Object identity and dimension alignment in parametric databases

Tsz S. Cheng, Shashi K. Gadia and Sunil S. Nair
Department of Computer Science
Iowa State University
Ames, Iowa 50011-1040
{tcheng,gadia,snair}@cs.iastate.edu; (515) 294-4377

Abstract. We propose an object-oriented model for uniform treatment of parametric databases, of which temporal, spatial, spatio-temporal and ordinary data are special cases. The integration of different forms of parametric data is achieved through the concept of dimension alignment. Such alignment is automatically handled by the system at the query level, yeilding a seamless interface for a user. We clarify the role of object identity in obtaining a good query language for object-oriented parametric databases.

Key words: Spatial data, temporal data, parametric data, SQL, relational databases, object-oriented databases, object identity, dimension alignment.

1. INTRODUCTION.

This paper is about parametric databases, of which spatio-temporal data, spatial data, temporal data and ordinary data are special cases. (By ordinary data we mean data which is space and time independent.)

In spatial databases, much attention has been paid to techniques for representing spatial regions [FB89, Gun88, GW87, RF88, SV89] and physical implementation aspects of spatial databases such as access methods for spatial data [BK90, Gut84, Gun89, Sa84, Sa90, SK90, SR87, LL91]. Approaches to spatial databases and spatial operators have appeared in [SV89, Gut88, OM88, AS91, CF79, CF81, CF84, Ro86, JC88], ODM89]. [GCT92T] gives a model and SQL-like query language for spatio-temporal database. Object oriented approaches have been considered in [OB89, MK88, MO86, Su92, AdS92].

Gadia and Vaishnav [GV85] introduced timestamps which are temporal elements, rather than intervals. A temporal element is a finite union of intervals. In [GV95] the timestamp in a tuple is applied at the attribute level rather to the tuple as a whole. A temporal element is a finite union of intervals. Temporal elements are closed under union, intersection and complementation and form a Boolean algebra. The use of temporal elements simplifies the query languages for temporal databases [GY91]. Pissinou and Makki [Pi91, PM92, PM93] view instance of a relationship R as a triple (x, R, y). The concept of time is incorporated in their model by adding timestamps to x and y, as well as to the relation R.

In this paper, we present an object-oriented model OOParaDB and a query language OOParaSQL. The type structure in OOParaDB is obtained by adding some high level features to the type structure in OODAPLEX of Dayal [Da89]. OOParaSQL handles a mix of spatio-temporal, spatial, temporal and ordinary data seamlessly.

We use certain subsets of parametric space, called parametric elements (or para-elements in short), as a unifying vehicle through which associative navigation is done in our query language. The only condition we assume of para-elements is that they are closed under the set theoretic union, intersection and complementation. (Complementation is with respect to a fixed parameter space in a given context.) Because of para-elements, we can store the whole description of a real world object in a single record without having to fragment it across potentially unbounded number of records. This makes queries higher level and closer to a natural language.

Para-elements, are structurally independent of the underlying parametric dimension; this helps us eliminate seams across types and dimension boundaries in parametric data. 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.

A unique characteristic of the parametric information is that properties of objects are defined at points in the underlying parametric space. The points in the parametric space and the values should not be cross-spliced. We elaborate this idea by an example. Consider a parametric database consisting of a single relation r with the scheme A B POINT, where A and B are ordinary attributes, and POINT is an attribute over the underlying parametric space. Suppose that a state of

CIKM '93 - 11/93/D.C., USA

© 1993 ACM 0-89791-626-3/93/0011 ....$1.50

r is given as $\{(a_1,b_1,p_1), (a_2,b_2,p_2)\}$. 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 = $\Pi_{AB}(r) \times \Pi_{POINT}(r)$ which evaluates to $\{(a_1,b_1,p_1), (a_1,b_1,p_2), (a_2,b_2,p_1), (a_2,b_2,p_2)\}$. The tuple (a_1,b_1,p_2) 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. In our model a relation similar to s cannot be computed.

We compare our approach to OODAPLEX of Dayal and Wuu [DW92,WD92]. We argue that OOParaSQL is higher level, more seamless and easier for expressing user queries. We also compare our own relational [CGN92T] and object-oriented query languages to show why due to type inheritance through object ids, syntactically similar queries in the two languages retrieve different results.

This paper is organized as follows. In Section 2 we introduce an application, called AgriDB, in agriculture environmental management which is used as a running example in this paper. In Section 3 we give a brief description of OODAPLEX [DW92,WD92]. We present our object-oriented model OOParaDB in Section 4, and query language OOParaSQL in Section 5. In Section 6 we compare OOParaDB to our relational model of [CGN92T] and to OODAPLEX of [DW92,WD92]. We conclude in Section 7.

2. AgriDB: A CASE STUDY IN AGRICULTURE.

In this section 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.

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) hypothetical data about chemicals. This data 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.

As is customary in agriculture, for this application we pair up the wells such that each well belongs to one and only one pair. A pair of wells is usually treated as a single entity (located at a single point) and the two wells in the pair are classified as up-gradient (u/g) and 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. OVERVIEW OF OODAPLEX TYPE STRUCTURE.

Most of the ideas presented in this paper are independent of the choice of a database paradigm. A particularly noteworthy example is dimension alignment to be introduced in Section 5. In this paper we clarify the role of object identities in the context of parametric databases, and these ideas are independent of the choice of a object oriented model. We have chosen the object oriented model OODAPLEX as vehicle to demonstrate our ideas. A brief description of OODAPLEX follows.

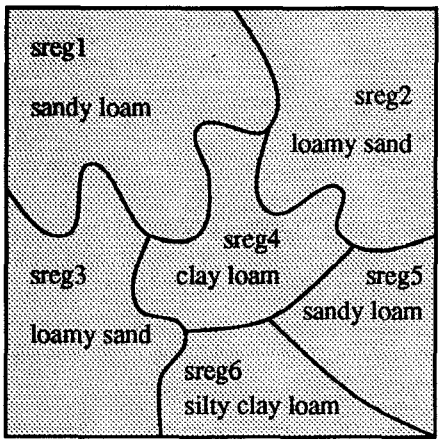
DAPLEX is a data definition and manipulation language for database systems [Sh81] which incorporates entities, and functional representation for data relationships. It supports type hierarchy and inheritance. OODAPLEX [Da89], an object-oriented extension of DAPLEX, adds abstraction, encapsulation of behavior and closure to DAPLEX, to treat tuples and collections as first class objects. In OODAPLEX, a real world entity is modeled by an object. The basic characteristic of an object is its distinct identity, which is immutable and persists for as long as the entity exists, and is independent of the object's internal value. Objects with similar properties and behaviors are grouped into *types*.

Properties of objects, relationship among objects, "computed" properties and relationships, and object behavior are uniformly modeled by functions. An example is shown below.

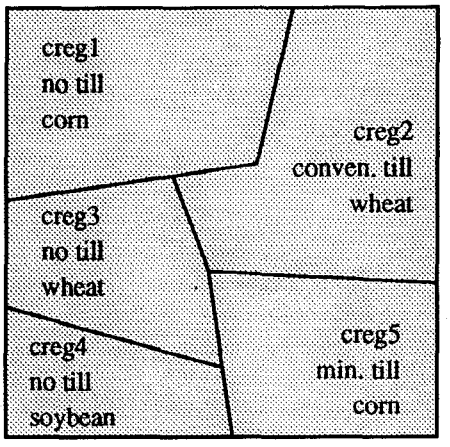
```
type person is object
    function NAME (person → string)
    function BIRTHDATE (person → date)
    function GENDER (person → sex)
```

Subtypes of types may be defined, forming *inheritance (isa) hierarchies*. It means that the set of instances of a subtype is a subset of the set of instances of its supertype and any function defined over a supertype can be applied to an object of its subtype[1]. A type may have more than one subtype and more than one supertype. A model for temporal data, also based on OODAPLEX, has been given in [WD92, DW92]. We use the term OODAPLEX for these later works.

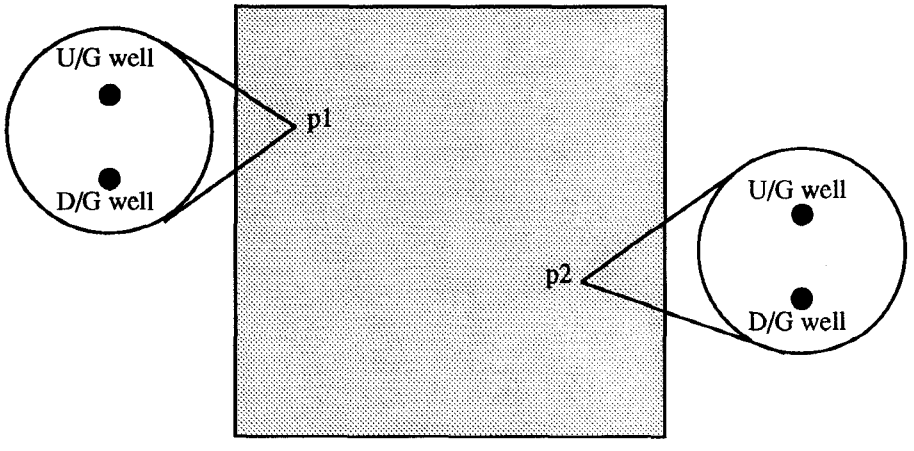
1. In our model we achieve this *isa* relationship through the use of a different construct called *object fragments*, which will be introduced soon.



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. In this example application, crops are not rotated through the year; thus crop information is only spatial, and time independent. A map of this data is 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 dependent. A map showing the location of wells is displayed above.

Figure 2.1. AgriDB: A case study

4. OUR MODEL.

In this section we introduce our model, OOParaDB for parametric databases. OOParaDB can handle various forms of parametric data (e.g. spatial, temporal, spatio-temporal and ordinary data). The query language OOParaSQL is presented in the next section. The model incorporates the concept of object identity, and captures the semantics of parametric objects through type hierarchy.

A relational model for parametric databases has been given in [CGN92T]. OOParaDB is based on our relational model and the type structure of OODAPLEX [Da89]. In Section 6.1, we compare OOParaDB with our relational model and with OODAPLEX.

4.1. Parametric elements. We assume an underlying universal parametric domain P. The user views it as a set of points in the parametric space. We postulate that certain sub-

sets of P, called *parametric elements*, are of interest to users, and they are closed under union ($\cup$), intersection ($\cap$), subtraction (-) and complementation ($\neg$). For example if the parametric domain is the set of time instants T = [0,NOW] = {0,1,...,NOW}, then the parametric elements are *temporal elements* [Ga88, GY88]. If the parametric domain is the spatial domain R then the parametric elements are *spatial elements* [GC92T]. If the parametric domain is $T \times R$ we get *spatio-temporal elements*. The closure properties of parametric elements make it possible to store the whole information about a stored or computed object in a single tuple. This results in a simpler query language, as discussed in more detail in Section 4. Note that we do not make specific assumptions about the constitution of P. We do not assume that P is discrete or continuous. Our main hypothesis is that parametric elements should have some reasonable description. In our model parametric elements are immutable type.

For ease of reading, we first show how AgriDB is represented as an object-oriented spatio-temporal database and then introduce the terminology.

4.2. The AgriDB application. Figure 4.1 shows an object-oriented design for AgriDB, our running example introduced in Section 2. The design includes two ordinary (space/time independent) types, two spatial types and one spatio-temporal type. The property functions forming the key are underlined (keys are introduced later). An instance of the schema of Figure 4.1 is given in Figure 4.2.

```
type chemical is object
    function CHEM-NAME (chemical → name-type)
type epa is chemical
    function MAX (chemical → conc-type)
    function MIN (chemical → conc-type)
type chems-in-wells is chemical
    function U/G-CONC (chems-in-wells →
                (spatio-temporal-element → conc-type))
    function D/G-CONC (chems-in-wells →
                (spatio-temporal-element → conc-type))
type soil is object
    function TEXTURE (soil → (spatial-element → texture-type))
type crop is object
    function CROP-NAME(crop → (spatial-element→ name-type))
function TILLAGE (crop → (spatial-element→ tillage-type))
```

Figure 4.1. An object-oriented design for AgriDB.

4.3. Object identity. An object is identified by its *object identity (oid)*, which is independent of its internal values and invisible to the user. This oid is unique to the object and persists as long as the object exists.

4.4. Property function. A *parametric function* is a mapping from an instance of para-element into the domain of the property (attribute) value. A *property function* is a mapping from an instance of the object type into parametric functions.

For example, in Figure 4.1, the function U/G-CONC is a property function. It takes an object of chems-in-wells type as input argument and returns a spatio-temporal function, which takes a spatio-temporal point and returns the concentration of the chemical (the object represented) in the up-gradient well.

4.5. Parametric type and type hierarchy. A *parametric type* can be defined by specifying its supertypes and a list of property functions. Objects of subtype inherit property functions from its supertypes. For examples, in Figure 4.1, chems-in-wells type is a spatio-temporal type consisting of the property functions U/G-CONC and D/G-CONC and chemical type is an ordinary type consisting of the property function CHEM-NAME. Since chems-in-wells type is a subtype of chemical type, any object of chems-in-wells type can inherit the property functions defined for chemical type.

4.6. Keyed types. We require that each type have some property functions as *key* functions. If a type does not have any key function of its own, it will inherit the key functions from its supertype. A key function takes only one value for each object, and this value cannot be taken by another object.

Consider chems-in-wells type in Figure 4.1. Since it has no key functions of its own, it inherits the key function CHEM-NAME from its supertype, which is chemical type. Thus, an object of the chems-in-wells type can be identified by the chemical name. As another example, we note that each object of crop type has a distinct CROP-NAME as it is the key function.

4.7. Object aliasing. In our model, an object can be an instance of only one type. However, two objects could represent the same entity in the real world at some points in the parametric space. To capture this relationship, we define the notion of *object aliasing*. An object o is an *object alias* (or simply *alias*) of an object o' at parametric element μ, if they both represent the same entity in the real world at μ. For example, in Figure 4.2, object c_2 of chemical type is an alias of object e_2 of chems-in-wells type at $p_1 \times$[0,NOW]. Likewise, object c_2 is an alias of object e_2 at $p_1 \times$[0,NOW]. When querying, this notion object aliases is completely hidden from the user; he/she thinks that an object and its fragment(s) are the same object at some parametric element μ. Note that an object of a subtype will always have alias(s) in the supertype, and the alias relationship is symmetric and transitive.

4.8. Parametric domain. The parametric domain of property function P of an object o, $[\![P(o)]\!]$, is defined as the domain of the corresponding parametric function, P(o). Motivated from our relational model [CGN92T], *homogeneity* is also assumed in our model: this says that $[\![P(o)]\!]$ is the same for every property P of an object o. The parametric domain of an object of type T, $[\![o]\!]$, is then $[\![P(o)]\!]$ for any property function P in T, because of the homogeneity assumption. The parametric domain of a set of objects O of type T, $[\![O]\!]$, is the union of $[\![o]\!]$ for all instances o in O. Now we can formally introduce *type inheritance restriction*: the parametric domain of an object in a subtype is a subset of the parametric domain of the corresponding object in a super

OID	CHEM-NAME
c_1	Atrazine
c_2	Simazine
c_3	Iodine
c_4	Lead

(a) chemicals - objects of chemical type

OID	OID_{chem}	MAX	MIN
d_1	c_1	3	0.05
d_2	c_2	35	0.05
d_3	c_3	20	0.05

(b) epas - objects of epa type

OID	OID_{chem}	U/G-CONC	D/G-CONC
e_1	p_1×[0,NOW]∪ p_2×[0,NOW] c_1	p_1×[0,NOW] 1.0 p_2×[0,5] 1.5 p_2×[6,NOW] 3.5	p_1×[0,NOW] 0.9 p_2×[0,10] 1.4 p_2×[11,NOW] 2.9
e_2	p_1×[0,NOW] c_2	p_1×[0,9] 10.0 p_1×[10,NOW]12.2	p_1×[0,NOW] 9.2

(c) well-chems - objects of chems-in-wells type

OID	TEXTURE
s_1	$sreg_1$∪ $sreg_5$ sandy loam
s_2	$sreg_2$∪ sreg3 loamy sand
s_3	$sreg_4$ clay loam
s_4	$sreg_6$ silty clay loam

(d) soils - objects of soil type

OID	CROP-NAME	TILLAGE
r_1	$creg_1$∪ $creg_5$ corn	$creg_1$ no till $creg_5$ min till
r_2	$creg_2$∪ $creg_3$ corn	$creg_2$ no till $creg_3$ conven till
r_3	$creg_4$ soybean	$creg_4$ no till

(e) crops - objects of crop type

Figure 4.2. Instances of AgriDB.

type [DW92,WD92].

4.9. Restructuring. Our operators are high level and they automatically invoke restructuring. Restructuring allows object fragments residing in operands to collapse together to form new objects. In addition we introduce an operator to do explicit restructuring. Explicit restructuring is sometimes necessary to change the key of a type before it can be queried. We illustrate the idea of restructuring through an example.

Example 1. The key of crops set is CROP-NAME. Suppose we want to change this key to TILLAGE. This requires restructuring of the crops set.

5. QUERYING IN THE MODEL.

In this section we introduce a SQL-like language OOParaSQL to query parametric data. Although a query language is available in OODAPLEX, we feel that it does not support high level constructs arising in parametric data. Because our query language OOParaSQL is higher level and specially designed for parametric data, it is user friendly, and yet very powerful. Although the problem of impedance mismatch exists in OOParaSQL for application programmers, we only consider it a minor. (Impedance mismatch also seems to exist in OODAPLEX.[1])

There are two important features of OOParaSQL: dimension alignment and restructuring. The dimension alignment allows a user to query different forms of parametric data seamlessly in the same query. Restructuring automatically invokes the concept of object aliasing to form objects returned by a query.

5.1. Dimension alignment. It is sometimes necessary to align dimensions while expressions are being evaluated. For example, a user can write reg ∩ μ, where reg is purely a spatial region, and μ is a temporal element. The system does the dimension alignment automatically as needed, by padding the whole spaces in the missing dimensions of operands. In

1. Although there is a "for" statement in OODAPLEX, it is different from the "for" statement in programming languages such as PASCAL.

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. An interesting corollary of this phenomenon is that [[.]] operator can be applied to ordinary data. Thus, in the context of spatial alignment, if a is an ordinary constant, then [[a]] evaluates to R, and in the spatio-temporal context it evaluates to $R \times T$. Our concept of alignment, presented here yields substantial dividends in terms of ease of querying. We note that removing a dimension can be problematic. The reason is that if we remove time dimension from an attribute value such as $\langle reg_1 \times [0,5]$ red$\rangle$ to obtain $\langle reg_1$ red$\rangle$, in subsequent alignments it will be treated as $\langle reg_1 \times T$ red$\rangle$. This amounts to manufacturing new information not present in the database, which should be avoided. Similar problems occur if we remove space dimensions.

OOParaSQL is three sorted. It consists of *parametric expressions*, *boolean expressions* and *set expressions*. These expressions are mutually recursive, and evaluate to parametric elements, boolean values TRUE or FALSE, and sets of objects, respectively.

5.2. Parametric expressions. The para-expressions are defined as follows:

- A constant para-element is a para-expression. An example of such an expression is reg$\times$[0,5] which is spatio-temporal.
- If P is a property function, then [[P]] is a para-expression which evaluates to the parametric domain of P. For an illustration, let us consider the set well-chems in Figure 4.2. Then [[U/G-CONC]] is a spatio-temporal expression. For the object e_1 in well-chems, the expression [[U/G-CONC]] evaluates to $p_1 \times [0,NOW] \cup p_2 \times [0,NOW]$.
- [[P 1θP2]] is a parametric expression where P1 and P2 are property functions. For an illustration, let us again consider the set well-chems in Figure 4.2. Then [[U/G-CONC $\neq$ 1.0]] is a spatio-temporal expression. As U/G-CONC is spatio-temporal, the ordinary constant 1.0 will be aligned to become spatio-temporal function taking 1.0 as its value on $R \times T$. For the object e_1 in well-chems, [[U/G-CONC $\neq$ 1.0]] evaluates to $p_2 \times [0,NOW]$.
- If e is a set expression, then [[e]] is a para-expression, whose value is the union of parametric domains of objects in the set e. For example, [[crops]] is $creg_1 \cup creg_2 \cup creg_3 \cup creg_4 \cup creg_5$. This construct is a source of powerful nesting among OOParaSQL 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 $\neg\mu$. For example, [[U/G-CONC $\neq$ 1.0]] $\cup$ [D/G-CONC = 2.9]] is a para-expression. For the object e_1 in well-chems set in Figure 4.2 the expression evaluates to $p_2 \times [0,NOW]$.

5.3. Boolean expressions. The boolean expressions are defined as follows:

- If μ and ν are para-expressions, then $\mu \subseteq \nu$ is a boolean expression.
- We define AθB to be an abbreviation of the boolean expression [[AθB]] $\neq \emptyset$.
- If f and g are boolean expressions then so are f *or* g, f *and* g and *not* f.

5.4. Object-ids in para and boolean expressions. In our model the value of a property function can be a function from a para-element taking oids as its values. OOParaSQL allows all the constructs introduced above to be extended to oids. The only restriction is that oids can only be compared with other oids for equality.

Example 2. The construct [[DEPT(e)=m]] has been used in Section 6.1. In this construct DEPT(e) is an oid function and m is an oid.

5.5. Set expressions. Now we discuss expressions. First we introduce union, difference and intersection. Then we introduce the SQL-like select expression.

- If O_1 and O_2 be sets of objects of type T, then O_1 union O_2, O_1 difference O_2, and O_1 intersection O_2 are also sets of objects of type T. These operators are computed object-wise. The semantics of union is as follows. The semantics of difference and intersection is defined similarly.

  O_1 union O_2 is computed as follows. We first compute the union of O_1 and O_2 as a set, and then collapse each pair of objects of O_1 and O_2 which agree on all key functions into a single object. Note that this may give a run time error if the two objects being collapsed together have different value at some non-key function at the same point in the parametric space. Such an error may be indicative of some inconsistency in the database.
- The select statement in our query language is the most interesting one. In OOParaSQL it has the following form.

```
select SelectList [: Key]
restricted_to ParaExp
from TypeSets
where BoolExp
```

  Here SelectList is the list of property functions, ParaExp is a parametric expression, and BoolExp is a boolean expression. A default key of the result is automatically determined by the system, but a user has the option of explicitly specifying it.

  The above OOParaSQL statement is executed as follows. First, a virtual literal cross product of sets in TypeSets is formed. For each object o in the cross product, the boolean value BoolExp(o) is computed. If BoolExp(o) is false, the object o is rejected. Otherwise, ParaExp(o), an instance of para-element, is evaluated. If this instance is empty, the object o is rejected. Otherwise, we restrict o to ParaExp(o) to obtain an object o'. The property functions P of the resulting object o' are then retrieved. Finally, if necessary, the resulting set of objects is restructured so that its key is Key.

5.6. Query examples. To illustrate the expressive power of our query language OOParaSQL we give some examples. All our examples are for the AgriDB database (see Figures 2.1, 4.1 and 4.2). Our examples illustrate dimension align-

ment at the query level.

Example 3. Query *find the region where soil texture is of type clay loam* can be expressed as follows.

```
[[select CROP-NAME(c)
restricted_to [[CROP-NAME(c)=soybean]]
from crops c]]
```

Note that this query returns a spatial domain, which is a subset of R.

Example 4. Query *find the maximum allowable concentration of atrazine* can be expressed as follows.

```
select MAX(e)
from epas e
where CHEM-NAME(e)=atrazine
```

We note that this query returns an ordinary value as its result, with no explicit mention of spatial or time domains where it is valid.

Example 5. Query *find the information about the wells, which are located in the region where soybean is grown and 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 c
restricted_to [[select CROP-NAME(p) restricted_to
                    [[CROP-NAME(p)=soybean]] from crops p]]
    ∩ [[select s restricted_to [[TEXTURE(s)=clay loam]]
       from soils s]]
    ∩[[D/G-CONC(c) > (select MAX(e) from epas e
            where CHEM-NAME(e)=atrazine)]]
    ∩ [[CHEM-NAME(c)=atrazine]]
from well-chems c
```

We note that interesting dimension alignment takes place in the above query. For example, the queries of Examples 3 and 4 are nested here as subqueries. These nested queries return spatial domain and ordinary value, respectively. The results of these queries are aligned to spatio-temporal because of the presence of chems-in-wells their context.

6. COMPARISON WITH OTHER MODELS.

In this section we compare our model with other models closely related to ours. First, we compare our object-oriented model OOParaDB with our own relational model [CGN92T]. This comparison shows some pronounced differences between our two models, particularly due to type inheritance in OOParaDB. Then we compare our model with the temporal extension of OODAPLEX of [WD92, DW92] and argue that our query language OOParaSQL is higher level and more user friendly than the query language in OODAPLEX.

To make the above comparisons, it is enough to consider a temporal database. Our remarks are about the inherent philosophy behind these models. Temporal data is much simpler than spatial data, and our remarks only become amplified if one migrates to spatial or spatio-temporal data. We use a personnel database as our benchmark example. In our OOParaDB the type definitions for a personnel database are as follows. We give three types, person, emp and management as shown in Figure 6.1

```
type person is object
  function NAME (person → (temporal-element → name-type))
  function SSNO (person → (temporal-element → ssno-type))
  function GENDER (person → (temporal-element → gender-type))
  function DOB (person → (temporal-element → date-type))
type emp is person
  function SALARY (emp → (temporal-element → salary-type))
  function DEPT (emp → (temporal-element → management))
type management is object
  function DEPT-NAME (management → (temporal-element →
                                          name-type))
  function MANAGER (management → (temporal-element → emp))
```

Figure 6.1. An object-oriented design for personnel DB.

6.1. Comparison with our relational model. Now we compare our object-oriented model OOParaDB with our relational model [CGN92T]. To allow us to make our comparison as clear as possible, we give a relational database scheme which is syntactically closest to the design in Figure 6.1. We include three relational schemes as follows.

```
person (NAME, SSNO, GENDER, DOB)
emp (NAME, SALARY, DEPT)
management (DEPT, MANAGER)
```

Here we have two designs for a personnel database: first for the object-oriented model OOParaDB of this paper, and second for our relational model [CGN92T]. Although, these designs capture the same information, because of type inheritance in OOParaDB, the first also captures the complex semantics, *an employee is a person* and *a manager is an employee*. Now we can compare the two models at the query level. To show the subtle difference between the object-oriented and relational nature of the two models, we give the following example.

Example 6. Consider the query *give full details of managers of John*. In our object-oriented model, the above query is expressed as follows. The object MANAGER(m), in the select clause, represents one of possibly many managers of John and it is of employee type. Therefore, the query retrieves the NAME, SSNO, GENDER, DOB, SALARY and DEPT values of John's manager(s). Note that due to the restricted_to clause, each manager report is provided only during the period when he/she managed John.

```
select     MANAGER(m)
restricted_to  [[DEPT(e) = m]]
from       emp e, management m
where      NAME(e) = "John"
```

NAME	SSNO	GENDER	DOB	SALARY	DEPT

The information retrieved by OOParaSQL query

The following is a relational expression which is syntacti-

cally similar to the one above. However, it retrieves a result which is quite different from the one above. The only information the following query gives is the DEPT and MANAGER attribute values from the management relation.

```
select m.*
restricted_to [[e.DEPT=m.DEPT]]
from emp e, management m
where e.NAME = "John
```

NAME	DEPT

The relation scheme retrieved by relational query

To obtain the same result as in the object-oriented case, for the relational model, the query has to be expressed as follows.

```
select    p.*, e2.SALARY, e2.DEPT
restricted_to [[e1.DEPT = m.DEPT]]
              ∩ [[m.MANAGER = e2.NAME]]
from emp e1, management m, emp e2, person p
where    e1.NAME = "John"
```

6.2. Comparison with OODAPLEX. We compare our work to [WD92, DW92]. In [WD92, DW92] generic time type is *point* type, while in our model it is temporal-element type. Also, OODAPLEX uses primitives = and ≤ among instants, while we use the primitives = and ⊆ among temporal elements. Our primitives are higher order leading to simpler queries. Moreover, our primitives readily extend from temporal databases to parametric databases. Now we give two examples to illustrate our point.

Example 7. Consider the following query taken from [WD92,DW92]: *list all employees who have worked for department which has ever been headed by a manager named John.* The query is expressed as follows in the two languages.

```
OOParaSQL:select NAME(e)
          from emp e, management m
          where [[DEPT(e) = m]] ≠ ∅ and
              [[NAME(MANAGER(m)) = John]] ≠ ∅

OODAPLEX: for each e in extent(employee) where
              for some d in extent(department)
                  for some t1 in extent(time)
                      dept(e)(t1) = d and
                      manager(d)(t2) = 'John'
                  print(name(e))
              end
```

Example 8. The query *give information about managers who were managers at least during [11,50]* is expressed as follows in the two languages. Note that our query makes use of the restructuring function. The construct :MANAGER changes the key of management to MANAGER and r is an object pointer for the result.

```
OOParaSQL:select MANAGER(r)
          from management: MANAGER r
          where [[MANAGER(r)]] ⊇ [11,50]

OODAPLEX:for each e in extent(emp) where
             for all t in [11,50]
                 for some m1 in extent(management)
                     name(manager(m1)(t)) = name(e)
             for each t in lifespan(e)
                 print(time:t, name:name(e),
                       dept:dept-name(dept(e)))
             end
         end
```

Even though the papers [WD92,DW92] only treat temporal data, the notion of parameterization is potentially available in OODAPLEX. In OODAPLEX the user thinks in terms of instants, where as in OOParaDB the user thinks in terms of temporal elements. At the syntactic level, temporal elements give rise to temporal expressions. This makes OOParaSQL higher level than OODAPLEX. Temporal expressions are natural, making user queries simple.

7. CONCLUSION

We have presented an object-oriented models for parametric databases. We have shown how spatial data, temporal data and ordinary data can be integrated using dynamic alignment of dimensions to achieve seamless handling across space and time boundaries. Another feature of our query language OOParaSQL treats "or", "and" and "not" of natural languages symmetrically. This is not elaborated in this paper, but can be found in [CGN92T]. The model also achieves considerable physical data independence as a user views spatio-temporal domains as sets of points and manipulates them through set theoretic operators.

The para-expressions lead to algebraic optimization which takes complex semantics of parametric data into account [NG92]. We are lead to an extensible framework which extends naturally when more complex data such as, but not limited to, incomplete information [GNP92] is considered. The concept of dynamic dimension alignment would lead to seamless integration of parametric multi-databases.

In our model it is sometimes necessary to compute a new type. For example, this is true of restructuring operator. When a new type is created, it is desirable to place it appropriately in the type hierarchy. For example, consider the set of objects of crop type in Figure 4.2(e). Here the key is CROP-NAME. Suppose we restructure this set twice, first to make TILLAGE as its key, and then to restore the old key CROP-NAME. Clearly, this is an identity operation, but the system is unaware that the type of the resulting set is the same as the one we start with. Appropriate formalization to capture this semantics in general is under study.

ACKNOWLEDGMENT. We thank U. Sunday Tim of Agricultural and Biosystems Engineering Department at Iowa State University for providing us the AgriDB application. We also thank Hanan Samet and Walid Aref for their useful comments. The relational model for spatio-temporal database reported here was developed by the second author

of this paper, and Vimal Chopra in [GC92T,GCT92T]. We also thank the anonymous referees for their helpful comments. Finally, we thank Niki Pissinou and Kia Makki for their suggestions which helped us improve our presentation.

REFERENCES

[BK90] Beckman, N. and Kriegel, H.P. *The R*-tree: An efficient and robust access method for points and rectangles.* Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Atlantic City, May 1990, pp 322-331.

[CGN92T] Cheng, Tsz S., Shashi Gadia and Sunil Nair. *Relational and object-oriented parametric databases.* Technical Report TR-92-42, Computer Science Department, Iowa State University.

[Da89] Dayal, Umeshwar. *Queries and views in an object-oriented data model.* Proceedings of Second Workshop on Database Programming Languages, 1989.

[DW92] Dayal, Umeshwar and Gene T.J. Wuu. *A uniform approach to processing temporal queries.* Proceedings of the Eighteenth International Conference on Very Large Data Bases, 1992, pp 407-418.

[FB89] Frank, A.U., and R. Barrera. *The Fieldtree: A data structure for geographic information systems.* Lecture Notes in Computer Science, Vol 409, pp 167-190.

[Ga86] Gadia, Shashi K. *Toward a multi-homogeneous relational model for a temporal database.* Proc. of the IEEE International Conference on Data Engineering, pp 390-331.

[Ga88] Gadia, Shashi K. *A homogeneous relational model and query languages for temporal databases. ACM Transactions on Database Systems*, Vol 13, pp 418-448, 1988.

[GC92T]Gadia, Shashi K. and Vimal Chopra. *A relational model and SQL-like language for seamless query of spatial data.* Technical Report TR-92-05, Computer Science Department, Iowa State University.

[GCT92T] Gadia, Shashi K., Vimal Chopra and U. Sunday Tim. *Seamless SQL-like query of spatio-temporal data and a case study in agricultural environment management.* Technical Report TR-92-21, Computer Science Department, Iowa State University.

[GNP92]Gadia, Shashi K., Sunil S. Nair and Yiu-Cheong Poon. *Incomplete information in relational temporal databases.* Proceedings of Eighteenth International Conference on Very Large Data Bases, 1992.

[Gun88] Günther, Oliver. *Efficient structures for geometric data management.* Lecture Notes in Computer Science, Vol 337, 1988.

[Gun89] Günther, Oliver. *The design of the cell tree: an object oriented index structure for geometric databases.* Proceedings of the Fifth IEEE International Conference on Data Engineering, Los Angeles, Feb 1989, pp 598-605.

[Gut84] Guttman, A. *R-trees: a dynamic index structure for spatial searching.* Proceedings of the SIGMOD conference, Boston, June 1984, pp 47-57.

[Gut88] Guting, R. H. *Georelational algebra: a model and query language for geometric database systems.* Conference on Extending Database Technology (EDBT '88), Venice, 1988, pp 506-527.

[GW87] Günther, Oliver and Wong, Eugene. *A dual space representation for geometric data.* Proceedings of the 13th International conference on Very Large Data Bases, Brighton, England, 1987, pp 501-506.

[GWJ91]Gupta, A., E. Weymouth and R. Jain. *An extended object-oriented data model for large image bases.* Proceedings of Advances in Spatial Databases, 2nd Symposium, SSD '91, Zurich, Switzerland, August 28-30, 1991, pp 45-62.

[GY88] Gadia, Shashi K and Chuen-Sing Yeung. *A generalized model for a relational temporal database.* Proc. ACM SIGMOD International Conference on Management of Data, 1988.

[LL91] Li, K.-J and R. Laurini. *The spatial locality and a spatial indexing method by dynamic clustering in hypermap systems.* Proceedings of Advances in Spatial Databases, 2nd Symposium, SSD '91, Zurich, Switzerland, August 28-30, 1991, pp 207-224.

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

[OM88] Orenstein, Jack A. and Manola, Frank A. *PROBE: spatial data modeling and query processing in an image database application.* IEEE Transactions on Software Engineering, Vol 14, No 5, May 1988, pp 611-629.

[Pi91] Pissinou, Niki. *Time in object databases.* Ph.D. thesis, Department of Computer Science, University of Southern California, Los Angeles, California, 1991.

[PM92] Pissinou, Niki and K. Makki. *A unified model and methodology for temporal object databases.* Proceedings of the International Conference on Information and Knowledge Management, 1992.

[PM93] Pissinou, Niki and K. Makki. *T-3DIS: An approach to temporal object databases.* International Journal on Intelligent and Cooperative Information Systems, Vol.2, No. 2.

[RFS88] Roussopoulos, N., Faloutsos, C. and Sellis, T. *An efficient pictorial database system for PSQL.* IEEE Transactions on Software Engineering, Vol 14, No 5, May 1988, pp 639-650.

[Sa84] Samet, H. *The quadtree and related hierarchical data structures.* ACM Computing Surveys, Vol 16, pp 187-260, 1984.

[Sa90] Samet, H. *The design and analysis of spatial data structures.* Addison-Wesley Publication Company INC, 1990.

[Sh81] Shipman, David. *The functional data model and the data language* DAPLEX. ACM Transactions on Database Systems, Vol 6, No. 1, March 1981, pp 140-173.

[SK90] Seeger, B. and Kriegel, H.P. *The Buddy-tree: An efficient and robust access method for spatial data*

base systems. Proceedings of the 16th International Conference on Very Large Data Bases, Brisbane, Australia, 1990, pp 590-601.

[Sn87] Snodgrass, Richard. *The temporal query language TQuel*. ACM Transactions on Database Systems, Vol 12, 1987, pp 247-298.

[SR87] Sellis, T., Roussopoulos, N and Faloutsos, C. *The R+-tree: A dynamic index for multi dimensional objects*. Proceedings of the 13th International conference on Very Large Data Bases, Brighton, England, 1987, pp 507-518.

[SV89] Scholl, M. and Voisard, A. *Thematic map modeling*. Lecture Notes in Computer Science, Vol 409, pp 167-190.

[Vo91] Voisard, A. *Towards a toolbox for geographic user interfaces*. Proceedings of Advances in Spatial Databases, 2nd Symposium, SSD '91, Zurich, Switzerland, August 28-30, 1991, pp 75-98.

[WD92] Wuu, Gene and Umeshwar Dayal. *A uniform model for temporal object-oriented databases*. To appear in the Proceedings of Data Engineering, 1992.