Overview

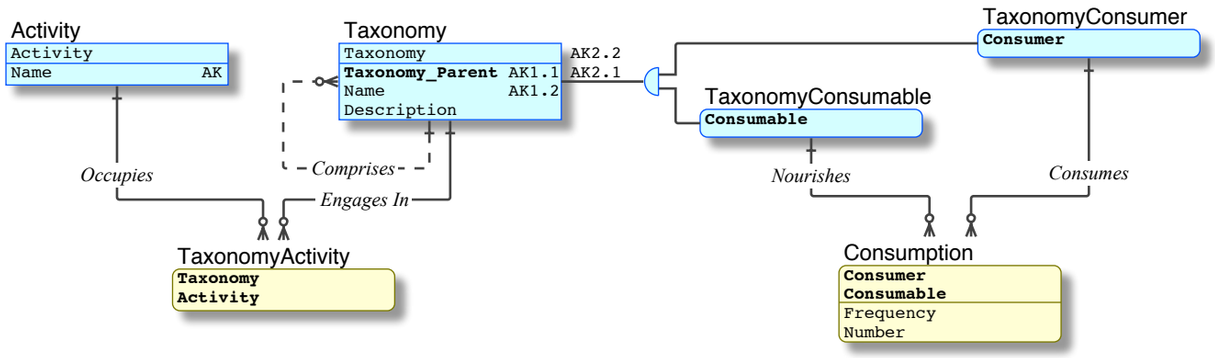
A Predicate is simply a single sentence in precise technical English, that evaluates to true or false. A query is simply the trial of a Predicate or a number of them chained together. This document identifies all Predicates that are relevant to the modelling of Relational Databases (as opposed to the querying thereof), and illustrates how to determine them from an IDEF1X data model.

In consideration of the fact that the *Relational Model* is based on First Order Logic, the entire database is:

defined in the form of Predicates.

Thus an implementation is an implementation of those Predicates. Hence the very real possibility of DKNF (the Codd intention per the *RM*, not the deranged and fragmented mathematical definition).

While that is true, it is a theoretical understanding, and we do not normally discuss the database, or Predicates, in those terms. All Constraints on the data are Predicates, not only those that use the `CONSTRAINT` keyword. These are usually discussed as **Constraints**, rather than as Predicates, with the remainder being discussed as Predicates. For clarity, all Predicates are shown here, along with their implementation, and their rendition in an IDEF1X model.



IDEF1X Notation

Predicate Type	Example (Predicate Declaration)	Implementation	IDEF1X Notation
Existence Independent	`Activity is independent`	(No differentiation) `CREATE TABLE`	Square corners
Existence Dependent	`Consumption is dependent on TaxonomyConsumable`		Round corners

Existence Basetype	`Taxonomy is a non-exclusive basetype`
	`  Taxonomy is any of { Consumer | Consumable }`
	`Person is an exclusive basetype`
	`  Person is one of { Male | Female }`
		(No differentiation)
		`CREATE TABLE`
			Semicircle with X (Exclusive) or without (Non-exclusive).
			Solid line.
			Parent PK is child PK.
			Cardinality 1::1 is not shown
			Verb Phrase is always `Is`

Existence Subtype	`TaxonomyConsumable is a non-exclusive subtype of Taxonomy`
	`TaxonomyConsumer is a non-exclusive subtype of Taxonomy`
	`PersonMale is an exclusive subtype of Person`
	`PersonFemale is an exclusive subtype of Person`		

Identification	`Activity is primarily  identified by ( Activity )`
	`Activity is alternately identified by ( Name )`
	`Consumption is identified by ( Consumer, Consumable )`
		`CONSTRAINT PRIMARY KEY`
		`CONSTRAINT UNIQUE`
		`CREATE UNIQUE INDEX`
			Primary Key [1]: above the line
			Alternate Key: `AK`

Relationship Forward	`Activity occupies 0-to-n TaxonomyActivities` `Taxonomy engages in 0-to-n TaxonomyActivities`	`CONSTRAINT FOREIGN KEY`	Line and Verb Phrase
Relationship Converse	`TaxonomyActivity is an occupation of 1 Activity` `TaxonomyActivity is an engagement of 1 Taxonomy`		Line and Verb Phrase *read in converse*
Relationship Identifying	`Consumption is identified by TaxonomyConsumable` `Consumption is identified by TaxonomyConsumer`	(The difference lies in the Primary Key of the child) `CONSTRAINT FOREIGN KEY`	Solid line. Parent PK is used to form the child PK
Relationship Non-identifying	(None, just the absence of the Identifying declaration)		Dashed line. Parent PK is an attribute in the child.
Descriptor [2] [3]	`Consumption is described by ( Frequency, Number )`	Column definition	Attributes are below the line.

There are, of course, additional constraints of various kinds, that are not depicted in the diagrammatic model. High end practitioners depict these as simple Extensions to IDEF1X, to afford a model that is complete. Likewise, Functions are exposed.

Check Intra-table	`CHECK Salary > 1000`	`CONSTRAINT CHECK`	(C) `Name_IsValid_ck`
Check Extra-table	`CHECK ( dbo.Taxonomy_IsLeaf_fn (Taxonomy ) = 1 )`	`CONSTRAINT CHECK` `CREATE FUNCTION`	(C) `TaxonomyConsumer_IsLeaf_ck` (F) `Taxonomy_IsLeaf_fn`

For databases that conform to either **Open Architecture Standards** or **OLTP Standards**, all writes are performed via a set of exposed SQL ACID Transactions, and direct writes to the tables are not `GRANTed`. Such Transactions stand as declared Constraints, but they are not Predicates.

Transaction Extra-table	`BEGIN TRAN … DML … COMMIT TRAN`	`CREATE PROC` `CREATE FUNCTION`	(T) `Taxonomy_Add_tr`

1. The *Relational Model* demands a Primary Key, which is migrated as a Foreign Key, the remainder are Alternate Keys. "Candidate Key" is not merely a breach of the *RM*, it is a rebellion against it. Those who use it implement physical record IDs as "primary keys", ensuring pre-1970 Record Filing Systems, which have none of the integrity, power, or speed, of a Relational Database. It also demonstrates the denial (schizophrenia) of "theoreticians", in that they insist on using "logical" terms for physical objects, thus denying their physical considerations. Further, in the early stages of modelling, when Keys are determined, once one of the candidates has been elected as Primary, the remainder are no longer candidates. Thus they maintain a fantasy of a "logical" perspective over a physical database.
2. The Keys are the Facts in a Relational Database. Descriptors describe the Facts, the Keys. Of the Predicates, Descriptors are the least relevant, as they do not define the structure of the database, as the others do, and they are clearly visible in the model. The syntax is loose, eg. they may be stated in singular or converse form.
3. The "theoreticians" do not understand Predicates, they "discovered" a small fraction of what we have been using since 1984, only recently. Further, in their typical schizophrenic style (denial of facts, of science), they ignore the relevant Predicates (blue cells), and obsess about the most irrelevant ones (pink cell).

Copyright © 2015 Software Gems Pty Ltd

Usage

Importance

Over and above their theoretical importance, Predicates [4] constitute an important tool in the data modelling exercise. While modelling the data, it is important to declare the Predicates that the model implements, and to evaluate them as such. This in turn allows the model to be evaluated, from the perspective of the Predicates. The precision and veracity of the declarations can be evaluated. This exposes any issues in the model, and identifies any changes that are required for the next iteration of the model. Thus the Predicates form an important feedback loop that elevates the exercise.

Formal Declaration

1. For technically qualified readers, the Predicates can be 'read' directly from the IDEF1X data model, ie. the notation in the Model is clear and definitive [5]. Formal declarations are not required.
2. For users, since the data model constitutes a communication platform between the modeller and the user community, for the very same reasons above, it is essential that the users can understand it, and verify it. Therefore any model that is presented to the users should include the Predicates, as formal declarations. Here only the important Predicates need to be declared, the Descriptors are plainly obvious in the model.
3. Likewise, when the audience consists of novices to Relational Databases, or to the data modelling exercise, or to IDEF1X, I include formal declarations of the Predicates in the data model.

Table is Not a Predicate

As detailed above, they are many types of Predicates in a database, as prescribed in the *Relational Model*. Unfortunately, by virtue of the evidence produced since its advent, the "theoreticians" [6] who allege to be serving the Relational Database industry have little or no understanding of the *RM* or of Predicates. The latest in their series of insane fragments, posed as usual in isolation and in denial of other facts (in typical schizophrenic style), is their proposal that a table should be named for the "predicate" that it "represents". Although the postulation is ridiculous to qualified people, novices believe it, and attempt to apply it, an act which results much confusion [7] [8]. Thus it requires address here.

- A table does not "represent" a "predicate" or Predicate. First, a table is an implementation, not a "representation". Second, as detailed on the previous page, a table is an implementation of *many* Predicates, and of different *types* of Predicates.
- There is no single "predicate" or Predicate that a table "represents".
- Due to their obsession with "functional dependencies", the "theoreticians" are aware only of the Predicates that are apparent from that stupefying exercise, and even that, only in a fragmented theoretical sense. These happen to be the least relevant Predicates, the Descriptors, they are completely unaware of the relevant ones, and of the entire set of Predicates. Thus the "predicates" that they evaluate for a table, that they postulate as "representing" a table, are the least relevant, the least applicable.
- Further, since each Fact usually has many Descriptors, one is tricked into the typical activity of the insane: choose *one* of the many Descriptors, knowing from the outset that none of them are relevant or applicable.
- The result is, instead of table names such as `Student`, they advise `StudentIsCalled`, or `StudentLivesAt`.

A table is an implementation of the many Predicates that apply, it does not "represent" any single Predicate. A table should be named for the rows that it contains, in the singular.

For a further comparison on this subject, visit **Predicate vs Table**.

File Fails Predicate

In a discourse regarding Predicates there is one final point to be made, noting that these are dark days for the industry, when the "teachers" are marketing pre-1970's Record Filing Systems as "relational". In contrast to Relational tables, which are the subject of Predicates, files in an RFS have no Predicates, First Order Logic is not used, no genuine modelling is performed. The Relational integrity, power, and speed that Relational Databases have, is lost, or worse, is not known as a post-1970 possibility. Specifically, the following Predicates are absent:

- **Existence Independent/Dependent:** All files are independent; dependencies and Relational Keys are not implemented. This is a breach of the Access Path Independence rule prescribed in the *Relational Model*, which results in fixed navigation and more (not the mythological less) joins.
- **Existence Basetype/Subtype:** Completely unknown, not implemented. `OR` gates and `XOR` gates are an alien species.
- **Identification:** Relational identifiers (which are logical) are not known, not used. The files contain records (not rows), and they are accessed by physical Record Id. Further, the records are often not unique, this constitutes another breach of the *RM*.
- **Relationship Forward/Converse:** Relationships are in fact physical links between records, not relationships between logical Keys. Thus the Predicates and Verb Phrases do not apply, they do not have a similar or parallel meaning.
- **Relationship Identifying/Non-identifying:** Since Relational Keys are not implemented, all relationships Non-identifying. The Relational context, specifically Extension, is not implemented.
- **Descriptor:** Since the files are in fact spreadsheet views of the data (ie. flattened; derived relations, not base relations) and normalisation is absent, and since the notion of a database being a collection of Facts is unknown, the descriptor fields are merely located by association, not by Descriptors of a Fact. That is to say, they do obsess about the Descriptors, but the Descriptors they obsess about, are incorrect, located in the wrong files.

4. The "theoreticians" do not understand Predicates, their relevance, let alone their usage in the modelling exercise. They know of only the irrelevant Predicates, the relevant Predicates, and their practical value, is lost to them.
5. Separate to the fact that UML is not a standard; that is contains a million notations; that each designer "does his own thing", it is grossly inadequate for use in defining a Relational Database. It simply does not have the richness or the notation that the standard for modelling Relational Databases has. None of the Predicates are articulated in the model.
6. Beware of the "theoreticians" who allege that they serve this industry, the Relational Database space. In forty five years, they have produced precisely nothing that relates to the industry or to the *Relational Model*, there has been no progress since Dr E F Codd. More importantly, absolutely everything that they *have* produced is a regression to pre-1970s ISAM Record Filing Systems, which they justify in terms of "mathematical definitions", and market heavily. Such works are fraudulently labelled as "relational".
7. That confusion is precisely the goal of the insane, to infect humans with their insanity. Thus they obtain the validation that they desperately seek.
8. Note that in forty five years, they have not been able to make the distinction between base relations (tables) and derived relations (views), and that each has different requirements. They then try to "normalise" derived relations, which by definition, are flattened (de-normalised). There is no end to the insanity of the deranged.

Copyright © 2015 Software Gems Pty Ltd