

The Filter/Refine Strategy: A Study On The Land-Use Resource Dataset In Norway

Gjermund Hanssen

Department of Mathematical Sciences and Technology, Norwegian University of Life Sciences,
PO Box 5003, N-1432 Ås, Norway
gjermund.hanssen@vegvesen.no

Abstract. The use of a spatial index followed by an exact geometry check (a filter/refine strategy) is commonly applied in spatial database management systems. Its efficiency in selecting a small subset of geographic features from a large feature collection is undebatable. But there is a problem with the strategy: For a pure overlap test in the filter phase, the filter/refine strategy gets less optimal as the query window grows larger. Loosely spoken, the reason for this is that the candidates falling completely within the area of the query window are not included in the result although a decision can be made from the filter phase. We argue that an overlap test followed by an inside test will reduce the query costs from quadratic to linear as the query window doubles in north- and east dimension. A study on the Land-use resource data set in Norway (DMK) gives an indication of this.

1 Introduction

Spatial selection is an operation that returns from a set of objects those that fulfill a spatial predicate. In this paper, emphasis is put on the *intersection* predicate¹, given a query window and the land-use resource dataset in Norway (DMK). Two examples:

- Find all cultivated land intersecting a query window.
- Find all lakes intersecting a query window.

To efficiently process the window query, most vendor and research engines (including Oracle Spatial and PostgreSQL) use a two-stage filter/refine strategy. Loosely spoken, the filter removes (a possibly large) non-candidate set, and the remaining set is examined in the second stage. The motivation is that the first step is much “cheaper”, in computational terms, than the second step. And if the filter is efficient, then the expensive second step will have little to do.

If the query window is large, compared to the geometries it is selecting, then the filter gets less efficient. The reason for this is that a pure overlap test in the filter step will leave the expensive second step much to process. Fig. 1 gives an indication of the problem (Measurements taken from the DMK dataset. The number of MBRs intersecting the query window need to be processed).

¹ According to a set-based model of space, it is used to denote any configuration in which two objects share one or more common points. It includes more fine-grained topological relationships, for example, meet, overlap, covered.by, covers, inside, and contains.

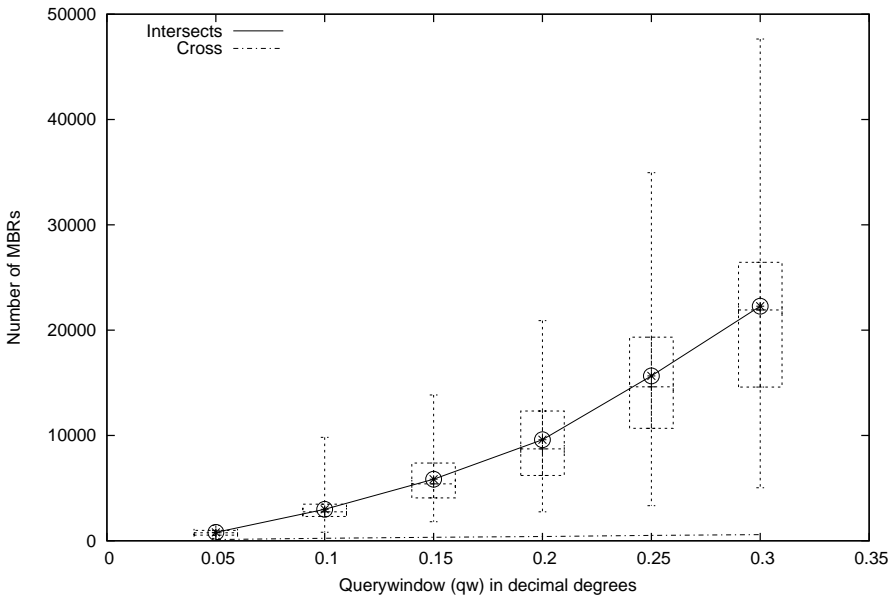


Fig. 1. Quadratic and linear growth in MBRs

One solution is to bypass the expensive second step by checking for containment. If a MBR is fully within the query window, then it can be safely included in the result (without involving the expensive second step). MBRs that are crossing the borders of the query window (a small number as indicated in Fig. 1) would need to go through the refinement phase.

1.1 Related work

Ravi and Ravada [8] describe how to reduce geometry–geometry comparisons by first filtering using the interior approximations of geometries (in addition to and after comparing the exteriors, i.e., the MBRs). They implement this technique as part of the R-tree indexes in *Oracle Spatial* and argue that the query performance improves by more than 50% for most queries on *real* spatial datasets.²

² US census blocks.

2 The theory: The filter/refine strategy

With a spatial index present the window query can be processed in two phases [2, 4, 5]:

1. The first phase is called the filter step, where all Minimum Bounding Rectangles (MBRs)³ overlapping the query window is found. The search is “pruned down” with computationally cheap intersection tests between the query window and MBRs—at most four computations are needed to determine whether the two rectangles intersect. The result, a superset of the final result, is passed to the next step.
2. The second phase is called the refinement step, which runs on just the subset returned by the first phase. For the MBRs in the candidate set, a check—a computationally expensive one—is needed to confirm if the spatial objects they represent really fulfill the query condition, i.e., do they really intersect with the query window.

Fig. 2 illustrates the idea.

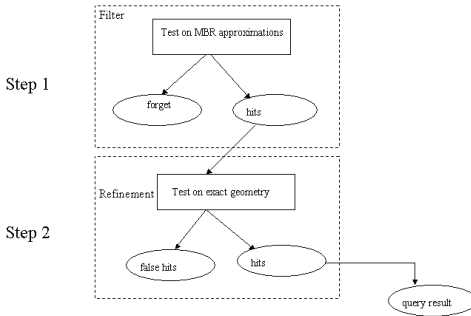


Fig. 2. Filter-refine strategy

An important property of the filter/refine strategy is that it is efficient for small query windows. The reason for this is that the number of candidates passed to the refinement phase is small; thus, step 2 will have little to do.

³ Approximations of geometric features are used to minimize CPU-expensive geometric tests. The Minimum Bounding Rectangle (MBR) is the traditional approximation [3], it is represented with only two points: the lower left and upper right points enclosing the geometry it represents—like an envelope.

3 Method and the experimental setup

Elements in the experimental setup include:

- PostGIS⁴, which is an extension to the PostgreSQL object-relational database management system⁵. PostGIS supports all the objects and functions specified in the OGC “Simple Features for SQL”⁶. PostGIS includes support for GIST-based R-Tree spatial indexes.
- Land-use resource data for part of Norway (Max extent: 9.58 58.87 southwest and 12.87 62.69 northeast in decimal degrees), stored in a table with 1,392,436 tuples in PostGIS/PostgreSQL. For the spatial component the total number of vertices is 143,386,857 (143 millions) of an average of 103 in each polygon. The maximum number of vertices in one polygon is 31,632; the minimum number of vertices in one polygon is 4. More fine-grained characteristics for the most meaningful categories⁷ are given in Table 1.
- A client program with functionality to: (1) select the size of a query window according to the users need; (2) randomly position the query window within the max extent of the data; and (3) issue SQL-queries to PostGIS and collect the results.

Two experiments are carried out. For both, query windows randomly positions generated are constructed with different sizes: 0.05, 0.1, 0.15, 0.2, 0.25 and 0.3 decimal degrees (in north- and east-dimension). A series of queries are executed within each group. To get statistically meaningful results, samples are *only* included if the aggregated area of the relevant⁸ MBRs are larger than the area of the query window. The sample size within each group is 100.

3.1 Experiment 1

The purpose of the first experiment is to collect facts about: (1) how many MBRs intersect the query window; and (2) how many MBRs are within the query window. Based on (1) and (2) the number that crosses the borders of the query window is calculated (the difference).

Two SQL-queries are employed:

Query 1

```
select count(the_geom)
from dmktest
where the_geom && GeometryFromText(' '+QueryWindow+',4326)
```

⁴ <http://postgis.refrations.net/>

⁵ <http://www.postgresql.org/>

⁶ <http://www.opengis.org/techno/specs/99-049.pdf>

⁷ Urban Area, Road and Railway, Not Mapped Area and Unclassified Area are left out.

⁸ Lake FTEMA 3000, Urban areas FTEMA 5200, Roads and railways FTEMA 7200, and Not Mapped Area FTEMA 9300 do not qualify, although they are part of the DMK dataset.

Table 1. Characteristics of the DMK data set

Name	Atil	Total	Vertices		
			Avg	Max	Min
Ur, steinrys	31	259090	58	1139	4
Komb. myr/fastmark	15	2656	111	492	16
Myr med lauvskog	14	283304	84	1637	4
Myr med blandingskog	13	477928	90	1544	4
Fjell i dagen	29	1270270	85	6946	4
Myr med granskog	12	5846018	88	3230	4
Grunnlendt mark	28	896417	51	4186	4
Anna jorddekt fastmark	27	7513764	60	15503	4
Myr	11	10178729	114	15730	4
Lauvskog	26	6245030	80	9651	4
Blandingskog	25	4060846	94	4682	4
Barskog	24	70115483	127	14665	4
Gjdsla beite	23	1111286	59	1185	4
Overflatedyrka jord	22	136199	53	438	4
Fulldyrka	21	9620123	84	4555	4
Grustak	32	46568	57	837	4
Grunnlendt myr	16	244	81	168	26

(The QueryWindow variable is computer generated, && is an overlap operator, which tests whether one features's bounding box overlaps that of another)

Query 2

```
select count(the_geom)
from dmktest
where the_geom @ GeometryFromText('"+QueryWindow+"',4326)
```

(The @ operator returns true if the_geom's bounding box is completely contained in the Query-Window's bounding box)

Query 1 counts the geometries whose MBRs intersect the query window; query 2 counts the geometries whose MBRs fall completely within the query window. The difference (query 1 - query 2) gives us the number of MBRs that would need to go through the refinement phase.

3.2 Experiment 2

The purpose of the second experiment is to give an indication of how query execution time is influenced when exact geometry processing is applied for: (1) only those geometries whose MBRs cross the borders of the query window; and (2) for all geometries with MBRs intersecting the query window (cross and within).

Three SQL-queries are employed:

Query 3

```
select count(intersection(the_geom,
    envelope(GeometryFromText('"+QueryWindow+"'::Box3d,4326)))
from dmkttest
where the_geom && GeometryFromText('"+QueryWindow+"'::Box3d,4326)
```

Query 4

```
select count(intersection(the_geom,
    envelope(GeometryFromText('"+QueryWindow+"'::Box3d,4326)))
from dmkttest
where the_geom && GeometryFromText('"+QueryWindow+"'::Box3d,4326)
and not the_geom @ GeometryFromText('"+QueryWindow+"'::Box3d,4326)
```

Query 5

```
select count(the_geom)
from dmkttest
where the_geom @ GeometryFromText ('"+QueryWindow+"'::Box3d,4326)
```

The result (the number of geometries) from Query 3 is the same as the combined result from query 4 and query 5. But, as Fig. 3 soon will show, their execution time differs.

4 Results

The results from experiment 1 is captured in Fig. 1. We can read two things from this plot: (1) the number of MBRs intersecting the query window follows the area of the query window, i.e., when the length of the sides of the query window doubles in east- and north-dimension the number of MBRs become approximately four times larger; and (2) the number of MBRs that are crossing the borders (the difference between intersect and within) of the query window grows linearly with the length of the sides of the window.

The boxplot gives an indication of the variation of the observed values. For observations with relatively few MBRs intersecting the query window, the MBRs are relatively large; for observations with relatively many MBRs intersecting the query window, the MBRs are relatively small. Typically for the observations in the first group is samples taken from forestal and boggy rich land; for the last group, samples are taken from regions with greater diversity—cultivated land, semi-open land, open land etc.

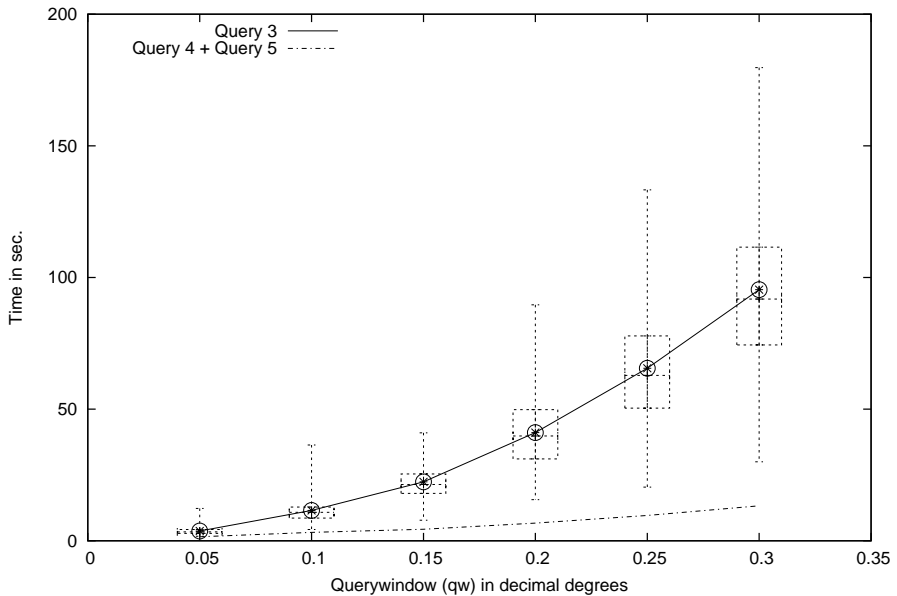


Fig. 3. Quadratic and linear growth in time

Fig. 3 shows the effect the size of the query window has on the response time measured in seconds; and, even more important, the effect reduced geometry processing (the inside filter) has on the response time.

The curve that grows slowest is the result of query 4 and query 5 executed in serial. For query 4 two filters are used: (1) a filter to prune away the MBRs that falls completely outside the query window; and (2) a filter to prune away the MBRs that are completely within the query window—it only concentrates on the MBRs crossing the borders of the query window—query 5, on the other hand, concentrates on *only* the MBRs falling completely within the query window. Taken together, all geometries that intersects the query window are counted.

The curve that grows fastest (query 3) uses *only* one filter, a filter to prune away the MBRs that fall completely outside the query window—a pure overlap test in the filter phase—the rest are passed to the refinement phase (see Fig. 2).

5 Discussion

Experiment 1 shows that the number of MBRs falling completely within a query window grows quadratic when the query window doubles in east- and north-dimension, and that the number of MBRs that are crossing the borders grows linearly. It means that for a pure overlap test in the filter phase, the filter/refine strategy gets less optimal as the query window grows larger. The reason for this is that the refinement phase will get

more to do than necessary. An optimal growth is linearly, while the filter/refine strategy will give a quadratic growth in the number of candidates that need exact geometry processing. By employing an inside filter the number of candidates passed to step 2 can be reduced from quadratic to linear as the query window doubles in north- and east dimension.

In experiment 2 this observation is directly reflected in reduced query costs. It indicates that an overlap test followed by an inside test will reduce the query costs from quadratic to linear as the query window doubles in north- and east dimension. It also indicates that the additionally inside test is so cheap that it can be ignored.

Even better results could be obtained if the two SQL queries were not executed in serial as separate sessions over the net (a suboptimal solution—and far from elegant). What we would like to see is a query optimizer that can handle this in one request.

5.1 Proposed solution: The filter/refine strategy with an inside test

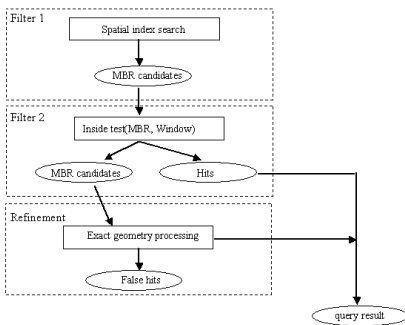


Fig. 4. Proposed solution

The idea is to *only* pass MBRs who cross the borders of the query window to the refinement phase; for the rest, a decision can be made from the filter phase.

Step 1 process the window query with a spatial index. The outcome is a MBR candidate set—candidates that intersects the query window—that possibly belongs to the query result.

Step 2 test whether the MBRs in the candidate set are *inside* the query window. If true, then it is also true that the spatial objects approximated by the MBRs belong to the query result.

In step 3 all remaining members of the candidate set are examined. This step requires the exact representation of the spatial objects.

The number of hits that can be derived from step 2 follows the growth of the query window—it grows quadratic; the number of candidates passed to step 3 grows linearly, i.e., only those MBRs that cross the borders of the query window need exact geometry processing.

An important property of this model is that it will perform significantly better than the traditional filter-refine model.

5.2 Lesson learnt

You have seen that for a pure overlap test in the filter phase, the filter/refine strategy gets less optimal as the query window grows larger. The reason for this is that the number of candidates passed to step 2 grows quadratic as the query window doubles in north- and east dimensions. You have seen indications of:

An overlap test followed by an inside test will reduce the query costs from quadratic to linear as the query window doubles in north- and east dimension.

The implication of this is that adding an inside test in the original filter/refine strategy will potentially give more optimally query processing for data sets with similar characteristics as DMK.

References

1. Guttman, A. *R-Trees: A dynamic index structure for spatial searching*, In Proceedings of ACM SIGMOD Conf. pages 47–57, 1984.
2. Orenstein, J. A. *Spatial Query Processing in an Object-Oriented Database System*, In Proceedings of ACM SIGMOD Conf. pages 326–333, 1986.
3. Papadias, D. Theodoridis, Y. Sellis, T. and Egenhofer, M. *Topological Relations in the World of Minimum Bounding Rectangles: A Study with R-trees*, In Proceedings of ACM SIGMOD Conf. pages 92–103, 1995.
4. Brinkhoff, T. Horn, H. Kriegel, H. P. and Schneider, R. *A Storage and Access Architecture for Efficient Query Processing in Spatial Database Systems*, In Proceedings of 3rd SSD Symposium, pages 357–376, 1993.
5. Brinkhoff, T. Seeger, B. Kriegel, H. P. and Schneider, R. *Multistep processing of spatial joins*, In Proceedings of ACM SIGMOD Conf. pages 197–208, 1994.
6. Sellis, T. K. Roussopoulos, N. and Faloutsos, C. *The R+-Tree: A Dynamic Index for Multi-Dimensional Objects*, In Proceedings of Very Large Databases (VLDB), pages 507–518, 1987.
7. Beckmann, N. Kriegel, H. P. Schneider, R. Seeger, B. *The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles*, In Proceedings of ACM SIGMOD Conference, pages 322–331, 1990.
8. Kothuri, R. K. and Ravada, S. *Efficient Processing of Large Spatial Queries Using Interior Approximations*, SSTD 2001, LNCS 2121, pages 404–421, 2001.