Index

A. List of Symbols

a	length of interval A	page 71
A	set of all index entries	page 18
a_j	name of jth attribute	page 18
A_j	set of extensions of attribute a_j	page 18
b	blocksize	page 16
B	set of distinct blocksizes	page 92
B_{dir}	maximum fanout of directory node	page 23
b_{dir}	minimum fanout of directory node	page 23
B_{equal}	expected number of bitmap vectors which are read for an equality encoded bitmap index for processing a range query	page 31
b_j	number of bitmap vectors in j's dimension	page 30
b_{ji}	the ith bitmap vector on the jth attribute	page 30
B_{leaf}	maximum fanout of leaf node	page 23
b_{leaf}	minimum fanout of leaf node	page 23
B_{range}	expected number of bitmap vectors which are read for a range encoded bitmap index for processing a range query	page 33
$Border(q)$	number of blocks that have to be accessed when an R^*_a-tree is used: $Inter(q) - Contain(q)$	page 46
bw	bandwith	page 92
BW	set of distinct bandwidths	page 92
c	cardinality of data space	page 90

C	set of distinct cardinalities of data space	page 92
c_j	cardinality of data space in jth dimension	page 18
$Contain(q)$	number of leaf nodes contained in query box q	page 46
cp	child pointer	page 23
d	number of dimensions	page 18
D	set of distinct number of dimensions	page 92
$d_a(x)$	density function of position of interval A	page 72
$d_b(y)$	density function of position of interval B	page 72
d_f	fractal dimension	page 69
$data_{entry}$	entry of a data node	page 48
dir_{entry}	entry of a directory node	page 47
DW	data warehouse	page 8
e	configuration vector $(d, t, c, qs, qd, b, sf, bw, t_l)$	page 92
E	set of all configurations: $\{(d, t, c, qs, qd, b, sf, bw, t_l) \mid (d, t, c, qs, qd, b, sf, bw, t_l) \in D \times N_t \times C \times Q_s \times Q_d \times B \times SF \times BW \times T_l, (qd \leq d) \wedge (c \leq t)\}$	page 92
$f(u, v)$	additional space when switching from fanout of u to fanout of v	page 53
$f(x, y)$	characteristic function to decide if two intervals intersect	page 72
$g(x, y)$	characteristic function to decide if one intervals is contained in the other	page 73
G	input set for creation of classification tree	page 97
h	height of tree	page 51
$h_1(a, b)$	PISA model: probability that A intersects B	page 72
$h_2(a, b)$	PISA model: probability that A contains B	page 73
i	index	
I	multi dimensional interval	page 18

$Inter(q)$	number of leafs intersecting query box q	page 46
j	index on the number of dimensions	
k	number of classes of the user defined density function	page 75
K	upper bound for additional space	page 53
l_{ik}	lower border of block i in dimension k	page 36
m	total number of bitmap vectors that are stored by an bitmap index in all dimensions	page 95
m_j	number of bitmap vectors that are stored by an bitmap index in the jth dimension	page 30
M	boxes for fractal dimension model	page 69
M_f	boxes filled for fractal dimension model	page 69
n	number of leaf nodes	page 36
N	set $\{1, \cdots, n\}$	page 69
n_i	number of nodes on level i n_0: number of leaf nodes n_1: number of inner nodes on level 1 n_h: number of root nodes ($n_h = 1$)	
n_u	number of leaf nodes of structure with fanout u	page 53
n_v	number of leaf nodes of structure with fanout v	page 53
N_t	set with distinct number of tuples	page 92
O	$O_1 \times \cdots \times O_d = \{0, \cdots, c_1 - 1\} \times \cdots \times \{0, \cdots, c_d - 1\}$	page 18
O_j	$\{0, \cdots, c_j - 1\}$	page 18
OP	set of operations	page 8
P	set of d-dimensional points (tuples)	page 36
p_i	SUM model: probability that rectangle i intersects query box q	page 66
q	vector with size of query box $q = (q_1, \cdots, q_d)$ (length in each dimension)	page 19

q_j	length of query box in jth dimension	page 19
qd	query box dimensions	page 90
Q_d	set with distinct query box dimensions	page 92
qs	query box size	page 90
Q_s	set of distinct query box sizes	page 92
R	Relation $R(a_1, \cdots, a_n, s)$	page 10
r_{ij}	size of leaf node i in dimension j	page 66
$\tilde{r}$	average length of rectangle in SUM model and PISA model (1-case)	page 74
r'	average length of rectangle in FRACTAL model	page 70
$\overline{r}$	average length of rectangle in GRID model	page 65
s_{dir}	size of a directory entry	page 46
s_{min}	select the index of the structure with the minimum value	page 97
s	number of index structures which are compared	page 93
s_0	number of slice for approximation of PISA model	page 133
S	data space $[0, 1)^d$	page 64
sf	scale factor	page 91
SF	set with distinct scale factors	page 92
t	number of tuples	page 36
T	$\{1, \cdots, t\}$	page 36
tid	tuple identifier	page 18
t_r	time for random block access to secondary memory	page 16
t_s	time for sequential block access to secondary memory	page 16
$t_i : E \to \mathbb{R}^+$	expected time for processing a range queries with index structure i	page 93
	$t_1 : E \to \mathbb{R}^+$ R-tree without aggregated data	page 94
	$t_2 : E \to \mathbb{R}^+$ R-tree with aggregated data	page 94

	$t_3 : E \rightarrow \mathbb{R}^+$ equality encoded bitmap index	page 95
	$t_4 : E \rightarrow \mathbb{R}^+$ range encoded bitmap index	page 96
TDB	target database	page 8
u_l	user defined density function for distribution of sizes of rectangles	page 75
u_{ik}	upper border of block i in dimension k	page 36
v	blocks per bitmap vector	page 95
x	leftmost point of interval A	page 72
y	leftmost point of interval B	page 72
Y_e	percentage error of modeled values for R^*/R_a^*-tree	page 80

B. Approximation of PISA Model

The integral in Equation 6.14 on page 72 and the integral in Equation 6.17 on page 73 cannot be computed for every $d_a(x)$ and $d_b(y)$ analytically. For these cases we apply approximation methods to compute the integrals numerically. Appendix B presents approximation method for computing the integral in Equation 6.14. Equation 6.17 is approximated similarly. We present the computation of the integral over the gray shaded area in Figure B.1. Functions u and l describe this area:

$$\begin{aligned} u(x) &= \begin{cases} x+a & : \quad 0 \leq x < 1-a-b \\ 1-b & : \quad 1-a-b \leq x \leq 1-a \end{cases} \\ l(x) &= \begin{cases} 0 & : \quad 0 \leq x < b \\ x-b & : \quad b \leq x \leq 1-a \end{cases} \end{aligned}$$

Figure B.1 shows the area divided in $s_0 = 14$ slices. In computations presented in this thesis the x-interval $[0, 1-a)$ is divided in $s_0 = 200$ equidistant slices. Integrals over two rectangles are calculated for each interval. One sum of rectangles calculates an upper bound (U) of the real value and another sum of rectangles calculates an lower bound (L) of the real result:

$$\begin{aligned} U(a,b) &= \sum_{x \in \{y | y=(1-a)\frac{i}{s_0} \wedge i \in \{0,\cdots,s_0-1\}\}} \left(\Phi_{\mu,\sigma}\left(x+\frac{1}{s_0}, u_{a,b}\left(x+\frac{1}{s_0}\right)\right)\right. && \text{(B.1)} \\ &- \Phi_{\mu,\sigma}\left(x, u_{a,b}\left(x+\frac{1}{s_0}\right)\right) && \text{(B.2)} \\ &- \Phi_{\mu,\sigma}\left(x+\frac{1}{s_0}, l_{a,b}\left(x+\frac{1}{s_0}\right)\right) && \text{(B.3)} \\ &+ \left.\Phi_{\mu,\sigma}\left(x, l_{a,b}\left(x\right)\right)\right) \end{aligned}$$

$$\begin{aligned} L(a,b) &= \sum_{x \in \{y | y\ (1-a)\frac{i}{s_0} \wedge i \in \{0,\cdots,s_0-1\}\}} \left(\Phi_{\mu,\sigma}\left(x+\frac{1}{s_0}, u_{a,b}\left(x\right)\right)\right. && \text{(B.4)} \\ &- \Phi_{\mu,\sigma}\left(x, u_{a,b}\left(x\right)\right) && \text{(B.5)} \\ &- \Phi_{\mu,\sigma}\left(x+\frac{1}{s_0}, l_{a,b}\left(x+\frac{1}{s_0}\right)\right) && \text{(B.6)} \\ &+ \left.\Phi_{\mu,\sigma}\left(x, l_{a,b}\left(x+\frac{1}{s_0}\right)\right)\right) \end{aligned}$$

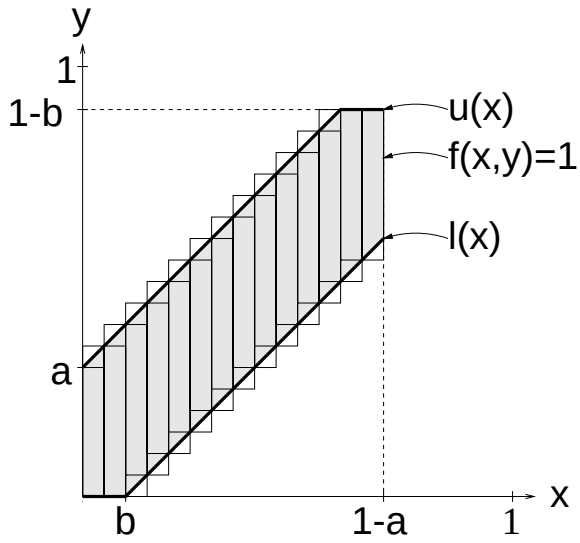

Figure B.1.: Approximation of the integral

The arithmetic average of these two sums is the approximation of the integral:

$$h_2(a,b) = \frac{1}{2}(U(a,b) + L(a,b)) \tag{B.7}$$

The equations shown here apply the distribution function $\Phi_{\mu,\sigma}(x)$ of normal distributed data as an example. Other distribution functions can be applied as well.

Zusammenfassung der Ergebnisse

In dieser Arbeit wird untersucht, welche Indexstrukturen Anfragen in typischen Data Warehouse Systemen effizient unterstützen. Indexstrukturen, seit mehr als zwanzig Jahren Forschungsgegenstand im Datenbankbereich, wurden in der Vergangenheit für transaktionsorientierte Systeme optimiert. Ein Kennzeichen dieser Systeme ist die effiziente Unterstützung von Einfüge-, Änderungs- und Löschoperationen auf einzelnen Datensätzen.

Typische Operationen in Data Warehouse Systemen sind dagegen komplexe Anfragen auf großen relativ statischen Datenmengen. Aufgrund dieser veränderten Anforderungen müssen Datenbankmanagementsysteme, die für Data Warehouses eingesetzt werden, andere Techniken nutzen, um komplexe Anfragen effizient zu unterstützen.

Zunächst wird ein Ansatz untersucht, der mit Hilfe eines gemischt ganzzahligen Optimierungsproblems eine *optimale* Indexstruktur berechnet. Da die Kosten für die Berechnung dieser *optimalen* Indexstruktur mit der Anzahl der zu indizierenden Datensätze exponentiell steigen, wird in anschließenden Teilen der Arbeit heuristischen Ansätzen nachgegangen, die mit der Größe der zu indizierenden Datensätze skalieren.

Ein Ansatz erweitert auf Bäumen basierende Indexstrukturen um aggregierte Daten in den inneren Knoten. Experimentell wird gezeigt, daß mit Hilfe der materialisierten Zwischenergebnisse in den inneren Knoten Bereichsanfragen auf aggregierten Daten wesentlich schneller bearbeitet werden.

Um das Leistungsverhalten von Indexstrukturen mit und ohne materialisierte Zwischenergebnisse zu untersuchen, wird das PISA Modell (*P*erformance of *I*ndex *S*tructures with and without *A*ggregated Data) entwickelt. In diesem Modell wird die Verteilung der Daten und die Verteilung der Anfragen berücksichtigt. Das PISA Modell wird an gleich-, schief- und normalverteilte Datensätze angepaßt. Experimentell wird gezeigt, daß das PISA Modell mit einer höheren Präzision als die bisher aus der Literatur bekannten Modelle arbeitet.

Die Leistung von Indexstrukturen hängt von unterschiedlichen Parametern ab. In dieser Arbeit werden zwei Techniken vorgestellt, die abhängig von einer bestimmten Menge von Parametern Indexstrukturen vergleichen. Mit Hilfe von Klassifikationsbäumen wird z. B. gezeigt, daß die Blockgröße die relative Leistung weniger beeinflußt als andere Parameter. Ein weiteres Ergebnis ist, daß Bitmap-Indexstrukturen von den Verbesserungen neuerer Sekundärspeicher stärker profitieren als heute übliche auf Bäumen basierende Indexstrukturen. Bitmap-Indexierungstechniken bieten noch ein großes Potential für weitere Leistungssteigerungen im Datenbankbereich.