Parametric databases: seamless integration of spatial, temporal, belief and ordinary data

Shashi K. Gadia
Department of Computer Science
Iowa State University
Ames, Iowa 50011-1040
gadia@cs.iastate.edu; (515) 294-4377

Abstract. Our model, algebra and SQL-like query language for temporal databases extend naturally to parametric data, of which spatial, temporal, spatio-temporal, belief and ordinary data are special cases.

1. INTRODUCTION.

The main characteristic of parametric data is that there is an underlying parametric space, and data values vary from one point in the parametric space to another; thus, data values are simply (partial) functions from the parametric space. Temporal data is a special cases of parametric data where the parametric space consists of instants of time. An example of temporal data value is department history of an employee. Spatial data is clearly another special case of parametric data.

Perhaps less obvious case of parametric data is belief data. The parameter space in case of belief data consists of people. We have encountered two interesting cases of belief data. One form of belief data arises in every-day record keeping which contain errors. In human experience errors are illusive: something known to be "correct" may be found "incorrect" tomorrow, and yet a day later we may find that the original information was in fact "correct". Belief data also arises in security databases, where there is a hierarchy of users, and upper users have different and confidential version of reality compared to lower users. In this case the users are points in the parametric space.

We introduce the concept of dimension alignment to integrate any mix of different types of parametric data into one seamless framework. The concept of dimension alignment extends to ordinary data treating it as a degenerate case of parametric data.

Fragile integrity of parametric data. It is important that the points in the parametric space and the values should not be cross-spliced. We elaborate this idea by an example. Consider a parametric relation $r = \{(a_1,b_1,p_1), (a_2,b_2,p_2)\}$ with the two ordinary attributes A and B, and an attribute POINT over the underlying parametric space. The meaning of the tuple (a_1,b_1,p_1) is that a_1,b_1 is valid at the point p_1. Similar remark applies to the tuple (a_2,b_2,p_2). If we consider A, B and POINT as independent attributes, we can compute the relation $s = \Pi_{AB}(r) \times \Pi_{POINT}(r) = \{(a_1,b_1,p_1), (a_1,b_1,p_2), (a_2,b_2,p_1), (a_2,b_2,p_2)\}$. The tuple (a_1,b_1,p_2) of s says that a_1 and b_1 are valid at point p_2 in the parametric space. Therefore the query s reports information not present in the database. We remark that in the temporal case of parametric data a relation such as s can be computed in some temporal query languages. However, in our model attributes A and B will be represented as functions from the parametric space, there is no need for an attribute such as POINT, and a relation similar to s cannot be ~~computed~~.

Associative navigation. In databases we retrieve information through associative navigation, i.e. through values which relate to other values; for example, the portion of John's department history during the time his salary was less than Mary's salary. If we denote salaries histories of John and Mary as functions A and B, respectively, then the subset $\{t: A(t)<B(t)\}$ of the parametric space, denoted μ, retrieves the instants when the required condition is satisfied. The required portion of John's salary history is his department history function restricted to the domain μ. Note that the parametric set $[[A\theta B]]$ defined as $\{t: A(t)\ \theta\ B(t)\}$ is the counterpart of the classical construct $A\theta B$.

Parametric elements. Suppose we are given a fixed parametric space *P*. Above we established the importance of subsets of *P*, however all subsets of *P* may not be of interest to us. We term the subsets of interest to us *parametric elements*, or *para-elements* in short. The nature of para-elements depends upon the nature of *P*, the parametric functions and queries arising in an application domain. We have only two requirements of para-elements: first, a para-element should have a finite description, and second, para-elements should be closed under set theoretic union, intersection and complementation. The closure properties allow us to deal with functions with fragmented domains without fearing that the database structure may fragment them across tuple boundaries. These domains become further fragmented when queries are executed. Para-elements also help us to treat "or", "and" and "not" of natural languages symmetrically in a query language. In other words our query languages become more user friendly.

In business applications of temporal databases, typically the parametric space [0,NOW] = {0,1, ..., NOW} of instants suffices. Supposing that every instant in [0,NOW] is to be a para-element, from the closure properties of para-elements it follows that every subset of [0,NOW] is a para-element. As another example, suppose the parametric space is $(-\infty,\infty)$, the set of all real numbers, functions are linear line segments, and that these functions are to be compared using = and ≤. In this case some [[f≤g]] evaluate to intervals. Thus para-elements are finite unions of intervals. Note that the fact that the parametric space is not a finite discrete set causes no problem. As another example suppose our database consists of polynomial functions. In that case we need approximation techniques to determine [[f=g]]. In general, the requirements in scientific applications can be considerably more complex. To model soil maps we need para-elements which can have rather complex shapes (see Figure 2.1). To model space and time varying properties, the parametric space may be three dimensional.

Dimension alignment. Problem of dimension alignment arises when a user has to deal with data which is a mix of different types of parametric data. We solve this problem in an interesting way without passing the complexity to a user. For example, a user can write reg $\cap$ μ, even if reg is purely a spatial region, and μ is a temporal element. The system does the dimension alignment automatically and dynamically as needed, by padding the whole spaces in the missing dimensions of operands. In this case reg $\cap$ μ will be treated as (reg $\times$ T) $\cap$ ($R \times \mu$), where T is the universe of time, and R is the universe of space. As another example, if a is an ordinary constant, then in the context of spatial data a will be treated as $\langle R \quad a\rangle$, the constant function whose value is a at every point in R.

This paper is only about data models and querying. We do not give a detailed bibliography. Instead, we cite works [Sn86,Ta86] in temporal databases, [AS92, Gut88,RFS88,SV89] in spatial databases and [SW92] in security databases. Our basic model for temporal databases has appeared in [Ga87,GY88].

2. AgriDB: AN EXAMPLE.

We introduce an application, called AgriDB, in agriculture environmental management. The application is a mix of spatial information, spatio-temporal information and ordinary data. The purpose of the application is to determine and make decisions about the environmental consequences of using various chemicals in agriculture. The following is a detailed description of the application. A summary and spatial maps where this application experiment is conducted are shown in Figure 2.1.

2.1. Description of the application. We are given a fixed spatial region, which we can assume to be a bounded portion of a plane. In this region varying soil textures prevail. In the same region several different crops are being grown with different tillage methods. Because of various reasons, varying from increasing crop production to pest-control, some chemicals are applied to the whole region. Some of these chemicals seep through the soil and contaminate ground water. The seepage depends on the chemical being applied, the crop type, the tillage method and the soil texture. We are given some U.S. Environmental Protection Agency (EPA) data about chemicals. (This data is hypothetical.) It specifies the maximum contaminant level and minimum detectable limit allowable concentrations of the chemicals in ground water. We are given some wells, where readings are taken from time to time to monitor contaminants in the ground water.

For this application we pair up the wells such that each well belongs to one and only one pair. It is customary to treat the pair of wells as a single entity and classify the wells in the pair as either up-gradient (u/g) or down-gradient (d/g) depending upon the direction of ground water flow. The direction of ground water flow is from the up-gradient well to the down-gradient well. The concentration of the chemicals in the down-gradient well is affected by the dilution effect due to the up-gradient well and hence there is a need to classify the wells as up-gradient (u/g) and down-gradient (d/g) to take this effect into account. Time is assumed to be acyclic.

3. OUR MODEL.

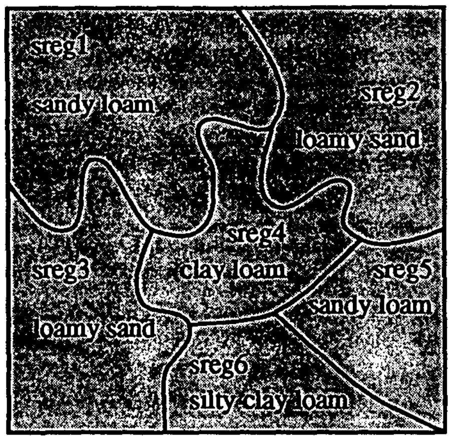
Attribute values are partial functions from para-elements. A tuple is imply a concatenation of parametric assignments. (We impose certain important requirements on our tuples but these are not covered here.) A *parametric relation* r over R, with K⊆R as its *key*, is a finite set of non-empty tuples, such that no key attribute value of a tuple changes from one parametric point to another, and no two tuples agree on all their key attributes. Figure 3.1 shows how AgriDB of previous section is modeled as a spatio-temporal database, called RelAgriDB.

The Query Language ParaSQL. We do not discuss the query language ParaSQL (read para-S-Q-L) in detail here. The most interesting statement in ParaSQL is the select statement. It has the following form

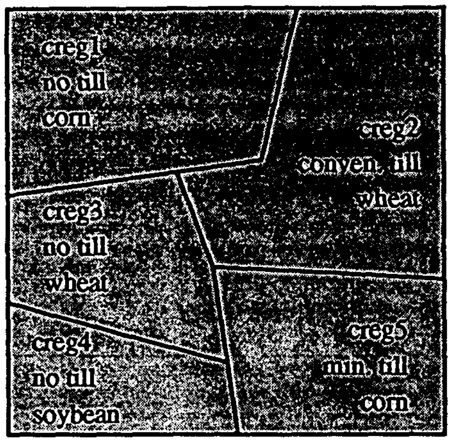
```
select attributeList
restricted to paraExpression
from relationList
where booleanExpression
```

The semantics of the select statement is as follows. First a (virtual) cross product of relations in the relationList is computed. Next we iterate over the tuples τ of the cross product. We check to see if τ satisfies booleanExpression. If not, τ is rejected. Otherwise τ is substituted in the paraExpression, which evaluates to a para-element denoted μ. Now we calculate the restriction of τ to μ and retrieve attributes in the attributeList.

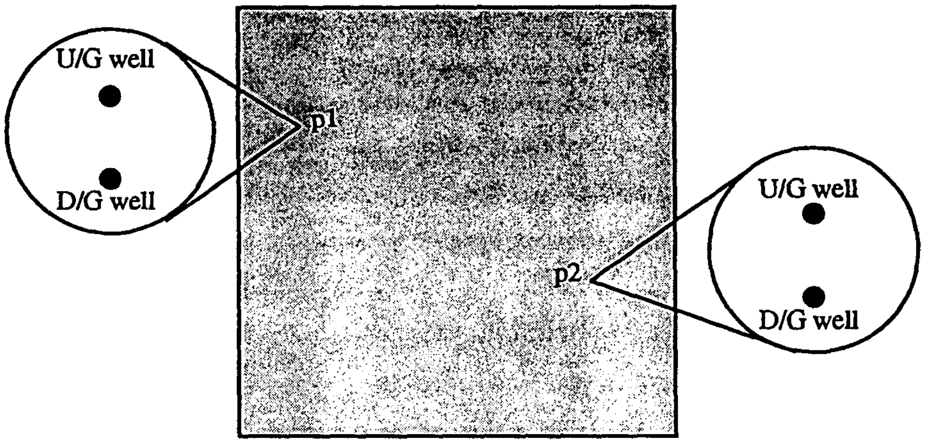
The select statement in ParaSQL works in a manner similar to its counterpart in classical databases. There are two main differences. First, we have a new "restricted to" clause. As our attribute values are functions encoding a lot of information, this clause allows us to



Soil texture map



Crop and tillage map



Wells map

- Information about soil texture. This data is only spatial and it is time independent. A map of soil texture is displayed above.
- Information about crops and tillage methods. This data is also only spatial and it is time independent. A map of this data is also displayed above.
- EPA data. This data specifies the maximum contaminant level and minimum detectable limit allowable concentrations of the chemicals in ground water. This data is ordinary; it is space independent, as well as time independent.
- Chemical readings. This data consists of readings taken from various wells at different times. It is space and time dpendent. A map showing the location of wells is displayed above.

Figure 2.1. AgriDB: A case study

TEXTURE
$sreg_1 \cup sreg_5$ sandy loam
$sreg_2 \cup sreg_3$ loamy sand
$sreg_4$ clay loam
$sreg_6$ silty clay loam

(a) The soil relation

CROP-NAME	TILLAGE
$creg_1 \cup creg_5$ corn	$creg_1$ no till $creg_5$ min till
$creg_2 \cup creg_3$ wheat	$creg_2$ no till $creg_3$ conven till
$creg_4$ soybean	$creg_4$ no till

(b) The crop relation

CHEM-NAME	MAX	MIN
atrazine	3	0.05
simazine	35	0.05

(c) The epa relation with concentration in parts per billion (ppb)

CHEM-NAME	U/G-CONC	D/G-CONC
$p_1 \times [0,NOW]$ atrazine $\cup\ p_2 \times [0,NOW]$	$p_1 \times [0,NOW]$ 1.0 $p_2 \times [0,5]$ 1.5 $p_2 \times [6,NOW]$ 3.5	$p_1 \times [0,NOW]$ 0.9 $p_2 \times [0,10]$ 1.4 $p_2 \times [11,NOW]$ 2.9
$p_1 \times [0,NOW]$ simazine	$p_1 \times [0,9]$ 10.0 $p_1 \times [10,NOW]$ 12.2	$p_1 \times [0,NOW]$ 9.2

(d) The chems-in-wells relation with concentration in parts per billion (ppb)

- The soil relation. This relation is spatial, but it is time independent. The key is TEXTURE.
- The crop relation. This relation is also spatial, but time independent. The key of the relation is CROP-NAME.
- The epa relation. This relation is space and time independent, and CHEM-NAME is its key.
- The chems-in-wells relation. For each chemical, it shows readings taken in various wells at different times. It is spatio-temporal. The key of this relation is CHEM-NAME.

Figure 3.1. RelAgriDB: the relational design for AgriDB of Figure 2.1

restrict our retrieval to the parametric domain specified by paraExpression. The second difference is that our select statement is very powerful. This is because parametric expression and boolean expression can themselves be very powerful. In fact they have a sublanguage of their own, covered next.

A sublanguage for associative navigation. The associative navigation in our model is done through para-expressions and boolean expressions, defined as follows.

- A constant para-element (e.g. $reg \times [0,5]$) is a para-expression.
- If A is an attribute, then [[A]] is a para-expression which evaluates to the domain of A.
- If A and B are attributes and θ is an operator, then $[[A\theta B]]$ is a para-expression. This extracts points in the parameter space where A and B are in θ-relationship.
- If e is a relational expression, then [[e]] is a para-expression, whose value is the union of domains of tuples in the relation computed by e. For example, [[crop]] is $creg_1 \cup creg_2 \cup creg_3 \cup creg_4 \cup creg_5$. This construct is a source of powerful nesting among ParaSQL expressions. It can also be used by itself as a query.
- If μ and ν are para-expressions, then so are $\mu \cup \nu$, $\mu \cap \nu$, $\mu - \nu$ and $\neg\mu$.
- If μ and ν are para-expressions, then $\mu \sqsubseteq \nu$ is a boolean expression.
- We define $A\theta B$ to be an abbreviation of the boolean expression $[[A\theta B]] \neq \emptyset$.

- If f and g are boolean expressions then so are $f \vee g$, $f \wedge g$ and $\neg f$.

Now we give a fairly complex example involving all four relations in RelAgriDB of Figure 3.1.

Example. The query *find the information about the wells, which are located in the region where soybean is grown, soil texture is of type clay loam, and for which the d/g concentration of atrazine exceeds the maximum allowable concentration* can be expressed as follows.

```
select chems-in-wells.*
restricted_to [[TEXTURE=clay loam]] ∩
    [[CROP-NAME=soybean]] ∩
    [[chems-in-wells.DB-CONC > epa.MAX]]
   from crop, soil, epa, chems-in-wells
where chems-in-wells.CHEM-NAME = atrazine and
   chems-in-wells.CHEM-NAME = epa.CHEM-NAME
```

The from clause consists of a mix of inputs: crop and soil are spatial, epa is ordinary, and chems-in-wells is spatio-temporal. They will all be aligned to be spatio-temporal.

4. STRUCTURES OF RELATIONS.

Two parametric relations are said to be *weakly equal* if their restrictions to same points are equal. Weak equality between parametric relations is weaker than the absolute equality. The concept of weak equality also extends to language constructs. For example, we define a (binary) operation $\oplus$ to be *weak* if weakly equal operands produce weakly equal results; more specifically if whenever r, s, are weakly equal to r', s', respectively then so are $r \oplus s$ and $r' \oplus s'$. Examples of weak operators abound. For example union, intersection and difference of parametric relations are weak operators. In fact [Ga87] shows that every classical operator gives rise to a weak operator in temporal databases. Because our parametric elements a priori have finite descriptions, this fact is easily extended to parametric databases.

Weak operators are important, but as a rule they have little bearing on the syntax and semantics of parametric databases. However, the weak operator [[r]], which retrieves the parametric domain of a relation r, is an important exception to this rule and plays a pivotal role in the syntax of parametric query languages. This is because it participates as a para-expression making our sublanguage for tuple navigation highly recursive and powerful. It also makes ParaSQL more user friendly; in the absence of this construct some selection like (sub)queries of natural languages would become joins in parametric query languages. Barring the weak operator [[r]], as a rule what makes parametric query languages nontrivial extension of query languages for classical databases are the non-weak constructs. In ParaSQL, the most important non-weak construct is $\mu \sqsubseteq \nu$, and because this construct is used in the where clause, our "where" is not a weak construct. The construct $\mu \sqsubseteq \nu$ makes sense in any parametric database irrespective of the type of parameter space.

5. INCOMPLETE INFORMATION.

We have given a model and algebra for temporal databases, which also extends to arbitrary parametric databases. In this model the counterpart of a para-element is a partial para-element, which is a pair of para-elements (l,u), such that $l \sqsubseteq u$. It says that we are sure of existence of an object in the parametric domain l, and also know that the object does not span beyond u. An attribute value is of the form (A,l,u), where l and u are as before, and A is a parametric function whose domain is a subset of u. This is a powerful and interesting formulation. We have shown that the structure of our query language remains the same as in the case of complete information. Generalizing the results of [Bi84] to parametric databases, we prove that our algebra operators are reliable, and they cannot be strengthened further to reduce uncertainty without sacrificing reliability. This work is formal in nature; a preliminary version for the temporal case, not taking the operator [[r]] into account, has appeared in [GNP92].

6. BELIEFS.

We have stated that belief data arises in errors and security databases. [BG90] formalizes a concept of error, the challenge there is that incorrect information can corrupt identity of objects making the meaning of information fragile. We show how we can let a belief propagate through algebraic operators. Belief data arising is security databases has been studied in [Ga91]. In both cases ParaSQL works.

7. ALGEBRAIC OPTIMIZATION.

In [NG92] we give an approach for algebraic optimization. We are extending it to ParaSQL. This amounts to treating the problem of optimization of different forms of parametric data generically. However, such optimization has to take the quantitative variations in data sizes in to account. For example spatial selection will perhaps return a bigger relation than a temporal selection.

8. CONCLUSION.

We have shown the viability of our approach by treating a varied class of database applications under the unified banner of parametric data. In our model, the integrity of parametric information is automatically guaranteed. Because we can store the whole description of a real world object in a single record, user queries are higher level and closer to a natural language.

Our model achieves a greater physical data independence; it allows parametric regions to be viewed as sets of points which are manipulated through set theoretic operations. Para-expressions, the syntactic counterpart of para-elements, are structurally independent of the underlying parametric dimension; this helps us eliminate seams across types and dimension boundaries in parametric data. Para-expressions make it possible for a user to treat "or", "and" and "not" of natural languages symmetrically. Para-expressions allow us to express queries which are selections in a natural language as selections in a query language.

To provide the user a seamless view of data, the boundaries between spatial, temporal, spatio-temporal and ordinary data are removed by dynamic alignment. Such alignment is done at the query level, and it is automatically handled by the system. This would allow us to integrate parametric multi-databases seamlessly. Another form of alignment is coordinate alignment. This arises when we have data with different resolutions. For example, we may have low resolution spatial database for the state of Iowa and high resolution database in the Story county. By aligning the coordinate systems of the two parameter spaces we should be able to obtain a seamless database with high resolution data for Story county and low resolution data in the rest of the state. We believe such database integration is a nontrivial, interesting and open problem. One work which seems to point toward a solution to this problem is [RFS92].

Our experience shows that almost all queries we have encountered in the temporal, spatial and security database literature are expressible in ParaSQL, and often our queries are simpler. In particular, our remarks apply to temporal query languages in [Sn86,Ta86], spatial query languages in [Gu88,RFS88,SV89,AS92] and security database query language in [SW92]. Even in case we have a complex mixture of incomplete spatial, temporal, spatio-temporal and ordinary data, with different beliefs such as in security, ParaSQL will work without essential modifications in structure. This should be very reassuring to databases industry and users of parametric databases.

ACKNOWLEDGMENT

We thank U. Sunday Tim of Agricultural and Biosystems Engineering Department at Iowa State University for providing us the AgriDB application.

REFERENCES

[AS92] Aref, Walid G. and Hanan Samet. *Spatial relations and their algebra.* Tech Report CS-TR-2865, Computer Science Dept., University of Maryland, 1992.

[BG90] Bhargava, Gautam and Shashi K. Gadia. *The concept of error in a database: An application of temporal databases to databases.* Proc. COMAD 1990 Inter. Conf. on Management of Data, New Delhi, 1990.

[Bi83] Biskup, Joachim. *A foundation of Codd's relational maybe-operations.* ACM-TODS, Vol 8, 1983.

[Ga87] Gadia, Shashi K. *A homogeneous relational model and query languages for temporal databases.* ACM-TODS, Vol 13, 1988.

[GC91] Gadia, Shashi K. and Tsz. S. Cheng. *A secure data model for multilevel identities.* Unpublished.

[GC92] Gadia, Shashi K., and Vimal Chopra. *A relational model and SQL-like language for seamless query of spatial data.* Tech. Rep. TR-92-05, Computer Science, Iowa State Univ.

[GNP92] Gadia, Shashi K., Sunil S. Nair and Yiu-Cheong Poon. *Incomplete information in relational temporal databases.* Proc. 18th VLDB, 1992.

[Gu88] Guting, R. H. *GeoDrelational algebra: a model and query language for geometric database systems.* Conf. on Extending Database Technology (EDBT '88), Venice, 1988.

[GY88] Gadia, Shashi K and Chuen-Sing Yeung. *A generalized model for a relational temporal database.* Proc. 1988 SIGMOD.

[NG92] Nair, Sunil S. and Shashi K. Gadia. *Algebraic optimization in a relational model for temporal databases.* Proc. First Inter. Conf. on Information and Knowledge Management, 1992.

[RFS88] Roussopoulos, N., Faloutsos, C. and Sellis, T. *An efficient pictorial database system for PSQL.* IEEE Trans. on Software Eng., Vol 14, 1988.

[RFS92] Read, Robert L., Donald S. Fussell and Avi Silberschatz. *A multi-resolution relational database model.* Proc. 18th VLDB, 1992.

[Sn86] Snodgrass, Richard. *The temporal query language TQuel.* ACM-TODS, Vol 12, 1987.

[SV89] Scholl, M. and Voisard, A. *Thematic map modeling.* Lecture Notes in Computer Science, Vol 409, 1990.

[SW92] Smith, Ken and Marianne Winslett. *Entity modeling in MLS relational model.* Proc. 18th VLDB, 1992.

[Ta] Tansel, Abdullah U. *Adding time dimension to relational model and extending relational algebra.* Information Systems, Vol 11, 1986.