

Relational completeness of query languages for annotated databases

Floris Geerts^{1,2} and Jan Van den Bussche¹

¹ Hasselt University/Transnational University Limburg

² University of Edinburgh

Abstract. Annotated relational databases can be queried either by simply making the annotations explicitly available along the ordinary data, or by adapting the standard query operators so that they have an implicit effect also on the annotations. We compare the expressive power of these two approaches. As a formal model for the implicit approach we propose the color algebra, an adaptation of the relational algebra to deal with the annotations. We show that the color algebra is relationally complete: it is equivalent to the relational algebra on the explicit annotations. Our result extends a similar completeness result established for the query algebra of the MONDRIAN annotation system, from unions of conjunctive queries to the full relational algebra.

1 Introduction

Recently, much attention has been paid to annotated databases [10, 4, 2, 5, 8, 7, 6, 3]. In querying annotated databases, there are two distinct approaches:

1. In *annotation propagation* [10, 4, 2, 6, 3], queries are directed primarily at the ordinary data, not the annotations: the latter are merely propagated to the query results. For example, when joining two relations, the annotations of two joined tuples would become annotations of the new joint tuple.
2. In *annotation querying* [8, 7, 5], queries can be directed to the annotations as well as to the ordinary data. For example, when joining two relations, two tuples might be considered joinable only if they have a common annotation. Such join queries are outside the scope of annotation propagation.

Of course, these two approaches are not competing; it is simply that in some applications we want annotation propagation, while in other applications we want to really query on the basis of annotations. As a matter of fact, annotation propagation can be precisely characterized [3] as that part of annotation querying that is invariant under arbitrary re-annotations, even those re-annotations that replace two different annotations by the same one.

In the present paper, we are concerned with full annotation querying, and here one can again distinguish two approaches: *explicit* and *implicit*.

1. In explicit querying, we simply make the annotations explicitly available along with the ordinary data; any standard query language can then be used

to query the database. For example, suppose we want to join annotated relations $R(A, B)$ and $S(A, C)$ not only on their common A -attribute, but also on common annotations. Then we simply model R as a relation $R(A, B, N)$, where N is an extra column holding the annotations, and likewise model S as $S(A, C, N)$, and write in SQL:

```
select R.*, S.*
from R, S
where R.A=S.A and R.N=S.N
```

A similar feature is provided by the ANNOT operator of the pSQL language in DBNotes [5], where we would write:

```
select R.*, S.*
from R, ANNOT(R) N1, S, ANNOT(S) N2
where R.A=S.A and N1=N2
```

2. In implicit querying, which is more in the spirit of annotation propagation, annotations are not explicitly addressed in query formulations. Rather, the standard query operators are adapted so that they have an effect not only on the ordinary data but also on the annotations. For example, in the query algebra of MONDRIAN [8], one would write the above join query as

$$\mu \sigma_{R.A=S.A}(R \times S),$$

where

- the Cartesian product operator \times is adapted so as to keep, for each joint tuple $r \cup s \in R \times S$ with $r \in R$ and $s \in S$, two sets of annotations: the annotations that r already had in R , and the annotations that s already had in S ;
- the selection operator σ simply propagates these sets of annotations;
- the new merge operator μ intersects the two sets of annotations.

A natural question now arises as to the relative expressiveness of explicit versus implicit annotation querying. This question was already addressed for the MONDRIAN query algebra, which has been shown to be equivalent to the positive relational algebra on explicit annotations [8]. In the present paper, we continue this investigation and extend it to the full relational algebra (as opposed to its positive fragment, which does not have the difference operator). Recall [1] that the relational algebra is much more powerful and complicated than its positive fragment. For instance, in the positive algebra only unions of conjunctive queries can be expressed, and containment and equivalence of queries is decidable; in the full relational algebra, all first-order logic definable queries can be expressed, and equivalence (let alone containment) is undecidable.

We will introduce *color relations* as a simple but general abstraction of annotated databases. A color relation is a standard database relation, where additionally every tuple is annotated by some set of “colors”. Moreover, we will introduce the *color algebra (CA)*, an adaptation of the relational algebra to deal with color relations. CA is inspired by, but different from, the MONDRIAN query algebra. The operators of CA always produce color relations as output;

in particular, in CA one cannot compute intermediate results that explicitly relate the colors of different tuples. Nevertheless, we will prove that CA can still express any expression of the full relational algebra on explicit annotations, as long as the latter expression starts from color relations and finally ends up in color relations.

Our result, while answering a natural question, is mainly of theoretical interest. Yet, good theoretical underpinnings of new database management features, such as annotation, are important. We hope that the elegant formalism provided by our color algebra can serve as a guide to the understanding and design of annotation query languages.

2 Color relations

Basically we assume as given an infinite set of *attributes*, an infinite set \mathbb{D} of *data values*, and an infinite set \mathbb{C} of *colors*. The sets \mathbb{D} and \mathbb{C} are disjoint; colors serve as an abstraction for annotation values.

1. A *relation schema* is a finite set R of attributes.
2. A *tuple* over R is a mapping $t: R \rightarrow \mathbb{D}$.
3. A *relation* over R is a finite set of tuples over R .
4. A *coloring* of a relation r is a subset r' of $r \times \mathbb{C}$, i.e., a set of tuple–color pairs where the tuples come from r , such that every tuple of r appears in r' , i.e., every tuple of r gets at least one color.
5. We call r the *underlying relation* of r' . We agree that whenever we denote a coloring by a primed letter, the unprimed letter stands for the underlying relation.
6. Colorings of relations over R are also called *color relations over R* .
7. A *database schema* \mathcal{S} consists of a finite set of relation variables x , each with an associated relation schema $\mathcal{S}(x)$.
8. A *color database* D over \mathcal{S} consists of a set of color relations $D(x)$, one for each relation variable x of \mathcal{S} , such that $D(x)$ is a color relation over $\mathcal{S}(x)$.

We can view a color relation r' alternatively as a mapping r' from r to $2^{\mathbb{C}}$, as follows:

$$r'(t) = \{c \mid (t, c) \in r'\}.$$

Note that, since every tuple gets at least one color, $r'(t)$ is never empty. For any subset $s \subseteq r$, the restriction of the mapping r' to s , which we denote by $r'|_s$, is of course a coloring of s . We will use this observation in the following section.

Remark 1. In our data model, we restrict attention to the coloring of entire tuples. In annotation systems such as DBNotes [2, 5], not just tuples in relations can be colored, but also individual components of these tuples. We can model this by multiple color relations, one for each attribute. The system MONDRIAN [8, 7] even allows the coloring of arbitrary subsets of projections of a relation. Even more generally, one can consider annotations of arbitrary combinations of records and sets [3]. Such complex structures can always be decomposed in multiple flat relations, however, and since the focus of this paper is on expressive power, our model of color relations is sufficient.

3 The color algebra

We are familiar with the classical relational algebra operations on relations: union (\cup), difference ($-$), natural join (\bowtie), renaming (ρ), selection (σ), and projection (π). We now define a number of analogous operations on color relations. The result of these operations is again a color relation.

Let r' and s' be two color relations over the same relation scheme R .

Union: $r' \cup s'$ is the standard set-theoretic union. This is a coloring of $r \cup s$.

Tuple difference: $r' \boxminus s'$ equals $r'|_{r \setminus s}$. It is thus a coloring of $r \setminus s$.

Full difference: $r' - s'$ is the standard set-theoretic difference. It is a coloring not of $r \setminus s$, but of

$$(r \setminus s) \cup \{t \in r \cap s \mid r'(t) \not\subseteq s'(t)\}.$$

For the definition of the next two operations, s' no longer needs to be over the same relation scheme as r' .

Tuple join: $r' \boxtimes s'$ equals

$$\{(t_1 \cup t_2, c) \mid t_1 \cup t_2 \in r \bowtie s \text{ and } c \in r'(t_1) \cup s'(t_2)\}.$$

It is a coloring of $r \bowtie s$.

Full join: $r \bowtie s$ is defined in the same way as $r \boxtimes s$, except that now we take the intersection $r'(t_1) \cap s'(t_2)$ rather than the union. It is thus a coloring not of $r \bowtie s$, but of

$$\{t_1 \cup t_2 \in r \bowtie s \mid r'(t_1) \cap s'(t_2) \neq \emptyset\}.$$

Renaming: if $A \in R$ and B is an attribute not in R , then $\rho_{A/B}(r')$ equals

$$\{(\rho_{A/B}(t), c) \mid (t, c) \in r'\},$$

with $\rho_{A/B}(t) = t|_{R-A} \cup \{(B, t(A))\}$ the classical renaming of a tuple. It is thus a coloring of $\rho_{A/B}(r)$.

Selection: if $A, B \in R$, then $\sigma_{A=B}(r')$ equals $r'|_{\sigma_{A=B}(r')}$.

Color selection: if $k \geq 2$ is a natural number, then $\sigma_{\text{color} \geq k}(r')$ equals $r'|_u$, where

$$u = \{t \in r \mid |r'(t)| \geq k\},$$

with $|r'(t)|$ denoting the cardinality of $r'(t)$, i.e., the number of distinct colors of t in r' .

Projection: if $X \subseteq R$, then $\pi_X^c(r')$ equals

$$\{(t|_X, c) \mid (t, c) \in r'\}.$$

This concludes the definition of the operations of the *color algebra*, abbreviated CA.

Example 1. A simple example is the CA-expression

$$\pi_X^c(x \bowtie \text{green}),$$

where `green` is a constant relation (over attributes disjoint from x) consisting of a single “green”-colored tuple and X consists of the attributes in x . This expression returns all tuples in x that are colored ‘green’; all colors of those tuples are returned as well.

For another example, the CA-expression

$$(x \bowtie (x \boxtimes y)) - (y \bowtie (x \boxtimes y))$$

applied to colored relations r' and s' , returns joint tuples $t_1 \cup t_2$ from the natural join of the underlying relations r and s (with $t_1 \in r$ and $t_2 \in s$); these joint tuples are colored by the colors t_1 has in r' , except for the colors t_2 has in s' . In particular, if t_1 has only colors that t_2 has too, then the joint tuple $t_1 \cup t_2$ is not returned at all, since in colored relations, each tuple must have at least one color.

As a final example, the expression

$$x - \sigma_{\text{color} \geq 3}(x)$$

returns all tuples in x that have at most two colors.

Remark 2. We remark that most of the operators in CA are intuitive except for maybe the color selection $\sigma_{\text{color} \geq k}$. This operator is necessary, however, to show the completeness of CA.

4 CA and the relational algebra

Let us reserve a special attribute `col` and agree that it is never used in the relation schemes of color relations. For any relation scheme R , we define the relation scheme $\bar{R} = R \cup \{\text{col}\}$. We can naturally view a color relation r over R as a relation over \bar{R} , as follows:

$$\{t \cup \{\text{col}, c\} \mid (t, c) \in r\}.$$

Conversely, any relation r over \bar{R} can be viewed as a color relation as follows:

$$\{(t|_R, t(\text{col})) \mid t \in r\}.$$

Beware that when we regard r as a *color* relation, it is a color relation over R , i.e., r 's relation scheme is just R , because the color attribute is implicit in color relations. Indeed, this is exactly the main feature of the color algebra: that colors are handled automatically. When we regard r as an ordinary relation, however, it is a relation over \bar{R} and the color attribute becomes explicitly visible.

Under the view of color relations as ordinary relations, we can apply classical relational algebra operations to color relations, and consider relational algebra

expressions with \bar{R} as result relation scheme to be producing color relations over R . It then becomes apparent that the classical relational algebra can actually simulate the color algebra. The simulation is given in Table 1. The table shows the simulation of the individual operations; the simulation of more complex expressions can be obtained using composition.

Table 1. Simulation of CA by relational algebra. In the case of $x \boxminus y$, the R refers to the relation scheme of the color relations x and y ; in the cases of $x \boxtimes y$ and $\sigma_{\text{color} \geq k}(x)$, the R (S) refers to the relation scheme of the color relation x (y). Moreover, in the simulation of $\sigma_{\text{color} \geq k}(x)$, the auxiliary attributes col_i are chosen such that they do not appear in R .

$x \cup y$	$x \cup y$
$x \boxminus y$	$(\pi_R(x) - \pi_R(y)) \bowtie x$
$x - y$	$x - y$
$x \boxtimes y$	$(x \bowtie \pi_S(y)) \cup (\pi_R(x) \bowtie y)$
$x \bowtie y$	$x \bowtie y$
$\rho_{A/B}(x)$	$\rho_{A/B}(x)$
$\sigma_{A=B}(x)$	$\sigma_{A=B}(x)$
$\sigma_{\text{color} \geq k}(x)$	$\pi_{\bar{R}} \sigma_{\bigwedge_{i \neq j} col_i \neq col_j} (\rho_{col/col_1}(x) \bowtie \dots \bowtie \rho_{col/col_k}(x))$
$\pi_X^c(x)$	$\pi_{X \cup \{col\}}(x)$

More interestingly, the converse simulation holds as well: every operation on color relations that is definable in the relational algebra is already definable in CA. More formally, to every color database schema \mathcal{S} we can associate the relational database schema $\bar{\mathcal{S}}$ which has precisely the same relation variables, but when relation variable x has relation scheme R in \mathcal{S} , then x has relation scheme \bar{R} in $\bar{\mathcal{S}}$. We will establish:

Theorem 1. *For every relational algebra expression over $\bar{\mathcal{S}}$ whose result relation scheme is of the form \bar{R} for some relation scheme R , there exists an equivalent CA-expression over \mathcal{S} .*

In proving this theorem, one cannot hope for a simple bottom-up syntax-directed translation from relational algebra to CA, such as we had with Table 1 for the other direction. For instance, consider in that table the line for $\sigma_{\text{color} \geq k}(x)$, but now read from right to left. More generally, the challenge is how to deal with relational algebra expressions that produce relations as intermediate results that explicitly relate colors from different tuples in the database.

5 Simulation of the relational algebra by the color algebra

In this section, we prove our theorem. It is actually sufficient to do this for a restricted fragment of the relational algebra, which we call the color-typed

relational algebra, denoted by RA^c . In order to define this fragment, we must first go from our one special color attribute col to an infinite set \mathcal{C} of color attributes, and agree that these are, like col , never used in relation schemes of color relations. Of course we put $col \in \mathcal{C}$. The color-typed restriction now only lies in a condition imposed on selections and renamings. Specifically, if e is an expression, then $\sigma_{A=B}(e)$ and $\rho_{A/B}(e)$ are only allowed if either A and B are both color attributes, or are both not color attributes. Expressions of the form $e_1 \cup e_2$, $e_1 - e_2$, $e_1 \bowtie e_2$, or $\pi_X(e)$ can be constructed just like in the classical relational algebra.

A result on the first-order completeness of many-sorted logic [9] implies that every relational algebra expression over a database schema $\bar{\mathcal{S}}$ with result relation scheme of the form \bar{R} can be expressed in RA^c . (We point out that this depends crucially on the disjointness of the universes \mathbb{D} of data values and \mathbb{C} of colors.) So, we indeed only have to prove the theorem for RA^c .

Our proof uses the technical notion of an *R-parameterized monadic database schema*, where R is a relation scheme. This is a relational database schema where every relation name has the same relation scheme \bar{R} . Equivalently, it can be viewed as a color database schema where every relation scheme has the same relation scheme R . Furthermore, an RA^c -expression f over such a schema is called *R-uniform* if it satisfies the following:

- f uses only renamings $\rho_{A/B}$ and selections $\sigma_{A=B}$ where A and B are color attributes;
- all projections π_X appearing in f satisfy $R \subseteq X$.

The intuition is that an *R-uniform* expression does not explicitly work with the attributes in R ; these attributes are merely dragged along as parameters.

We now show that CA can simulate *R-uniform* RA^c , in the following sense:

Lemma 1. *Let f be an R -uniform RA^c -expression over the R -parameterized monadic database schema \mathcal{S} . Let S be the result relation scheme of f .*

- If $S \cap \mathcal{C} = \emptyset$, i.e., $S = R$, then there exists a CA-expression $\text{sim}(f)$ such that $f(D)$ equals the relation underlying $\text{sim}(f)(D)$, for each color database D over \mathcal{S} .
- If $S \cap \mathcal{C} \neq \emptyset$, then for each equivalence relation E on $S \cap \mathcal{C}$, there exists a set $\text{sim}_E(f)$ of mappings from $S \cap \mathcal{C}$ to CA, such that $f(D)$ equals

$$\bigcup_E \bigcup_{\tau \in \text{sim}_E(f)} \sigma_{\wedge_{(col', col'') \in E} col' = col''} \bowtie_{col' \in S \cap \mathcal{C}} \rho_{col'/col'}(\tau(col')(D))$$

Proof. Assume that \mathcal{S} consists of the relation names z_1, \dots, z_n . We begin by refining the classical correspondence between the relational algebra and the relational calculus (first-order logic, FO) to *R-uniform* RA^c . The corresponding fragment of FO, which we denote by FO_R^c , is obtained as follows. Let

$R = \{A_1, \dots, A_m\}$. We use the A_j 's, plus all color attributes, as first-order variables. The allowed atomic formulas are of two forms:

1. $z_i(A_1, \dots, A_m, col')$ with $col' \in \mathcal{C}$. We abbreviate such formulas by $z_i(R, col')$.
2. $col' = col''$ with $col', col'' \in \mathcal{C}$.

The only variables that can be quantified are color attributes. It is then readily seen that R -uniform RA^c corresponds to FO_R^c under the active-domain semantics, with the understanding that, when evaluating a formula in a database D , the tuple of free variables A_1, \dots, A_m is only instantiated by R -tuples that actually appear in D .

We next apply the well-known quantifier elimination method for monadic first-order logic to FO_R^c . Concretely, this gives us that every FO_R^c formula can be written without quantifiers if we additionally allow predicates of the form $|z_\alpha(R)| \geq \ell$ in formulas, where $\ell \geq 1$ is a natural number, and α is a nonempty subset of $\{1, \dots, n\}$. The meaning of such a predicate, for a given tuple t over R , is that $|z_\alpha(t)| \geq \ell$, where $z_\alpha(t)$ equals

$$\{t' \in \bigcup_i z_i \mid t'|_R = t \ \& \ \bigwedge_{i \in \alpha} t' \in z_i \ \& \ \bigwedge_{i \in \hat{\alpha}} t' \notin z_i\},$$

where $\hat{\alpha}$ abbreviates $\{1, \dots, n\} - \alpha$.

Putting the quantifier-free formula in disjunctive normal form, and simplifying each conjunction, we obtain a disjunction of conjunctions of factors of the following possible forms:

- If $S \cap \mathcal{C} = \emptyset$, then each factor of the conjunction is of one of the following three forms:
 1. $|z_\alpha(R)| \geq 1$. This can be expressed in CA by $\boxtimes z_i - \bigcup_{i \in \alpha} z_i$.
 2. $|z_\alpha(R)| \geq \ell$ with $\ell \geq 2$. This can be expressed in CA by $\sigma_{\text{color} \geq \ell}(|z_\alpha(R)| \geq 1)$.
 3. $\neg(|z_\alpha(R)| \geq \ell)$. This can be expressed in CA by $\bigcup_i z_i \boxminus (|z_\alpha(R)| \geq \ell)$.
- If $S \cap \mathcal{C} \neq \emptyset$, then factors may additionally be of the following possible forms:
 4. $z_i(R, col')$ for some color attribute col' . This can be expressed in CA by z_i .
 5. $\neg z_i(R, col')$. This can be expressed in CA by $\bigcup_j z_j - z_i$.
 6. equalities and inequalities among color attributes.

Without loss of generality, we may assume that in each conjunction γ , the set of equalities and inequalities among color attributes is maximally consistent, involving all color attributes in $S \cap \mathcal{C}$. Such a maximally consistent set gives rise to an equivalence relation E_γ on the color attributes.

We now construct, for each conjunction γ , the following mapping τ from $S \cap \mathcal{C}$ to CA and put it in $\text{sim}_{E_\gamma}(f)$. For each color attribute col' , we take the CA-expressions for all factors of types 1–3 above, and also take the CA-expressions for all factors of types 4–5 that concern the particular color attribute col' . If there are no such factors of types 4–5 for col' , then we add the CA-expression $\bigcup_{z \in \mathcal{S}} \pi_\emptyset(z)$. We conjoin all these CA-expressions using \boxtimes . The resulting CA-expression then equals $\tau(col')$.

□

Our second lemma connects R -uniform expressions to general RA^c -expressions. Together with the first lemma, it establishes the theorem.

Lemma 2. *Let h be an RA^c -expression over \bar{S} with result relation scheme S , and let $R = S - \mathcal{C}$. Then there exist a natural number n ; CA-expressions e_1, \dots, e_n , all with result relation scheme R ; and an R -uniform RA^c -expression $f(z_1, \dots, z_n)$, such that the composition $f(e_1, \dots, e_n)$ is equivalent to h .*

Proof. By induction on the structure of e . If h is a relation name x , then $n = 1$; e_1 is x ; and f is z_1 .

If h is $h_1 \cup h_2$, by induction we have, for $j = 1, 2$, the natural number n_j , the sequence of CA-expressions $e^j = e_1^j, \dots, e_{n_j}^j$, and the RA^c -expression f_j . Then we put

$$\begin{aligned} n &:= n_1 + n_2 \\ e_1, \dots, e_n &:= e^1, e^2 \\ f &:= f_1(z_1, \dots, z_{n_1}) \cup f_2(z_{n_1+1}, \dots, z_n). \end{aligned}$$

The case where h is $h_1 - h_2$ is similar, but now f is $f_1 - f_2$.

If h is $h_1 \bowtie h_2$, we again begin by obtaining the ingredients for h_1 and h_2 by induction, as above. By Lemma 1, we can simulate f_1 and f_2 in CA. We now perform a case analysis based on how the result relation schemes S_1 and S_2 of h_1 and h_2 intersect with \mathcal{C} . There are four cases.

First, $S_1 \cap \mathcal{C} = \emptyset = S_2 \cap \mathcal{C}$. We put

$$\begin{aligned} n &:= 1 \\ e_1 &:= \text{sim}(f_1)(e^1) \bowtie \text{sim}(f_2)(e^2) \\ f &:= \pi_R(z_1). \end{aligned}$$

Second, $S_1 \cap \mathcal{C} = \emptyset$ and $S_2 \cap \mathcal{C} \neq \emptyset$. Let us first introduce the following derived CA operator: $x \triangleright y$ is an abbreviation for $x \bowtie (x \boxtimes y)$. Note that $r' \triangleright s'$, for color relations r' and s' , equals $\{(t_1 \cup t_2, c) \mid t_1 \cup t_2 \in r \bowtie s \ \& \ (t_1, c) \in r'\}$. Now in this case we take n to be the total number of expressions occurring in all sets $\text{sim}_{E_2}(f_2)$, for all equivalence relations E_2 on $S_2 \cap \mathcal{C}$. For each of those expressions g , we form $g' := g(e^2) \triangleright \text{sim}(f_1)(e^1)$, and all these expressions g' constitute the e_i 's. Denoting the relation name corresponding to g' by z_g , we can then use the following expression for f :

$$\bigcup_{E_2} \bigcup_{\tau \in \text{sim}_{E_2}(f_2)} \sigma_{\wedge_{(col', col'') \in E_2} col' = col''} \sigma_{\wedge_{(col', col'') \notin E_2} col' \neq col''} \boxtimes_{col' \in S_2 \cap \mathcal{C}} \rho_{col'/col'}(z_{\tau(col')}).$$

Third, $S_1 \cap \mathcal{C} = \emptyset$ and $S_2 \cap \mathcal{C} \neq \emptyset$. This case is symmetric to the previous case.

Fourth, $S_1 \cap \mathcal{C} \neq \emptyset \neq S_2 \cap \mathcal{C}$. In this case we use three kinds of CA-expressions:

1. $\tau_1(col')(e^1) \bowtie \tau_2(col')(e^2)$, with $col' \in S_1 \cap S_2 \cap \mathcal{C}$, and $\tau_j \in \text{sim}_{E_j}(f_j)$, for an equivalence relation E_j of $S_j \cap \mathcal{C}$, for $j = 1, 2$;
2. $\tau(col')(e^1) \triangleright \tau(col'')(e^2)$, with $col' \in (S_1 \cap \mathcal{C}) \setminus (S_2 \cap \mathcal{C})$ and $col'' \in S_2 \cap \mathcal{C}$, and τ_j as above;
3. $\tau(col'')(e^2) \triangleright \tau(col')(e^1)$, with $col'' \in (S_2 \cap \mathcal{C}) \setminus (S_1 \cap \mathcal{C})$ and $col' \in S_1 \cap \mathcal{C}$, and again τ_j as above.

So, n equals the total number of all possible CA-expressions of those three kinds. For all these expressions, which are all of the form $i \bowtie j$ or $i \triangleright j$, the underlying R -parameterized monadic database schema has corresponding relation names $z_{i,j}$. The expression f then becomes:

$$\begin{aligned}
& \bigcup_{E_1} \bigcup_{E_2} \bigcup_{\tau_1} \bigcup_{\tau_2} \sigma_{\Lambda_{(col', col'') \in E_1} col' = col''} \sigma_{\Lambda_{(col', col'') \notin E_1} col' \neq col''} \\
& \quad \sigma_{\Lambda_{(col', col'') \in E_2} col' = col''} \sigma_{\Lambda_{(col', col'') \notin E_2} col' \neq col''} \\
& \quad \bowtie_{col' \in S_1 \cap S_2 \cap \mathcal{C}} \rho_{col'/col'}(z_{\tau_1(col'), \tau_2(col')}) \\
& \quad \bowtie_{\substack{col' \in (S_1 \cap \mathcal{C}) \setminus (S_2 \cap \mathcal{C}) \\ col'' \in S_2 \cap \mathcal{C}}} \rho_{col'/col'}(z_{\tau_1(col'), \tau_2(col'')}) \\
& \quad \bowtie_{\substack{col'' \in (S_2 \cap \mathcal{C}) \setminus (S_1 \cap \mathcal{C}) \\ col' \in S_1 \cap \mathcal{C}}} \rho_{col'/col'}(z_{\tau_2(col''), \tau_1(col')}).
\end{aligned}$$

If h is $\rho_{A/B}(h_1)$ with A and B not in \mathcal{C} , then we put $n := n_1$; $e_i := \rho_{A/B}(e_i^1)$; and $f := f_1$.

If h is $\rho_{col'/col''}(h_1)$ with $col', col'' \in \mathcal{C}$, then $n := n_1$; $e_i := e_i^1$; and $f := \rho_{col'/col''}(f_1)$.

If h is $\sigma_{A=B}(h_1)$ with A and B not in \mathcal{C} , then we put $n := n_1$; $e_i := \sigma_{A=B}(e_i^1)$; and $f := f_1$.

If h is $\sigma_{col'=col''}(h_1)$ with $col', col'' \in \mathcal{C}$, then $n := n_1$; $e_i := e_i^1$; and $f := \sigma_{col'=col''}(f_1)$.

Finally, if h is $\pi_X(h_1)$, then we simulate f_1 in CA according to Lemma 1. Now if the intersection of the result relation scheme S_1 of h_1 with \mathcal{C} is empty, then we put $n := 1$; $e_1 := \pi_X^c(\text{sim}(f_1)(e^1))$; and $f := z_1$. If $S_1 \cap \mathcal{C} \neq \emptyset$, then we take n to be the total number of expressions occurring in all sets $\text{sim}_E(f_1)$, for all equivalence relations E on $S_1 \cap \mathcal{C}$. For each of those expressions g , we form $g' := \pi_{X-\mathcal{C}}(g)(e^1)$, and all these expressions g' constitute the e_i 's. Denoting the relation name corresponding to g' by z_g , we can then use the following expression for f :

$$\pi_X \bigcup_E \bigcup_{\tau \in \text{sim}_E(f_1)} \sigma_{\Lambda_{(col', col'') \in E} col' = col''} \sigma_{\Lambda_{(col', col'') \notin E} col' \neq col''} \bowtie_{col' \in S_1 \cap \mathcal{C}} \rho_{col'/col'}(z_\tau(col')).$$

□

Acknowledgements

Floris Geerts is a postdoctoral researcher of the FWO Vlaanderen and is supported in part by EPSRC GR/S63205/01.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
2. D. Bhagwat, L. Chiticariu, W.-C. Tan, and G. Vijayvardiya. An annotation management system for relational databases. *The VLDB Journal*, 14(4):373–396, 2005.
3. P. Buneman, J. Cheney, and S. Vansummeren. On the expressiveness of implicit provenance in query and update languages. In T. Schwentick and D. Suciu, editors, *Database Theory—ICDT 2007*, volume 4353 of *Lecture Notes in Computer Science*, pages 209–223. Springer, 2007.
4. P. Buneman, S. Khanna, and W.-C. Tan. On propagation of deletions and annotations through views. In *Proceedings 21st ACM Symposium on Principles of Database Systems*, pages 150–158. ACM Press, 2002.
5. L. Chiticariu, W.-C. Tan, and G. Vijayvardiya. DBNotes: A post-it system for relational databases based on provenance. In *Proceedings 2005 ACM SIGMOD International Conference on Management of Data*, pages 942–944. ACM Press, 2005.
6. G. Cong, W. Fan, and F. Geerts. Annotation propagation revisited for key preserving views. In *Proceedings 15th ACM International Conference on Information and Knowledge Management*, pages 632–641. ACM Press, 2006.
7. F. Geerts, A. Kementsietsidis, and D. Milano. *i*MONDRIAN: A visual tool to annotate and query scientific databases. In Y.E. Ioannidis et al., editors, *Advances in Database Technology—EDBT 2006*, volume 3896 of *Lecture Notes in Computer Science*, pages 1168–1171. Springer, 2006.
8. F. Geerts, A. Kementsietsidis, and D. Milano. MONDRIAN: Annotating and querying databases through colors and blocks. In *Proceedings 22th International Conference on Data Engineering*, page 82. IEEE Computer Society, 2006. 10 pages.
9. J. Van den Bussche and L. Cabibbo. Converting untyped formulas into typed ones. *Acta Informatica*, 35(8):637–643, 1998.
10. Y.R. Wang and S.E. Madnick. A polygen model for heterogeneous database systems: the source tagging perspective. In D. McLeod, R. Sacks-Davis, and H. Schek, editors, *Proceedings of the 16th International Conference on Very Large Data Bases*, pages 518–538. Morgan Kaufmann, 1990.