A Robust and Self-Tuning Page-Replacement Strategy for Spatial Database Systems

Thomas Brinkhoff

Institute for Applied Photogrammetry and Geoinformatics (IAPG)
Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven (University of Applied Sciences)
Ofener Str. 16/19, D-26121 Oldenburg, Germany
tbrinkhoff@acm.org

Abstract. For a spatial database management system, it is an important goal to minimize the I/O-cost of queries and other operations. Several page-replacement strategies have been proposed and compared for standard database systems. In the context of spatial database systems, however, the impact of buffing techniques has not been considered in detail, yet. In this paper, different page-replacement algorithms are compared for performing spatial queries. This study includes well-known techniques like LRU and LRU-K as well as new algorithms observing spatial optimization criteria. Experiments show that *spatial page-replacement algorithms* outperform LRU buffers for many distributions, but not for all investigated query sets. Therefore, a combination of spatial page-replacement strategies with LRU strategies is proposed and experimentally investigated. An algorithm is presented, which is self-tuning and adapts itself to different or changing query distributions. This *adaptable spatial buffer* outperforms LRU in respect to the I/O-cost by performance gains of up to 25%.

1 Introduction

Considering today's computers, it can be observed that the speed of the CPU and the size of the main memory are still dramatically increasing. In addition, spatial applications have become more sophisticated and the amount of spatial data as well as of non-spatial data demanded by them seems to grow with the size of available main memory. However, the time to access a randomly chosen page stored on a hard disk requires still about 10 ms [7]. As a result, the gap between CPU speed and size of main memory on the one hand and I/O-cost on the other hand has increased considerably with the consequence that the access to secondary storage is still a bottleneck for executing spatial queries and other operations. Several techniques are commonly used in order to optimize the I/O-performance of database systems. Among these methods, one technique is of special interest in context of this paper: the buffering of data.

A *buffer manager* caches a disk page, which has been read from secondary storage before. For a further request, such a page can be taken from main memory instead of reading it from disk. Essential for the performance gain of a buffer is to keep pages in main memory that are frequently requested. Pages no longer needed should be dropped out of the buffer as soon as possible. Therefore, the essential question is which page

published in: *Proceedings 8th International Conference on Extending Database Technology (EDBT 2002)*, Prague, Czech Republic, 2002;
Lecture Notes in Computer Science, Vol. 2287, Springer-Verlag, pp. 533-552.

should be removed from the buffer when a new page is read from secondary storage. The most common *replacement strategy* is *LRU* (least recently used): the LRU buffering policy replaces the page that has not been accessed for the longest time. However, the LRU policy has essential drawbacks: for example, it is not able to distinguish between pages that are accessed with a relatively high frequency over a longer period and pages which are requested infrequently over a long time but more frequently over a short period.

For standard database systems, many page-replacement algorithms have been proposed and investigated in literature (e.g. [3], [4], [9], [10]). An overview is given in [6]. In the area of spatial database systems, the effect of other page-replacement strategies than LRU has not been investigated, yet. Leutenegger and Lopez showed in their paper [8] the importance of a buffer for measuring the I/O-cost of R-trees [5]. Based on their buffer model, they especially demonstrated the effect of pinning top levels of an R-tree in the buffer.

In this paper, existing algorithms as well as new replacement strategies are presented and investigated in respect to their performance for different spatial queries and buffer sizes. First, the buffers considered in this paper are variants of the LRU buffer that use the type or the level of pages for improving the replacement. Besides, the *LRU-K page-replacement algorithm* by O'Neil, O'Neil, and Weikum [10] is considered. In addition, a new type of replacement algorithms – *spatial page-replacement algorithms* – is presented. These try to determine hot spots and candidates to be dropped out of the buffer according to the spatial properties of the pages. The experiments in this paper will show that such spatial page-replacement algorithms outperform LRU buffers for many distributions, but not for all investigated query sets. Therefore, a combination of spatial page-replacement strategies with LRU strategies is proposed. This algorithm divides the buffer into two parts managed by different page-replacement algorithms.

One essential objective is to minimize the effort of administration. Therefore, it is important that a buffer is self-tuning and adapts itself to the characteristics of the data stored in the database and to the type and sequence of the processed queries without human intervention. In case of the combination of spatial page-replacement strategies with LRU strategies, the division of the buffer into two parts must be automatically adapted to the current query profile. For this purpose, an algorithm will be proposed that changes the division according to the properties of the pages that are found in a special section of the buffer. This type of buffer will be called *adapting spatial buffer*.

Section 2 presents the basic page-replacement strategies investigated in this paper. Especially, spatial page-replacement algorithms are proposed. A performance evaluation follows in the third section. After presenting the data and query sets of the experiments, the results of the tests using different categories of page-replacement algorithms are presented. Section 4 presents a self-tuning and robust combination of spatial page-replacement strategies with LRU strategies. The experiments show that this combination improves the performance of spatial queries by up to 25% compared to LRU without the drawbacks of pure spatial page-replacement algorithms and without increasing the memory requirements. The paper concludes with a summary of the most important findings and an outlook to future work.

2 Page-Replacement Algorithms

Since the LRU policy has essential drawbacks, different page-replacement algorithms suitable for spatial database systems are presented in this section. The presentation starts with a type-based and a priority-based LRU page-replacement policy. Then, the LRU-K page-replacement algorithm is presented. Finally, spatial page-replacement algorithms are proposed.

2.1 Type-based and Priority-based LRU

For an LRU buffer, all (non-fixed) pages have the same type. Therefore, the only selection criterion is the time of the last access. If the buffer knows the type of a page, this information may be used for deciding if it is a suitable candidate to be dropped out of the buffer. For pages stored in a spatial database system, we typically distinguish three categories of pages: directory pages and data pages of the *spatial access method (SAM)* as well as object pages storing the exact representation of spatial objects (see e.g. [2]). Using a *type-based LRU (LRU-T)*, object pages would be dropped immediately from the buffer. Then, data pages would follow. Directory pages would be stored in the buffer as long as possible because it is assumed that they are more often required than data or object pages. For pages of the same category, the LRU strategy is used for determining the page to be removed.

A generalization of this approach is a *priority-based LRU strategy (LRU-P)*. In this case, each page has a priority: the higher the priority of a page, the longer it should stay in the buffer. For example, the object page may have the priority 0 whereas the priority of a page in an index depends on its height in the corresponding tree, if a tree-based spatial access method is used. The root has the highest priority. Such an approach is a generalization of a buffer that pins distinct levels of the SAM [8]. Figure 1 illustrates LRU-T and LRU-P.

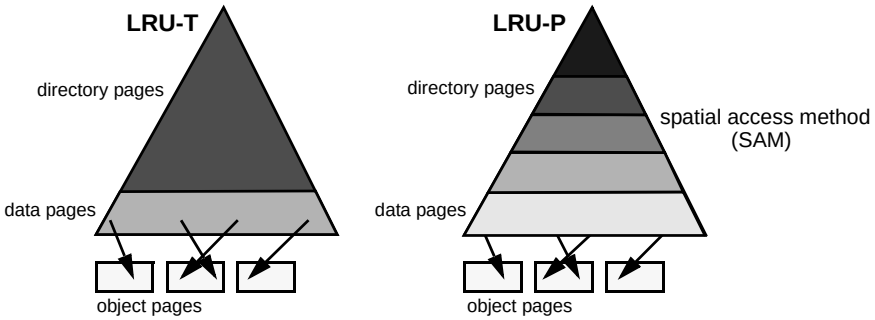

Fig. 1: Categories of pages; the darker a page, the higher its priority.

2.2 The LRU-K Page-Replacement Algorithm

In order to improve the replacement strategy, several solutions have been suggested in the literature. A typical representative of a replacement algorithm determining the next page to be replaced only by recording and analyzing the access history is the *LRU-K page-replacement algorithm* by O'Neil, O'Neil, and Weikum [10].

For recording the history, the LRU-K algorithm requires an additional data structure called *HIST*[1]: *HIST(p)* contains the time stamps of the *K* most recent references to a page *p*; *HIST(p,1)* denotes the time stamp of the last reference, *HIST(p,2)* the time stamp of the last reference before *HIST(p,1)*, and so on. The authors observed that it is typical that a page is often accessed immediately after it has been accessed before. Therefore, *HIST(p)* does contain only the last access of a sequence of correlated accesses. In order to decide whether accesses are correlated or not, an additional data structure may be necessary. In the following, two page accesses will be regarded as correlated if they belong to the same query.

A query typically requests several pages from the buffer. If a page *p* is requested, three cases may occur:

- Page *p* is in the buffer and the current access and the most recent reference to *p* are correlated: *HIST(p,1)* gets the value of the current time.
- Page *p* is in the buffer and the current access and the most recent reference to *p* are not correlated: The value of the current time is added to *HIST(p)* as new *HIST(p,1)*.
- Page *p* is not in the buffer: Among the pages in the buffer whose most recent reference to *p* is not correlated to the access to *p*, the page *q* with the oldest value of *HIST(q,k)* is determined. The page *q* is removed from the buffer and the value of the current time is added to *HIST(p)* as new *HIST(p,1)*[2].

The usage of *HIST(q,k)* allows determining which pages are frequently requested over a short period but infrequently used over a long time. For maintaining *HIST*, an LRU-K buffer typically requires more memory than a standard LRU buffer. For the case that a page that was dropped out of the buffer before is reloaded into the buffer, the information of *HIST* collected previously for this page should be available, i.e. the history information *HIST* must also be stored (in main memory) for pages which have left the buffer. This is an essential disadvantage. Therefore, the memory requirements of LRU-K buffers are not only determined by the number of pages in the buffer but also by the total number of requested pages. The space requirements of the history information of one page are quite low, but the number of such records can be become rather high. This number corresponds to the number of pages stored in the buffer during its lifetime.

The investigations in the original paper demonstrated for a standard database system that the LRU-K page-replacement strategy is worthwhile for rather small values of K.

2.3 Spatial Page-Replacement Algorithms

The LRU-K page replacement does not have any knowledge about the type or content of the stored pages. The priority-based LRU buffer uses the type of a page or some other structural properties of a page for performing a (pre-) selection. However, in a spatial database it may be reasonable to analyze also the content of pages for selecting the page that should be removed from the buffer. Spatial access methods optimize their structure according to different optimization criteria. Beckmann et al. proposed four criteria for the design of an R*-tree [1]:

---

[1] The definition given here slightly differs from the definition in the original paper. The functionality, however, is the same.

[2] Some special cases may occur in this situation that are not discussed here.

(O1) Minimizing the area of a directory rectangle.
(O2) Minimizing the overlap between directory rectangles.
(O3) Minimizing the margin of a directory rectangle.
(O4) Maximizing the storage utilization.

The criteria (O1) to (O3) are spatial criteria, which can be applied for designing a *spatial page-replacement strategy.*

It will be assumed in the following that a page p in a spatial database system contains a set of entries $e_i \in p$. For each of these entries, a *minimum bounding rectangle (MBR)* can be determined. For an object page, the entries may correspond to the spatial objects (or parts of them) stored in the page. For a data or directory page of an R-tree [5], these entries are the rectangles stored in the pages. In a quadtree [13], the quadtree cells match these entries. The same holds for z-values stored in a B-tree [11].

Maximizing the area of a page (A)

The optimization goal (O1) was to minimize the area of a (directory) page. The larger its area, the more frequently the page is assumed to be requested. That means that a page having a large area should stay in the buffer as long as possible. In other words, the (non-fixed) page with the smallest area is the first page to be dropped out of the buffer. The area of a page p is defined as the area of the MBR containing all entries of p.

Formally, a spatial page-replacement algorithm requires a function *spatialCrit(p)* computing the corresponding spatial criterion of a page p. In case of the variant A, this function is defined by the area of the MBR containing all entries of the page:

$$\text{spatialCrit}_A(p) = \text{area}(\text{mbr}(\{e \mid (e \in p)\}))$$

If a page d_p must be dropped out of the buffer, this page is determined as follows:

1. $C := \{ p \mid p \in \text{buffer} \wedge (q \in \text{buffer} \Rightarrow \mathit{SpatialCrit(p)} \leq \mathit{SpatialCrit(q)}) \}$
2. if $|C| > 1$ then: $d_p \in C$ is determined by using the LRU strategy
   else: d_p is the only element of C

Maximizing the area of the entries of a page (EA)

The strategy EA is also based on the optimization goal (O1). Instead of the area of a page, the sum of the areas of its entries is maximized. This sum is not normalized to the number of entries. That means that also optimization goal (O4) is considered by this algorithm. For directory pages of SAMs partitioning the data space completely and without overlap, the algorithms A and EA behave identically. The function $\mathit{spatialCrit}_{EA}(p)$ is defined as follows:

$$\text{spatialCrit}_{EA}(p) = \sum_{e \in p} \text{area}(\text{mbr}(e))$$

Maximizing the margin of a page (M)

According to optimization goal (O3), the margin of a page is maximized. The larger its margin, the longer a page will stay in the buffer. The margin of a page p will be defined as the margin of the MBR containing all entries of p. The function $\mathit{spatialCrit}_M(p)$ is defined as follows:

$$\text{spatialCrit}_M(p) = \text{margin}(\text{mbr}(\{e \mid (e \in p)\}))$$

Maximizing the margin of the entries of a page (EM)
This strategy is similar to the previous algorithm. Instead of the margin of the page, the sum of the margins of its entries is maximized. As for EA, this sum is not normalized to the number of entries. The function *spatialCrit$_{EM}$(p)* is defined as follows:

$$\text{spatialCrit}_{\text{EM}}(p) = \sum_{e \in p} \text{margin}(\text{mbr}(e))$$

Maximizing the overlap between the entries of a page (EO)
The fifth spatial page-replacement algorithm tries to maximize the overlap between the entries of a page. This overlap is the sum of the intersection areas of all pairs of entries whose MBRs overlap. For directory pages of spatial access methods, which partition the data space without overlap, this algorithm should not be applied. The function *spatialCrit$_{EO}$(p)* is defined as follows:

$$\text{spatialCrit}_{\text{EO}}(p) = \sum_{e \in p,\, f \in p,\, e \neq f} \frac{\text{area}(\text{mbr}(e) \cap \text{mbr}(f))}{2}$$

The area and the margin of a page or of its entries can be computed causing only a small overhead when a new page is loaded into the buffer. The computation of the overlap of the entries of a page is costlier – storing this information on the page may be worthwhile for this variant of a spatial page-replacement algorithm.

Figure 2 depicts the criteria of the five spatial page-replacement algorithms. The hatched areas and the bold boundaries illustrate the criterion of the respective strategy.

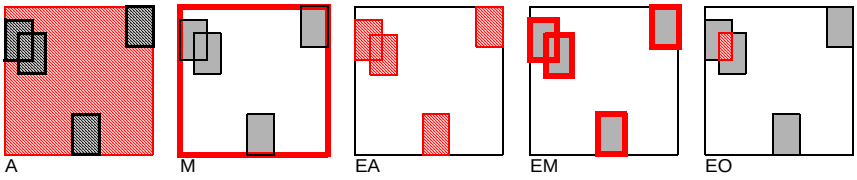

Fig. 2: Illustration of the criteria of the different spatial page-replacement strategies.

3 Performance Evaluation

The primary spatial database used in the following tests consists of geographical features of the mainland of the USA. These data originate from the USGS Mapping Information System (GNIS) (`http://mapping.usgs.gov/www/gnis/`) like the data sets of the SEQUOIA 2000 storage benchmark [14]. The data set consists of 1,641,079 points and extended spatial objects. The objects are managed by an R*-tree [1]. The maximum number of entries per directory page and per data page is 51 and 42, respectively. The R*-tree consists of 58,405 pages (thereof 1,660 directory pages = 2.84%) and has height 4. The pages of the spatial objects are stored in separate files and buffers. Only the pages accesses concerning the trees are reported in the following. A second database is used for validating the results of the experiments using database 1. It consists of the line and area features of a world atlas [12]. The data set comprises 244,472 polygons and 328,222 lines. The R*-tree managing these objects consists of 20,884 data pages and 617 directory pages (2.87%). Figure 3 illustrates the databases.

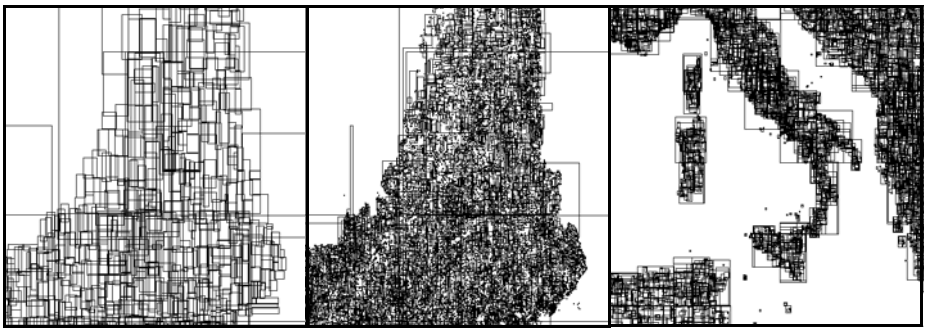

Fig. 3: The MBRs of a part of database 1 (area of Maine), the respective directory regions of the lowest level of the R*-tree, and the MBRs of a part of database 2 (area of Italy).

The size of the buffers in the following experiments are determined in respect to the size of the data sets ranging from 0.3% to 4.7%. Because of using relative buffer sizes, the results of the experiments should hold for the case of larger databases and buffers. For assessing the size of a buffer, the reader must also consider that the main memory of a database server is not only used for buffering pages of a single spatial data set.

Before performing a new set of queries, the buffer was cleared in order to increase the comparability of the results. For the same reason, only the relative performance will be reported in this paper. The tables containing the absolute number of disk accesses and other figures can be downloaded from `http://www.geodbs.de/buffer.html`. The reader will find there also the data sets and the query sets used by the following experiments.

3.1 The Query Sets

A further ingredient of an experiment is the set of queries. For a systematic investigation of a page-replacement strategy, query sets obeying different data distributions are required. Unfortunately, no benchmark for spatial databases offers a suitable set of queries. The following distributions try to cover a wide range of spatial query profiles.

Uniform distribution
The query sets *U-P* and *U-W-ex* consist of uniformly distributed query points and windows for performing point queries and window queries, respectively. *ex* denotes the reciprocal value of the extension of the query windows in one dimension; in the set *U-W-33*, the x-extension of a query window is 1/33 of the x-extension of the data space. Other values for *ex* in the experiments are 100, 333, and 1000. The query objects cover also the parts of the data space where no objects are stored.

Identical distribution
The query sets *ID-P* and *ID-W* consist of a random selection of objects stored in the database. For the window queries, the size of the objects is maintained. The distribution of the queries is the same as the distribution of the objects in the database.

Similar distribution
In the case of similar distributions, there exists a dependency between the distribution of the query objects and the database objects. This typically happens when two layers of a map are combined with some functional dependencies between them. The query sets *S-P* and *S-W-ex* were computed by randomly selecting US cities and towns from a file containing all US places. This file originates also from the USGS Mapping Information System.

Intensified distribution
For the intensified distribution, the probability of selecting a city from the same file that was already used for the similar distribution is correlated to the square root of the population of the city. These query sets are denoted by *INT-P* and *INT-W-ex*.

Independent distribution
In this case, the distributions of the database objects and of the query objects are independent from each other. These query sets were constructed using like the sets *S-P* and *S-W-ex* after flipping the x-coordinates of the spatial objects, i.e. an object lying in the west of the map retrieves an area in the east of the database and vice versa. These query sets are denoted by IND-P and IND-W-ex.

The number of queries per query set was set a value, so that the number of disk accesses was about 10 to 20 times higher than the buffer size in the case of the largest buffer investigated. For smaller buffers, this factor increases.

3.2 Type-based and Priority-based LRU

A performance comparison between the type-based and the priority-based LRU buffer (LRU-T and LRU-P) has shown no differences between both approaches in the case of larger buffers. Then, all or most of the directory pages are kept in main memory. Using small buffer sizes, LRU-P has beaten LRU-T for all investigated query sets.

The diagrams in Figure 4 show the performance gains using LRU-P. The results for the primary database are shown in the upper diagrams and for the second database in the lower diagrams. The left diagrams depict the results for performing uniformly distributed queries and the right diagrams of queries according to the intensified distribution. The performance gain is given in percent compared to using a standard LRU buffer (defined by |disk accesses of LRU| / |disk accesses of LRU-P| - 1).

The performance impact depends on the buffer size as well as on the size of the query regions. The largest performance gains are achieved for small buffers performing window queries of medium (and sometimes of small) size. In the case of database 1 using large buffers and small query windows (or point queries), the performance has not been improved. Instead, it has been worse compared to using a standard LRU buffer. An explanation of the good performance for small buffers is that it is important to store the upper levels of the R*-tree in the buffer. Is observation corresponds to the results of the work of Leutenegger and Lopez [8]. However, if the buffer becomes larger it will be possible to store the lower levels of the index. Then, the height of a page in the tree becomes less important for indicating its residence time in the buffer.

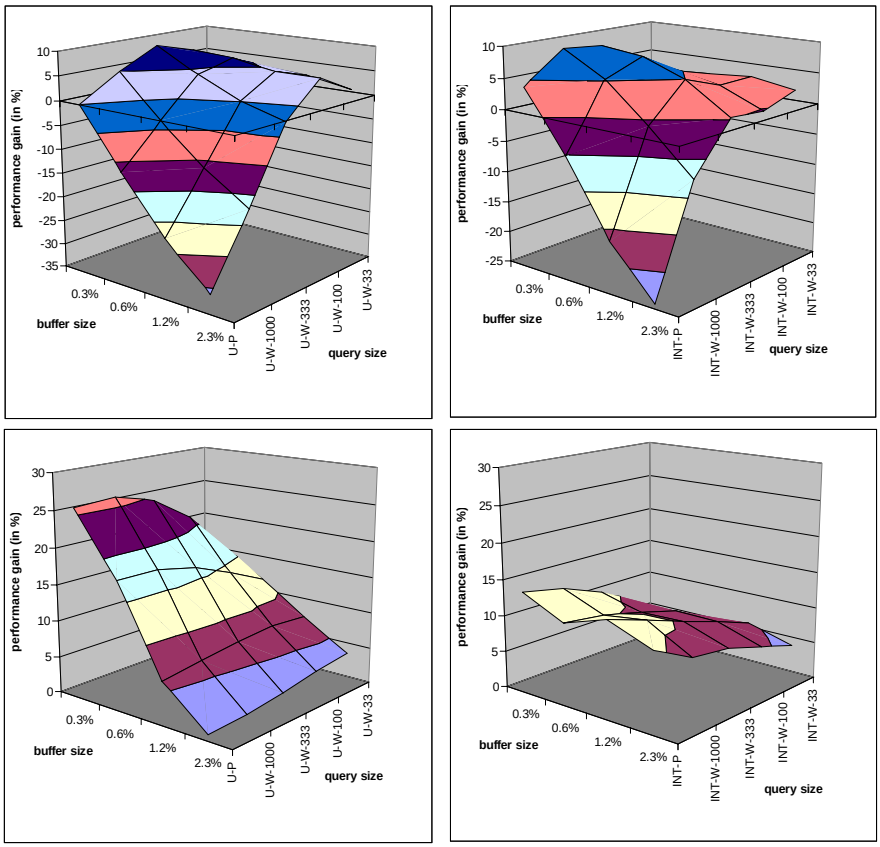

Fig. 4: Performance gain of LRU-P compared to LRU using database 1 (upper diagrams) and database 2 (lower diagrams).

3.3 LRU-K Buffers

In this subsection, the impact of using LRU-K buffers on the query performance will be investigated. In [10], the usage of LRU-2 showed a significant improvement compared to a standard LRU buffer whereas using LRU-3 had almost no impact compared to LRU-2. Figure 5 depicts the performance gains of using LRU-2, LRU-3, and LRU-5 buffers compared to an LRU buffer.

Especially for point queries and small and medium window queries, considerable performance gains are attained. Performance gains of 15% to 25% are achieved by using LRU-2. However, for large window queries, almost no improvement can be observed. In one case (*U-W-33*; 4.7% buffer), the performance is actually decreased. Surprisingly, the performance differences between the different types of query sets are not significant. The LRU-K buffer is not especially suitable – for example – for processing the intensified query sets. Like in [10], no significant difference between the performance of LRU-2, of LRU-3, and of LRU-5 can be observed. Therefore, LRU-2 is used as the representative for the comparison in Section 3.5.

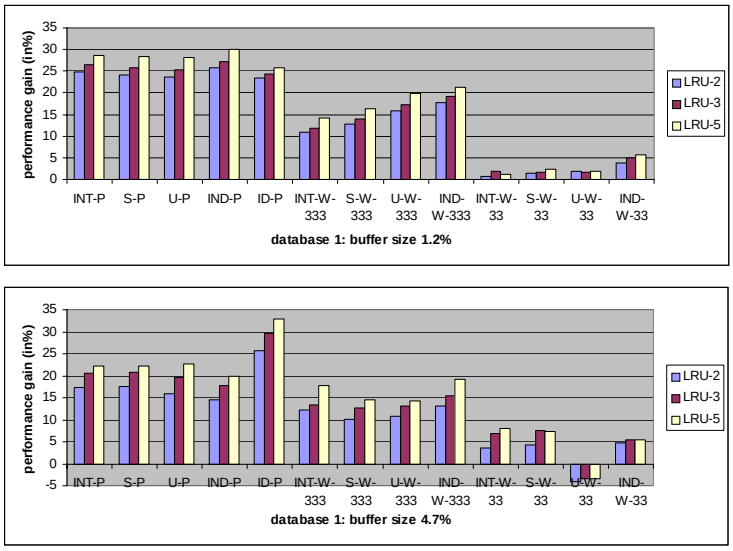

Fig. 5: Performance gain using LRU-K compared to LRU for the primary database.

3.4 Spatial Page-Replacement Algorithms

Now, the best spatial page-replacement strategy will be determined. Figure 6 depicts two diagrams representing the relative number of disk accesses. For each query set, the number of disk accesses required by using replacement criterion A is taken as 100%. For the other strategies, the number of accesses in comparison to this base is depicted in percent.

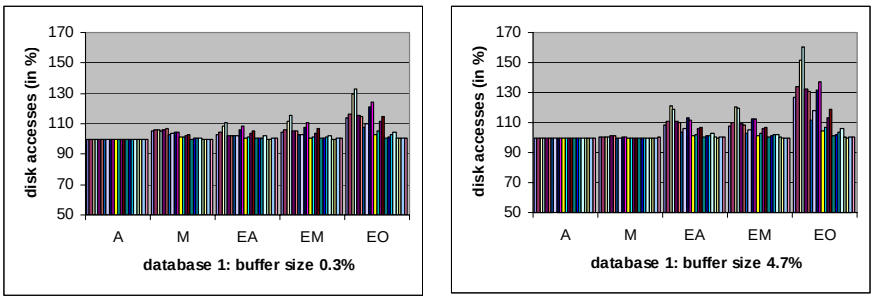

Fig. 6: Comparison of the performance of the different spatial replacement algorithms.

For the test series using a 0.3% buffer, the criterion page area (A) shows the best performance whereas the algorithm that uses the overlap between the entries (EO) has the worst performance. In the experiments using the larger buffer (4.7%), the algorithms A and M have about the same performance and EA, EM and EO lose the competition more clearly.

Taking these results and the fact that the area of a page can be determined without noticeable overhead, the algorithm maximizing the area of a page (A) is taken in the following comparisons as the representative of the spatial page-replacement strategy.

3.5 Comparison of the Different Types of Page-Replacement Algorithms

In this section, the different types of page-replacement algorithms are investigated. The algorithms LRU-P, A, and LRU-2 are compared to the standard LRU strategy. The results for both databases are presented.

3.5.1 Uniform Distribution

Figure 7 depicts the relative performance gains of the three algorithms using a uniform distribution of queries and a buffer size of 0.6% and of 4.7%. The left diagrams show the results for the primary database and the right diagrams for database 2.

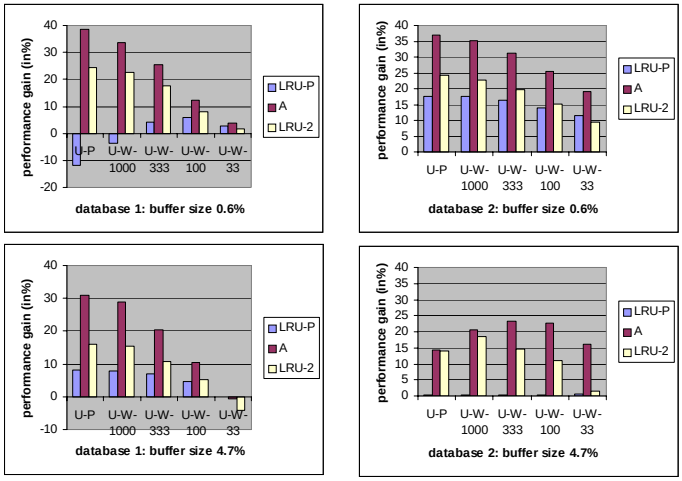

Fig. 7: Performance gain of the algorithms for the uniform distribution.

The LRU-P strategy is clearly the worst competitor. The clear winner of these tests is the buffer using the spatial page-replacement strategy. Especially in this case, subtrees of the SAM are often requested that have a large spatial extension. Then, the spatial page-replacement strategy using the area criterion profits most.

3.5.2 Identical and Similar Distributions

The results of the identical and similar distributions are depicted in Figure 8. The left diagrams show the results for database 1 and the right diagrams for the second database. In the most cases, the spatial page-replacement strategy A has the same or a better performance than the LRU-2 buffer. Performance gains of 30% are achieved. However, we can also observe that in some cases – see the lower right diagram – the performance gains collapse. Instead, a loss of performance can occur performing large window queries. An explanation for this behavior corresponds to the explanation given for the intensified distribution in the next subsection.

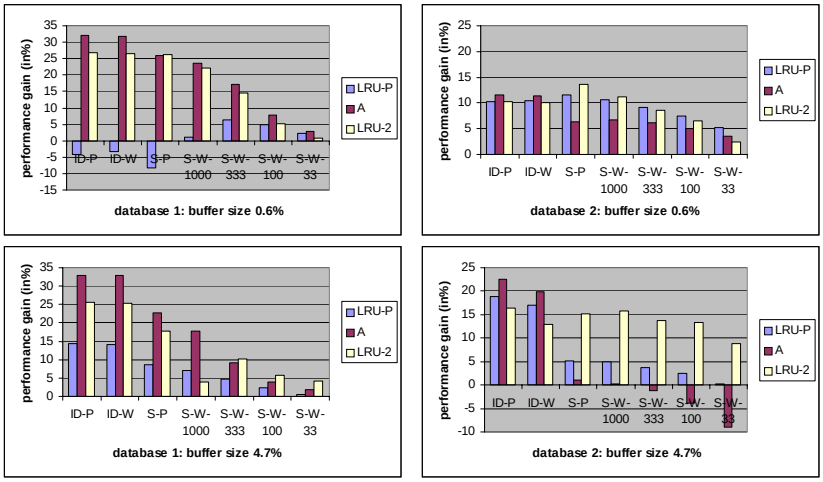

Fig. 8: Performance gain of the algorithms for the identical and similar distribution.

3.5.3 Independent and Intensified Distributions

The results for the independent and intensified distributions are presented in Figure 9. For test series using the independent query sets *(IND-x)*, there are significant discrepancies between the results of the spatial page-replacement strategy for database 1 and for database 2. For the primary database, this strategy achieves considerable performance gains. For the second database, however, we observe a significant increase of the I/O-cost. The reason is that this query set was constructed by flipping the x-coordinates. In the case of database 1 – the US mainland map – most query points meet again the mainland. Using the world map (i.e. database 2), this observation does not hold. Most query points meet water and can be performed only using the root page of the R*-tree.

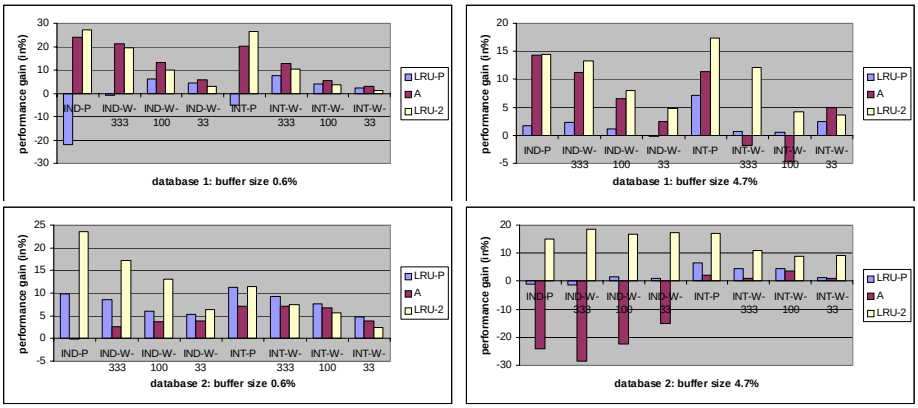

Fig. 9: Performance gain of the algorithms for the independent and the intensified distribution.

For both databases, the performance of the spatial page-replacement strategy is also inferior in the case of query sets obeying the intensified distribution. The reason is that

areas of intensified interest are not characterized by large page areas; typically, the opposite case occurs, i.e., in such regions there are often more objects stored than outside of such region. As consequence, the page area (or other spatial properties of the page or of its entries) tends to be smaller and not larger than the average.

In the other cases, the performance of the spatial page-replacement strategy is comparable to the I/O-cost of the LRU-2 buffer, which is the winner for the independent and intensified distribution.

4 Combining LRU-based & Spatial Page-Replacement Algorithms

The results of the last experiments have shown that it is not advisable to use pure spatial page-replacement algorithms in the case of some of the investigated query sets. In order to solve this drawback, an adaptable and robust combination of spatial page-replacement strategies with LRU strategies will be proposed and experimentally investigated in this section.

4.1 The Combination

The basic idea for combining LRU-based and spatial page-replacement algorithms is 1.) to compute a set of candidates by using LRU (or another page-replacement algorithm) and 2.) to select the page to be dropped out of the buffer from the candidate set by using a spatial page-replacement algorithm. The larger the candidate set, the larger is the influence of the spatial page-replacement algorithm, and the smaller the candidate set, the larger is the influence of the LRU algorithm. Figure 10 illustrates this effect.

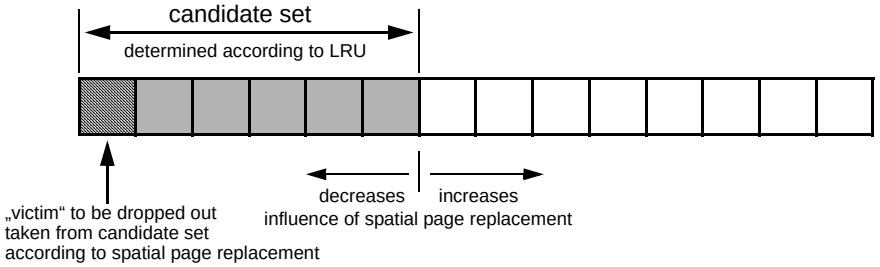

Fig. 10: Combination of LRU-based and spatial page-replacement algorithms.

4.2 A Self-tuning Algorithm for Computing the Size of the Candidate Set

An important question concerns the size of the candidate set. For the uniform query distribution, its size should be as large as possible. However, having an intensified distribution, a very small candidate set may be preferable. Therefore, the size of the candidate set should be adapted to the query distribution. For this task, we need a suitable algorithm. According to the introduction in Section 1, this algorithm should be self-tuning in order to minimize the efforts for an efficient database management. The size of the candidate set should be determined without the intervention of the database administrator.

The basic idea of the algorithm is to reserve a part of the buffer for storing pages that have already dropped out of the buffer according to the approach described in Section 4.1. In this part of the buffer, which is called *overflow buffer*, a victim is determined using the first-in first-out technique. If a requested page *p* is found in the overflow buffer, *p* is moved to the standard part of the buffer. In this situation, we can distinguish three different cases:

1. The number of pages in the overflow buffer having a better spatial criterion than *p* is higher than the number of pages having a better LRU criterion than *p*. Then, the LRU strategy seems to be more suitable than the spatial page-replacement strategy: the size of the candidate set will be decreased.
2. The number of pages in the overflow buffer having a better spatial criterion than *p* is smaller than the number of pages having a better LRU criterion than *p*. Then, the spatial page-replacement strategy seems to be more suitable than the LRU strategy: the size of the candidate set will be increased.
3. Both numbers are the same: Then, the size of the candidate set will not be changed.

An illustration of this approach can be found in Figure 11. This type of buffer is called ASB in the following, abbreviating *adaptable spatial buffer*.

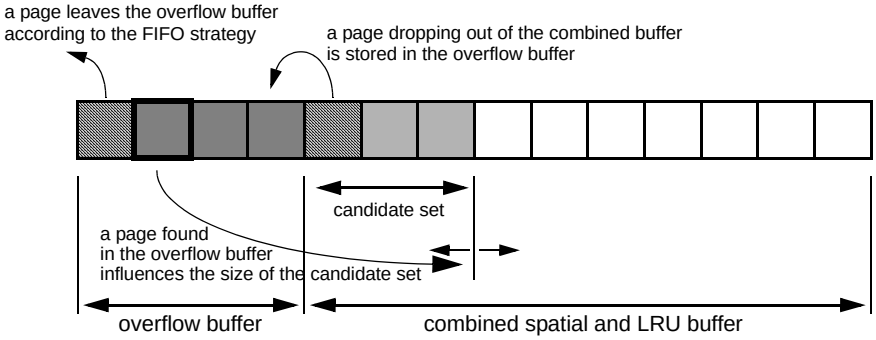

Fig. 11: Illustration of the adapting spatial buffer.

4.3 Performance Evaluation

In the following, the results of a performance comparison are presented. Figure 12 compares the performance gains of the spatial page-replacement strategy A with the combined approach using a candidate set of static size, i.e. using no overflow buffer. The size of the candidate sets was set to 50% (denoted in the legend of the diagram as SLRU 50%) and to 25% of the buffer (denoted as SLRU 25%).

As expected, we can observe that the performance of the combination of the LRU-based and spatial page-replacement algorithm shifts the behavior of the spatial page-replacement strategy into the direction of the LRU strategy. Consequently, the I/O-cost for query sets performed by algorithm A with large performance gains increases. In the other case, in which A has shown a loss of performance, the I/O-cost is decreased. In the most cases, the performance loss has become a (slight) performance gain. These observations especially hold for the case, in which the candidate set had a size of 25% of the buffer (SLRU 25%).

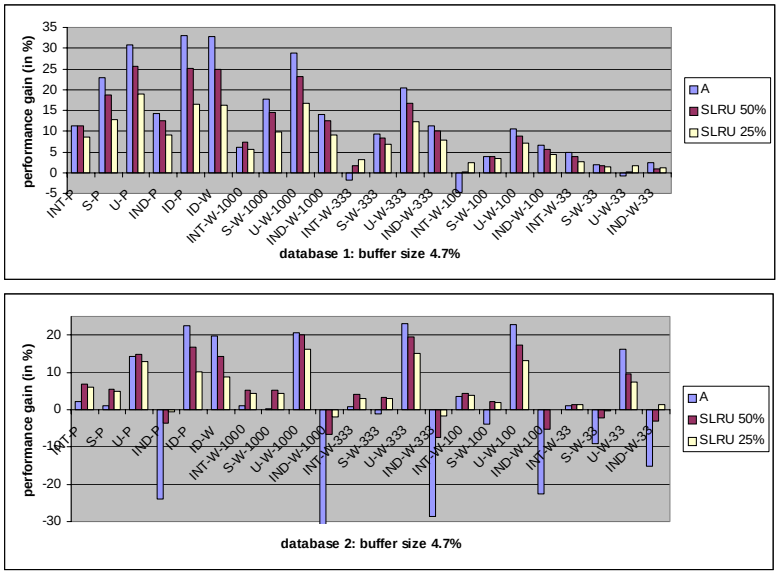

Fig. 12: Performance gains using a candidate set of static size.

In the following experiments, an overflow buffer is used for adapting the size of the candidate set. The size of the overflow buffer has been 20% of the complete buffer. The initial size of the candidate set has been 25% of the remaining buffer (i.e. 20% of the complete buffer). The size of the candidate set has been changed in steps of 1% of the remaining buffer. The memory requirements are not increased compared to the other page-replacement strategies.

Figure 13 depicts the performance gains of the spatial page-replacement strategy A, of the combination of LRU and algorithm A using a candidate set having a static size of 25% (denoted in the legend of the diagram as SLRU), of the adapting spatial buffer (denoted as ASB), and of LRU-2 compared to using only LRU.

Like in the experiments before, the behavior of the adapting spatial buffer is a mixture of the performance of an LRU buffer and the spatial page-replacement strategy. In contrast to SLRU, the behavior of ASB is more similar to algorithm A in the cases, in which A shows a very good performance, and more dissimilar to A in the cases, in which A shows a loss of performance. In all cases, a performance gain can be achieved by using ASB.

Comparing the behavior of ASB with LRU-2, we can observe a similar I/O-cost in many cases. However, there are still several cases, in which LRU-2 outperforms ASB. Nevertheless, the reader should be aware about the fact that an LRU-2 buffer requires storing the history information of pages that have already left the buffer. Because this space requirement cannot be predicted (in real life), it is not considered in the experiments. The adapting spatial buffer does not require storing information about pages that have dropped out of the buffer.

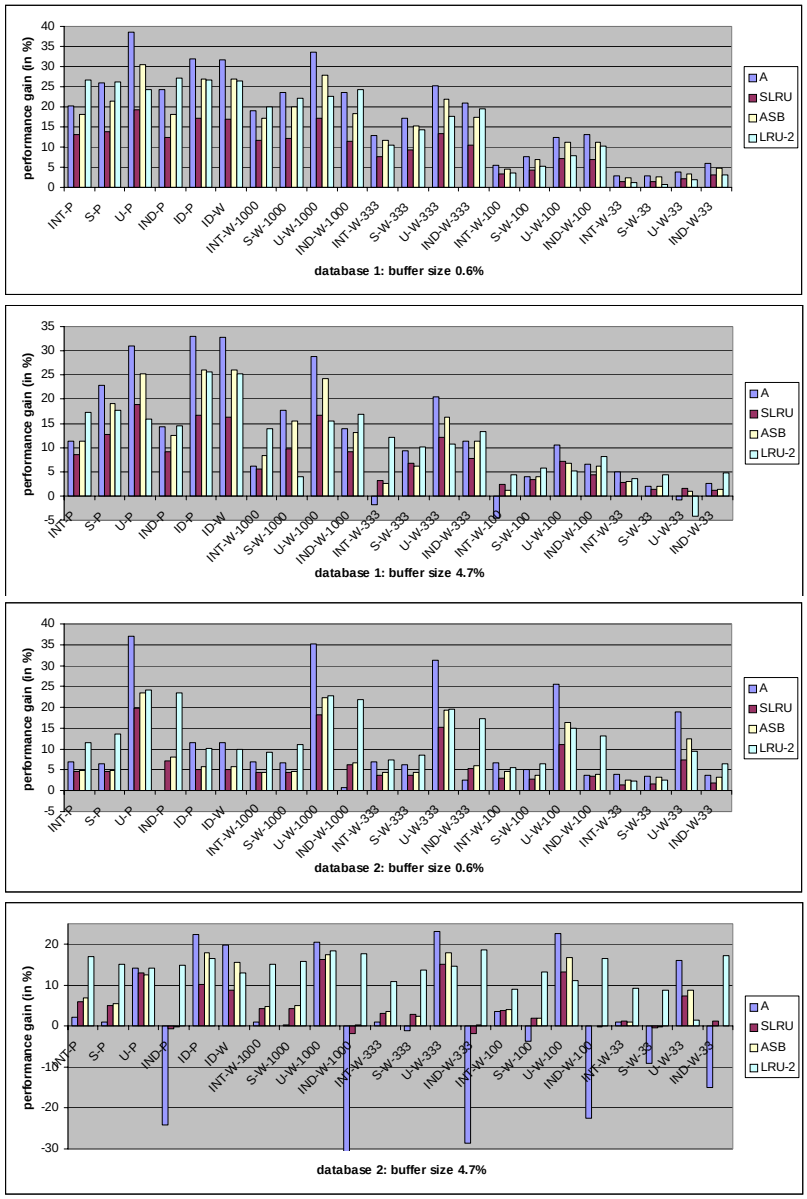

Fig. 13: Performance gains of A, SLRU, ASB, and LRU-2 compared to LRU.

For the last experiment, three query sets are concatenated: *INT-W-33*, *U-W-33*, and *S-W-33*. Figure 14 shows the size of the candidate set during this test using the adapting spatial buffer with the same parameter settings as in the experiments before.

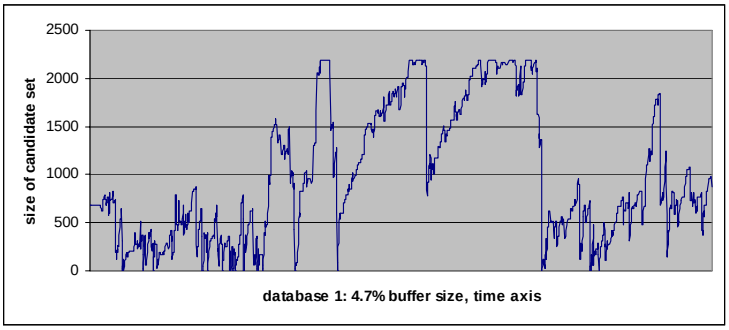

Fig. 14: Size of the candidate set using ASB for a mixed query set.

In the first phase, the queries are distributed according to the intensified distribution. Starting with a size of 684, the size drops to an average value of about 300. During this phase, the influence of the LRU strategy is predominating. Then, the queries are distributed according the uniform distribution. Consequently, the size of the candidate set increases to an average value of about 1620. During this second phase, the buffer primarily applies the spatial page-replacement strategy. During the last phase, the queries obey the similar distribution. Thus, the average size of the candidate set drops to an average of about 650. The influence of the LRU strategy and of the spatial page-replacement strategy is approximately balanced.

5 Conclusions

In order to optimize the I/O-performance of a database system, the buffering of pages in main memory is one of the most important techniques. However, in the context of spatial database systems, this topic has not been considered in detail. The impact of different page-replacement algorithms on the performance of spatial queries is one of the open questions. Therefore, existing page-replacement strategies have been investigated in this paper in respect to their impact on the performance of spatial queries. Furthermore, new replacement algorithms – so-called *spatial page-replacement algorithms* – have been presented. These algorithms try to determine pages to be dropped out of the buffer according to spatial properties like the area, the margin and the overlap of pages and page entries.

The tests have shown that spatial page-replacement algorithms have a very good performance for many query distributions. However, they are not robust. For some distributions, their performance is worse than the performance of a standard LRU buffer. Therefore, a combination of spatial page-replacement algorithms with other buffer policies like LRU has been proposed. The basic idea of this combination is 1.) to compute a set of candidates by using LRU (or another policy) and 2.) to select the page to be dropped out of the buffer from the candidate set by using a spatial page-replacement algorithm. For a robust and self-tuning algorithm, it is necessary to adapt the size of the candidate set to the requested query profiles. An algorithm for dynamically

adjusting the size of the candidate set has been presented. This adapting spatial buffer achieves considerable performance gains. In the cases in which the pure spatial page-replacement algorithms clearly outperforms other policies, the combination achieves performance gains of up to 25% compared to LRU. In contrast to the pure spatial page-replacement, the I/O cost increases for none of the investigated query distributions.

Future work consists of the following tasks: 1. to extend the experiments by investigating the influence of the size of the overflow buffer and by distinguishing random and sequential I/O, 2. to study the influence of the strategies on updates and spatial joins, and 3. to investigate the impact of (spatial) page-replacement algorithms on the management of moving spatial objects in spatiotemporal database systems.

6 References

1. Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B.: The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. In: Proceedings ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, 1990, 322-331.
2. Brinkhoff, T., Horn, H., Kriegel, H.-P., Schneider, R.: A Storage and Access Architecture for Efficient Query Processing in Spatial Database Systems. In: Proceedings 3rd International Symposium on Large Spatial Databases, Singapore, 1993. Lecture Notes in Computer Science, Vol. 692, Springer, 357-376.
3. Chen, C.M., Roussopoulos, N.: Adaptive Database Buffer Allocation Using Query Feedback. In: Proceedings 19th International Conference on Very Large Data Bases, Dublin, Ireland, 1993, 342-353.
4. Chou, H.T., DeWitt, D.J.: An Evaluation of Buffer Management Strategies for Relational Database Systems. In: Proceedings 11th International Conference on Very Large Data Bases, 1985, Stockholm, Sweden, 127-141.
5. Guttman, A.: R-trees: A Dynamic Index Structure for Spatial Searching. In: Proceedings ACM SIGMOD International Conference on Management of Data, Boston, 1984, 47-57.
6. Härder, T., Rahm, E.: Datenbanksysteme: Konzepte und Techniken der Implementierung. Springer, 1999.
7. IBM: Hard Disk Drives. `http://www.storage.ibm.com/hardsoft/diskdrdl.htm`
8. Leutenegger, S.T., Lopez, M.A.: The Effect of Buffering on the Performance of R-Trees. In: Proceedings of the 14th International Conf. on Data Engineering, Orlando, 1998, 164-171.
9. Ng, R.T., Faloutsos, C., Sellis, T.K.: Flexible Buffer Allocation Based on Marginal Gains. In: Proceedings ACM SIGMOD International Conference on Management of Data, Denver, CO, 1991, 387-396,
10. O'Neil, E.J., O'Neil, P.E., Weikum, G.: The LRU-K Page Replacement Algorithm for Data Disk Buffering. In: Proceedings ACM SIGMOD International Conference on Management of Data, Washington, DC, 1993, 297-306.
11. Orenstein, J.A., Manola, F.A.: PROBE Spatial Data Modelling and Query Processing in an Image Database Application. IEEE Transactions on Software Engineering, Vol.14, No.5, 1988, 611-629.
12. Rossipaul Medien GmbH: Der große Weltatlas - Unsere Erde multimedial. CD-ROM edition, 1996.
13. Samet, H.: Applications of Spatial Data Structures. Addison-Wesley, 1990.
14. Stonebraker, M., Frew, J., Gardels, K., Meredith, J.: The SEQUOIA 2000 Storage Benchmark. In: Proceedings ACM SIGMOD International Conference on Management of Data, Washington, DC, 1993, 2-11.