Towards a Minimal Set of Operations for Nested Relations

Marc H. Scholl
Technical University of Darmstadt, West Germany

Since the first publications on non-first-normal-form relations in the late 70's and early 80's, a variety of formalizations of the data structure of and operations for nested relations have been devised. The data structure itself is defined almost identical in the several approaches, only some subtle differences concerning special cases can be observed. For instance, both the VERSO relations [AB84] and the PNF relations of [RKS85] do not allow relations without an "atomic key", i.e. the set of attributes forming a key must not contain relation-valued attributes, which, however, is allowed in [SS84/86]. However, concerning operations for nested relations, a much broader scale of languages was proposed. Formal operations were defined in algebra, calculus and SQL style. In this position paper we want to compare several proposals of algebras for nested relations. We distinguish between "minimal" extensions of the flat algebra (as for example the ones given by [FT83] or [RKS85])—i.e. those that try to get along with the "Nest" and "Unnest" operations—, and the "maximal" extensions that supply nested operations (e.g. [AB84, SS84/86]).

Others have investigated equivalences between algebras and calculi for nested relations, e.g. [RKS85], van Gucht, Abiteboul and Beeri (the latter ones see this workshop). This issue is related to the problem of identifying a minimal subset of algebraic operators that give the full expressive power of a certain calculus. (The problem of the powerset operation is treated e.g. by Beeri). In the sequel we concentrate on a specific topic: a minimal subset of the algebra from [SS84/86] can be shown to consist of the usual selection, union and unnest operation plus extended projection. In particular, nest, difference and Cartesian product (and thus join) can be expressed in terms of the other operations. This exciting result can be derived by utilizing the concept of "*dynamic constants*" as introduced in [SS84/86]. Therefore, we argue in favor of maximally extended algebras as opposed to the 'minimal' extensions. Problems with empty subrelations and corresponding null values (cf. [Scho86]) are responsible for the fact, that the maximally extended algebra operations can *not* be expressed by unnesting, applying flat algebra operations and finally nesting back. Rather, a nested algebra must be defined on its own [AB84, SS84/86] to achieve the desired expressive power. This is the reason why others have restricted the scope of considered relations to certain normal forms (e.g. PNF) or changed the definitions of basic algebra operations to reflect these normal forms. With the maximally extended algebras we do not need such restrictions.

The algebra defined in [SS84/86] allows application of relational expressions at any place in an algebra operation, where attributes occur that are relation-valued, i.e. subrelations. As an example consider a nested relation representing departments and employees

$$Dept(dno, dname, \ldots, mgrno, Emp(eno, ename, \ldots)).$$

Here, *mgrno* stands for the employee number of the department manager. Then a query asking for the department information of the CS department along with the manager's data can be formulated as a nested project-select expression

$$\pi[dno, dname, \ldots, \sigma[eno = mgrno](Emp)](\sigma[dname = \text{'Computer Science'}](Dept)).$$

Notice, that *mgrno* is neither an attribute of *Emp* nor a constant in the usual sense, but nevertheless is used to select on *eno*. The solution is that *mgrno* is a ***dynamic constant*** w.r.t. *Emp* in the above schema. In fact, all attributes that occur 'along the path' from a subrelation to the root of the schema tree have a fixed value for all subtuples. By extending the schema forest of a nested relational database by an additional common root node "DB", this notion could be extended to capture also the other separate relations contained in the database.

These dynamic constants can now be used to define nest, difference, and Cartesian product operations in terms of nested projection plus selection and unnest. First of all, consider nesting: those fragments of tuples in the relation are grouped into a subrelation whose other fragments have identical values. This can be expressed as (for ease of notation we consider only a two attribute relation $R(a, b)$):

$$\nu[BS = (b)](R) \equiv \pi[a, \pi[b](\sigma[a' = a](\pi[a : a', b](R))) : BS](R).$$

Notice, how R itself is used as a dynamic constant w.r.t. R. The notation $\pi[\ldots, a : a', \ldots](R)$ is used to denote renaming, which in the above formulation is necessary to avoid ambiguity. The intuition is that for each tuple in R—the argument of the nested projection—the value of R—the dynamic constant—is the same. Thus the concept of a dynamic constant corresponds to certain scoping rules. Notice that the use of R as a dynamic constant in projections on R was not included in the original definition from [SS84/86], but can be used without introducing recursion.

Similarly, the difference of two relations, say $R(a, b)$ and $S(a, b)$, can be expressed by a nested algebraic query:

$$R - S \equiv \sigma[\sigma[a = a' \wedge b = b'](\pi[a : a', b : b'](S)) = \emptyset](R).$$

This formulation closely corresponds to intuition ("select all tuples in R that do not exist in S"). Note that the nested selection is not an essential extension, but can be expressed via usual selection and nested projection [SS84/86].

Finally, we show that also Cartesian product can be derived with the nested algebra. For two relations $R(R_1, R_2)$ and $S(S_1, S_2)$ the product $RS(R_1, R_2, S_1, S_2) = R \times S$ can be obtained by a nested projection plus unnest:

$$RS = R \times S \equiv \mu[S](\pi[R_1, R_2, S](R)) \equiv \mu[R](\pi[R, S_1, S_2](S))$$

While the 'minimal algebras' are claimed to be sufficient for the manipulation of nested relations (at least those in some appropriate normal form), we showed that the extensions of the flat algebra in the style of [SS84/86] or [AB84] *do add* to the power of the algebra. A thorough treatment of empty sets and null values [Scho86] must be used to define the nested algebra in terms of the flat one plus nest and unnest. Moreover, by using the nested algebra together with the powerful construct of dynamic constants, we can even find a minimal set of operations without difference, Cartesian product and nesting. An obvious advantage of maximally extended algebras over minimally extended ones is that they can be used as an efficient interface of nested relational DBMSs (also see Bidoit and Paul in this workshop).

References:

[AB84] Abiteboul, S., Bidoit, N., "Non First Normal Form Relations to Represent Hierarchically Organized Data", *Proc. ACM PODS*, Waterloo, 1984

[FT83] Fischer, P.C., Thomas, S.J., "Operators for Non-First-Normal-Form Relations", *Proc. IEEE COMPSAC*, 1983

[RKS85] Roth, M.A., Korth, H.F., Silberschatz, A., "Extended Algebra and Calculus for ¬1NF Relational Databases", *Techn. Rep. TR-84-36*, Revised Version, University of Texas at Austin, 1985

[Scho86] M.H. Scholl, "Theoretical Foundation of Algebraic Optimization Utilizing Unnormalized Relations", *ICDT '86 Proc. Int. Conf. on Database Theory*, Rome, LNCS 243, Springer, Berlin, Heidelberg, 1986, 380–396

[SS84/86] H.-J. Schek, M.H. Scholl, "The Relational Model with Relation-Valued Attributes", Techn. Rep. TU Darmstadt, No. DVSI-1984-T1, *Information Systems 11*, 2 (June 1986), 137–147

Bibliography: See H.-B. Paul, in this workshop handout.