three facts in one place*, it is refuted.

Under the title [**Incomplete Categorisation and Non-Exclusive Child Entities**], the proposal contains a Generic with a single Category, a Basetype with a single Subtype⁶.

- The title is incoherent: [Incomplete Categorisation] is an IDEF1X concept (separately, an illegal one), [Non-Exclusive] is an IEEE concept, and the two have no relation whatsoever to each other. (If a relation is suggested, it must be defined.) As a consequence, the content under the title can be nothing but confused
- The notion of a Generic with a single Category is a gross Normalisation error.
 - it is treated in full in the Subtype document
 - the issue is also treated in §3.1, and again rejected
 - it is identified in a different context in §4.5, and again rejected.
- The notion of a Generic with a single Category is then used, together with another of its demented kind, to form a justification for yet another proposal which is a *relation* (still not defined) and a symbol (clearly defined but premature)
 - the justification is invalid because the basis is invalid, refuted
 - since the justification relies upon the IDEF1X[Incomplete] notion, it needs to be appreciated that that latter notion is invalid, because it breaks the definition of a Subtype [§3.1]. Hence it cannot be used for anything
 - two instances of an invalid notion do not make the notion valid
 - the relation proposed needs definition. In any case, it cannot be defined because it is invalid

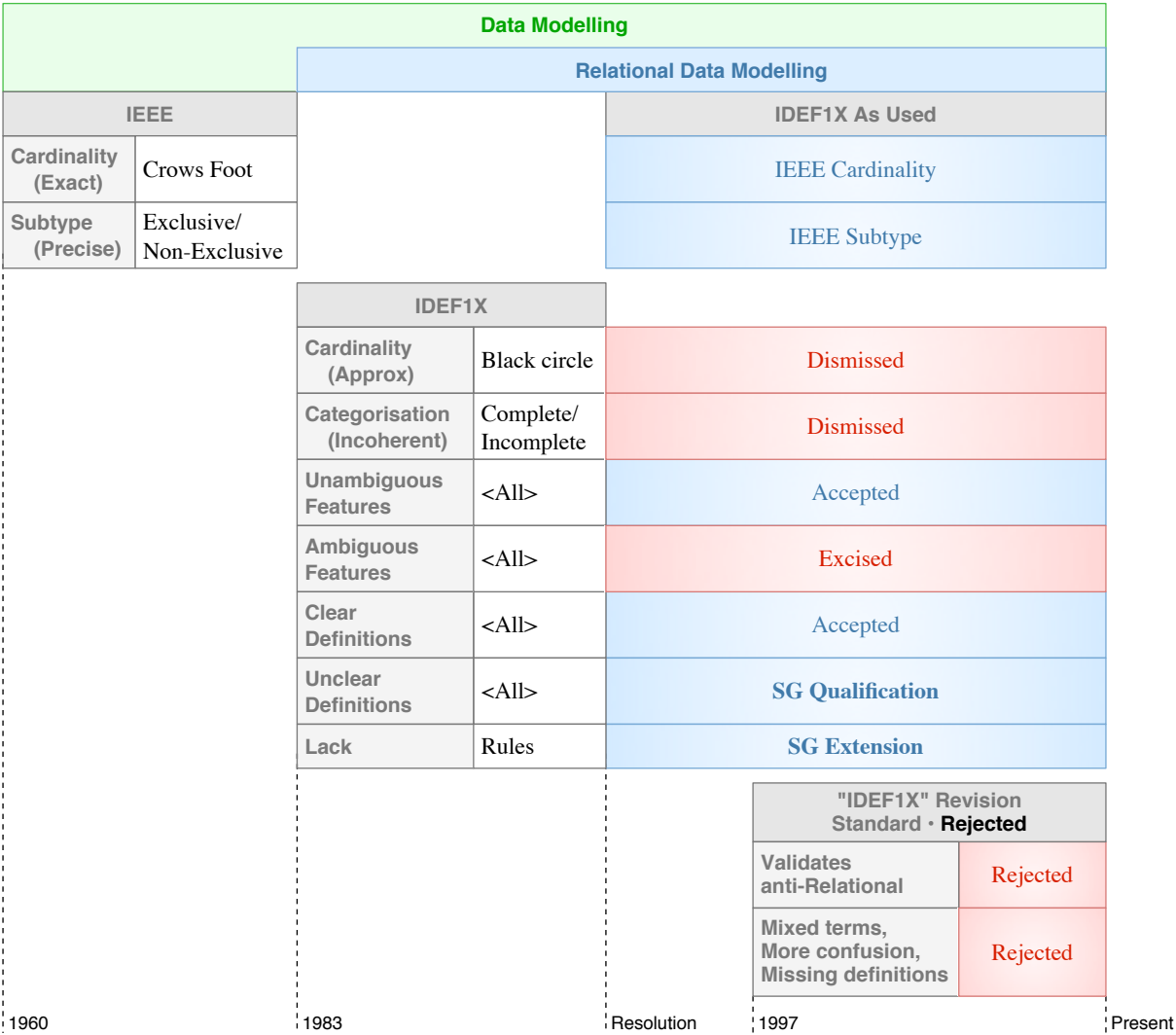
Thus the entire proposal under the title [Incomplete Categorisation and Non-Exclusive Child Entities] is rejected. There is no point in further definition of the proposal, because the basis is invalid, such will predictably suffer the same fate.

⁶ I will not reproduce an error, and thus afford it some credibility.

5 Notation {Progression|Regression}

5.1 Appearance of Related Standards

The chronological order (left to right) of related conventions and standards. Simplified, for discussion:



Software Gems' assets that are relevant to IDEF1X are shown. Standards that apply to the implementation (eg. **SG Relational Data Modelling Standard**; **SG OLTP Standard**; etc), which have commercial value, are not shown here.

5 Notation {Progression|Regression}

5.2 IEEE vs IDEF1X Comparison, SG Resolution

After becoming technically familiar with the previous sections, the following comparison may be contemplated, and thus the resolution taken by Software Gems, in their 1993 papers (above). And of course promulgated, over the last thirty years.

IEEE			IDEF1X	Determination	Used
Subtype	Exclusive	1::1	Not Defined	Retained	Retained
	Non-Exclusive	1::1-to-n		Retained	Retained
Optional Attribute		1::0-to-1		Retained	Retained
Category (Exclusive)	Complete	Complete-1	1::1	Redundant	
		Complete-2	1::1	Irrelevant	
	Incomplete		1::0-to-1	Invalid	
<div><div>Caveat • IDEF1X Terms IDEF1X <i>Complete</i> has two meanings, but <i>Incomplete</i> needs to be understood first: <i>Incomplete</i> A Basetype without a Subtype is permitted <i>Complete-1</i> All Basetypes have at least one Subtype <i>Complete-2</i> The set of Subtypes is complete (English, common understanding).</div><div>This is neither a full definition, nor exhaustive list of issues re IDEF1X, it covers only the subject matter in this paper. The rest are given summary treatment.</div></div>					
Relational Model	Relational Features			Accepted	Accepted
	Anti-Relational Features			Rejected	
Relational Model <ul style="list-style-type: none">• <i>Incomplete</i> is prohibited• <i>Complete-1</i> is always true, the definition is redundant• The remaining meaning of Complete is <i>Complete-2</i>. But it is irrelevant.• IDEF1X does not provide a definition for <i>Non-Exclusive</i> Subtype (common in reality)• Thus the IDEF1X <i>classification</i> for Category is dismissed, and the IEEE concept & notation for <i>classification</i> of Subtype is retained.					

Due to high-end customers requiring IDEF1X compliance, and the high-end practitioners' wide use of *ERwin*, the great Data Modelling tool (it squirts SQL), which provided IEEE notation, those with an IQ greater than 80 used it, such that it became the *de facto* standard. In general, the standard was written by an academic, it is chock-full of errors, quite useless to a practitioner, thus he had to resolve all issues to make useable. It is likely that that resolution was written up and standardised by a few of us.

The academics in the field of database science are strongly established as suppressing Dr E F Codd's *Relational Model*, and promoting their anti-Relational methods as "Relational". They have not heard about IEEE; or Subtypes for fifty years, or about IDEF1X; or its resolution; or *ERwin*, for forty years, because their contrived notion of "industry" excludes reality. Nicola's interest in Relational paradigm and actual industry use is therefore welcome.

5 Notation {Progression|Regression}

5.3 IEEE, IDEF1X, Proposal

- After becoming technically familiar with the previous sections, finally, a summary re the proposal of new definitions and symbols. This employs the content from the chart in Nicola's proposal, the latest version, arranged in order of appearance.
- the resolution of IDEF1X defects thirty years ago, rendered in pink (*apprehend that first*)
 - the resolution herein re Nicola's proposal, rendered in purple (*second, and not before the first*)
- ① the Category classification was rejected, because one member is invalid
- for it is not logically possible to reject one member of a classification but not the classification, because it is the classification that allowed the errant member, and not the member itself, that is defective
- ② The common classification (promoted by the *ERwin Methods Guide* p 67) was likewise rejected because it is non-standard; not a singular definition; and naïve. Nicola does not address that classification
- the resolution of all IDEF1X defects, not only the issue herein, was published to *Software Gems*' customers in 1993
- ③ the resolution herein re Nicola's proposal, rendered in purple (*third, and not before the first*)
- The arrows show the evident sequence of Nicola's progression. First, with disregard to ① and ②, as well as IEEE notation. Second, reverting to the original documents, without realising its poor state: when the source material is garbage, the processed material can be nothing but.
 - The error here is the standing one for academics in this space: abject ignorance of industry standards; and of real world knowledge and implementation, due to their contrived notion of "industry".

