Interpolation as a Bridge Between Table and Array Representations of Geofields in Spatial Databases

Piotr Bajerski(✉)

Institute of Informatics, Silesian University of Technology
Akademicka 16, 44-101 Gliwice, Poland
piotr.bajerski@polsl.pl

Abstract. Development of database technology facilitates wider integration of diverse data types, which in turn increases opportunities to ask ad hoc queries, and gives new possibilities of declarative queries optimization. For more than a decade, work on supporting multidimensional arrays in databases has been carried out, which led to such DBMSs as rasdaman, SciDB and SciQL. However, the DBMSs lack the ability to handle queries concerning geographic phenomena varying continuously over space (called geofields) which were measured in irregularly distributed nodes (e.g. air pollution). This paper addresses this issue by presenting an extension of SQL making possible to write declarative queries referencing geofields, called geofield queries. Geofield query optimization opportunities are also shortly discussed.

Keywords: Spatial databases · Array databases · SQL · GIS · Geofield · Coverage · Interpolation

1 Introduction

We can observe how functionality migrates from GIS applications to databases. This allows for better data integration, ACID guarantees, ad hoc querying, and automatic optimization of processing, when a declarative language, such as SQL, is used. Design and implementation of conventional relational database management systems (RDBMSs) have been driven by business requirements, and, as a consequence, they only partially meet scientific community requirements [16]. Especially during remote sensing and simulations, a lot of data is gathered that is naturally stored as multidimensional arrays [12,17]. However, storing, querying and modifying such multidimensional arrays are not supported by conventional RDBMSs. Such problems led to the development of array DBMSs. The appearance and development of array DBMSs have been driven by attempts to make multidimensional arrays the first class citizens in databases, and to integrate multidimensional arrays with relations. Currently, there is work in progress on adding multidimensional arrays support to ISO SQL standard [12].

Phenomena in space can be conceptualized as fields, varying continuously over space, and/or objects, being discrete spatial entities. To emphasize connections with geographical space in the paper, following [9], the terms *geofield*

S. Kozielski et al. (Eds.): BDAS 2015, CCIS 521, pp. 427–436, 2015.
DOI: 10.1007/978-3-319-18422-7_38

and *geoobject* are used in place of the terms *field* and *object*. In the OGC specifications the term *coverage* is used as a synonym of *geofield* [6]. In the paper queries concerning geofields will be called *geofield queries*. Geofields are divided into quantitative geofields and qualitative geofields. A *quantitative geofield* is a function assigning elements of its domain values measured on interval or ratio scale. A *qualitative geofield* is a function assigning elements of its domain values measured on nominal or ordinal scale, e.g. numbers of intervals of a geofield values. The domain of geofield may be 2D or 3D spatial, 3D or 4D spatio-temporal or of a higher dimension. As far as spatial data are concerned, the most common is 2D spatial, however, in the last years significant progress has been done in spatio-temporal prediction methods [8], which allows to generalize concepts presented in the previous work [4,5].

Geofields are naturally represented by multidimensional arrays, when they are results of remote sensing or simulations on a regular grid – such a representation will be called *array representation*. However, geofields values are also often measured in monitoring networks with irregularly placed nodes, and such results are naturally stored in classical database tables – such a representation will be called *table representation*. Computing answers to geofield queries usually requires finding geofields values in places in which they had not been measured. In the case of the array representation simple and fast interpolation methods (e.g. developed for rasters processing) can be used. However, in the case of the table representation, more sophisticated and computationally intensive methods, such as Kriging, are needed [7]. A formula used for estimation of geofield values using its table representation will be called *geofield (mathematical) model*.

In the traditional approach to processing table representations of geofields, a database is only used for storing their point measurements. As a consequence, geofields processing leads to unloading the measurements and using external tools to compute array representations of the geofields. The result arrays are usually stored in a file system for further processing. Array DBMSs solve the problem partially by allowing to load the computed array representations back to the database. However, they miss the opportunity to include a conversion between table and array representations in the query optimization process.

The main idea of the presented work consists in adding operations on geofields to SQL and taking them into account during the query optimization phase. Of these operations, the conversion between table and raster representations of geofields is of key importance, as it can easily dominate the query execution time [3].

The remainder of the paper is organized as follows. section 2 outlines the Peano relation and PNR representation, as these concepts are fundamental to previous work on geofield queries optimization and influence the extension of SQL for geofield queries described in section 3. Section 3 contains the main contribution of the paper. Section 4 shortly describes how the presented SQL extension can help to speed up geofield query processing. It summarizes previously published geofield optimization rules and presents new ideas connected

with optimization of quantitative geofields. Section 5 consists of a survey of related work. The paper is concluded in section 6.

2 Peano Relation and PNR Representation

The concepts of the Peano relation and the PNR representation are presented in [5], while [4] discusses efficiency of generating the PNR representation of a qualitative geofield based on the PNR representation and a geofield variability model. A thorough discussion of the concepts and exhaustive experimental results for 2D qualitative geofields are given in [2]. This section quotes the most important informations on the Peano relation and the PNR representation, which are used in the rest of the paper.

The PNR representation is based on *discrete coordinates*, in which sides lengths of the elementary quadrants are used as units [2]. This resembles approach used in array DBMSs, however, all geofields generated in a given query share the same discrete coordinates, which simplifies computations. The elementary quadrant can be treated as a synonym of the cell in array terminology. The area in which values of the given geofield must be computed to answer a query is called *geofield context region*. Elementary quadrants are ordered by a Peano N space-filling curve, and their ranges are stored in a relation valued attribute. Such a representation is called *PNR representation*. The PNR representation of a geofield consists of the PNR representation of zones in which geofield model values belong to the same intervals.

A quantitative geofield has predefined virtual attributes: `location` and `value`. The attribute `location` provides spatial coordinates of the center of an elementary quadrant (array cell), and the attribute `value` – the geofield value computed for this location. A qualitative geofield has predefined attributes `interval`, `lBound`, `uBound` and `shape`. The attribute `interval` stores the interval number described by the given tuple, the attributes `lBound` and `uBound` store lower and upper bounds of the interval respectively, and the attribute `shape` stores Peano keys of all elementary quadrants approximating fragments of the computational space in which values of the geofield model fall into this interval.

3 SQL Extension for Geofield Queries

In this section, an extension of SQL for geofield queries is presented. It consists of a data type, called `geofield`, the `create geofield` statement, and a short description how they can be integrated with SQL dialects used in array DBMSs. Quantitative geofields are created as multidimensional arrays, so the arrays operations can be used to process them. The PNR representation of qualitative geofields may be interpreted as a quadtree (or an octree, etc.) representation of a multidimensional array, so operations on such arrays may be easily adjusted to the PNR representation. The main difference between processing multidimensional arrays in array DBMSs (see section 5) and the approach presented in this

paper is the possibility of generating on-the-fly only the needed parts of geofields during query processing in the presented approach.

Let us assume, that we have a table storing point measurements of ambient air pollution:

```
create table air_pollution (
   measurement_id integer primary key,
   pollutant_code varchar not null,
   measurement_location geometry not null,
   measurement_time timestamp not null,
   measured_value double not null,
   measurement_quality varchar not null,
   check (measurement_quality in ('ERROR', 'POOR', 'GOOD'))
);
```

3.1 Create Geofield Statement

The most basic and important part of the SQL extension is the `create geofield` statement:

```
create [virtual | materialized] [qualitative | quantitative]
                geofield <geofield_name> as
select <column_name | expression> as measurement_location,
       [<column_name | expression> as measurement_time,]
       <column_name | expression> as measurement_value
from <measurement_table_name>
where [<conditions_on_measurements> and]
     CRS = <CRS_code> and
     interpolation = <interpolation_parameters> and
     [square_classifier = <square_classifier_parameters> and]
     [intervals = <intervals_boundary_definion>] and
     [context_region = <geoobject | subquery | array_range> ]
     [resolution = <integer | double <unit>>];
```

The names `measurement_location`, `measurement_time` and `measurement_value` are treated as keywords, allowing to designate a column or an expression as, respectively, the locations of the measurement points, the time instants or the time periods of the measurements, and the measured values.

The `create geofield` statement can be used on its own or can be embedded into other SQL statements, such as: `select`, `insert` or `update`.

Geofields are by default virtual, which means that the `create geofield` statement only adds the definition of a geofield to the database dictionary. The `materialized` keyword has been added to improve the integration with multidimensional arrays (an example is given later), and as an optimization mechanism.

There is no clause for geofield domain definition in the `create geofield` statement, as the estimation of geofield values depends on the interpolation method and node search rule used. Instead, the user can define a geofield context region specifying an area on which geofield estimation can be undertaken. An attempt to estimate a geofield value outside its context region will return a

special marker `UGV` (Unknown Geofield Value). The `resolution` column allows the user to determine the size of the elementary quadrants. It defaults to the array cell size used in a query.

All geofield properties definitions stored in the database dictionary can be overwritten when the geofield is used, e.g. in the `select` statement the default interpolation method or its default parameters can be changed, or in the `insert` statement the default context region can be redefined.

As in geostatistics usually Kriging is the best choice [7], to simplify geofield definition, a special registry of variograms models is added. Also, a registry for geofield variability models used by quandrant classifiers [4] is created.

The following statement creates an exemplary geofield of airborne suspended matter, called `suspended_matter`:

```
create virtual geofield suspended_matter as
select measurement_location, measurement_time,
       measurement_value
from air_pollution
where pollutant_code = 'SUSPENDED_MATTER' and
     measurement_quality = 'GOOD' and
     CRS = 'EPSG:4326' and
     interpolation = '{method: OK, semivariogramID: 71,
          nodeSearchType: QUADS, nodeSearchN: 6,
          nodeSearchR: 11 km}' and
     resolution = 5 m;
```

The geofield `suspended_matter` mathematical model uses measurements of airborne suspended matter from the table `air_pollution` (condition `pollutant_code = 'SUSPENDED_MATTER'`) of good quality (condition `measurement_quality = 'GOOD'`), and is based on Ordinary Kriging (OK) with semivariogram with ID = 71. A quadrant search of measurement nodes, with six nodes for quadrant and the maximal distance of the search equal to 11 km, is used. Elementary quadrants (array cells) have side length of five meters. No context region is defined, so it must be explicitly provided in a query referencing the geofield, or it must be inferred from the query.

The definition of the geofield `suspended_matter` can be used to compute a quantitative geofield of the average-yearly distribution in 2013 of airborne suspended matter in the context region specified by coordinates given by a polygon:

```
select location, value
from suspended_matter
where year(measurement_time) = 2013 and
      context_region = polygon(49.95,18.38, 50.55,18.38,
                               50.55,19.58, 49.95,19.58, 49.95,18.38);
```

A qualitative geofield, defined by four intervals: $(-\infty, 110]$, $(110, 165]$, $(165, 220]$, and $(220, \infty)$ $\mu g/m^3$, can be generated, using the BPO square classifier [4] (with: geofield variability model stored in the registry with ID = 10, and distortion coefficient = 0.8), by following query:

```
select interval, shape
from suspended_matter
where year(measurement_time) = 2013 and
     square_classifier = '{type: BPO, geofieldVariabilityModelID:
                           10, beta: 0.8}' and
     intervals = (110, 165, 220) and
     context_region = polygon(49.95,18.38, 50.55,18.38,
                         50.55,19.58, 49.95,19.58, 49.95,18.38);
```

3.2 Geofield as a Table Column

The following example illustrates how geofield can be used as a virtual column in a table:

```
create table city (
   city_id integer primary key,
   city_name varchar not null,
   city_boundary geometry not null,
   sm_pollution geofield
);
```

Using the exemplary geofield `suspended_matter`, information about Gliwice can be inserted by statement:

```
insert into city(city_id, city_name, city_boundary, sm_pollution)
values (1, 'Gliwice', <WKT_GEOM>,
        suspended_matter where context_region = city_boundary);
```

In the `insert` statement the value of the column `context_region` is set to the boundary of Gliwice (`<WKT_GEOM>` is a placeholder for the polygon defining boundary of Gliwice). Geofields can be defined inline, but defining them as separate statements (as in the running example) simplifies other statements, removes redundancy, and makes the code more readable.

The following example shows how a geofield may be added as a virtual attribute of an array-typed column. The following example extends the example from the listing 2 from [12], describing ISO SQL extension proposal.

```
create table landsat_scenes_air_pollution (
 id integer not null,
 acquired date not null,
 scene row (band1 integer,
            ...,
            band5 integer,
            sm_pollution geofield default suspended_matter
                                    where time = acquired
           )
       array [ x ( 1 : 5000 ) , y ( 1 : * ) ]
);
```

In the `create table` statement default geofield is provided. The condition `time = acquired` imposes temporal restriction on the geofield. The geofield context region is taken from the raster boundaries. Adding the `materialized` keyword

to the geofield definition would result in computation of its raster representation during a row insertion. If the user changes the interpolation method in a query, a new geofield raster representation will be computed using the modified geofield model.

4 Optimizability of Geofield Queries

This section shortly discuses opportunities given by using virtual (dynamically computed) geofields in declarative queries. The simplest and the most basic operation – in the context of this paper – is the conversion of a geofield from its table representation to an array representation. As the cost of the conversion is determined by interpolation, minimization of the number of places in which geofield values must be computed leads to significant query processing speedup [2, 4]. This can be achieved by integration of the conversion with an array constructor as shown in the previous section. If there are any conditions on geofield values in the query, passing them to the conversion called in an array constructor may significantly speedup query processing, as shown in [4] for qualitative geofields creation. This is especially profitable for qualitative geofields, but may be also used for quantitative geofields.

If a geofield query references more than one qualitative geofield, using the concepts presented in section 2 may significantly speedup query processing. The PNR representation may be used internally by a DBMS, and the intermediate result can be converted from the PNR representation to the array representation for further processing. The ideas presented in [5] and [4] may be easily generalized into more than 2D space, as far as an interpolation method is available for required dimensionality [8].

Let us assume, that a query states that values of a raster R must belong to an interval *i1*, and values of a geofield G must belong to an interval *i2*. We can distinguish three strategies for the query evaluation: 1) compute the geofield G for the intersection of the domains of the raster R and the geofield G, read the raster R, perform the selections on its values and find the cells which fulfill both conditions; 2) read the raster R, find the cells which values belong to the interval *i1*, compute the geofield G only for these cells and check if the computed values belong to the interval *i2*; 3) compute the geofield G and find its cells belonging to the interval *i2*, read only the chunks of the raster R which contain these cells, and check if they belong to the interval *i1*. As the interpolation is time expensive, usually the second strategy should be the best. However, if the selection on the geofield G values has high selectivity and the raster R is very large, then the third strategy may be the best. If both selectivities are low, then the first strategy may be the best, as anyway the whole geofield G must be computed and the whole raster R must be read. As experiments and theoretical analysis showed [2], usually qualitative geofield computation using quadrant classifiers [4] is much faster when the geofield context region consists of a small number of large quadrants, than when it consists of larger number of smaller quandrants. This can be explained by a rule of thumb telling that

the quadrant classification cost is proportional to the number of its boundary elementary quadrants, while the raster computation cost is proportional to the number of its cells (elementary quadrants). As shortly discussed, even in such a simple case, the choice of the best strategy is not clear and a cost optimizer is needed [2,4]. The problem complicates significantly, when more geofields should be computed and combined by some operations, e.g. finding areas where some pollutants exceed given thresholds — this problem reminds the classical join order problem from relational databases [2].

5 Related Work

In [13] an idea and implementation of enriching the DBMS with interpolation were presented. The implementation was based on database procedures and temporary tables, not on an algebraic operation for geofield estimation, as is in this paper. The work was presented long before any significant development of array DBMSs took place, and included only the table representation of geofields, while this paper presents approach encompassing both representations. However, some ideas from [13] are still valid for array databases: geofield values interpolation during a query evaluation, choosing an interpolation method in a query, and specifying in a query which measurement values should be used in interpolation.

In [12] two approaches to the integration of multidimensional arrays with relations are distinguished: a) *array-as-attribute* – arrays are treated as the other data types, which allows to define columns of the array type, but prevents from usage of arrays without tables; and b) *array-as-table* – arrays are on the same level as tables, which allows to use arrays without tables, but does not allow to define columns of the array type.

In the work on the ISO SQL extension the array-as-attribute approach is followed [12], which is consistent with the support of one dimensional arrays in the current ISO SQL standard. Conversion between an array and a table is performed using the `unnest` and `nest` operators. However, during this conversion no geofield values can be estimated, what is possible in the presented SQL extension.

The rasdaman was the first array DBMS [12]. From the beginning it has been developed as a layer above conventional RDBMSs. The rasdaman follows the array-as-attribute approach. Each cell of an array may store one or more values of types specified in the array declaration. The coordinates of array cells are defined as bounded or unbounded integers. Only the simple nearest neighbor interpolation is supported for scaling [15].

SciDB is a DBMS designed and implemented to address scientific needs [17]. It implements only the array-as-table approach and does not support tables. SciDB provides linear algebra, but no spatial interpolation method (such as Kriging) is available. The package SciDBR provides integration of R with SciDB, which makes it possible to utilize sophisticated spatial interpolation methods, however, declarative geofield queries are not supported and their optimization is not possible.

SciQL is an array DBMS based on column-oriented DBMS MonetDB [10] and follows the array-as-table approach. In [10] only interpolation for time series is mentioned.

Some commercial and open source RDBMSs have been extended with raster data support, following the array-as-attribute approach. For example, Oracle RDBMS provides a set of functions and procedures for raster processing. As a consequence, usually procedural code in PL/SQL is needed, which limits the possibility of queries optimization. The current Oracle version 12c Release 1 supports only 2D rasters and provides only basic interpolation methods for rasters (near neighbor, bilinear, biquadratic, cubic and average) [14]. Also PostGIS provides support for 2D rasters with similar interpolation/resampling methods: nearest neighbor, bilinear, cubic, cubic spline, Lanczos, and inverse distance weighted [1].

To the best of the authors knowledge, none of the related DBMSs nor the ISO SQL extension under development contains SQL capabilities similar to the extension described in section 3 of the paper. As a consequence, optimizability discussed in section 4 is not possible in these DBMSs.

6 Conclusions

The presented extension of SQL for geofields processing widens data integration capabilities, gives new optimization possibilities, and eliminates the problem of unloading data from database for interpolation in external tools and loading the results back to the database for further processing. As discussed in the paper, adding the geofield support to the DBMS is more than just adding an interpolation function, as this extension significantly influences query optimization. Interpolation during query processing may be time consuming, but on the other hand, it eliminates the problems connected with measurement points selection based on the predicates included in the query, measurements updates, CRS conversions, as well as raster scaling, shifting and rotating. This can be especially valuable in a multi-resolution database [11], when an array representation of a geofield may be generated on-the-fly in the required resolution and CRS, instead of rescaling its previously generated and stored representation. The presented approach also allows to easily change the geofield model used (to change the interpolation method and/or adjust its parameters). The impact of such changes on the query answer can be easily evaluated as the difference may be computed by another SQL query.

Distinguishing between qualitative and quantitative geofields is not very important to the users writing queries, but has very significant impact on query optimizability.

References

1. PostGIS 2.1.4dev Manual. SVN Revision (12916)
2. Bajerski, P.: Using Peano Algebra for Optimization of Domain Data Sets Access during Analysis of Features Distribution in Space, PhD Dissertation. Poland, Gliwice (2006) (in Polish)
3. Bajerski, P.: Optimization of geofield queries. In: 1st International Conference on Information Technology, IT 2008, pp. 1–4 (2008)
4. Bajerski, P.: How to efficiently generate PNR representation of a qualitative geofield. In: Cyran, K.A., Kozielski, S., Peters, J.F., Stańczyk, U., Wakulicz-Deja, A. (eds.) Man-Machine Interactions. AISC, vol. 59, pp. 595–603. Springer, Heidelberg (2009)
5. Bajerski, P., Kozielski, S.: Computational model for efficient processing of geofield queries. In: Cyran, K.A., Kozielski, S., Peters, J.F., Stańczyk, U., Wakulicz-Deja, A. (eds.) Man-Machine Interactions. AISC, vol. 59, pp. 573–583. Springer, Heidelberg (2009)
6. Baumann, P.: OGC® GML Application Schema – Coverages, Open Geospatial Consortium, OGC 09-146r2, http://www.opengis.net/doc/GML/GMLCOV/1.0.1
7. Cressie, N.: Statistics for Spatial Data. Wiley (1995)
8. Cressie, N., Wikle, C.K.: Statistics for Spatio-Temporal Data. Wiley (2011)
9. Goodchild, M.F., Yuan, M., Cova, T.J.: Towards a general theory of geographic representation in GIS. International Journal of Geographical Information Science 21(3), 239–260 (2007)
10. Kersten, M., Zhang, Y., Ivanova, M., Nes, N.: SciQL, a query language for science applications. In: Proceedings of the 2011 EDBT/ICDT Workshop on Array Databases, Uppsala, Sweden, March 25, pp. 1–12 (2011)
11. Kozioł, K., Lupa, M., Krawczyk, A.: The extended structure of multi-resolution database. In: Kozielski, S., Mrozek, D., Kasprowski, P., Małysiak-Mrozek, B., Kostrzewa, D. (eds.) BDAS 2014. CCIS, vol. 424, pp. 435–443. Springer, Heidelberg (2014)
12. Misev, D., Baumann, P.: Extending the SQL array concept to support scientific analytics. In: Conference on Scientific and Statistical Database Management, SSDBM 2014, Aalborg, Denmark, June 30-July 02, p. 10 (2014)
13. Neugebauer, L.: Optimization and evaluation of database queries including embedded interpolation procedures. In: Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data, Denver, Colorado, May 29-31, pp. 118–127 (1991)
14. Oracle and/or its affiliates: Oracle® Spatial and Graph. GeoRaster Developer's Guide, 12 c Release 1 (12.1) E49118-04 (November 2014)
15. rasdaman GmbH: rasdaman Query Language Guide, 9.0 edition (2014)
16. Stonebraker, M., Becla, J., DeWitt, D., Lim, K., Maier, D., Ratzesberger, O., Zdonik, S.: Requirements for science data bases and SciDB. In: Fourth Biennial Conference on Innovative Data Systems Research, CIDR 2009, January 4-7, Asilomar, CA, USA (2009) (Online Proceedings)
17. Stonebraker, M., Brown, P., Zhang, D., Becla, J.: Scidb: A database management system for applications with complex analytics. Computing in Science and Engineering 15(3), 54–62 (2013)