Context-sensitive Relational Division

Richard Elling Moe
Department of information and media studies
University of Bergen

ABSTRACT
We argue that the division operator in relational algebra is poorly equipped to handle a certain type of universal queries. This parallels earlier investigations into the power of the division operator, but instead of taking such shortcomings as reasons for dismissing the entire division approach to universal queries, we propose a solution that preserves the ideas behind Codds' division.

KEY WORDS
Relational algebra, Division, Universal queries, Database query languages.

1 Introduction

Today, SQL is by far the dominant database query language. Despite this, relational algebra still holds ground. There may be several reasons for its survival.

First, the algebra is a procedural language, whereas SQL is flagged as being declarative. This makes the algebra more suitable for low-level specification of queries and thereby a valuable analytic tool for query-optimization techniques. Furthermore, it is a common conception that declarative languages are more user-friendly than procedural ones but, remarkably, in the case of SQL vs relational algebra the opposite has been claimed [18].

Secondly, The relational algebra has proved to be a good framework for theorizing about query languages. Partly because of its solid footing in mathematical disciplines, and because it manages to define the notion of a query language simply in terms of a small number of independent elementary operations.

Finally we believe that the study of query languages as such requires an awareness of the possible alternative approaches that could be taken. The algebra is important simply in virtue of being *different* from SQL because it thereby represents a different way of thinking about, and dealing with, database queries. We shall keep this consideration in mind as a guideline for the discussion.

In connection with databases, a universal query is a request for the retrieval of data where the specification involves quantification similar to the kind one would express using universal quantifiers in formal logic. Given a database with facts about employees, the departments that employ them, the projects they work on and which projects are run by each department, a universal query could be, for instance:

(1) *Find the employees who work on all projects run by department 5.*

Universal queries are somewhat complicated compared to the typical database query. They have received a fair amount of attention, both with respect to their specification [1, 2, 3, 6, 9] and the algorithms for processing them [10].

Database query languages come in many shapes, with different approaches to handling universal queries. We shall concentrate on the relational algebra where the *division* operator is typically called into action when specifying a universal query. However, it has been pointed out that the division operator isn't suitable for all kinds of universal quantification [1, 3, 6, 7, 9]. We shall identify yet another type of universal query where the standard division is not up to scratch, namely when the quantification relies on contextual information.

In keeping with the view that the individual characteristics that makes a language unique among languages are in themselves valuable, we shall be conservative in the effort to solve our particular kind of query. Thus we try to remain faithful to the original approach to universal queries. For reference, the discussion will touch upon other approaches but our main focus will be on the ideas behind Codds' division operator [2].

2 Basics

There is a wealth of literature on database theory and query languages and notation and terminology varies. We adopt that of Elmasri/Navathe [11].

In its purest form, the relational algebra comprises the selection and projection operators σ and π, respectively, and the set operations $\cup$, $-$ and $\times$. In principle, these operators are all that is needed. This set of operators defines the expressive power of relational algebra[1]. Any set of operators which is equally expressive, or more, is said to be *relationally complete*.

This way of measuring the expressive power readily lends itself to other languages. In fact, relational completeness has become a minimum requirement for relational database query languages. The relational *calculus*

[1]We disregard extensions of the algebra designed to add to its power, such as aggregate functions and recursive closure operators.

and the algebra are equivalent with respect to expressive power [13, 17], whereas the power of SQL goes beyond relational completeness [14].

Other operators can be added to the algebra for practical reasons, such as the join $\bowtie$ and division $\div$. Even if their introduction makes no difference when it comes to expressive power, certain types of queries becomes easier to express. Thus it may be held that they represent an improvement with respect to the expressive *convenience* of the language.

3 Division

Division ($\div$) operates on two relations where the attribute-set of the first properly includes that of the second. Given relations R and S with attribute-sets X and Y, respectively[2], such that $Y \subset X$; $R \div S$ denotes a relation with attribute-set $X - Y$. Specifically, $t \in R \div S$ iff $\{t\} \times S \subseteq R$.

The division operator was designed for universal queries, which otherwise are complicated to express. Take for instance the case of finding those employees who work on all projects run by department number 5. Suppose the database includes the following relation-schemas[3].

- $EMP(\underline{SSN}, DNO)$ relates the social security numbers of employees to the identification-numbers of the departments that employ them.
- $PROJ(\underline{PNUM}, DNUM)$ relates each project, given by an identification-number, to the department that controls it.
- $WO(\underline{ESSN, PNO})$ relates employees and the projects they work on.

Using the division operator $\div$ query (1) may be formulated as follows:

$$\begin{aligned} &PRO5(PNO) \leftarrow \pi_{PNUM}(\sigma_{DNUM=5}(PROJ)) \\ &RESULT \leftarrow WO \div PRO5 \end{aligned}$$

The division operator $\div$ has proved useful in many situations, but some voices has been raised against its usefulness and generality. In response, generalized division operators have been proposed to mend such problems. We shall return to this matter shortly, but first let us take a brief look at an alternative approach to handling universal queries.

3.1 Relational comparison operators

Ever since the early days of SQL it has been well known that universal queries can be expressed as *relational comparisons*, using for instance the `contains` operator. (It should be remarked though that the `contains`-operator is no longer part of the SQL standard. Even so, it is still referred to in the context of SQL [11, 12].) Date [5, 7, 8] argues for the introduction of such operators into the relational algebra and maintain that they offer a more convenient solution for universal queries.

Adapting his approach, query (1) might look like this:

$$\pi_{SSN}(\sigma_{\pi_{PNUM}(\sigma_{DNUM=5}(PROJ)) \subseteq \pi_{PNO}(\sigma_{ESSN=SSN}(WO))}(EMP))$$

Observe that the relational comparison appears as a test for the inclusion of the result of one query in the result of another. That is, on this approach, σ-conditions may include operators on complex domains, in this case relations, rather than the simple domains that the original algebra was confined to, i.e. numbers and character-strings, with (in-)equality operators, 'less than', 'greater than' etc. Furthermore, this introduces the kind of *nested queries* we find in SQL. (*Correlated* nested queries in fact, but we will make a point of that later on.)

We feel that this represents quite a departure from the original algebra. To some degree it even introduces an SQL line of thinking into it. For the present discussion we wish to keep the focus on the traditional relational algebra. So, we return to the division approach to universal queries.

3.2 Generalized Divisions

There has been numerous reports of $\div$ being obscure and difficult to learn [1, 9, 15]. Furthermore, Date [6, 9] points out a mismatch between the intuitive reading of a query such as *Find those who works on all projects run by department 5.* and the results of its algebra solution presented above. In the case when department 5 runs no projects at all, $WO \div PRO5$ would return all employees, rather than none as one might expect. The problem is fixed by a suitable generalization of the division operator. Also, Date [4, 7] claims that his relational comparison operators, of the kind described in the previous section, avoid problems of this sort.

These may well be impracticalities of the division operator but more relevant for our discussion are the claims that $\div$ does not cover the full range of universal queries [1, 3, 7, 9].

Dadashzadeh [3] and Carlis [1] identify certain types of universal queries for which $\div$ is not sufficient. They propose generalized division operators to overcome the problems. However, because of the complex nature of these operators, which involves the use of relational comparison, we feel that the original principle of division has been lost in the approach.

4 Division and context

We now look into a kind of universal query for which the standard division operator seems to be of little use. Consider the following example:

[2]Technically, the attributes are ordered within the relation-schema. We do not go to this level of detail here since relation-schemas may well be defined without such a requirement and since the reader can easily induce the necessary ordering on the fly.

[3]The example is adapted from Elmasri/Navathe [11].

(2) *Find the employees who work on all projects run by their own department.*

Contrast this with query (1) for those who work on the projects run by department 5. Two important differences can be pointed out. First, whereas the projects belonging to department number 5 can be extracted prior to applying division, it is impossible to secure the agreement between the departments of employees and projects by means of a single selection-condition to be applied independently of a division. We elaborate this point by considering the following set:

$$\{x \mid (x,y) \in EMP \text{ and } x \in WO \div \rho_{(PNO)}(\pi_{PNUM}(\sigma_{DNUM=y}(PROJ)))\}$$

(ρ is a renaming-operator, i.e. $PNUM$ is renamed to PNO.) This expression denotes the desired result of our query. Unfortunately it is not an expression of the relational algebra. However, this illustrates that query (2) corresponds to a *series* of applications of $\div$, each with a different 'divisor'. Moreover, these divisors are based on the contents of the database. I.e. when the database state changes, so does the set of actual divisors. Hence, such a series of divisions can not be specified once and for all, as we would need in order to create a general solution for our query.

The second point we emphasize is that with ordinary division there are only two possible roles an attribute can play. Either that of being a *target*. I.e. an attribute for which values are to be retrieved and presented in the result of the division. Otherwise, the attribute is *common* to both relations and is there for the target-attributes to be measured up against. Consider the definition of $R \div S$ above and recall that $Y \subset X$. Here, $X - Y$ contain the targets, which are measured up against the remaining attributes, i.e. those common to R and S. In the example above ($WO \div PRO5$) $ESSN$ is the only target and PNO is the only common attribute. Now, given that all attributes in a division are either targets or common attributes there is no room for bringing other attributes into the picture. Here lies a weakness. Other attributes may also hold useful information. In a sense, such attributes define a context which may be relevant in the case of universal queries. In query (2), the task of finding those who work on all the projects of *their own* department rely on such contextual attributes, namely $DNUM$ and DNO.

We have identified a universal query for which additional information must be brought into the division itself. (As opposed to being administered before or after a division is applied.) Furthermore, this information resides in attributes that are contextual to the division. Apparently the ordinary division operator $\div$ is ill-equipped for the task, and is certainly not a convenient mechanism for expressing this query.

Could it be that the concept of division is simply out of place here? We think not and aim to develop a generalized division operator appropriate for the job.

A recent survey of the literature [16] has revealed no obvious division-based solution to our particular problem. Except perhaps an approach [4] which we are reluctant to adopt. Not that it lacks merit, but we find it a bit too radical for our goals. Nevertheless, we present it briefly before proposing our own solution.

4.1 Correlated nested queries

SQL supports the feature that is often referred to as *correlated nested queries*. I.e. that a sub-query may refer to attributes belonging to its super-queries. Had this mechanism been at our disposal in the algebra, a solution to query (2) might have looked something like the following:

$$EMP \bowtie_{SSN=ESSN} (WO \div \pi_{PNUM}(\sigma_{DNUM=DNO}(PROJ)))$$

Observe that the σ-operation refers to the attribute DNO which belongs to the EMP-relation. That is, the occurrence of an attribute inside the division-expression is correlated with one on the outside.

The trouble is that this is not an acceptable expression in relational algebra because in a selection $\sigma_\varphi(R)$ it is required that the attributes referred to in the condition φ must belong to the relation R.

To our knowledge, no precise definition has been given of the syntax and semantics for an algebra where conditions may refer to attributes outside their normal scope. Nor have we made any attempt to explicate this ourselves. Our reluctance to adopt this approach comes from the judgement that it would have too great an impact on the language. Even if it should turn out that the introduction of correlated nested queries leaves the expressive power of relational algebra unchanged, it would nevertheless alter the dynamics of the language and the line of thinking we employ when dealing with queries. This, we feel, would be such a major change in the pragmatic dimension of the language that the algebra would not be the same.

4.2 Context-sensitive division

Given our lack of insight into the expressive power of an algebra with correlated nested queries, the question remains whether the solution sketched in the previous section rests on powers beyond relational completeness. If so, is such a degree of complexity inherent to our type of query? Fortunately, this is an easy question to resolve. Recall that the relational calculus and the algebra are equivalent when it comes to expressive power. Hence, a calculus solution to our problem would prove that such queries are within the range of pure relational algebra. As it happens, the calculus can easily handle the query:

$$\begin{aligned}&\{e.SSN \mid EMP(e) \text{ and}\\ &\quad \forall x(\text{if } PROJ(x) \text{ and } x.DNUM = e.DNO \text{ then}\\ &\qquad \exists w(WO(w) \text{ and } w.ESSN = e.SSN\\ &\qquad\quad \text{and } w.PNO = x.PNUM))\}\end{aligned}$$

Now our goal is to propose a generalized division operator that supports our particular kind of query, in a convenient manner, while staying within the boundaries of relational completeness. Let us start by exploring how query (2) could be solved using only the standard repertoire of relational algebra. The following expression should do the job:

$$\pi_{ESSN}(WO) - \\ \pi_{SSN}(\pi_{SSN,PNUM}(\sigma_{DNO=DNUM}(EMP \times PROJ)) - WO)$$

Here lies the basic idea behind our new division operator, but there are some considerations that should be taken into account.

It is reasonable to expect a division operator to take two relations as arguments. Note that there are three relations involved in the expression above, but this can be circumvented by a touch of trickery. Note also that the expression includes a σ-operator. To accommodate this, we let the σ-condition be a parameter to the new division operator.

The roles of targets and common attributes are retained for the new division, but contextual attributes may also appear. By letting the list of targets be a parameter to the operator, the roles of attributes are fully specified: the targets are given explicitly, the common attributes can be identified from the relation-schemas and the remaining attributes are contextual.

Given that contextual attributes are now allowed on the scene, and that the condition parameter provides a channel for feeding extra information into a division, it seems that we have what we need for our specific problem. So, the time has come to spell out the details formally and define ourselves a suitable operator.

Let R and S be relation-schemas with attribute sets X and Y respectively. Suppose $T \subseteq X$, $X \cap Y = C$ and $C \cap T = \emptyset$. Let φ be a boolean condition over $X \cup Y$. Now we define *context-sensitive* division $R \frac{T}{\varphi} S$ to be equivalent with

$$\pi_T(R) - \pi_T(\pi_{T \cup C}(\sigma_\varphi(\pi_{X-C}(R) \times S)) - \pi_{T \cup C}(R))$$

Here, T corresponds to the targets, while C contain the common attributes. For the intended use, T and C will be non-empty.

Finally, equipped with this new division operator we can express query (2) conveniently as follows

$$EWO \leftarrow EMP \ltimes_{SSN=ESSN} WO \\ WO'(SSN, DNO, PNUM) \leftarrow \pi_{SSN,DNO,PNO}(EWO) \\ RESULT \leftarrow WO' \frac{SSN}{DNO=DNUM} PROJ$$

It is easily seen that the new operator is indeed a generalization of $\div$: If no contextual attributes are present and the condition evaluates to true in all cases, then context-sensitive division boils down to being the same as ordinary division.

For the record, query (2) can easily be formulated by means of relational comparison operators.

$$\pi_{SSN}(\sigma_{\pi_{PNUM}(\sigma_{DNUM=DNO}(PROJ)) \subseteq \pi_{PNO}(\sigma_{ESSN=SSN}(WO))}(EMP))$$

Note that this also involves correlated nested queries.

As it happens, the new division operator is suitable for solving another class of universal queries. Let R be a relation with a single number-valued attribute N. The query for the least number in R would typically be solved by means of aggregate functions. But, in fact, this *is* a universal query and can easily be formulated as such in SQL and relational calculus. In the algebra however, it seems that $\div$ is not up for the task. Luckily, context-sensitive division offers a fairly convenient solution:

$$(\rho_{(M)}(R) \ltimes_{M=N} R) \frac{M}{M \geq N} R$$

5 Conclusion

Our discussion has revolved around relational algebra and certain universal queries. Today, SQL is by far the dominant query language for relational databases. Despite this, the algebra still holds ground and one reason is that it is *different* from SQL, representing a different mode of thinking about and handling queries. In this respect, too many 'enhancements' with elements borrowed from SQL could make the algebra obsolete. Our approach has been conservative, wanting to preserve the differences between the languages in order to preserve the relevance of both. Accordingly, we have remained faithful to the division-approach to universal queries rather than to adopt mechanisms from the realm of SQL, such as relational comparison operators or correlated nested queries.

We have identified a type of universal query for which the original division operator seems to be of little use. This parallels other investigations into universal queries and relational algebra. For instance [1, 3], which propose generalized division operators to mend shortcomings of $\div$ in connection with certain other types of universal queries. However, it seems that in these cases the original division operator has been generalized beyond recognition. One might ask whether these should be referred to as *division*-operators at all.

For our kind of problematic query, we propose a generalized division operator while making an effort to retain the fundamental idea of division. In doing so we demonstrate that the type of query we have discussed need not be seen as yet another nail in the coffin for the division-approach to universal queries.

References

[1] J. V. Carlis, HAS, a Relational Algebra Operator or Divide is not Enough to Conquer, *Proceedings of the second international conference on data engineering*, IEEE Computer Society, 1986

[2] E. F. Codd, Relational completeness of data base sublanguages. In Justin, R. J. (ed) *Data base systems*, Courant computer science symposia series 6, Englewood Cliffs, N.J.: Prentice-Hall, 1972

[3] M. Dadashzadeh, An improved division operator for relational algebra, *Information systems* 14(5), 1989

[4] C. J. Date, On relational division, *Discussion on Database Debunkings website* www.dbdebunk.com, 2003

[5] C. J. Date, *An introduction to database systems*, Addison Wesley, 2004

[6] C. J. Date, H. Darwen, Into the great divide, *Relational database writings 1989-1991*, Addison Wesley, 1992

[7] C. J. Date, H. Darwen, Relation-valued attributes or Will the real first normal form please stand up? *Relational database writings 1989-1991*, Addison Wesley, 1992

[8] C. J. Date, H. Darwen, Toward a reconstituted definition of the relational model Version 1 (RM/V1), *Relational database writings 1989-1991*, Addison Wesley, 1992

[9] C. J. Date, H. Darwen, Divide and Conquer?, *Relational database writings 1991-1994,* Addison Wesley, 1994

[10] G. Graefe, R. L. Cole, Fast algorithms for universal quantification in large databases, *ACM Transactions on database systems*, Vol. 20, No 2, 1995

[11] R. Elmasri, S. B. Navathe, *Fundamentals of database systems* , Addison Wesley, 2003

[12] M. Kifer, A. Bernstein, P. Lewis , *Database systems*, Addison Wesley, 2004

[13] A. Klug, Equivalence of relational algebra and relational calculus query langages having aggregate functions, *Journal of the ACM* 29 No. 3, 1982

[14] L. Libkin, Expressive power of SQL, *Theoretical Computer Science* 296, 2003

[15] L. I. McCann, On making relational division comprehensible, *ASEE/IEEE Frontiers in education conference*, 2003

[16] A. Trovåg, On the expressive convenience of relational algebra, *Master thesis*, Department of information and media studies, University of Bergen, In preparation 2004

[17] J. Ullman, *Principles of Database and Knowledgebase systems, vol. 1*, Computer Science Press, 1988

[18] C. Welty, D. W. Stemple, Human factors comparison of a procedural and a nonprocedural query language, *ACM Transactions on database systems*, vol. 6. No. 4, 1981