International Journal of Database Theory and Application
Vol. 5, No. 4, December, 2012

An Optimization Technique for Spatial Compound Joins Based on a Topological Relationship Query and Buffering Analysis in DSDBs with Partitioning Fragmentation

Xinyan Zhu[1], Chunhui Zhou[2], Wei Guo[1], Di Chen[1] and Kezhong Liu[2]

[1]State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan China

[2]Navigation School, Wuhan University of Technology, Wuhan China

**Correspondent Author: Chunhui Zhou* (church_zhou@whut.edu.cn)*

Abstract

Spatial Partitioning Fragmentation (SPF) is a popular method to partition data in Distributed Spatial Databases (DSDBs). The issue of cross-border queries is an inherent problem however with distributed spatial data queries based on partitioning fragmentation given a continuity and strong correlation of geospatial data. In the case of partitioning fragmentation, a global spatial join can be translated into multiple sub-joins, and then divided into 2 groups: Cross-Border Joins (CBJs) and Non-Cross-Border Joins (NCBJs). The CBJ approach is essential for process efficiency in a distributed spatial query. A compound join based on a topological relationship inquiry and a buffering analysis is a crucial class of spatial queries. This article studies compound join optimization for spatial queries in a DSDB, and proposes a set of theorems and rules for the optimization of CBJs, contributing a removal rule and a filtering rule. This article supplies a Partition Fragmentation Join Strategy (PFJS) to resolve the compound join problem based on these rules. Experimental results show that the PFJS can improve the efficiency of CBJs, when compared with the Naive Join Strategy (NJS) or the Spatial Semi-Join Strategy (SSJS). The PFJS contributes to the optimization of spatial compound joins.

Keywords: *Spatial compound join; topological relation; buffer analysis; partitioning fragmentation; distributed spatial database*

1. Introduction

The production, management, maintenance and application of spatial data are distributed in a Geographic Information System (GIS). These distribution features predispose a move toward distributed database systems. One of the challenges for a distributed spatial database is the performance of distributed queries [1]. The spatial join is an important operation that affects the efficiency of spatial queries [2]. Spatial join optimization therefore is a crucial problem to be addressed when optimizing spatial queries.

Differing from the horizontal and vertical fragmentation evident in a traditional distributed database, distributed spatial database fragmentation can be classified as partitioning or layer fragmentation [3]. Partitioning fragmentation (spatial partition or horizontal fragmentation) involves decomposition of spatial data for the same geographical coverage to database tables stored at different sites. Layer fragmentation (thematic fragmentation or vertical fragmentation) involves storage of spatial data for the same geographical coverage at different sites within different thematic layers. In fact, data can be distributed through a combination of these two methods. In partitioning fragmentation, distributed spatial data management has some special characteristics, which include cross-border spatial correlation issues and cross-border seamless query problems.

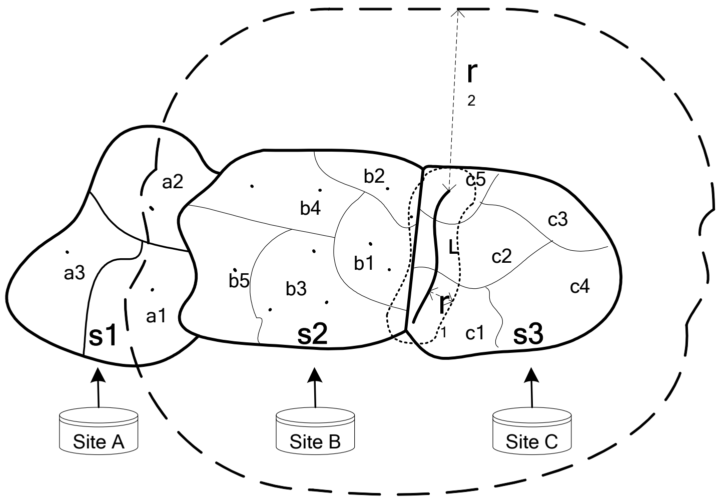


Figure 1. Cross-border Query Issues in a DSDB with SPF

In Figure 1, spatial data are partitioned by 3 regions, and 3 fragments s1, s2, and s3 are produced. These fragments are distributed at site A, B, and C respectively. From Figure 1, we observe that to query spatial objects adjacent to *b5* stored in *Site B*, the objects *a1* and *a2* in *Site A* are necessarily involved. Moreover, a line *L* is in fragment s3 at site C, to query spatial objects within a buffer radius r_1 from *L*, the objects in *Site B* may become involved. In addition, the length of the buffer radius can be variable, thus, buffer zone inquiries may involve those fragments not adjacent to each other. For example, as shown in Figure 1, if a buffer radius increases to r_2, A buffer zone query for line *L* in fragment *s3* may cross fragment *s2* to reach *s1*. All these queries are called cross-border spatial queries. This cross-border problem in query processing generated by spatial correlation does not exist in centralized databases.

This article focuses on optimization of compound queries, a specific type of cross-border query. Assuming that there are two spatial relations, City and River:

City (Name, ProvName, Pop, Shape)

River (Name, Length, Shape)

City has these attributes; city name (*Name*), province name (*ProvName*), urban population (*Pop*) and geometry (*Shape*), while River has these attributes; river name (*Name*), river length (*Length*) and geometry (*Shape*). To query all cities within a distance of 20 km from the Changjiang River with a population of more than 100,000, the following SQL statement Q1 can be used:

```
Q1: Select C.Name
    From City C, River R
     Where Intersects ( C.Shape , Buffer( R.Shape, 20)) = TRUE  AND
             R. Name = "Changjiang" AND  C.Pop > 100000
```

In this query, a *buffer* operation is used to obtain the buffer zone, and the *Intersects* predicate is used to determine whether a city *intersects* with the buffer zone. Therefore, the query Q1 is called a compound query that combines a buffer analysis and a topological relationship query. The query Q1 is also called a compound join because it involves two arguments.

This article addresses join optimization of compound queries based on SPF. Research related to spatial queries including the spatial topological relationship join and the distributed spatial join is presented in the next section. Section 3 proposes join optimization principles for compound spatial queries. Section 3.1 presents a description of the decomposition of a spatial query when spatial data are partitioned by region. The definitions of the Cross Border Join (CBJs) and Non-Cross-Border Join (NCBJs) are also presented in this section. Section 3.2 introduces the predicates of the spatial topological relationship. A classification of spatial joins and a definition of the compound join are provided in this section. Sections 3.3-3.5 present a set of detailed concepts, theorems, and rules for compound query fragment join optimization. In Section 4, the Partitioning Fragmentation Join Strategy (PFJS) for compound queries is discussed in detail, the formalized transformation rule in the PFJS is presented, and the complexity of the PFJS is analyzed. Section 5 elaborates on the design of experiments to test the proposed theory and strategy. Two strategies from the literature are compared with the PFJS in this section. The experimental results show that the PFJS performs better than the other strategies. Section 6 presents the conclusions drawn from this research.

2. Related Work

In a centralized spatial database, the optimization of a spatial query employs two-step methods involving filtering and refinement [4, 5].

In the first step, filtering, complex spatial objects are given approximations.These approximate objects are used for the query. The target objects in the filtering step are not accurate results of the original query, but candidates that need to be refined in the second step. At present, Minimum Bounding Rectangles (MBRs) are the most commonly used approximation of spatial objects. MBRs in general, are, stored in spatial indices for example, an R/R* tree [6, 7] or a quad tree [8]. The second step is refinement. Refinement deploys a geometric algorithm to obtain accurate query results. To reduce the grain size of spatial query optimization, Park proposed a two-stage optimization strategy—Earlier Separates Filter and Refinement (ESFAR) [9]. ESFAR spatial queries are dependent on spatial indices, spatial query optimizations, spatial join algorithms and other factors. In the ESFAR method, spatial indices are used to reduce the search space, and spatial query optimization algorithms are also adopted to reduce execution time by adjusting the implementation steps. In general, spatial join algorithms are important to the efficiency of spatial query operations.

Seamless cross-border spatial queries present inherent problems in a distributed spatial database based on partitioning fragmentation., At present, there is an extensive body of research available concerning spatial join optimization. However, few studies exist on spatial join optimization in a DSDB based on SPF. The existing spatial join optimization studies have certain shortcomings: (1) Studies on spatial join predicates are not sufficiently comprehensive. Most of these studies use *Intersects* whose join optimizations are not applicable to all spatial topological relationship predicates. For example, a topological *Disjoint* query cannot use the *Intersects* optimization method — especially in a distributed partitioning fragmentation context. Disregarding such distinctions leads to incorrect results. (2) There is no existing research on compound query optimization involving spatial analysis predicates (*Buffer*, *Distance*, *Union*, *Intersection*, *Difference*, *SymDifference*, *ConvexHull*, etc.) in spatial fragment joins. In a distributed database, spatial fragment joins combined with spatial analysis predicates may cross certain non-adjacent spatial fragments (as shown in Figure 2). Such fragment joins have a significant influence on the performance of queries. (3) In the literature concerning centralized spatial database join optimization, several important methods have been proposed. The Seed Tree Join [10], Spatial Hash Join [11], and the Slot Index Spatial Join [12], can optimize the topological spatial joins on two datasets with a single index or no index. These are three-step methods: (1) Partitioning of the join datasets with rectangular cells. (2) Joining each pair of datasets of the corresponding spatial rectangular regions. (3) Merging these intermediate results to obtain a final result. In a DSDB based on partitioning fragmentation, it is impossible to provide corresponding regions in cross-border joins, and spatial fragments are not necessarily rectangular (as shown in Figures 1 and 2). Therefore, these methods cannot be applied to cross-border joins in a DSDB based on partitioning fragmentation.

Jacox and Samet reviewed studies on spatial join technology [13]. Based on their review, it appears that research on distributed spatial joins is quite limited. Three dominant strategies exist for spatial joins with the Intersects predicate in distributed spatial query processing. The first one is the Naive Join Strategy [14]. This strategy transmits all relationship records from one site to another. The primary problem with this strategy is the cost of calculation and transmission. The second strategy is the Semi-Join Strategy [14, 15], which uses the traditional distributed join method to reduce the transmission cost. In [14], Abel and his colleagues advised the use of a Spatial Semi-Join, which is a spatial version of the relational semi-join. Superfluous spatial objects are eliminated through semi-joins on spatial approximations. The semi-join strategy is, in most instances, more efficient than the naive strategy [15]. The third strategy is the MR2 (multiple step with remote indices, version 2) [16], which takes full advantage of the parallel mechanism inherent in a distributed environment and the local spatial indices. Spatial join incurs the highest cost in spatial query processing. The MR2 strategy enables parallel processing of the geometry, and avoids the cost incurred in the construction of spatial indices for remote datasets. The MR2 strategy chooses the smaller of the two datasets to be transmitted in the network, and effectively blocks objects that do not pertain to the final results from progressing to the next step. The MR2 strategy is more efficient than the other two strategies discussed. However, the MR2 strategy demands modification of the spatial index. Assuming that an R* tree exists within the local site; then the local reference of the R* tree node should be revised to the global reference in the DSDB. Therefore, objects in the spatial database need to possess their global object identification. The R* tree join algorithm requires revision to conform to the references of the global identification. Ramirez's research is based on homogeneous spatial databases and remote spatial index sharing. Index sharing is based on the Global Object Identification in a heterogeneous distributed database, which is difficult to achieve. Therefore, the adaptability of MR2 is extremely restricted. These spatial join strategies do not consider the specificity of partitioning fragmentation.

In a distributed environment, the fragmentation or distribution of spatial data is quite important. In [17], Zhu has classified the fragmentation into several categories, such as horizontal fragmentation, vertical fragmentation, thematic fragmentation, zonal fragmentation and mixed fragmentation. Based on the zonal (partitioning) fragmentation, Zhu discussed the cross-border issues of topological join, and proposed a framework to solve topological queries. However, a compound query is more complicated and costly than a pure topological query. Further research on this problem is necessary and would make a significant contribution.

In general, the existing research includes a handful of studies on the optimization of spatial joins and distributed spatial joins, which are the basis of this study. Nevertheless, for compound query optimization involving both spatial join and buffer analysis operations, especially in a DSDB based on partitioning fragmentation, the current studies cannot meet the requirements for optimization.

3. The Join Optimization Principle for Compound Queries

When spatial data are divided by regions into multiple sub-datasets and stored in different sites, the spatial query must be assigned to these sites for execution. As a spatial join has two parameters, it is decomposed into multiple sub-joins. After processing all sub-joins, the results are merged for the final result. This section will classify spatial joins, and discuss the theory and method of compound join optimization.

3.1 The Classification of Spatial Topological Relationship Predicates and Spatial Joins

At present, the definition of a topological relationship set includes eight relationships based on the 9-Intersection Set Model [18]. The OGC (Open GIS Consortium) have developed a geographic information system standard for spatial query [19, 20]. This standard emphasizes that a spatial database should implement basic GIS operations. More than eight spatial topological relationship predicates exist, including *Crosses*, *Disjoint*, *Within*, *Contains*, *Equals*, *Touches*, *Intersects*, and *Overlap Disjoint*, *Touch*, *Cross*, *In* (*Within*) and *Overlap* are the five basic relationships with characteristics of completeness and exclusiveness [21, 22]. Several commercial spatial databases and spatial data engines, e.g. Oracle Spatial 11g [23] and ArcSDE 9.2 [24], refer to OGC standards in their implementation.

There are certain regularities in the CBJs with different spatial topological predicates. Zhu Xinyan classified the spatial topological relationships into two classes, as shown in Table 1, for the optimization of topological joins based on Partitioning fragmentation [17]. According to the class of a predicate, Zhu preselected a corresponding filterer for the joining fragments, thus pure topological queries can be optimized fairly well.

Table 1. The Classification of Spatial Topological Relationship Predicates

Classification	The names of Spatial Topological Relationship
1st class	*Within, Contains, Crosses, Equals, Touches, Intersects, Overlaps*
2nd class	*Disjoint*

However, compound queries are more complex than pure topological queries. Once a buffer analysis operation is involved in a spatial query, a general filterer based on spatial indices fails to function properly. Thus, this paper reclassifies spatial join queries. Not only topological relationship queries, but also the buffer analysis operations are considered. The classification is shown in Table 2. Zhu discussed the first and second class joins in the case of partitioning fragmentation. Join optimization of the third class is emphasized in this article. The join optimization of the fourth class is beyond the scope of this article and discussed elsewhere.

Table 2. Compound Spatial Joins Classification

Classification	Buffer operation	The name of Spatial Topological Relationship	Query name
1st class	*no*	*Within, Contains, Crosses, Equals, Touches, Intersects, Overlaps*	*Topological-intersect join Query*
2nd class	*no*	*Disjoint*	*Topological-disjoint join Query*
3rd class	*yes*	*Within, Contains, Crosses, Equals, Touches, Intersects, Overlaps*	*Compound Query*
4th class	*yes*	*Disjoint*	*Compound-disjoint Query*

In Section 1, the query Q1 is a third class spatial join. The third class of spatial join is just the compound join. Like the ordinary spatial join (the first class), if the data are partitioned by regions, then the decomposition of the query and optimization of the sub-queries is necessary.

3.2 The Cross-border Spatial Query Principle Based on Partitioning Fragmentation

The global spatial join must be decomposed into sub-joins between each pair of the *data fragments*. Assume that the global spatial relations (or spatial tables) R and S are partitioned seamlessly by *n* polygons $p_1, p_2, ..., p_n$. If these polygons do not overlap with each other, and encompass the entire spatial coverage seamlessly, then the polygon set P = $\{p_1, p_2, ..., p_n\}$ is called a Partition Set.

The partitioned fragments (PF) of R and S are:

$$PF_P(R) = \{R_1, R_2, \ldots, R_n\},$$

$$PF_P(S) = \{S_1, S_2, \ldots, S_n\}.$$

Here, PF indicates the partitioned fragments, the subscript P represents the Partition Set P, and R_i and S_i ($1 \le i \le n$) indicate the partitioned fragments of R and S respectively (as shown in Figure 2 (a) with $n = 3$).

To process a cross-border spatial query, the global join is translated into several fragment joins in accordance with the allocation rules [25,26]:

$$R \bowtie_{\theta_{sp}} S = \bigcup_{i=1}^{n} R_i \bowtie_{\theta_{sp}} \bigcup_{j=1}^{n} S_j = \bigcup_{i=1, j=1}^{n} (R_i \bowtie_{\theta_{sp}} S_j) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

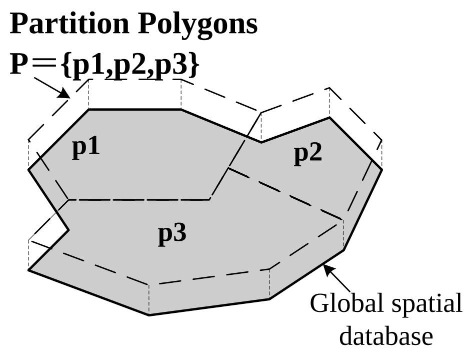
where, the symbol $\bowtie$ indicates a join, and θ_{sp} is the spatial join predicate, which can be a pure topological predicate (the 1st class), or a compound predicate (the 3rd class).

Zhu has proposed a method to divide the Spatial fragment joins, which are also called sub-joins, as depicted in (1), into two groups as follows:

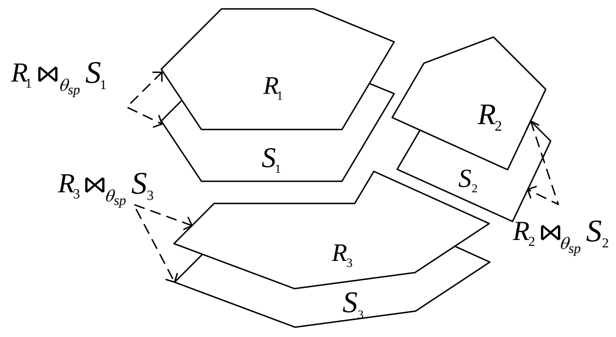
$$\text{Group 1:} \quad (R_1 \bowtie_{\theta_{sp}} S_1), \ (R_2 \bowtie_{\theta_{sp}} S_2), ..., (R_n \bowtie_{\theta_{sp}} S_n)$$

$$\text{Group 2:} \quad (R_1 \bowtie_{\theta_{sp}} S_2), \ldots, (R_1 \bowtie_{\theta_{sp}} S_n), (R_2 \bowtie_{\theta_{sp}} S_1), (R_2 \bowtie_{\theta_{sp}} S_3), \ldots, \quad (2)$$
$$, (R_2 \bowtie_{\theta_{sp}} S_n), \ldots, (R_n \bowtie_{\theta_{sp}} S_1), \ (R_n \bowtie_{\theta_{sp}} S_2), (R_n \bowtie_{\theta_{sp}} S_{n-1})$$

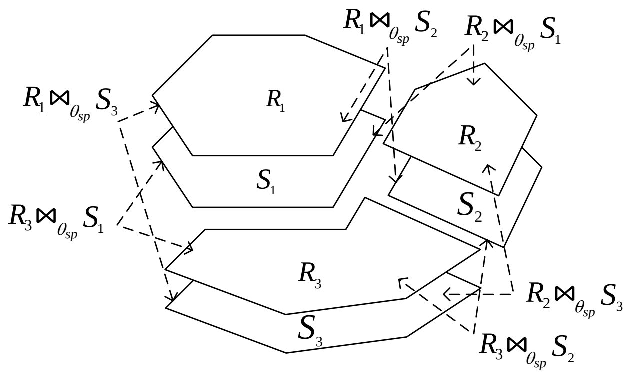
In Group 1, the two fragments of each sub-join have the same spatial extent; therefore, these sub-joins are called *same region joins* (or *non-cross-border joins*, NCBJs), as shown in Fig. 2(b) The join optimization problems in Group 1 have been resolved in centralized databases by existing methods. In Group 2, the two fragments in each sub-join do not overlap in their spatial extents; thus, these sub-joins are called *different regions joins* (or *cross-border joins*, CBJs), as shown in Figure 2(c). When the global spatial relation R and S are both partitioned by *n* regions, the number of fragment joins is n^2; the number of NCBJs is *n*, and the number of CBJs is *n*(n -1)*. Thus, the number of CBJs is greater than NCBJs and with the increase of regions *n*, the number of CBJs also grows rapidly. So, CBJs have two distinguishing features: (1) The data involved in CBJs are distributed; (2) Usually, there is a large quantity of CBJs.



（a）Partition set (n=3)


$R_1 \bowtie_{\theta_{sp}} S_1$
R_1
S_1
R_2
S_2
$R_3 \bowtie_{\theta_{sp}} S_3$
R_3
S_3
$R_2 \bowtie_{\theta_{sp}} S_2$

（b）Group 1：non-cross-border joins


$R_1 \bowtie_{\theta_{sp}} S_2$
$R_2 \bowtie_{\theta_{sp}} S_1$
$R_1 \bowtie_{\theta_{sp}} S_3$
R_1
R_2
S_1
$R_3 \bowtie_{\theta_{sp}} S_1$
S_2
R_3
$R_2 \bowtie_{\theta_{sp}} S_3$
S_3
$R_3 \bowtie_{\theta_{sp}} S_2$

（c）Group 2：cross-border joins

Figure 2. The Translation of the Global Join to Fragment Joins [17]

Therefore, optimization of CBJs is essential to the efficiency in processing compound queries.

3.3 The Buffer Zone Boundary-restricting Theorem of a Spatial Fragment

In a DSDB, a Compound Query may not only cross the boundaries of adjacent fragments, but also cross non-adjacent fragments (as shown in Figure 1). Whereas, the allocation rules in equation (1) indicate that, in a global query, more partitioned fragments will produce more CBJs. As in the previous analysis, whether or not these CBJs with buffer operations are useless cannot be determined statically. Failure to determine the uselessness of these CBJs could result in the waste of significant transmitting and computing resources.

Further discussion of optimization problems in compound queries requires certain essential definitions:

Definition 1. MBR Expansion: Assuming that the lower-left and upper-right point coordinates of a spatial object's MBR are (x_{min},y_{min}) and (x_{max}, y_{max}). Then, the MBR Expansion with a certain distance d ($d > 0$) of the object is a rectangle, with its lower-left and upper-right point coordinates being ($x_{min} - d$, $y_{min} - d$) and ($x_{max} + d$, $y_{max} + d$). The MBR of an object a is denoted as MBR(a), and the MBR Expansion of this object with a distance d is written as Expand(MBR(a), d).

Definition 2. A buffer of spatial objects and a spatial relation: The buffer of a spatial object is the object resulting from a buffer operation on the original object with a certain distance d ($d > 0$). The buffer of object a with a distance d is written as Buffer(a, d). The buffer of a spatial relation is a collection of objects obtained by performing a buffer operation on each object in the spatial relation. The buffer of a spatial relation R with a distance d is written as Buffer(R, d).

The MBR of the buffer of a spatial relationship R is a rectangle and denoted as MBR(Buffer(R, d)).

For any spatial fragment X, the MBR of X is marked as MBR(X). The Expansion of MBR(X) is expressed as Expand(MBR(X), d). If the radii of buffer operations on different objects in the spatial relation are different from each other, d acquires the maximum buffer radius.

Theorem 1. The buffer zone boundary-restricting theorem of a spatial fragment: the MBR of a spatial fragment's buffer with a certain distance d ($d > 0$) does not exceed the MBR expansion with a distance d ($d > 0$) of its partitioning region's border.

Proof: In Figure 3, s is the border polygon of spatial fragment X. The MBR of the buffer of s is expressed as MBR(Buffer(s, d)). The MBR Expansion of s is expressed as Expand(MBR(s), d). Based on the definitions of MBR and the MBR Expansion, they both are rectangles, and can be expressed by lower-left and upper-right corner coordinate values.

Assume that the bounding coordinates of MBR(Buffer(s, d)) in the four directions are (XL, YL, XH, YL); then, from the definition of MBR, the bounding values of the buffer of s (i.e., buffer(s, d)) in the four directions are (XL, YL, XH, YL). Therefore, based on the definition of the buffer operation, the bounding coordinates of s itself in the four directions are (XL+d, YL+d, XH-d, YL-d). Then, by the definition of MBR, the boundary values of MBR(s) in the four directions are (XL+d, YL+d, XH-d, YL-d). By the definition of MBR Expansion, the lower-left and upper-right point coordinates of Expand(MBR(s), d) are (XL, YL, XH, YL). Therefore,

$$\text{MBR(Buffer}(s, d)) = \text{Expand(MBR}(s), d) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Assume that PS is the acronym of Point Set [1]; then, according to the definition of a Partition Set (in Section 3.1), PS(X) $\subseteq$ PS(s) is true; then, PS(Buffer(X, d)) $\subseteq$ PS(Buffer(s, d)); therefore, PS(MBR(Buffer(X, d))) $\subseteq$ PS(MBR(Buffer(s, d))), combined with (3), that is,

$$\text{PS(MBR(Buffer(X}, d))) \subseteq \text{PS(Expand(MBR}(s), d))$$

[1] Point set topological spatial relationship is described in detail by Egenhofer and Franzosa [18].

Therefore, Theorem 1 is proved.

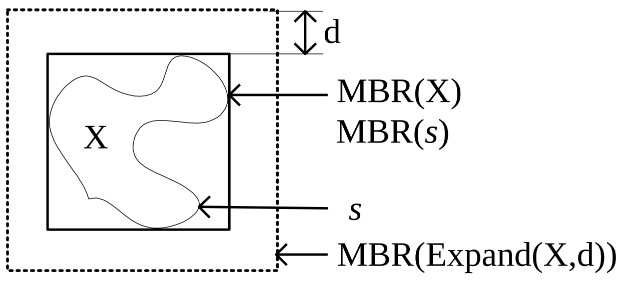


Figure 3. An Illustration of the Proof for Theorem 1

Applying Theorem 1, confirms that the result of the buffer operation on fragment X will not exceed the Expand(MBR(*s*), *d*); therefore, a number of useless fragment joins from cross-border buffer queries are excluded. An issue that must be pointed out here is that MBR(X) and MBR(S) are not necessarily equal; X is fixed, but the partition border *s* might exceed the extent of X. To remove the fragment joins, either MBR(X) or the MBR(*s*) can be chosen and stored in the metadata as the *extent* property of the spatial fragment. One approach involves storage of the MBR of partition border *s* in the *extent* during fragment partitioning, and the other approach involves recalculation of the MBR of the fragment X and then storing it in the *extent*. In this article, MBR(X) and MBR(*s*) are not distinguished from each other. In the following discussion, both are expressed as MBR (X) and represent the *extent* property of the spatial fragment.

3.4 The Removal Rule for Fragment Joins of a Compound Query

According to theorem 1, removal rule of a CBJ can be obtained. Usually, only one parameter involves a buffer operation in the two CBJ parameters (if two parameters both involve buffer operation, it can be turned into a buffer operation based on one parameter; the buffer radius is the sum of buffer radii on the two parameters). In order to make it easy to discuss, this article distinguishes the two spatial fragments that participate in the compound join, and gives their definitions as follows:

Definition 3. Main Fragment and Secondary Fragment: In partitioning fragmentation, a sub-join of a compound join can be expressed as $Y \bowtie_{\theta_{sp3}} Buffer(X,d)$. Here X, Y are zonal fragments; X is the main fragment for the *Buffer* operation performed, and Y is the secondary fragment.

In this difinition, sp3 means the join is a spatial join and it belongs to the third class (a compound join). Based on definition 3, here puts forward **the removal rule of the CBJs of a compound join**.

Rule 1. The removal rule of the compound join: To perform the compound join of two spatial fragments X and Y, assuming X is the main fragment, if the MBR Expansion of the main fragment X does not *Intersect* the MBR of the secondary fragment Y, this fragment join can be excluded. Therefore, if the value of the expression *Intersects* (Expand(MBR(X), *d*), MBR(Y)) is FALSE, the fragment join $Y \bowtie_{\theta_{sp3}} Buffer(X,d)$ can be eliminated.

The removal rule determines which fragment join will be removed. To illustrate, assume that the buffer operation will perform on X in the compound join of X and Y. The buffer of X with a radius *d* is X'. Then, the compound join can be translated into the first class of fragment joins, which is a pure spatial topological join of X' and Y. If MBR(X') and MBR(Y) do not *Intersect*, the fragment join can be excluded. Here, MBR(X') are equal with Expand(MBR(X), d). Therefore, if the Expansions of X and the MBR of Y are disjoint, then the compound join of X and Y is useless and can be eliminated.

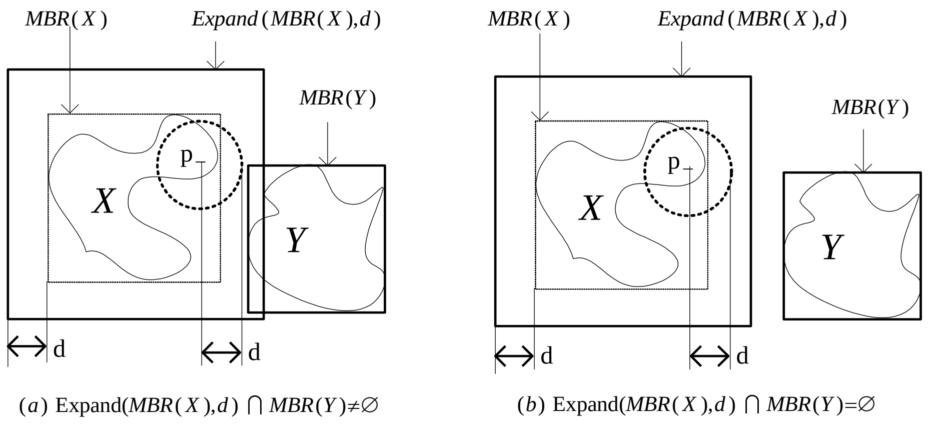

$MBR(X)$
$Expand(MBR(X),d)$
$MBR(Y)$
p
X
Y
d
d
(a) Expand($MBR(X),d) \cap MBR(Y) \neq \varnothing$
$MBR(X)$
$Expand(MBR(X),d)$
$MBR(Y)$
p
X
Y
d
d
(b) Expand($MBR(X),d) \cap MBR(Y) = \varnothing$

Figure 4. An Instance of Removing Buffer Joins between Fragments

From Figure 4, we observe that the buffer zone of point P exceeds the MBR of fragment X. In Figure 4(a), the Expand(MBR(X), *d*) *Intersects* with the MBR(Y), and the fragment join of X and Y in the cross-border query is to be retained; in Figure 4(b), the Expand(MBR(X), *d*) does not *Intersect* with MBR(Y); therefore, the fragment join of X and Y can be excluded. Rule 1 determines removals or retentions dynamically in run-time and removes the useless cross-border joins to optimize a global query.

Rule 1 can reduce the number of the cross-border fragment joins of a compound query in partitioning fragmentation. Rule 1 is **the Removal Rule for Fragment Joins** of a compound query.

3.5 The Filtering Rule for Fragment Joins of a Compound Query

After the application of the Removal Rule (Rule 1), the fragment joins retained from the compound query can be further optimized by reducing the number of spatial objects that participate in the buffer and join operations. From Sections 3.3 and 3.4, the MBR Expansions of the joined fragments form the basis for determining which CBJs are to be removed. Similarly, the MBR Expansions of the joined fragments can be employed as the basis of the filtering method for the CBJs.

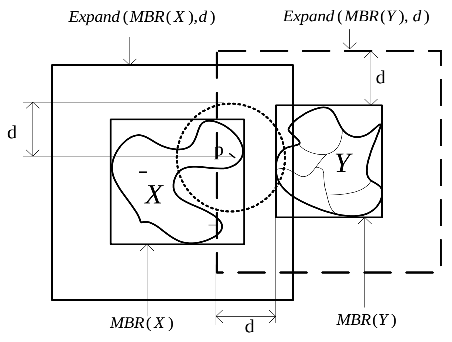


Figure 5. The Filtering Window for a Buffer Join

From Figure 5, two fragments X and Y will form a CBJ, while X is the main fragment. If the MBR Expansion of X *Intersects* the MBR of Y, then the fragment join of X and Y is retained in the removal step for further processing. If fragment X is to be involved in the buffer operation, according to Theorem 1, the buffer result of X does not exceed the MBR Expansion of X; therefore, objects in Y that *Intersects*[2] any object of the buffer result of X must *Intersect*[3] Expand(MBR(X), *d*). Conversely, an object in X whose buffer zone *Intersects* any object in Y must *Intersect* with Expand(MBR(Y), *d*). Therefore, the Expand(MBR(Y), *d*) can be used as a window to filter X, which not only reduces the number of objects that participate in the fragment join, but also reduces the number of objects involved in buffer operations, optimizing the fragment join.

Theorem 2 is given and proved as follows.

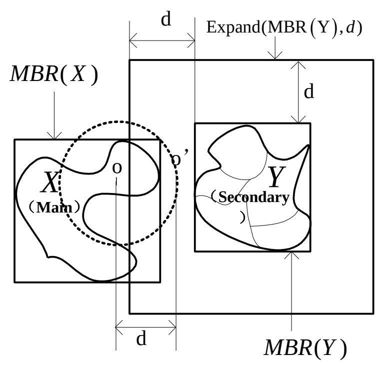


Figure 6. An Illustration of the Filtering Theorem of the Fragment Join

Theorem 2. The filtering theorem for a compound fragment query. For a fragment join of a compound query, if objects in the main fragment that satisfy the query must *Intersect* with the MBR Expansion of the secondary fragment.

[2] Here the predicate Intersects can be replaced by any predicates except Disjoint; that is all the predicates belongs to the first class of spatial relationship, as described in Table 1.

[3] Here the predicate Intersects cannot be replaced by any other predicates listed in Table 1.

Proof: In Figure 6, X is the main fragment and Y is the secondary fragment. Assume that X and Y (that include the interior and the boundary of X and Y) can be expressed as two point sets:

$$PS_X = \{ P / P \in X \},\quad PS_Y = \{ P / P \in Y \}$$

Similarly, MBR(X) and MBR(Y) are expressed as two point sets:

$$PS_{MBR(X)} = \{ P / P \in MBR(X)\},\quad PS_{MBR(Y)} = \{ P / P \in MBR(Y) \}$$

Then,

$$PS_Y \subseteq PS_{MBR(Y)} \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (4)$$

Apagoge is used here. *o* is an object in X that does not *Intersect* with Expand(MBR(Y), *d*), and *Disjoint*(*o*, Expand(MBR(Y), *d*)) is true. According to the definitions of the MBR Expansion, the *Disjoint* predicate and the distance between two spatial objects in the OGC Simple Feature Access Specification (OGC, SFA Common), it holds that:

$$\text{Distance (o, MBR(Y))} > d$$

Combining with formula (4), then:

$$\text{Distance (o, Y)} > d,$$

Therefore, *Disjoint* (*Buffer* (*o*, *d*), Y) is true. According to the classification in Section 3.2, the object *o* and the objects in Y do not satisfy the third class of spatial relationship with buffer radius *d*. Theorem 2 is proved.

Theorem 2 demonstrates that Expand(MBR(Y), d) can be applied to filter fragment X, and called the **Filtering Rule for Fragment Joins** of a compound query.

Rule 2. The Filtering Rule for Fragment Joins: To process a compound join of fragments X and Y, If the join cannot be removed, assuming X is the main fragment, before calculating the buffer of the main fragment, the MBR Expansion of secondary fragment Y is used to filter main fragment X.

3.6 The Join Optimization Principle for CBJs and its Formalization

By using the Removing Rule, useless CBJs are removed. The Filtering Rule filters and reduces the spatial objects needed to be transmitted and calculated in the buffer operation. The Filtering Rule also reduces the spatial objects needed for the calculation of spatial topological relations. This is the key idea behind optimization of CBJs. This strategy for join optimization is especially aimed toward the spatial data partitioning fragment method, and therefore termed the Partitioning Fragments Join Strategy (PFJS).

If both X and Y are spatial fragments partitioned by two polygons SX and SY, and X is the main fragment and the buffer radius is d. from the above discussion, the filter $\theta^f_{sp3}(X)$ is defined as:

$$\theta^f_{sp3}(X) = \{I(a) \mid \exists X,\ \exists Y, \exists S_X,\ \exists S_Y,\ X \in PF_{S_Y},\ Y \in PF_{S_X},\ \forall a \in X, \\ \text{Intersects(a, Expand(MBR(}Y\text{), d))=true} \} \quad \dots (5)$$

The filter $\theta^f_{sp3}(X)$ can be marked as $\theta^f_{Expand(MBR(Y),d)}(X)$, which represents a filter for the PFJS. Assuming X and Y are spatial fragments partitioned by two polygons S_X and S_Y. X is the main fragment and the buffer radius is *d*. The θ^{cb}_{sp3} join on X and Y, where *cb* indicates a cross-border join, can be translated into an equivalent process including the following three steps: first, perform the filter operation $\theta^{f}_{Expand(MBR(Y),d)}(X)$; second, apply the buffer operation with a radius d on the filtered results; and, finally, complete the spatial join (θ_{sp3}). The rule is expressed as:

$$\eta_d(X) \bowtie_{\theta^{cb}_{sp3}} Y = \eta_d(\sigma_{\theta^{f}_{Expand(MBR(Y),d)}}(X)) \bowtie_{\theta_{sp-3}} Y \qquad (6)$$

4. Comparison and Analysis of Three Strategies of Compound Queries

In the case of partitioning fragmentation, a global compound query is decomposed into multiple sub-queries (also named as sub-join or fragment-join). These sub-queries are homogeneous with the global query. They are all compound queries, but the data concerned is quite different. For such a fragment join, Naive Join Strategy, Semi-Join Strategy and Spatial Semi-Join Strategy [13] can be used to get the result. Based on the discussion in section 3, this article proposes a specific method for CBJs. In order to determine the direction of data transmission, assume that two fragments R_i and S_j are located at Site A and B, and assume Size(R_i) < Size(S_j). Thus, usually R_i should be transmitted from site A to site B for minimal transmission cost.

4.1 Naive Join Strategy (NJS)

The Naive Join Strategy, put forward by D.J.Abel in 1995, transmits all the records of one parameter from one site to another [14]. Then the spatial join algorithm for centralized spatial databases is used. The main steps are as follows:

(a) Transmit R_i from Site A to Site B;

(b) Perform buffer operation on R_i at Site B, and obtain R_i';

(c) Build a spatial index to R_i' at Site B;

(d) Perform spatial topological join for R_i' and S_j at site B, and obtain the result.

The method is simple and adaptive when the number of objects in R_i is small. Otherwise, it has a high cost for transmission and execution because many useless spatial objects are transmitted among sites and their spatial relationships are fruitlessly calculated.

4.2 Spatial Semi-Join Strategy (SSJS)

The Spatial Semi-Join Strategy reduces the cost of transmission. The main processing steps are as follows:

(a) Perform buffer operation on R_i at Site A, and obtain R_i';

(b) Write R_i' into the database at site A and build a spatial index;

(c) Calculate the MBR for every object in S_j at Site B, then get a non-repeating set of MBR, called S_j';

(d) Transmit S_j' to Site A;

(e) Write S_j' into the database at site A and build a spatial index;

(f)Complete the Spatial Semi-Join operation "*Intersects*(R_i'.shape, S_j'.shape) = TRUE " on R_i' and S_j' at Site A and obtain the join results R_i";

(g) Transmit R_i" to Site B;

(h) Write R_i" into the database at site B and build a spatial index;

(i) Execute the spatial operation "*Intersects*(R_i".shape, S_j.shape)=TRUE " on R_i" and S_j at Site B and obtain the join result R_j" .

SSJS is much more complex than NJS. At the step (d), S_j' or R_i' need to be transmitted to the other site depending on which one is smaller. The same question is considered at step (g). SSJS is more efficient than NJS because it reduces transmission costs.

4.3 The Partition Fragments' Join Strategy (PFJS)

In the framework of a DSDB, the FPJS needs a data dictionary module. The data dictionary and scheduler can be located on the same machine, which are independent of local spatial databases. The data dictionary stores important metadata for every fragment, including MBR information, the size of fragments, etc. The steps of PFJS are as follows:

(a) On the site where the scheduler exists, find the MBRs of fragment R_i and S_j, compute Expand(MBR(R_i), *d*) and Expand(MBR(S_j), *d*) (e.g. d=10KM);

(b) At Site A, filter fragment R_i with Expand(MBR(S_j), *d*), obtain the result R_i';

(c) At Site B, filter fragment S_j with Expand(MBR(R_j), *d*), obtain the result S_j';

(d) Compare the size of R_i' and S_j', if Size(R_i')<Size(S_j'),transmit R_i' to the B site ,otherwise, transmit S_j' to at Site A;

(e) Assume that the data transmitted is R_i', compute the buffer for R_i' at site B, then get R_i";

(f) Write R_i" into the database at site B and build a spatial index;

(j) At Site B do spatial topological join for R_i"and S_j', then obtain the result.

If the dataset transmitted is S_j', calculate the buffer for S_j' at Site A, then execute the intersection similarly. PFJS takes the characteristics of CBJs into account, simplifies the solution and ensures optimal efficiency.

4.4 Complexity Analysis of PFJS

Assume that the global spatial relation R and S are partitioned by *n* regions. Then, both R and S have *n* fragments. The global join of R and S will have n^2 sub-joins; the number of NCBJs is *n*, while the number of CBJs is *n**(*n* - 1). The number of CBJs is greater than that of NCBJs. The number of CBJs also grows rapidly as *n* increases. In general, NCBJs can be solved efficiently because the relevant datasets are well indexed and limited to a single local database. The two datasets of CBJs are usually distributed, and the process of the CBJs refers to data transmission, spatial index reconstruction, and spatial-related calculation, including calculation of buffer, calculation of spatial topological relations, etc. For a CBJ expressed as

$$\eta_d(R_i) \bowtie_{\theta_{sp3}^{cb}} S_i ,$$

assume that the sizes of R_i, S_i are M_i, N_i; then, the cost of the CBJ can be expressed as

$$\begin{aligned} C &= C_{I/O}+C_{buff}+C_{tran}+C_{join} \\ &= C_1*k_1*(M_i + N_i) + C_2*k_2*M_i + Min(C_3*k_2*M_i, C_3*N_i) + C_4*k_2*M_i*N_i \end{aligned} \quad (7)$$

In formula (7), the total cost is comprised of I/O cost ($C_{I/O}$), buffer-calculating cost (C_{buff}), data transmitting cost (C_{tran}), and joining cost (C_{join}). For a compound query, C_{buff} and C_{join} are dominant. In a distributed environment, C_{tran} may be significant if the bandwidth is limited. The costs of three join strategies can be expressed as:

$$C_{NS} = C_1*(M_i + N_i) + C_2*M_i + Min(C_3*\alpha*M_i, C_3*N_i) + C_4*M_i*N_i \quad (8)$$

$$\begin{aligned} C_{SSJS} &= C_1*k_1*(M_i + N_{i+}\Delta) + C_2*M_i + (Min(C_3*k_2*\alpha*M_i, C_3*N_i) + \beta*N_i) \\ &\quad + (C_4*k_2*M_i*N_i + C_5*M_i*N_i) \end{aligned} \quad (9)$$

$$C_{PFJS} = C_1*k_1*(M_i + N_i) + C_2*k_2*M_i + Min(C_3*k_2*\alpha*M_i, C_3*N_i) + C_4*k_2*M_i*N_i \quad (10)$$

In formula (8-10), C_1, C_2 and C_3 are the average I/O cost, buffer-calculating cost, transmitting cost of each object. C_4 is the average joining cost of each object pair. C5 is the average cost of calculating the spatial topological relation of a MBR and a spatial object (generally less than C4).The coefficients k_1, k_2, k_3 and k_4 indicate the proportion of the relevant data to the whole dataset. α is a coefficient that describes the change of data size caused by buffer calculation. Δ indicates the increment of data size in I/O operation caused by obtaining the MBRs and executing a Semi-Join; while β indicates the average data size of MBR.

Analysis shows that the complexity of I/O, buffer calculation, and transmission are O(n). However, that of joining is O(m*n). If the data set related to fragment join is invariable, then α, β, Δ, C1, C2, C3 and C4 are almost fixed. Because of $0\leq k_1\leq 1$ and $\Delta>0$, PFJS is superior in I/O cost. k_2 is the key value for CBJ optimization since $0\leq k_2 \leq 1$. PFJS reduced the cost of buffer calculation, whereas both SSJS and NJS could not reduce buffer calculation costs since $0\leq k_2\leq 1$. For the transmission and joining costs, SSJS is usually better than NJS, but worse than the PFJS. The filtration of SSJS and PFJS are not done in the same way, and therefore their k_2 values are not same. SSJS is better than PFJS when PFJS filtration cannot play a good role (k_2 is close to 1). Considering the characteristics of CBJs, by reducing k_2, the PFJS processes CBJs efficiently. The time complexity of PFJS is lower than that of NJS and SSJS.

5. Experiments and Analysis

Experiments were designed to test and verify the performance of PFJS. A DSDB environment was set up. Several data sets were processed then stored in this DSDB. Two different spatial queries are selected to prove the applicability of PFJS. The first one is a spatial join with a pure partitioning fragmentation, while the second one with a mixed fragmentation.

5.1 Experimental Environment and Dataset

Experiments were performed on four workstations (DELL PowerEdge SC430, CPU P4 3.0GHz, memory 2GBytes, disk space 80GBytes) connected through a 100Mbps Switch. The operating systems are Windows 2003 Server. The Database Systems (DBSs) are Oracle 10G. The Internet Communication Engine (ICE) is employed to construct the distributed environment. The Spatial Components built-in Oracle 10G are applied to calculate the spatial topological relationship between objects. The development platform is Microsoft Visual C++ 2005. A prototype DSDB is developed through an agent-based model. The Local Spatial Database Systems are Oracle Spatial 10g; the prototype has a global schema and a management system, which is also known as a Multi-Spatial Database System (MSDBS).

The data are a part of the fundamental geographical data of China. The scale is 1:250,000. There are three datasets: Dataset 1 (Regions) is the municipal boundaries of thirteen provinces (Beijing, Tianjing, Hebei, Shanxi, Shanghai, Jiangsu, Zhejiang, Shandong, Anhui, Hubei, Jiangxi, Hunan, Henan), and are partitioned into three parts. They are stored as Regions1 (East China), Regions2 (North China) and Regions3 (Central China). Dataset 2 (Highways) includes the highways in these three regions, and is partitioned into three parts as in Dataset 1; these three parts are stored as Highways1, Highways2 and Highways3. Dataset 3 (Railways) includes railway networks of China, and not partitioned yet. The objects in Dataset 1 are all polygons and, Dataset 2 and 3 are polylines. An overview of the entire data is presented in Figure 7.

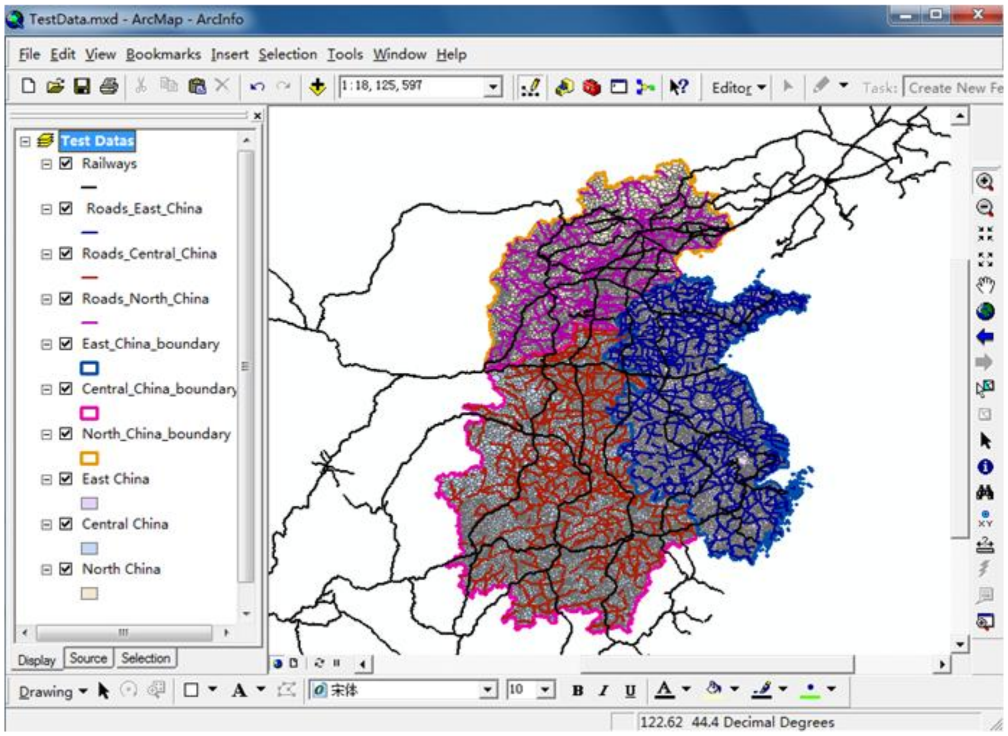


Figure 7. An Overview of All Test Data

The details of the datasets are described in Table 3.

Table 3. Fragments Partition and Data Size of All Test Data

	Regions			Highways			Railways
	Regions1 (East China)	Regions2 (North China)	Regions3 (Central China)	Highways1 (East China)	Highways2 (North China)	Highways3 (Central China)	(China)
Number of objects	8605	4966	7399	1622	925	1040	1763
Data size	63.0MB	40.5MB	79.0MB	3.4MB	2.4MB	3.3MB	4.0MB
Site	Site 1	Site 2	Site 3	Site 1	Site 2	Site 3	Site 4

5.2 Methodology

The following two queries were applied to analyze the relationship between the partitioning of data and the efficiency of compound join.

Query 2:

```
SELECT  S.name
FROM  Regions S, Highways H
WHERE  Intersect(S.shape, Buffer(H.shape, 5, 'unit=KM'))= 'TRUE';
```

Query 3:

```
SELECT  S.name, R.name
FROM  Regions S, Railways R
WHERE  Intersect(S.shape, Buffer(R.shape, 5, 'unit=KM'))= 'TRUE';
```

The meanings of both queries are similar. However, the extent and the partitioning of the datasets are different. This makes the significance of two queries different.

In Query 2, both datasets were partitioned into three parts in the same way. Then, the global query (Query 2) was decomposed into nine sub-queries SQ1–SQ9 according to formula (1) in Section 3.1, as depicted in Table 4. The nine sub-queries were divided into two groups based on formula (2). SQ1–SQ3 in Group 1 were NCBJs, and dissolved by the local Spatial DBMS. SQ4–SQ9 in Group 2 were CBJs, and analyzed.

In Query 3, the partitioning methods of the two datasets different. The municipal boundaries were partitioned in the same way as in Query 2, while the Railways were not yet partitioned. Therefore, Query 3 was decomposed into three sub-queries SQ10–SQ12, depicted in Table 4. SQ10–SQ12 are CBJs.

Table 4. The Fragmentation and Sub-queries Qrouping of the Test Queries

	Query 2		**Query 3**	
	Sub-Queries	**Joining Fragments**	**Sub-Queries**	**Joining Fragments**
Group 1: Non-cross-border fragment sub-queries	**SQ1**	Regions1↔ Highways1		
	SQ2	Regions2↔ Highways2		
	SQ3	Regions3↔ Highways3		
Group 2: Cross-border fragment sub-queries	**SQ4**	Regions1↔ Highways2	**SQ10**	Regions1↔ Railways
	SQ5	Regions1↔ Highways3	**SQ11**	Regions2↔ Railways
	SQ6	Regions2↔ Highways1	**SQ12**	Regions3↔ Railways
	SQ7	Regions2↔ Highways3		
	SQ8	Regions3↔ Highways1		
	SQ9	Regions3↔ Highways2		

In the Multi-Spatial-DBMS framework used in this work, a Data Dictionary Module (DDM), Agents, and a Scheduler Program (SP) were implemented. In the integration, the global schema of Regions, Highways and Railways were mapped to the local schemata through the DDM. The MBRs and other information pertaining to each fragment used in PFJS were stored in the DDM. The Scheduler collected the mapping information from DDM and translated the global queries (Query 2 and Query 3) into fragment joins (sub-queries) according to the formulae presented in Section 3.1. Thus, each sub-query was implemented by three strategies as depicted in Section 4. The results were compared and analyzed.

In this article, SQ4–SQ12 were implemented with three methods, the Naive Join Strategy, Semi-Join Strategy and the PFJS. The costs of these three methods in each phase were compared and analyzed, including filtering (FiltTime), buffering (BuffTime), transmitting (TranTime), joining (JoinTime), and the total cost (TotalTime).

5.3 Comparison of Performance in Processing a Compound Query with Partitioning Fragmentation

Query 2 is a typical compound query in partitioning fragmentation. Its sub-queries SQ4–SQ9 are CBJs. The six sub-queries are implemented by each strategy described in Section 4.

The comparison of the total cost of the six sub-queries performed by three strategies is shown in Figure 8. For different sub-queries, the three strategies exhibit different performance. Overall, the SJS performs better than NJS, while PFJS improves the performance even more.

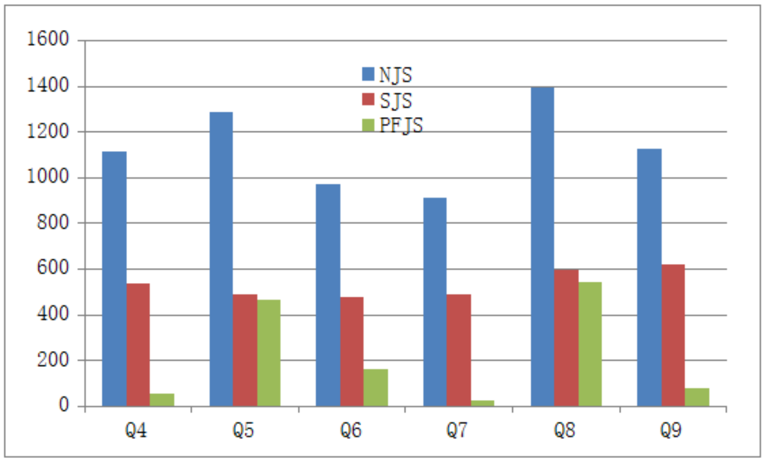


Figure 8. Comparison of the Total Cost of the Three Strategies (in Seconds)

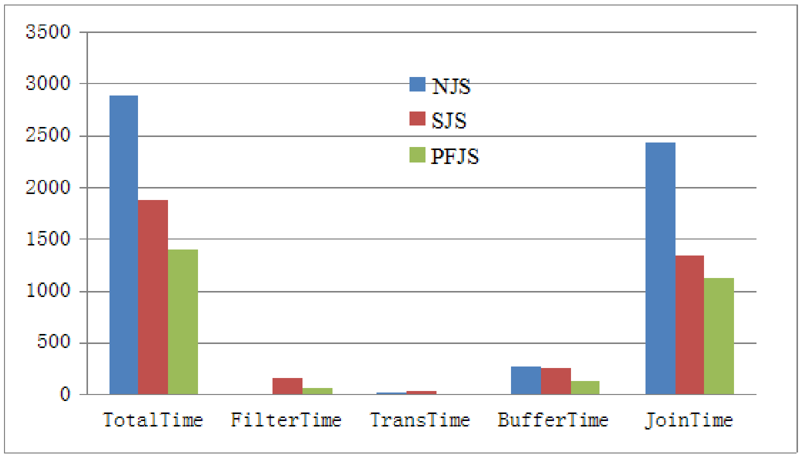


Figure 9. Comparison of the Overall Costs of the Three Strategies (in Seconds)

The overall results are presented in Figure 9. The costs of the total, filtering, data-transmitting, buffering and joining are compared. These costs are different in their order of magnitude, indicating that the buffering and joining costs are dominant in the compound joins, while the filtering and transmission costs are lower. If the data size in the compound join is increased, the differences will become more obvious. The time-consuming characteristic of the spatial operation (spatial joining, buffering) is verified here. In a distributed environment, the CPU (BuffTime, JoinTime) and communication costs (TransTime) are dominant; in the experiment, LAN was used, however, in the WAN, the transmission cost may increase significantly. The NJS did not use any filtration optimization, and subsequently all its primary costs are quite high. The spatial SJS considers the characteristics of the distributed environment, and reduces the transmission and joining costs. The spatial SJS neglects the filtering of the objects required for the buffer calculation; therefore, it cannot reduce the buffering cost. PFJS considers the buffer calculation in compound queries, and takes partitioning into account. The MBR Expansion of the fragment is used for filtering in the PFJS, and works efficiently. The number of objects required to perform the buffer calculation and joining are reduced, as well as the objects required to be transmitted. The PFJS obtained the greatest efficiency among all three strategies.

5.4 Comparison of Performance in Processing a Compound Query with Mixed Fragmentation

Based on the partitioning and distribution schema of the data, Query 3 was a compound query with mixed fragmentation. The sub-queries SQ10–SQ12 are CBJs. The three CBJs were implemented with the NJS, SJS and PFJS.

In the mixed fragmentation case, the overall performance of the PFJS is significantly better than that of SJS, as shown in Figure 10. Through an extra filtering in PFJS, the buffering and joining calculation costs in the PFJS were greatly reduced. The transmission cost was also the lowest among the three strategies. The SSJ reduced the joining cost, but did not reduce the buffering cost. The buffering cost is much higher than the joining cost for the size of the railway dataset.

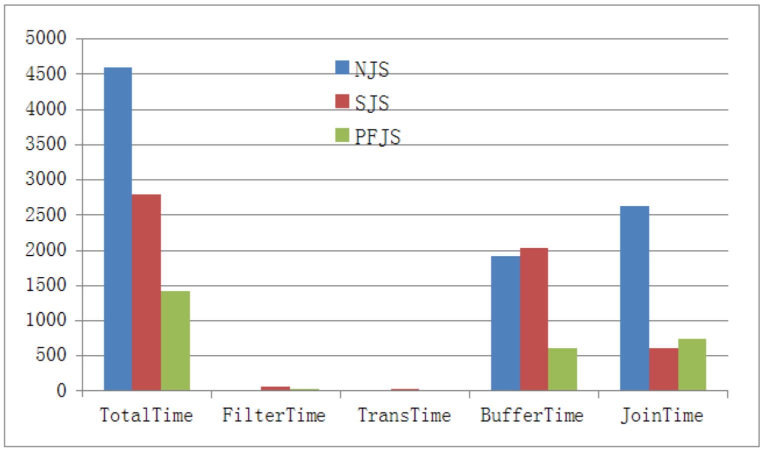


Figure 10. Comparison of Costs of the Three Strategies with Mixed Fragmentation (in Seconds)

In most cases of partitioning fragmentation or mixed fragmentation, the PFJS can filter the objects efficiently. The PFJS, nonetheless, is not always the best method for compound queries. In certain cases of mixed fragmentation, if the MBRs of the two fragments concerned in the sub-queries are very close, the filtering optimization of PFJS may not be obvious. In an extreme case, if the MBRs are the same, the filtering step of the PFJS is invalid, and then the performance of the PFJS is worse than that of the NJS.

6. Summary and Future Prospects

The cross-border spatial query is an inherent problem in a DSDB based on partitioning fragmentation. A key technology is the optimization of CBJs. In this article, two approaches to improve CBJs are considered: one is to reduce the number of fragment joins (removing the useless fragment joins); the other is to reduce the number of objects that participate in the fragment joins. Based on a classification of spatial queries, this article proves two theorems and proposes two rules for the optimization of

CBJs. The join strategy PFJS is also proposed. Experiments were designed to verify the performance of PFJS. The results indicate that PFJS is efficient.

However, there are limitations in the proposed solution because distributed spatial query processing is quite complicated, especially with partitioning fragmentation. Many issues must be considered in future research, for example, the selection of sites should take the capabilities of the nodes and the networks into account. Questions abound such as how to generate a global scheduling plan for optimal or moderate efficiency and methods of parallel execution for spatial-related calculations.

Acknowledgements

This work was supported by the the National Natural Science Foundation of China under Grant No. 40971232, 41271401 and 51279151; the National High-Tech Research and Development Plan of China under Grant No. 2011AA010502; the National Science and Technology Supporting Program of China under Grant No. 2012BAH35B03. It was also supported by the Fundamental Research Funds for the Central Universities.

References

[1] M. R. Ramirez and J. M. d. Souza, “Distributed Spatial Query Processing over MPI”, http://cronos.cos.ufrj.br/publicacoes/reltec/es59603.pdf: 8, **(2003)**.
[2] M. R. Fornari, J. L. D. Comba and C. Lochpe, “Query optimizer for spatial join operations”, Proceedings of the 14th annual ACM international symposium - Advances in geographic information systems, Virginia, USA, **(2006)**, [doi: 10.1145/1183471.1183508].
[3] R. Laurini, “Spatial multi-database topological continuity and indexing: a step towards seamless GIS data interoperability”, International Journal of Geographical Information Science, vol. 12, no. 4, (1998), pp. 30, [doi: 10.1080/136588198241842].
[4] J. A. Orenstein, “Spatial query processing in an object-oriented database system”, SIGMOD Rec., vol. 15, no. 2, **(1986)**, pp. 326–336, [doi: 10.1145/16894.16886].
[5] T. Brinkhoff, H. P. Kriegel, R. Schneider and B. Seeger, “Multi-step processing of spatial joins”, SIGMOD Rec., vol. 23, no. 2, **(1994)**, pp. 197–208, [doi: 10.1145/191843.191880].
[6] A. Guttman, “R-trees: a dynamic index structure for spatial searching”, Proceedings of the 1984 ACM SIGMOD international conference on management of data, Boston, Massachusetts, ACM, **(1984)**.
[7] N. Beckmann, H. P. Kriegel, R. Schneider and B. Seeger, “The R*-tree: an efficient and robust access method for points and rectangles”, SIGMOD Rec., vol. 19, no. 2, **(1990)**, pp. 322–331.
[8] H. Samet, “Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS”, Addison-Wesley, Reading, **(1990)**.
[9] H. H. Park, Y. J. Lee and C. W. Chung, “Spatial query optimization utilizing early separated filter and refinement strategy”, Information Systems, vol. 25, no. 1, **(2000)**, pp. 1–22.
[10] M. L. Lo and C. V. Ravishankar, “Spatial joins using seeded trees”, Proceedings of the 1994 ACM SIGMOD international conference on management of data, Minneapolis, Minnesota, United States, ACM, **(1994)**, pp. 209–220, [doi: 10.1145/191839.191881].
[11] M. L. Lo and C. V. Ravishankar, “Spatial hash-joins”, Proceedings of the 1996 ACM SIGMOD international conference on Management of data, Montreal, Quebec, Canada, ACM, **(1996)**, pp. 247–258, [doi: 10.1145/233269.233337].
[12] N. Mamoulis and D. Papadias, “Slot index spatial join”, IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 1, **(2003)**, pp. 211–231, [doi: 10.1109/TKDE.2003.1161591].
[13] E. Jacox and H. Samet, “Spatial join techniques”, ACM Transactions on Database Systems, vol. 32, no. 1, **(2007)**, pp. 44, [doi: 10.1145/1206049.1206050].
[14] D. J. Abel, B. C. OOi, K. L. Tan, R. Power and J. X. Yu, “Spatial Join Strategies in Distributed Spatial DBMS”, Proceedings of the 4th International Symposium on Advances in Spatial Databases, **(1995)**, Springer-Verlag, pp. 348–367, [doi: 10.1007/3-540-60159-7_21].
[15] K. L. Tan, B. C. Ooi and D. J. Abel, “Exploiting spatial indexes for semijoin-based join processing in distributed spatial databases”, IEEE Transactions on Knowledge and Data Engineering, vol. 12, no. 6, **(2000)**, pp. 920–937.
[16] M. R. Ramirez and J. M. d. Souza, “Distributed Processing of Spatial Join”, Tese DSc, Coppe/UFRJ, **(2001)**, pp. 8.

[17] Z. Xinyan, Z. Chunhui, G. wei and X. Yu, "Distributed Spatial Data Fragmentation and Cross-Border Topological Join Optimization", Journal of Software, vol. 22, no. 2, **(2011)**, pp. 269 −284.
[18] M. J. Egenhofer and R. D. Franzosa, "Point Set Topological Spatial Relations", International Journal of Geographical Information Systems, vol. 5, no. 2, **(1991)**, pp. 161–174, [doi: 10.1080/02693799108927841].
[19] Open Geospatial Consortium (OGC), Simple feature access - Part 1: Common architecture, OpenGIS Implementation Specification for Geographic information, **(2006)**.
[20] Open Geospatial Consortium (OGC), Simple feature access - Part 2: SQL option, OpenGIS Implementation Specification for Geographic information, **(2006)**.
[21] Open GIS Consortium (OGC), OpenGIS Simple Features Specification For SQL Revision 1.1, http://www.opengeospatial.org/standards/sfs.c, **(1999)**.
[22] E. Clementini, P. D. Felice and P. van Oosterom, "A Small Set of Formal Topological Relationships Suitable for End-User Interaction", Proceedings of the Third International Symposium on Advances in Spatial Databases, **(1993)**, Springer-Verlag, [doi: 10.1007/3-540-56869-7_16].
[23] R. Kothuri and A. Godfrind, "Pro Oracle Spatial for Oracle Database 11g", New York, Springer-Verlag, **(2007)**.
[24] ESRI, **(2007)**, ArcSDE 9.2 Developer Help, http://edndoc.esri.com/arcsde/9.2/.
[25] D. Kossmann, "The state of the art in distributed query processing", ACM Computing Surveys (CSUR), vol. 32, no. 4, **(2000)**, pp. 422–469, [doi: 10.1145/371578.371598].
[26] M. T. Ozsu and P. Valduriez, "Principles of Distributed Database Systems (Second Edition)", Tsinghua Express, **(2002)**.