1 Introduction

I have been answering a few database design Questions, and providing IDEF1X-compliant Data Models as part of that, but it appears some seekers are unfamiliar with the Standard and the notation. So they ask "What is IDEF1X ?" or "What does this symbol mean ?" and consequently we go off on a side trip. Here is a Normalised answer.

1.1 Scope

This document introduces IDEF1X and identifies the notation, so that standard-compliant Data Models can be read and fully understood. There are many closely related subjects, which require formal education; that is *not* provided here:

- the *Relational Model* [1]
- Normalisation [2] [3] in general, and the specific Relational Normalisation identified in the *Relational Model* [4]
- the IDEF1X Methodology or Relational database design

1.2 Layout

This document is referenced from multiple contexts. It is therefore laid out as follows:

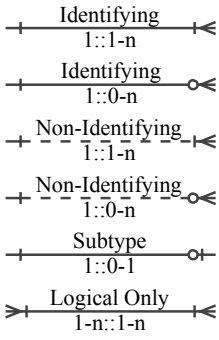
- the Notation, if that is all you need
- a reasonably detailed Introduction to the IDEF1X Standard
  The Data Model is not just about Notation; in order for it to be fully understood, a short introduction to the Standard and terminology is required. That is provided on next three pages, using an example; and the relevance of each item is discussed.
- Page 5 identifies Related articles
- my Extensions to the Standard, a stricter implementation still.

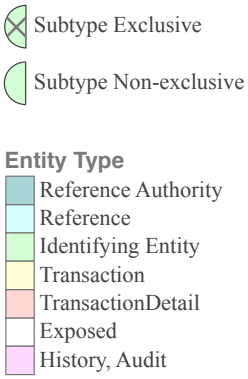
1.3 Notation

IDEF1X Method & Notation

Item	Definition	Display
Key		
Primary Key:	Unique	Above line
Alternate Key:	Other Unique key	AK
Inversion Entry:	Non-Unique index	IE
Entity		
Independent:	Exists without a parent	Square
Dependent:	Dependent on a parent	Round
Relation		
Identifying:	Parent PK forms child PK	Solid line
Non-Identifying:		Broken line
Parent PK is *always* a migrated FK in child		Bold

IEEE Relation Notation





1 The genuine article; not the hijacked and mangled article that is marketed by famous authors who have no evident knowledge of the *Relational Model*; not the fragmented"normal forms" produced by mathematicians who have no IT qualifications or capability . All my data models are examples of genuine Normalisation, as prescribed by Codd.

2 Note that the "normal forms" are merely abstract mathematical definitions, they do not provide a *method* to achieve either Normalisation, or the "normal forms" which they allege. They are about as relevant to Normalisation and the *Relational Model* as the knowledge of hair thickness on the tail of an elephant is relevant to the knowledge of elephants. Databases are concrete, not abstract. Thousands have wasted millions of hours trying to Normalise data using these absurd "definitions" that have nothing to do with the subject.

3 Duplication in any area, is something to be avoided at all costs. *Never Duplicate Anything* and *Don't Repeat Yourself*, are cute modern labels coined by those who are unaware that Normalisation is an architectural principle that has been established in the industry since the 1970's. Normalisation is not just a database design method. The abstract mathematical definitions called "normal form" bear little relation to it. Normalisation preceded the Relational paradigm.

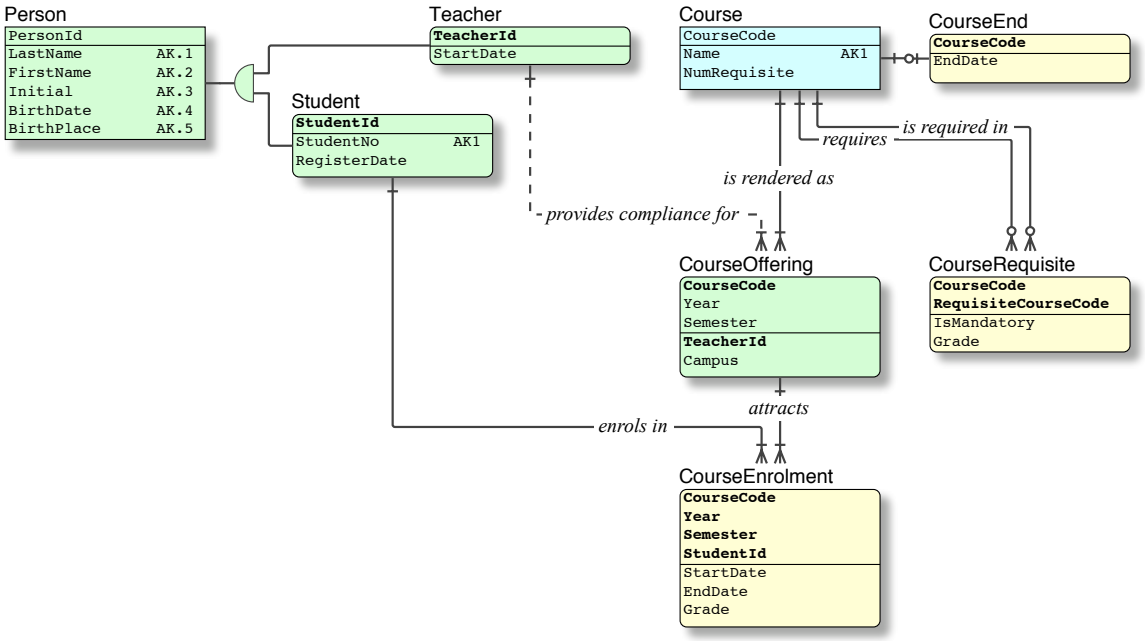
4 Codd progressed it and expressed a specific Normal Form as a minimum requirement for the Relational Model.

1.4 Context & Purpose

- IDEF1X stands for **I**ntegrated **Def**inition [5], and belongs to a set of six such definitions, each of which is tightly constructed and integrated with the others [6]. **1** is for Information Modelling, **1X** is for Relational Data Modelling.
- It is a stricter implementation of the ***Relational Model*** by Dr E F Codd. It is based on the theory and techniques invented by Codd (the RM) and Dr P P S Chen (the Entity Relation Model). The initial technique (methodology) was developed by R R Brown, T L Ramey and D S Coleman. It was progressed and developed to this diagrammatic form by Robert Brown in the early 1980s. I have used it exclusively since 1985.
- It is a **methodology**, specifically for modelling Relational databases.
- It was accepted and declared as a **standard** by National Institute of Standards & Technology (FIPS 184) in 1993. (IDEF0 may be used for Function Modelling.)
- It allows the data to be modelled:
  - in the **context** of the whole organisation
  - **independent** of the applications that use it
  - **completely** (exposing all complexities and subtleties).
  - using a **standard notation** (set of symbols) and nomenclature, so that all teams and users are served. This eliminates ambiguity, and ensures that the *complete* set of information (identified by the standard) is communicated in the *single* diagram.
- Competent use of the Methodology produces databases that are easy to change and extend, as well as easy to use, by users [7] and future applications. The result is a more genuinely Relational database, which allows expansion, rather than replacement due to the limitations otherwise [8].
- IDEF1X is widely accepted in organisations that require compliance to **Standards** (Government departments, US DoD, aircraft manufacturers, large banks). It is viewed as a mark of quality, a genuine Relational database.

2 Example

College Enrolment (Simplified)



5 It does not stand for ICAM, as that ever-changing study in mediocrity authored by the unqualified masses suggests at this present time. ICAM was indeed the first major, and ongoing, project to use it. It was first published under ICAM auspices, by the above authors.

6 Data models and applications which are developed in isolation, lacking integration, are of course in a state of dis-integration.

7 Unfortunately, these days many power users have more database skills than most developers.

8 If the database needs to be "re-factored", by definition, it is not a Relational database; it is merely a data container for object classes, devoid of the power, speed and integrity of a Relational database. Unlike Relational Databases, these data heaps need to be changed whenever the object classes change.

3 Element

Entity

- **Independent** entities exist on their own, they have square corners.
- **Dependent** entities exist only in the context of an Independent parent; they have round corners.

Key

According to the RM, a **Key** is a single or multiple columns *of data* that uniquely identify a row, they are called **Identifiers**. Therefore `IDENTITY` values, GIDs, IDs, are not Keys, and they do not replace Keys.

•A Key is equivalent to the entity and the entity is equivalent to the Key; it is important to understand this from the first step of modelling. The Person Key is the Person

- The **Primary Key** of each entity is above the line, in the prescribed order; the other Attributes are below.
- In the early modelling stages there may be several **Candidate Keys** (unique) for a table; as the model progresses to resolution, one of those is chosen as Primary; since the election is over, the remainder are no longer Candidates, they are **Alternate Keys**, shown as **AK**. Legal numbering is used to identify the separate indices, and the sequence of columns within each index.
- Of course, Primary Keys and Alternate Keys are `UNIQUE` and `NOT NULL`.
- Non-Unique indices which allegedly assist searches, are called **Inversion Entries**, shown as **IE**. Since such access yields more than one row, it is not a Key [9].

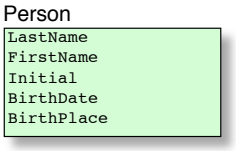
Relational Key

Consistent with the Relational Model, IDEF1X promotes the concept of **Relational Keys** [10] (strong, natural, meaningful keys, that maintain context in the Data Hierarchy), and promotes their articulation in the resulting database.

- The methodology uses a formal approach to Identifiers, as prescribed by Codd, thus they are elevated; their meaning is preserved in the model; and Relational power is retained in the database .
- Relational Keys are made from the data; they are a natural product of genuine Normalisation.
- Compound or composite Keys (multiple columns) are ordinary fare in a Relational database, get used to it.

Surrogate

At some earlier stage of modelling, the Person table had the following structure:



The Primary Key is a good, natural, unique Relational Key or Identifier, and it cannot be dropped. When a table has many children, or *levels* or children, *and* the Primary Key is large (in bytes, not in number of columns), it is sensible to use a short identifier. Thus `PersonId` is added; it takes the place of the Primary Key; and the original Primary Key is deemed an Alternate Key (refer to the example above).

- By the definition of Key in the *RM*, it is not a Key by any means (it is not based on data; it is invisible to he user; etc); it is a Record Number, not a row iIdentifier. Commonly an `IDENTITY` column or `GID`,
- The term "surrogate key" is therefore grossly incorrect, but it is commonly used and commonly misunderstood. People naturally expect some of the qualities of a Key, and it has none. Incorrect naming guarantees confusion.
- As per the *RM*, the rows in a Relational table must be unique; that function requires a Key, and that Key cannot be removed. The RecordId cannot provide uniqueness of rows.
- RecordIds are physical row locators, they are not logical Keys.
- Therefore the RecordId such as `PersonId` is always an *additional* Key and index, causing additional overhead.
- RecordIds should be contemplated *after* Normalisation, which produces Relational Keys, one of which is the Primary Key, the position of which is usurped (not replaced); and not before [11].
- When a RecordId is implemented, the Relational (join) power between the table and all its children, and all the tables above the subject table in the Data Hierarchy, is lost. It breaks the **Access Path Independence Rule** in the RM, and therefore code relating the tables below the subject table, to tables above, must access the subject table to pick up the logical Keys that were lost. Therefore they must be chosen carefully, and they must be minimised [12].

---

9 The term "non-unique key" is a contradiction; a key is an unique identifier; a non-unique unique identifier is an absurdity.

10 That is a subject in itself, beyond the scope of this document. It is explicitly detailed in the *Relational Model*. The Relational Key is a requirement, not a desire; without Relational Keys, by definition, the data is not a Relational database. Without Relational Keys the Relational power, and performance is lost. Further, contrary to popular belief, many more indices and far more joins are required.

11 Starting a data modelling exercise with surrogate keys on every perceived entity cripples the exercise, and eliminates the possibility of genuine Normalisation; it is a spreadsheet view of the universe, a developer's scratchpad, not a genuine analysis of the data. Such unnormalised data heaps then require (a) "de-normalisation", and (b) many more indices than Normalised databases do.

12 In addition to crippling the modelling exercise, the use of surrogate keys on every table results in (c) eliminating the Relational (join) power between *every pair* of related tables. The resulting data heap is not Relational in any aspect; it requires (d) more joins (not less as the mythology suggests). There is no substitute for authentic Normalisation by a qualified modeller.

Relation

- A child entity is *related* to a parent entity by migrating the Primary Key of the parent as a **Foreign Key** (reference) in the child; they are shown in bold font [13]. Other modellers may show it as "FK".
  - A Nullable Foreign Key or "optional parent", although permitted in IDEF1X, is a gross Normalisation error [14], which is easily eliminated by correct Normalisation. It is not allowed quality databases.
- There are two types of Relations, the difference is important:
  - **Identifying** Relations (solid lines) are those where the parent Primary Key is used to form the child PK; this is the essence of good Identifiers (Relational Keys). This continues the Data Hierarchy or context of the parent.
  - **Non-Identifying** Relations (broken lines) are those where they do not. The Data Hierarchy or context is discontinued.
- The *RM* and IDEF1X encourage the use of **Role Names** such as `StudentId` and `TeacherId`, which reflect the table name, or the restricted scope (domain) of the parent PK (`PersonId`). This adds meaning and further clarifies the model [15].
- The notation for Relations in IDEF1X is quite different to IEEE (crows feet), and it takes a bit of getting used to. Since the latter is more universally understood, it is often used instead [16].

Associative Table

- In the logical model, **many-to-many** Relations are acceptable, and they are drawn as Relations.
- In the physical model, these many-to-many Relations are implemented as **Associative Tables**, with a one-to-many Relation from each parent. The table consists *only* of the FKs to each parent; which forms its PK, and eliminates duplicates very nicely [17].
  - All many-to-many tables are not Associative tables. `CourseEnrolment` is not an Associative table, as it contains additional `columns`; it is an ordinary table, visible in the logical model.
  - It does however, reflect the fact that the `Student::CourseOffering` Relation is many-to-many. In some early stage of the model, it would have been a Relation, which was the association identified at the time, not a table. When the `Student::CourseOffering` columns were identified, and its location determined, the Relation was rendered as a table.

Verb Phrase

The Relations comprise some of the more fundamental Business Rules in the Data Model.

- The words on the Relations are called the **Verb Phrase**, it gives *meaning* to the Relation.
  - The entities are the *subjects* in the database, and the Relations are the *actions* that take place between the subjects, expressed as verbs. Hence the name Verb Phrase
  - It should be read from the parent to the child, as in: `Course is rendered as CourseOffering` and Every `Teacher provides compliance for zero-or-many CourseOfferings`.
- The **Cardinality** (zero, one, many) at the child end of the Relation indicates the singular or plural form.
- The Relation from the child to the parent can be identified by simply reversing the Verb Phrase (using sensible English, not literal reversal): `CourseOffering is the rendering of one Course`.
- When reading the Relations, the many-to-many tables between Identifying entities should be ignored; the Verb Phrase applies to the entity at the other end of the association (the original many-to-many Relation).
  - Read the Verb Phrase from one parent [through the many-to-many table] to the parent at the other end, as in:
    `Student enrols in CourseOffering`
    `Every CourseOffering attracts one-or-many Students` (`CourseEnrolment` is read through).

13 There is nothing in the parent that references the child. And implementing something that does, is a circular reference, a gross error.

14 There are additional data and referential integrity violations. Normalisation prevents insanity.

15 For a more complex example, refer to the Advanced Data Model.

16 An amendment where nothing is lost and something is gained, is an improvement. This amendment is accepted among IDEF1X practitioners. (If you suffer from the simplistic, black-or-white thinking that suggests such amendments are "non-compliant", note that condition prevents you from designing Relational databases, and this document is not for you.)

17 An additional Id[iot] key, which is non-relational, and its index, are 100% redundant. Likewise, failure to identify that the two Foreign Keys are unqiue, will produce duplicates.

Subtypes are required in almost every database, but unfortunately they are not understood, and thus rarely implemented; hopefully there is enough information here to improve your databases. The several possible Subtype structures can be rendered with precision and completeness in IDEF1X, in which they are officially called **Categories**. However, since the later IEEE notation provides even more precision; it is used here [16].

- The semicircle shows the *Relation* is a Subtype (one connection for the generic Type; one for each Subtype). If the symbol has an X through it, it is an Exclusive structure, otherwise it is Non-exclusive ('inclusive' has a different meaning).
- The Category cluster (everything connected to the semicircle) should be viewed as a single unit; this is the essence of the concept. The generic Type cannot exist without at least one Subtype.
- For each Subtype, there can be at most one row belonging to the generic Type.
- The Primary Key for every Subtype is the Primary key for the generic Type. Role names are called for here.

Exclusive Subtype

- For each generic Type, there must be one, and only one Subtype [18]. Refer to the Exclusive Subtype example.
- A mandatory **Discriminator** column, which is located in the generic Type, identifies which Subtype it is.
- The Verb Phrase is always: `Is One Of`

Non-exclusive Subtype

- For each generic Type, there must be at least one, and possibly more Subtypes [18]. Refer to page 2.
- There is no Discriminator.
- The Verb Phrase is always: `Is Any Of`

- Note the difference between a Subtype (`Student`) and an optional column table (`CourseEnd`).
- Both Subtypes and optional column tables are the product of authentic Normalisation. Thus they:
  - eliminate Nulls in the database
  - support minimal and correct indexing: if `Student` and `Teacher` were not Normalised as Subtypes, their attributes would be located in `Person`; `StudentNo` would be Nullable, and its index would not be unique.

4 Related

- Generally, a good modelling tool is required to produce IDEF1X Data Models: each such tool allows you to exercise the Methodology; implements the Standard to varying degrees; and has its own features & impediments.
  - *ERwin* is the only modelling tool that faithfully implements the standard, and allows a single integrated model.
- However, for those who understand the methodology and who can apply it without the assistance and constraint of a modelling tool, only a simple drawing tool is required. If you have an Apple Macintosh with *OmniGraffle*, you can erect IDEF1X-compliant Data Models quite easily, using my IDEF1X Stencil.
- A more complex IDEF1X-compliant Advanced Data Model, comprising several pages, and demonstrating the Extensions, may be of interest. The document is fully enabled as a PDF: click on any shaded object to display its detail.

5 Extension

I am often asked, exactly what is it that I do, to make my Data Models so easy to comprehend. I implement three features in all my Data Models, which are technically outside the IDEF1X standard*. They are a result of decades of implementing genuine Relational databases (according to the* Relational Model *in its original un-perverted form) and my studies in cognitive science. I have also been influenced by the famous* Apple Style Guide*, since 1984.*

5.1 Hierarchical Layout

The Data Hierarchy is fundamental to the *Relational Model* [19]. In fact, Codd's Relational Normalisation demands that the data is Normalised into a valid Data Hierarchy *first*, as a pre-requisite, before it can be Normalised *relationally* [20]. The Relational Key or Identifier is a product of such Normalisation.

18 Such rules are implemented as Declarative Constraints in SQL in the database, and enforced by the server. The more detailed examples include full DDL and code. For Non-exclusive Subtypes, the permitted combinations can be declared (expand the code provided).

19 It is noted that while practitioners with genuine knowledge of the *Relational Model*, and the IDEF1X Standard, maintain and support concepts that are central to the *RM*, many of the authors who have written books about the *RM* are evidently quite unaware of these central concepts. Therefore, the great body of material written "about the *RM"*, has very little to do with the genuine *RM*. The result is, the non-*RM* that is marketed is known as the *RM* (a crime of commission, fraud), and people are robbed of the benefit of the real *RM* (a separate crime of omission). Much like SQL is commonly known and judged by the non-SQLs, which are *not* SQL.

20 A number of important Relational concepts and requirements, such as the Data Hierarchy; Hierarchical Normalisation; and Relational Normalisation are absent from the books that allege to be about the *Relational Model*. The mathematicians who define the unrelated and abstract "normal forms" have not figured it out in the forty three years that have passed.

*Data objects, indeed, all objects in the universe, belong to the **Natural Hierarchy**; objects that are presented in this hierarchy are easy to understand. Conversely, objects that are presented without order, or worse, in reverse order, are difficult to understand and to* accept*, because it violates the natural order in ones mind. This extension substantially improves comprehension of the model by both technical teams and non-technical users. Additionally, it exposes ambiguities re entities and their Relations (re Codd's Normalisation) that the modeller must resolve, that would otherwise remain undetected.*

5.1 Hierarchical Layout (continued)

To be clear, the Data Hierarchy is an essential part of the data modelling, and particularly of the *RM*, it is not an extension; the extension is the diagramming layout guidelines, that (a) renders the Data Hierarchy explicitly in, and (b) provides increased meaning and cognition of, the data model, further realising the spirit and intent of the *RM* and the IDEF1X Standard.

- The Relational, Standard, concept of Identifiers is implemented positionally, vertically; the dependencies are plain.
  - Child tables (Identifying Relations) are positioned directly below their parents.
  - Referencing tables (Non-identifying Relations) are positioned below the referenced table, with an offset; not directly below it.
- Subtypes are positioned to the right of the generic Type, indicating that they are on the same level in the Data Hierarchy, and emphasising its one-to-zero-or-one Relation.
- Optional column tables are likewise positioned to the right of the parent entity.
- This naturally produces a model in which:
  - the Data Hierarchy is visually explicit
  - the progression of the Relational Key, is plainly visible
  - the diagram, the entities in the model, are naturally top-to-bottom, left-to-right
  - each entity is positioned in the hierarchical context of every other entity [21], and can be evaluated as such
  - The controlling tables (reference, look-up), which are also the smallest (in terms of rows), appear at the top of the model; the Identifying entities appear in the middle stratum; and the transactional or transaction detail tables, which are also the largest, appear at the bottom.

5.2 Concision

The goal is to maintain completeness while preventing information overload. This is not really an extension to the Standard, rather, it is a presentation method for the Data Model.

The human mind can process only so much information at a time. If this limit is exceeded, the mind rejects the entire *subject matter, and works around it. If the diagram is too large; or there are too many objects on the page; or there is too much complexity in a small diagram, the mind rejects it. This is not conducive to the successful communication of the model to others, thus it impedes the success of the database.*

- Therefore the model is drawn in sections that are easy to understand, clusters of closely related tables, officially called **Subject Areas**.
- Because printing, electronic distribution and binding are additional considerations, each Subject Area is limited to an A4 or US Letter page. Thus large models are drawn on several pages, rather than on a single large diagram.
- The first page is an Entity Relation level diagram (sans Attributes); it introduces the entire model; the position of the tables in relation to other tables; and all Relations.
- Each subsequent page contains a Subject Area, rendered at the Attribute/Column level [22].
- The tables in each Subject Area page are presented in exactly the same relative position as they appear on the first page, the ERD.

5.3 Collapsed Entity

Dividing the model into Subject Areas improves understanding and facilitates document production, but it creates a new problem: since a table *must* be shown wherever it is referenced as a parent, the Subject Area pages now contain duplicate instances of such tables (which a single large diagram does not) [3]. The solution is simple enough:

- draw the table, with its full definition, in its home Subject Area, once
- use a Collapsed Entity symbol wherever it is used as a reference.

21 White space is your friend, just ask any Art student. The empty space on each page allows entities to be added without having to repeat the entire layout process. Diagrams that squeeze the entities into some arbitrary rectangle; or that invert parent-child relations, cannot be reasonably expected to be understood, or to attract approval.

22 For the level of presentation that is relevant to my posts, the Domain/Datatype is generally not required.

