On Roth, Korth, and Silberschatz's Extended Algebra and Calculus for Nested Relational Databases

ABDULLAH U. TANSEL
Bilkent University
and
LUCY GARNETT
Barnard M. Baruch College

We discuss the issues encountered in the extended algebra and calculus languages for nested relations defined by Roth, Korth, and Silberschatz [4]. Their equivalence proof between algebra and calculus fails because of the keying problems and the use of extended set operations. Extended set operations also have unintended side effects. Furthermore, their calculus seems to allow the generation of power sets, thus making it more powerful than their algebra.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic; H.2.1 [**Database Management**]: Logical Design—*data models, normal forms*; H.2.3 [**Languages**]: *data manipulation languages*

General Terms: Languages, Theory

Additional Key Words and Phrases: Equivalence of algebra and calculus, nested relations, relational algebra, relational calculus

1. INTRODUCTION

Roth, Korth, and Silberschatz (RKS) defined algebra and calculus languages for Nested Form (NF) relations, and for Partitioned Normal Form (PNF) for a subset of such relations [4]. They extended relational algebra operations to work within the domain of NF relations that are in PNF. They also attempted to show the equivalence of the relational algebra and calculus languages and stated that "with the assistance of these extended operators we prove the equivalence of the NF relational calculus and NF relational algebra" [4, p. 390]. For the reader's convenience and for clarity, we briefly summarize essential points of RKS's article [4] before we provide our comments.

Authors' current address: Barnard M. Baruch College, City University of New York, NY 10010.

A nested relation scheme consists of zero-order names (attributes) that are atomic, and higher order names (attributes) that are nested relation schemes. In an instance of an NF relation, zero-order names assume atomic values from their associated domains, and higher-order names assume nested relations that are composed of the values in their respective domains. A relation structure is $\langle R, r\rangle$, where R is the nested relation scheme and r is its instance. The Tuple Relational Calculus (TRC) includes the $\in$ operator and a set-building formula $(s[i] = (u \mid \psi'(u)))$, where $\psi'(u)$ is a formula, in addition to the usual definitions given in [4, p. 393]. The Relational Algebra (RA) includes two new operators **Nest**(ν) and **Unnest**(μ): "The basic set of operators work exactly as before except the domains may be atomic or set-valued" [4, p. 394].

RKS restricted the class of NF relations to relations that are in PNF. An NF relation $\langle R, r\rangle$ is in PNF if $A_1, A_2, \ldots, A_n \rightarrow X_1, X_2, \ldots, X_m$, where $A_1, A_2, \ldots, A_n$ and $X_1, X_2, \ldots, X_m$ are the zero- and higher-order attributes of R, respectively. Recursively, the relation structure $\langle X_i, t[X_i]\rangle$ for any tuple $t \in r$ and any attribute name X_i should be in PNF as well. Consequently, a nested relation without any zero-order attributes is in PNF if and only if it contains one single tuple, $k = 0$ [4, p. 397].

RKS also extended RA operations to work within the class of PNF relations. These include extended union, extended intersection, extended difference, extended Cartesian product, extended natural join, and extended projection [4, sect. 6]. Let's call this algebra Extended Relational Algebra (ERA). As an example, the extended union, $r \cup^e s$ places t_1 and t_2 into the result when $t_1 \in r$ and $t_2 \in s$ and t_1 and t_2 disagree on their zero-order attributes. If t_1 and t_2 agree on their zero-order attributes, then the corresponding higher-order attributes are combined (collapsed) to form a single tuple in the extended union. This rule is applied recursively on each higher-order attribute. The other operations are similarly extended. RKS showed that PNF relations are closed under extended algebra operations [4, Theorem 6.1, p. 402].

RKS gave two theorems on equivalence of their RA and relational calculus languages. For proofs they provided a translation method from RA to the relational calculus. This is followed by a translation method from relational calculus to the ERA. They also mentioned a keying method:

> In order to avoid problems where $\nu_A(\mu_A(r)) \neq r$, and so that the extended operators do not interact improperly, we assume each database relation $(r, q, \ldots)$, their nested relations, and relations created by collecting constants into a limited domain, have an implicit keying attribute (or set of attributes) whose values uniquely determines the values of all other attributes. We consider this attribute to be added to each relation before it is used and removed when the relation is projected or presented as the final result, using appropriate algebra operations. A key can always be added to a relation by making a side-by-side copy of the relation with itself and using one of the copies as a key . . . . If r is a relation with arity n, then a side-by-side copy can be made as follows:
>
> $$\sigma_{1=n+1 \wedge 2=n+2 \wedge \cdots \wedge n=n+n}(r \times r).$$

> The first n attributes of this new relation then serve as the key . . . . Note that relations that are in partitioned normal form already satisfy these key constraints. [4, p. 409].

After this observation, RKS ignored the issue of keying in their inductive translation from calculus to algebra. Yet the need for keying is subtle, and its incorporation in this process is not straightforward, especially in the translation of the set-building formula. In the following sections, we provide our comments on [4].

2. ISSUES

2.1 On Keying

The keying method described by RKS does not work. The keyed relation is only in PNF if the original relation is in PNF. Hence, the extended operations will not generate the intended result when applied to such keyed nested relations.

Consider the relation structure $\langle R, r_1 \rangle$ and $\langle R, r_2 \rangle$ given in Figure 1. $r_3 = r_1 \cup r_2$ is given in Figure 2. r_3 is calculated according to the standard definition. This is also indicated in [4, p. 394]: "The basic set of operators work exactly as before . . . ." Now translate $r_1 \cup r_2$ into TRC: $r_3 = \{t \mid t \in r_1 \vee t \in r_2\}$. The translation method is straightforward and is provided in [4, p. 404]: "The basis and five cases (case 1–5) for $\cup$, $-$, $\times$, π and σ are as in [6]." Take this TRC expression and translate it back to an equivalent RA expression. Algorithm 1 from [4, p. 407] creates the graph shown in Figure 3. The domain of D_t is

$$
\begin{aligned}
D_t^1 &= \pi_1(r_1) \cup \pi_1(r_2), \\
D_t^2 &= \pi_2(r_1) \cup \pi_2(r_2), \\
D_t &= D_t^1 \times D_t^2 = \big(\pi_1(r_1) \cup \pi_1(r_2)\big) \times \big(\pi_2(r_1) \cup \pi_2(r_2)\big).
\end{aligned}
$$

D_t is given in Figure 2. The relations r_1 and r_2 are in PNF, whereas D_t is not. Thus, D_t needs a key. By applying the keying method, we obtain D_t', which is given in Figure 4. However, r_1 and r_2 also have to be keyed for the correct translation. Call them r_1' and r_2'. Now, we are ready to translate this TRC formula:

$$
\begin{aligned}
&t \in r_1 \Rightarrow \Rightarrow D_t' \cap^e r_1' = r_1'' && \text{(case 1 in basis, [4, p. 410])}, \\
&t \in r_2 \Rightarrow \Rightarrow D_t' \cap^e r_2' = r_2'' && \text{(case 1 in basis, [4, p. 410])}, \\
&t \in r_1 \vee t \in r_2 \Rightarrow \Rightarrow r_1'' \cup^e r_2'' && \text{(case 1 in induction, [4, p. 410])}, \\
&r_4 = \pi_{3,4} && (r_1'' \cup^e r_2'').
\end{aligned}
$$

r_4 and intermediate relations are given in Figure 5. Clearly, $r_3 \neq r_4$. The keying method fails because extended set operators do not take the whole key copy. RKS mentioned that the first copy (the first n attributes) functions as a

A	B'
	B
1	a
	b

r_1

A	B'
	B
1	a
	c

r_2

Fig. 1. Relation structures $\langle R, r_1 \rangle$ and $\langle R, r_2 \rangle$.

A	B'
	B
1	a b
1	a c

Fig. 2. $D_t, r_3 = r_1 \cup r_2$.

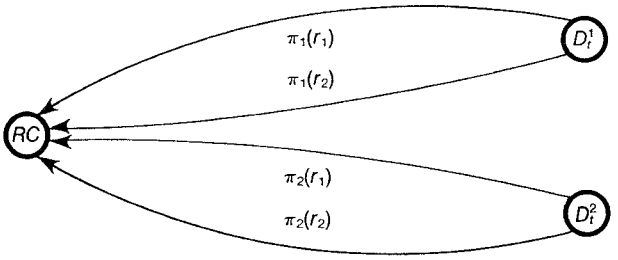

Fig. 3. Graph created by Algorithm 1.

A	B'	A	B'
	B		B
1	a b	1	a b
1	a c	1	a c

Fig. 4. D_t'.

key. However, RKS did not show how this can be achieved by using the RA (ERA) oeprations. One possible solution is to make the key copy as one single attribute. This can be done by nesting the key copy into one single attribute [2], that is, $v_{x=\{A, B'\}}(D_t')$. Although X provides a key for the relation D_t', this still does not work since extended set operations are based on atomic

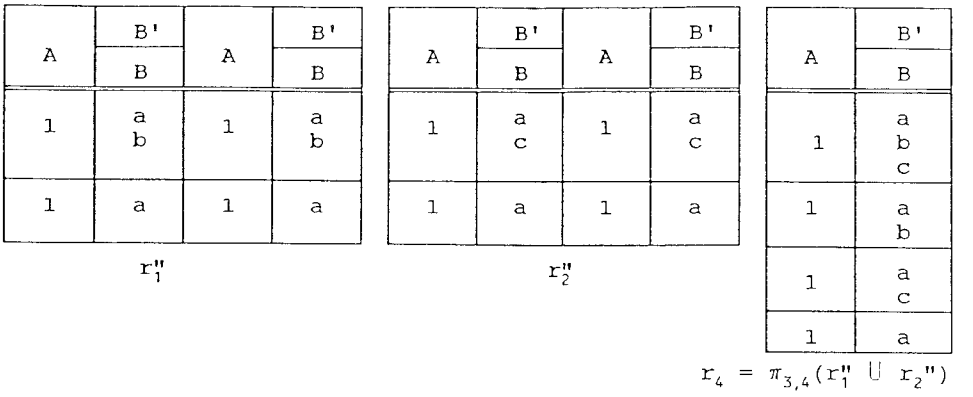

Fig 5. r_4 and intermediate relations

attributes. Redefinition of extended set operations to allow nested attributes in the key part is one possible solution. However, it will be in conflict with the main theme of RKS's approach in TRC to RA translation.

Given that the keying method suggested by RKS does not work, as an alternative try straightforward labeling of each tuple by a unique atomic value. The labeling should be done in such a way that identical tuples of different relations should be assigned the same key label. Considering that all of the database relations and limited domains are keyed by this method, the translation method of RKS for the union of the relations in Figure 1 gives the result in Figure 2, which is correct. The keying method is simple. However, incorporating it into the translation method of RKS requires several modifications: All of the relations should be keyed even if only one of them is not in PNF; the key attributes should be retained when the projection operation is used (unlike RKS's proposition to remove them [4, p. 409]); before applying a nest operation, the related key attributes should be removed, etc. The example given in [4, p. 413] can be corrected if the relations are keyed by the above labeling method with the mentioned modifications. However, we have found the following example for which the translation method of RKS from calculus to algebra does not work even if this keying method is used:

$$\{t^{(1)} \mid (t[1] = \{s \mid s \in r_1 \wedge \neg s[1] = \text{'}e\text{'}\}) \vee (t[1] = \{x \mid x \in r_2\})\}.$$

This TRC expression denotes the sets made up from all of the values in r_1, excluding 'e' or r_2, but not both. r_1 and r_2 are single-column relations. r_1 is $\langle a\rangle, \langle b\rangle, \langle c\rangle$, and r_2 is $\langle d\rangle, \langle e\rangle, \langle f\rangle$. This expression is safe according to the definition of safety given by RKS [4, p. 393]. Before translating this expression into the relational algebra, transform it to eliminate the $\wedge$ operator:

$$\{t^{(1)} \mid (t[1] = \{s \mid \neg(\neg s \in r_1 \vee s = \text{'}e\text{'})\}) \vee (t[1] = \{x \mid x \in r_2\})\}.$$

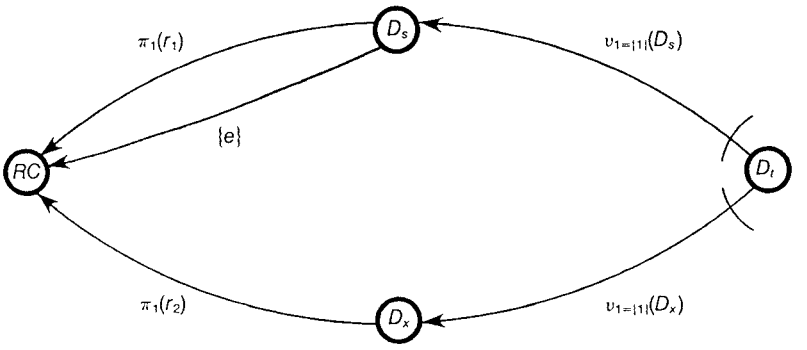

Fig. 6. Graph for limited domains.

Algorithm 1 from [4, p. 407] creates the graph depicted in Figure 6. The expressions for the limited domains are as follows:

$$D_s = \pi_1(r_1) \cup \{e\},$$
$$D_x = \pi_1(r_2),$$
$$D_t = \upsilon_{1=\{1\}}(D_s) \cup \upsilon_{1=\{1\}}(D_x).$$

The relations are given in Figure 7. Note that we keyed these relations by labeling. Now, do the translation (similar to the example given in [4, p. 413].

$$s \in r_1 \Rightarrow\Rightarrow E_1 = D_t \times r_1$$
$$s = \text{`}e\text{'} \Rightarrow\Rightarrow E_2 = \sigma_{4=\text{`}e\text{'}}(D_t \times D_s)$$
$$\neg s \in r_1 \vee s = \text{`}e\text{'}) \Rightarrow\Rightarrow E_3 = \big((D_t \times D_s) -^e E_1\big) \cup^e E_2$$
$$\neg\big(\neg s \in r_1 \vee s = \text{`}e\text{'}\big) \Rightarrow\Rightarrow E_4 = \big(D_t \times D_s\big) -^e E_3$$
$$t[1] = \{s \mid \cdots\} \Rightarrow\Rightarrow E_5 = \pi_{1,3}\big(\upsilon_{3=\{3\}}\big(\pi_{1,2,4}(E_4)\big)\big)$$
$$x \in r_2 \Rightarrow\Rightarrow E_6 = D_t \times r_2$$
$$t[1] = \{x \mid \cdots\} \Rightarrow\Rightarrow E_7 = \pi_{1,3}\big(\upsilon_{3=\{3\}}\big(\pi_{1,2,4}(E_7)\big)\big)$$
$$\{t^{\{1\}} \mid \cdots\} \Rightarrow\Rightarrow E = E_5 \cup^e E_7$$

Finally, E creates the relation $\{\langle abcdef\rangle\}$, which is not correct. The correct result should be $\{\langle abc\rangle, \langle def\rangle\}$. Note that, even if one were to intersect both E_5 and E_7 with D_t before the last step, the result would still be incorrect (it seems this is necessary although RKS do not use it). In particular, this would lead to the result, $\{\langle abce\rangle, \langle def\rangle\}$. So, RKS's translation method from calculus to algebra is not correct.

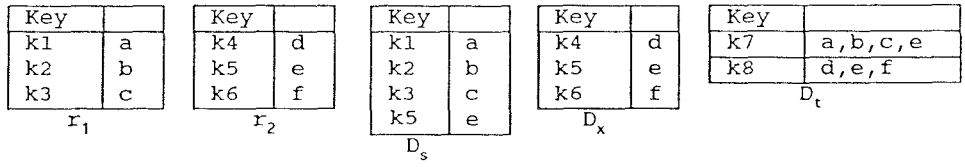

Fig. 7 Relations in the translation.

In their reply, RKS state the following:

> Tansel and Garnett apply Algorithm I of RKS to this keying method even though Algorithm I was designed for a different method. It is not at all surprising that the algorithm fails for such an input. . . . The actual problem with this example is that the extended algebra operators ($-^e$, $\cup^e$) are interacting in a bad way with the limited domains. If no keys and standard difference and union operators are used in this example RKS's methodology works. The use of extended operators and not keys are most likely the main problem with RKS's methodology.

2.2 On the Interpretation of Calculus Objects

RKS did not give an interpretation for calculus objects explicitly. However, the overall approach of their article implies that they are using the standard interpretation given by Ullman for calculus objects [6]. In translating TRC formulas, RKS followed Ullman's method of intersecting interpretation of formulas by the domains of free variables involved. Again, they used extended set interaction in place of set intersection. To translate $\{t \mid \psi(t)\}$ they used

$$D_t \cap^e \{t \mid \psi(t)\}$$

and stated "since ψ is safe, intersection with D_t does not change the relation denoted, so we shall have proved the theorem" [4, p. 409]. This is not true unless the interpretation of calculus objects and D_t are keyed. Consider again the TRC formula for the set union and the relations r_1 and r_2 of Figure 1. D_t is given in Figure 2. Translation of the subformula $t \in r_1$ does not give r_1 because

$$D_t \cap^e \{t \mid t \in r_1\}$$

produces $\langle 1\{a\}\rangle$ in addition to the original tuple of r_1. Again, keying is needed to avoid the affect of extended set intersection. The keying method used by RKS does not work. A unique key, such as labeling of tuples, works. Both D_t and the interpretation of $\{t \mid t \in r_1\}$ should be keyed. In this case, the result of the above expression becomes r_1. However, keying interpretation of calculus objects is not obvious and may involve serious complications.

2.3 On the Extended Difference Operation

Extended set difference (extended set intersection) creates an unintended side effect that leads to the creation of empty results. RKS stated that "since our model does not include null values or empty sets, the operations are well

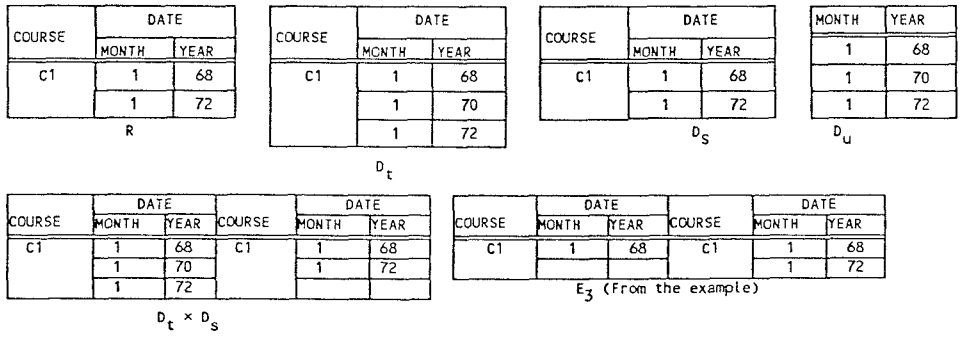

Fig. 8. Possible instance for relation R.

defined" [4, p. 399]. Furthermore definitions of extended difference and extended intersection do not allow empty components in tuples; such tuples are eliminated from the result. In the translation from TRC to ERA, some tuple components may be identical, which causes extended set difference to eliminate such tuples because these components become empty. Such cases occur because the Cartesian product of relations are taken to make the operand relations compatible. Consider the example given by RKS in [4, p. 413] and a possible instance for the relation R, depicted in Figure 8. Domains D_s, D_t, and D_u are given in the same figure. Now, look at the expression E_4 [4, p. 413]. The first two subexpressions of E_4, $(D_t \times D_s) -^e E_1$ and $(D_t \times D_s) -^e E_2$, return an empty set, as expected. On the other hand, the third subexpression, $(D_t \times D_s) -^e E_3$, does not create the expected tuple, but instead returns the empty set because the D_s components of both relations are identical. Therefore, E_4 returns the entire set $D_t \times D_s$, contrary to what RKS expected. Finally, E gives the first two columns of E_4, that is, D_t, as the result, which is not correct. Note that this problem occurs independent of keying. So, we do not bother to key as we trace the example.

2.4 On the Expressibility of Extended Set Operations in RA

RKS gave a method for expressing the extended union of ERA in terms of the basic relational algebra operators:

> Briefly, this can be done by unnesting the operands, decomposing relations into several projections . . . [3, 4] . . . . This procedure works for relations that are in PNF. If they are not in PNF, the appropriate algebra operators can be used to add a key to each relation or nested relation so that the relations are PNF (see Section 7, the above procedure applied . . . . [4, p. 400].

The second part is not correct. Regardless of the keying method used, this procedure does not correctly convert extended set operations that involve relations that are not in PNF since keying does not allow tuples to be considered even if they agree on the atomic attributes. Keying only ensures interaction of the identical tuples. This is contrary to the logic of extended

operations. A procedure for expressing extended set operations involving any nested relation can be found in [5].

2.5 On the Expressive Power of TRC

RKS sought to avoid uncontrolled creation of power sets [4, p. 393]. Consider the following TRC expression and the relation r with one attribute:

$$\{t \mid \exists u(t[1] = u[1] \wedge t[1] = \{s \mid s \in u[1] \wedge s \in r\})\}.$$

This TRC expression generates the power set of the relation r. It does not seem to violate any of the safety rules given in [4, p. 394]. The definition of the set-building formula [4, p. 392], $s[i] = \{u \mid \psi'(u)\}$, implies that ψ' is allowed to have other free variables in addition to u [4, p. 393]. The definition of safety does not explicitly put constraints on the set-building formula. Given the example in RKS's article, in particular, that used to translate the nest operation [4, p. 405], it seems reasonable to assume that the above expression is also safe. However, it is known that the power set of a relation cannot be expressed in the RA. Obviously, the method proposed by RKS cannot translate the above TRC expression into RA. The translation will not produce an ERA expression that gives the power set of r. So, the TRC defined by RKS is more powerful in expressive power than the RA they defined.

In their reply, to avoid creation of power sets, RKS proposed augmenting constraint 4c [4, p. 394], as u may not be the free variable s in the formula $\psi(s)$ of the expression $\{s \mid \psi(s)\}$. However, we do not think this completely solves the problem, since a counterexample can easily be formulated. If the generation of power sets is not eliminated, this leads to a discussion of recursive queries and the type of calculus language needed to cope with them.

3. CONCLUSION

We have provided our observations on Roth, Korth, and Silberschatz's article [4]. We have found the article useful in understanding the nested relational model, and algebra and calculus languages for nested relations. RKS claimed that they proved the equivalence of algebra and calculus languages for nested relations. However, there are theoretical as well as technical questions on the validity of this claim for relations in PNF and relations not in PNF, as is demonstrated by the above observations. An equivalence proof for algebra (including a looping construct) and calculus languages for nested relations can be found in [1].

ACKNOWLEDGMENTS

The authors thank Gio Wiederhold and the reviewers for their valuable comments.

REFERENCES

1. GARNETT, L., AND TANSEL, A. U. Equivalence of relational algebra and calculus languages for nested relations. *Comput. Math. Appl. 23*, 10 (1992), 3-25.

2. OZSOYOGLU, G., AND OZOSOYOGLU, M. Z. An extension of relational algebra for summary tables. In *Proceedings of the 2nd International Workshop on Statistical Database Management* (Los Altos, Calif., Sept. 1983). pp. 202–212.
3. ROTH, M. A. Theory of non-first normal form relational databases. Ph.D. dissertation, Dept. of Computer Science, Univ. of Texas, Austin, May 1986.
4. ROTH, M. A., KORTH, H. F., AND SILBERSCHATZ, A. Extended algebra and calculus for nested relational databases. *ACM Trans. Database Syst. 13*, 4 (Dec. 1988), 390–417.
5. TANSEL, A. U., AND GARNETT, L. Temporal relational data model. Tech. Rep., Baruch College–CUNY, New York, Mar. 1991.
6. ULLMAN, J. D. Principles of database systems. 2nd ed. Computer Science Press, Potomac, Md., 1982.

Received May 1989; revised June 1990; accepted February 1991