Alternative Technologies

Observations on the 1969 Relational Operations

by David McGoveran, Alternative Technologies

Codd's 1969 introduction of the relational model was clearly a work in progress. The original paper contained many diamonds in the rough, some of which would be cut to perfection in time and some of which lay dormant. With the clarity that only hindsight can bring, the paper even contained a few bits that would have to be characterized – with appropriate charity – as false starts. Each of those false starts can be understood in the light of the formal mathematics that underpinned Codd's thinking.

The operations which Codd defined in 1969 bear less resemblance to those so well recognized today as defining the relational algebra than one might expect. Some differences are due to the way in which relations were initially defined, a definition that was also to evolve over time as Codd's relational model matured. Some differences have to do with Codd's apparent desire to address physical issues like storage, which caused him to entangle discussion of these with discussion of formal issues. Some differences have to do with Codd's attempt to be mathematically precise and to maintain the integrity of the mathematical theory of relations as he applied and expanded it so as to meet the needs of data representation. In this review of the original relational operators, I'll try to point out some of these influences, motivate them, and point to how they would eventually evolve. Most important, I hope to convince you that some of these earlier definitions may actually have been somewhat superior.

Over the last thirty-five years, vendors and well-meaning writers pushed relational operations into the hands of less and less well-trained users. As I've noted many times, however, the concept of the "end user" of computing systems has changed greatly from 1969 to the present and, with it, the concept of "end user query language." Codd's clearly did not perceive his

www.AlternativeTech.com mcgoveran@AlternativeTech.com

operations as forming an end user query language and did not even expect all users to, as he put it, "… be directly concerned with these operations." He clearly expected certain users (namely, "information systems designers" and "people concerned with data bank control" – which today we might call "application developers" and "database administrators", respectively) to be expert with the operations so that they could derive relations from other relations, presumably on behalf of other, less expert users.

Codd starts his discussion of operations on relations by reminding the reader that relations are sets and so set operations are applicable to them. Without much explanation, he warns that sometimes the result might not be a relation. This is our first clue that something will change between this first exposition of the relational model and what will come to be known as the relational algebra: Codd's initial set of operations does not have the property of closure. In fact, Codd does not even seem to be concerned with establishing this or any other algebraic property, and it is unclear as to whether or not he recognized its importance at this point. For example, although he notes that the union of a binary relation and a ternary relation is not a relation, he does not suggest a "relational union" operation or other relational analogues of the set operations that would preserve closure. He might have suggested, for example and as an obvious first approximation, restricting the operands of a "relational union" to relations of like degree. From that point, a further restriction to compatible domains (or even derived covering domains) would have been a next obvious step.

Codd's first operation is column permutation, which he calls simply "permutation." Introduction of the permutation operation is motivated by Codd's reliance on the mathematical theory of relations (part of set theory), which led him to define his relations in a similar manner. His 1969 definition of relation included "indexing" (a mathematical term that is best understood as "numeric labeling" and having little to do with an "index" as used for data access) of columns, so that column order is significant in the mathematical model. Note that this is merely a mathematician's way of maintaining column identity for purposes of reference and not an ordering that is intrinsically meaningful. Indeed, this is the standard approach to "column" differentiation in the mathematical theory of relations.

Codd defined the permutation operation on ***n***-ary relations as the permutation (i.e., a change in

order) of its columns, of which there are ***n***-factorial possibilities. He noted that the permutation of binary relation does have a different meaning than that of the original relation: it is the converse relationship. The semantics of binary relations (in mathematical relation theory) are fairly intuitive, roughly, column **A** has relationship ***R*** to column **B**. Let ***R'*** be the converse of ***R***. If columns **A** and **B** are permuted, we say that **B** has the converse relationship ***R'*** to **A**. (For example, if the relationship ***R*** means **A** is the parent of **B**, then ***R'*** means **B** is the child of **A**, "is parent of" and "is child of" being converse relationships.) By contrast, the semantics of ***n***-ary relations (namely, how each column is related to every other column) are less clear. In fairness, Codd may have been relying implicitly upon mathematical relation theory and the well-known reduction of an ***n***-ary relation to a set of binary relations, assuming the interpretation was obvious. In 1969, Codd bypassed the problem and the meaning of an arbitrary permutation of an ***n***-ary relation was not given explicitly. However, he did point out that the set of queries answerable by a relation is the same set as that for its permutations. In other words, the permutation operation does not change the informational content or semantics of a relation. This shows he at least gave some thought to the meaning of relations under permutation and that his understanding was consistent with the mathematical theory of relations.

This observation seems like a slight contradiction with the distinction in meaning of a binary relation and its permutation, but a bit of thought about binary relations will show why it is a trivial difference. It also raises a question: Why bother defining the permutation operation in a data model if it does not manipulate the informational content? The answer lies in Codd's attention to formal detail. Having defined relations as having significant column ordering, the formalism demands an operation to manipulate this mathematical *artifact*. Without it, accidental differences of column ordering among relations would have made other, more useful operations difficult or even impossible to define.

Codd next defined the selection operator. He began by defining the now familiar concept of a projection as the result of selecting ***k*** columns from an ***n***-ary relation (***k*** < ***n*** + 1), then removing any duplication from the resulting array. Codd's selection operator was defined as a combination of this permutation and projection. Note that this duplicate removal means that there is potentially a one-to-many relationship between every row in the projection and rows in the relation. Note also that it would have been sufficient to define the result as a set, but Codd

seemed perpetually pulled between thinking as a mathematician and thinking of how the relational model was to be implemented on a computer. Throughout the 1969 paper he often writes of relations in terms of their computer representation as arrays, and so incorporates the computer step of duplicate removal in those arrays. Although Codd was reasonably careful with his choice of words, subtle distinctions such as relation (the formal model) versus array (the physical representation) were often lost on readers and contributed to many corruptions of the model. As with permutation of an n-ary relation, Codd did not explain what a projection of an ***n***-ary relation means.

Codd's join operations are more complex than either permutation or selection, and quite different from join as it is understood today. Today, join is thought of as an operation on two relations of arbitrary degree and is treated (erroneously) by most practitioners as if it is always valid. By contrast, Codd defined join as an operation on two *binary* relations *that satisfy a particular constraint*. The fact that Codd defined join only on binary relations is an artifact of the relational model's lineage from mathematical relation theory. He addresses this problem by showing how to treat any ***n***-ary relation as a binary relation for the purposes of the join operation, namely by partitioning the columns into two groups and treating each group as a composite column. While the rather sticky problem of how this arbitrary grouping and subsequent ungrouping affects the semantics of the join operation (consider how dependencies could be rewritten to accommodate grouping and ungrouping, and when this rewrite would or would not be valid) is simply ignored by Codd, it does provide a clear syntax and implementation.

What is much more important than this oversight is the constraint that Codd placed on his 1969 definition of the join operation, and its effect. Outside of the set operations, join is the only primitive dyadic relational operation which Codd defined. Codd's definition of join required that every join be lossless; that is, that the relations to be joined were joinable without loss of information. This definition in terms of the result and a constraint does not provide a unique construction of a join whenever the two relations have a many to many relationship over the join keys, a fact that Codd called a "point of ambiguity". The point is that more than one relation might then satisfy the join definitional requirements for any given pair of relations. He then went on to recognize the existence of and define cyclic (a.k.a. join) dependencies, and defined the constraints that would guarantee that an n-adic join involving cyclic dependencies would be

lossless. For reasons we won't go into, he gives the special name of *cyclic n-join* to this case and distinguishes it definitionally from the "linear join" such as the natural join.

It is worth noting that Codd's requirement that his join operations be lossless means that the result of a join can be nonloss decomposed into the original relations. This definition, although not applied to a normalization procedure, anticipates both ordinary reducibility (from his linear n-join requirement) such as is required for third normal form, and the criteria for decomposability used in reducing a relation to fifth normal form (from his cyclic n-join requirement). Given Codd's intended use of the join operation in support of derived relations, it seems likely that he originally thought of all relations in the stored set as satisfying the non-loss requirements and therefore being joinable if they had domains in common. Relations that cannot be joined losslessly are forever isolated (i.e., they cannot participate in relational operations) in Codd's 1969 relational model, and only two relational operations (permutation or decomposition by projection) are then allowed on them. Of course, it was possible that two relations might also be combined via set union or intersection. Codd remained silent on this point, did not give explicit definition to the set operations on relations, and does not seem to have recognized (in the 1969 paper at least) the semantic relationship between set intersection and relational join. Whether this was an oversight or Codd thought it too obvious to point out, we will probably never know.

Finally, Codd defines the relational analog of function composition in terms of a join, followed by a projection of all columns except the join keys. Once again, Codd defines the operation in terms of binary relations, and then extends the definition to n-ary relations exactly as he did for the join operations. He does not explain the semantics of the operation, and does not consider how column composition affects semantics in the case of n-ary relation composition.
No analog of the familiar restriction operation was defined in 1969 paper. The earliest published version seems to have appeared in the 1970 paper. This mathematical generalization to relations defined as the restriction of a function to a subset of its domain was defined as a dyadic operation in which one relation was restricted to values found in a second relation. Again, its meaning was not given much consideration and, I believe, left too much to ad-hoc invention by Codd's readers.

Finally, it is worth noting that the 1970 article, which introduced a normal form for relations, continued Codd's initial disposition toward constraining what he considered to be a relation and legitimate relational operations. Not only did he maintain his original definition of join as inherently lossless, but he also assumed that the unnormalized collection of relations would satisfy two conditions. First, the graph of interrelationships of the nonsimple domains would be a collection of trees. In other words, every domain was either simple or strictly contained in some other domain. Second, every component domain of every primary key would be simple. His normalization procedure then results in a set of relations in which every domain is simple. Taken together, Codd's assumptions and definitions anticipate much of normalization as we know it today. Only when relations are not restricted as Codd originally required does much of the subsequent development in "normalization theory" become relevant. His thinking was visionary indeed.

Afterward, 2014

This paper was originally written in 2005. After re-reading it recently, I made a number of edits and additions for clarity and to fix a few grammatical errors. None of these change the substance of the original paper. Of course, much more could be written about the issues raised here and I hope to do that writing in the future.

Please also note that the address in the footer is out of date. Correspondence may be addressed to David McGoveran, POB 2097, Boulder Creek, CA 95006 and will be forwarded to my Florida residence.