6. Performance Models for Tree-Based Index Structures

Make it as simple as possible, but not simpler. Albert Einstein

In the previous chapter we showed that aggregated data in inner nodes increases the performance of tree-based index structures. Results in the previous chapter are based on experiments. We would now like to predict analytically the performance for the index structure with or without the extension. In order to design analytical models, three in literature well known performance models for index structures without aggregated data are used. We generalize the models such that they are applicable to model index structures with aggregated data [Jürgens and Lenz, 1999a].

We then introduce a fourth model that considers the distribution of data and is able to model index structures with and without aggregated data. Experiments evaluate which model is suited best for different data sets. Additionally, we apply the models to show for what kinds of data and for what kinds of queries the extension increases the performance the most.

6.1. Introduction

We take three existing models from the literature for modeling the performance of tree structures and expand these models to model index structures with aggregated data. The presented performance models work for trees, which organize the data space in multidimensional rectangles. Members of the R-tree family and many more index structures fulfill this condition. The estimated number of blocks intersecting a given query box corresponds to the number of disk accesses needed to perform the query.

6.2. Fit for modeling

The concept of adding aggregated data to the inner nodes of an index structure specified in Definition 5.2 on page 48 can be applied to most tree-based index structures. In order to apply the performance models that are presented in this chapter, we refine the previous Definition 5.2.

Definition 6.1 All tree-based index structures that fulfill the *fit for aggregation definition* (Definition 5.2 on page 48) and that store regions as hyper-rectangles parallel to the axis are called *fit for modeling*. Each $region$ in $(region, agg, ptr)$ is a d-dimensional hyper-rectangle as described in Definition 3.2 on page 18.

A large number of index structures fulfill this additional condition. Especially the members of the R-tree family accomplish this. Structures like the kd-tree or kdb-tree are also *fit for modeling*. Structures like the cell-tree, the SS-tree, SR-tree, or the UB-tree do not base their internal structure on rectangles but on other type of regions. Therefore, these kinds of index structures are not supported by the following performance models.

6.3. Performance models for access leaf nodes

This section presents performance models for index structures with and without aggregated data. All models consider only accesses to *leaf* nodes. The main goal of the extended structure is to minimize the number of disk accesses with minimal enlargement of the structure. Hellerstein et al. investigates the tradeoff between redundant data and access overhead in a similar context [Hellerstein et al., 1997b]. Here, we assume that all *non-leaf* nodes are stored in main memory and all *leaf* nodes are read from secondary memory. We keep the first few levels of the tree in main memory. According to Leutenegger et al. *pinning first levels in main memory gets the greatest improvements for point queries and just small improvements for range queries* [Leutenegger and Lopez, 1998]. But *pinning the upper levels of a tree in the buffer does not hurt in comparison with other buffer management policies like least recently used (LRU)*. After defining these basic assumptions, we start reviewing and extending existing performance models.

The following models have some similarities. All models first estimate $Inter(q)$, the number of rectangles intersecting the query box q. In the case of an R^*_a-tree the number of completely contained rectangles is estimated by $Contain(q)$. The number of accesses of the R^*_a-tree is given with $Border(q) = Inter(q) - Contain(q)$ as shown in Equation 5.1 on page 46. In the following we assume that the data space is normalized to $S = [0,1)^d$ where d is the number of dimensions. The size of the query box is denoted by $q = (q_1, \cdots, q_d)$. The data space in Definition 3.1 $O = O_1 \times \cdots \times O_d = \{0, \cdots, c_1 - 1\} \times \cdots \times \{0, \cdots, c_d - 1\}$ is transformed into $\{0, \frac{1}{c_1}, \cdots, \frac{c_1-1}{c_1}\} \times \cdots \times \{0, \frac{1}{c_d}, \cdots, \frac{c_d-1}{c_d}\} = S$. Therefore, all tuples of data space O can be mapped into the data space S.

6.3.1. GRID model

The GRID model [Theodoridis and Sellis, 1996] is the simplest of the models considered in this chapter. This model assumes that all rectangles of the leaf nodes form a regular grid. No overlaps are allowed and there is no *dead space*. That means that the

whole data space is filled with hyper-rectangles. The only input parameters for the GRID model are the number d of dimensions of the data space and the number n of leaf nodes that are necessary to store all data entries. The average length of a data rectangle in the GRID model is given by:

$$\overline{r} = \frac{1}{\sqrt[d]{n}} \tag{6.1}$$

Figure 6.1 shows an example with a two-dimensional data space with $n = 64$ data rectangles (leaf nodes) and each rectangle has the length of $\overline{r} = \frac{1}{8}$ in both dimensions. We estimate the number of rectangles which intersect the query box completely by (cf. left part of Figure 6.1):

$$Inter_G(q) = \prod_{j=1}^{d} \min\left\{\frac{q_j}{\overline{r}} + 1, \frac{1}{\overline{r}}\right\} \quad \forall q \in 2^I \tag{6.2}$$

The number of gray shaded rectangles in the left part of Figure 6.1 corresponds to the number of blocks which are accessed when an R^*-tree is used. In this example 16 rectangles are read for a query box of size $q = (\frac{3}{8}, \frac{3}{8})$. The subscript G indicates the GRID model. The $\min$ operator in Equation 6.2 is necessary for larger query boxes. Otherwise the estimator would calculate for query boxes larger than $q = (\frac{\overline{r}-1}{\overline{r}}, \frac{\overline{r}-1}{\overline{r}})$ in at least one dimension the a wrong number of intersecting data rectangles.

Theodoridis et al. do not consider aggregated data in the inner nodes [Theodoridis and Sellis, 1996]. Therefore, we extend the GRID model. For the R_a^*-tree we estimate the number of rectangles completely contained in the query box. This is the number of white rectangles inside the gray shaded frame in the right part of Figure 6.1. Then we calculate the estimator for the number of rectangles inside the query box by:

$$Contain_G(q) = \prod_{j=1}^{d} \max\left\{\frac{q_j}{\overline{r}} - 1, 0\right\} \quad \forall q \in 2^I \tag{6.3}$$

In the above example the query box contains four rectangles completely. We save this number of disk accesses when a structure like the R_a^*-tree is used in comparison with use of an R^*-tree.

The number of necessary disk accesses for an R_a^*-tree according to the GRID model is therefore the difference between the two estimators:

$$Border_G(q) = Inter_G(q) - Contain_G(q) \quad \forall q \in 2^I \tag{6.4}$$

The GRID model has the main benefit that it is very easy and fast to compute to get a rough estimate of the needed disk accesses for data that is uniformly distributed over the data space. Only the number of *leaf* nodes and the size of the query box is needed. The GRID model neither considers the size of the different data rectangles nor the distribution of rectangles. The next model will overcome one of these drawbacks and use the size of the rectangles.

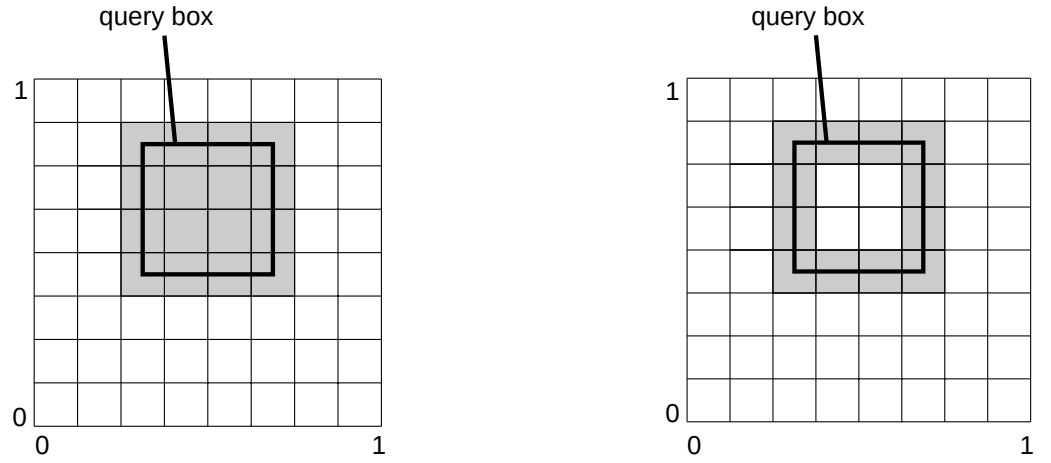


Figure 6.1.: GRID model with $d = 2$, $n = 64$, and $\overline{r} = \frac{1}{8}$

6.3.2. SUM model

The SUM model is more precise than the GRID model, but it is also more complex to compute. The first part of the SUM model is developed according to two independently published approaches [Kamel and Faloutsos, 1993], [Pagel et al., 1993]. If the data space S is normalized to $[0, 1)^d$ the probability for a point query to access a given rectangle equals the size of that rectangle. For example, if the size of a rectangle is 0.15 relative to the data space, the probability of a point query to access this rectangle is 15 %. According to the SUM model the probability that a rectangle intersects with the rectangle of query box q is the length of the rectangle extended in each dimension by the length of the query box in each dimension (cf. Figure 6.2). The size of the query box q and the size of rectangle i are needed for computation only. The actual positions of the query boxes and the actual positions of rectangles are not needed in these computations. Opposite to the approaches from literature where the case that the sum $q_j + r_{ij}$ can become greater than 1 is not excluded, we solve this problem by using a $\min$-function. For each rectangle $i = 1, \cdots, n$ we calculate the probability $p_i = \prod_{j=1}^{d} \min\{q_j + r_{ij}, 1\}$, so that it intersects the query box. The sum of all these probabilities provides the exptected number of rectangles intersecting the query box q according to SUM model as:

$$Inter_S^n(q) = \sum_{i=1}^{n} \underbrace{\prod_{j=1}^{d} \min\{q_j + r_{ij}, 1\}}_{p_i} \quad \forall q \in 2^I \tag{6.5}$$

The superscript n indicates that all n rectangles are used to calculate the expected value and the subscript S indicates that the SUM model is used.

In the second part, we extend the SUM model and consider the use of aggregated data. For a tree-based index structure with aggregated data inside the index

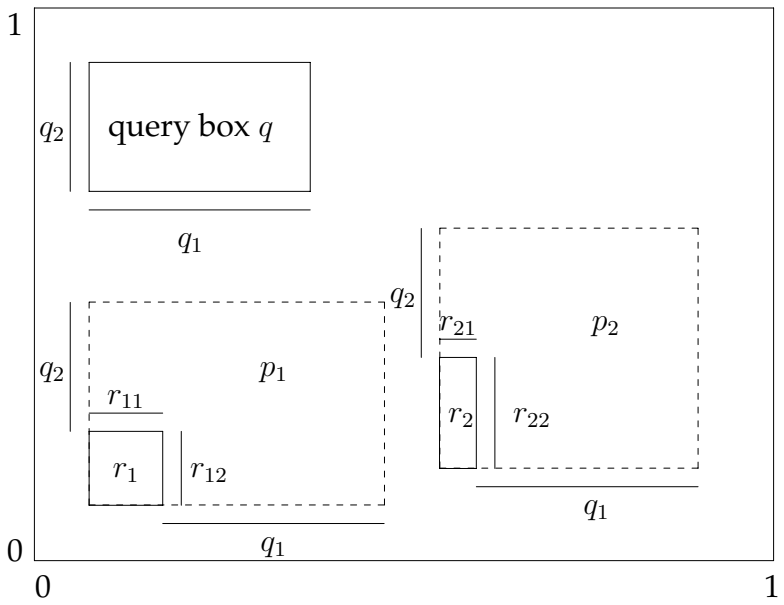


Figure 6.2.: Illustration of p_1, p_2 for SUM model

structure we need to estimate the probability that a query box q completely contains a rectangle i (*e. g.* rectangles a and c in Figure 5.2 on page 45). To the best of our knowledge this measure is not presented in literature before. From Figure 6.3 one can derive $p'_i = \prod_{j=1}^{d} \max\{q_j - r_{ij}, 0\}$. To prevent $q_j - r_{ij}$ to become negative, we use the $\max$ function. The expected number of rectangles completely contained in the query box is computed by:

$$Contain_S^n(q) = \sum_{i=1}^{n} \underbrace{\prod_{j=1}^{d} \max\{q_j - r_{ij}, 0\}}_{p'_i} \quad \forall q \in 2^I \tag{6.6}$$

According to Equation 5.1 the expected number of blocks to be read from secondary memory is the difference between the number of nodes intersecting the query box and the number of nodes completely contained in the query box:

$$Border_S^n(q) = Inter_S^n(q) - Contain_S^n(q) \tag{6.7}$$

With the given functions $Inter_S^n(q)$ and $Border_S^n(q)$ we are able to predict the number of disk accesses for a given query box for tree structures with and without materialized aggregated data in the inner nodes according to the SUM model.

6.3.3. Equivalence of GRID model and SUM model

The GRID model and the SUM model seem to be quite different. However, under some preconditions they area equal.

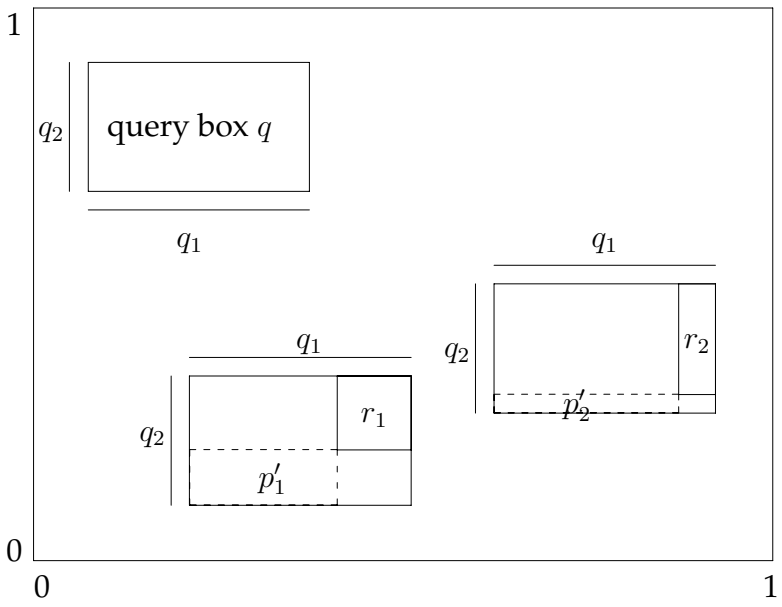


Figure 6.3.: Illustration of p'_1, p'_2

Theorem 6.1 Assume that in the SUM model there are no overlaps between *leaf* nodes, the *leaf* nodes cover the whole data space (completeness), and all *leaf* nodes have the same size (equisize). Then it follows that $Inter_S^n(q) = Inter_G(q)$ and $Border_S^n(q) = Border_G(q) \quad \forall q \in 2^I$.

Proof:

We denote the length of a rectangle in the SUM model as r.

$$\begin{aligned}
Inter_S^n(q) &= \sum_{i=1}^{n} \prod_{j=1}^{d} \min\{q_j + r, 1\} \\
&= nr^d \prod_{j=1}^{d} \min\left\{\frac{q_j}{r} + 1, \frac{1}{r}\right\} \\
&= \underbrace{n\left(\frac{1}{\sqrt[d]{n}}\right)^d}_{=1} \prod_{j=1}^{d} \min\left\{\frac{q_j}{\overline{r}} + 1, \frac{1}{\overline{r}}\right\} \\
&= \prod_{j=1}^{d} \min\left\{\frac{q_j}{\overline{r}} + 1, \frac{1}{\overline{r}}\right\} \\
&= Inter_G(q)
\end{aligned}$$

Similar transformations prove $Contain_S^n(q) = Contain_G(q)$. Therefore, $Border_S^n(q) = Border_G(q)$ and the models are equivalent under the above mentioned assumptions. □

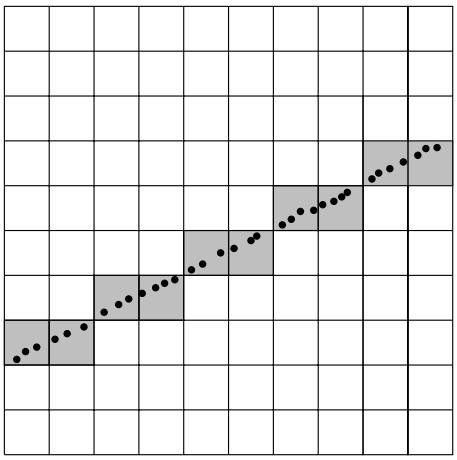


Figure 6.4.: Example of a one-dimensional figure in a two-dimensional data space, $M = 100$, $M_f = 10$, $d_f = 1$

From Theorem 6.1 it follows that the SUM model is accurate, if the rectangles of the leaf nodes form a grid.

6.3.4. FRACTAL model

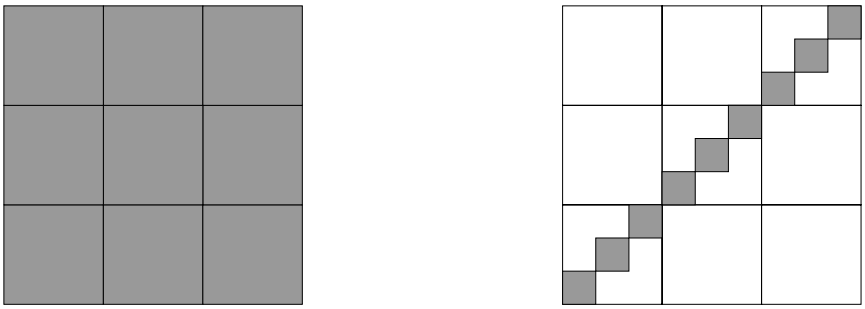
The FRACTAL model is based on the concept of fractal dimensions [Faloutsos and Kamel, 1994]. Here, we concentrate on the number of fractal dimensions d_f of real data sets and use this concept to quantify the deviation from the uniform distribution of data.

We describe the main idea for the calculation of d_f: A grid with M squares is laid over the real dataset and for each cell of the grid it is tested, if at least one point of the real data set is contained in this cell. The number of cells that are filled with at least one point is called box count M_f. The ratio of the number of cells of the grid and the box count gives a parameter for the deviation from the uniform distribution of data.

Figure 6.4 shows points of a real data set forming a line segment. The dimensionality of the data space is $d = 2$. We lay a grid with $M = 100$ cells over the real data set. Then we count number of grid cells that contain at least one data point as box count $M_f = 10$. The following equation calculates the fractal dimension from these values by:

$$d_f = d\frac{\log(M_f)}{\log(M)} \tag{6.8}$$

In the example in Figure 6.4 the fractal dimension is $d_f = 1$. We detected a one-dimensional real data set (the line segment) in the two-dimensional data space. The FRACTAL model applies the fractal dimension d_f to calculate the average length of



GRID model FRACTAL model

Figure 6.5.: $d = 2$, $d_f = 1$, $n = 9$, $\Rightarrow$ GRID model: $\overline{r} = \frac{1}{3}$, FRACTAL model: $r' = \frac{1}{9}$

data rectangles. This length is computed by:

$$r' = \frac{1}{\sqrt[d_f]{n}}, \tag{6.9}$$

We give an example on how d_f is used. We assume an index structure occupies $n = 9$ leaf nodes. The left part of Figure 6.5 shows the rectangles of a GRID model which assumes that the rectangles form a regular grid. The GRID model computes a value of $\overline{r} = \frac{1}{3}$ as the average length of rectangles. The FRACTAL model takes the fractal dimension d_f into account and estimates from this d_f the average length of the rectangles as $r' = \frac{1}{9}$. The size of the rectangles are shown on the right side of Figure 6.5. This example shows that the concept of fractal dimensions considers the part of the data space which the real data sets occupies. Opposite to that the GRID model assumes that the whole data space is used. Given the size of the query box $q = (q_1, \cdots, q_d)$ and the expected size of one data rectangle r' the FRACTAL model estimates the number of rectangles intersecting the query box similar to the SUM model:

$$Inter_F(q) = \sum_{i=1}^{n} \prod_{j=1}^{d} \min\{q_i + r', 1\} \quad \forall q \in 2^I \tag{6.10}$$

Similar, we calculate the number of rectangles completely contained in the query box as:

$$Contain_F(q) = \sum_{i=1}^{n} \prod_{j=1}^{d} \max\{q_i - r', 0\} \quad \forall q \in 2^I \tag{6.11}$$

The number of rectangles that are accessed by an R_a^*-tree is therefore given as:

$$Border_F(q) = Inter_F(q) - Contain_F(q) \quad \forall q \in 2^I \tag{6.12}$$

The subscript F indicates the use the FRACTAL model. Next, some equivalences between the different models are investigated.

6.3.5. Equivalence between FRACTAL model, SUM model, and GRID model

We presented three different models. However, under specific conditions these models compute the same result. We formulate these equivalences in Theorem 6.2.

Theorem 6.2 Assume that in FRACTAL model and in SUM model there are no overlaps between *leaf* nodes, the *leaf* nodes cover the whole data space (completeness), and all *leaf* nodes have the same size. It follows that the FRACTAL model, the SUM model, and the GRID model estimate the same values.

Proof: If the whole data space is filled, each cell of the grid contains at least one point. Therefore, the box count $M_f = M$. From this follows $d_f = d\frac{\log(M)}{\log(M)} = d$. Therefore, $r' = \frac{1}{\sqrt[d]{n}} = \frac{1}{\sqrt[d]{n}} = \overline{r}$.

$$\begin{aligned} Inter_F(q) &= \sum_{i=1}^{n}\prod_{j=1}^{d}\min\{q_i + r', 1\} \\ &\overset{r'=\overline{r}}{=} \sum_{i=1}^{n}\prod_{j=1}^{d}\min\{q_i + \overline{r}, 1\} \\ &= Inter_S^n(q) \\ &\overset{\text{Theorem 6.1}}{=} Inter_G(q) \end{aligned}$$

Similar transformation can be done to proof $Contain_F(q) = Contain_S^n(q) = Contain_G(q)$. Therefore, $Border_F(q) = Border_S^n(q) = Border_G(q)$ and the three models are equivalent under the above defined assumptions. □

6.4. PISA model

This section introduces a new approach to estimate the number of disk accesses for a given range query. This model considers the actual distribution of data rectangles and the distribution of locations of query rectangles. The new model is called **P**erformance model for **I**ndex **S**tructures with and without **A**ggregated data (PISA) [Jürgens and Lenz, 1999a]. First we calculate the probability that two rectangles intersect with each other. Although we only present the two-dimensional case, the term rectangle refers to a multidimensional interval as noted before. We assume that the intersection probabilities in all dimensions are independent from each other. If the distributions are dependent on each other, we can still extend our model to this case. However, the model is getting more complex (cf. Section 6.8). For the case of two independent dimensions we consider two rectangles as shown in Figure 6.6. The two rectangles only intersect, if and only if, they intersect in each dimension. Rectangles are projected on each dimension separately. We define that the corresponding intervals A and B have length $a, b \in [0, 1)$. We consider the two intervals

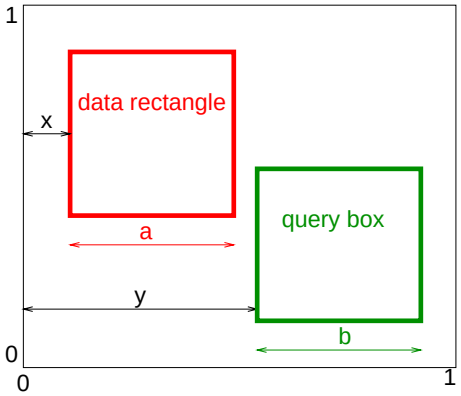


Figure 6.6.: One-dimensional projection of data rectangle and query box

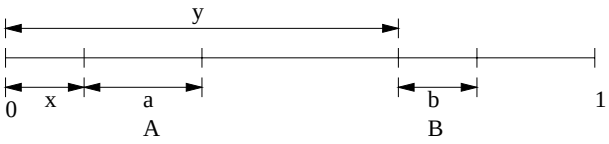


Figure 6.7.: The position of interval A of length a and interval B of length b

$A = [x, x + a)$ and $B = [y, y + b)$ where x is a realization of the random variable X distributed over $[0, 1 - a)$ and y is a realization of the random variable Y distributed over $[0, 1 - b)$, (cf. Figure 6.7). The variable x denotes the space between the leftmost point of interval A and 0. Similar the variable y denotes the space between the leftmost point of interval B and 0. We define an indicator function $f : [0, 1]^2 \rightarrow \{0, 1\}$ to decide whether two intervals A and B intersect or not:

$$f(x, y) = \begin{cases} 1 & : \quad A \text{ intersects } B \\ 0 & : \quad \text{otherwise} \end{cases} = \begin{cases} 1 & : \quad (y \geq x - b) \wedge (y \leq x + a) \\ 0 & : \quad \text{otherwise} \end{cases} \tag{6.13}$$

The gray shaded area in Figure 6.8 represents all possible combinations with the two intervals intersecting each other. If x and y are equal, the rectangles intersect definitely. The probability that two given rectangles intersect each other is calculated. We assume that the positions of data and query rectangles are distributed according to some parametric density function over the data space. Let d_a and d_b be the density functions of the positions of the intervals A and B. The probability that the two intervals intersect is P(A intersects B):

$$h_1(a, b) = \begin{cases} \frac{1}{(1-a)(1-b)} \int_0^{1-a} \int_0^{1-b} f(x, y)\, d_a(x)\, d_b(y)\, dy\, dx & : \quad a + b < 1 \\ 1 & : \quad \text{otherwise} \end{cases} \tag{6.14}$$

The PISA model uses Equation 6.14 later to adapt to different distributions. To compute the probability that a rectangle intersects the query box in multidimensional

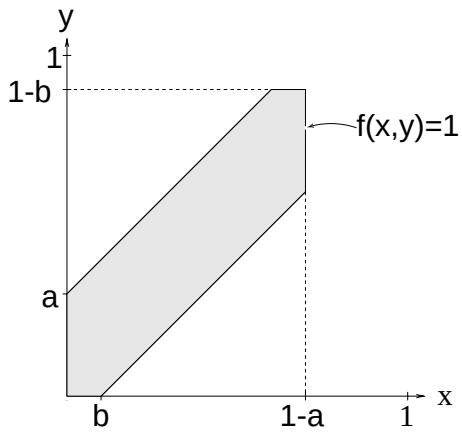


Figure 6.8.: A intersects B, $f(x, y) = 1$, in the gray shaded area

data space, h_1 is applied in all dimensions and the different probabilities are multiplied under the above assumption of dimensional independence. The total expected number of accesses is given by:

$$Inter_P^n(q) = \sum_{i=1}^{n} \prod_{j=1}^{d} h_1(q_j, r_{ij}) \quad \forall q \in 2^I \tag{6.15}$$

If aggregates in the inner nodes of the tree are available, there is no need to access *leaf* nodes whose rectangles are completely contained in the query box. We apply the same reasoning as for the derivation of $Inter_P^n(q)$. Firstly, we define an indicator function $g : [0, 1]^2 \rightarrow \{0, 1\}$ that decides whether one interval is completely contained in the other one:

$$g(x, y) = \begin{cases} 1 & : \quad A \text{ contains } B \\ 0 & : \quad \text{otherwise} \end{cases} = \begin{cases} 1 & : \quad (y \geq x) \wedge (y < x + a - b) \\ 0 & : \quad \text{otherwise} \end{cases} \tag{6.16}$$

The gray shaded area in Figure 6.9 represents the combinations of x and y values where the B is completely contained in A. If b is greater than a, it is not possible for B to be completely contained in A. The probability that two intervals intersect each other is calculated by P(A contains B):

$$h_2(a, b) = \begin{cases} \frac{1}{(1-a)(1-b)} \int_0^{1-a} \int_0^{1-b} g(x, y)\, d_a(x)\, d_b(y)\, dy\, dx & : \quad a > b \\ 0 & : \quad \text{otherwise} \end{cases} \tag{6.17}$$

The function h_2 is used to compute the expected number of rectangles that are completely contained in the query box. This formula is similar to Equation 6.15 for the case without aggregated data:

$$Contain_P^n(q) = \sum_{i=1}^{n} \prod_{j=1}^{d} h_2(q_j, r_{ij}) \quad \forall q \in 2^I \tag{6.18}$$

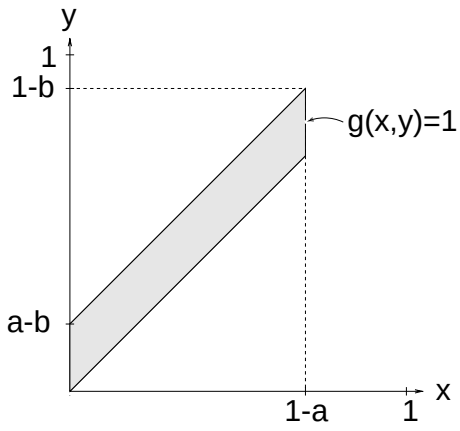


Figure 6.9.: A contains B, $g(x, y) = 1$, in the gray shaded area

Equation 6.15 and Equation 6.18 consider the distribution of the rectangles and the actual distribution of the query boxes. Next, we apply the PISA model to a structure like the R_a^*-tree. We calculate the expected number of accesses as the difference of the two previously calculated functions:

$$Border_P^n(q) = Inter_P^n(q) - Contain_P^n(q) \quad \forall q \in 2^I \tag{6.19}$$

6.5. Computational Efficiency of SUM model and PISA model

With a large number of rectangles, the performance measures $Border(q)$ of SUM model and PISA model are expensive to compute, because we have to compute the actual size of each rectangle. One idea to speed-up computation is to assume that all rectangles are quadratic and have the same size. The average length $\tilde{r}$ of all rectangles is given by:

$$\tilde{r} = \frac{1}{n} \sum_{i=1}^{n} \tilde{r_i} \quad \text{with} \quad \tilde{r_i} = \sqrt[d]{\prod_{j=1}^{d} r_{ij}} \tag{6.20}$$

If $\tilde{r}$ is used to get *a faster to compute* expected number of inner blocks according to the SUM model, it follows:

$$Contain_S^1(q) = n \prod_{j=1}^{d} \max\{q_j - \tilde{r}, 0\} \quad \forall q \in 2^I \tag{6.21}$$

The superscript 1 indicates that just one rectangle (the *average* rectangle) represents all rectangles. One could argue that this simplification works only well for data

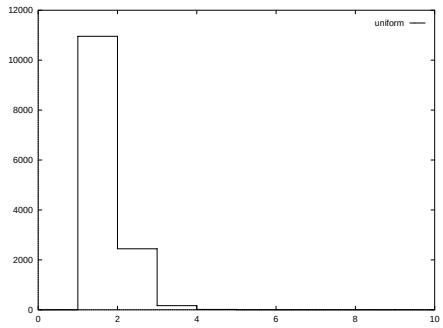


Figure 6.10.: Histogram of number of rectangles for uniformly distributed data

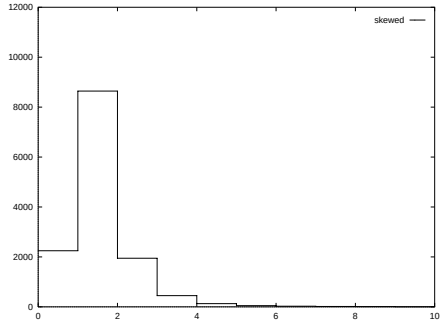


Figure 6.11.: Histogram of number of rectangles for skewed data

where all rectangles have approximately the same size, i. e. standard deviation is neglectable.

Another way of refining the model is to classify all rectangles into k different size classes. This is done in the following way: we transform each rectangle to a square having the same area as the rectangle. The length of this square classifies the rectangle as follows:

Definition 6.2 Let k be the number of different classes. The number of rectangles belonging to class l is given by:

$$u_l = \left| \left\{ i | i \in \{1, \cdots, n\} \wedge \frac{l}{k} \leq \tilde{r}_i < \frac{l+1}{k} \right\} \right|, \quad l \in \{0, \cdots, k-1) \tag{6.22}$$

Figure 6.10, Figure 6.11, and Figure 6.12 show histograms for the three different data sets. The number of classes is set to $k = 100$, which is equivalent to a class width of $\frac{1}{k} = 0.01$.

The histograms show that less than the first 10 values represent the distribution of the length of the rectangles.

Each class l is represented by a rectangle with equal length in all dimensions. The length is the center of the interval $\left[\frac{l}{k}, \frac{l+1}{k}\right)$. Having the size of the representative

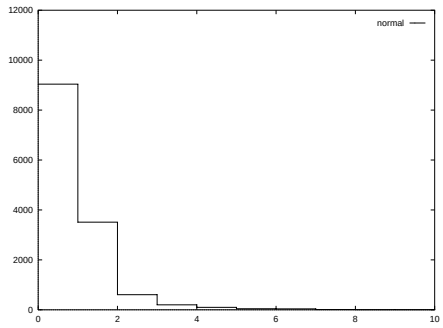


Figure 6.12.: Histogram of number of rectangles for normally distributed data

rectangle and the number of rectangles of each class, the estimator for inner nodes according to SUM model is:

$$Contain_S^k(q) = \sum_{l=0}^{k-1} u_l \prod_{j=1}^{d} \max\left\{q_j - \frac{l+0.5}{k}, 0\right\} \quad \forall q \in 2^I \tag{6.23}$$

The superscript k shows that k classes are used. In this chapter the SUM model and PISA model are used in three different *precision modes*:

- The 1-rectangle case. The *average* rectangle according to Equation 6.20 represents all rectangles (*e. g.* $Contain_S^1$ in Equation 6.21).
- The k-rectangle case. We map every rectangle into exactly one out of k classes. Each class is represented by one rectangle (*e. g.* $Contain_S^k$ in Equation 6.23).
- The n-rectangle case. Here we use the actual size of each rectangle. (*e. g.* $Contain_S^n$ in Equation 6.6).

The measures $Inter_S^1$, $Contain_S^1$, $Inter_P^1$, $Contain_P^1$, $Inter_P^k$, and $Contain_P^k$ are defined analogously.

6.6. Adapting PISA model to different distributions

This section shows how PISA is adapted to different distributions. We will focus on uniform, skewed, and normal distributions to show in detail how the model is adaptable.

6.6.1. Uniformly distributed data

We start to show the flexibility of PISA model with uniformly distributed data. Uniformly distributed data is shown in Figure 5.9 on page 58. The density functions

$d_a(x)$ and $d_b(y)$ with parameter constraints $0 \leq a, b < 1$ are:

$$d_a(x) = \begin{cases} \frac{1}{1-a} & : \quad 0 \leq x \leq 1-a \\ 0 & : \quad \text{otherwise} \end{cases} \tag{6.24}$$

$$d_b(y) = \begin{cases} \frac{1}{1-b} & : \quad 0 \leq y \leq 1-b \\ 0 & : \quad \text{otherwise} \end{cases} \tag{6.25}$$

From Equation 6.14 we get the probability that two intervals intersect [Lenz and Jürgens, 1998]:

$$h_1(a,b) = \begin{cases} \frac{a+b-a^2-b^2-ab}{(1-a)(1-b)} & : \quad a+b<1 \\ 1 & : \quad \text{otherwise} \end{cases} \tag{6.26}$$

The probability that one interval completely contains the other interval is computed by combining Equation 6.17 and Equation 6.25 to:

$$h_2(a,b) = \begin{cases} \frac{a-b}{1-b} & : \quad a>b \\ 0 & : \quad \text{otherwise} \end{cases} \tag{6.27}$$

Equation 6.15 on page 73 for $Inter_P^n(q)$ and Equation 6.18 on page 73 for $Border_P^n(q)$ allow us to predict the performance of the index structures with and without aggregates in the structure for a given query box size.

6.6.2. Skewed data

Next, we investigate how PISA can be adapted to skewed data, cf. Figure 5.10 on page 59. Its density function is given by:

$$d_a(x) = \begin{cases} \frac{2x}{1-a} & : \quad 0 \leq x \leq 1-a \\ 0 & : \quad \text{otherwise} \end{cases} \tag{6.28}$$

$$d_b(y) = \begin{cases} \frac{2y}{1-b} & : \quad 0 \leq y \leq 1-b \\ 0 & : \quad \text{otherwise} \end{cases}$$

The empirical one-dimensional density function can be seen in the left part of Figure 5.10 on page 59. The distribution of the rectangles is presented in right part of Figure 5.10. Most points are in the upper right corner of the data space.

We use Equation 6.14 and the definition of d_a and d_b in Equation 6.28 to calculate the probability that two intervals intersect each other (for $a+b<1$):

$$\begin{aligned} h_1(a,b) = & \frac{4}{(1-a)^2(b-1)^2}\left(\frac{b^2}{2} - ab^2 + \frac{3a^2b^2}{4} - b^3\frac{4ab^3}{3}\right. \\ + & \frac{7b^4}{12} + \frac{1}{3}(1-a-b)^3(a+b) - \frac{1}{3}b^3(a+b) \\ + & \left.\frac{1}{4}(1-a-b)^2(a^2-b^2) - \frac{1}{4}b^2(a^2-b^2)\right) \end{aligned} \tag{6.29}$$

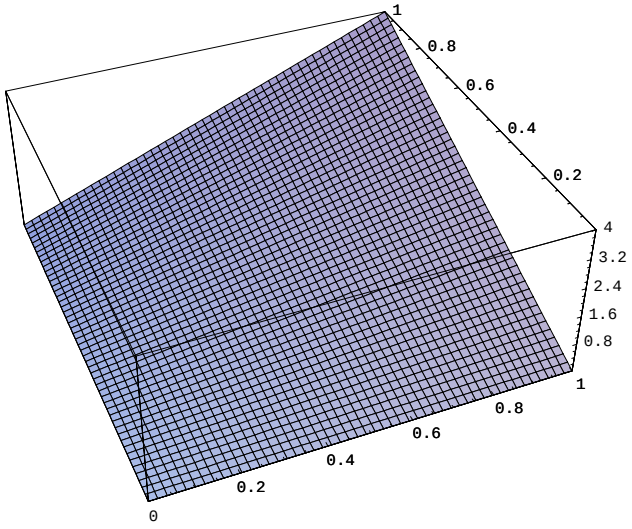


Figure 6.13.: Two-dimensional distribution $d_{a,b}(x,y) = d_a(x) * d_b(y)$ (over $[0,1)^2$ skewed data)

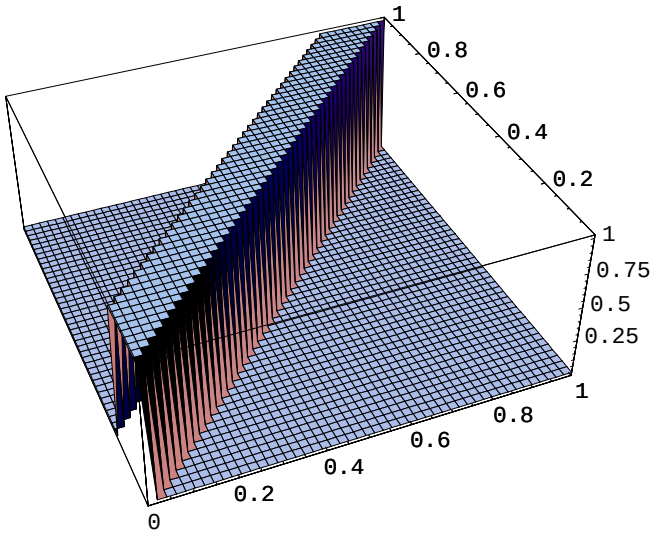


Figure 6.14.: Function $f(x,y)$

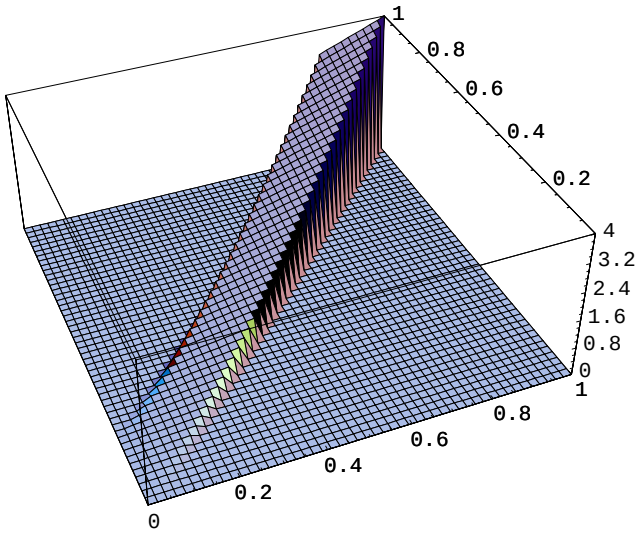


Figure 6.15.: $d_{a,b}(x,y) * f(x,y)$

Similarly we combine Equation 6.17 and the definition of d_a and d_b to derive the probability that one interval contains the other one for skewed data as (for $a > b$):

$$h_2(a,b) = \frac{\frac{4}{3}(1-a)^3(a-b) + (1-a)^2(a-b)^2}{(1-a)^2(b-1)^2} \tag{6.30}$$

Figure 6.13, Figure 6.14, and Figure 6.15 illustrates the two-dimensional case with respect to h_1. Figure 6.13 shows the two-dimensional density function $d_{a,b}(x,y) = d_a(x) * d_b(y)$. Figure 6.14 displays the indicator function $f(x,y)$. Figure 6.15 represents the integrand of Equation 6.17 as a the product of the functions $f(x,y) * d_{a,b}(x,y)$. The volume under this function is the result of the integral which is calculated as h_2.

For uniformly distributed data and for skewed data we can compute the performance measures $Border(q)$ as closed expressions.

6.6.3. Normally distributed data

Next, we adapt the PISA model to 'bell-shaped' normal distributed data. The normal distribution is a symmetric distribution with no closed expression for its distribution function. The test data is generated with the Box-Muller method [Box and Muller, 1958]. To standardize normal distributed random almost on $[0,1]$ the functional parameters $\mu = \frac{1}{2}$ and $\sigma = \frac{1}{8}$ are chosen. The left side of Figure 5.11 on page 59 shows the empirical distribution of one-dimensional data. The right side of Figure 5.11 shows the distribution of the rectangles when normally dis-

tributed points are indexed by an R^*-tree. The density functions are ($\mu \in \mathbb{R}$, $\sigma \in \mathbb{R}_+$):

$$d_a(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{6.31}$$

$$d_b(y) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(y-\mu)^2}{2\sigma^2}} \tag{6.32}$$

Equation 6.14 and Equation 6.17 cannot be solved analytically if d_a and d_b are the normal density functions. Therefore, we apply numeric approximation and approximate the distribution functions by a polynome from [Ibbetson, 1963]. Further approximations are necessary in order to compute the integrals in Equation 6.14 and Equation 6.17. Appendix B on page 133 presents these computations in detail.

$$h_2(a,b) = \frac{1}{2}(U(a,b) + L(a,b)) \tag{6.33}$$

The two functions $U_{a,b}$ and $L_{a,b}$ are shown in Equation B.1 and Equation B.4 on page 133. They are upper and lower bounds of function h_2. Therefore, h_2 is approximated by the average of both values.

6.7. Model evaluation

This section shows the results of model evaluation using the R^*-tree and R^*_a-tree as physical data structures. The R^*-tree is chosen as a representative of multidimensional tree-based index structures. The R^*-tree is a balanced structure, well known for its robustness and ability to adapt well to different data distributions. We compare results of model evaluation and measurements. The experiments were run with uniformly, skewed, and normally distributed data. In each of the following experiments we generated ten different trees with 1,000,000 tuples in each tree. The maximum fanout is $B_{leaf} = 102$ and the minimum fanout is $b_{leaf} = 41$. The average fanout is approximately 70. We assume quadratic query boxes and the size of the query box varies between 0.05 % and 6.4 % of the data space. Larger query boxes are not of interest, because the use of an index structure is slower than a sequential scan of the whole data set if more than 7 % of all leaf nodes have to be accessed.

Figures 6.16 to 6.21 use half-logarithmic scales. The x-axis represents the size of the query box relative to the whole data space. The y-axis shows the percentage deviation:

$$Y_e = \frac{|\text{modeled values} - \text{measured values}|}{\text{measured values}} * 100 \tag{6.34}$$

A graph with stars shows the deviation of the GRID model and the a graph with diamonds represents the deviation of the FRACTAL model. The graphs of the SUM model are marked with triangles and the results of PISA model are labeled with small squares. SUM model and PISA model use three precision modes as described in Chapter 6.5.

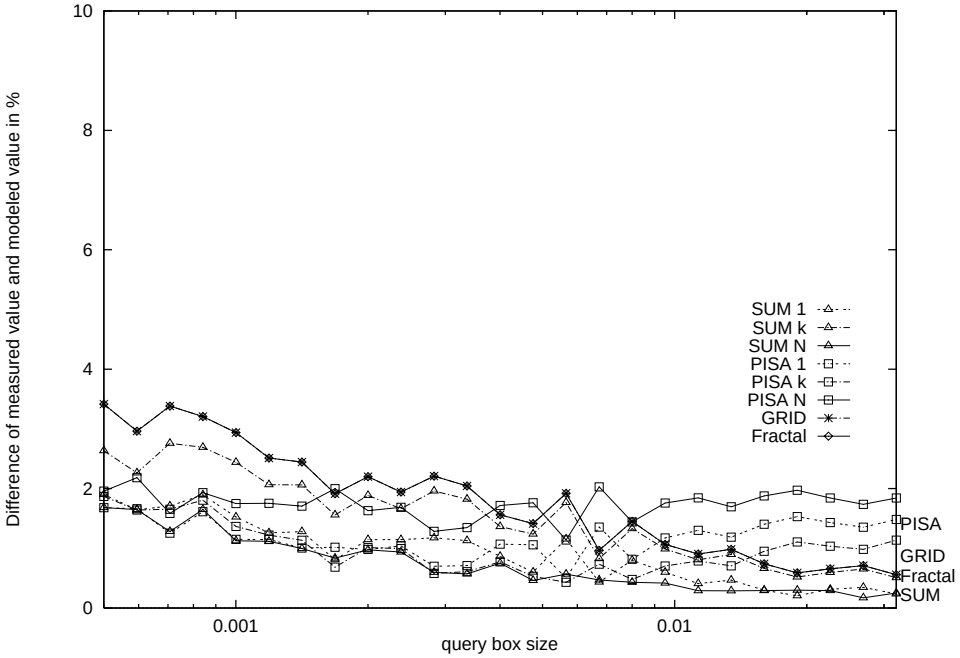


Figure 6.16.: Percentage of error Y_e of R^*-tree for uniformly distributed data

6.7.1. Uniformly distributed data

Figure 6.16 presents the percentage error Y_e for the R^*-tree for uniformly distributed data. The SUM model is slightly better than the PISA model.

Figure 6.17 shows the deviations for the R_a^*-tree. The PISA model is more precise than the SUM model, but more important than the model is the precision mode that is used in the experiments. However, we think that for uniformly distributed data all models are sufficiently precise, *i.e.* the percentage error for R^*-tree is less than 4 % and the percentage error for R_a^*-tree is less than 8 %.

6.7.2. Skewed data

For skewed data the results are shown in Figure 6.18 for the R^*-tree and in Figure 6.19 for the R_a^*-tree. The PISA model is more precise than GRID model, FRACTAL model, or SUM model. The precision mode has very little influence on the results. In Figure 6.18 the SUM model and GRID model are very close to each other and have an error rate of 35 % while the PISA model has an error of approximately 5 % for the R^*-tree. Evidently the PISA model is significantly better than the other models in case of an R_a^*-tree.

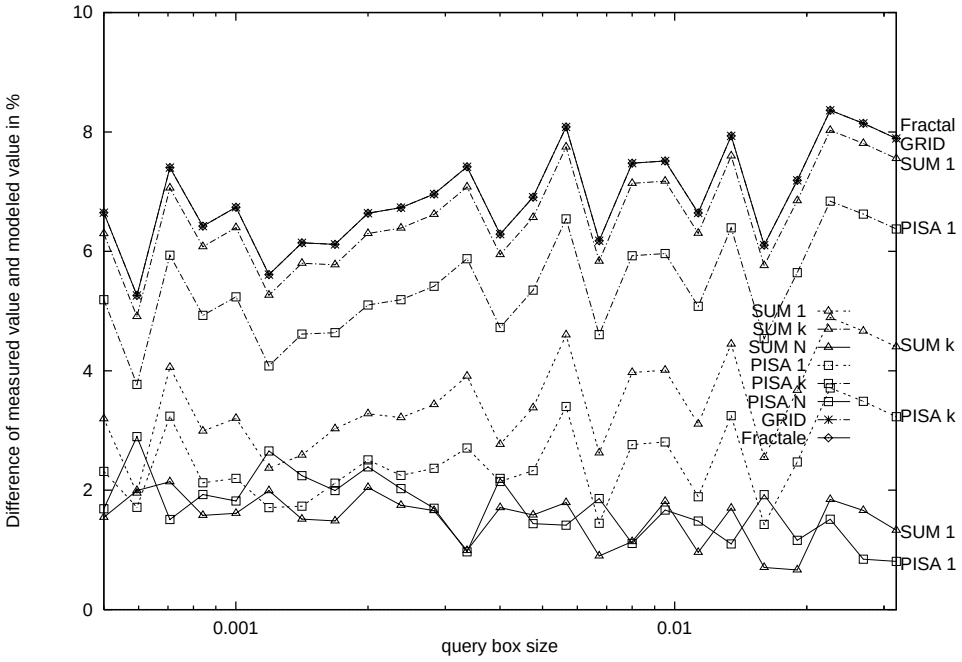


Figure 6.17.: Percentage of error Y_e of R_a^*-tree for uniformly distributed data

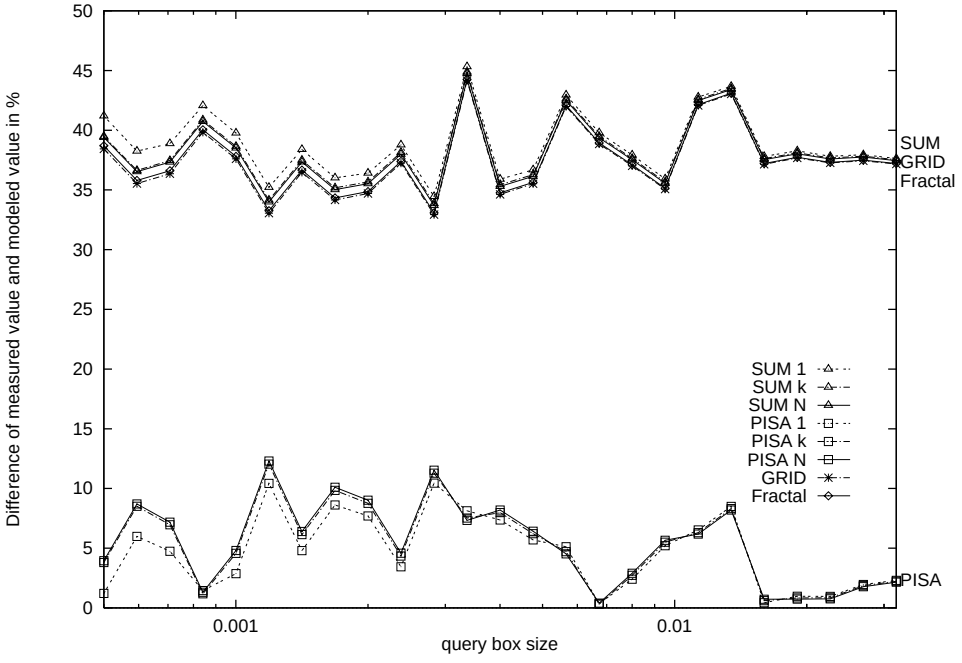


Figure 6.18.: Percentage of error Y_e of R^*-tree for skewed data

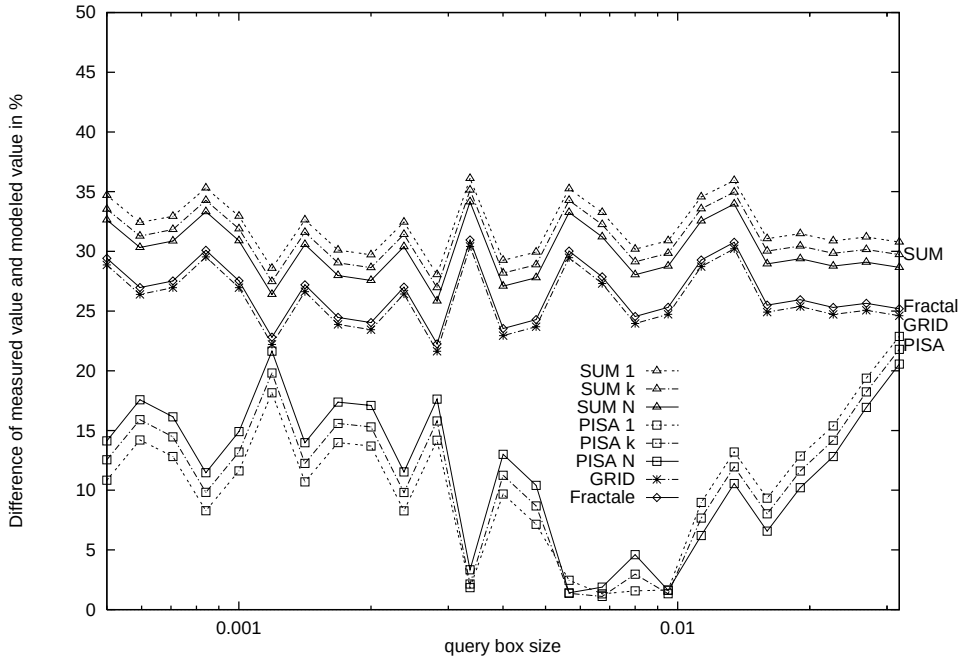


Figure 6.19.: Percentage of error Y_e of R_a^*-tree for skewed data

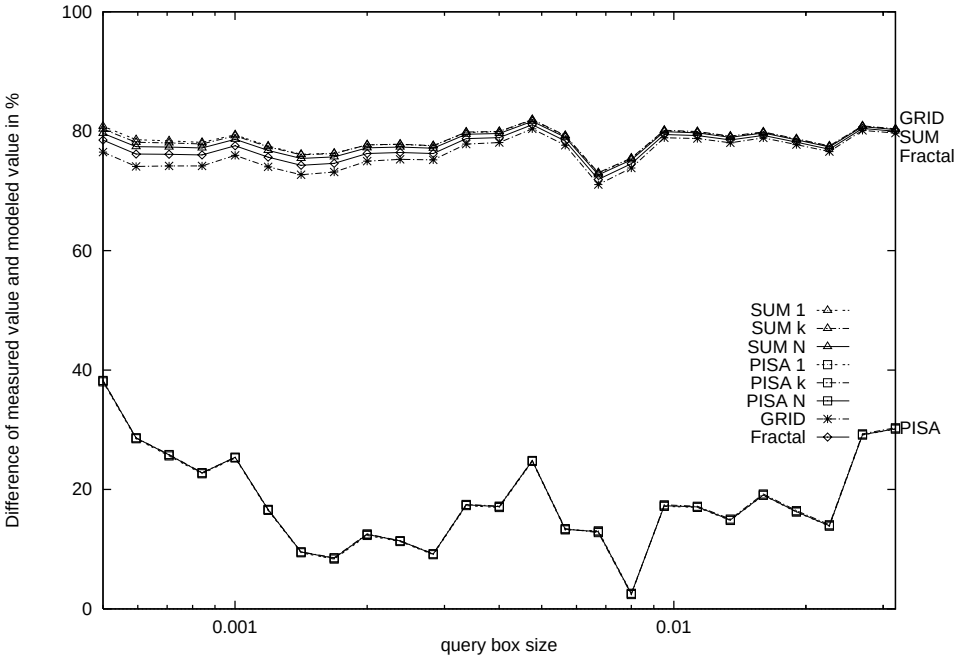


Figure 6.20.: Percentage of error Y_e of R^*-tree for normally distributed data

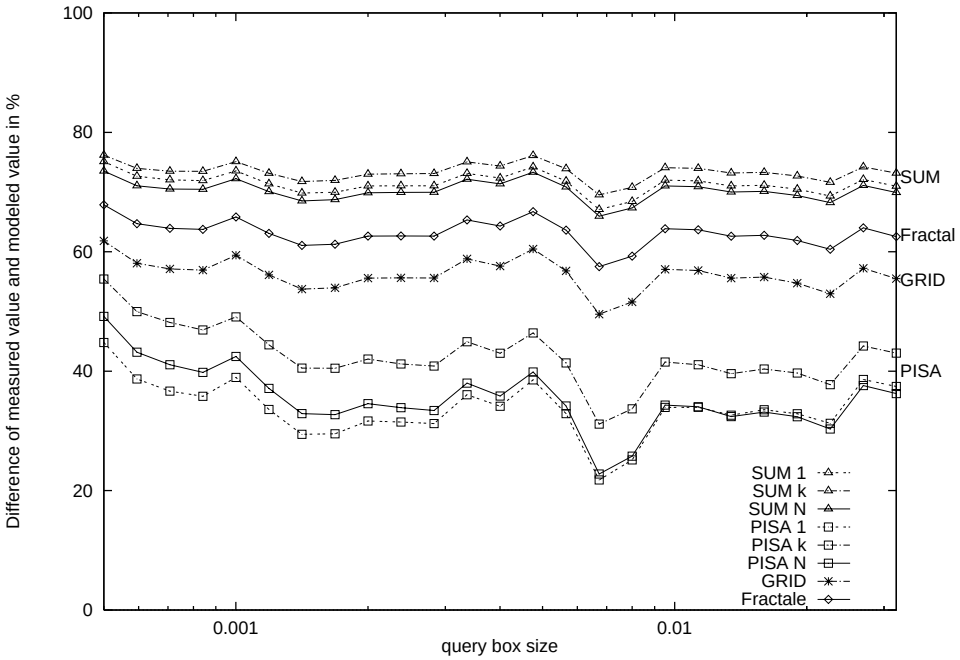


Figure 6.21.: Percentage of error Y_e of R^*_a-tree for normally distributed data

6.7.3. Normally distributed data

Figure 6.20 and Figure 6.21 show results for normally distributed data. The PISA model is much more precise than the other models. For the R^*-tree the PISA model has error rates of approximately 20 % while the SUM and GRID model have error rate of 80 % (!). Notice that for this model the precision mode has nearly no influence. It is much more important to choose the right model. For the R^*_a-tree the PISA model reaches error rates of 40 %, but is clearly more precise than the GRID or SUM model.

The evaluation of the models show that PISA model dominates all the other models. This is true due to the fact that PISA model considers the actual distribution of data and queries. The SUM model and the FRACTAL model assume implicit uniformly distributed data. Usually, real world data is not uniformly distributed. Therefore, we believe that the PISA model has a great impact on performance evaluations on real world data.

6.8. PISA model for dependent data

One assumption of the PISA model is that the joint distribution of the data can be modeled by marginal distribution on each dimension separately, i. e. independently. In many real applications this assumption does not hold and correlations between different dimensions in the data exist. We show how the PISA model is extended to use joint distributions. In this chapter, the two-dimensional case is considered

only, but extension is straightforward and can be applied to any fixed number of dimensions.

Let $d_a(x, y)$ be the two-dimensional density function of the location of data rectangles, and let $d_b(x, y)$ be the two-dimensional density function of the location of the query boxes. Let r_1, r_2 be the length and width of the data rectangle, and let q_1, q_2 denote the length and width of the query box. The probability h_1 that rectangles intersect is computed similarly as for the one-dimensional case. If $(q_1 + r_1 \leq 1) \wedge (q_2 + r_2 \leq 1)$ and $(0 \leq r_1, r_2, q_1, q_2 < 1)$ the probability is calculated as:

$$h_1(r_1, r_2, q_1, q_2) = \frac{\int_0^{1-r_1} \int_0^{1-r_2} \int_0^{1-q_1} \int_0^{1-q_2} f(x_1, x_2) f(y_1, y_2) d_a(x_1, y_1) d_b(x_2, y_2) dy_2 dx_2 dy_1 dx_1}{(1 - r_1)(1 - r_2)(1 - q_1)(1 - q_2)} \tag{6.35}$$

If a structure like the R_a^*-tree is used, we need the probability h_2 that the query box contains completely a given rectangle. If $(q_1 > r_1) \wedge (q_2 > r_2)$ and $(0 \leq r_1, r_2, q_1, q_2 < 1)$, the probability is:

$$h_2(r_1, r_2, q_1, q_2) = \frac{\int_0^{1-r_1} \int_0^{1-r_2} \int_0^{1-q_1} \int_0^{1-q_2} g(x_1, x_2) g(y_1, y_2) d_a(x_1, y_1) d_b(x_2, y_2) dy_2 dx_2 dy_1 dx_1}{(1 - r_1)(1 - r_2)(1 - q_1)(1 - q_2)} \tag{6.36}$$

If $(q_1 \leq r_1) \vee (q_2 \leq r_2)$ the rectangle is too small to be completely contained inside the query box and we set $h_2(r_1, r_2, q_1, q_2) = 0$.

6.9. Extension of models

The models are based on the assumption that the leaf nodes are stored on a disk system while all inner nodes are held in main memory. The access to the secondary memory is about factor 10^5 slower than accessing main memory. Therefore, the cost models for the above described models consider only the leaf node level. In some applications this assumption does not hold and we have to extend the model to other levels of the tree.

Assume the tree has a height of h and the first j levels ($j \in \{0, \cdots, h-1\}$) are stored in main memory. Then $h - j$ levels are stored on secondary memory. For this case the above described models are applied for each level. Altogether the models are applied $h - j$ times and the results of all levels are summed up to get the total estimator.

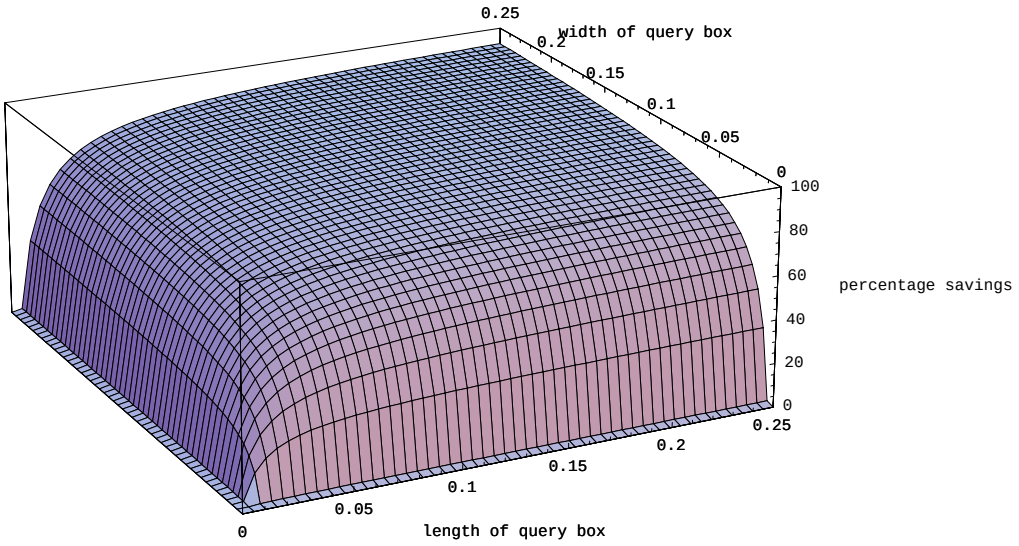


Figure 6.22.: Percentage savings of disk accesses of R^*_a-tree and R^*-tree by length and width of query box (PISA model, 40.000 leaf nodes with uniformly distributed data)

6.10. Applications of models

In this section we investigate under which conditions (i. e. size, form, number of dimensions) the use of aggregated data yields better performance compared to neglecting aggregated data.

6.10.1. Savings of R^*_a-tree depending on the query box size and form

Figure 6.22 shows the estimated savings in percent of the R^*_a-tree over the R^*-tree for different sizes of the query box. The x- and y-coordinates are the length and the width of the query box. The z-coordinate shows the savings in percent. The R^*_a-tree outperforms the R^*-tree for large quadratic query boxes. Points close to the x-axis or y-axis represent thin or short queries, so called 'spaghetti' queries. For this class of queries there is no or only small benefit of using the R^*_a-tree, whereas for large quadratic queries the savings are significant. Notice the scale of Figure 6.22. Only for very thin of very short query boxes are there no savings using the R^*_a-tree.

6.10.2. Savings of R^*_a-tree depending on the number of dimensions

Figure 6.22 shows that savings depend on the size and form of the query box. It is evident that not only the query box sizes influences savings. The data also heavily

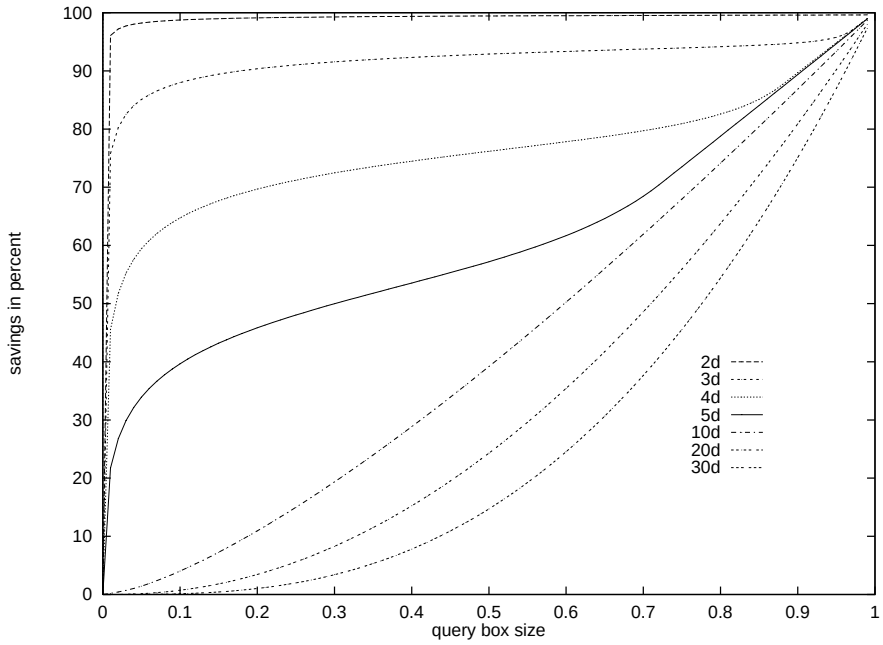


Figure 6.23.: Savings of accesses for different number of dimensions

influences the performance of the index structure.

Figure 6.23 shows the percentage of savings of the structure with and without aggregated data for different number of dimensions. It is assumed that there is an index structure with $n = 10^6$ leaf nodes and the number of dimensions is varied. We use only quadratic query boxes and vary the size of the query box relatively to the data space. Figure 6.23 shows that the structure is efficient for few dimensions, i. e. $d \leq 5$. For the two and three-dimensional case the structure rapidly reaches savings of close to 90 %. For up to 5 dimensions there can be an significant performance improvement gained with the aggregated data. For more than ten dimensions the improvements are rather small. As is well known the R^*-tree is not very efficient for high dimensional data. Our extension to the R^*-tree with aggregated data is most efficient in cases where the R^*-tree is efficient. Consequently this extension should be applied in case of less than five dimensions and large query boxes.

6.11. Summary

This chapter discussed models for non-extended and models for extended multidimensional index structures. The key idea of the extension is to store materialized aggregates in the inner nodes of the tree structure. The use of this materialized data reduces the processing time of range queries.

To estimate the performance of multidimensional index structures various existing performance models are studied and extended. A new **P**erformance model for

Index **S**tructures with and without **A**ggregated data (PISA) is presented. We show how PISA can be adapted to uniformly, skewed, and normally distributed data. The R^*-tree and R_a^*-tree is taken as reference index structures to evaluate the accuracy of the models. The evaluation of the models confirms that the PISA model is more precise than other existing models. Applications of models show for what kind of data and for what kind of queries the extension of the R^*-tree to the R_a^*-tree is relative efficient.