Reply to Sólmundur Jónsson

Sólmundur Jónsson writes:

The presentation by Hugh Darwen, "How to Handle Missing Information Without Using Nulls" is very interesting and I can see that 6NF can solve some problems. But with all these tables aren't we getting very close to [violating] Codd's Information Principle "All information in the database must be cast explicitly in terms of values in tables and in no other way"? Especially by [having a] separate table for each meaning of the Nulls? Aren't we hiding away information in table names?

Hugh Darwen and Chris Date reply:

Either we have relvar names or we don't. *The Third Manifesto* requires relation variables. It doesn't actually state that a variable must have a name, but all of our illustrative examples are based on an assumption that every variable has a name that is unique within its scope. *Foundation for Future Database Systems: The Third Manifesto,* Chapter 13, *The Inheritance Model,* includes IM Prescription 9, which is not (and cannot be) part of *The Third Manifesto per se* but does require every variable to be named. Furthermore, our discussion of OO Proscription 2 in Chapter 9 effectively suggests that we have always assumed that every variable has a name. Now we are considering making that assumption explicit in our next revision of *The Third Manifesto.*

Assuming, then, that we have named relation variables, the question is asking if Hugh Darwen's presentation involves "hiding away information in variable names", suggesting (a) that there is something unusual about the proposed design, compared with relational database designs in general, and (b) that the unusual thing in question is that the design involves using variable names in an unorthodox manner, *viz*., to hide information.

Regarding (a), we claim that there is nothing unusual. The proposed database is just a collection of relation variables whose possible values are restricted according to declared constraints.

Regarding (b), we claim that no information is hidden in those names. Of course, those names do stand for relvars and relvars certainly do carry information, so it might be argued that the names carry information too, in a sense. But we claim that this state of affairs cannot be regarded as either "hiding information" or as a violation of *The Information Principle*—especially since the notion of "hiding information" and *The Information Principle* itself are both somewhat imprecise at this time. The names could be changed arbitrarily without affecting the design at all. The operations available on the variables denoted by those names are exactly those general operations that are normally supported—their effects are in no way determined by the actual choice of names. In fact, it most definitely *is* the case that all the information represented in the database is "cast in the form of" relations. Every statement of belief is represented by some tuple in one of those relation variables, and every one of those tuples represents some statement of belief. These statements of belief *constitute* the information represented in the database.

Now, it might be observed that the proposed design violates *The Principle of Orthogonal Design*, proposed by C.J. Date and David McGoveran in *Relational Database Writings 1991-1994*, Chapter 4: *A New Database Design Principle*. But violating that Principle, we argue, does not imply violating Codd's *Information Principle.*[1] In any case *The Principle of Orthogonal Design*, like normalization, is only a recommendation, not a "legal requirement". We think that *The Information Principle* was an attempt by Codd to summarize the most important "legal requirement" of The Relational Model of Data. Furthermore, an alternative design that does not violate *The Principle of Orthogonal Design*, as suggested in the presentation, would still involve relvar names.

Consider CALLED, EARNS, SALARY_UNK and UNSALARIED. Suppose we decide to dispense with UNSALARIED. Now every tuple in CALLED either has a matching tuple in EARNS and no matching tuple in SALARY_UNK, or has a matching tuple in SALARY_UNK and not in EARNS, or has no matching tuple in either SALARY_UNK or EARNS. Now *The Principle of Orthogonal Design* is not violated,[2] and yet any information deemed to be hidden in the name SALARY_UNK is presumably still hidden there. (But we claim none is hidden.)

The important thing about relvar names is that they do presumably (and definitely, in the example given in the presentation) have different meanings in their real world interpretations, implying that the relvars correspond to different predicates. The system doesn't know the real world interpretations, of course, but it does know that SALARY_UNK represents the predicate SALARY_UNK (*t*) and UNSALARIED the predicate UNSALARIED (*t*), where *t* can be meaningfully replaced by a tuple of type TUPLE { Id *Id_type*[3] }. The real world interpretations of SALARY_UNK (*t*) and UNSALARIED (*t*) are the predicates given on Slide 9 of the presentation.

Notice that a tuple *per se* does not constitute a statement of belief. For example, the tuple TUPLE { Id 1234 }, considered in isolation, does not convey any information at all. The presence of that tuple in UNSALARIED, however, tells the system that UNSALARIED (TUPLE { Id 1234 }) is currently true and tells *us* that "The person with id 1234 has no salary" is currently true. Contrariwise, the absence of that tuple from that relvar tells the system that UNSALARIED (TUPLE { Id 1234 }) is currently false and us that "The person with id 1234 has no salary" is currently false. So, in order to convey information, a tuple has to be regarded as representing an instantiation of a specific predicate, yielding a proposition whose truth is determined by the appearance or non-appearance of that tuple in the corresponding relvar. Every predicate is a declarative sentence, possibly and usually containing some parameters[4] (standing for nouns). As far as the system is concerned, a relvar name stands for the part of that relvar's predicate that remains when the parameters are

---

[1] One of us (Chris Date) is on record to the effect that a violation of *The Principle of Orthogonal Design* **is** a violation of *The Information Principle*, but neither of us any longer believes this to be the case. Please accept our apologies if you have been misled by this previous assertion (given, for example, in *An Introduction to Database Systems, 7th edition*).

[2] At least, not the Principle in its current form, which has been found to be insufficiently precise. A reformulation of that Principle is currently being sought.

[3] The actual type is not relevant to the issue here. In the next paragraph we assume it to be INTEGER.

[4] Logicians often call these variables, but we prefer to avoid that term because of possible confusion with the programming language concept of the same name. Our "relvar" is an abbreviation for "relation variable".

removed.[5] Given a relvar name *and* an appropriate tuple, we can construct the corresponding proposition, with real nouns in place of the predicate's parameters. Thus, given a relvar name *and* the zero or more tuples constituting the body of that relvar, we have a proposition[6] that is believed to be true—in other words, some information. Take away the set of tuples *or* the relvar name, and we don't have information![7]

Finally, we would like to make a comment on "separate table for each meaning of the Nulls". The comment is not intended to be taken very seriously, nor is it meant unkindly. No, we don't propose to have a separate table for each meaning of Nulls. There's no such thing as a Null. We cannot entertain the idea of having a table for any meaning of something that doesn't exist! We would rather characterise our design (a trifle loosely) as involving a relvar for every distinct kind of statement[8] that we wish to be represented in the database.

[5] We have encountered the terms *predicate letter* and *predicate constant* for this concept, but we prefer to avoid those terms too.
[6] Formed by anding together the zero or more propositions represented by the tuples of that relvar body.
[7] We remark in passing here that it is perhaps unfortunate that *The Information Principle* was so imprecisely stated.
[8] For "kind of statement", read "predicate".