INTEGRITY IN THE RELATIONAL DATA MODEL

INFORMATICA 3/92

Keywords: relational model, entity integrity, referential integrity, logic approach, consistency

Mario Radovan
Fakultet ekonomije i turizma, Pula
Inštitut »Jožef Stefan«, Ljubljana

The paper shows that the general integrity rules are simply "corollaries" of the (traditional) definitions of primary and foreign keys. Further, it is shown why (and how) the database-specific integrity rules should be treated as expressions of that knowledge which is contained, built-in in our "language, atitudes and measures". Such a conception of integrity allow us to reduce the (rather informal) concept of "correctness" of databeses to the request for its consistency (as a theory).

INTEGRITETA V RELACIJSKEM MODELU: Članek dokazuje, da so splošna pravila integritete preprosto korolarji tradicionalnih definicij primarnega in zunanjega ključa. Dokaže tudi zakaj (in kako) bi pravila integritete specifična za dano bazo, morala biti obravnavana kot izrazi (zapisi) tistih znanj, ki so vsebovana (vgrajena) v našem "jeziku, stališčih (odnosih) in merilih". Tako pojmanje integritete nam omogoča da (precej neformalen) pojem "pravilnosti" baze podatkov zvedemo na zahtevo po njeni konsistentnosti (kot teorije).

1. Introduction

"We begin with a little philosophy", says Date, <Dat 90, p. 275>, at the beginning of the chapter on Relatinal Integrity Rules. In this article we begin in the same way; indeed, we also continue in this way because it seems that the basic concepts concerning integrity in general are not yet defined suitably nor clearly enough.

Our discusion is concerned with the approach and definitions given in <Dat 90> because Date is not only one of the active contibutors to the field od databases,

but also one of the most influental writer on the topic. We are not speaking so much in terms of the "right" or "wrong" way of doing things as in terms of more or less suitable ways of conceiving and defining basic concepts in data modeling in general, and especialy of integrity constraints (rules).

2. The Logic Approach

A set of seminal ideas on the "logic approach" to data modeling was collected in <Gal 78>; among subsequent contributions, it seems that the most influental on the topic was <Rei 84>. An analysis and a proposal for (re)defining inference and integrity for knowledge bases, defined in terms of clausal logic and SLDNF-resolution, was given in <Rad 87>.

Generally speaking, the conceptual foundation of any "informal knowledge" always involves "some logic". Such a logic consists (at least) of a language (defined on the level of syntax (form) and semantics (interpretation, meaning)), and a (set of) inference rule(s) which allows us to infer consequences which follow from the given knowledge. Therefore, a "conceptual foundation" of any "knowledge" bring us to a "system" in whose language such a knowledge should be unambigously expressed, and by whose rules of inference (all and only) the consequencess of this knowledge could be inferred. What is known as "relational data model" is also such a "system".

The logic approach defines a way of conceiving "facts" and "knowledge", and a way of speaking about them. By "model" we mean a set of facts (states, values), while a "theory" is a set of propositions (assertions, knowledge, ...) concerning some model.

In accordance with such a conception, the content of a data/knowledge base is seen as a theory which expresses knowledge concerning some "world". If such a theory is correct and complete, it "tells the truth and only the truth" about the world, and we say that the world is its model. (Incidentally, in such a context, we would prefer to speak of the "relational system", rather than of the "relational model".)

3. The General Integrity Rules

In the relational data model (system), the integrity rules are traditionaly divided into two clases: general and data-base-specific rules. The first class contains only two rules, known as the entity integrity rule and the referential integrity rule. Although there is much "theoretical debate" (not always very clear!) about these rules, their meaning is fairly simple and clear.

The entity integrity rule says that a thing about which we (want to) speak must be identified, while the referential integrity rule says that what we refer to must exist.

Although these rules are concerned with the relational data model (system), it seems that those requests should be a minimal condition of any "afirmative speech" to be considered inteligible! (Therefore, they should form a part of any system (language), except those of poetry. Of course, hypotetical speech (in religion, politics, ...) is

possible (and common!) even without real identification and existence.)

3.1. The Entity Integrity Rule

In spite of the simplicity of the entity integrity rule, there seems to be much vagueness and disagreement concerning its meaning and definition than one would expect. In <Dat 90, p. 279>, Date states this rule as:

"No component of the primary key of a base relation is allowed to accept nulls".

He specifies (p. 280): "... we take 'null' to mean, simply, a value or representation that is understood by convention not to stand for any real value of the applicable atribute" (underlined M.R.). In fact, the same was stated on p. 251, where we find: "null is not a value".

To evaluate Date's conception of the entity integrity rule, it seems apropriate to cite his definition of candidate (and primary) key also; in <Dat 90, p. 277> we find:

"Atribute K (possibly composite) of relation R is a candidate key for R if and only if it satisfies ...

1. Uniqueness: At any given time, no two tuples of R have the same value for K.
2. Minimality: If K is composite, then no component of K can be eliminated without destroying the uniqueness property".

Discusing the entity integrity rule, Date rightly emphasizes that "all kind of problems" arise if the key is (partially) null. Among other things, he stresses that in such a case we do not even know (in general) whether the tuple with a null in its key "represents" some of the entities about which we already have a tuple in the relation.

However, a "kind of problem" also springs from Date's concluding fragment of the section on entity integrity; he says:

"... it is commonly but erroneously thought that the entity integrity rule says something along the lines of 'Primary key values must be unique'. It does not. That uniqueness requirement is a part of the basic definition of the primary key concept per se. (underlined M.R.) The entity integrity rule says (to repeat) that primary keys in base relations must not contain any nulls".

In fact, it seems that there really are such "thoughts" as Date mentiones above; whether they are really "erroneous" or not, we will try to see in what follows. For example, in <Gro 90, p. 270> (a respectable book, in my opinion) Groff and Weinberg give the following definition:

"Entity integrity: The primary key of a table must contain a unique value in each row ..."

Evidently, this definition says exactly what Date denies! Now, which of the two (in fact, three) autors has an "erroneous thought"? And why, and where does the error come from? That is the question we shall try to answer now.

First of all, both positions seem to be coherent. Which one will then be the "loser"? We claim: The entity integrity rule itself!

For Groff and Weinberg, the

position is fairly clear: They give a definition, without much argumentation. The only objection which could be raised against their position is that such a definition makes the entity integrity rule superfluous. Namely, it only repeats what is already contained in the concept of uniqueness in the very definition of kandidate key!

Date's position seems somehow different. However, it seems that the difference arises mainly from his imprecise definition of the candidate key given above. Namely, what does it in fact mean that "no two tuples of R have the same value for K"?

Let us try to understand it by means of an example. Let the sheme of the relation R be

R(A, B, ...),

and let two tuples from R be

<a1, null, ... >
<a1, null, ... >.

Now, does this pair of tuples violate the above uniqueness request or not?

No, because according to Date, "null is not a value"! Therefore, it is impossible even to speak about the "same value"! And in such a case, it seems necessary to have Date's definition of the entity integrity rule, so as to "expell" the nulls from the key! Namely, otherwise "all kind of problems" would arise: adressibility of tuples, first of all ("null is not a value"!).

However, there was one more requirement in the definition of candidate key: minimality! And this one warns us clearly enough that if we assign to the key atributes something that "is not a value", we must lose uniqueness! And by that, we also deviate from the definition of the key (i.e., we lose even the key itself!). Namely, even though by assigning null to an atribute which is a component of the key, the component is not really "eliminated", it is eliminated in a functional sense: it is not "in function" any more! (A component can guarantee (or support) the uniqueness of the key only by being instantiated to a value - and not to something which is "not a value"!)

Therefore, we must once more conclude that the entity integrity rule (at least, as a rule!) is superfluous, because it only repeats what necessarily follows from the mere definition of the candidate key. Namely, a careful reading of the candidate key definition tells us that the candidate key atributes must not be null.

By the above claim of "imprecision" in Date's definition of the candidate key, we mean that with the expression "same value", Date should have really intended "a value"! Well, there is a negation in his definition of uniqueness, and that always make things less clear. Namely, as we already stated, "not a value" in fact satisfies the request for "no ... the same value"! However, it seems easy to escape this (rather sofistic) trap. We should only reformulate the uniqueness request, in perhaps the following a way:

"Any two (different) tuples of R must have different values for K."

Now, from "have different values" we should be allowed to infer "have value"! And then all additional rules concerning "nulls in key" become superfluous.

3.2. The Referential Integrity Rule

For discussing the referential integrity rule we should first give a definition of foreign key. This concept is pretty clear, however it seems very "unsuitable" for being caught in an "elegant" definition. Date, <Dat 90, p. 282>, states it in the following way:

"Atribute FK (possibly composite) of base relation R2 is a foreign key if and only if it satisfies the following ...

1. Each value of FK is either wholly null or wholly nonnull.

...

2. There exists a base relation R1 with primary key PK such that each nonnull value of FK is identical to the value of PK in some tuple of R1".

Now, the referential integrity rule, Date defines, <Dat 90, p. 284>, with the following words:

"The database must not contain any unmatched foreign key values".

Unfortunately, it seems that there are "some problems" even with this definition. Namely, according to the definition of foreign key, if some "nonnul value" i.e. "value" (all "values" are "nonnull", if null "is not a value"!) of a foreign key FK would be unmatched, it would not be a foreign key! (Read carefully clause (2) of the foreign key definition!) Therefore, in the context of the given definition of foreign key, the referential integrity rule seems to be superfluous (as was the entity integrity key in the context of the definition of candidate key).

With the above discussion, we didn't mean to criticize Date's exposition; in fact, he himself says that "... the foreign key and referential integrity concepts are defined in terms of one another. That is, it is not possible to explain what a foreign key is without mentioning the idea of referential integrity (at least tacitly) ...", <Dat 90, p. 285>. Incidentally, we "feel" that the last sentence is too strong. For it is not very clear why it should not be posible to define a foreign key if (some of) its values would be simply (momentarily) unmatched.

Let us conclude the discusion on general integrity rules with a few remarks and estimates of their position and role in the relational data model (system).

First of all, we hold that there is an esential difference between the request for "uniqueness" (entity integrity) and the one for "matching" (referential integrity). Namely, it is very clear that uniqueness (adresibility) is of an essential importance for the model. Therefore, it should be defined (included) at the level of the syntax of the language (system).

On the other hand, we hold that the referential integrity request is of no such importance on the theoretical level. Namely, what is in fact the (cognitive) difference between a null-valued foreign key and an unmatched one? Let as look at an example.

Let R1 and R2 be relations, and B be a foreign key of R2 in R1:

R1(A, ..., B, ...)
R2(B, ...),

and let

<a1, ..., null, ... >
<a2, ..., b2, ... >
<a3, ..., b3, ... >

be tuples in R1. Further, let the relation R2 contain a tuple

<b2, ... >,

but not the tuple

<b3, ... >.

Now, the first two tuples of R1 do not violate the referential integrity rule (the rule allows nulls), while the third one does. However, we think that there is not much difference among them, at least not at the theoretical level of the data model (system). Namely, the first tuple refers to "something unknown", while the third refers to "something unknown, designated as b3".

Of course, there is a difference between the first and the third tuple of R1 on the operative (practical) level. Perhaps by inserting null (instead of b3) in the third tuple above, we would not bring in much information about "the world": in fact, "the reality" would be "even more uknown"! However, this "even more" is useful, because - with it, expressed as null - we know that the "referred entity" is actually unknown. And that could be a useful information, not about "the world", but about our knowledge of it! For without a null, we could erroneously hold, in accordance with "common sense", that there is a tuple b3 in R2.

In accordance with what is stated above, we hold that the request for referential integrity is a problem which belongs (much) more to the level of implementation (software) than to the basic theoretical (conceptual) level of the model (system). Therefore, we put forward three possible ways of "theoretical treatment" of this rule, where our preference goes from the first downwards.

1. Referential integrity (as a request for obligatory matching) could be ommited as an explicit rule, and as an implicit part of the foreign key definition.

2. Referential integrity could be (and in fact actually is) stated inside (as an implicit part of) the definition of foreign key. In that case, "the problem" is raised on the level of syntax, and there is no more need of a special (explicit) theoretical treatment.

3. Finaly, if we insist to keep referential integrity (as a rule) in the system, then a referential integrity violation should be reduced to the inconsistency of the database as a theory. To achieve this, we should simply define a rule as a "generally holding knowledge (truth)", in the following way:

"For every fk, such that fk is a value of a foreign key, there is (in the database) a primary key with value fk."

Such a rule would then be part of the (knowledge contained in) the database.

Now, if an "actual content" of the database (which is always finite) would allow us (or the DBMS) to infer (compute) that:

"There is such a value of an atribute(s) defined as a foreign key (in some relation) which is actually not a value (instance) of the refered primary key",

then such an inference (result) would contradict the integrity rule stated above, and show the inconsistency of the database content. Naturally, such a control (checking) should be performed before (every) update of a database.

4. Database-specific Integrity

According to Date, a database-specific integrity rule (constraint) "can be regarded as a condition that all correct states of the database are required to satisfy", <Dat 90, p. 437>. Such an assertion immediately lead to the question: What does it mean for a state to be "correct"?

Earlier, in <Dat 90, p. 275>, we find: "... database consists of some configuration of data values, ... (which) ... is supposed to 'reflect reality' (i.e., it is supposed to be a model ... of some piece of the real world ...)". (As we already stated, it would be much more suitable to say that a database is a theory rather than a model. Namely, it is the theory we intend to "reflect reality"!)

"Now, some configurations of values simply do not make sense, in that they do not represent any possible state of the real world", continues Date.

All that sounds well; however, there are too many "critical" (not well defined) concepts here: in addition to "correctness", we now also have "sense" and "possible states of the real world". Let us start with the last one.

Date says (same page) that "weights cannot be negative in the real world". Sounds wise. However, strangely enough, temperature can! Moreover, even the water level of a river can be negative! (I must confess that this "fact" perplexes me anew whenever I hear such a report. I always have an impression that a river somehow went underground!)

With these examples, we simply wanted to ilustrate that it is not suitable to speak in terms of what "can" or "cannot" be, or what is "(im)possible" in the "real world" (which is itself not a very clear concept!). Therefore, we propose to move the discusion to the level of contradiction and (in)consistency, which are well defined concepts, and also "manipulable" in the computational sense.

Philosophically speaking, the first reason for such an approach would be that there is no such a thing as a "mere fact": Facts are always something already shaped in (or by means of) our language and measures! (Well, even if (at some "deep level") there is the True Reality, it is - by definition - indiscernible for us, because the "eyes" and the evaluation of the "seen" will always be ours!)

Now, (database-specific) integrity constraints would express that (relevant) knowledge which is already "built-in" (contained) in our language and measures. Such integrity constraints would not protect the database from "asserting impossible things" but would simply prevent such changes which would lead to contradiction with the knowledge which tacitly holds in the language and which is formally expressed by the database-specific integrity constraints. For example, a constraint concerning (our conception of!) weight could state (in some DDL of DBMS):

"There is no such a thing as a weight less than zero."

Now, if some update would lead to a state in which some value for the atribute WEIGHT would become "less than zero", it would mean that this update would make the database inconsistent! In such a way, the concepts of "cor-

rectness", "sense" and "(in)possibility" are reduced to the basic (and "most suitable") concept of consistency.

Inconsistent theory don't have a model; therefore, we can say with certainty that an inconsistent database would not speak of the "real world" (not even of an "ireal" one, at least not inteligibly). (Note that in such a "conceptual (data) system", integrity constraints are part of the database (theory), although they do not in general state "isolated facts" but "rules" which single facts must not contradict.)

Of course, it is not practically possible to prevent all possible errors in the database on the formal level. However, we hold that those preventive measures which are possible should be best done (defined) by means of (database-specific) integrity rules, where integrity is reduced to the concept of consistency.

5. Conclusion

By discussing integrity rules in the relational data model, we tried to show and emphasize the suitability of the logic, proof--theoretic approach to the definition (or conceptual foundation) of data models in general.

By claiming that general integrity rules are "superfluous" we didn't mean that their content is irelevant, but that this content could be stated (expressed) within definitions of the primary and foreign key. In that case, general integrity rules (as defined e.g. in <Dat 90>) could be treated as simple "corollaries" (i.e., evident consequences) of those definitions.

On the other hand, the database-specific integrity rules should be treated as expressions of that knoledge which is contained (built-in) in our language, measures and atitudes. Database-specific integrity rules are, in fact, general knowledge wich should be thought (and treated) as part of the database. Such a conception and role of the integrity allows us to reduce the concept of "database correctness" to its consistency (as a theory). Of course, at the operative level (i.e., at the level of the DBMS) they would serve to prevent those updates which would bring the database to inconsistency.

REFERENCES

<Dat 90> Date, C.J.: An Introduction to Database Systems, Vol. 1 (Fifth ed.), Addison-Wesley, 1990.

<Gal 78> Gallaire, H., Minker, J. (eds): Logic and Data Bases, Plenum Press, 1978.

<Gro 90> Groff, R.J., Weinberg, N.P.: Using SQL, McGraw-Hill, 1990.

<Rad 87> Radovan, M.: "On Deduction and Consistency in Clausal Logic", Acta Analytica, No 3, 1987.

<Rei 84> Reiter, R.: "Towards Logical Reconstruction of Relational Database Theory", in Brodie, L.M. et all (eds): On Conceptual Modeling, Springer-Verlag, 1984.