Nr.: FIN-05-2012

Techniken zur forensischen Datenhaltung - Ausgewählte studentische Beiträge

Martin Schäler, Sandro Schulze, Alexander Grebhahn, Veit Köppen, Andreas Lübcke, Gunter Saake (Hrsg.)

Arbeitsgruppe Datenbanken

Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg

Technical report

Nr.: FIN-05-2012

Techniken zur forensischen Datenhaltung - Ausgewählte studentische Beiträge

Martin Schäler, Sandro Schulze, Alexander Grebhahn, Veit Köppen, Andreas Lübcke, Gunter Saake (Hrsg.)

Arbeitsgruppe Datenbanken

Technical report (Internet)
Elektronische Zeitschriftenreihe
der Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg
ISSN 1869-5078

Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg

Impressum (§ 5 TMG)

Herausgeber:
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Der Dekan

Verantwortlich für diese Ausgabe:
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Martin Schäler
Postfach 4120
39016 Magdeburg
E-Mail: schaeler@iti.cs.uni-magdeburg.de

http://www.cs.uni-magdeburg.de/Technical_reports.html

Technical report (Internet)
ISSN 1869-5078

Redaktionsschluss: 01.10.2012

Bezug: Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Dekanat

Vorwort

Die vorliegende Beitragssammlung enthält herausragende studentische Beiträge, die im Rahmen wissenschaftlicher Teamprojekte seit Sommersemester 2010 entstanden sind. Die Beiträge gaben wesentliche Impulse für die Forschung der Arbeitsgruppe Datenbanken an der Otto-von-Guericke Universität Magdeburg. Insbesondere für das vom BMBF geförderte Verbundprojekt „Digi-Dak“ (FKZ 13N10817) sind wichtige Beiträge geleistet worden. So basiert die erfolgreiche Teilnahme einer Studentengruppe am weltweiten SIGMOD Programming Contest 2012 (Platz 8) zu wesentlichen Teilen auf der hier geleisteten Grundlagenarbeit.

Die Beiträge sind chronologisch geordnet und behandeln im wesentlichen Arbeiten zum Thema Abfrageoptimierung mittels *mehr- bzw. hochdimensional Indexstrukturen*. Schwerpunkte in diesem weitläufigen Gebiet sind Klassifikation verschiedener Indexverfahren, experimentelle Evaluierung und die umfassende Erforschung von zusätzlichen Einflüssen durch Distanzmetriken und stochastischer Verteilung der Daten.
Weiterhin ist eine Übersichtsarbeit zum hochaktuellen Thema Sichere Datenbank Infrastrukturen mit Schwerpunkt *SQL-Injection* enthalten.

Magdeburg, im September 2012

die Herausgeber

Inhaltsverzeichnis

Vergleich von Indexverfahren für hochdimensionale Datenräume

David Broneske, Denis Dietze, Maik Lampe, Andreas Meier
Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg
{David.Broneske, Denis.Dietze, Maik.Lampe, Andreas2.Meier}@st.ovgu.de

***Zusammenfassung*—In den letzten zwei Jahrzehnten veränderten sich die Einsatzmöglichkeiten von Datenbanken wesentlich. Ursprünglich für die Speicherung von reinen Textinformationen vorgesehen, versuchen Datenbankmanagementsysteme heute auch den performanten Zugriff auf Videos und Bildern zu ermöglichen. Neben der fehlenden, einheitlichen Struktur verschiedener Bild- und Videodaten, stellt deren hohe Dimensionalität eines der größten Probleme dar. Infolge des *Fluchs der hohen Dimensionen (Curse of Dimensionality)* sind viele Zugriffsstrukturen bei einer hohen Anzahl an Dimensionen nicht mehr in der Lage Suchoperationen gegenüber dem Full Table Scan zu beschleunigen. Zur Lösung dieses Problems präsentieren wir in diesem Paper Verfahren, die auch bei hohen Dimensionen eine effiziente Suche gegenüber dem Full Table Scan ermöglichen. Dazu betrachten wir verschiedene Hash-/Baumverfahren, Klassifikatoren sowie optimierte sequentielle Suchalgorithmen und erläutern kurz deren Funktionsweise. Anschließend vergleichen wir diese Verfahren anhand von ausgewählten Evaluierungskriterien. Aufbauend auf den deraus resultierenden Vor- und Nachteilen der einzelnen Verfahren, geben wir abschließend Empfehlungen, welche Verfahren für weitere Forschungstätigkeiten betrachtet werden sollten.**

I. Einleitung

Diese Arbeit ist während der Forschung im Projekt "Digitale Fingerspuren", kurz Digi-Dak[1], entstanden. Innerhalb dieses Projekts liegt ein Schwerpunkt auf der Verwendung von Datenbanksystemen zur Verwaltung von Fingerabdrücken, die als hochdimensionale Bilddaten vorliegen. Dabei ist insbesondere eine effiziente Suche nach ähnlichen Datensätzen von Interesse. Aufgrund der großen, vorhandenen Datenmenge ist eine sequentielle Suche im gesamten Datenbestand, auch *Full Table Scan* genannt, nicht in einem zeitlich vertretbaren Rahmen realisierbar. Dies ist zum einen durch die *Zugriffslücke* begründet, die die stark abweichenden Geschwindigkeiten von für Datenbanksystemen verwendeten Speichersystemen beschreibt [9], als auch durch rechenaufwändige Vergleichsoperationen. Ein Full Table Scan muss bei einem großen Datenbestand daher ständig auf die relativ langsamen Festplatten zugreifen sowie jeden Datensatz untersuchen [6, 12]. Folglich sind für das Projekt vor allem auf hochdimensionale Daten spezialisierte Zugriffsverfahren interessant, die diese häufigen Lesezugriffe vermeiden und wesentlich weniger Datensätze betrachten müssen. Diese Verfahren sollen in dieser Arbeit insbesondere hinsichtlich ihrer Skalierbarkeit für hohe Dimensionen untersucht werden.

Die größte Herausforderung hochdimensionaler Datenräume stellt der Fluch der hohen Dimensionen dar [5, 8, 15, 22]. Dieser besagt, dass die Leistung vieler Zugriffsverfahren mit steigender Dimension signifikant abnimmt, was hauptsächlich durch die Konzentration der Daten im Raum verursacht wird. Dies führt zu einer sinkenden Selektivität, sodass Indexe ihre Stärken gegenüber einem Full Table Scan einbüßen und keine Geschwindigkeitsvorteile mehr ermöglichen. Insbesondere für Suchanfragen nach den ähnlichsten Datensätzen zu einem gegebenen Datensatz sind signifikante Geschwindigkeitseinbrüche die Folge [27]. Während sich der Fluch bei gängigen Verfahren schon ab einer Dimensionalität von 10 bemerkbar machen soll, erwähnen Weber et al. eine Dimensionsanzahl von 610 als obere, theoretische Schranke für clusterbasierte und partitionierende Verfahren [27].

Damit hochdimensionale Datenräume trotz des Fluchs der hohen Dimensionen effizient indexiert werden können, werden in der Fachliteratur viele, teils stark unterschiedliche Verfahren vorgestellt [2, 5, 18, 23, 24, 25]. Unser wesentlicher Beitrag ist, eine Übersicht über diese Verfahren sowie eine Einschätzung ihrer Stärken und Schwächen zu geben. Nachdem wir notwendige Grundlagen erläutert haben, stellen wir im Abschnitt III spezialisierte Hashverfahren vor. Im folgenden Abschnitt IV erläutern wir Baumverfahren, während sich Abschnitt V mit Verbesserungen des Full Table Scans beschäftigt. Im Abschnitt VI werden space-filling curves beschrieben. Als letzte Kategorie werden im Abschnitt VII Klassifikatoren beschrieben. Im Abschnitt VIII vergleichen wir die vorgestellten Verfahren miteinander und beurteilen daraufhin die Stärken und Schwächen der Verfahren. Den Abschluss bilden der Abschnitt IX, der verwandte Arbeiten vorstellt, sowie die Zusammenfassung im Abschnitt X.

II. Grundlagen

Für die Suche nach ähnlichen Datensätzen in hochdimensionalen Datenräumen sind vor allem *Nearest Neighbor Search* und *k-Nearest Neighbor Search* als Anfragearten von Interesse. Weiterhin kann in diesen Datenräumen zwischen *approximierenden* und *exakten* Suchverfahren unterschieden werden [1, 22].

[1]Digitale Fingerspuren, https://omen.cs.uni-magdeburg.de/digi-dak/cms/front_content.php?idart=40&lang=1, letzter Zugriff: 13.01.2011

Grundlage von (k-)Nearest Neighbor Search ist eine Distanzfunktion beziehungsweise eine Ähnlichkeitsfunktion, die in eine Distanzfunktion umgeformt werden kann. Diese beiden Funktionen definieren ein numerisches Maß, dass beschreibt, wie ähnlich zwei Datensätze zueinander sind. Die Distanzfunktion liefert dazu einen Wert zwischen 0 ($\equiv$ Identität) und ∞ ($\equiv$ keine Ähnlichkeit) zurück [5, 13].

Nearest Neighbor Search beschreibt eine Anfrageart, in der für das gegebene Anfrageobjekt der ähnlichste Datensatz der Datenbank gefunden werden soll. Demzufolge wird genau der Datensatz gesucht, für den die Distanzfunktion in Kombination mit dem Anfrageobjekt den kleinsten Wert zurückliefert.

K-Nearest Neighbor Search bezeichnet dagegen eine Anfrageart, in der für das gegebene Anfrageobjekt eine Menge gefunden werden soll, die nur die k ähnlichsten Datensätze der Datenbank beinhaltet. Folglich werden genau die k Datensätze gesucht, für die die Distanzfunktion in Kombination mit dem Anfrageobjekt die k kleinsten Werte zurückliefert. Nearest Neighbor Search stellt somit einen Sonderfall der k-Nearest Neighbor Search dar, indem k den Wert 1 besitzt.

In Abhängigkeit vom Anwendungsfall wird zwischen exakter und approximierender Suche unterschieden. Für eine exakte Suche ist es zwingend notwendig, die k ähnlichsten Datensätze zu finden. Approximierende Suchen fordern lediglich, k Datensätze zu finden, die dem Anfrageobjekt möglichst ähnlich sind. Im Unterschied zur exakten Suchen müssen also nicht die wirklich k ähnlichsten Datensätze gefunden werden, sondern es reicht k Datensätze zu finden, die dem Anfrageobjekt im Vergleich zum restlichen Datenbestand relativ ähnlich sind. Die Intention der approximierenden Suche ist es, Ergebnisqualität gegen eine höhere Verarbeitungsgeschwindigkeit einzutauschen [1, 22]. Interessant sind für das Digi-Dak Projekt vor allem Zugriffsstrukturen, die eine definierte Güte der Ergebnisqualität garantieren können. Dadurch ist es möglich, minimale Qualitätskriterien zu definieren, die in jedem Fall erfüllt werden [1].

III. Hashverfahren

Nachfolgend stellen wir Hashverfahren vor, die sich insbesondere für hochdimensionale Datenräume eignen. Für Datenräume niedriger Dimensionalität stellen Hashverfahren bereits etablierte Ansätze dar, um verschiedene Daten effektiv zu indexieren [7]. Normalerweise ist für diese gefordert, dass sogar ähnliche Datensätze einen vollkommen unterschiedlichen Hashwert erhalten. Werden jedoch die k nächsten Nachbarn gesucht, sind diese Funktionen unpassend, da der Hashwert keine Auskunft über die Ähnlichkeit der Datensätzen liefert. Jedoch haben Hashverfahren den entscheidenen Vorteil, dass sie dimensionsreduzierend arbeiten.

Im Unterschied zu klassischen Hashverfahren, werden für die Suche nach nächsten Nachbarn in hochdimensionalen Daten Funktionen gesucht, die wissentlich gleiche Hashwerte erzeugen und somit Kollisionen verursachen. Tellez und Chavez fassen solche Funktionen zu einer Funktionsfamilie $\mathcal{H}$ zusammen, wobei jede Funktion h $(P1, P2, r, cr)$-sensitiv ist [24]. Diese Sensitivität ist gegeben, wenn für alle Datensätze $p, q \in \mathcal{R}^d$ folgende Bedingungen erfüllt sind:

1) $wenn\ ||p-q|| < r,\ dann\ Pr[h(p) = h(q)] > P1$
2) $wenn\ ||p-q|| > cr,\ dann\ Pr[h(p) = h(q)] < P2$

Pr definiert dazu eine Wahrscheinlichkeitsfunktion, die Aussagen über die Gleichheit der Hashwerte von zwei betrachteten Datensätzen erlaubt. Bedingung 1 fordert demzufolge für zwei Datensätze p und q, die auf Basis einer Norm höchstens r Einheiten voneinander entfernt sind, dass die Wahrscheinlichkeit, dass beide Datensätze den selben Hashwert besitzen, größer als $P1$ sein soll. Sind sie jedoch weiter als cr Einheiten voneinander entfernt, besagt Bedingung 2, dass die Wahrscheinlichkeit gleicher Hashwerte kleiner als $P2$ sein soll. Intuitiv formuliert sollen diese beiden Sensitivitätsbedingungen gewährleisten, dass zueinander ähnliche Datensätze nach Möglichkeit den selben Hashwert erhalten und somit in das selbe Bucket einsortiert werden. Nicht ähnliche Datensätze sollen dagegen aufgrund ihrer verschiedenen Hashwerte in unterschiedliche Buckets einsortiert werden.

A. Locality Sensitive Hashing

Locality Sensitive Hashing stellt ein bekanntes Verfahren für hochdimensionale Datenräume dar, dass mit Hilfe einer $(P1, P2, r, cr)$-sensitiven Hashfamilie die nächsten Nachbarn zu einem gegebenen Datensatz findet [1, 10, 17, 24, 29]. Locality Sensitive Hashing basiert auf n Hashtabellen, wobei jeder Tabelle jeweils eine Hashfunktion zugeordnet ist. Jede dieser n Hashfunktionen entsteht durch die Konkatenation mehrerer zufällig ausgewählter Funktionen aus der Familie $\mathcal{H}$. Danach werden für alle Datensätze der Datenbank die Hashwerte für jede der n Tabellen mit der zugehörigen Funktion berechnet und die Datensätze in die jeweiligen Buckets einsortiert.

Die Suche nach den zum Anfrageobjekt nächsten Nachbarn erfolgt in zwei Schritten. Zuerst werden die n Hashwerte des Anfrageobjekts durch Nutzung der zu jeder Hashtabelle zugeordneten Funktion berechnet. Dadurch wird ermittelt, in welchem Bucket jeder Hashtabelle sich das Anfrageobjekt befindet. Durch Vereinigung aller Datensätze dieser Buckets aus den n Hashtabellen wird die Kandidatenmenge gebildet. Im anschließenden zweiten Schritt wird für jeden Datensatz der Kandidatenmenge geprüft, ob die Distanz zum Anfrageobjekt kleiner als der bisher gefundene k-nächste Nachbar ist. Ist dies der Fall, wird der Datensatz in die Ergebnismenge aufgenommen und der am weitesten entfernte Datensatz der Ergebnismenge entfernt. Nachdem alle Datensätze der Kandidatenmenge in dieser Weise überprüft wurden, enthält die Ergebnismenge die k-nächsten Nachbarn.

Dieser Ansatz funktioniert in der Theorie sehr gut, scheitert jedoch in der Praxis häufig an der Definition konkreter $(P1, P2, r, cr)$-sensitiver Hashfunktionen. Weiterhin sind viele dieser Funktionen sehr komplex und auf bestimmte Datensätze optimiert, sodass ihre Allgemeingültigkeit nicht garantiert ist.

B. Locality Sensitive Hashing - ausgewählte Funktionen

Tellez und Chavez umgehen die Definition von

$(P1, P2, r, cr)$-sensitiven Hashfunktionen durch eine andere Vorgehensweise [24]. Anstatt direkt eine Funktionsfamilie $\mathcal{H}$ zu definieren, die die Lokalität der betrachteten Datensätze berücksichtigt, konstruieren die Autoren direkt Hashwerte, die Ähnlichkeiten erhalten können. Dazu werden die folgenden drei Schritte ausgeführt.

1) Bestimme zufällig n Datensätze, Prototypen genannt, aus dem gesamten Datenbestand.
2) Berechne die Distanz jedes Datensatzes zu den Prototypen.
3) Ordne für jeden Datensatz die Prototypen aufsteigend nach der Distanz und definiere diese Permutation der Prototypen als Hashwert für den Datensatz.

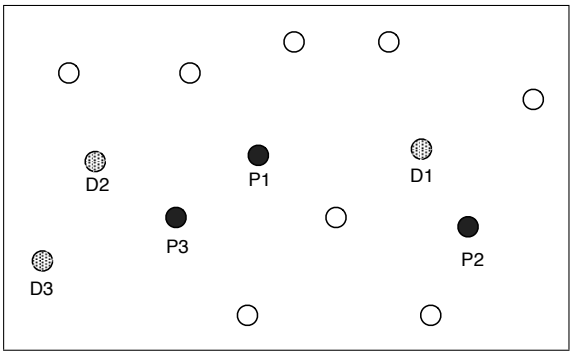


Abbildung 1. Beispiel der Permutationen mit P1-P3 als Prototypen

Zur Verdeutlichung ist in Abbildung 1 ein Beispiel dargestellt, bei dem diese Hashwerte exemplarisch für die markierten Punkte berechnet wurden. Die Datensätze P1, P2 und P3 wurden zufällig als Prototypen unter den Datensätzen ausgewählt. Folglich ergeben sich nach der Distanzberechnung und dem Sortieren nach aufsteigender Distanz für die markierten Punkte folgende Hashwerte.

- $h(D1) = (P2, P1, P3)$
- $h(D2) = (P3, P1, P2)$
- $h(D3) = (P3, P1, P2)$
- $h(P1) = (P1, P3, P2)$
- $h(P2) = (P2, P1, P3)$
- $h(P3) = (P3, P1, P2)$

Diese Werte können nun in einer Hashtabelle zur Indizierung verwendet werden. Dabei wird, wie am Beispiel der Datensätze D2, D3 und P3 ersichtlich ist, aufgrund der gleichen Hashwert die Ähnlichkeit zwischen den Datensätzen erhalten. Hierbei ist es entscheidend, wie viele Prototypen für die Daten gewählt werden. Bei einer zu geringen Anzahl an Prototypen wird die Ähnlichkeit nicht hinreichend erhalten. Eine zu hohe Anzahl an Prototypen belegt mehr Speicherplatz und bedeutet zusätzlichen Rechenaufwand bei der Berechnung des Hashwertes [24]. Durch diese Herangehensweise mit Permutationen verlagert sich somit die Herausforderung der Definition passender Hashfunktionen zur besten Anzahl an Prototypen. Allerdings geben Tellez und Chavez kein Verfahren an, um diese Anzahl bestimmen zu können.

Als Alternative entwickelten wir den Ansatz, Histogramme als ähnlichkeitsbeschreibende Merkmale zu nutzen, der nach unserem Kenntnisstand in dieser Form noch nicht in anderen Publikationen erwähnt wurde. Die Grundidee ist, dass ähnliche Bilder auch zueinander ähnliche Histogramme ihrer Grauwerte besitzen. Neben der Möglichkeit über das gesamte Bild ein Grauwerthistogramm zu berechnen, kann das Bild auch vorher gekachelt werden. Durch die Unterteilung in einzelne Kacheln, für die jeweils Grauwerthistogramme bestimmt werden, würden somit lokale Ähnlichkeiten in einzelnen Bildteilen berücksichtigt werden. Dadurch kann ein Bild auf Kosten eines höheren Speicherbedarfs exakter beschrieben werden, was für die Ähnlichkeitssuche von Vorteil ist. Auf diese Weise ließen sich insbesondere Exact-Match Anfragen zur Suche nach der Existenz eines Datensatzes in der Datenbank sehr schnell durchführen, wenn die Histogramme als Hashwerte genutzt werden würden. Aber auch für die Suche nach den k-nächsten Nachbarn ließe sich das Verfahren in Kombination mit einer passenden Zugriffsstruktur eventuell nutzen.

Dieser histogrammbasierte Ansatz erfordert einige Vorverarbeitungen. So müssen Bilder den selben Farbraum besitzen, um ein Vergleich der Histogramme zu ermöglichen. Eine Möglichkeit ist, alle Bilder in die Grauwertdarstellung zu konvertieren. Weiterhin ist eine Skalierung aller Bilder zu einer einheitlichen Größe sinnvoll, da sich so die einzelnen Farbverteilungen direkt miteinander vergleichen lassen. Als Alternative wäre auch eine relative Darstellung der einzelnen möglichen Farbausprägungen denkbar, wodurch das Verfahren bildgrößeninvariant wird. Allerdings steigt der nötige Rechenaufwand, da Bilder weniger spezifisch beschrieben werden und somit bei der Suche mehr Kandidaten untersucht werden müssen. Wesentlich ist auch die Anwendung von Bildnormalisierungen, wie Kontrast- oder Helligkeitsanpassungen, um aufnahmebedingte Schwankungen der Histogramme auszugleichen.

Xiang et al. stellen in ihrer Arbeit ein ähnliches Verfahren vor, dass auf Farbhistogrammen der Rot-, Grün- und Blauanteile sowie auf einer Grauwertdarstellung der Bilder basiert [28]. Die Autoren berechnen diese Histogramme über das komplette Bild, was den Vorteil schafft, dass ihr Verfahren resistent gegen Drehungen und Skalierungen ist. Ferner wird durch die Anwendung von Filtern das Verfahren auch robuster gegen Bildmanipulationen wie Verzerrungen und Bildrauschen. Die Histogramme werden anschließend als Hashwerte genutzt, sodass sich dadurch die LSH-Funktionalität ergibt.

Hashverfahren haben, wie in den vorgestellten Ansätzen gezeigt wurde, den Vorteil, dass sie sehr gut mit der Dimensionalität der Daten skalieren. Aufgrund der Abbildung der hochdimensionalen Daten auf wenige Dimensionen besitzen sie eine gewisse Resistenz gegen den Fluch der hohen Dimensionen und skalieren somit unter Umständen besser als einfache Bäumverfahren. Ein wesentlicher Nachteil besteht jedoch darin, dass die meisten Hashverfahren für hochdimensionale Datenräume in der Regel nur approximativ arbeiten und somit nicht garantieren können, die wirklich k-nächsten Nachbarn eines Anfrageobjekts zu finden. Exact-Match Anfragen lassen sich dagegen sehr effizient durchführen. Ein weiterer Makel von Hashverfahren wird durch die Verteilung der Daten in

der Datenbank sichtbar. Bei gleichverteilten Daten werden die Buckets in etwa gleich groß. Hingegen können einige Buckets bei nicht gleichverteilten Daten wesentlich größer als andere Buckets werden. Zudem besteht das Problem, dass Buckets, die sehr wenige Datensätze enthalten bei zu großen Werten von k gar keine k-nächsten Nachbarn liefern können. Aufgrund der fehlenden Sortierung von Buckets kann zudem nicht einfach das nächste Bucket im Speicher selektiert werden, da dieses keine ähnlichen Datensätze enthalten muss. Somit muss generell ein Kompromiss zwischen der Anzahl von Buckets und der maximal enthaltenen Datensätze gefunden werden.

IV. Baumverfahren

Bäume stellen in Datenbanksystemen einige der wichtigsten Zugriffsstrukturen dar, um den Zugriff auf bestimmte Datensätze zu optimieren und zu beschleunigen [7]. Sie versuchen aufgrund der hierarchischen Einteilung des Datenraums den Aufwand der Suche nach Datensätzen auf eine Komplexität von $O(\log(N))$ zu minimieren.

Für Baumverfahren existieren zwei wesentliche Ansätze, den Raum aufzuteilen [5, 22]. Es gibt so genannte *Space-Partitioning Methods*, die den Raum aufgrund vorbestimmter oder berechneter Grenzen einteilen. Dabei kann es leicht vorkommen, dass einige Bereiche sehr dicht und andere eher weniger dicht besetzt sind, was die Selektivität einschränkt. Eine andere Herangehensweise verfolgen *Data-Partitioning Methods*, die den Raum aufgrund der Position der Datenpunkte hierarchisch einteilen. Folglich werden auch nur Teilräume geteilt, in denen sich Datensätze befinden. Dies wird unter anderem durch MBRs (*Minimum Bounding Rectangles*) erreicht, welche im R-Baum Verwendung finden [5]. Nachfolgend werden neben dem R-Baum weitere Baumverfahren zur Suche in hochdimensionalen Datenräumen erläutert.

A. R-Baum

Der R-Baum ist die direkte Erweiterung des B-Baums auf k-Dimensionen. Er besteht aus Zwischenknoten und Blattknoten, die die Datenobjekte enthalten. Ein Zwischenknoten korrespondiert hierbei mit mehreren MBRs, die ihre untergeordneten MBRs vollständig umschließen [21]; der Aufbau ist in Abbildung 2 und 3 dargestellt. Ein MBR wird aus 2 Punkten im Raum konstruiert, die als gegenüberliegende Punkte des (Hyper-) Rechtecks anzusehen sind. Sie spannen somit die Diagonale des Rechtecks auf.

Werden die k-nächsten Nachbarn für ein Anfrageobjekt im Datenraum gesucht, so werden angrenzende Rechtecke bis hin zu den Blattknoten verfolgt. Jeder in diesen Rechtecken enthaltene Punkt wird bezüglich seiner Distanz zum Anfrageobjekt untersucht und, sofern er zu den momentan k-nächsten Nachbarn gehört, in einer Ergebnismenge gespeichert. Punkte, die durch das Einfügen von neuen Punkten nicht mehr zu den k-nächsten Nachbarn gehören, werden aus der Ergebnismenge entfernt. Somit enthält die Ergebnismenge nach Untersuchung aller relevanten Punkte die k-nächsten Nachbarn.

Der R-Baum existiert in verschiedenen Varianten, zum Beispiel als R^+- oder R*-Baum [5, 21, 7], die Daten nur in

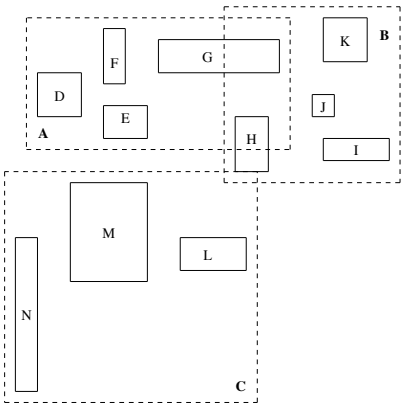


Abbildung 2. Raumaufteilung des R-Baums, übernommen aus [21]

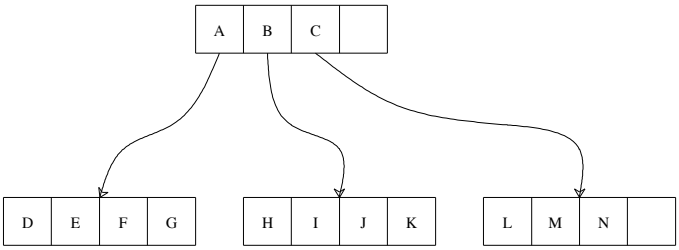


Abbildung 3. Beispiel eines R-Baums, übernommen aus [21]

den Blattknoten speichern oder versuchen, Überlappungen von MBRs zu verhindern. Alle Varianten haben bei bestimmten Datenverteilungen und -dimensionen Vor- und Nachteile, sodass es keine Variante gibt, die als allgemein besser angesehen werden kann. Obwohl der R-Baum ursprünglich zur Indexierung des 2-dimensionalem euklidischen Raumes entworfen wurde [11], wird er heute ebenfalls für hochdimensionale Datenräume verwendet. Allerdings stellt sich insbesondere in hohen Dimensionen die Behandlung von Überlappungen der Rechtecke als Hauptproblem des R-Baums heraus. Als Lösungansatz wurde eine neue Variante des R-Baumes entwickelt, die als X-Baum bezeichnet und nachfolgend erläutert wird.

B. X-Baum

Der von Berchtold et al. vorgeschlagene X-Baum basiert auf dem R*-Baum und erweitert ihn dahingehend, dass die Anzahl von Überlappungen der MBRs weiter reduziert wird [3]. Vor allem in höheren Dimensionen steigt die Überlappung der Rechtecke, was bei Suchanfragen zu einer größeren zu untersuchenden Anzahl von Knoten führt und damit die Effizienz des Baumes verschlechtert [5]. Diese Überlappungen entstehen, wenn Datenobjekte eine große Ausdehnung besitzen oder wenn Knoten aufgespaltet werden müssen.

Der X-Baum versucht Aufspaltungen von Knoten genau dann zu verhindern, wenn diese zu einem hohen Grad an Überlappungen von MBRs führen würden. Stattdessen wird der betrachtete Knoten zu einem Superknoten deklariert, der doppelt so groß wie ein normaler Knoten ist. Dieses Verfahren kann beliebig oft auf Superknoten durchgeführt werden, sodass

diese ein beliebiges Vielfaches der Größe normaler Knoten einnehmen können. Der wesentliche Unterschied zwischen normalen Knoten und Superknoten ist, dass normale Knoten weiter aufgespaltet werden können, um hierarchisch durchsucht zu werden. Superknoten speichern dagegen lediglich Einträge und lassen sich nicht weiter aufspalten. Somit müssen Superknoten bei Anfragen linear durchsucht werden. Dies ist zwar rechenaufwändiger als die klassische Suche im R-Baum, bietet im Gegensatz zur Suche bei vielen Überlappungen aber eine höhere Effizienz.

Bei Suchanfragen wird der X-Baum wie ein normaler R-Baum durchsucht. Wird dabei ein Superknoten erreicht, so wird dieser linear durchsucht, anstatt wie normale Knoten hierarchisch durchsucht zu werden. Das Einfügen von Einträgen in den X-Baum verhält sich wie beim R*-Baum, indem ein geeignetes Blatt gesucht und das Datenobjekt eingefügt wird. Wird ein Knoten zu groß, so wird zuerst eine Aufspaltung in Betracht gezogen. Dazu wird der Spaltungsalgorithmus des R*-Baums ausgeführt und das mögliche Ergebnis bewertet. Falls zu viele Überlappungen entstehen würden, wird ein Algorithmus eingesetzt, der versucht den Grad der Überlappung zu minimieren. Sollte auch hier keine den Anforderungen genügende Spaltung gefunden werden, wird diese nicht ausgeführt und der Knoten zu einem Superknoten deklariert.

Laut Katayama und Satoh haben die MBRs der bisherigen Baumverfahren jedoch einen entscheidenden Nachteil [14]. Sie teilen den Raum in Bereiche mit kleinem Volumen, aber relativ großem Druchmesser. Da der Durchmesser einen großen Einfluss auf die Suche nach den nächsten Nachbarn hat, kann es mehrmals vorkommen, dass eine Region aufgrund ihrer breiten Ausdehnung in die Suche mit einbezogen wird, obwohl sie keine nächsten Nachbarn enthält. Katayama und Satoh führen deshalb den SR-Baum ein, der diesem Phänomen entgegen wirken soll [14].

C. SR-Baum

Der SR-Baum versucht die Suche nach nächsten Nachbarn zu beschleunigen, indem er die die Daten umschließenden Bereiche mit minimalem Volumen und minimalem Durchmesser erstellt. Hierfür werden MBRs mit *Minimum Bounding Spheres* kombiniert, wie dies in Abbildung 4 zu sehen ist. Die Minimum Bounding Spheres werden dabei durch den Mittelpunkt und den Durchmesser definiert. Der Schnitt dieser beiden Körper ergibt laut Katayama und Satoh einen Körper mit minimalem Durchmesser und minimalem Volumen, was die Vorteile beider Raumeinteilungsverfahren vereint. Zudem wird durch die kompakte Speicherung der Sphere, die durch den Mittelpunkt und den Radius beschrieben werden kann, der zusätzlich benötigte Speicherplatz gering gehalten.

Bei der Suche nach den nächsten Nachbarn wird der Baum per Tiefensuche durchsucht, wobei die Knoten bevorzugt werden, die die kleinste Distanz zum Anfragepunkt haben. Diese Suche besteht aus zwei Etappen: Zuerst wird eine Menge von Kandidaten bestimmt, die als nächste Nachbarn in Frage kommen. Danach wird diese Menge noch einmal verfeinert, indem jedes Blatt besucht wird, dessen Region einen der

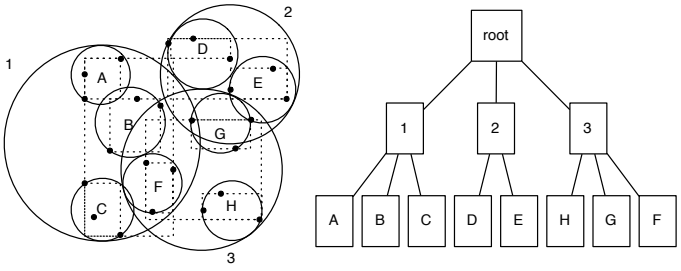


Abbildung 4. SR-Baum und dessen Minimum Bounding Rectangles und Minimum Bounding Spheres, übernommen aus [21]

Kandidaten überlappt. Der Algorithmus terminiert, wenn keine weiteren Blätter mehr zu besuchen sind.

Gegenüber dem R-Baum wird hierbei die Suche nach den nächsten Nachbarn durch die kleiner gewordenen Regionen effizienter [21]. Dafür benötigt der SR-Baum aber komplexere Algorithmen beim Einfügen, zum Beispiel zur Bestimmung der Spheres, und mehr Speicher. Diese Speicherkosten führen jedoch oft auch zu zusätzlichen Festplattenzugriffen, da der ganze Baum nicht immer im Arbeitsspeicher gehalten werden kann. Dadurch kann die Performance des SR-Baums stark beeinträchtigt werden. Eine Zugriffsstruktur, die den zusätzlichen Festplattenzugriffen entgegen wirkt, ist der A-Baum [20].

D. A-Baum

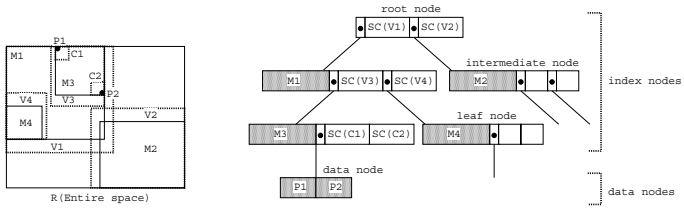
Der A-Baum wurde von Sakurai et al. entwickelt und stellt eine Symbiose aus R-Baum, SR-Baum und dem VA-File, welches im Abschnitt V noch näher betrachtet wird, dar [20]. Grundlage der Entwicklung waren folgende Beobachtungen, die bei der Nutzung des R-Baums, des SR-Baums und des VA-Files getroffen wurden.

1) Für Realdaten und synthetische, geclusterte Daten sind Baumstrukturen gegenüber dem VA-File im Vorteil, da Bäume für hochdimensionale Datenräume flexibel auf die Verteilung vorhandener Daten reagieren.
2) Jedoch ist die benötigte Größe eines Knotens dieser Bäume direkt proportional zur Dimensionsanzahl. Dies ist darin begründet, dass die Koordinaten der Minimum Bounding Rectangle und Spheres mit der Dimensionszahl umfangreicher werden.
3) Weiterhin wird die Selektivität des Baumes, also das möglichst frühe Verwerfen von nicht vielversprechenden Knoten, durch die Minimum Bounding Spheres weniger beeinflusst als angenommen.

Aufgrund dieser Beobachtungen entwickelten Sakurai et al. den A-Baum, der lediglich auf MBRs basiert und zusätzlich eine Approximation innerhalb des MBRs einführt. Für die Approximation nutzen die Autoren so genannte *Virtual Bounding Rectangles*. Diese haben die Aufgabe, darunter liegende MBRs oder Datenpunkte zu approximieren. Dabei werden die Diagonalen des aktuellen MBRs und dessen Kindknoten ins Verhältnis gesetzt und das Ergebnis quantisiert, welches als Bit-String dargestellt wird. Weiterhin ist es wichtig, dass Virtual Bounding Rectangles immer mindestens so groß wie

die in ihnen enthaltenen MBRs sind. Dadurch wird gewährleistet, dass relevante Datenpunkte aufgrund einer zu kleinen Approximation des umgebenden Rechtecks, nicht ignoriert werden.

In Abbildung 5 ist ein Beispiel für einen A-Baum dargestellt. Der Wurzelknoten enthält nur die Virtual Bounding Rectangles V1 und V2 seiner Kindknoten, die den gesamten Datenraum approximieren. Alle anderen *index nodes* speichern dagegen ihr eigenes MBR sowie die Virtual Bounding Rectangles ihrer Kindknoten. So speichert der erste Kindknoten der Wurzel sein Minimum Bounding Rectangle M1 und die Virtual Bounding Rectangles V3 und V4 seiner Kindknoten. Die Datenpunkte, wie etwa P1 und P2, werden dagegen direkt in einer Art Blattknoten abgelegt.



(a)VBR and MBR (b)Tree structure

Abbildung 5. Raumaufteilung und korrespondierender A-Baum, übernommen aus [21]

Der Vorteil dieser Baumstruktur ist, dass die Approximationsdaten der Virtual Bounding Rectangles platzsparender als echte Minimum Bounding Rectangles sind und somit eine größere Anzahl von Bounding Rectangles gleichzeitig im Arbeitsspeicher gehalten werden. Weiterhin können bei einer Suchanfrage sowohl das MBR des aktuellen Knotens als auch die Virtual Bounding Rectangles seiner Kindknoten betrachtet werden. Dies ermöglicht ein frühes Ausdünnen des Baumes bei der Suche, da Kindknoten nicht zwangsläufig betrachtet werden müssen.

V. Optimierte Sequentielle Suche

Obwohl die vorherigen Verfahren einen sehr performanten Zugriff auf Daten in mehrdimensionalen Räumen gewährleisten, stoßen sie bei sehr hohen Dimensionen an ihre Grenzen. Weber et al. nennen die Zahl 610 als oberste Dimensionsgrenze für Cluster- und Partitionsverfahren [27]. Ab dieser Dimensionszahl soll ihrer Ansicht nach jede Suche mit einer dieser gewählten Indexstrukturen zu einer sequentiellen Suche ausarten.

Weber et al. untersuchen in ihrer Arbeit verschiedene Verfahren hinsichtlich ihrer Skalierung mit der Dimensionalität der Daten. Hierbei gehen die Autoren insbesondere auf Bäume, die lediglich Minimum Bounding Rectangles nutzen, ein. Nach dem zugrundeliegenden Kostenmodell unterliegen diese Verfahren den Autoren zufolge bereits ab einer Dimensionen von 20 dem Fluch der hohen Dimensionen. Somit muss bei der Nearest Neighbor Suche ab dieser Dimensionszahl fast der gesamte Baum durchsucht werden. Aber auch Baumverfahren, die Minimum Bounding Spheres nutzen, verlieren laut Weber et al. ihren Effizienzvorteil gegenüber einer sequentiellen Suche ab einer Dimensionszahl von 26. Weiterhin leiten die Autoren aus ihrem Kostenmodell ab, dass auch clusterbasierte und partitionierende Indexverfahren in der Praxis schon bei wesentlich geringeren Dimensionen als 610 dem Fluch der hohen Dimensionen unterliegen. Folglich sollte bei hohen Dimensionen der Fokus mehr auf der Optimierung der sequentiellen Suche liegen, anstatt auf der Nutzung von Indexverfahren. Als Verbesserung der sequentiellen Suche stellen Weber et al. daher das *Vector Approximation File* (VA-File) vor [27].

A. VA-File

Anstatt wie R-Bäume den Datenraum hierarchisch aufzuteilen, partitioniert das VA-File den Raum in hyperrechteckige Zellen, die durch einen eindeutigen Bit-String identifiziert werden. Dafür wird für jede Dimension i eine kleine Anzahl an Bits b_i, die meist zwischen 4 und 6 beträgt, genutzt. Der Wertebereich der Dimension wird anschließend in 2^{b_i} Partitionen unterteilt, wobei die entstandenen Partitionen nicht gleich groß sondern möglichst gleichmäßig befüllt sein sollen. Sei b die Summe aller b_i, so wird der Raum in 2^b hyperrechteckige Zellen unterteilt. Diese Zellen bleiben bei allen weiteren Update-, Einfüge- und Löschoperationen konstant und werden benutzt, um das Approximationfile aufzubauen.

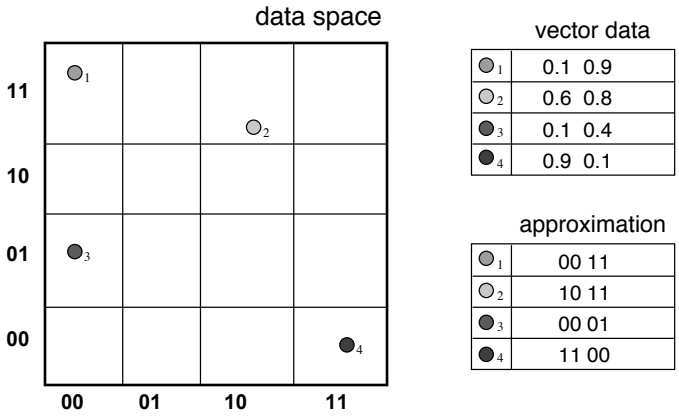


Abbildung 6. Aufteilung des Raumes durch VA-File, rechts oben die Punkte mit Vektordaten, darunter das Approximationfile, übernommen aus [21]

Im Approximationfile wird für jeden Punkt der Bit-String der Zelle gespeichert, in die er eingeordnet wird. Diese Approximation führt zu einem geringen Speicherbedarf, sodass das gesamte Approximationfile im Arbeitsspeicher gehalten werden kann. Bei der Suche nach den nächsten Nachbarn wird das gesamte Approximationfile durchsucht und aufgrund der rechteckigen Zellen der Abstand zum Anfragepunkt berechnet. Für die aufgrund dieser Approximation gefundenen, besten Kandidaten wird anschließend noch einmal die Distanz zwischen dem Anfragepunkt und den exakten Vektoren der Kandidaten bestimmt. Durch die Sortierung dieser Distanzwerte werden abschließend die k-nächsten Nachbarn bestimmt.

Einen großen Einfluss auf die Performance des VA-Files hat hierbei die Selektivität. Wenn trotz Approximation zu viele

Kandidaten für die anschließende exakte Distanzberechnung verbleiben, hebt sich der Geschwindigkeitsgewinn durch das kleinere Approximationfile auf. Laut Weber et al. verbessert sich jedoch die Selektivität mit steigender Anzahl an Datensätzen, was durch eine empirische Studie belegt wird [27].

VI. Space-filling Curves

Bei Ähnlichkeitsanfragen stellt das Fehlen einer totalen Ordnung in mehrdimensionalen Datenräumen eine der wesentlichen Herausforderungen für den Zugriff auf hochdimensionale Daten dar. Verfahren wie der B-Baum nutzen die Existenz von totalen Ordnungen im eindimensionalen Raum, sodass zum Beispiel über eine $\leq$-Relation ähnliche Datensätze effizient gesucht werden können. Im mehrdimensionalen Raum existieren dagegen totale Ordnungen und somit solche Relationen nicht, sodass Verfahren wie der B-Baum auch nicht angewendet werden können. Böhm et al., Gaede und Günther, Valle et al. sowie Tao et al. schlagen deshalb die Nutzung von space-filling curves zur Reduktion hochdimensionaler Räume auf eindimensionale Räume vor [4, 7, 25, 23].

A. Z-Ordering

Die Idee hinter space-filling curves ist, einen hochdimensionalen Raum durch eine Funktion zu beschreiben, die nur von einer Variable abhängig ist. Zusätzlich versucht diese Funktion zu gewährleisten, dass Datenpunkte, die im Raum nah beieinander liegen, es auch auf dem Graphen der Funktion tun. Dazu teilt die Funktion den Raum in kleinere Teilräume wie etwa Quadrate auf. Durch rekursive Definition der Funktion kann die abbildbare Dimension des Raumes erhöht werden, sodass die Teilräume durch Hyperwürfel dargestellt werden.

Gaede und Günther stellen in ihrer Arbeit verschiedene space-filling curves vor [7]. Basierend auf verschiedenen Experimenten führen sie die Hilbert-Kurve und das z-Ordering als praktikabelste Methoden für den Zugriff auf hochdimensionale Daten an. Das z-Ordering, welches auch als *Z-Kurve* bezeichnet wird, ist in Abbildung 7 für einen zweidimensionalen Raum dargestellt.

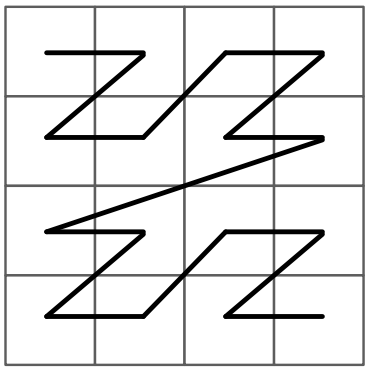


Abbildung 7. Struktur einer Z-Kurve im zweidimensionalen Raum

Infolge der Abhängigkeit der beschreibenden Funktion von lediglich einer Variable besitzt die Z-Kurve eine Dimensionalität von eins. Dadurch ist es möglich, die Z-Kurve und somit die durch sie definierten Teilräume mit gängigen eindimensionalen Zugriffsstrukturen zu indexieren. Bayer verallgemeinerte dazu den B-Baum unter Nutzung der Z-Kurve zum *Universal B-Baum* (UB-Baum) [2].

Sollen für ein Anfrageobjekt die k-nächsten Nachbarn bestimmt werden, wird das Anfrageobjekt zuerst über die Z-Kurve in seinen Teilraum abgebildet. Innerhalb dieses Teilraums kann nach den k-nächsten Nachbarn gesucht werden. Werden dort die k-nächsten Nachbarn nicht gefunden, können unter Ausnutzung der durch die Z-Kurve existierenden totalen Ordnung die Teilräume um das Anfrageobjekt herum untersucht werden. Diese Teilräume befinden sich meistens auch auf der Z-Kurve in der Nähe des Anfrageobjekts. Allerdings existieren, wie in Abbildung 7 zu sehen ist, auch Teilräume für die das nicht gilt. So besitzt der Teilraum in der zweiten Reihe der vierten Spalte laut Z-Kurve andere benachbarte Teilräume als es eigentlich der Fall ist. Diese Sonderfälle müssen daher gesondert behandelt werden, um eine exakte Suche zu erreichen.

Tao et al. benutzen in ihrer Arbeit *LSB-Bäume*, die mehrere Zugriffsverfahren in sich vereinen [23]. So werden zuerst unter Nutzung von LSH mehrere Datenpunkte in die selben Buckets einsortiert. Anschließend werden die Buckets auf eine Z-Kurve abgebildet und per B-Baum indexiert, was von den Autoren aber nicht explizit als UB-Baum bezeichnet wird.

Durch die Nutzung von LSH sind die LSB-Bäume in der Regel approximierend, wenngleich durch die Kombination mehrerer LSB-Bäume zu *LSB-Wäldern* eine nahezu perfekte Ergebnisgüte erreicht werden kann [23]. Allerdings erwähnen Tao et al. auch rigorous-LSH, dass eine Modifikation des LSH-Verfahrens darstellt. Im Gegensatz zum klassischen LSH, kann diese Variante auch exakte Anfragen auf Kosten der Geschwindigkeit durchführen.

Tao et al. führen in ihrer Arbeit eine umfangreiche Evaluation durch [23]. Der einfache LSB-Baum ist dabei aufgrund seiner starken Approximation mindestens immer eine Zehnerpotenz schneller als alle anderen untersuchten Verfahren. LSB-Wälder führen zwar zu erhöhten I/O-Kosten, liefern dafür aber auch wesentlich genauere Ergebnisse. Dabei ist zu bemerken, dass LSB-Wälder in dieser Evaluation das zweitschnellste Verfahren darstellen und im Vergleich zum Full Table Scan eine etwa zehnfach höhere Geschwindigkeit bieten. Zugleich skalieren die LSB-Verfahren auch teilweise stark sublinear mit der Dimensionalität und der Größe des Datenbestandes. So führt eine Erhöhung der Dimensionalität von 25 auf 100 lediglich zu einer 2,5-fach höheren I/O-Last. Für höhere Dimensionen wurden allerdings keine Ergebnisse angegebenen, sodass nicht davon ausgegangen werden kann, dass diese Skalierungsfähigkeit bei noch höheren Dimensionen erhalten bleibt.

VII. Klassifikatoren

Klassifikatoren stellen eine weitere Möglichkeit dar, die für eine Anfrage zu betrachtende Anzahl an Datensätzen zu reduzieren und somit eine höhere Geschwindigkeit zu

erzielen. Dazu werden sämtliche Datensätze jeweils einer von n zuvor definierten Klassen zugewiesen. Dabei wird indirekt angenommen, dass ähnliche Datensätze derselben Klasse zugeordnet werden [16]. Infolge dieser Annahme kann durch die Bestimmung der Klasse des Anfrageobjekts die Anzahl der Kandidaten stark eingeschränkt werden. Anschließend werden nur noch die Datensätze näher untersucht, die derselben Klasse wie das Anfrageobjekt angehören. In diesem Abschnitt werden zwei Verfahren erläutert, die mithilfe von Klassifikatoren die Suche in großen Bilddatenbanken beschleunigen. Dabei liegt der Fokus vor allem auf Besonderheiten der Verfahren, die für das Digi-Dak Projekt von besonderem Interesse sind.

A. K-Nearest Neighbor Klassifikation

Wang et al. nutzen einen K-Nearest Neighbor Klassifikator zur Bestimmung der Klasse eines Fingerabdrucks, um dadurch die Suche nach zu diesem ähnlichen Fingerabdrücken in einer Datenbank zu beschleunigen [26]. Die Grundlage dieser Arbeit ist die Unterteilung von Fingerabdrücken in vier verschiedenen Klassen, die charakteristische Muster beschreiben.

Die Aufgabe des Klassifikators ist es, aus extrahierten Features auf die Klasse des Fingerabdrucks zu schließen, der als Anfrageobjekt genutzt wird. Dazu muss der Klassifikator vor seinem Einsatz trainiert werden. Im Gegensatz zu Clustering-Verfahren nutzen Klassifikatoren während ihrer Trainingsphase vom Nutzer vorgegebene Informationen, wie etwa die Klassenzugehörigkeit von Trainingsdaten. Dadurch wird das Ziel verfolgt, dass Klassifikatoren anhand von Merkmalen der Trainingsdaten lernen, auf die zugehörige Klasse zu schließen. Clustering-Verfahren versuchen dagegen lediglich eine Struktur der Daten, zum Beispiel Konzentrationen in bestimmten Regionen des Datenraumes, zu bestimmen.

Wang et al. trainieren ihren Klassifikator mit Hilfe von 500 Fingerabdrücken, von denen sie selbst sowohl die Merkmale als auch die Klassenzugehörigkeiten bestimmt haben. Sowohl für das Training als auch für die Klassifikation von neuen Fingerabdrücken führen sie mehrere Verarbeitungsschritte durch.

1) Das Bild wird normiert, per Gauß-Filter geglättet und anschließend das Bildhistogramm angepasst.
2) Durch die Analyse von 8x8 Pixels großen Blöcken werden Features extrahiert und diese in einem 48-dimensionalen Featurevektor gespeichert.

Diese Feature-Vektoren werden per Clustering als Datenpunkte in verschiedene Cluster gruppiert. Die Autoren bestimmten dazu in eigenen Experimenten eine Clusteranzahl von neun als besten Wert. Zusätzlich wird jedem Cluster eine der vier Fingerabdruckklassen zugeordnet, die der Cluster und die darin enthaltenen Datenpunkte repräsentieren.

In der Klassifikationsphase von Anfrageobjekten wird zuerst deren Nähe zu allen Clustern bestimmt. Der von Wang et al. verwendete 3-Nearest Neighbor Klassifikator betrachtet anschließend die drei Cluster, die die geringste Distanz zum aktuellen Anfrageobjekt aufweisen. Unter Beachtung der mit diesen Clustern assoziierten Klassen ordnet der Klassifikator dem Anfrageobjekt dann die am häufigsten aufgetretene Klasse zu. Abschließend brauchen zur Suche nach den ähnlichsten Datensätzen nur noch die Kandidaten betrachtet werden, die derselben Klasse angehören wie das Anfrageobjekt.

Die Autoren führten keine Tests mit einer Datenbank durch, sondern evaluierten lediglich ihr Klassifikationssystem bestehend aus Feature-Extraktion, Clustering und 3-Nearest Neighbor Klassifikator. Wang et al. geben an, dass ihr Klassifikator eine Genauigkeit von 91,5% erreicht. Da dieses Klassifikationssystem keine 100% Genauigkeit erreicht, ist dieses Verfahren zur Suche in Datenbanken approximierend. Zudem kann keine generelle Ergebnisgüte angegeben werden, die aussagt, wie sehr das Ergebnis der approximativen Suche mit dem Ergebnis einer exakten Suche übereinstimmt. Dies ist dadurch begründet, dass im Falle einer Fehlklassifikation durch den Klassifikator, Kandidaten untersucht werden, die keinerlei Ähnlichkeit zu den tatsächlich ähnlichsten Kandidaten aufweisen müssen. Insbesondere diese Eigenschaft erschwert die Nutzung von Klassifikatoren für Fingerabdruckdatenbanken, da durch eine Fehlklassifikation des Klassifikators die weitere Untersuchung der Kandidaten irrelevant ist.

B. Künstliche neuronale Netze

Durch den großen Einfluss des verwendeten Klassifikators auf die Ergebnisgüte der Suche, ist es zweckmäßig, einen genaueren Klassifikator zu verwenden. Zacharias und Lal verwenden für ihr Klassifikationssystem anstatt eines K-Nearest Neighbor Klassifikators ein *künstliches neuronales Netz* (KNN) [30]. Im von den Autoren durchgeführten Benchmarks klassifizierte das System Fingerabdrücke mit einer Genauigkeit von 97,14% korrekt.

Wie in der Arbeit von Wang et al., arbeitet das System in mehreren Stufen. Zuerst werden einige Vorverarbeitungen durchgeführt, bevor ein Bild eines Fingerabdrucks klassifiziert wird. Dazu wird das Bild normalisiert, Gradienten im Bild bestimmt und abschließend die für die Feature-Extraktion interessanteste Bildregion bestimmt. Danach wird ein Featurevektor, bestehend aus verschiedenen Grauwerthistogrammen, in das künstliche neuronale Netz weitergeleitet. Auf Basis der Klassifikation des Netzes können dann die Kandidaten genau untersucht werden, was Zacharias und Lal allerdings nicht näher ausführen.

Zacharias und Lal verwenden für ihre Arbeit ein mehrschichtiges feed-forward Netz, das die einfachste Art künstlicher neuronaler Netze darstellt. Zur Veranschaulichung ist in Abbildung 8 ein zweischichtiges feed-forward Netz dargestellt. Ein künstliches neuronales Netz besteht immer aus Knoten, die als Perzeptren oder Neuronen bezeichnet werden, und Kanten zwischen diesen Perzeptren. Ein Perzeptron i erhält über seine eingehenden Verbindungen einen n-dimensionalen Eingangsvektor und gewichtet dessen Komponenten mit konstanten Gewichten w_{i1} bis w_{in}. Diese Werte werden aufsummiert und eine interne Funktion entscheidet, ob das Perzeptron eine Ausgabe liefert oder nicht. Perzeptren können in mehreren Schichten angeordnet sein, bei denen die Ausgabe der Schicht k als Eingang der Schicht $k + 1$ verwendet wird. Der Ausgang der letzten Schicht liefert das Klassifikationsergebnis.

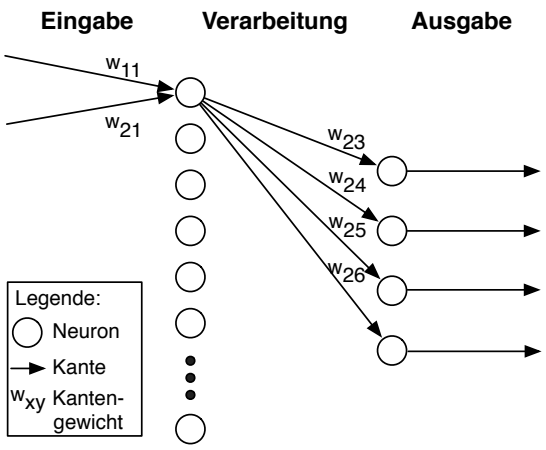


Abbildung 8. Allgemeiner Aufbau eines feed-forward Netzes mit reduzierter Kantenanzahl

Zacharias und Lal verwenden ein vierschichtiges feed-forward Netz. Schicht 1 besteht auf 64 Perzeptren und dient als Eingabe für 64 Features. Schicht 2 mit 29 Perzeptren und Schicht 3 bestehend aus 8 Perzeptren verarbeiten diese Daten. Schicht 3 leitet seine Ausgabe an die Ausgabeschicht weiter, die aus 5 Perzeptren besteht. Somit wird ein 64-dimensionaler Eingangsvektor durch das feed-forward Netz in eine von 5 Klassen kategorisiert.

Dieses feed-forward Netz wird mit 348 Fingerabdruckbildern trainiert. Training bedeutet bei feed-forward Netzen lediglich die Anpassung der Gewichte der Eingangskanten jedes Perzeptrons. Die Struktur des Netzes oder die Anzahl von Perzeptren bleiben unverändert und müssen vorher sinnvoll gewählt werden. Für das eigentliche Training mittels *Backpropagation* [19] wird das Netz mit einem Eingabevektor angeregt und die Ausgabe des Netzes mit der erwarteten Klasse verglichen. Existiert keine Übereinstimmung, werden die Gewichte von der letzten Schicht hin zur vorderen Schicht angepasst.

In der Evaluierung ließen Zacharias und Lal lediglich die Klassen von Testbildern durch das Klassifikationssystem bestimmen. Für einen Testdatensatz von 105 Bildern erreichte es eine korrekte Klassifikation in 97,14% der Fälle. Mit einer leicht veränderte Netzstruktur und anderen Trainingsdaten konnte das System 96,92% von 455, teilweise verrauschten, Testbildern korrekt klassifizieren. Damit gehört dieses System auch zu den approximierenden Verfahren, da es keine 100 prozentige Genauigkeit garantieren kann.

Aufgrund unterschiedlicher Testverfahren ist kein genereller Vergleich mit dem im vorherigen Abschnitt vorgestellten K-Nearest Neighbor möglich. Allerdings benötigen beide Verfahren ein rechenaufwändiges Offline-Training, was für eine sich ständig ändernde Fingerabdruckdatenbank nachteilig ist. Aufgrund der frühen Filterung von Kandidaten durch den Klassifikator, muss dieser möglichst aktuell gehalten werden, um den Datenbestand korrekt widerspiegeln zu können.

VIII. Evaluation

In den vorherigen Abschnitten wurden verschiedene Verfahren vorgestellt, die in diesem Abschnitt miteinander verglichen werden. Da sich die Verfahren teilweise stark voneinander unterscheiden und keine Benchmarks unter gleichen Testkriterien vorliegen, basiert der Vergleich größtenteils auf Abschätzungen. Demzufolge können in diesem Vergleich auch keine absoluten Aussagen getroffen werden, sondern nur relative Einschätzungen, die den Full Table Scan als Bezugspunkt haben.

A. Evaluationskriterien

Die vorgestellten Verfahren werden im folgenden Vergleich durch folgende Punkte charakterisiert.

1) Klassifikation
2) Art der Suche
3) Approximierend oder exakt
4) Unterstützte Nearest Neighbor Anfragen
5) Offline Training/Online Einfügen
6) Art der benutzten Features
7) Speicherbedarf
8) Skalierung
    a) Mit Dimensionalität
    b) Mit Datensatzanzahl
9) Geschwindigkeit

Das Kriterium *Klassifikation* beschreibt eine grobe Kategorisierung des Verfahrens. Mit dem Kriterium *Art der Suche* wird grob beschrieben, wie das jeweilige Verfahren die nächsten Nachbarn sucht. So berechnet Locality Sensitive Hashing für ein Anfrageobjekt zuerst die Hashwerte für alle Tabellen. Anschließend wird das Anfrageobjekt mit allen Datensätzen verglichen, die sich in den selben Buckets wie das Anfrageobjekt befinden. Für den Vergleich ist es weiterhin wesentlich, ob das Verfahren *approximierend oder exakt* arbeitet. Arbeitet ein Verfahren approximierend, wird zusätzlich die Erfolgsrate angegeben, die die Korrektheit der Ergebnisse beschreibt. Das Kriterium *Unterstützte Nearest Neighbor Anfragen* gibt Auskunft, ob ein Verfahren lediglich den nächsten Nachbarn oder die k-nächsten Nachbarn finden kann. Weiterhin ist es wesentlich, ob ein Verfahren problemlos mit neu eingefügten Datensätzen umgehen kann oder ob es eine Art *Offline Training* benötigt. So basieren Klassifikatoren häufig auf einem durch vorangegangenes Training gelernten Modell, dass neu eingefügte Datensätze eventuell nur unzureichend berücksichtigen kann. In diesem Fall wäre ein Offline Training mit allen bekannten Datensätzen notwendig, was einerseits Rechenaufwand benötigt und andererseits die Auswertung von Anfragen für eine bestimmte Zeit unterbindet. Ferner ist die *Art der benutzten Features* relevant, dass Auskunft darüber gibt, ob vor der Verarbeitung eines Datensatzes noch zusätzliche Vorverarbeitungen nötig sind.

Zur Beurteilung der Performance der Ansätze, wurden den Arbeiten, sofern dies möglich war, die Werte der folgenden Kriterien entnommen. Nannten die Arbeiten diese nicht, so

Tabelle I
EINSCHÄTZUNG DER VERFAHREN NACH DEN ERLÄUTERTEN KRITERIEN

Kriterium	Full Table Scan	LSH	R-Baum/X-Baum/ SR-Baum/A-Baum	VA-File	Z-Kurve	Künstliche neuronale Netze/ K-Nearest Neighbor
Klassifikation	seq. Suche	Hashing	Baum	optimierte seq. Suche	Space-filling curve	Klassifikator
Art der Suche	Ganze Datenbank durchsucht	Per Hashen gefundene Kandidaten durchsucht	Baumsuche	Approx. seq. Suche, danach Kandidaten durchsucht	Objekte auf Z-Kurve in Nähe vom Anfrageobjekt durchsucht	Objekte mit derselben Klasse wie Anfrageobjekt durchsucht
Approximierend oder Exakt	Exakt	Approx. (Erfolgsrate unbekannt)	Exakt	Exakt	Exakt, approx.* (Erfolgsrate 97%)	Approx. (Erfolgsrate: K-Nearest Neighbor $\leq 90\%$ Künst. neuro. Netze $\leq 97\%$)
Nearest Neighbor unterstützt	K-Nearest Neighbor	bedingt K-Nearest Neighbor	K-Nearest Neighbor	K-Nearest Neighbor	K-Nearest Neighbor	bedingt K-Nearest Neighbor
Offline Training/ Online Einfügen	Nein / Ja	Nein / Ja	Nein / Ja	Nein / Ja	Nein / Ja	Ja / bedingt
Genutzte Features	allgemeingültig oder speziell	allgemeingültig oder speziell	allgemeingültig oder speziell	allgemeingültig oder speziell	allgemeingültig oder speziell	meist speziell
Speicherbedarf	◯	◯	$-$ (SR-Baum $--$)	$-$	◯	◯
Skalierung						
Dimensionalität	◯	$+$	$+$	$+$	$++$	$+$
Datensatzanzahl	◯	$+$	$+$	$+$	$+$	$+$
Geschwindigkeit	◯	$+$	$+$ (A-Baum $++$)	$+$	$++$	$+$

Vergleich mit Full Table Scan: $--$ wesentlich schlechter, $-$ schlechter, ◯ ähnlich Full Table Scan, $+$ besser, $++$ wesentlich besser
* nach [23] hängt dies von der verwendeten LSB-Baum-Variante ab

wurden sie aufgrund der Eigenschaften des jeweiligen Verfahrens geschätzt. Dabei stellen alle angegebenen Werte keine Absolutangaben dar, sondern sind relativ zum Full Table Scan definiert. Das Kriterium *Speicherbedarf* spezifiziert, wie viel Daten zusätzlich zu den eigentlichen Datensätzen für die Verfahren gespeichert werden müssen. Mit dem Kriterium *Skalierung* wird angegeben, wie gut das betrachtete Verfahren mit einer höheren Datendimensionalität beziehungsweise der Anzahl von Datensätzen in der Datenbank skaliert. Das abschließende Kriterium *Geschwindigkeit* gibt dagegen Auskunft über die allgemeine Performance des Verfahrens.

B. Evaluation der Verfahren

Aufgrund dieser Kriterien evaluieren wir die vorgestellten Verfahren und stellen diese in Tabelle I gegenüber.

Zur Bewertung der vorgestellten Verfahren nutzen wir den Full Table Scan als Referenzverfahren. Es ist ersichtlich, dass gerade die Bäume mehr Speicherplatz in Anspruch nehmen und vor allem der SR-Baum aufgrund der Speicherung von Minimum Bounding Rectangles und Minimum Bounding Spheres wesentlich mehr Speicher als der Full Table Scan verbraucht. Dafür erzielen die Bäume im Vergleich zum Full Table Scan gute bis sehr gute Performancegewinne. Laut Böhm et al. ist auch der X-Baum schneller als der R-Baum, da im X-Baum Überlappungen der einzelnen Rechtecke vermieden werden [4]. Zur Wahrung des Vergleichs mit den anderen Verfahren, wurde der X-Baum allerdings in Bezug auf die Geschwindigkeit nicht mit ++ bewertet. Böhm et al. stufen die Geschwindigkeit des SR-Baums ähnlich zu der des X-Baums ein, da trotz höherem Speicherbedarf des SR-Baums, lediglich geringe Performanceunterschiede bestehen. Der A-Baum hingegen wurde daraufhin optimiert, bei möglichst wenig Speicherbedarf einen hohen Geschwindigkeitsgewinn zu ermöglichen. Durch die gleichzeitige Betrachtung von zwei Generationen pro Knoten kann der A-Baum relativ früh entscheiden, welcher der darunter liegenden Knoten relevant zu sein scheint. Somit stellt sich der A-Baum als das schnellste Baumverfahren heraus.

Wie in Tabelle I ersichtlich ist, schneiden Bäume in Bezug auf die Skalierung mit höheren Dimensionen relativ gut ab. Allerdings legen die verwendeten Quellen nahe, dass diese Skalierbarkeit bis zu maximal 64 Dimensionen gegeben ist. Bei einer höheren Datendimensionalität sind die Geschwindigkeitsvorteile der Bäume nicht mehr gegeben. Insbesondere für mehr als 64 Dimensionen bietet sich daher das VA-File an, da es auch bei dieser Dimensionalität mit sublinearem Aufwand arbeitet. Dazu muss lediglich mehr Speicher verwendet

werden, da pro Dimension eine bestimmte Anzahl an Bits zur approximativen Raumaufteilung genutzt werden.

Die Einschätzung der Leistungsfähigkeit der Z-Kurve stellt sich als schwierig heraus, da Tao et al. diese nicht als eigenständige Zugriffsstruktur nutzen. Stattdessen integrieren die Autoren die Z-Kurve in ein System, dass über Locality Sensitive Hashing die Dimension der Daten reduziert und anschließend per Z-Kurve auf einen B-Baum abbildet. Die Verwendung der Z-Kurve ermöglicht die Nutzung von B-Bäumen, die leistungsfähige eindimensionale Zugriffsstrukturen darstellen. Dadurch wird das gesamte Datenbanksystem sowohl skalierbar bezüglich der Datensatzgröße als auch schnell in Bezug auf die Anfragegeschwindigkeit. Die gute Skalierbarkeit der Dimensionalität ist dagegen ein Resultat aus der Kombination von LSH und der Z-Kurve.

Locality Sensitive Hashing, K-Nearest Neighbor und künstliche neuronale Netze versuchen dagegen mittels approximativer Suche ihre Geschwindigkeit zu steigern. Dazu tauschen sie Ergebnisqualität gegen Rechenzeit ein, sodass sie sowohl gut mit der Dimensionalität als auch mit der Größe des Datensatzes skalieren. Dies führt insgesamt zu einer guten Geschwindigkeit, wobei teilweise die Qualität der Ergebnisse nicht allgemein beziffert werden kann. Die Unterstützung von k-Nearest Neighbor Anfragen ist bei diesen Verfahren auch nur bedingt gegeben. Da sie das Anfrageobjekt mit einer durch Approximation begrenzten Menge M von Kandidaten vergleichen, können im Fall $|M| < k$ nicht die k-nächsten Nachbarn aus der Menge M bestimmt werden. Die restlichen Datensätze der Datenbank können dagegen nicht mittels dieser Verfahren untersucht werden, da die Approximationen keine Ähnlichkeitsaussagen über diese Restmenge in Bezug zum Anfrageobjekt erlauben. In der Praxis sollte dieses Problem bei einer ausreichend hohen Anzahl von Datensätzen und sinnvollen Werten für k allerdings nicht auftreten.

Klassifikatoren, wie der K-Nearest Neighbor Klassifikator oder die künstlichen neuronalen Netze, benötigen zudem ein zeitaufwändiges Offline-Training, um ein eigenes Modell zur Beschreibung des gesamten Datensatzes zu erzeugen. Demzufolge sind Einfüge-Operationen sowie Aktualisierungen des Datenbestandes nur bedingt möglich, da diese zu wesentlichen Veränderung des Datenbestandes führen können. In diesem Fall müssen die Klassifikatoren erneut trainiert werden, sodass ihr Einsatz bei einer sich ständig ändernden Datenbasis nachteilig sein kann.

IX. Verwandte Arbeiten

Aufgrund des begrenzten Platzes behandelt dieser Überblick nur wesentliche Verfahren, anstatt alle bekannten Zugriffsstrukturen der letzten Jahre erschöpfend zu behandeln. Folglich ist es sinnvoll, an dieser Stelle auf Arbeiten zu verweisen, die teilweise andere Zugriffsstrukturen erläutern und vergleichen. Weber et al. haben dazu eine der wesentlichen Arbeiten verfasst [27]. Die Autoren führen ein Kostenmodell ein, aufgrund dessen die Autoren verschiedene Klassen von Verfahren vergleichen. Dabei gehen sie vor allem auf Bäume ein, die Space-Partitioning Methods oder Data-Partitioning Methods nutzen. Zudem erläutern sie auch das VA-File, dass in einem eigenen Performancetest evaluiert wird.

Böhm et al. stellen in ihrer Arbeit viele relevante Verfahren vor [4]. So erläutern sie verschiedene Baumverfahren, wobei nicht nur alle bekannten R-Baum Varianten sondern auch der SR-Baum, X-Baum, LSD-Baum und Pyramidenbaum abgedeckt werden. Zudem stellen sie auch Space-Filling Curves mit deren Eigenschaften umfassend vor. Den Abschluss bildet ein Vergleich der Verfahren auf Basis der jeweiligen Vor- und Nachteile. Dabei liegt der Fokus vor allem auf der Behandlung von steigenden Dimensionen sowie dem Speicherverbrauch.

Gaede und Günther präsentieren mit ihrer Arbeit eine der umfassendsten Übersichten zu mehrdimensionalen Zugriffsstrukturen [7]. Die Autoren führen zwar keine Performancetests durch, stellen dafür aber die Vor- und Nachteile auch von sehr exotischen Verfahren heraus. Dabei werden vor allem verschiedene Space-Filling Curves, Gridfiles, Hashing Algorithmen und Baumverfahren erläutert. Neben den bekannten Standardverfahren, werden zum Beispiel auch der BSP-Baum, BD-Baum, das Twin Grid File und das BANG File sowie das PLOP Hashing vorgestellt. Eine wesentliche Erkenntnis dieser Arbeit ist, dass jede Zugriffsstruktur bestimmte Idealbedingungen verlangt, um seine Stärken gegenüber anderen Verfahren auszuspielen. So stellen vor allem die nötigen Anfragetypen, die Verteilung der Daten oder der zur Verfügung stehende Speicher wichtige Parameter dar, die die Auswahl eines Verfahrens beeinflussen.

X. Zusammenfassung

In dieser Arbeit stellten wir mehrere Zugriffsstrukturen zur Suche in hochdimensionalen Datenräumen vor. Dabei konnten wir darlegen, dass nicht nur approximative Verfahren, die Ergebnisgüte gegen Rechenzeit eintauschen, eine hohe Geschwindigkeit ermöglichen. Auch exakte Verfahren, insbesondere baumbasierte Verfahren, begegnen dem Fluch der hohen Dimensionen wesentlich besser als klassische Zugriffsverfahren relationaler Datenbankenmanagementsysteme. Als zukünftiges Forschungsthema ist neben der Erforschung neuer Verfahren vor allem die Anwendung der empfohlenen Verfahren anzusehen. So ließen sich zwar auf Basis von Benchmarks wichtige Eigenschaften ableiten, aber anwendungsspezifische Eigenschaften wie die Verteilung der Daten im Raum, Zugriffsmuster auf die Daten oder Hardwarebeschränkungen können aussichtsreiche Verfahren disqualifizieren. Von besonderem Interesse ist zudem auch die Suche nach einer möglichen oberen Schranke der Dimensionalität. So existiert wohlmöglich auch für die leistungsstärksten Verfahren eine Grenze, an der sie keinen Effizienzvorteil gegenüber dem Full Table Scan mehr erzielen.

Literatur

[1] Alexandr Andoni und Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.

[2] Rudolf Bayer. The universal b-tree for multidimensional indexing: general concepts. In *Proc. Intl. Conf. on*

Worldwide Comp. and its Appl. (WWCA), pages 198–209. Springer-Verlag, 1997.

[3] Stefan Berchtold, Daniel A. Keim, und Hans-Peter Kriegel. The x-tree: An index structure for high-dimensional data. In *Proc. Intl. Conf. on Very Large Data Bases (VLDB)*, pages 28–39. Morgan Kaufmann Publishers Inc., 1996.

[4] Christian Böhm, Bernhard Braunmüller, Markus Breunig, und Hans-Peter Kriegel. High performance clustering based on the similarity join. In *Proc. Intl. Conf. on Inf. and Knowl. Mgmt. (CIKM)*, pages 298–305. ACM, 2000.

[5] Christian Böhm, Stefan Berchtold, und Daniel A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Comput. Surv.*, 33(3):322–373, 2001.

[6] Hakan Ferhatosmanoglu, Aravind Ramachandran, Divyakant Agrawal, und Amr El Abbadi. Data space mapping for efficient i/o in large multi-dimensional databases. *Inf. Syst.*, 32(1):83–103, March 2007.

[7] Volker Gaede und Oliver Günther. Multidimensional access methods. *ACM Comput. Surv.*, 30:170–231, 1998.

[8] Aristides Gionis, Piotr Indyk, und Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proc. Intl. Conf on Very Large Data Bases (VLDB)*, pages 518–529. Morgan Kaufmann Publishers Inc., 1999.

[9] Goetz Graefe. The five-minute rule 20 years later (and how flash memory changes the rules). *Commun. ACM*, 52(7):48–59, July 2009.

[10] Kristen Grauman. Efficiently searching for similar images. *Commun. ACM*, 53(6):84–94, 2010.

[11] Antonin Guttman. R-trees: a dynamic index structure for spatial searching. In *Proc. Intl. Conf. on Mgmt. of Data (SIGMOD)*, pages 47–57. ACM, 1984.

[12] Joseph M. Hellerstein, Michael Stonebraker, und James Hamilton. Architecture of a database system. *Found. Trends databases*, 1(2):141–259, February 2007.

[13] Gisli R. Hjaltason und Hanan Samet. Index-driven similarity search in metric spaces. *ACM Trans. Database Syst.*, 28(4):517–580, 2003.

[14] Norio Katayama und Shin'ichi Satoh. The sr-tree: an index structure for high-dimensional nearest neighbor queries. In *Proc. Intl. Conf. on Mgmt. of Data (SIGMOD)*, pages 369–380. ACM, 1997.

[15] Flip Korn, Bernd-Uwe Pagel, und Christos Faloutsos. On the 'dimensionality curse' and the 'self-similarity blessing'. *IEEE Trans. on Knowl. and Data Eng.*, 13 (1):96–111, January 2001.

[16] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proc. Conf. on Emerging AI Appl. in Comp. Eng.: Real Word AI Syst. with Appl. in eHealth, HCI, Info. Retr. and Perv. Tech.*, pages 3–24. IOS Press, 2007.

[17] Anand Kumar, C. V. Jawahar, und R. Manmatha. Efficient search in document image collections. In *Proc. Asian Conf. on Comp. Vis. (ACCV)*, pages 586–595. Springer-Verlag, 2007.

[18] Eyal Kushilevitz, Rafail Ostrovsky, und Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. In *Proc. Symp. on Theory of Compu. (STOC)*, pages 614–623. ACM, 1998.

[19] D. E. Rumelhart, G. E. Hinton, und R. J. Williams. *Learning internal representations by error propagation*, pages 318–362. MIT Press, 1986.

[20] Yasushi Sakurai, Masatoshi Yoshikawa, Shunsuke Uemura, und Haruhiko Kojima. The a-tree: An index structure for high-dimensional spaces using relative approximation. In *Proc. Intl. Conf. on Very Large Data Bases (VLDB)*, pages 516–526. Morgan Kaufmann Publishers Inc., 2000.

[21] Timos K. Sellis, Nick Roussopoulos, und Christos Faloutsos. The r+-tree: A dynamic index for multi-dimensional objects. In *Proc. Intl. Conf. on Very Large Data Bases (VLDB)*, pages 507–518. Morgan Kaufmann Publishers Inc., 1987.

[22] Arnold W. M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, und Ramesh Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1349–1380, December 2000.

[23] Yufei Tao, Ke Yi, Cheng Sheng, und Panos Kalnis. Efficient and accurate nearest neighbor and closest pair search in high-dimensional space. *ACM Trans. Database Syst.*, 35(3):1–46, 2010.

[24] Eric Sadit Tellez und Edgar Chavez. On locality sensitive hashing in metric spaces. In *Proc. Intl. Conf. on Simil. Search and Appl. (SISAP)*, pages 67–74. ACM, 2010.

[25] Eduardo Valle, Matthieu Cord, und Sylvie Philipp-Foliguet. High-dimensional descriptor indexing for large multimedia databases. In *Proc. Conf. on Info. and Know. Mgmt. (CIKM)*, pages 739–748. ACM, 2008.

[26] Sen Wang, Wei Wei Zhang, und Yang Sheng Wang. Fingerprint classification by directional fields. In *Proc. IEEE Intl. Conf. on Multimodal Interfaces (ICMI)*, page 395. IEEE Computer Society, 2002.

[27] Roger Weber, Hans-Jörg Schek, und Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proc. Intl. Conf. on Very Large Data Bases (VLDB)*, pages 194–205. Morgan Kaufmann Publishers Inc., 1998.

[28] Shijun Xiang, Hyoung-Joong Kim, und Jiwu Huang. Histogram-based image hashing scheme robust against geometric deformations. In *Proc. Work. on Multi. & Sec. (MM&Sec)*, pages 121–128. ACM, 2007.

[29] Xin Yang, Qiang Zhu, und Kwang-Ting Cheng. Near-duplicate detection for images and videos. In *Proc. ACM Workshop on Large-scale Multi. Retr. and Mining (LS-MMRM)*, pages 73–80. ACM, 2009.

[30] Geevar C. Zacharias und P. Sojan Lal. Combining singular point and co-occurrence matrix for fingerprint classification. In *Proc. ACM Bangalore Conf. (COMPUTE)*, pages 1–6. ACM, 2010.

Implementierung und Evaluierung hochdimensionaler Indexstrukturen

Björn-Erik Aust, Alexander Grebhahn, Tran Tuan Nguyen, Reimar Schröter
Institut für Technische und Betriebliche Informationssysteme (ITI)
Fakultät für Informatik
Otto-von-Guericke-Universität
Magdeburg, Deutschland
{*Bjoern-Erik.Aust, Alexander.Grebhahn, Tran.Nguyen, Reimar.Schroeter*}*@st.ovgu.de*

***Zusammenfassung*—Die Anwendungsbereiche von heutigen Datenbanksystemen haben sich von der Arbeit mit reinen textbasierten Daten losgelöst und arbeiten mittlerweile beispielsweise auf der Basis von Daten mit multimedialen Inhalten. Diesbezüglich gewinnen effiziente Indexstrukturen zur Ähnlichkeitssuche in hochdimensionalen Datenräumen zunehmend an Bedeutung [4]. Die grundlegende Herausforderung, welche dabei zu nennen ist, ist der *Fluch der hohen Dimensionen*. Die vorliegende Arbeit beschäftigt sich mit ausgewählten Indexstrukturen für die Suche im hochdimensionalen Datenraum und deren quantitativer Evaluierung. Dabei werden wir auf die Pyramidentechnik und das *L*ocal *P*olar *C*oordinate (LPC)-File näher eingehen. Außerdem werden wir das RABT-File vorstellen, welches im Rahmen dieser Arbeit entstanden ist. Der Fokus dieser Arbeit liegt auf der quantitativen Evaluierung der Indexstrukturen. Dazu werden wir unser Evaluierungsverfahren und die einzelnen Evaluierungskriterien vorstellen. Die Ergebnisse der Evaluierung werden wir mittels eines Intra- und eines Inter-Indexstrukturvergleiches darlegen. Hierbei wurden für die Indexstrukturen auf der Basis der Intra-Indexstrukturvergleiche Parameterprofile angelegt. Anschließend wurden diese Parameterprofile für die Inter-Indexstrukturvergleiche verwendet.**

***Keywords*-Indexstrukturen; Hochdimensional; RABT-File; Intra- und Inter-Indexstrukturvergleich**

I. Einleitung

Diese Arbeit entstand im Rahmen des Forschungsprojektes zur digitalen Daktyloskopie[1]. Bei der digitalen Daktyloskopie werden unterschiedliche Fingerabdrücke miteinander verglichen, damit diese erfasst und auf ihre Ähnlichkeit untersucht werden können. Diese Ähnlichkeitssuche kann mittels einer Extraktion von Eigenschaften beziehungsweise Features aus diesen Fingerabdrücken durchgeführt werden. Hierbei werden mehrere Features aus den verschiedenen Fingerabdrücken in einen mehrdimensionalen Vektorraum transformiert [4]. Auf Basis der erfassten Feature-Vektoren sollen effiziente Ähnlichkeitsanfragen im mehrdimensionalen Datenraum durchgeführt werden. Daher ist im Rahmen dieser Arbeit die Unterscheidung zwischen multi- und hochdimensionaler Datenräume relevant. Eine solche Unterscheidung ist notwendig, da Indexstrukturen im multidimensionalen Datenraum effizient, aber ab einer bestimmten Anzahl von Dimensionen langsamer bezüglich der Antwortzeit als die sequentielle Suche sind. Diese Problematik wird als *Fluch der hohen Dimension* (*Curse of Dimensionality*) bezeichnet [4]. Im Kontext dieser Arbeit sprechen wir von hochdimensionalen Datenräumen bei mehr als zehn Dimensionen. Weiterhin wird angenommen, dass die Dimensionen unabhängig zueinander und gleich priorisiert sind.

In dieser Arbeit werden wir einige Indexstrukturen vorstellen, welche laut Literatur effizient bezüglich hochdimensionaler Ähnlichkeitsanfragen arbeiten [2, 8]. Diese Indexstrukturen wurden implementiert und werden in dieser Arbeit mit anderen Indexstrukturimplementierungen verglichen, welche durch ein Vorgängerprojekt umgesetzt wurden (siehe [6]). Folglich muss geklärt werden, wann eine Indexstruktur als effizient anzusehen ist, damit diese untereinander vergleichbar sind.

Im Rahmen der Recherchephase konnte keine quantitative Evaluierung mehrerer Indexstrukturimplementierungen aufgefunden werden. Aufgrund dessen ist das Ziel dieser Arbeit eine quantitative Evaluierung verschiedener implementierter Indexstrukturen und die Erstellung eines Verfahrens zum Vergleich der unterschiedlichen Indexstrukturen. Aus dieser quantitativen Evaluierung können Rückschlüsse gezogen werden, ob die gewählten Indexstrukturen effizient im hochdimensionalen Datenraum sind.

Zur Evaluierung der Indexstrukturen wurde bereits von einem Vorgängerprojekt ein Framework entwickelt. Hierbei wurde der KD-Baum, ein Vector-Approximation (VA)-File, ein prototypbsiertes LSH-Verfahren und eine Form der sequentiellen Suche implementiert. Dieses Framework wurde übernommen und wurde im Rahmen dieser Arbeit erweitert. Zur Erweiterung des Frameworks wurde eine Genauigkeitsberechnung, eine Möglichkeit zur Datenclusterung, weitere Indexstrukturen und eine Speicherung von erzeugten Zufallsdaten zur Sicherstellung der Wiederholbarkeit implementiert. Als weiterer Indexstrukturen haben wir die Pyramidentechnik, das LPC-File, das RABT-File und den R-Baum umgesetzt. Weiterhin wurde ein Evaluierungskonzept festgelegt, welches in dieser Arbeit vorgestellt und angewendet wird.

Wir werden im Abschnitt II dieser Arbeit thematische Grundlagen erläutern und auf ausgewählte Indexstrukturen eingehen, die im Rahmen dieses Projektes implementiert

[1] https://omen.cs.uni-magdeburg.de/digi-dak,letzter Zugriff: 26.06.2010

wurden. Des Weiteren werden im Abschnitt III die Grundidee des RABT-Files und die Erweiterungen des Frameworks erläutert. Der Abschnitt IV beschäftigt sich mit dem Evaluierungskonzept. In diesem Zusammenhang werden die Ergebnisse des Intra- und Inter-Indexstrukturvergleichs präsentiert. Abschließend werden wir im Abschnitt VI die Ergebnisse dieser Arbeit zusammenfassen und einen Ausblick auf zukünftige Forschungsfelder geben.

II. Grundlagen

In dieser Arbeit bauen wir auf den Ergebnissen eines Vorgängerprojektes auf [6]. Broneske et al. erläuterten, was exakte und approximierende Suchverfahren sind, welche unterschiedlichen Suchanfragen existieren (z.B. k-nächste Nachbarn Anfragen (kNN)), was hochdimensionale Indexstrukturen im Anwendungsbeispiel der digitalen Daktyloskopie bedeuten, wie sich der *Fluch der hohen Dimensionen* äußert und wie ausgewählte Indexstrukturen zur Ähnlichkeitssuche arbeiten. Hierbei wurden die Vor- und Nachteile der einzelnen Indexstrukturen und die folgenden Indexstrukturen erläutert. Zum einen wurden Verfahren wie das LSH (Locality Sensitive Hashing) [11], das VA-File [18], die Space-Filling Curves [16] veranschaulicht. Zum anderen wurden Baumverfahren wie der R- [10], X- [3], SR- [13] und A-Baum [17] beschrieben. Für diese Erkenntnisse und das aufgearbeitete Fachwissen verweisen wir den Leser auf die Arbeit des Vorgängerprojektes. Nachfolgend werden wir in diesem Abschnitt auf die Evaluierungsgrundlagen, die Pyramidentechnik und das LPC-File eingehen, um sich bei der Evaluierung darauf beziehen zu können.

A. Evaluiuerungsgrundlagen

Der Fokus dieser Ausarbeitung liegt in der Evaluierung der im Framework vorhandenen Indexstrukturen. Zur Evaluierung werden die Eigenschaften von Indexstrukturen erfasst. Eine geeignete Indexstruktur für eine Ähnlichkeitssuche sollten laut Qin Lv et al. die nachfolgenden Eigenschaften erfüllen [15]. Zum einen sollten die zu untersuchende Indexstruktur genaue Ergebnisse liefern (Precision) [15]. Dies bedeutet, dass die Indexstrukturen die gleichen beziehungsweise ähnliche Ergebnisse wie die sequentielle Suche liefern sollten. Damit in der Evaluierung die Ergebnisse approximierender Verfahren ausgewertet werden können, muss eine Untersuchung der Genauigkeit vorgenommen werden. Eine weitere Eigenschaft von Indexstrukturen, die betrachtet werden muss, ist die zeitliche Effizienz [15]. In dieser Arbeit wird die zeitlichen Effizienz über bessere Antwortzeiten bei exakten Suchanfragen und kNN-Anfragen der Indexstrukturen festgelegt. Bezüglich der Eigenschaften von Indexstrukturen kann ebenfalls die Speichereffizienz analysiert werden [15]. Hierbei ist ein geringer Speicherverbrauch der Indexstruktur von Vorteil, welcher idealerweise minimal größer ist, als der Speicherverbrauch des Datenbestandes. Eine weitere Eigenschaft für Indexstrukturen zur Ähnlichkeitssuche ist die Effizienz bezüglich hochdimensionaler Datenräume [15]. Hierbei wird erfasst, ob die Indexstrukturen auch für hochdimensionale Datenbestände geeignet sind.

B. LPC-File

Das LPC-File ist eine Indexstruktur, die erstmals von Guang-Ho Cha et al. veröffentlicht wurde und zusätzlich zum eigentlichen Punkt x sogenannte Polarkoordinaten („**L**ocal **P**olar **C**oordinate“) abspeichert [8]. Wie das VA-File, dass im Vorgängerprojekt dieser Ausarbeitung [6] und ursprünglich von Roger Weber eingeführt wurde [18], teilt das LPC-File den Raum zunächst in hyperrechteckige Zellen. Diese Zellen werden durch einen eindeutigen Bit-String identisch zum VA-File identifiziert. Ein Unterschied der beiden Verfahren besteht in der Raumaufteilung. Während das VA-File die Größe der hyperrechteckigen Zellen je nach Füllgrad zuteilt, ist die Größe aller Zellen im LPC-File identisch. Dementsprechend wird jeder Dimension d die gleiche Anzahl von Bits b zugewiesen und der Wertebereich der Dimensionen in 2^{bd} Partitionen unterteilt. Eine weitere Abgrenzung des LPC-File gegenüber dem VA-File ergibt sich durch die Polarkoordinaten. Diese bestehen aus dem Radius r und dem Winkel ϕ und bilden zusammen das Tupel $\langle r, \phi \rangle$. Wie in Abbildung 1 (Zelle 01 10) ersichtlich, ergibt sich der Radius r durch die Distanz des lokalen Koordinatenursprung *O*, der sich in der linken unteren Ecke der Zelle befindet, und dem Punkt x. Der gesuchte Winkel ϕ berechnet sich aus dem Winkel zwischen dem Punkt x und der Raumdiagonalen der Zelle, die durch den Koordinatenursprung zu der gegenüberliegenden Ecke verläuft. Als Ergebnis wird der Punkt x durch das Tripel $\langle c, r, \phi \rangle$ repräsentiert, wobei c die lokale Zelle ist, in dem sich der gegebenen Punkt befindet. Das berechnete Tripel für den Punkt x kann jedoch ebenfalls dem Punkt x' (Abbildung 1) zugewiesen werden, die berechneten Koordinaten sind demnach nicht eindeutig. Im abgebildeten Beispiel handelt es sich lediglich um einen Punkt mit den gleichen Koordinaten, im hochdimensionalen Raum erhöht sich diese Anzahl. Beispielsweise werden im dreidimensionalen Raum alle Punkte auf einer Kreisbahn um die Raumdiagonale gleiche Koordinaten zugewiesen.

Die Vorteile des LPC-File gegenüber dem VA-File ergeben sich nach Guang-Ho Cha et al. durch die effizientere Berechnung der kNN gegenüber dem VA-File [8]. Wie beim VA-File werden sowohl der minimal als auch der maximal mögliche Abstand jedes Punktes zum gesuchten Punkt q berechnet. In der Abbildung 1 (Zelle 11 01) wird dargestellt, wie beim VA-File die Berechnung des kleinsten $dist_{min}(VA)$ und des größten Abstands $dist_{max}(VA)$ zur Zelle, in der sich der jeweilige Punkt x befindet, ermittelt wird. Durch die Nutzung der Polarkoordinaten im LPC-File kann dieser minimale $dist_{min}(LPC)$ und maximale $dist_{max}(LPC)$ Abstand genauer berechnet und

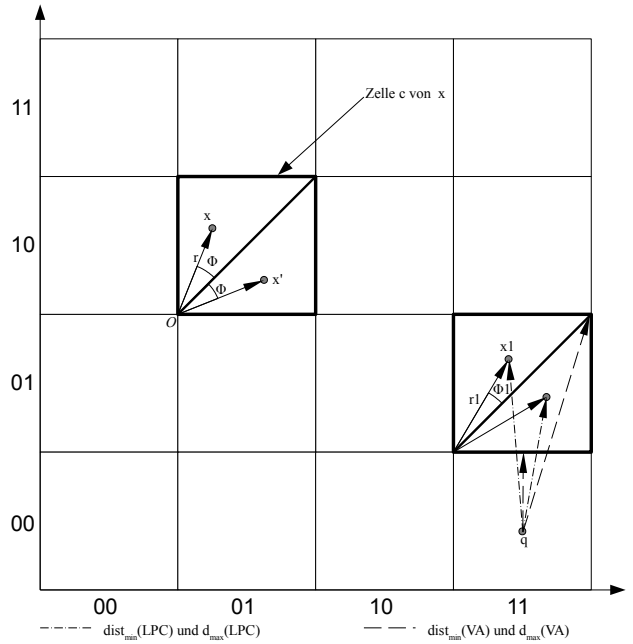


Abbildung 1. Raumaufteilung des LPC-File und Darstellung der Polarkoordinaten (Zelle 01 10) & Vergleich von $dist_{min}$ und $dist_{max}$ vom VA-File und LPC-File (Zelle 11 01).

somit der Suchraum der kNN weiter eingegrenzt werden. Dementsprechend verringert sich die Anzahl der Kandidaten, die zur exakten Distanzberechnung zum gesuchten Punkt q ermittelt wurden. Eine anschließende Sortierung dieser Kandidaten nach ihren absoluten Abständen zu q, ergibt die gesuchten k nächsten Nachbarn.

C. Pyramidentechnik

Bei der Pyramidentechnik handelt es sich um eine Methode der Zerlegung hochdimensionaler Datenräume, die sich als Basis für mehrdimensionale Indexstrukturen eignet [2]. Durch sie wird ein n-dimensionaler Raum zuerst in 2n Pyramiden aufgeteilt. Hierbei sei angemerkt, dass die Nummerierung nach einem speziellen Schema verläuft. Dabei gilt die Definition, dass ein normalisierter n-dimensionaler Punkt x in der Pyramide p_i liegt mit:

$$i = \begin{cases} j_{max} & \text{wenn } x_{j_{max}} < 0.5 \\ (j_{max} + n) & \text{wenn } x_{j_{max}} \geq 0.5 \end{cases}$$

$$j_{max} = (j | \forall k, 0 \leq (j,k) < n, j \neq k : |0.5 - x_j| \geq |0.5 - x_k|))$$

Eine Visualisierung dieser Aufteilung des Raumes ist in Abbildung 2 für einen zweidimensionalen Raum gezeigt. Die Pyramiden werden in der Abbildung P0 bis P3 genannt. Zur effizienteren Verwaltung des Raumes werden diese Pyramiden ihrerseits in Stümpfe aufgeteilt. Dabei gibt es, je nach den zu unterstützenden Anfragetypen, unterschiedliche Arten der Aufteilung. So wird in Berchtold et al. bei der Durchführung von Bereichsanfragen die Pyramiden parallel zu ihrer Basis unterteilt [2]. Während in Dong-Ho Lee et al. zur Unterstützung von kNN Anfragen eine sphärische Aufteilung von der Spitze aus durchgeführt wird [14]. Da im Rahmen dieses Projektes kNN Anfragen durchgeführt werden sollen, wurde bei der hier beschriebenen Implementierung eine sphärische Aufteilung der Pyramiden vorgenommen. Dabei muss untersucht werden, in wie viel Stümpfe eine Pyramide aufgeteilt werden sollte. Ein erster Einblick dazu ist im Abschnitt Intra-Indexstrukturvergleiche gegeben. Bei der dort evaluierten Implementierung wird die Anzahl der Stümpfe vorher festgelegt, wobei alle Stümpfe gleich groß sind. So wird bei einer Stumpfanzahl von y ein normalisierter Punkt x in den Stumpf i eingefügt, wenn: $\lfloor i \rfloor = \frac{x_d * y}{dist_{max}+1}$. Wobei x_d die Distanz des Punktes zum Raummittelpunkt und $dist_{max}$ die maximal mögliche Distanz zum Raummittelpunkt sind.

Soll eine kNN Anfrage ausgeführt werden, wird eine inkrementelle Berechnung vorgenommen, die sich an den in von Dong-Ho Lee et al. beschriebenen Algorithmus anlehnt [14]. Damit beispielsweise die zwei nächsten Nachbarn des Punktes q aus Abbildung 2 berechnet werden können, müssen zuerst die Distanzen des Punktes zu den Pyramiden bestimmt werden. Ist dies geschehen, werden die Pyramiden nach ihrer Distanz zum Punkt q aufsteigend sortiert in eine Prioritätsliste eingefügt.

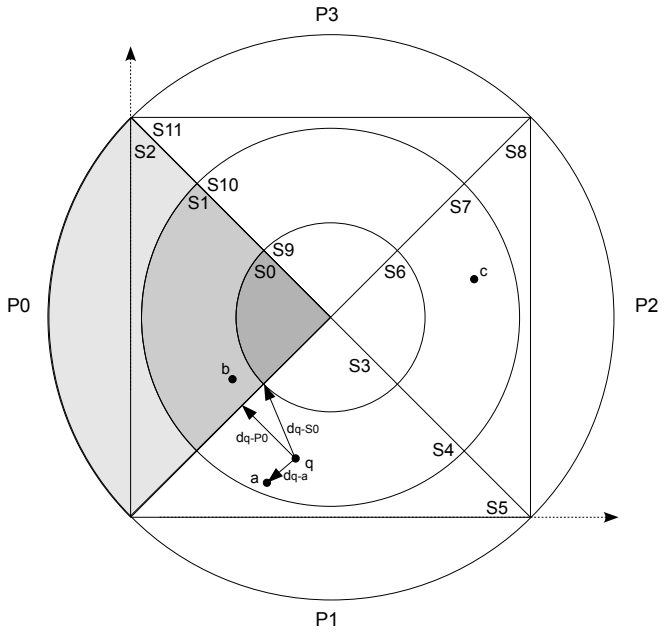


Abbildung 2. Distanzberechnungen der Pyramiden Technik bei einer kNN Anfrage.

Nachdem dies erfolgt ist, wird das jeweils erste Element aus der Liste solange entfernt und betrachtet, bis k Nachbarn

gefunden wurden und die größte Distanz der Nachbarn zum Punkt kleiner ist als die Distanz des ersten Elementes der Liste. Dabei kann es sich um Pyramiden und um Pyramidenstümpfe handeln.

1) *Pyramide:* Handelt es sich bei dem Element um eine Pyramide müssen die Distanzen der Pyramidenstümpfe zum Anfragepunkt berechnet werden. Ist dies geschehen, werden die Pyramidenstümpfe mit ihren berechneten Distanzen in die Prioritätenliste eingefügt.
2) *Pyramidenstumpf:* Wird ein Pyramidenstumpf als erstes Element der Liste zurückgegeben, müssen die Distanzen aller Punkte dieses Stumpfes zu dem Anfragepunkt berechnet werden. Diese Punkte werden dann anders als in der von Dong-Ho Lee beschriebenen Implementierung in eine eigene Ergebnisliste eingefügt.

Aufgrund der oben beschriebenen Abbruchbedingung handelt es sich bei dem Algorithmus um ein exaktes Verfahren. Eine genauere Beschreibung dieses Algorithmus ist in [14] gegeben.

III. Implementierte Frameworkerweiterungen

Zusätzlich zu der Evaluierung der Indexstrukturen, die in Abschnitt IV beschrieben ist, wurde das existierende Framework noch an einigen Stellen erweitert. So wurden nicht nur existierende Indexstrukturen implementiert. Es wurde auch eine Indexstruktur entwickelt, die auf der Grundidee der Dimensionsaufteilung und -reduktion basiert. Sie wird in der Arbeit RABT-File genannt. Des Weiteren wurde ermöglicht, die existierenden Indexstrukturen in Hinblick auf ihre Genauigkeit zu vergleichen. Zusätzlich wurde die Möglichkeit geschaffen neben gleichverteilten Datensätzen auch geclusterte Datensätze zu erzeugen. Neben der Datenverteilung wurde sichergestellt, dass die Evaluierungen auch zu einem späteren Zeitpunkt mit den gleichen Datensätzen wiederholt werden können.

A. RABT-File

Viele Indexstrukturen leiden unter dem *Fluch der hohen Dimensionen* und sind ab einer bestimmten Dimensionsanzahl schlechter als die sequentielle Suche. Außerdem ist es bei einigen Indexstrukturen möglich, dass ein Wertüberlauf (beispielsweise bei Verwendung eines ungeeigneten Datentyps) auftreten und das Ergebnis verfälschen kann beziehungsweise deren Verwendung unmöglich macht. Aus diesem Grund entwickelten wir im Rahmen dieser Arbeit, angelehnt an der Idee von Boehm, eine neue Indexstruktur [4]. Jedoch besteht beim Verfahren von Boehm keine Möglichkeit zur Dimensionsreduktion. Daher wurde das RABT-File entwickelt, welches eine Reduktion von Dimensionen unterstützt und auch in sehr hohen Dimension gut skalierbar ist. Dabei sollen die Suchanfragen parallelisierbar sein, um eine Verkürzung der Antwortzeit zu bewirken. Das RABT-File basiert auf Dimensionsaufteilung und -reduktion, daher ist das Verfahren approximativ. Die Funktionsweise wird in der Abbildung 3 dargestellt.

Zunächst wird eine Dimensionsaufteilung durchgeführt. In diesem Zusammenhang wird jeder Datenpunkt in n gleichgroße Segmente aufgeteilt, wobei Überschüssige Dimensionen vernachlässigt werden. Anschließend erfolgt auf der Grundlage von mehrdimensionalen Indexstrukturen, beispielsweise der Pyramidentechnik oder dem KD-Baum, die Erstellung von n Basisindexstrukturen. Infolgedessen wird das i-te Segment des Datenpunktes in die i-te Basisindexstruktur eingefügt, wobei gilt $1 \leq i \leq n$.

Eine weitere Verfeinerung des RABT-Files besteht in der Nutzung der Dimensionsreduktion. Falls diese Einstellungsmöglichkeit ausgewählt wurde, wird zusätzlich zu der beschriebenen Dimensionsaufteilung eine Reduktion durchgeführt. Hierbei wird die Anzahl der Dimensionen der zu erstellenden Basisindexstrukturen um einen bestimmten Faktor verringert. Dazu werden die Werte der einzelnen Dimension beispielsweise durch Summen- oder Mittelwertbildung vereinigt. Im Rahmen dieser Arbeit wurde eine Implementierungsvariante unter Nutzung der Summenbildung zur Dimensionsreduktion ausgewählt. Beispielsweise wird in Abbildung 3 der Punkt {1,2,3,4,5,6,7,8} in zwei Punkte {1,2,3,4} und {5,6,7,8} aufgeteilt. Anschließend wird die Dimensionsreduktion mit Dimensionsreduktionsfaktor 2 angewendet, sodass die neuen Punkte {3,7} und {11,15} entstehen.

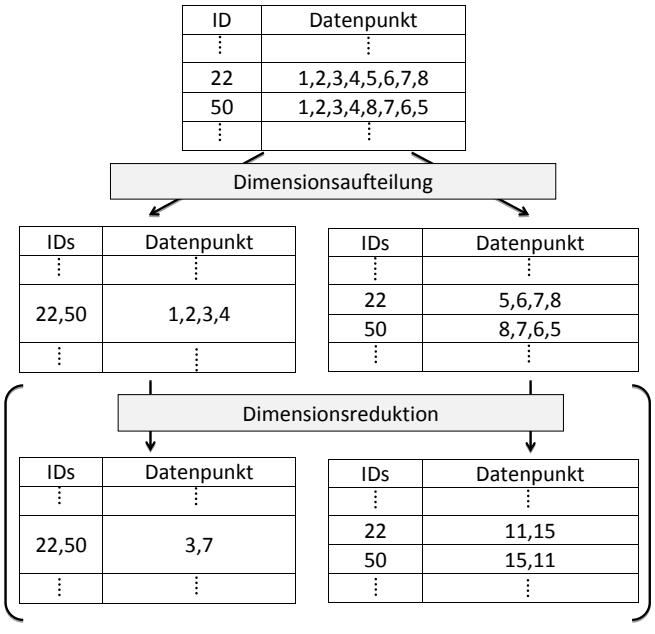


Abbildung 3. RABT-File mit 2 Basisindexstrukturen und Dimensionreduktionsfaktor von 2.

Anfragen an das RABT-File werden folgendermaßen be-

handelt. Eine Punktanfrage wird realisiert, indem die Dimensionsaufteilung und -reduktion auch auf dem Anfragepunkt q durchgeführt wird. Anschließend wird auf jede i-te Basisindexstruktur eine Punktanfrage mit dem i-ten Segment des Anfragepunktes q gestellt, dabei gilt $1 \leq$ i $\leq$ n. Da jeder Punkt in der Basisindexstruktur auch mit der zugehörigen RABT-File-ID gespeichert wurde, können die Ergebnisse der einzelnen Basisindexstrukturen auf den original gespeicherten Punkt zurückgeführt und einfach mit dem Anfragepunkt q verglichen werden. Wenn beide Punkte identisch sind, wird der gefundene Punkt zurückgegeben.
Durch eine Parallelisierung der Suche in den jeweiligen Basisindexturkturen mittels Threads, kann zusätzlich die Suche beschleunigt werden. Wenn eine Basisindexstuktur beziehungsweise ein Thread die RABT-File-ID und somit den gesuchten Punkte gefunden hat, wird das Ergebnis sofort zurückgegeben und alle anderen Threads gestoppt.
Bei kNN-Anfragen werden aus jeder Basisindexstuktur die potentiellen kNN herausgesucht. Wie bei Punktanfragen kann die Laufzeit der Suche auch durch das Parallelisierungskonzept verbessert werden. Anders als bei Punktanfragen muss hier gewartet werden bis alle Threads mit ihrer Suche fertig sind und somit alle Basisindextrukturen die potenzielle kNN gefunden haben. Die gefundenen potentiellen Lösungskandidaten aus den unterschiedlichen Basisindexstrukturen werden anschließend zu einer Menge zusammengefasst, wobei Duplikate entfernt werden. Im Folgenden werden die kNN der Gesamtmenge unter Nutzung der sequentiellen Suche bestimmt. Bei der kNN-Anfragen kann keine Genauigkeit von 1 garantiert werden, da durch die Aufsplittung und Dimensionsreduktion Semantik verloren ging.

B. Erweiterung des Frameworks

Genauigkeitsberechnung: Eine Berechnung der Precision wurde vorgenommen, um exakte und aproximierende Verfahren bezüglich ihrer Genauigkeit untersuchen zu können. Bei der Precision handelt es sich um ein Maß, dass angibt, wie viel Prozent der von der sequentiellen Suche gefundenen k nächsten Nachbarn auch von der zu evaluierenden Indexstruktur gefunden wurden. Damit diese Berechnung auch bei Spezialfällen, wie bei einer kreisförmigen Anordnung aller Punkte um den gegebenen Punkt funktioniert, wurde die Precision über die Distanzen der zurückgegebenen Punkte berechnet. Dadruch werden nicht nur die von der sequentiellen Suche gefundenen kNN betrachtet, sondern auch alle Punkte, die den gleichen Abstand wie der k-te Punkt der sequentiellen Suche haben.

Datenclusterung: Zusätzlich zu der Möglichkeit gleichverteilte Daten zu erstellen, wurde das Framework dahingehend erweitert, dass geclusterte Daten erzeugt werden können. Damit ein erster Überblick über das Verhalten der implementierten Indexstrukturen gewonnen werden kann, wurde eine einfache Art der Clusterbildung implementiert. Dabei handelt es sich um eine parametrisierte Mittelwertbildung zwischen dem erzeugten Punkt und dem Mittelpunkt des Clusters. Dabei wird bei einer Clusterstärke von s jeder der geclusterten Punkte innerhalb eines Radius von $\frac{1}{2s+2}$ um den Clustermittelpunkt verschoben. Ein Beispiel für diese Berechnung wird in Abbildung 4 aufgezeigt. In der Abbildung besitzt der Clustermittelpunkt C_0 eine Clusterstärke von 1. Das bedeutet, dass alle Punkte, die zu diesem Cluster gehören sollen, in die Mitte zwischen ihrer originalen Position und der Position des Clustermittelpunktes verschoben werden. Die verschobene Position des Punktes x lässt sich also durch die Gleichung $x' = \frac{x+(s*m)}{s+1}$ berechnen, wobei m der Clustermittelpunkt ist. Hierbei sei erwähnt, dass alle Cluster die gleiche Clusterstärke besitzen. Des Weiteren können noch die Anzahl der Cluster und der prozentuale Anteil geclusterter Punkte eingestellt werden.

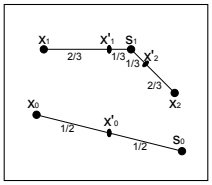


Abbildung 4. Punktverschiebung mit Clusterstärken von s=1 bei C_0 und s=2 bei C_1.

Wiederholbarkeit sicherstellen: In dem Framework ist es möglich Punktanfragen mit Datenpunkten zu stellen, die nicht in der Indexstruktur enthalten sind. Zur Verifizierung der Evaluierungsergebnisse sollten die Anfragen wiederholbar sein. Dazu ist es nötig diese Punkte zu speichern und auch für neue Anfragen zu verwenden. Infolgedessen wurde das Framework dahingehend erweitert, dass jedem Datensatz eine Menge von Zufallsdatensätzen zusortiert werden kann.

IV. Evaluierung

In diesem Abschnitt gehen wir auf die quantitative Evaluierung der implementierten Indexstrukturen ein. Dazu werden wir das verwendete Evaluierungskonzept erläutern und anschließend die Evaluierungsergebnisse vorstellen. Um das Evaluierungskonzept vorzustellen, gehen wir auf die Evaluierungsdurchführung, die -parameter und die -umgebung ein.

A. Evaluierungsdruchführung

Indexstrukturen können unterschiedliche Einstellungsmöglichkeiten besitzen. Diese haben zur Folge, dass bei der Evaluierung der Indexstrukturen neben dem Vergleich der Indexstrukturen untereinander (Inter-Indexstrukturvergleiche) auch Intra-Indexstrukturvergleiche durchgeführt werden sollten. Beim Intra-Indexstrukturvergleich wird eine Indexstruktur mit unterschiedlichen Einstellungsparametern mit sich selbst verglichen. Somit soll eine geeignete Ausprägung der Parameter gefunden werden. Um die Indexstrukturen zu

evaluieren wurden Intra- und Inter-Indexstrukturvergleiche vorgenommen. Die Intra-Indexstrukturvergleiche wurden durchgeführt, um optimale Einstellmöglichkeiten der Indexstrukturen identifiziert zu können. Auf der Grundlage der Intra-Indexstrukturvergleiche wurden Parameterprofile für die Inter-Indexstrukturvergleiche festgelegt. Somit bilden die Ergebnisse der Intra-Indexstrukturvergleiche die Basis für die Inter-Indexstrukturvergleiche.

B. Evaluierungsparameter

Zur Evaluierung der Indexstrukturen untereinander, wurden Vergleichsparameter identifiziert. Hierbei wurden Vergleichsparameter wie die Antwortzeit [7, 20], die Anzahl der Datensätze [2, 7], die Anzahl der Dimensionen [14] und die Precision [20] erfasst. Die identifizierten Vergleichsparameter wurden zum Inter-Indexstrukturvergleich bei der Evaluierung eingesetzt, da wir diese als zielführend angesehen haben. Nachfolgend werden die unterschiedlichen Vergleichs- bzw. Evaluierungskriterien erläutert.
Eine Eigenschaft der Indexstrukturen ist die Genauigkeit. Damit diese in der Evaluierung erfasst werden kann, wurde die Precision der Indexstrukturen in den Testdurchläufen berechnet.
Die zeitliche Effizienz, welche bereits als Eigenschaft der Indexstrukturen vorgestellt wurde, wird in der Evaluierung als Vergleichskriterium eingesetzt. Damit die zeitliche Effizient abgebildet werden kann, wird zwischen der Antwortzeit bezüglich Punkt- und kNN-Anfragen der Indexstrukturen unterschieden. Bei der Antwortzeit wird die Zeit gemessen, die die Indexstruktur benötigt, um die jeweilige Suchanfragen durchzuführen.
Die Speichereffizienz wird als Eigenschaft der Indexstrukturen im Rahmen dieser Arbeit nicht untersucht, da während der Entwicklungsphase nötige Implementierungsdetails gefehlt haben.
Die Eigenschaft der Effizienz bezüglich der Dimensionen der Indexstrukturen kann als Parameter mit Hilfe der Dimensionsgröße untersucht werden, weshalb dieser Parameter in die Evaluierung integriert wurde. Dieser Parameter soll einen Aufschluss über das Verhalten der Indexstrukturen im hochdimensionalen Datenraum liefern.
Einige Indexstrukturen können vermutlich bei einer steigenden Anzahl von Datensätzen effizienter als andere Indexstrukturen sein. Um ein solches Verhalten zu untersuchen, wurde die Anzahl der Datensätze als Evaluierungskriterium mit aufgenommen.

C. Evaluierungsumgebung

Im Rahmen der Evaluierungsphase mussten gleiche Testbedingungen für die Indexstrukturen hergestellt werden, um möglichst reproduzierbare und valide Ergebnisse zu erzeugen. Dies bedeutet, dass die Indexstrukturen auf der Grundlage der gleichen Datensätze verglichen wurden. Weiterhin wurden die Daten innerhalb dieser Datensätze im geschlossenen Intervall [0,255] gleichverteilt. Hierbei sei angemerkt, dass die Daten auch geclustert werden könnten. Eine Evaluierung unter Berücksichtigung der Clusterung wurde in dieser Evaluierung nicht vorgenommen.
Eine weitere Anforderung der Testdurchläufe ist die Wiederholbarkeit. Dazu wurden die erstellten Datensätze, einschließlich der Zufallsdatensätze gespeichert und in den Testergebnissen hinterlegt. Dies unterstützt die Transparenz und Wiederholbarkeit der Testdurchläufe. Zur Herstellung der Wiederholbarkeit wurde eine homogene Hardwareumgebung beim Evaluieren und eine homogene Java-Version bei der Implementierung der Indexstrukturen verwendet. Als Hardwareumgebung wurde ein Rechner mit einem Intel Core i5 Prozessor mit jeweils 3,33GHz unter Windows7 eingesetzt. Zur Implementierung wurde als Entwicklungsumgebung Eclipse[2] verwendet. Zur Testumgebung wurde eine Java Virtual Machine (JVM) mit einer Java-Version 1.6 mit 64bit benutzt. Dieser JVM wurden 4096MB RAM zugewiesen.
Der Test einer Indexstruktur wurde in der Evaluierung mit 10 Testdurchläufen wiederholt. Aus den einzelnen Ergebnissen der Testdurchläufe wurde jeweils der Mittelwert gebildet. Die sich daraus ergebenen Mittelwerte werden zum Vergleich der Indexstrukturen verwendet. Außerdem erfolgte vor den Testdurchläufen eine „Aufwärmungsphase“ mit mehreren Testdurchläufen, da Java eine automatische Inline Optimierung durchführt [1].
In dieser Arbeit wurden die Indexstrukturen bezüglich ihrer Suchzeiten bei unterschiedlicher Anzahl von k-nächsten Nachbaren, Dimensionen und Datensätze untersucht. Eine solche Untersuchung wurde für die Intra- und Inter-Indexstrukturvergleiche durchgeführt. Die Ergebnisse dieser Vergleiche werden in den beiden nachfolgenden Abschnitten vorgestellt.

D. Intra-Indexstrukturvergleiche

Nachfolgend werden die Ergebnisse der Intra-Indexstrukturvergleiche vorgestellt. Diese Ergebnisse werden als Grundlage zur Erstellung der Parameterprofile der Indexstrukturen für die Inter-Indexstrukturvergleiche verwendet. Somit spiegeln die Parameterprofile die Ergebnisse der Intra-Indexstrukturvergleiche wieder und zeigen tabellarisch die jeweils besten Parametereinstellungen der Indexstukturen bei unterschiedlichen Dimensions- und Datensatzanzahlen auf.

Pyramidentechnik: Beim Intra-Indexstrukturvergleich wurden zuerst eine kNN Anzahl von zehn verwendet. Die Tabelle I zeigt das Parameterprofil beziehungsweise die Ergebnisse des Intra-Indexstrukturvergleiches

[2]http://www.eclipse.org, letzter Zugriff: 26.06.2010

der Pyramidentechnik. Als Parameter für die Evaluierung der Pyramidentechnik wurde die Anzahl der sphärischen Pyramidenstümpfe festgelegt. Anhand dieses Parameterprofils lässt sich die Annahme ableiten, dass bei einer höheren Dimensionsanzahl weniger sphärische Pyramidenstümpfe ratsamer sind. Außerdem kann geschlussfolgert werden, dass bei einer höheren Anzahl an Datensätze mehr sphärische Pyramidenstümpfe vorteilhaft sind. Jedoch gab es bei den Tests bezüglich der Pyramidentechnik eine Anomalie bei 100 Dimensionen, welche unsere Annahme nicht unterstützt. Der Pyramidentechnik wurde für die Suchanfrage bei 15 kNN (Tabelle II) und für eine Punktanfrage (Tabelle III) evaluiert. Durch die Evaluierungsergebnisse mit 15 kNN können die vorher getätigten Annahmen bestütigt werden. Jedoch stützen die Ergebnisse der Punktanfrage nur die Annahme, dass bei steigender Dimensionsanzahl weniger Pyramidenstümpfe empfehlenswerter sind.

Pyramiden-stümpfe		Dimensionen 10	20	50	100	200	1.000
Daten-sätze	**1.000**	3	4	1	3	1	1
	10.000	3	3	1	2	1	1
	100.000	8	3	4	6	4	2

Tabelle I
PARAMETERPROFIL DER PYRAMIDENTECHNIK BEI KNN MIT K=10.

Pyramiden-stümpfe		Dimensionen 10	20	50	100	200	1.000
Daten-sätze	**1.000**	2	3	1	1	1	1
	10.000	6	2	3	3	5	4
	100.000	8	3	5	5	3	3

Tabelle II
PARAMETERPROFIL DER PYRAMIDENTECHNIK BEI KNN MIT K=15.

Pyramiden-stümpfe		Dimensionen 10	20	50	100	200	1.000
Daten-sätze	**1.000**	8	1	8	3	1	5
	10.000	8	8	2	5	1	1
	100.000	2	6	1	6	3	8

Tabelle III
PARAMETERPROFIL DER PYRAMIDENTECHNIK BEI PUNKTANFRAGEN.

LPC: Der Parameter für das LPC-File, welcher für die Intra-Indexstrukturvergleiche relevant ist, ist die Anzahl der Dimensionsteilungen. Das in der Tabelle IV angegebene Parameterprofil wurde für ein k von 10 erstellt. Mittels dieser Ergebnisse der Intra-Vergleiche des LPC-File für k=10 wurde ersichtlich, dass eine höhere Dimensionsteilung zielführender ist. Die Ergebnisse des Intra-Indexstrukturvergleichs für 15 kNN und der Punktanfrage für das LPC-File wurden in den Tabellen VI und V abgebildet. Für die Vergleiche bei 15 kNN und bei der Punktanfrage wurde ebenfalls geschlussfolgert, dass eine höhere Dimensionsteilung vorteilhaft ist. Allerdings kann diese Annahme für die Punktanfrage nicht konstant über alle Vergleichsparameter beibehalten werden.
Des Weiteren wurde keine Evaluierung mit 1.000 Dimensionen durchgeführt, da laut der Implementierung das Verhalten des LPC-Files der sequentiellen Suche entsprechen würde. Jedoch ist das LPC-File mit deutlich erhöhtem Rechenaufwand verbunden und würde die Antwortszeit um ein vielfaches erhöhen.

Dimensions-aufteilung		Dimensionen 10	20	50	100	200	1.000
Daten-sätze	**1.000**	16	30	24	30	24	-
	10.000	30	16	16	30	30	-
	100.000	12	8	4	24	30	-

Tabelle IV
PARAMETERPROFIL DES LPC-FILES BEI KNN MIT K=10.

Dimensions-aufteilung		Dimensionen 10	20	50	100	200	1.000
Daten-sätze	**1.000**	30	16	30	30	30	-
	10.000	12	30	30	30	24	-
	100.000	30	16	24	24	30	-

Tabelle V
PARAMETERPROFIL DES LPC-FILES BEI KNN MIT K=15.

Dimensions-aufteilung		Dimensionen 10	20	50	100	200	1.000
Daten-sätze	**1.000**	16	30	16	30	30	-
	10.000	30	16	6	30	8	-
	100.000	12	8	30	24	30	-

Tabelle VI
PARAMETERPROFIL DES LPC-FILES BEI PUNKTANFRAGEN.

R-Baum: Der R-Baum wurde in dieser Arbeit nicht vorgestellt, da dieser bereits vom Vorgängerprojekt näher erläutert wurde. Im Rahmen dieser Arbeit wurde der R-Baum aus dem GeoCraft Open-Source Framework[3] übernommen und in das Framework integriert. Die folgenden Tabellen zeigen die Parameterprofile beziehungsweise die Evaluierungsergebnisse der Intra-Indexstrukturvergleiche für ein kNN von 10 (Tabelle VII), 15 (Tabelle VIII) und für eine Punktanfrage (Tabelle IX). Die erforderlichen Parameter wurden in diesen Tabellen

[3]http://wush.net/trac/geocraft/browser, letzter Zugriff: 30.06.2010

als Tupel angegeben. Diese Tupel beinhalten die minimale und maximale Anzahl der Punkte einer Seite. Aus den Ergebnissen der Evaluierung ließen sich keine Annahmen über das Verhalten des R-Baums erstellen, da das Verhalten nicht nachvollziehbar war.

	(min,max)	Dimensionen					
		10	**20**	**50**	**100**	**200**	**1.000**
Daten-sätze	**1.000**	(2,8)	(1,6)	(5,10)	(3,9)	(5,10)	(3,6)
	10.000	(2,4)	(3,9)	(5,10)	(2,8)	(3,6)	(1,6)
	100.000	(3,6)	(2,8)	(2,8)	(5,10)	(5,10)	(5,10)

Tabelle VII
PARAMETERPROFIL DER R-BAUM BEI KNN MIT K =10.

	(min,max)	Dimensionen					
		10	**20**	**50**	**100**	**200**	**1.000**
Daten-sätze	**1.000**	(1,6)	(4,12)	(3,9)	(3,6)	(4,12)	(2,8)
	10.000	(2,4)	(2,4)	(4,12)	(2,4)	(3,9)	(4,12)
	100.000	(1,3)	(1,3)	(5,10)	(3,6)	(1,3)	(1,3)

Tabelle VIII
PARAMETERPROFIL DES R-BAUM BEI KNN MIT K =15.

	(min,max)	Dimensionen					
		10	**20**	**50**	**100**	**200**	**1.000**
Daten-sätze	**1.000**	(2,8)	(1,6)	(3,6)	(1,3)	(3,6)	(3,9)
	10.000	(2,4)	(5,10)	(5,10)	(3,6)	(3,9)	(3,9)
	100.000	(3,6)	(2,8)	(2,8)	(5,10)	(3,6)	(2,8)

Tabelle IX
PARAMETERPROFIL DES R-BAUM BEI PUNKTANFRAGEN.

Weitere Indexstrukturen: Zur Evaluierung des VA-Files, welches vom Vorgängerprojekt implementiert wurde, ist als Parameter die Anzahl der Nachbarzellen erforderlich. Auf einen Intra-Vergleich zur Festlegung der Anzahl der Nachbarzellen für das VA-File wurde im Rahmen dieser Arbeit verzichtet, da der Fokus dieser Arbeit nicht auf dem VA-File liegt. Außerdem war das uns zugängliche VA-File eine Eigenauslegung des Vorgängerprojektes und ein Beweis auf praktische Relevanz steht noch aus. Für die Inter-Indexstrukturvergleiche bezüglich des VA-Files wurde als Anzahl der Nachbarzellen der Wert zwei festgelegt, da dieser Wert als Standardwert vorgegeben ist.
Zudem werden zum prototypbasierten LSH und zum KD-Baum keine Parameterprofile angegeben, da diese Indexstrukturen in der vorliegenden Implementierung keine Einstellmöglichkeiten beziehungsweise Parameter besitzen.
Für das RABT-File wurde die Pyramide als Basisindexstruktur eingesetzt, da diese als Basisindexstruktur bei unseren Testdurchläufen mit dem RABT-File die bestmöglichen Ergebnisse lieferte. Bei der Evaluierung des RABT-File wurde herausgefunden, dass die Vergleichsparameter abhängig voneinander sind. Dies wurde festgestellt, da sich beispielsweise bei einer Veränderung der Parameter des RABT-File die Antwortzeit verbessert. Jedoch hat sich bei diesen Parametereinstellungen die Precision verschlechter. Wiederum gilt, dass mittels anderer Parameter eine Verschlechterung der Antwortzeit eine Verbesserung Precision zur Folge hatte.

E. Inter-Indexstrukturvergleiche

Bei den Inter-Indexstrukturvergleichen werden wir einen Vergleich der Indexstrukturen untereinander vornehmen. Des Weiteren wird durch die Nutzung der erstellten Parameterprofile gewährleistet, dass jeweils die optimalen Parameterkombinationen genutzt und demnach die besten Ergebnisse für die jeweilige Indexstruktur erzielt werden.
Alle aufgeführten Diagramme (Abbildung 6 - 14) stellen die Antwortzeiten der Indexstrukturen in der gleichen Reihenfolge dar. Aus diesem Grund gilt für die Diagramme die gleiche Legende, die in Abbildung 5 dargestellt wird. Zudem sei angemerkt, dass keine Evaluierung des LPC-File bei jeweils 1.000 Dimensionen stattgefunden hat. Symbolisch wird daher immer der maximal darstellbare Wert in dem jeweiligen Diagramm angegeben.
Zunächst werden wir die Auswertung der Testszenarien bezüglich der kNN-Anfragen vornehmen. In dem Bereich werden wir zusätzlich eine Unterscheidung der exakten und der approximierenden Verfahren durchführen. Bei den exakten Anfragen werden wir eine detaillierte Auswertung von kNN mit k=10 und k=15 vornehmen.

kNN Anfrage - approximierende Indexstrukturen: Bei den im Framework vorhandenen approximierenden Verfahren handelt es sich um das RABT-File und um das prototypbasierte LSH. Da das VA-File eine eigene Implementierungsvariante und nicht wie nach Weber et al. ein exaktes Verfahren ist [18], müssen wir dies in diesem Abschnitt berücksichtigen und diese Indexstruktur ebenfalls als approximierendes Verfahren behandeln. Die allgemeinen Ergebnisse der kNN Anfragen werden in den Diagrammen der Abbildungen 6 - 11 dargestellt. Die Indexstrukturen LSH und das VA-File liefern in allen Bereichen sehr gute zeitliche Ergebnisse, daher konnten sie im Verhältnis zu den anderen Verfahren kaum in den Diagrammen dargestellt werden. Das RABT-File bietet im Gegensatz dazu keinen vergleichbaren zeitlichen Vorteil zu den anderen approximierenden und gegenüber den exakten Verfahren. Dies liegt unter Anderem daran, dass die Indexstruktur als Basisindexstruktur die Pyramidentechnik, ein exaktes Verfahren, verwendet. Wie im Folgenden noch genauer aufgeführt wird, besitzt diese Basisindexsturktur keinen zeitlichen Vorteil bezüglich der approximierenden Verfahren. Aus diesem Grund kann das RABT-File keine vergleichbaren zeitlichen Vorteile wie andere approximierende Verfahren erzielen.

Da es sich bei den hier evaluierten Verfahren auch um approximierende Indexstrukturen handelt, werden bei der

Datensätzte	kNN	LSH	RABT	VA-File
1.000	**10**	0.19 - 0.36	0.71 - 0.84	1.0
	15	0.27 - 0.36	0.79 - 0.94	1.0
10.000	**10**	0.47 - 0.51	0.57 - 0.95	1.0
	15	0.37 - 0.66	0.67 - 0.97	1.0
100.000	**10**	0.32 - 0.47	0.53 - 0.66	1.0
	15	0.30 - 0.54	0.54 - 0.69	1.0

Tabelle X
PRECISION DER APPROXIMIERENDEN INDEXSTRUKTUREN.

kNN Anfrage nicht immer alle echten k nächsten Nachbarn gefunden, die durch die sequentielle Suche eindeutig gegeben sind. Demnach müssen die Ergebnisse in den Diagrammen (Abbildung 6 - 11) immer in Bezug zu der Tabelle X betrachtet werden, in dem der Prozentsatz der gefundenen echten k nächsten Nachbarn angegeben ist.
Das LSH-Verfahren ist eine Indexstruktur mit einer sehr schnellen Antwortzeit bei kNN Anfragen. Jedoch besitzt dieses Verfahren gemäß der Tabelle X eine relativ schlechte Precision. Dennoch kann festgehalten werden, dass die Precision von LSH umso besser ist, je mehr Datensätze und somit mehr Punkte in den durchsuchten Regionen vorhanden sind. Daher ist das Verfahren von Bedeutung, wenn schnell Teilergebnisse bezüglich der k nächsten Nachbarn von großen Datenmengen berechnet werden sollen. Im Gegensatz dazu liefert das RABT-File bei den zeitlichen Messungen deutlich schlechtere Werte als das LSH Verfahren, dafür ist die Precision im Allgemeinen deutlich höher. Des Weiteren konnten wir überraschend feststellen, dass das VA-File neben der schnellen Antwortzeit trotz der approximierenden Auslegung eine konstante Precision von 1.0 liefert. Dies kann unter Anderem an den durchgeführten Testszenarien liegen, bei den die verwendeten Daten gleichverteilt sind.

kNN Anfrage - exakte Indexstrukturen: Bei den exakten Verfahren ist eine genauere Betrachtung der Verhaltensweisen bei k=10 und k=15 notwendig. Im Allgemeinen ist eine schnellere Anfragebearbeitung der kNN Suche als im sequentiellen Vergleich erwünscht. In Abbildung 6, in der es sich um einen Datensatz von 1.000 Daten und ein k=10 handelt, war dieser Aspekt lediglich bis 50 Dimensionen von allen vorhandenen Indexstrukturen erfüllt. Schon bei einer Menge von 10.000 Datensätze (Abbildung 7) war dieser Aspekt selbst bei 20 Dimensionen nicht mehr gegeben. Bei der Messung der kNN mit k=10 und 100.000 Datensätze verzeichnete die sequentielle Suche von 10 bis 1.000 Dimensionen bessere Ergebnisse als jedes andere exakte Indexverfahren. Da die Verfahren KD- und R-Baum stark vom *Fluch der hohen Dimensionen* betroffen sind, steigt die Antwortzeit ab einer Dimension von 100, wie in den Abbildungen 6 - 8 ersichtlich, sehr stark an. Die besten allgemeinen Resultate unter den exakten Verfahren, im Vergleich zur sequentiellen Suche, wurden von der

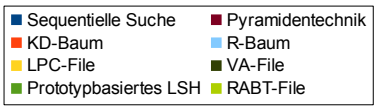


Abbildung 5. Legende der Tabellen.

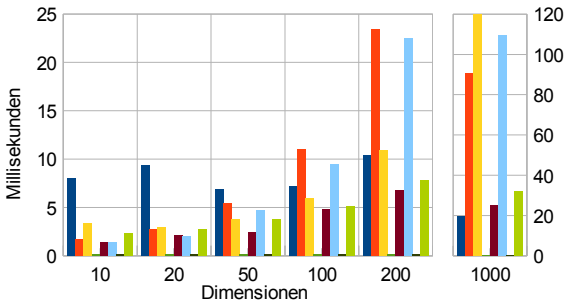


Abbildung 6. 1.000 Datensätze bei kNN mit k = 10.

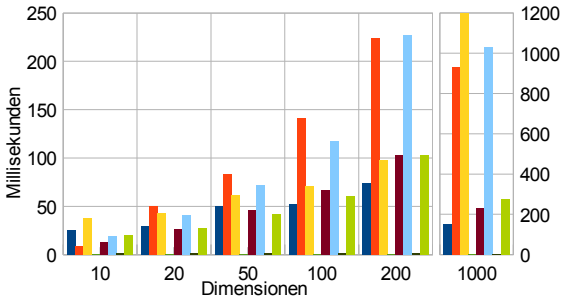


Abbildung 7. 10.000 Datensätze bei kNN mit k = 10.

Pyramidentechnik erzielt.

Bei der Abbildungen 9, in der die kNN Anfrage mit k=15 und der Datensatzgröße 1.000 abgebildet ist, wird auffällig, dass die Antwortzeiten der sequentiellen Suche überdimensional hoch sind. Vor allem im Vergleich mit der Abbildung 6 mit k=10 wird ersichtlich, dass der Aufwand der sequentiellen Suche bei einer wachsenden Anzahl von k schnell zunimmt. Da die anderen implementierten Indexverfahren diesbezüglich relativ stabile Antwortzeiten im Vergleich zu k=10 aufweisen, schneiden diese Verfahren in Bezug zur sequentiellen Suche deutlich besser ab. Dieses Verhältnis wird auch bei den Testszenarien mit 10.000 Datensätzen im Vergleich von k=10 (Abbildung 7) und k=15 (Abbildung 10) deutlich. Demnach stieg die Antwortzeit der sequentiellen Suche bei höherer kNN Anzahl deutlich an, während die Antwortzeit der anderen Indexstrukturen relativ stagnierte. Aus diesem Grund liefern sogar bis 200 Dimensionen alle anderen Indexstrukturen eine bessere Antwortzeit als die sequentielle Suche. Erst bei der Dimension

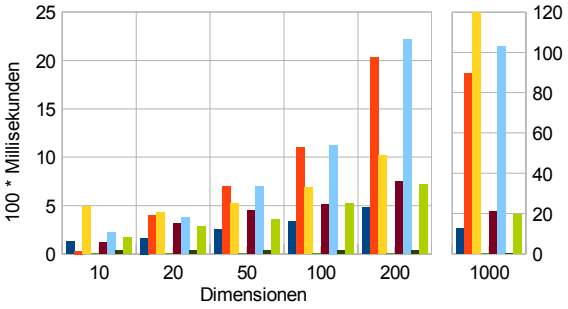


Abbildung 8. 100.000 Datensätze bei kNN mit k = 10.

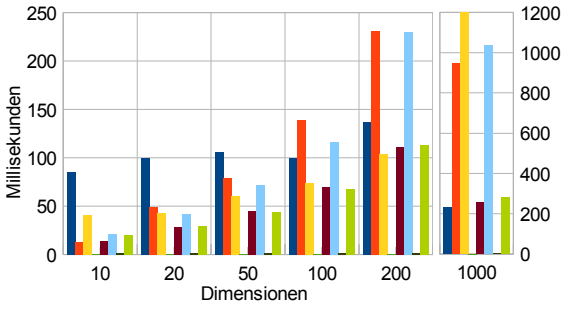


Abbildung 10. 10.000 Datensätze bei kNN mit k = 15.

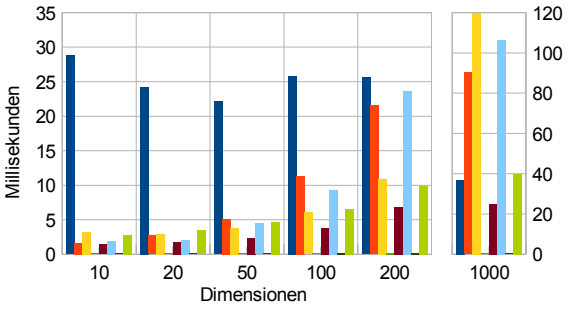


Abbildung 9. 1.000 Datensätze bei kNN mit k = 15.

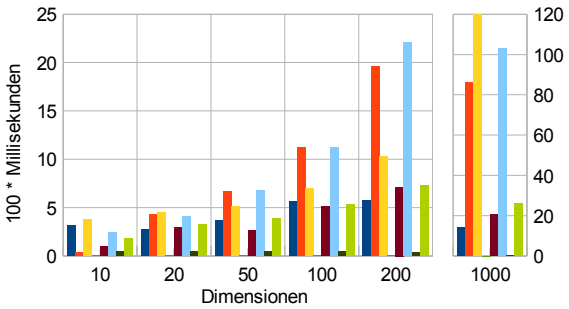


Abbildung 11. 100.000 Datensätze bei kNN mit k = 15.

von 1.000 ändert sich dieses Verhältnis wieder zu Gunsten der sequentiellen Suche, die in diesem Bereich die besten Ergebnisse erzielt. Steigt die Anzahl der Datensätze, wie in Abbildung 11 dargestellt, auf 10.000 an, besitzt die sequentielle Suche klare Vorteile bezüglich der Antwortzeit. Lediglich die Pyramidentechnik kann als einziges exaktes Verfahren zumindest bis 100 Dimensionen die Antwortzeiten der sequentiellen Suche unterbieten. Im Allgemeinen kann jedoch beim Vergleich von k=10 und k=15 eine deutliche Verbesserung der Antwortzeit aller Indexstrukturen gegenüber der sequentiellen Suche festgestellt werden. Es kann demnach die Annahme getroffen werden, dass eine weitere Erhöhung von k dieses Verhältnis noch deutlicher zum Ausdruck bringt.

Punktanfrage: Bei den Testszenarien bezüglich der Punktanfragen betrachten wir ebenfalls die Anzahl der Datensätze 1.000 (Abbildung 12), 10.000 (Abbildung 14) und 100.000 (Abbildung 14).
Bei 1.000 Datensätze benötigen vor allem die Indexstrukturen KD-Baum und das LPC-File in allen Dimensionen längere Antwortzeiten als die sequentielle Suche. Im Gegensatz dazu, erhielt die Pyramidentechnik in allen Dimensionen die schnellste Antwortzeit und war zudem in jedem Fall besser als die sequentielle Suche. Des Weiteren ergab auch die Messung der Antwortzeit vom LSH Verfahren und des RABT-File eine schnellere Antwortzeit als die sequentielle Suche. Aus dem Punktanfrageverhalten des R-Baum konnte festgestellt werden, dass diese Indexstruktur bei höheren Dimensionen ihren Geschwindigkeitsnachteil bezüglich der sequentiellen Suche ausgleichen kann.
Die Evaluierung der Datensatzgröße von 10.000 in Abbildung 13 ergab wiederum als schnellstes Indexverfahren die Pyramidentechnik. Zudem vergrößerte sich der relative Geschwindigkeitsvorteil dieses Verfahrens gegenüber der sequentiellen Suche um ein Vielfaches im Vergleich zu 1.000 Datensätzen. ähnlich gute Ergebnisse konnten auch durch das RABT-File, welches die Pyramidentechnik als Basisindexstruktur verwendet und durch das LSH-Verfahren erzielt werden. Eine deutliche relative Verbesserung bezüglich der Anfragezeit konnte zudem der KD-Baum erreichen, der auch um ein vielfaches besser als die sequentielle Suche war. Der R-Baum hingegen konnte seinen Geschwindigkeitsvorteil erst in den höheren Dimensionen erreichen, wohingegen das LPC-File in jedem Fall die schlechtesten Ergebnisse aufzuweisen hat.
In Abbildung 14 werden nun die Unterschiede zwischen den sehr schnellen und den langsamen Antwortzeiten extremer. Während die Pyramidentechnik, der KD-Baum, das RABT-File und das LSH-Verfahren kaum im Diagramm darstellbar sind, ergaben die sequentielle Suche, das LPC-File und das VA-File die schlechtesten Ergebnisse. Lediglich der R-Baum konnte ab 20 Dimensionen den Geschwindigkeitsvorteil in höheren Dimensionen untermauern.

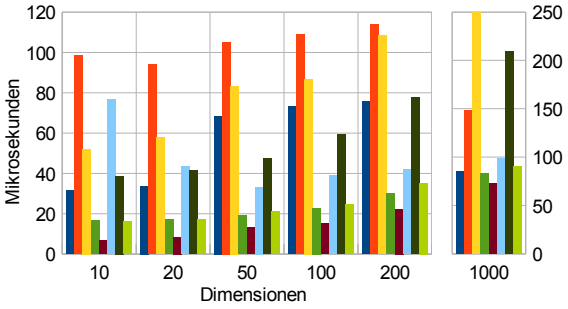


Abbildung 12. 1.000 Datensätze bei Punktanfragen.

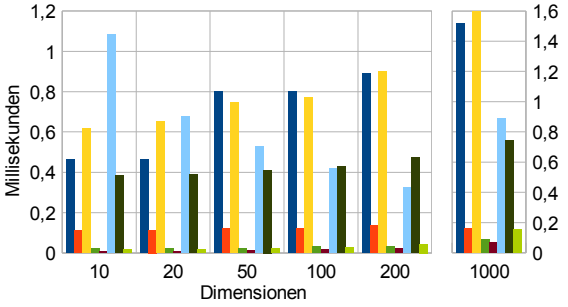


Abbildung 13. 10.000 Datensätze bei Punktanfragen.

Zusammenfassung der Inter-Indexstrukturvergleiche: Bei den durchgeführten Evaluierungen haben wir großen Wert auf die Vergleichbarkeit der Indexstrukturen gelegt. Die vorgestellten Parameterprofile der Intra-Indexstrukturvergleiche legten für uns die Grundlage für einen bestmöglichen Vergleich. Bei den Evaluierungen der kNN Anfragen erzielte vor allem das VA-File die besten und schnellsten Ergebnisse. Eine Untersuchung des Verhaltens bei geclusterten Daten ist jedoch anzuraten, da wahrscheinlich die Precision von 1.0 durch die Gleichverteilung der Daten zu begründen ist. Die Auswertungen der Evaluierung der Punktanfragen lieferten als bestes Ergebnis die Pyramidentechnik, die mit Abstand den Vergleich dominierte.

V. Verwandte Arbeiten

Im Bereich der multidimensionen Indexstrukturen existieren einige Übersichtsarbeiten, in denen verschiedene Indexstrukturen miteinander verglichen wurden. Beispiele dafür sind die Arbeiten von Böhm et al. [5], Gaede et al. [9] und Weber et al. [19]. In den ersten beiden Arbeiten wurden die verschiedenen Verfahren nur qualitativ mit ihren Vor- und Nachteilen gegenübergestellt. Von Weber et al. und in einigen anderen Arbeiten wurde eine quantitative Evaluierung durchgeführt. Jedoch beschränken sich diese Vergleiche auf eine sehr begrenzte Anzahl von Indexstrukturen. Außerdem sind die Evaluierungsergebnisse

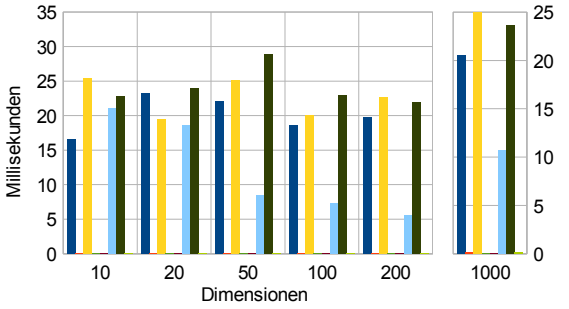


Abbildung 14. 100.000 Datensätze bei Punktanfragen.

in diesen Arbeiten vielfach nicht reproduzierbar.

VI. Zusammenfassung & Ausblick

In der vorliegenden Arbeit haben wir verschiedene Indexstrukturen implementiert und evaluiert. Durch die Evaluierung sind wir zur Erkenntnis gekommen, dass die Eigeninterpretation des VA-Files die besten Evaluierungsergebnisse bezüglich der kNN-Anfrage liefert. Im Bereich der Punktanfrage konnte die Pyramidentechnik die besten Ergebnisse aufweisen und lieferte zudem die besten Werte der exakten Verfahren bezüglich der kNN-Anfragen.

Die zukünftigen Projekte können sich mit verschiedenen offenen Themen beschäftigen, die noch nicht behandelt wurden. Unter Anderen können neue Clustermethoden entwickelt und eine Evaluierung mit geclusterten Datenmengen durchgeführt werden. Des Weiteren wäre eine Evaluierung der verschiedenen Indexstrukturen im Hinblick auf ihren Speicherverbrauch anzuraten. Eine Möglichkeit dies durchzuführen ist in der Java-Api durch die Klasse `java.lang.instrument.Instrument`[4] gegeben. Des Weiteren kann angemerkt werden, dass im Bereich der Intra-Indexstrukturvergleiche weiterer Forschungsbedarf besteht. Diesbezüglich können auch verschiedene Implementierungsvarianten einer Indexstruktur verglichen werden. Außerdem könnte angedacht werden, verschiedene Indexstrukturen miteinander zu kombinieren. Zusätzlich können verschiedene Indexstrukturen mit dem im R-Baum verwendeten *minimal bounding rectangles* erweitert werden.

Für die Implementierung der Pyramidentechnik nehmen wir eine statische Anzahl von Pyramidenstümpfe. Zur Optimierung könnte diese Anzahl in Zukunft durch die Menge der Punkte in der Pyramide dynamisch bestimmt werden.

Beim RABT-File verlieren wir durch die einfache Funktion der Dimensionsreduktion, die einer Summenbildung entspricht, viel Informationsgehalt. Eine bessere Funktion könnte dieses Problem verringern.

Nach Cha et al. kann auf der Grundlage vom LPC-File

[4] http://download.oracle.com/javase/6/docs/api/java/lang/instrument/Instrumentation.html, letzter Zugriff: 29.06.2010

der Grid Cell (GC)-Baum entwickelt werden [7]. Dieses Verfahren verspricht ein effizienteres Anfrageverhalten als das LPC-File und leidet ebenfalls nicht unter dem *Fluch der hohen Dimensionen*.

Das implementierte VA-File des Vorgängerprojektes entspricht nicht der Funktionsweise die von Weber et al. vorgestellt wurde [18]. Daher wäre eine Neuimplementierung sinnvoll. Jedoch sollte die aktuelle Version des VA-Files im Framework beibehalten bleiben, da dieses bei den vorgestellten Evaluierungen die besten Ergebnisse bezüglich der kNN-Anfragen lieferte.

Für die sequentielle Suche sollte eine optimierte Variante entwickelt werden, die beispielsweise durch die Parallelisierung die Antwortzeit der Anfragen verkürzt.

Ein weiteres Forschungsfeld könnte im Bereich von Anwendungen liegen, bei denen einzelne Features also Dimensionen der Datensätze wichtiger sind als andere. Eine Indexstruktur, die sich mit diesem Bereich befasst, ist der von King-ip Lin et al. vorgestellte Telescopic-Vector (TV)-Tree [12].

Des Weiteren könnte eine Evaluierung auf der Basis von realen Daten neue Erkenntnisse über das Verhalten der Indexstrukturen liefern.

LITERATUR

[1] The Java HotSpot Performance Engine Architecture. Website, 2011. Online erreichbar auf http://java.sun.com/products/hotspot/whitepaper.html, letzter Zugriff: 29.06.2010.

[2] Stefan Berchtold, Christian Böhm, and Hans-Peter Kriegal. The pyramid-technique: towards breaking the curse of dimensionality. *SIGMOD Rec.*, 27:142–153, 1998.

[3] Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel. The X-tree: An Index Structure for High-Dimensional Data. In *VLDB*, volume 22, pages 28–39, 1996.

[4] Christian Böhm. *Efficiently Indexing High-Dimensional Data Spaces*. PhD thesis, Ludwig-Maximilians-Universität München, 1998.

[5] Christian Böhm, Stefan Berchtold, and Daniel A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Comput. Surv.*, 33:322–373, September 2001.

[6] David Broneske, Denis Dietze, Maik Lampe, and Andreas Meier. Vergleich von Indexverfahren für hochdimensionale Datenräume. Technical report, Institut für Technische und Betriebliche Informationssysteme (ITI) der Otto-von-Guericke-Universität Magdeburg, Deutschland, 2010.

[7] Guang-Ho Cha and Chin-Wan Chung. The GC-tree: a high-dimensional index structure for similarity search in image databases. *IEEE Transactions on Multimedia*, 4(2):235–247, 2002.

[8] Guang-Ho Cha, Xiaoming Zhu, P. Petkovic, and Chin-Wan Chung. An efficient indexing method for nearest neighbor searches in high-dirnensional image databases. *IEEE Transactions on Multimedia*, 4(1):76–87, 2002.

[9] Volker Gaede and Oliver Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998.

[10] Antonin Guttman. R-trees: A Dynamic Index Structure for Spatial Searching. In *SIGMOD*, pages 47–57. ACM, 1984.

[11] Piotr Indyk and Rajeev Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *STOC*, pages 604–613, 1998.

[12] King ip Lin, H. V. Jagadish, and Christos Faloutsos. The TV-tree – an index structure for high-dimensional data. *VLDB*, 3:517–542, 1994.

[13] Norio Katayama and Shin'ichi Satoh. The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data*, pages 369–380. ACM Press, 1997.

[14] Dong-Ho Lee and Hyoung-Joo Kim. An Efficient Technique for Nearest-Neighbor Query Processing on the SPY-TEC. *IEEE Trans. on Knowl. and Data Eng.*, 15:1472–1486, 2003.

[15] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. Multi-probe LSH: efficient indexing for high-dimensional similarity search. In *VLDB*, volume 33, pages 950–961, 2007.

[16] G. M. Morton. A computer oriented geodetic data base and a new technique in file sequencing. In *IBM Germany Scientific Symposium Series*, 1966.

[17] Yasushi Sakurai, Masatoshi Yoshikawa, Shunsuke Uemura, and Haruhiko Kojima. The A-tree: An Index Structure for High-Dimensional Spaces Using Relative Approximation. In *VLDB*, volume 26, pages 516–526, 2000.

[18] Roger Weber and Stephen Blott. An Approximation-Based Data Structure for Similarity Search. Technical Report Zurich, Switzerland, 1997.

[19] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In *VLDB*, volume 24, pages 194 – 205. Morgan Kaufmann Publishers Inc., 1998.

[20] Rui Zhang, Beng Chin Ooi, and Kian-Lee Tan. Making the Pyramid Technique Robust to Query Types and Workloads. In *Proceedings of the 20th International Conference on Data Engineering*, ICDE '04, pages 313 – 324, 2004.

Secure Database Infrastructures

Fabian Benduhn
OvG-Universität Magdeburg
Fakultät für Informatik
Magdeburg
fabian.benduhn@st.ovgu.de

Albrecht Hultsch
OvG-Universität Magdeburg
Fakultät für Informatik
Magdeburg
albrecht.hultsch@st.ovgu.de

René Mäkeler
OvG-Universität Magdeburg
Fakultät für Informatik
Magdeburg
rene.maekeler@st.ovgu.de

***Zusammenfassung*—Datenbanken finden in vielen Bereichen der Wirtschaft und Wissenschaft Anwendung. In nahezu allen Firmen werden Datenbanken für verschiedene Anwendungsbereiche, zum Beispiel der Buchhaltung, verwendet. Daher ist die Sicherheit von Datenbanken ein wertvolles Gut, das es zu schützen gilt. Es gibt viele Gefahren, die die Sicherheit von Datenbanken beinträchtigen können. Um diesen Gefahren zu begegnen und die Sicherheit von Datenbanken zu erhalten kommen verschiedene Verfahren und Modelle zum Einsatz, um die Sicherheit von Datenbanken erhöhen. In dieser Arbeit werden wir verschiedene Modelle der *Zugriffskontrolle* betrachten. Danach werden wir die Gefahr der *SQL-Injection* erläutern und Möglichkeiten zur Detektion von SQL-Injections beschreiben. Dabei werden wir besonders auf die Anwendung von *Stored Procedures*, als Möglichkeit zur Verhinderung von SQL-Injections, eingehen.**

***Keywords*-database; security;**

I. Introduction

Datenbanken sind aus heutigen IT-Systemen nicht mehr wegzudenken. Sie werden in nahezu allen Anwendungsbereichen verwendet. Durch diese starke Verbreitung ergeben sich auch besondere Anforderungen an die Sicherheit. Eine der verbreitesten Bedrohungen, sind SQL-Injection-Angriffe (SQLIA), bei der ein Angreifer die SQL-Anweisungen, die an die Datenbank geschickt werden, manipuliert. Durch diese Manipulation kann er Informationen erhalten oder Aktionen ausführen, die normalerweise für ihn nicht zugänglich würen. Im Jahr 2010 zählten sie zu den häufigsten 10 Angriffen auf Datenbanksysteme [26]. Es existieren verschiedene Techniken zur Abwehr solcher Angriffe. Eine besondere Rolle spielen in diesem Zusammenhang *Stored Procedures*, die bei einer systematischen Abwehr von SQLIA als Hilfsmittel eingesetzt werden können. Eine *Stored Procedure* ist, wie der Name schon sagt, eine Prozedur die im Datenbank-Management-System (DBMS) gespeichert wird. Die Grundlage der Sicherheit einer Datenbank stellt jedoch die Rechteverwaltung dar. Durch sie wird festgelegt auf welche Ressourcen ein Nutzer zugreifen darf. Ohne eine solche Rechteverwaltung hätte also jeder Nutzer vollen Zugriff auf jegliche Ressourcen.

Beginnen werden wir mit einer Vorstellung der Rechteverwaltung innerhalb von Datenbanken bzw. drei grundlegender Modelle, die für diese genutzt werden. Dabei werden wir die Modelle bezüglich ihrer besonderen Eigenschaften vergleichen und insbesondere auf die Anwendung der Modelle in weit verbreiteten DBMS eingehen. Anschließend werden wir SQL-Injection-Angriffe näher betrachten. Es erfolgt eine mögliche Einteilung in verschiedene Klassen von SQL-Injection-Angriffen sowie eine Beschreibung verschiedener Möglichkeiten zur Detektion und Prävention. Im Anschluss werden wir einen Überblick über erweiterte Verfahren zur Erkennung und Abwehr von SQL-Injection-Angriffen untersuchen und evaluieren. Abschließend erfolgt eine Betrachtung verwandter Arbeiten, die im Umfang dieses Papiers nicht verwendet wurden, jedoch weitere Informationen zu diesem Thema zur Verfügung stellen, sowie ein Ausblick auf zukünftige Arbeiten.

II. Rechteverwaltung

Eine geeignete Rechteverwaltung stellt in jedem modernen DBMS die unverzichtbare Grundlage zur Gewährleistung einer umfassenden Sicherheit in einer Datenbankanwendung dar [11]. Alle weiteren Sicherheitsmechanismen können hauptsächlich darauf abzielen eine festgelegte Rechteverwaltung zu schützen. Aus diesem Grund betrachten wir im Folgenden zunächst grundlegende Zugriffsmodelle zur Vergabe von Rechten in einem DBMS. Auf Basis der vergebenen Rechte wird festgelegt, welche Aktionen durch bestimmte Benutzer durchgeführt werden können. Anschließend werden wir ausgewählte DBMS bezüglich der unterstützten Zugriffsmodelle vergleichen. Die Zugriffkontrolle überprüft die vom Benutzer ausgeführten Aktionen auf Basis der in der Rechteverwaltung hinterlegten Rechte. Die zwei Grundfunktionen der Zugriffskontrolle sind die Authentifizierung und die Autorisierung [23]. Bei der Authentifizierung wird die Identität eines Nutzers anhand der Daten, die der Nutzer dem DBMS zur Verfügung stellt (z.B. Nutzername und Passwort), ermittelt und überprüft. Die Autorisierung überprüft dann die Rechte dieses Nutzers und vergleicht sie mit den Rechten, die der Nutzer benötigt, um die gewünschte Aktion (z.B. Lesen oder Ändern von Datensätzen) durchzuführen. Falls der Nutzer alle nötigen Rechte besitzt wird die Aktion durchgeführt. Sollte dies nicht der Fall sein, so wird die

Aktion vom DBMS zurückgewiesen. Die zwei Schritte der Authentifizierung und Autorisierung lassen sich in folgendem Satz zusammenfassen [23]:

> Nachdem ich nunmehr ihre Identität kenne und weiß wer sie sind, lassen sie mich nachschauen ob ich ihnen erlauben werde zu tun, was sie gerne möchten

A. Zugriffsmodelle

Für die Rechteverwaltung können unterschiedliche Zugriffsmodelle verwendet werden. Im Folgenden werden wir die Funktionsweise der drei grundlegenden Zugriffsmodelle beschreiben, sowie ihre Vor- und Nachteile erläutern.

1) Discretionary Access Control (DAC): Das DAC-Modell basiert auf einer Menge von Regeln, sogenannten Autorisierungen, welche durch ein Tripel {S, O, A} definiert werden. Wobei S für ein Element aus der Menge der Subjekte bzw. Nutzern, O für das Objekt, welches verwendet werden soll und A für die Aktionen steht, die dem Nutzer S auf dem Objekt O erlaubt oder verboten sind. Diese Regeln werden vom Besitzer der Datei festgelegt und vom DBMS durchgesetzt. Durch den Umstand, dass für jede Kombination von Nutzer, Element und erlaubter bzw. verbotener Aktion ein solches Tripel existiert, ist dieses Modell sehr flexibel bezüglich der Rechtevergabe. Die Rechte können so mit einer minimalen Granularität bezüglich O und S vergeben werden. Aus dem selben Grund skaliert das Modell allerdings relativ schlecht. Bei vielen Nutzern und vielen Objekten ist es aufwändig zu erstellen, zu erhalten und zu schützen. Der tatsächliche Aufwand hängt hier von der Implementierung des Modells ab. Die gängigsten Möglichkeiten die Rechte-Informationen zu speichern sind die Verwendung einer Tabelle, einer Matrix oder einer Liste (Access-Control List). Bei der Verwendung einer Liste kann man sich darauf beschränken, diejenigen Regeln zu speichern, die wirklich relevant sind. Die restlichen Rechte ergeben sich implizit. Das Modell hat außerdem den Nachteil, dass es keinen Unterschied zwischen Subjekten bzw. Nutzern und Prozessen macht. Ein Prozess hat die gleichen Rechte, wie der Nutzer in dessen Namen er läuft. So kann zum Beispiel ein schädliches Computerprogramm Informationen für unbefugte Nutzer zugänglich machen, indem es den Inhalt von Dateien, auf die bestimmte Nutzer keinen Zugriff haben, in andere Dateien schreibt, auf die diese Nutzer zugreifen dürfen. Damit wird die Vertraulichkeit der Daten verletzt. Ein solcher Vorfall bleibt in diesem Fall unbemerkt. Eine Möglichkeit dieses Problem zu umgehen bietet das später vorgestellte *Mandatory Access Control-Modell (MAC)*.

2) Role-Based Access Control (RBAC): Das RBAC-Modell stellt eine Weiterentwicklung des DAC-Modells dar. Es basiert, genau wie das DAC, auf einer Menge von Regeln bzw. Tripeln {S, O, A}. Der Unterschied zum DAC-Modell besteht in der Menge der Subjekte S. Diese kann nun zusätzlich Gruppen enthalten, die mehrere Subjekte zusammenfassen. Dadurch ist es möglich Benutzergruppen festzulegen. Die Rechte einer solchen Benutzergruppe kann man auf diese Weise einfach und schnell einem Nutzer zuweisen. Auch das Ändern der Rechte ist einfacher, da man nur die Rechte der Benutzergruppe und nicht mehr die Rechte für jeden Nutzer einzeln ändern muss. Dadurch verringert sich der Aufwand beim Erstellen und Verwalten der Rechte erheblich, insbesondere wenn viele Nutzer die selben Rechte bekommen. Jedoch erlaubt das RBAC-Modell auch keine Differenzierung zwischen Nutzern und Prozessen, die mit den Rechten, der Benutzer agieren und damit ist die Vertraulichkeit von Daten auch hier nicht gesichert.

3) Mandatory Access Control (MAC): Das MAC-Modell ermöglicht einen Umgang mit der Problematik eines unberechtigten Informationsflusses zwischen Personen, indem es eine Menge von Sicherheitsstufen erstellt. In diese Sicherheitsstufen werden sämtliche Informationen und Nutzer eingeteilt. An dieser Stelle teilt sich das Verfahren in zwei Varianten auf, die zwei verschiedene Sicherheitsaspekte durchsetzen. Die erste Variante nach Bell-LaPadula [4] sichert die Vertraulichkeit von Daten. Es gelten hier zwei Regeln. Die erste Regel nennt sich *No-Read-Up*. Dies bedeutet, dass der Nutzer keine Dateien lesen kann, die über seiner Sicherheitsstufe liegen. Er kann also nur Dateien lesen, die auf seiner eigenen oder auf einer darunter liegenden Stufe liegen. Durch die zweite Regel *No-Write-Down* wird verhindert, dass der Nutzer Informationen erstellt oder verändert, die unter seiner eigenen Sicherheitsstufe liegen. Er kann also nur Informationen schreiben, die auf seiner eigenen oder einer darüber liegenden Sicherheitsstufe liegen. Durch den auf diese Weise kontrollierten Informationsfluss ist die Vertraulichkeit der Informationen gewährleistet. Der Informationsfluss dieser Variante wird in Abbildung 2 visualisiert. Die zweite Variante nach Biba [7] befasst sich mit der Integrität von Informationen. Auch hier gibt es zwei Regeln. Diese lauten hier jedoch *No-Read-Down* und *No-Write-Up*. Durch das *No-Read-Down* wird verhindert, dass ein Nutzer Informationen von einer Quelle lesen kann, die als weniger sicher als er selbst eingestuft wurde. Die Regel *No-Write-Up* verhindert, dass der Nutzer Informationen an eine höhere Sicherheitsstufe als der eigenen weitergeben kann. Somit wird verhindert, dass Informationen aus unsicheren bzw. weniger sicheren Quellen als gesicherte Informationen eingestuft werden können. Dargestellt wird der Informationsfluss dieser Variante in Abbildung 1. Auf diese Weise kann sowohl die Vertraulichkeit als auch die Integrität der Daten gesichert werden. Das MAC-Modell hat allerdings auch erheblich Nachteile. Es ist sehr strikt und lässt keine individuellen Rechte für bestimmte Nutzer zu. Es hat außerdem einen erheblichen Erstellungsaufwand, da für jede Information und jeden Nutzer eine Sicherheitsstufe definiert werden muss. Wenn man beide Varianten des MAC-Modells kombiniert, was auch möglich ist, muss man sogar für jede Information

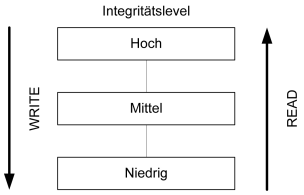


Abbildung 1. Modell nach Biba

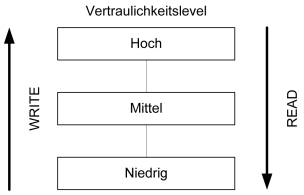


Abbildung 2. Modell nach Bell-LaPadula

und jeden Nutzer zwei Sicherheitsstufen erstellen. Der Vorteil ist jedoch das, sowohl die Vertraulichkeit als auch die Integrität der Daten gesichert wird.

B. Vergleich der Zugriffsmodelle

In Tabelle I zeigen wir, dass die drei vorgestellten Modelle unterschiedliche Vor- und Nachteile haben. Durch die individuelle Rechtevergabe des DAC-Modells wird die feinste Granularität bezüglich der Rechtevergabe erreicht. Da das RBAC-Modell auf dem DAC-Modell aufbaut, kann es einerseits dieselbe feine Granularität erreichen, anderseits kann es durch die Vergabe von Rechten an Benutzergruppen auch eine gröbere Granularität erreichen. Das MAC-Modell hat, da es nur auf den Sicherheitsstufen aufbaut, keine Möglichkeit Rechte individuell an Nutzer zu vergeben. Der Vorteil dieses Modells liegt im Schutz der Integrität und/oder der Vertraulichkeit der Daten.

Modell	Granularität	Integrität	Vertraulichkeit
DAC	fein	-	-
RBAC	fein + grob	-	-
MAC (Bell-LaPadula)	grob	-	+
MAC (Biba)	grob	+	-

Tabelle I
VERGLEICH DER ZUGRIFFSMODELLE

C. Vergleich von Datenbankmanagement-Systemen bezüglich der Zugriffsmodelle

Nachdem wir die drei grundlegenden Zugriffsmodelle gegenüber gestellt haben, untersuchen wir inwiefern sie in der Praxis zum Einsatz kommen. Dazu vergleichen wir drei ausgewählte DBMS bezüglich der implementierten Zugriffsmodelle. Für diesen Zweck haben wir PostgreSQL, MySQL und Oracle Database ausgewählt. PostgreSQL und Oracle Database bieten beide eine native Unterstützung für das RBAC-Modell und damit auch für das DAC-Modell an. MySQL unterstützt jedoch nur das DAC-Modell. Das MAC-Modell wird von keinem der ausgewählten DBMS explizit unterstützt. Eine einfache Version des MAC-Modells kann jedoch realisiert werden, indem jeder Tabelle eine Spalte für die Sicherheitsstufe hinzugefügt wird und auf Applikationsebene an jede Anfrage eine *Where* Klausel angehängt wird, mit der alle Daten herausfiltert werden, die dem Nutzer nicht zugänglich sein sollen. Eine andere Möglichkeit zur Umsetzung des MAC-Modells ist die Verwendung von Sichten. Mit Hilfe der definierten Sichten und einer zusätzlichen Spalte, welche Informationen über das Sicherheitslevel der jeweiligen Zeile beinhaltet, könnte man die *No-Read-Down* oder die *No-Read-Up* Regel einfach durchsetzen. Die *No-Write-Down* und *No-Write-Up* Regeln kann man (in PostgreSQL) mithilfe von *OnUPDATE* und *OnINSERT* Regeln realisieren.

III. SQL-INJECTION

Im vorherigen Kapitel haben wir festgestellt, dass durch eine geeignete Rechteverwaltung die Vertraulichkeit, sowie die Integrität von Daten sichergestellt werden kann. Allerdings bietet sie keinen umfassenden Schutz und es besteht die Möglichkeit für einen Angreifer die Rechteverwaltung zu umgehen. Eine der größten Bedrohungen in diesem Zusammenhang sind SQL-Injection-Angriffe. Wie bereits erwähnt wurde, gehörte die Art von Angriffen im Jahr 2010 zu den Top-10 der größten Schwachstellen in Webanwendungen und stellen folglich immer wieder eine Bedrohung an die Nutzer einer Anwendung dar, die sich über die sichere Verwahrung ihrer oft sensiblen Daten sorgen müssen. Bei einem SQLIA verändert der Angreifer Teile einer SQL-Anfrage durch das Hinzufügen neuer SQL-Schlüsselwörter oder -Operatoren. Dies ermöglicht dem Angreifer beispielsweise einen unauthorisierten Zugriff auf sensible Daten, um diese auszulesen oder zu verändern. Eine der häufigsten Ursachen für die Entstehung von Schwachstellen zur SQL-Injection ist die fehlende Validierung von Benutzereingaben im Programmcode. In der vorhandenen Literatur können verschiedene Möglichkeiten zur Klassifizierung von SQLIA gefunden werden. Zwei davon werden im Folgenden näher beschrieben. Zur Erläuterung der Angriffe an einem Beispiel wird eine einfache SQL-Anfrage, wie sie in Verbindung mit einem Login-Dialog auftreten könnte, in Abbildung 4 a) dargestellt. Auf Grundlage dieser Beispielanfrage wird die Funktionsweise einiger der vorgestellten SQLIA visualisiert.

A. Klassifikation nach Halfond

Eine Möglichkeit zur Klassifikation von SQLIA ist in Abbildung 3 dargestellt [13] . Es wird sowohl zwischen verschiedenen Angriffsarten als auch den Zielen der Angriffe unterschieden.

1) Tautologien: Tautologie-basierte Angriffe zielen auf die Umgehung eines Authentifizierungsprozesses ab, in Folge dessen der Angreifer Daten aus der Datenbank auslesen kann. Dazu wird in der `WHERE`-Bedingung einer Anfrage eine Tautologie, wie in Abbildung 4 b), beispielsweise

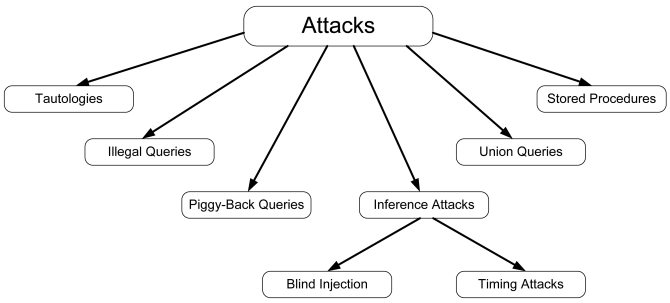


Abbildung 3. Klassifikation der SQL-Injection-Angriffe nach Halfond [13]

`OR 1=1`, verwendet. Da diese modifizierte Bedingung immer mit true evaluiert wird, kann nicht mehr entschieden werden, welche exakten Daten als Antwort auf die Anfrage zurückgegeben werden. Stattdessen werden alle Datensätze der entsprechenden Tabelle als Ergebnis geliefert und die Passwortabfrage für diesen Nutzer wurde folglich ausgehebelt. Die Zeichenfolge `--` stellt das Kommentarzeichen in diesem Anwendungsbeispiel dar und sorgt im Folgenden dafür, dass der Code, der hinter dieser Zeichenfolge steht, nicht weiter ausgewertet wird.

```
1 SELECT name, passw
2 FROM users
3 WHERE
4 username = '%username%'
5 AND
6 passw = '%password%';
```

(a) urspr. Anfrage

```
1 SELECT name, passw
2 FROM users
3 WHERE
4 username = ' ' OR 1 = 1 -- '
5 AND
6 passw = '%password%';
```

(b) Tautologie

```
1 SELECT name, passw
2 FROM users
3 WHERE
4 username = 'user1' '
5 AND
6 passw = '%password%';
```

(c) Illegale Anfrage

```
1 SELECT name, passw
2 FROM users
3 WHERE
4 username = ' ';
5 DROP TABLE users; -- '
6 AND
7 passw = '%password%';
```

(d) Piggy-Back-Anfrage

```
1  SELECT name, passw
2  FROM users
3  WHERE
4  username = ' ';
5  SELECT CASE
6  WHEN (username like 'a%')
7  THEN pg_sleep(10)
8  ELSE pg_sleep(0) END
9  FROM userdata; -- '
10 AND
11 passw = '%password%';
```

(e) Timing-Attack

```
1 SELECT name, passw
2 FROM users
3 WHERE
4 username = ' ' UNION
5 SELECT name, passw
6 FROM users; -- '
7 AND
8 passw = '%password%';
```

(f) UNION-Anfrage

Abbildung 4. Beispiele zu SQL-Injection-Angriffen

2) Illegale Anfragen: Die Verwendung illegaler Anfragen hat die Sammlung von Informationen über die Datenbank oder das zugrundeliegende Datenbankmanagementsystem zum Ziel. Dazu soll das DBMS veranlasst werden Fehlermeldungen zu erzeugen, um die darin enthaltene Information auszuwerten. Voraussetzung für diese Art von Angriffen ist, dass Fehlermeldungen für den Benutzer sichtbar dargestellt werden. Angreifer führen gezielt syntaktisch falsche Anfragen aus, die einen Fehler in der Datenbank verursachen. Ein Beispiel für eine solche illegale Anfrage ist in Abbildung 4 c) dargestellt. Das vom Angreifer zusätzlich eingefügte Zeichen `'` löst in diesem Fall einen Parser-Fehler in der Datenbank aus. Werden die Fehlermeldungen nun dem Angreifer zugänglich gemacht, können daraus Informationen über das zu Grunde liegende DBMS, Tabellen- oder Spaltennamen gewonnen werden, welche die Grundlage für weitere Angriffe bilden.

3) Piggy-Back-Anfragen: Diese Art der SQL-Injection dient zur Ausführung multipler SQL-Anfragen. Dazu versucht der Angreifer zusätzliche Anfragen mit dem originalen SQL-Statement zu koppeln. Beispielhaft ist dies in Abbildung 4 d) beschrieben. Dort wird die vorhergehenden Anweisung mittels `;` abgeschlossen und anschließend ein weiteres Statement an die SQL-Anfrage angefügt. In diesem Fall hätte die Ausführung des angefügten Statements die Löschung der gesamten Tabelle zur Folge. Diese Schwachstelle entsteht, wenn die Konfiguration der Datenbank multiple Anfragen innerhalb eines einzigen Strings erlaubt und die Benutzereingaben vor der Ausführung einer SQL-Anfrage unzureichend validiert werden. Ist diese Schwachstelle vom Angreifer identifiziert, ist es ihm möglich, jede Art von SQL-Kommando zu nutzen, inklusive der Ausführung von Stored Procedures.

4) Inference-Anfragen: Werden keine Fehlermeldungen von der Datenbank angezeigt, erlaubt diese Art des Angriffs aus dem Verhalten der Datenbank auf eine true/false-Anfrage Rückschlüsse über Datenbanktyp, Tabellenschema oder Spaltennamen zu ziehen. Dabei wird zwischen zwei Arten unterschieden. Mit Hilfe einer *Blind SQL-Injection* gewinnt der Angreifer Informationen aus der Darstellung der Webseite in Folge einer SQL-Injection. Wird eine Bedingung dabei zu true evaluiert, so wird die Webseite weiterhin normal angezeigt. Bei einer Evaluierung der Bedingung zu false, weicht die Darstellung der Seite jedoch von der normalen Darstellung ab. Die *Timing Attacks* hingegen erlauben dem Angreifer die Gewinnung von Informationen aus der Antwortzeit der Datenbank. Dazu werden datenbank-spezifische Funktionen verwendet, beispielsweise `pg_sleep()` in PostgreSQL, die die Antwort der Datenbank auf eine true/false-Anfrage je nach evaluiertem Ergebnis verzögern. Aus der Messung dieser Verzögerung lassen sich wiederum Rückschlüsse auf Tabellen- oder Spalteninformationen ziehen. Ein Beispiel für eine Anfrage im Rahmen einer solchen Timing-Attack ist in Abbildung 4 e) dargestellt.

5) UNION-Anfragen: Das Ziel der Verwendung von UNION-Anfragen ist der unauthorisierte Zugriff auf

Daten durch Umgehung der jeweiligen Authentifizierung. Angreifer koppeln das ursprüngliche SQL-Statement dazu mit einem weiteren Statement der Form `UNION SELECT <...>`. Das Ergebnis eines solchen Angriffs sind die zurückgelieferten Daten der ursprünglichen SQL-Anfrage inklusive der Daten aus der gekoppelten UNION-Anfrage. Abbildung 4 f) ist ein Beispiel für eine solche UNION-Anfrage zu entnehmen. Voraussetzung für die Ausführung einer solchen UNION-Anfrage ist die Kenntnis über die Anzahl der Spalten der Ergebnisdaten aus dem ursprünglichen SQL-Statement, da diese mit der Anzahl selektierter Spalten in der UNION-Anfrage übereinstimmen muss. Diese Informationen über die Tabellenstruktur llassen sich beispielsweise durch Verwendung der bereits vorgestellten *Illegalen Anfragen* erhalten.

6) Stored Procedures: SQLIA dieser Art zielen auf die Ausführung vorhandener Stored Procedures in der Datenbank ab. Voraussetzung dafür sind Informationen über das zu Grunde liegende DBMS. Die Kenntnis darüber erlaubt Angreifern die Ausführung bestimmter Stored Procedures, die vom Hersteller des DBMS mitgeliefert werden und gegebenenfalls mit einer Anwendung interagieren. Eine weitere Schwachstelle ist eine unzureichende Validierung der Benutzereingaben für eine Stored Procedure. Dieses Problem wird in Abschnitt IV genauer erläutert.

B. Klassifikation nach Nystrom

Eine weitere Möglichkeit zur Klassifikation von SQLIA nach Nystrom wird im Folgenden beschrieben [19]. Grundlage der Unterscheidung verschiedener Angriffe stellt die Informationen dar, die dem Angreifer zur Verfügung steht und auf Basis dessen der Angriff ausgeführt werden kann. Während Halfond die Angriffe nach der Struktur des injizierten Codes unterscheidet, klassifiziert Nystrom die verschiedenen Arten der Angriffe nach den Bedingungen, unter denen sie stattfinden.

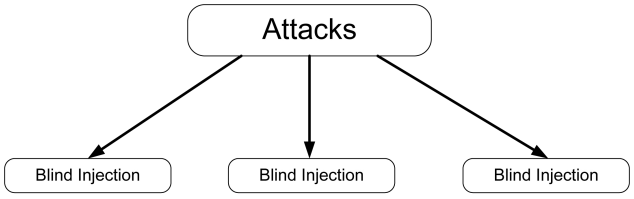


Abbildung 5. Klassifikation der SQL-Injection-Angriffe nach Nystrom [19]

1) Full View Attacks: In diesem Angriffsszenario stehen dem Angreifer ungefilterte Daten und Fehlermeldungen der Datenbank zur Verfügung. Jede Fehlermeldung ermöglicht es ihm, mehr Informationen über die zu Grunde liegende Datenbank sowie Tabellen- und Spaltenstruktur zu gewinnen.

2) Blind SQL-Injection: Fehlermeldungen werden in diesem Angriffsszenario vor dem Angreifer verborgen und von der Anwendung lediglich im Backend in einem Log verwaltet. Der Angreifer kann somit keine Informationen aus Fehlermeldungen gewinnen. Stattdessen werden true/false-Anfragen verwendet und auf Grundlage der, gegebenenfalls von Normalfall abweichenden, Darstellung einer Seite Rückschlüsse auf die Datenbank bzw. Daten gezogen.

3) Autonome SQL-Injection-Angriffe: Diese Angriffsart beschreibt autonome Würmer, denen es möglich ist ausführbaren Code in einer Datenbank zu speichern. Die Grundlage für diese Angrifftsart ist sowohl eine Schwachstelle für eine SQL-Injection (z.B. mangelhafte Eingabevalidierung) innerhalb einer Applikation sowie eine weitere Schwachstelle innerhalb von Datenbanken, die von dieser Applikationen genutzt werden. Dies ermöglicht es dem Wurm, dort Code zu speichern und auszuführen.

Nachdem wir eine Übersicht über mögliche SQLIA, sowie verschiedene Klassifizierungen gegeben haben, werden wir uns im restlichen Teil dieser Arbeit mit Methoden zur Detektion und Prävention von SQLIA auseinandersetzen.

IV. Grundlegende Techniken zur Abwehr von SQLIA

Im vorhergehenden Kapitel haben wir SQLIA als Bedrohung für Datenbankanwendungen erläutert. Jetzt stellen wir die grundlegenden Möglichkeiten zur Prävention und Detektion derartiger Angriffe dar und zeigen deren Grenzen auf.

A. Allgemeine Prinzipien

Bereits durch die Einhaltung einiger allgemeiner Prinzipien können viele mögliche Angriffe ausgeschlossen werden. Die Wichtigsten werden im folgenden kurz erläutert.

1) Fehlermeldungen unterdrücken: Wie wir in Abschnitt III gesehen haben, können Angreifer kritische Informationen durch die Analyse von Fehlermeldungen gewinnen. Um dies zu verhindern, empfiehlt es sich auftretende Fehlermeldungen nicht an den Benutzer weiterzuleiten [2]. Soll der Benutzer trotzdem über bestimmte Ereignisse informiert werden, so sollte man auf Applikationsebene Fehlermeldungen generieren, die möglichst keine Informationen über die Datenbank enthält.

2) Least Privileges: Das DBMS selbst sollte in einem Prozess laufen, der nur die absolut notwendigen Rechte zum Betrieb der Datenbank enthält [2]. Auf diese Weise kann der Schaden, der durch einen erfolgreichen Angriff auf die Datenbank entsteht, begrenzt werden.

3) BlackBox-Testing: Um Sicherheitslücken zu identifizieren kann die Anwendung getestet werden. Hierzu gibt es auch ein automatisiertes Verfahren, das systematisch nach Sicherheitslücken sucht und für Web-Anwendungen geeignet ist [15]. Ein hierzu geeignetes Open-Source Tool

ist z.B. WAVES[1]. Als Grundlage der Analyse dient eine Liste von bekannten Angriffsarten und -mustern. Hierbei sollte beachtet werden, dass Tests nur dazu dienen können Sicherheitslücken aufzudecken und nicht deren Abwesenheit feststellen zu belegen.

B. Defensive Programming

Durch eine konsequente Überprüfung der Benutzereingaben können SQLIA verhindert werden. Diese Vorgehensweise wird *Defensive Programming* genannt. Um eine ausreichende Sicherheit zu gewährleisten sind dabei verschiedene Prinzipien zu beachten [13].

1) Prüfung des Eingabetyps: Bereits durch eine Prüfung des Eingabetyps kann in vielen Fällen ein SLQIA ausgeschlossen werden. Soll etwa eine Zahl vom Benutzer eingegeben werden, können alle davon abweichenenden Anfragen zurückgewiesen werden.

2) Negative Pattern Matching: Eine weitergehende Überprüfung kann durch ein Pattern Matching durchgeführt werden. Beim sogenannten Negative Pattern Matching legt man fest, welche Art von Ausdrücken nicht in der Eingabe enthalten sein dürfen. Dies kann entweder durch ein simples Überprüfen auf Teilstrings geschehen oder z.B. mithilfe von regulären Ausdrücken. So kann man z.B. überprüfen ob die Eingabe den Teilstring *OR 1=1* enthält. Mit dieser Technik ist es allerdings schwierig alle möglichen SQLIA auszuschließen, da diese vielfältige Formen annehmen können.

3) Positive Pattern Matching: Beim Positive Pattern Matching legt man fest, wie gültige Eingaben aussehen müssen. In vielen Fällen kann auf diese Weise ein guter Schutz gewährleistet werden. Dazu ist es allerdings nötig die Möglichkeiten bei der Benutzereingabe stark zu begrenzen. Wenn der Benutzer allerdings große Freiheiten bei der Eingabe haben soll, z.B. beliebige Zeichenketten eingeben darf, stößt dieses Verfahren an seine Grenzen.

4) Eingabekodierung beachten: Mit Hilfe von bestimmten Verschleierungs-Techniken ist es oft möglich einen Pattern-Matching Mechanismus zu umgehen. Dazu kann bereits das gezielte Einfügen von zusätzlichen Whitespace-Zeichen ausreichen. Wenn beim Pattern Matching zum Beispiel nach der Zeichenkette *OR 1=1* gesucht wird, kann die Eingabe OR 1 = 1 nicht erkannt werden. Durch reguläre Ausdrücke ist diese einfache Art der Verschleierung leicht zu verhindern, es gibt aber auch einfallsreichere Varianten, z.B. die Verwendung von Hexadeximal-Kodierungen, statt dem eigentlichen Operator [1]. Je nach Eingabeverarbeitung, wird diese Kodierung genau wie ein regulärer Operator weiterverarbeitet.

5) Alle Eingabequellen beachten: Eine weitere Fehlerquelle, die zu Sicherheitslücken führt, ist das Übersehen von alternativen Eingabemöglichkeiten. Es sollte stets darauf geachtet werden diese mit einzubeziehen. So ist es z.B. denkbar, dass eine Anwendung einerseits über eine grafische Benutzeroberfläche bedienbar, aber auch über die Kommandozeile verwendbar ist. Diese alternative Eingabemöglichkeit, bietet eine weitere Möglichkeit für SQLIA, falls sie vom Entwickler nicht beachtet wird.

[1] http://waves.sourceforge.net

C. Stored Procedures

In diesem Abschnitt wird erläutert, wie durch Verwendung von Stored Procedures die Umsetzung der Abwehrtechniken unterstützt werden kann. Trotz der Vorteile, die uns Stored Procedures bieten, stellen wir fest, dass sie auch einige potentielle Schwachstellen besitzen.

1) Stored Procedures als Sicherheitsmaßnahme: Ein Problem der bisher erläuterten Abwehr-Techniken ist, dass sie alle auf Applikationsebene implementiert werden müssen. Das heißt es muss für jede Applikation sichergestellt werden, dass sie eine hinreichende Eingabeüberprüfung bereitstellt. Es ist außerdem schwierig festzustellen, ob alle Sicherheitslücken geschlossen sind, z.B. aufgrund unterschiedlicher Eingabequellen. Desweiteren gibt es auf Applikationsebene keine Möglichkeit Angriffe zu verhindern, die auf dem Kommunikationsmedium ansetzen, d.h. ein Angreifer kann die Informationen zwischen Applikation und Datenbank manipulieren und ggf. dort eine SQL-Injection durchführen. Da die Sicherheitsmechanismen auf Applikationsebene arbeiten, können solche Angriffe nicht erkannt werden. Durch die systematische Verwendung von Stored Procedures können diese Probleme vermieden werden. [5]. Alle vorgesehenen SQL-Anfragen werden dabei durch eine auf Datenbankebene definierte Menge von Stored Procedures bereitgestellt. Dadurch wird eine zusätzliche Abstraktionsebene geschaffen, welche die Applikationsebene von der eigentlichen Datenbankebene entkoppelt. So können sensible Informationen über das DBMS und die Tabellenstruktur vor dem Angreifer verborgen werden. Potenzielle Angriffe werden damit erschwert. Außerdem wird dadurch die Eingabeüberprüfung auf die Datenbankebene verlagert, was einen Angriff auf der Kommunikationsebene erschwert, der z.B. durch Auslesen und Manipulieren des Bitstroms durchgeführt werden kann. Dynamische Anfragen werden durch Übergabe von Parametern an die Stored Procedure realisiert. Innerhalb der Stored Procedure kann auf Grundlage der Parameter mit einer EXEC-Anweisung eine dynamische SQL-Anfrage generiert werden, die vom DBMS ausgeführt wird. Der Vorteil gegenüber der Generierung der Anfragen auf Applikationsebene liegt darin, dass die Anzahl der möglichen Schwachstellen durch die Stored Procedures beschränkt wird. Anstatt jede Applikation einzeln abzusichern, genügt es sicherzustellen, dass die definierte Menge an Stored Procedures eine ausreichende Sicherheit gewährleisten. Ein weiterer Vorteil ist, dass eine Manipulation der Anfrage auf dem Kommunikationsmedium ausgeschlossen werden kann, weil die SQL-Anfrage erst auf Datenbankebene erzeugt und überprüft wird.

2) Schwachstellen von Stored Procedures: Stored Procedures können das Absichern eines Datenbanksystems gegen SQL-Injection Angriffe erleichtern, allerdings bieten sie auch mögliche Schwachstellen, die Angriffe auf das DBMS erlauben. Einige Schwachstellen sind relativ klar erkennbar, über andere lassen sich kaum klare Aussagen treffen. Insbesondere können wir nicht ausschließen, dass es weitere Schwachstellen gibt. Die Literatur zu diesem Thema erlaubt es nicht diesbezüglich Aussagen zu treffen.

3) SQL-Injection Angriffe: Insofern eine Stored Procedure mittels einer EXEC-Anweisung eine dynamische SQL-Anfrage generiert, kann sie grundsätzlich für einen SQL-Injection-Angriff verwendet werden [25]. Die übergebenen Parameter müssen auf eine potenzielle SQL-Injection überprüft werden, um die Ausführung der Anweisung gegebenenfalls zu verhindern. Hierbei gelten die gleichen Prinzipien, wie bei der Überprüfung von Eingaben auf Anwendungsebene. Insbesondere kann eine Typüberprüfung bereits durch das DBMS bereitgestellt werden. Das Vorhandensein und der Umfang einer solchen automatischen Typüberprüfung ist DBMS-spezifisch und sollte bei der Implementierung der Stored Procedures berücksichtigt werden.

4) Potentielle Sicherheitslücken: Weitere Sicherheitslücken bei Stored Procedures sind vorstellbar, jedoch haben wir allerdings eine Literaturlücke diesbezüglich identifizert. Ein Grund für diese Lücke könnte darin bestehen, dass die technische Realisierung der Stored Procedures vom verwendeten DBMS abhängig ist. Allgemeingültige Ergebnisse können deshalb nur schwer zu gefunden werden. Eine potentielles Problem, das Stored Procedures betrifft sind Buffer-Overflow-Angriffe. Unsere Recherche bezüglich behobener Sicherheitslücken in kommerziellen Systemen (z.B. PostgreSQL[2] hat ergeben, dass diese Schwachstelle in der Vergangenheit häufig Standardprozeduren betroffen hat. Dies lässt uns vermuten, dass auch selbstdefinierte Prozeduren potenziell durch Buffer-Overflow-Angriffe angreifbar sein können. Eine genauere Analyse des Bedrohungspotentials erscheint hier sinnvoll, ist uns im Rahmen dieser Arbeit aber nicht möglich. Buffer-Overflow Angriffe sind aus anderen Bereichen der Software-Technik bekannt. Es sollte insbesondere geprüft werden, inwiefern Ergebnisse zu diesem Thema auf den Datenbankkontext übertragen werden können. Grundsätzlich gilt es die Integrität der definierten Stored Procedures sicherzustellen. Für einen umfassenden Schutz ist es also auch nötig, die gespeicherten Prozeduren vor unberechtigtem Zugriff zu schützen.

D. Probleme der Abwehrtechniken

Wir haben festgestellt, dass durch grundlegende Abwehrtechniken ein SQLIA auf Applikationsebene verhindert werden kann. Durch die Verwendung von Stored Procedures wird die Umsetzung erleichtert und die Überprüfung auf Datenbankebene verschoben. Dadurch wird eine zusätzliche Sicherheit gewährleistet. Aber auch durch die Verwendung von Stored Procedures, ist es in der Regel kaum möglich eine absolute Sicherheit vor SQL-Injection Angriffen zu realisieren. Die Umsetzung der Abwehrtechniken ist aufgrund der vielfältigen Angriffsmöglichkeiten problematisch. Das liegt darin begründet, dass die Wirksamkeit der verwendeten Schutzmechanismen vor allem von den Fähigkeiten und der Erfahrung der Entwickler abhängt. Dieses Problem verstärkt sich, je mehr Stored Procedures in einem System vorhanden sind. Aufwändig ist auch die nachträgliche Umstellung von bestehenden Systemen auf die Verwendung von sicheren Stored Procedures

In der Praxis lässt sich beobachten, dass viele Anwendungen Sicherheitslücken beinhalten, und zwar auch wenn diese grundsätzlich bekannt sind [16]. Das heißt zum Beispiel, dass nachdem eine Sicherheitslücke bekannt wurde diese zwar behoben wird, aber an weiteren Stellen im Code finden sich weiterhin äquivalente Lücken.

Eine Möglichkeit dieses Problem anzugehen, ist die automatisierte Erstellung von *Prepared Statements*, die in vielen Fällen ausreichend sind und analog zu Stored Procedures verwendet werden können. Dazu kann der *Prepared-Statement-Replacement*-Algorithmus verwendet werden [22]. Dabei wird der vorhandene Quellcode analysiert und eine Code-Struktur generiert, die geeignete *Prepared Statements* enthält.

Um einen besseren Schutz zu gewährleisten empfiehlt sich die Verwendung von erweiterten Techniken zur Prävention und Detektion von SQL-Injection Angriffen. Im nächsten Kapitel stellen wir eine Auswahl solcher Techniken vor, die einen Überblick über verschiedene Vorgehensweisen geben soll.

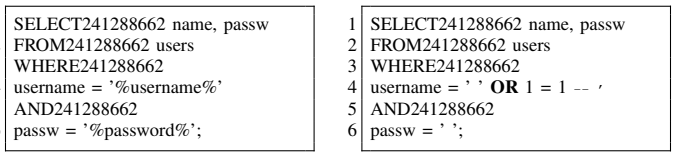
V. Erweiterte Techniken zur Abwehr von SQLIA

Wir haben festgestellt, dass einige grundlegende Techniken zur Abwehr von SQLIA existieren. Die Umsetzung der Techniken, ist in der Praxis allerdings problematisch. Aus diesem Grund stellen wir einige erweiterte Techniken vor, die vorallem zum Ziel haben, die Absicherung einer Datenbankanwendung zu automatisieren.

A. SQL-Randomization

Die *SQL-Randomization* ist eine Erweiterung der *Instruction Set Randomization* (ISR) [17]. Dieses Konzept erzeugt neue Instanzen der SQL-Sprache zur Verhinderung von SQLIA. SQL-Statements, welche in der Anwendung zum Einsatz kommen, werden dafür in eine spezielle Form gebracht. Ein zufällig ausgewählter Schlüssel wird dazu an jedes SQL-Keyword angefügt. Die Anwendung ist in der Lage, den SQL-Code nach der nutzerseitigen Eingabe von Daten zu de-randomisieren und jedes SQL-Keyword auf Vorhandensein des entsprechenden Schlüssels zu überprüfen.

[2]http://www.postgresql.org/support/security.html



```
1 SELECT241288662 name, passw
2 FROM241288662 users
3 WHERE241288662
4 username = '%username%'
5 AND241288662
6 passw = '%password%';
```

(a) Randomisiert

```
1 SELECT241288662 name, passw
2 FROM241288662 users
3 WHERE241288662
4 username = ' ' OR 1 = 1 -- '
5 AND241288662
6 passw = ' ';
```

(b) Manipuliert

Abbildung 6. Beispiel für SQL-Randomization

SQL-Schlüsselwörter, die in Folge einer SQL-Injection im Code erscheinen, besitzen diesen Schlüssel nicht und werden somit beim Parsen des SQL-Codes einen Fehler auslösen. Ein solcher Prozess soll beispielhaft in Abbildung 6 dargestellt werden. In Abbildung 6 a) zeigen wir das Ergebnis der Randomisierung einer Anfrage. Allen Schlüsselwörter wurde dementsprechend ein ausgewählter Schlüssel angefügt. In Abbildung 6 b) hingegen wird die randomisierte Anfrage nach einer SQL-Injection dargestellt. Da der Angreifer den Schlüssel der Randomisierung nicht kennt, kann bei der de-Randomisierung festgestellt werden, dass der Ausdruck `OR 1 = 1` ungültig ist, da das Schlüsselwort nicht über den entsprechenden Schlüssel verfügt. In der Literatur lassen sich bereits einige prototypische Umsetzungen dieser Technik finden. *PatchyRand* stellt eine Erweiterung des PostgreSQL JDBC-Treibers dar, welcher um eine Komponente zur SQL-Randomization modifiziert wurde. *SQLrand* hingegen beschreibt ein System, welches SQL-Randomization auf Grundlage eines spezifischen CGI-Scripts ausführt wird.

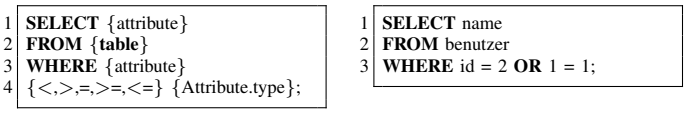
B. Kontext-sensitive Verfahren

Kontext-sensitive Verfahren sind Verfahren, welche bei einer Analyse einer Anfrage auch den Kontext mit betrachten, in welchem die einzelnen Teile der Anfrage stehen. Dadurch erhöht sich die Menge der Informationen und damit auch die Sicherheit mit der eine SQLIA richtig erkannt werden kann. Durch die Analyse des Kontextes erhöht sich jedoch auch die Komplexität der Implementation der Analyseverfahren. Als nächstes werden drei Kontext-sensitive Verfahren betrachtet.

1) Kontext-sensitive String-Evaluation (CSSE): Die CSSE ist ein Verfahren zur Abwehr von SQLIA, welche erst kurz vor der Auswertung der SQL-Anfrage ausgeführt wird [20]. Zuerst unterteilt das Verfahren die SQL-Anfrage in Code-Fragmente. Die nächste Aufgabe besteht darin, den Unterschied zwischen den Code-Fragmenten, die vom Entwickler als sicher eingestuft wurden und denen, die vom Benutzer stammen, zu erkennen. Dies realisiert es anhand von Code-Mustern oder Anfrage-Mustern, die vom Entwickler definiert wurden. Nachdem es nun die aktuelle Anfrage mit den Mustern abgeglichen hat, ist es in der Lage zu sagen, welche Anfrageteile vom Entwickler als sicher eingestuft worden sind und welche Daten vom Benutzer eingegeben wurden. Die Anfrageteile, die vom Nutzer stammen, werden als unsicher eingeordnet. Die zweite Aufgabe des Systems besteht nun darin alle unsicheren Anfrageteile weiter zu untersuchen bis sie als sicher oder unsicher eingestuft werden können. Bei der Anfrage in 7 b.) in würde das System die Werte *name*, *benutzer*, *id*, *2 OR 1 = 1* als Benutzereingabe erkennen. Diese Anfrageteile werden nun als unsicher eingestuft und überprüft. Nehmen wir nun an, dass vom Entwickler festgelegt wurde, dass der Nutzer keine SQL-Operatoren oder Befehle eingeben darf. Bei der Überprüfung bemerkt das System jedoch die SQL-Anweisung *OR*. Dies stellt eine Verletzung der Regeln dar. Das System blockiert diese Anfrage und gegebenenfalls meldet es diese Regelverletzung beim Administrator. Diese Anfrage und das Muster sind sehr einfach gehalten. Bei normalen Systemen mit mehreren verschiedenen, möglicherweise verschachtelten Anweisungen, lässt ist schnell erkennbar, dass eine solche Regelmenge bereits relativ groß werden kann.

2) Positive Tainting: Tainting bedeutet vergiften oder in unserem Fall markieren [14]. Das traditionelle Tainting ist das *Negative Tainting*. Es bedeutet dass ein negatives Merkmal, in unserem Fall ein schädlicher Codeabschnitt, markiert wird. Anhand der Markierung kann nun das Programm die Stelle desinfizieren. In unserem Fall würde eine SQL-Anweisung mit einer negativen Markierung nicht ausgeführt werden. Dies ist jedoch der traditionelle Ansatz, welcher sicher für die Detection von SQLIA als zu aufwändig erweist, da die Menge der schädlichen SQL-Anweisungen praktisch unendlich ist. Der zweite Ansatz beschreibt das *Positive Tainting*. Hierbei werden nicht die schädlichen Codefragmente markiert, sondern jene als sicher eingestuften. Die Menge aller sicheren Codefragmente wird vom Administrator in einer Menge von Mustern oder vom Entwickler mithilfe von fest codierten Codeblöcken definiert. Der Vorteil des Positiv Tainting ist, dass bei hinreichender Genauigkeit der Muster alle SQL-Injections erkannt werden und sich das Beschreiben der sicheren SQL-Anweisungen als verhältnismäßig einfach darstellt. Zusätzlich ist dieses Verfahren flexibel, da die Menge der Muster angepasst werden kann. Als Nachteil hingegen erwies sich die Tatsache, dass bei einer Vielzahl von zulässigen SQL-Anweisungen auch die Menge der als sicher definerten Blöcke wächst, was wiederum zur Folge hat, dass auch der Wartungsaufwand erhöht wird.

3) Syntax-Aware Evaluation: Bei der *Syntax-Aware Evaluation* wird die Syntax einer SQL-Anweisung analysiert [14]. Das Verfahren unterteilt die Anweisung in einzelne



```
1 SELECT {attribute}
2 FROM {table}
3 WHERE {attribute}
4 {<,>,=,>=,<=} {Attribute.type};
```

(a) Muster

```
1 SELECT name
2 FROM benutzer
3 WHERE id = 2 OR 1 = 1;
```

(b) Konkrete Anfrage

Abbildung 7. Beispiel für CSSE

Token und klassifiziert diese in SQL-Schlüsselwörter, Operatoren und Literale. Nun analysiert dieses Verfahren die einzelnen *Token* zusammen mit dem Kontext in welchem sie stehen. Der Vorteil dieses Verfahrens liegt darin, dass lediglich der Kontext eines *Tokens* betrachtet wird. Als Nachteil ergibt sich die Implementierung der Analyse, welche sehr komplex werden kann, da der Kontext ebenfalls betrachtet wird.

C. Syntaxanalyse-Verfahren

Eine SQL-Injection verändert die Struktur einer Anfrage. Anhand dieser Veränderung kann sie durch eine Syntaxanalyse erkannt werden. Hierzu existieren verschiedene Verfahren. Sie unterscheiden sich vor allem darin, wie eine Abweichung von den erlaubten Anfragestrukturen festgestellt wird. Einerseits variiert die Art und Weise wie ein Vergleich durchgeführt wird, andererseits auch der Zeitpunkt an dem die Menge der erlaubten Anfragen festgelegt wird. Tabelle II zeigt eine Übersicht dieser Verfahren.

1) Verwendung einer Erweiterten Grammatik: Bei diesem Verfahren werden Benutzereingaben durch Metazeichen gekennzeichnet [21]. Auf Grundlage einer einfachen Grammatik für die SQL-Befehle wird eine erweiterte Grammatik erzeugt. Die Menge der erlaubten Anfragen kann festgelegt werden, indem man nur für ausgewählte SQL-Befehle die Verwendung der Metazeichen zulässt. Sie können dadurch z.B. auf Literale oder bestimmte Operatoren beschränkt werden. Zur Laufzeit wird anhand dieser Grammatik über die Gültigkeit einer Anfrage entschieden.

2) Verwendung von Zustandsautomaten: Bei diesem Verfahren geht man davon aus, dass für den Datenbankzugriff ausschließlich Stored Procedures verwendet werden. Für jede Stored Procedure wird von einem Parser ein Zustandsautomat erstellt, der die Struktur der Anfrage beschreibt [25]. Für dynamische Anfragen, die zur Laufzeit auftreten, wird auch ein Automat erstellt und mit dem zugehörigen Automaten verglichen. Das besondere an diesem Verfahren ist, dass es vollautomatisiert ablaufen kann.

3) Parse Tree Validation: Bei der SQL Parse Tree Validation wird der Parse-Baum der ursprünglichen SQL-Anfrage mit der Anfrage verglichen, die durch Einsetzen der Benutzereingabe ensteht [9]. Der Parse-Baum repräsentiert die Struktur einer SQL-Anfrage. Die Stellen an denen die Benutzereingabe eingebunden werden kann, sind durch leere Blattknoten repräsentiert. Eine Benutzereingabe entspricht dem Einsetzen einer Konstanten in diesen Blattknoten. (Sollen die Eingaben SQL-Befehle enthalten dürfen, ist eine Anpassung des Verfahrens notwendig.) Enthält die Eingabe eine SQL-Injection, so wird aus dem Blattknoten ein innerer Knoten, d.h. die Struktur des Baumes verändert sich. Diese Veränderung kann durch einen Vergleich mit dem Parse-Baum der SQL-Anweisung nach der Benutzereingabe erkannt werden. Die Möglichkeit, dass ein SQL-Injection Angriff die Struktur des Parse-Baumes nicht verändert, wird dadurch ausgeschlossen, dass Kommentare als eigenständige Tokens im Baum behandelt werden. Dadurch kann ein Angreifer einen bestehenden Teil des Baumes nicht entfernen und ersetzen. Das Verfahren zeichnet sich besonders dadurch aus, dass es vollständig zur Laufzeit durchgeführt wird. Dadurch ist es möglich SQL-Anfragen, deren Struktur erst zur Laufzeit generiert wird, gegen SQLIA´s zu schützen. Eine Open-Source Java Implementierung steht zur Verfügung. [3]

4) Dynamic Query Matching: Bei diesem Verfahren wird eine Menge zulässiger SQL-Anfragen in einem Masterfile gespeichert [10]. Diese werden in einem XML Format (XSQL) gespeichert. Zur Laufzeit auftretende SQL-Anfragen werden mit Hilfe eines Parsers automatisch in das XSQL Format übersetzt. Es kann nun geprüft werden ob die Anfrage zulässig ist. Wenn es beim Vergleich zu einem exakten Treffer kommt, d.h. die Anfrage steht schon im Masterfile, wird sie an das DBMS weitergeleitet. Ansonsten wird festgestellt, ob die Anfrage eine gewisse Ähnlichkeit aufweist. Hierzu wird eine zeichen-basierte Distanz errechnet und mit einem Schwellenwert verglichen. Wird die Anfrage akzeptiert, so wird sie außerdem in das Masterfile aufgenommen. Ein Vorteil dieses Verfahrens ist, dass es relativ einfach zu implementieren ist. Der Vergleich der Anfragen kann durch das XML-Format durch einfache Matching-Algorithmen realisiert werden. Eine Schwierigkeit besteht allerdings in der Wahl eines geeigneten Schwellenwertes.

5) Lernende Verfahren: Die Menge der erlaubten Anfragen wird dadurch festgelegt, dass das System anhand von Beispielanfragen ein Modell aufbaut [24]. Davon abweichende Anfragen können identifiziert werden und stellen potenzielle Angriffe dar. Das Modell umfasst Anfragemodelle für jeden verwendeten Datentyp, die geeignet gewählt werden müssen. Für den Datentyp String werden beispielhaft folgende Anfragemodelle vorgeschlagen: Länge, Zeichenverteilung, Präfixe/Suffixe und String-Struktur. Es ist auch möglich datentyp-unabhängige Modelle zu entwickeln z.B. anhand der Bitrepräsentation. Allerdings funktioniert das Verfahren besser, wenn möglichst spezielle Modelle verwendet werden. Das System baut anhand dieser Anfragemodelle ein statistisches Modell für den jeweiligen Datentyp auf.

Nachdem die funktionalen Aspekte verschiedener, erweiterter Techniken zur Detektion und Prävention in diesem Kapitel vorgestellt wurden, möchten wir uns im Folgenden Kapitel mit der Evaluierung dieser Verfahren auseinandersetzen.

VI. Evaluierung

Bei der Evaluierung der in Abschnitt V vorgestellten Techniken werden zwei unterschiedliche Schwerpunkte gelegt. Zum einen sollen die Verfahren bezüglich ihrer Ergebnisse zur Detektion, Abwehr und Prävention von SQLIA

[3] http://www.cse.ohio-state.edu/paolo/software/javasqlguard.htm

Vergleichsstruktur	Definition	Anpassbarkeit
Grammatik [21]	Manuell	Nein
Zustandsautomaten [25]	Automatisch	Nein
Parsebaum [9]	Automatisch	Nicht nötig
XSQL-Format [10]	Manuell	Dynamisch
Anfragemodelle [24]	Implizit	Nein

Tabelle II
VERGLEICH VON SYNTAXANALYSE-VERFAHREN

verglichen werden, sowie auf Auswirkungen bezüglich der Performanz der Testsysteme. Der zweite Schwerpunkt liegt auf der Interpretation dieser Ergebnisse unter dem Gesichtspunkt von Umfang und Durchführung der Evaluierung einzelner Techniken. Es soll folglich untersucht werden, ob die vorgestellten Ergebnisse repräsentativ für das jeweilige Verfahren sind. Wir möchten dabei betonen, dass die im folgenden vorgestellten Ergebnisse von uns im Zuge dieser Arbeit lediglich aus der vorhandenen Literatur zusammengetragen wurden und nun miteinander verglichen werden.

A. Evaluierung bezüglich Ergebnisse der Verfahren

Die vorgestellte Verfahren basierend auf *SQL-Randomization*, *PatchyRand* [18] und *SQLrand* [8] werden nur eingeschränkt auf die unterschiedlichen Typen von SQLIA getestet. Während für *PatchyRand* keine Evaluierung vorhanden ist, ermöglicht *SQLrand* die Erkennung von SQLIA der Typen Tautologien, Piggy-Back-Anfragen, UNION- sowie Inference-Anfragen [13]. Im Worst-Case führt *SQLrand* zu einer Verzögerung von 6,5ms je ausgeführtem Query und besitzt somit nicht-signifikante Auswirkungen auf einen Großteil von Anwendungen. *PatchyRand* beschreibt die Auswirkungen auf Verarbeitungszeiten ebenfalls als gering, stellt jedoch keine Tests oder Werte zur Verfügung. Techniken, die auf der *Verwendung einer erweiterten Grammatik* [21] aufbauen erlauben die Erkennung aller Arten von von SQLIA ohne das Risiko von False-Positive-Ergebnissen zu erhöhen und unter geringen Auswirkungen auf die Performanz einer Anwendung. *Lernende Verfahren* [24] werden bezüglich vier konkreten Angriffsszenarien verglichen, welche unter anderem mit den hier vorgestellten Piggy-Back-Anfragen und Inference-Anfragen vergleichbar sind. Die Evaluierung ergab, dass das Verfahren diese Angriffe erkennt, jedoch zunächst unter der Einschränkung einer hohen False-Positive-Rate (0,37% aller gültigen Anfragen), welche sich jedoch nach einer exakten Anpassung des Systems für die individuellen Datentypen signifikant reduzieren ließ (0,013%). Desweiteren werden keine Angaben bezüglich möglicher Auswirkungen auf die Performanz gegeben. Techniken basierend auf dem *Dynamic Query Matching* [10] erwiesen sich bei ihrer Evaluierung dazu in der Lage, theoretisch alle Arten von SQLIA zu erkennen, allerdings sind diese Resultate für reale Anwendungen nur bedingt anwendbar. Weiterhin ist dieses Verfahren auf Grund der Abhängigkeit von Schwellenwerten anfällig für False-Positive-Ergebnisse und bezüglich möglicher Auswirkungen auf die Performanz eines Systems werden keine Angaben gemacht. Evaluierte Ergebnisse der auf *Parse Tree Validation* [9] basierender Techniken ergaben die Detektion von SQLIA der Typen Tautologien, UNION-Anfragen, Piggy-Back-Anfragen sowie allgemein der Verwendung von Kommentaren innerhalb einer SQL-Anfrage. Letztgenannt werden in der Literatur als ein Speziallfall für diese Technik vorgestellt. Während der Evaluierung wurden zudem Auswirkungen auf die Performanz einer Applikation in Abhängigkeit von den verwendeten Testdaten festgestellt. Dabei wurde zwischen standardmäßigen, relativ kurzen SQL-Anfragen und einem so genannten *Extreme Load* unterschieden. Je nach verwendeter Anfrage-Art unterscheidet sich auch Verzögerungszeit bei der Verarbeitung einer Anfrage, jedoch maximal um 3%. Die Ergebnisse von Verfahren, welche eine *Kontext-sensitive String-Evaluation (CSSE)* [20] durchführen, basieren stark auf der Wahl entsprechender Muster, da die Erkennung von Angriffen auf Grundlage von Abweichungen bezüglich entsprechender Muster geschieht. Gleiches gilt für False-Positive-Ergebnisse, welche sich durch eine geeignete Wahl von Mustern stark reduzieren lassen. Auswirkungen auf die Ausführungszeit befinden sich in einem nicht-signifikanten Bereich, da eine Anfrage, welche eine Benutzereingabe verwendet, lediglich mit dem entsprechenden Muster verglichen werden muss.

B. Bewertung bezüglich von Umfang und Qualität der Evaluierung

SQLrand wurde zunächst auf einem von den Entwicklern selbst erstellten Testsystem getestet. Anschließend wurden weitere Tests auf bestehenden Anwendungen, wie beispielsweise phpBB v2.0.5, einem Open-Source Bulletin Board [4], durchgeführt. *PatchyRand* hingegen wird in keinem konkreten Testszenario untersucht. Die Verfahren basierend auf der *Verwendung einer erweiterten Grammatik* wurden mit Hilfe von Referenzdaten evaluiert, die von Forschern auf Grundlage verschiedener, bestehender Anwendungen ermittelt wurden. *Lernende Verfahren* wurden auf Grundlage von Daten, die durch die Simulation eines Besuchs einer Webseite gesammelt wurden, getestet. Für jede der drei Phasen zum Trainieren des Modells, zum Festlegen von Schwellwerten sowie für die Analyse der False-Positive-Rate wurden auf diese Art Datensätze gesammelt. Die vorhandenen Informationen zu Verfahren basierend auf dem *Dynamic Query*

[4] https://www.phpbb.de/

Matching hingegen ermöglichen keine Rückschlüsse auf das Testsystem oder die Testdaten, die für eine Evaluierung zum Einsatz komment. Dementsprechend sind die Ergebnisse der vorgestellten Evaluierung nur eingeschränkt verwendbar. Diese Nachvollziehbarkeit des Testsystems ist wiederum bei der Evaluiering der Verfahren, die die *Parse Tree Validation* zur Grundlage haben, gegeben. Zusätzlich wurden bei der Evaluierung verschiedene Typen von Testdaten, sowohl für normale Anfragen als auch für den *Extreme Load*, verwendet. Die Verfahren, welche auf der *Kontext-sensitiven String-Evaluation* basieren, wurden ebenfalls mit Hilfe des Open-Source Bulletin Board phpBB getestet um zu evaluieren, wie bekannte Schwachstellen in bestehdenden Anwendungen durch das jeweilige Verfahren behoben werden können.

VII. Verwandte Arbeit

Auf Grund von begrenzten Ressourcen war es nötig, im Rahmen dieser Arbeit eine Eingrenzung des Themenbereiches vorzunehmen. An dieser Stellen sollen jedoch Arbeiten angeführt werden, die sich mit den Themen befassen, die wir nicht behandeln konnten. Wir haben insbesondere Arbeiten nicht berücksichtigt, die sich sehr speziell mit konkreten DBMS beschäftigten und dadurch sehr von der Implementierung des DBMS abhängig waren.

A. Rechteverwaltung

Eine weiterführende Behandlung des Themas Rechteverwaltung findet sich bei Bertino [6]. Es werden insbesondere spezielle Datenbankmodelle, wie z.B. Objektorientierte-Datenbanken und XML-Datenbanken, betrachtet.

B. Angriffe auf DBMS

Wir haben den Schwerpunkt unserer Betrachtung auf SQLIA gelegt. Es existieren noch weitere mögliche Bedrohungen für Datenbankanwendungen. Eine Übersicht, die insbesondere auch Angriffe auf das Kommunikationsmedium und z.B. Denial of Service Angriffe mit einschließt findet sich bei Bertino [5].

Es sei auch noch einmal auf die Arbeit von Halfond verwiesen, die eine Übersicht über weitere spezielle Detektions- und Präventionstechniken für SQLIA gibt [13] .

Viele Beispiele für erweiterte SQLIA, speziell für SQL Server, finden sich in der Arbeit von Chris Anley [3].

C. Spezielle Sicherheitsaspekte

Unser Schwerpunkt lag in der Betrachtung von SQLIA, die grundsätzlich alle Sicherheitsaspekte wie Integrität, Authentizität und Vertraulichkeit bedrohen. Zur Sicherung der Datenintegrität, existieren Ansätze, die wir nicht betrachtet haben. Eine Anwendung des Clark-Wilson-Modells, welches die Integrität der Daten sichern soll, auf den Datenbankbereich findet sich bei Xiaocheng [12].

VIII. Zusammenfassung und Ausblick

In dieser Arbeit haben wir grundlegende Aspekte, die bei der Konzeption einer sicheren Datenbank-Infrastruktur zu beachten sind, vorgestellt. Zunächst haben wir eine Übersicht über vorhandene Modelle für die Zugriffskontrolle und Rechteverwaltung innerhalb von Datenbanken gegeben. Anschließend wurden verschiedene Arten von SQL-Injection-Angriffen vorgestellt und klassifiziert sowie allgemeine Prinzipien zur Erkennung und Prävention von SQLIA beschrieben. Ein weiterer Schwerpunkt waren die Stored Procedures, mit deren Hilfe die Sicherheit weiter verbesser werden kann. Wir haben jedoch festgestellt, dass die Verwendung von Stored Procedures ebenfalls Probleme mit sich bringt, da auch diese nicht frei von potentiellen Schwachstellen sind. Auf Grund einer von uns identifizierten Literaturlücke bezüglich dieses Themas konnten wir jedoch im Zuge dieser Arbeit nur einen Teil dieser Schwachstellen näher untersuchen. Nachdem wir festgestellt haben, dass ein umfassender Schutz durch die von uns beschriebenen, grundlegenden SQLIA-Abwehrtechniken schwierig umzusetzen ist, haben wir eine Reihe erweiterter Techniken vorgestellt, die die Erkennung und Prävention von SQLIA sowohl verbessern, als auch automatisieren sollen. Nach der Beschreibung dieser Verfahren haben wir diese bezüglich ihrer Ergebnisse sowie deren Repräsentativität evaluiert. SQLIA stellen sowohl gegenwärtig als auch in der näheren Zukunft ein großes Problem bei der Verwendung von Datenbanken dar. Auf Grund der in dieser Arbeit angedeuteten Vielfalt möglicher SQLIA besteht somit auch zukünftig Anlass zur weiteren Entwicklung neuer Methoden sowie Modifikation bestehender Techniken um SQLIA zu erkennen und abzuwehren. Der in dieser Arbeit identizifierte Mangel an wissenschaftlicher Literatur bezüglich Stored Procedures stellt ebenfalls einen Ausgangspunkt für weiterführende Arbeiten dar. Eine umfassende Analyse potenzieller Schwachstellen, sowie der Funktionsweisen eines Angriffs auf die jeweilige Sicherheitslücke und Vorschläge zur Erkennung, Abwehr und Prävention wären denkbar. Ein weiterer Aspekt zukünftiger Arbeiten könnte die Entwicklung einheitlicher Evaluationskriterien für Verfahren zur Abwehr und Prävention von SQLIA sein. Diese würden die Evaluierung und den Vergleich neuer Methoden vereinfachen und eine gemeinsame Grundlage für den Vergleich verschiedener Techniken bieten.

Literatur

[1] The sql injection and signature evasion. Tech. rep., Imperva - Data Security for the Data Center, 2007.

[2] Quick guide to sql injection attacks and defenses. Tech. rep., IT Security Research & Penetration Testing Team, March 2010.

[3] Anley, C. Advanced sql injection in sql server applications. Tech. rep., Next Generation Security Software Ltd., 2002.

[4] BELL, D. E., AND LAPADULA, L. J. Secure computer systems: Mathematical foundations and model. Tech. Rep. M74-244, The MITRE Corporation, 1973.

[5] BERTINO, E., BRUSCHI, D., FRANZONI, S., NAI-FOVINO, I., AND VALTOLINA, S. Thread modelling for sql servers: Designing a secure database in a web application. Tech. rep., Purdue University, 2005.

[6] BERTINO, E., AND SANDHU, R. Database security – concepts, approaches, and challenges. Tech. rep., Purdue University, West Lafayette, 2005.

[7] BIBA, K. J. Integrity considerations for secure computer systems. Tech. Rep. MTR - 3153, The MITRE Corporation, 1977.

[8] BOYD, S. W., AND KEROMYTIS, A. D. Sqlrand: Preventing sql injection attacks. In *In Proceedings of the 2nd Applied Cryptography and Network Security (ACNS) Conference* (2004), pp. 292–302.

[9] BUEHRER, G., WEIDE, B. W., AND SIVILOTTI, P. A. G. Using parse tree validation to prevent sql injection attacks. In *Proceedings of the 5th international workshop on Software engineering and middleware* (2005), pp. 106–113.

[10] DAS, D., SHARMA, U., AND BHATTACHARYYA, D. K. n approach to detection of sql injection attack based on dynamic query matching. *International Journal of Computer Applications 1* (2010), pp. 28–34.

[11] DI VIMERCATI, S. D. C., FORESTI, S., AND SAMARATI, P. Authorization and access control. In *Security, Privacy and Trust in Modern Data Management*, M. Petkovic and W. Jonker, Eds. Springer-Verlag, 2007.

[12] GE, X., POLACK, F., AND LALEAU, R. Secure databases: an analysis of clark-wilson model in a database environment. In *Advanced Information Systems Engineering - 16th International Conference, CAiSE 2004* (2004), pp. 7–11.

[13] HALFOND, W. G., VIEGAS, J., AND ORSO, A. A classification of SQL-Injection attacks and countermeasures. In *Proceedings of the IEEE International Symposium on Secure Software Engineering* (2006).

[14] HALFOND, W. G. J., ORSO, R., AND MANOLIOS, P. Using positive tainting and syntax-aware evaluation to counter sql injection attacks. In *Proceedings of the Special Interest Group on Software Engineering (SIGSOFT)* (2006), ACM Press, pp. 175–185.

[15] HUANG, Y.-W., HUANG, S.-K., LIN, T.-P., AND TSAI, C.-H. Web application security assessment by fault injection and behavior monitoring. In *Proceedings of the 12th international conference on World Wide Web* (2003), ACM, pp. 148–159.

[16] HUANG, Y.-W., YU, F., HANG, C., TSAI, C.-H., LEE, D.-T., AND KUO, S.-Y. Securing web application code by static analysis and runtime protection. In *WWW '04: Proceedings of the 13th international conference on World Wide Web* (2004), ACM, pp. 40–52.

[17] KC, G. S., KEROMYTIS, A. D., AND PREVELAKIS, V. Countering code-injection attacks with instruction-set randomization. In *Proceedings of the 10th ACM conference on Computer and communications security* (2003), ACM, pp. 272–280.

[18] LOCASTO, M. E., AND KEROMYTIS, A. D. Pachyrand: Sql randomization for the postgresql jdbc driver. Tech. rep., Department of Computer Science, Columbia University, 2005.

[19] NYSTROM, M. *SQL Injection Defenses*. O'Reilly Media, March 2007.

[20] PIETRASZEK, T., BERGHE, C. V., V, C., AND BERGHE, E. Defending against injection attacks through context-sensitive string evaluation. In *Recent Advances in Intrusion Detection (RAID)* (2005), pp. 124–145.

[21] SU, Z., AND WASSERMANN, G. The essence of command injection attacks in web applications. In *Conference record of the 33rd ACM SIGPLAN-SIGACT symposium on Principles of programming languages* (New York, NY, USA, 2006), vol. 41, ACM, pp. 372–382.

[22] THOMAS, S., WILLIAMS, L., AND XIE, T. On automated prepared statement generation to remove sql injection vulnerabilities. *Inf. Softw. Technol. 51* (2009), 589–598.

[23] TSOLKAS, A., AND SCHMIDT, K. *Rollen und Berechtigungskonzepte*. Vieweg+Teubner, 2010.

[24] VALEUR, F., MUTZ, D., AND VIGNA, G. A learning-based approach to the detection of sql attacks. In *Proceedings of the Conference on Detection of Intrusions and Malware & Vulnerability Assessment* (2005), pp. 123–140.

[25] WEI, K., MUTHUPRASANNA, M., AND KOTHARI, S. Preventing sql injection attacks in stored procedures. In *Australian Software Engineering Conference* (2006), pp. 191–198.

[26] WILLIAMS, J., AND WICHERS, D. The ten most critical web application security risks, 2010.

Implementierung und Evaluierung von Distanzfunktionen im Digi-Dak-Framework

Matthias Koch
Fakultät für Informatik
Otto-von-Guericke Universität
Magdeburg, Deutschland

Martin Tobies
Fakultät für Informatik
Otto-von-Guericke Universität
Magdeburg, Deutschland

Christoph Neubüser
Fakultät für Informatik
Otto-von-Guericke Universität
Magdeburg, Deutschland

***Zusammenfassung*—Neben der Wahl effizienter Indexstrukturen gewinnen Distanzfunktionen immer mehr an Einfluss im Bereich des Multimedia-Retrieval. Distanzfunktionen werden dazu verwendet, den Abstand zwischen zwei Objekten zu berechnen. Dies sollte gerade für Multimedia-Daten möglichst exakt dem menschlichen Wahrnehmungsvermögen entsprechen. Die vorliegende Arbeit beschäftigt sich mit der Beschreibung und Charakterisierung verschiedener Distanzfunktionen, um eine fundierte Grundlage für Wahl von Distanzfunktionen in zukünftigen Anwendungen zu bieten. Weiterhin wird exemplarisch eine Umsetzung und Evaluierung innerhalb des Digi-Dak-Frameworks vorgenommen. Die Umsetzung zielt dabei auf eine einfache Programmierschnittstelle sowie eine intuitive Benutzungsoberfläche. Auf Seiten der Evaluierung werden die Distanzfunktionen vor allem im Hinblick auf Präzision und Laufzeitverhalten mit verschiedenen Indexstrukturen genauer untersucht.**

***Keywords*-Distanzfunktionen, Hochdimensionale Strukturen, Multimedia-DBMS**

I. Einleitung

Diese Arbeit entstand im Rahmen des Forschungsprojekts *Digi-Dak* (für engl. *Digital Daktyloscopy*). Die digitale Daktyloskopie beschäftigt sich mit der Mustererkennung von Fingerspuren, um vor allem die kriminalistische Forensik zu unterstützen. Dabei werden die Fingerspuren zunächst erfasst und anschliessend miteinander verglichen. Zum Vergleich der Fingerspuren können unterschiedliche Merkmale (engl. Features) aus diesen Spuren extrahiert werden. Um Ähnlichkeitsanfragen durchführen zu können, werden diese Merkmale in einen multidimensionalen Vektorraum transformiert. Damit effiziente Zugriffe auf diesen hochdimensionalen Datenraum möglich sind, wurden in vorangehenden Arbeiten bereits verschiedene Indexstrukturen betrachtet und in Framework integriert. Über verschiedene Distanzfunktionen werden dann die Merkmale miteinander verglichen.

Da zu Beginn dieses Teilprojekts nur die Euklidische und die Manhattan-Distanz zum Vergleich von Merkmalen in das Framework integriert waren, beschäftigt sich diese Arbeit im Folgenden mit dem Vergleich weiterer Distanzfunktionen. In II werden daher zunächst Eigenschaften von Distanzfunktionen vorgestellt und auf Basis dieser Eigenschaften eine Unterteilung vorgenommen. Anschließend erfolgen in III eine Abgrenzung zu verwandten Arbeiten und in IV eine Beschreibung verschiedener Distanzfunktionen. In V wird dann zunächst das entwickelte Konzept für die Implementierung vorgestellt, bevor in VI eine Evaluation hinsichtlich der Rechendauer, Präzision und Aufrufe der Distanzfunktionen durchgeführt wird. Mit Hilfe der Evaluation soll festgestellt werden, inwiefern sich die Distanzfunktionen hinsichtlich der genannten Evaluierungskriterien auf den gegebenen Featurevektoren verhalten. Abschließend wird in VII eine kurze Zusammenfassung der gewonnenen Erkenntnisse und Ergebnisse vorgestellt.

II. Grundlagen

Distanzfunktionen werden, im Kontext von Multimedia-Datenbanksystemen, auf verschiedene Merkmale (engl. *Features*) angewendet. Damit kann zum Beispiel die Ähnlichkeit zweier Datenbankobjekte bestimmt werden. Im Allgemeinen dienen Distanzfunktionen der Approximierung der menschlichen Wahrnehmung. Die betrachteten Merkmale können in unterschiedlichen Datentypen vorliegen. Dabei unterscheidet man im Allgemeinen zwischen folgenden Feature-Datentypen [1]:

- *Punktdaten:* Ein Array fester Länge $[1..n]$ für alle Objekte in der Datenbank, wobei die Werte zum Beispiel in float oder int kodiert vorliegen und jeweils eine Dimension repräsentieren.
- *Wahrscheinlichkeitsverteilungen:* Beschreiben das Häufigkeitsverhalten disjunkter Intervalle, die beispielsweise durch Grauwert-Histogramme repräsentiert werden.
- *Sequenzen:* Eine Liste von Werten beliebigen Typs, um zum Beispiel eine Tondynamik zu beschreiben.
- *Binärdaten:* Ein Array fester Länge $[1..n]$ für alle Objekte in der Datenbank, wobei die Werte vom Typ *boolean* sind. Ein Beispiel hierfür ist eine Fähigkeits-Matrix für Mitarbeiter einer Firma, in der festgehalten wird, über welche Fähigkeiten ein bestimmter Mitarbeiter verfügt.
- *Intervall:* Ein Array fester Länge $[1..n, 1..n]$, wobei jeder Featurewert durch zwei Werte vom Typ *integer* oder *float* definiert wird, die eine untere und obere Schranke angeben. Als Beispiel hierfür könnte die Präsens über eine bestimmte Zeit von Mitarbeitern genannt werden.

Aufgrund der in diesem Projekt verwendeten Featuredaten wurden im Weiteren ausschließlich Punktdaten-

und Wahrscheinlichkeitsverteilung-basierte Distanzfunktionen betrachtet. Bei Distanzfunktionen, die auf Histogrammen bzw. Wahrscheinlichkeitsverteilungen arbeiten, werden die Featurewerte ebenfalls in Punktdaten umgewandelt. Das hat vor allem den Grund, dass Punktdaten universell sind und bereits mathematische Ansätze dafür bekannt sind.

Damit eine bestimmt Funktion oder Funktion als Distanzfunktion bzw. Distanzfunktion deklariert werden kann, muss diese vollständig oder partiell folgende Eigenschaften bzw. Axiome erfüllen[1]. Dabei seien X, Y, Z drei ungleiche Elemente einer Objektmenge O und d eine binäre Funktion $d(X,Y) \mapsto \mathbb{R}$, die den Abstand zweier Elemente zueinander berechnet:

- *Selbstidentität (SI):* Der Abstand eines Elements zu sich selbst ist 0.
  $\forall X \in O : d(X,X) = 0$
- *Positivität (POS):* Der Abstand zweier ungleicher Elemente ist größer als 0.
  $\forall X \neq Y \in O : d(X,Y) > 0$
- *Symmetrie (SYM):* Der Abstand von Element X zu Y ist gleich dem Abstand von Element Y zu X.
  $\forall X,Y \in O : d(X,Y) = d(Y,X)$
- *Dreiecksungleichung (DU):* Der Abstand von Element X zu Z ist kleiner oder gleich dem addierten Abstand von Element X zu Y zu Z.
  $\forall X,Y,Z \in O : d(X,Z) \leq d(X,Y) + d(Y,Z)$

Abhängig von den erfüllten Eigenschaften kann im Folgenden bestimmt werden zu welcher Klasse von Distanzfunktionen eine bestimmte Funktion zuzuordnen ist [1]:

- *Distanzfunktionen* müssen allen Eigenschaften einer Funktion genügen. Es müssen Selbstidentität, Positivität, Symmetrie und die Dreiecksungleichung gelten.
  $C_D : SI \wedge POS \wedge SYM \wedge DU$
- *Pseudo-Distanzfunktionen* müssen das Kriterium der Positivität nicht erfüllen; Abstände können also negativ sein.
  $C_{PD} : SI \wedge SYM \wedge DU$
- *Semi-Distanzfunktionen* erfüllen die Dreiecksungleichung nicht; der Abstand zweier Objekte X und Z kann also größer sein, als der addierte Abstand über X, Y und Z. Daraus folgt insbesondere, dass negative Abstände existieren.
  $C_{SD} : SI \wedge POS \wedge SYM$
- *Semi-Pseudo-Distanzfunktionen* verletzen sowohl die Positivitätseigenschaft als auch die Dreiecksungleichung.
  $C_{SPD} : SI \wedge SYM$

Diese Einteilung dient der besseren Differenzierung zwischen verschiedenen Distanzfunktionen. So kann beispielsweise festgestellt werden, wie sich das Fehlen verschiedener Eigenschaften auf die Performance der einzelnen Distanzfunktionen auswirkt und welche Klassen sich dementsprechend unter den gegebenen Bedingungen besser zur Berechnung von Ähnlichkeiten eignen. Neben den genannten Eigenschaften Selbstidentität, Positivität, Symmetrie und Dreiecksungleichung existieren noch weitere Eigenschaften, die der Vollständigkeit halber kurz erwähnt werden sollen. Da diese Eigenschaften für die Unterteilung der Distanzfunktionen keine Relevanz haben, wird auf sie nicht weiter eingegangen [1].

- *Translationsinvarianz:* Der Abstand von zwei Elementen bleibt erhalten, wenn diese um den selben Translationsvektor verschoben werden.
  $\forall X,Y : d(X,Y) = d(X+T, Y+T)$
- *Skalierungsinvarianz:* Der Abstand von zwei Elementen bleibt erhalten, wenn eines der Elemente um einen bestimmten Faktor skaliert wird.
  $\forall X,Y : d(X,Y) = d(X, S \cdot Y)$
- *Rotationsinvarianz:* Der Abstand von zwei Elementen bleibt erhalten, wenn diese um den selben Rotationsvektor gedreht werden.
  $\forall X,Y : d(X,Y) = d(R \cdot X, R \cdot Y)$

Zusätzlich zu den beschriebenen Eigenschaften kann für jede Distanzfunktion des Weiteren ein Einheitskreis dargestellt werden, der alle Punkte definiert, die zu einem bestimmten Referenzpunkt einen Abstand von 1 aufweisen. Der Referenzpunkt, zu dem der Abstand jedes weiteren Punktes berechnet wird, liegt dabei im Koordinatenursprung. Alle Punke auf oder innerhalb dieses Einheitskreises können als ähnlich angesehen werden. Ebenso können mit Hilfe des Einheitskreises die zuvor beschriebenen Eigenschaften von Distanzfunktionen abgelesen werden [1]. Beispielhaft ist der Einheitskreis der euklidischen Distanzfunktion in Abbildung 1 dargestellt. Der Referenzpunkt ist hierbei der Nullpunkt und der eigentliche Kreis bildet sich folglich aus allen Punkten, die einen Abstand von 1 zum Nullpunkt aufweisen. Dabei muss der Einheitskreis einer Distanzfunktionen nicht zwangsläufig kreisförmig sein, was bei einigen der in IV vorgestellten Distanzfunktionen ersichtlich wird.

III. Verwandte Arbeiten

Es existieren bereits verschiedene Arbeiten, die sich mit der Verwendung von Distanzfunktionen in hochdimensionalen Datenbanken beschäftigen. Dabei handelt es sich zumeist um Übersichtsarbeiten, die mehrere Distanzfunktionen miteinander vergleichen. Dazu zählen zum Beispiel Kokare *et al.* [2] und Bugatti *et al.* [3]. Erstere konnten eine signifikante Verbesserung in der Performance nachweisen, wenn die Canberra- oder Bray-Curtis-Distanzfunktion verwendet wurde. Verglichen wurde in diesem Fall mit der, in konventionellen Image-Retrieval-Anwendungen, oft verwendeten Euklidischen oder Mahalanobis-Distanzfunktion. Dabei wurde die Wahl einer anderen Distanzfunktion mit einer Vielzahl von Problemen begründet, die bei der Verwendung von Euklidischer und Mahalanobisc-Distanzfunktion auftreten. Als Hauptprobleme wurden hierbei vor allem die geringe

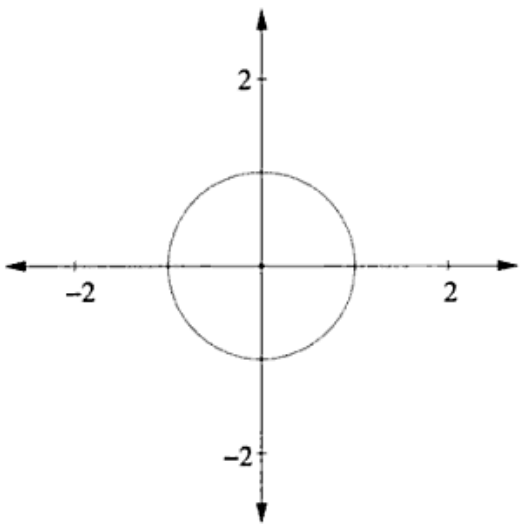


Abbildung 1. Einheitskreis der euklidischen Distanzfunktion [1]

Anfragegenauigkeit, sowie hohe Speicherkosten bei steigender Featureanzahl und Fehleranfälligkeit genannt. Durch die Verwendung der Canberra-Distanzfunktion, an Stelle der Euklidischen Distanzfunktion, konnte eine Performance-Verbesserung von 65% - 78% erreicht werden.

In der Arbeit von Bugatti et al. [3] wurden weitere Distanzfunktionen vorgestellt und das Zusammenwirken dieser Funktionen mit dem verwendeten Feature-Extraktor beschrieben. Neben der Canberra-Funktion wurden in dieser Arbeit auch Funktionen wie zum Beispiel Jeffrey Divergence und Chi Square betrachtet, die auf Grauwert-Histogrammen als Feature-Extraktor basieren. Wie bereits in Kokare et al. [2] schnitt die Canberra-Distanzfunktion für die beschriebenen Anwendungsfälle am besten ab. Dennoch wird konstatiert, dass trotzdessen oft die Minkowski-Distanzfunktionen verwendet werden, obwohl diese eine geringe Performance aufweisen.

Eine weitere wichtige Beobachtung besteht darin, dass Funktionen feature-spezifisch sind [4]. Die einzelnen Distanzfunktionen eigenen sich also entweder gut oder schlecht im Hinblick auf die Verwendung mit bestimmten Feature-Daten. Bei einer Sammlung zahlreicher verschiedener Features in einem Featurevektor besteht demnach die logische Schlussfolgerung, sogenannte Multi-Funktionen zu verwenden. So könnte beispielsweise ein Featurevektor aus zwei Teilen - Punktwerte und Histogrammwerte - bestehen. Eine entsprechende Multi-Funktion könnte die Distanz nun sowohl durch Anwendung der Canberra- als auch der Jeffrey-Divergence-Funktion auf die entsprechenden Teile des Featurevektors berechnen. Die Priorisierung der einzelnen Teildistanzen obliegt hierbei dem speziellen Anwendungsfall.

Die zentrale Problemstellung in Multimedia-DBMS besteht nicht nur in dem Ermitteln einer Distanzfunktion, welche die menschliche Wahrnehmung ausreichend immitiert, sondern auch in der Unterstützung hochdimensionaler Featurevektoren [5]. Dabei hat sich der häufig genutzte Einsatz der Hauptkomponentenanalyse (PCA) aufgrund von nicht-linearer Korrelationen als nicht sinnvoll erwiesen. Ein hybrider Ansatz verwendet daher neben der Hauptkomponentenanalyse zusätzlich verschiedene Techniken nicht-linearer neuronaler Netze [5]. Um die menschliche Wahrnehmung in den Prozess mit einzubeziehen, wird eine Klassifizierung der Multimedia-Daten durch Probanden vorgenommen. Der hybride Ansatz erzielte Erfolge vor allem durch die Reduzierung der Dimensionen und konnte somit eine optimierte Zugriffsmethode für schnelle und präzise Indexierung bereitstellen.

Die vorliegende Arbeit beschäftigt sich im Gegensatz zu den verwandten Arbeiten ausschließlich mit dem Vergleich und der Evaluation verschiedener Distanzfunktionen. Dabei werden aufgrund der Featuredaten des spezifischen Anwendungsfalls jedoch nur Punkt- und Histogramm-Datentypen betrachtet. Gerade im Hinblick auf zukünftige Arbeiten sollte verstärkt darauf Acht gegeben werden, dass das menschliche Wahrnehmungsvermögen nicht auf metrischen Grundsätzen beruht [6] und daher die Annahme eines metrischen Ähnlichkeitsmodels [7] nicht gegeben ist.

IV. Distanzfunktionen

Das folgende Kapitel soll eine Zusammenfassung über potenziell geeignete Distanzfunktionen für das Digi-Dak-Framework bereitstellen. Dabei werden die einzelnen Distanzfunktionen mathematisch durch ihre Berechnungsvorschrift definiert und, wenn bekannt, der entsprechenden Distanzklasse zugeordnet. Sofern möglich werden außerdem die Herkunft sowie bekannte Anwendungsbeispiele aufgezeigt. Der Fokus liegt dabei auf Abstandsfunktionen, welche auf Punkt- oder Histogramm-Daten arbeiten. Andere Funktionen, welche beispielsweise auf Binärdaten, Sequenzen oder Mengen arbeiten, werden nicht als geeignet erachtet, da sie höchstwahrscheinlich nicht mit den zugrundeliegenden Datentypen übereinstimmen[1]. Nichtsdestotrotz kann, wie bereits erwähnt, eine Konvertierung zwischen den verschiedenen Feature-Datentypen vorgenommen werden.

Während der folgenden Ausführungen seien stets die beiden Feature-Vektoren X und Y der Länge n definiert.

A. Minkowski

Die *Minkowski*-Funktionen sind eine Familie von Distanzfunktionen, welche auf Punkt-Daten definiert ist. Die allgemeine Form ist gegeben durch:

$$d_{L_m}(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^m \right)^{\frac{1}{m}} \quad (1)$$

[1]Das Framework arbeitet bislang auf synthetischen Feature-Daten.

Die einzelnen Vertreter der Minkowski-Familie unterscheiden sich lediglich durch die Wahl der Norm m, woraus im Wesentlichen vier Gruppen resultieren:

- Fraktionale Minkowski-Funktionen: $m < 1$
- Manhattan-Distanzfunktion: $m = 1$
- Euklidische Distanzfunktion: $m = 2$
- Max/Chebychev-Distanzfunktion (L_{max}): $m \to \infty$

Aus der mathematischen Form ergibt sich für die Manhattan- und Euklid-Distanzfunktion eine geringe bis mittlere, für die fraktionalen und Max-Distanzfunktionen eine entsprechend hohe Berechnungskomplexität.

Heutige Datenbankanwendungen werden von der euklidische Distanzfunktion dominiert [3]. Dies ist vor allem darauf zurückzuführen, dass sie in der euklidischen Geometrie die kürzeste Strecke zwischen zwei gegebenen Punkten definiert. Eine Vielzahl mathematischer Verfahren basieren auf dieser Tatsache und werden bereits aktiv im Bereich Multimedia-Retrieval eingesetzt [1], [2]. Auch die Manhattan-Distanz kommt aufgrund ihrer sehr einfachen Berechnung und demzufolge kleinen Berechnungszeit, häufiger zum Einsatz [3]. Prinzipiell ist ein geeigneter Normwert empirisch zu bestimmen. Vergangene Arbeiten [8] empfehlen einen Startwert von $m = 0.3$, mit welchem bereits gute Ergebnisse erzielt werden können.

In der Abbildung 2 sind die Einheitskreise verschiedene Vertreter der Minkowski-Familie dargestellt.

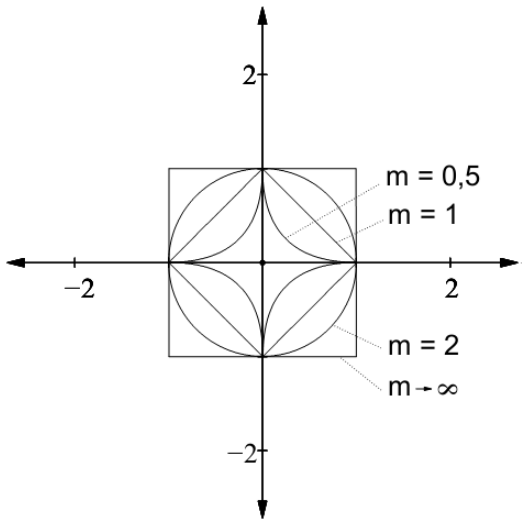


Abbildung 2. Funktionen aus der Minkowski-Familie [1]

B. Gewichtete Minkowski

Ein großer Nachteil der Minkowski-Distanzfunktionen besteht in der Tatsache, dass keine Gewichtung der einzelnen Komponenten eines Feature-Vektors vorgenommen werden kann. Alle Dimensionen werden gleichwertig in die Berechnung einbezogen. Die Gewichtete Minkowski-Abstandsfunktion erweitert die Minkowski-Funktionen durch einen Gewichtungsvektor W:

$$d_{L_m^w}(X,Y) = \left(\sum_{i=1}^{n} w_i |x_i - y_i|^m \right)^{\frac{1}{m}} \quad (2)$$

Es kommt dabei insbesondere zu einer Stauchung oder Streckung einzelner Dimensionen. Dabei sollte lediglich eine Priorisierung einzelner Dimensionen gegenüber Anderen, und nicht etwa eine Skalierung der gesamten Distanz erfolgen. Um dieses Verhalten sicherzustellen, müssen die Gewichte in der Summe dem Wert 1 entsprechen:

$$|W| = \sum_{i=1}^{n} w_i = 1 \quad (3)$$

Sie gehört somit zur Klasse der Pseudo-Distanzfunktionen. In Abbildung 3 sind abermals Vertreter der Minkowski-Familie dargestellt, jedoch beträgt dieses mal das Gewicht der X-Dimension das Doppelte des Gewichts der Y-Dimension. Da die Möglichkeit besteht, eine Dimension mit dem Wert 0 zu gewichten, verletzt die Gewichtete Minkowski-Funktion die Positivitäts-Eigenschaft.

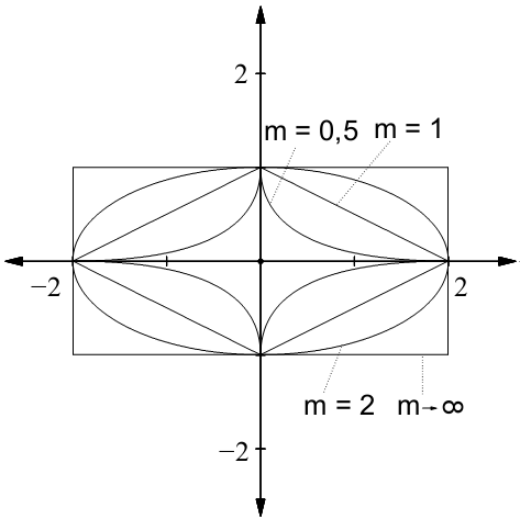


Abbildung 3. Doppelte Gewichtung einer Dimensionen [1]

Bezüglich der Berechnungskomplexität ist die Gewichtete Minkowski-Distanzfunktion mit der Minkowski-Distanzfunktion vergleichbar, da sie lediglich die Multiplikation mit den entsprechenden Gewichten erweitert wird.

C. Quadratic-Form

Auch der *Quadratic-Form*-Distanzfunktion liegt zugrunde, dass für die Ähnlichkeitswahrnehmung zweier Objekte gewisse Eigenschaften von höherer Priorität sind als Andere. Sie gehört ebenso wie die Gewichtete Minkowski-Funktion zur Klasse der Pseudo-Distanzfunktionen und ist formell definiert durch:

$$d_Q(X,Y) = (X-Y)^T \cdot \mathbf{A} \cdot (X-Y) \quad (4)$$

Durch entsprechende Parametrisierung der $n \times n$ Matrix kann sowohl eine Skalierung als auch eine Rotation des Einheitskreises erzeugt werden. Dabei existieren zwei Spezialfälle: Wählt man A als Einheitsmatrix, so entspricht die Distanzfunktion der quadrierten euklidischen Distanzfunktion. Handelt es sich hingegen um eine Diagonalmatrix, so entspricht sie der quadrierten, gewichteten euklidischen Distanzfunktion.

Als annehmbarer Startwert für die einzelnen Matrix-Komponenten hat sich $a_{ij} = 1 - \frac{d_{i,j}}{max[d_{i,j}]}$ mit $d_{i,j} = |x_i - y_i|$ erwiesen [3]. Weitere Optimierungen müssen auch hier empirisch erfolgen. In Abbildung 4 wird beispielsweise eine Rotation um 40 Grad sowie eine Skalierung der X-Dimension um $\sqrt{0,25} = 0,5$ erzeugt [1].

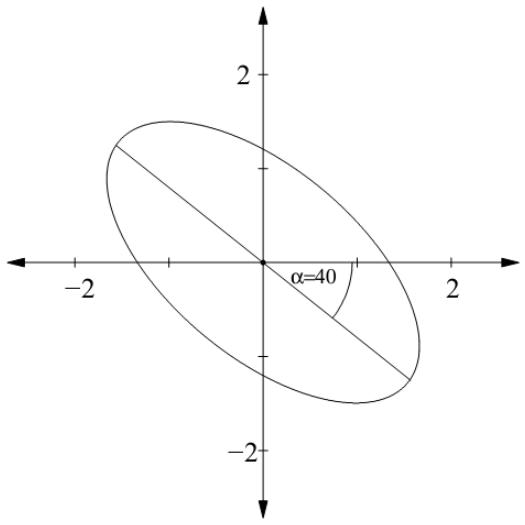


Abbildung 4. Rotation und Skalierung des Einheitskreises [1]

Auch die Quadratic-Form-Distanzfunktion kann durch Wahl der Matrix A in ihrer Berechnungskomplexität variieren.

D. Dynamical-Partial

Sowohl die Minkowski- als auch die Quadratic Form-Abstandsfunktion sind in ihrer Berechnungsstrategie als statisch einzuordnen. Dies führt allerdings gerade in Multimedia-Retrieval Anwendungen zu Problemen [9]: Zum einen ist die Ähnlichkeit zweier Objekte meist nur durch einige wenige Dimension begründet. Sind sich also zwei Objekte hinsichtlich einiger weniger Dimensionen sehr ähnlich, in vielen Anderen allerdings sehr unterschiedlich, so wird insgesamt keine Ähnlichkeit festgestellt. Zum Anderen ist die Ähnlichkeit zwischen verschiedenen Objektpaaren nicht immer auf die gleichen Dimensionen zurückzuführen. Beispielsweise können sich ein Objekt A und B in den ersten 10 Dimensionen ähneln, während Objekt C und D in den nächsten 20 Dimensionen geringe Abstände aufweisen. Mit Hilfe der *Dynamical-Partial*-Funktion können diese Probleme behoben werden (siehe Abbildung 5).

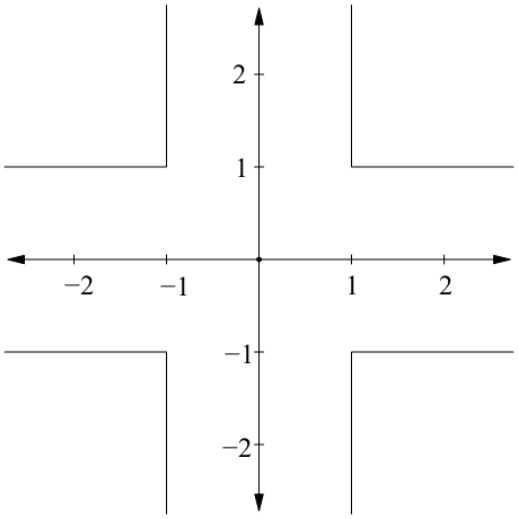


Abbildung 5. Unabhängigkeit von größer werdenden Abständen [1]

Die Dynamical-Partial-Abstandsfunktion gehört zur Klasse der Semi-Pseudo-Distanzfunktionen [1]. Ihr dynamische Charakter liegt darin begründet, dass nur die m kleinsten Abstandswerte zur tatsächlichen Distanzberechnung berücksichtigt werden [9]. Zunächst wird der Abstand der einzelnen Komponenten zweier Feature-Vektoren berechnet:

$$\delta_i = |x_i - y_i| \quad (5)$$

Anschließend wird die Menge Δ_m mit den m kleinsten Abstandswerte gebildet:

$$\Delta_m = \{\text{die kleinsten m } \delta\text{-Werte aus } (\delta_1, \delta_2, ..., \delta_n)\} \quad (6)$$

Darauf aufbauend kann schließlich die dynamisch-partielle Distanz berechnet werden, wobei die Norm r wiederum empirisch für den individuellen Anwendungsfall zu bestimmen ist:

$$d_{DP_{m,r}} = \left(\sum_{\delta_i \in \Delta_m} \delta_i^r \right)^{\frac{1}{r}} \quad (7)$$

Im Wesentlichen erweitert die Dynamical-Partial-Distanzfunktion die Minkowski-Distanzfunktion durch eine Sortierung der Abstandswerte sowie einer abschließenden Aufsummierung. Bei Nutzung intelligenter Datenstrukturen ist die Berechnungskomplexität analog zur Minkowski-Abstandsfunktion einzuschätzen.

E. Bray-Curtis

Die *Bray-Curtis*-Abstandsfunktion gehört zur Klasse der Semi-Pseudo-Distanzfunktionen (siehe Anhang A) und berechnet den Abstand auf Punkt-Daten. Sie erweitert die Manhattan-Funktion durch eine Normalisierung [2]:

$$d_{BC}(X,Y) = \sum_{i=1}^{d} \frac{|x_i - y_i|}{x_i + y_i} \tag{8}$$

Damit ist die Funktion sehr sensitiv gegenüber kleinen Änderungen, während größere Unähnlichkeiten immer mehr an Einfluss verlieren. Der Wertebereich liegt dabei im Intervall $[0, 1]$.

Für die Bray-Curtis-Funktion dürfen keine negativen Feature-Werte verwendet werden, da es andernfalls zu verfälschenden Distanzwerten kommen kann. So ergeben beispielsweise die Werte 10 und -11 einen Distanzwert von -21, und überschreiten somit das genannte Intervall. Die Berechnungskomplexität ist aufgrund der mathematischen Form analog zur im Anschluss folgenden Canberra-Distanzfunktion als gering einzuschätzen.

F. Canberra

Auch die *Canberra*-Distanzfunktion normalisiert die Abstände der einzelnen Dimensionen ähnlich der Bray-Curtis-Funktion. Zur Berechnung des Normalisierungswertes werden jedoch die absoluten Dimensionswerte addiert, wodurch auch die Berechnung negativer Feature-Werte zuverlässig erfolgen kann. Sie gehört zur Klasse der Distanzfunktionen (siehe Anhang A) und ist formell definiert durch [2], [3]:

$$d_C(X,Y) = \sum_{i=1}^{n} \frac{|x_i - y_i|}{|x_i| + |y_i|} \tag{9}$$

Ein großer Nachteil, welcher sowohl bei der Canberra- als auch der Bray-Curtis-Funktion vorliegt, besteht darin, dass die Normalisierung nur durch die jeweils aktuellen Dimensionswerte erfolgt. So ergeben beispielsweise die Werte 1000 und 1050 eine Distanz von lediglich $0,024$; die Werte 0 und 50, welche die gleiche Wertdistanz aufweisen, hingegen eine maximale Distanz von 1. Hier wäre eventuell eine andere Art der Normalisierung vorteilhaft, welche den gesamten Wertebereich der jeweiligen Dimension in Betracht zieht. Die Berechnungskomplexität wurde als gering eingeschätzt [3].

Die Canberra-Distanzfunktion wurde bereits erfolgreich für die Analyse von DNA Sequenzen in der Bioinformatik und in der Erkennung von Computer-Intrusion verwendet [10].

G. Chi-Quadrat

Die *Chi-Quadrat*-Distanzfunktion wird auf Histogramm-basierte Features angewendet und wird der Klasse der Semi-Pseudo-Distanzfunktionen zugeordnet [1]. Dabei wird der Abstand zwischen verschiedenen Histogrammen, mit der absoluten Wahrscheinlichkeit bestimmter Ereignisse, berechnet. Formell ist die Chi-Quadrat-Distanzfunktion wie folgt definiert, wobei *X* und *Y* Häufigkeitsverteilungen und m_1 die tatsächliche Häufigkeit repräsentiert [11]:

$$d_{Chi^2}(X,Y) = \sum_{i=1}^{n} \frac{(x_i - m_i)^2}{m_i} \tag{10}$$

mit

$$m_i = \frac{(x_i + y_i)}{2} \tag{11}$$

Zur Verwendung der Chi-Quadrat-Distanzfunktion müssen die Histogramme in einheitlich definierte Intervalle unterteilt sein, damit für die weitere Berechnung der Punktdatentyp verwendet werden kann. Die Punkte bilden dabei eine Häufigkeitsverteilung, mit Hilfe derer die Differenzen zwischen erwarteten und tatsächlichen Häufigkeiten bestimmt werden können. Die Differenzen werden im weiteren Verlauf der Berechnung quadriert, normiert und aufsummiert und bilden abschließend ein Maß für den Abstand der Histogramme. Für den Fall, dass die erwartete und die tatsächliche Häufigkeit gleich sind, beträgt der Abstand zwischen den Histogrammen 0 [1]. Die Einheitskreisdarstellung der Chi-Quadrat-Distanzfunktion ist in Abbildung 6 dargestellt. Dabei ist erkennbar, dass der Einheitskreis nicht geschlossen ist, was sich durch die fehlende Positivität begründen lässt [1].

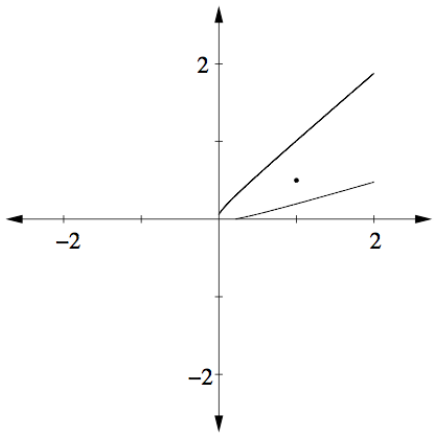


Abbildung 6. Einheitskreis der Chi-Quadrat-Funktion [1]

Anwendung findet diese Distanzfunktion vor allem in der Statistik, aber auch im Bereich des *Texture Retrieval* [3].

H. Kullback-Leibler

Die *Kullback-Leibler*-Abstandsfunktion ist laut Definition keine Distanzfunktion, da Selbstidentität und Dreiecksungleichung nicht erfüllt werden [1]. Die Abstandsfunktion führt einen Entropievergleich von zwei

Häufigkeitsverteilungen durch, wobei, ähnlich zu der Chi-Quadrat-Distanzfunktion, Histogramme mit einheitlich definierten Intervallen notwendig sind, um den Punktdatentyp anwenden zu können. Die Punkte müssen dabei folgende Eigenschaften erfüllen [12]:

$$i = 1 \ldots n : 0 \leq p[i] \leq 1 \tag{12}$$

und

$$\sum_{i=1}^{n} p[i] = 1 \tag{13}$$

In Abbildung 7(a) ist der Einheitskreis der Kullback-Leibler-Abstandsfunktion dargestellt. Sie repräsentiert eine zweidimensionale Projektion eines dreidimensionalen Einheitskreises, die sich auf vorheriger Bedingung gründet, da in diesem Fall bei n Dimensionen nur $n-1$ Dimensionen voneinander unabhängig sind [1].

Die Entropie $H_p(X)$ einer Zufallsvariablen X repräsentiert das Maß der Ungenauigkeit einer Zufallsvariablen. Sie ist über die Wahrscheinlichkeiten px für jedes Ereignis x wie folgt definiert [1], [12]:

$$H_p(X) = \sum_{x \in X} p(x) \cdot \log_2 \frac{1}{p(x)} \tag{14}$$

Wird dabei angenommen, dass

$$p(x) = 0; p(x) \cdot \log_2 \frac{1}{p(x)} = 0 \tag{15}$$

gilt, haben unmögliche Ereignisse keinen Einfluss auf die Entropie der Zufallsvariablen. Der Kullback-Leibler-Abstand ist, entsprechend der Definition, über zwei Verteilungen p und q der Ereignismenge X definiert: [1]

$$\begin{aligned} D(p\|q) &= \sum_{x \in X} p(x) \cdot \log_2 \frac{p(x)}{q(x)} \\ &= \sum_{x \in X} p(x) \cdot \log_2 \frac{1}{q(x)} - H_p(X) \end{aligned} \tag{16}$$

Die Abstandsfunktion gibt in Bezug auf $p(x)$ und $q(x)$ an, wieviele Bits zusätzlich zur Kodierung benötigt werden. Durch die Verwendung der Logarithmus-Funktion verfügt die Kullback-Leibler-Abstandsfunktion über eine höhere Berechnungskomplexität, als beispielsweise die Chi-Quadrat-Distanzfunktion.

I. Jeffrey Divergence

Die *Jeffrey Divergence* basiert auf der Kullback-Leibler-Abstandsfunktion. Im Gegensatz zur Kullback-Leibler-Abstandsfunktion verfügt die Funktion jedoch über ein besseres numerisches Verhalten und ist symmetrisch. Dadurch ist die Jeffrey Divergence stabiler und robuster in Bezug auf Rauschen und Größe der Histogramm-Bins [13]. Sie ist formell definiert durch [3]:

$$d_J(X,Y) = \sum_{i=1}^{n} \left(x_i \log \frac{x_i}{m_i} + y_i \log \frac{y_i}{m_i} \right) \tag{17}$$

$$m_i = \frac{x_i + y_i}{2} \tag{18}$$

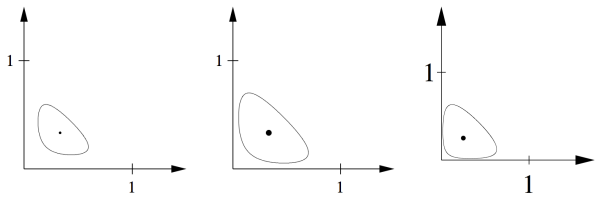
Die Einheitskreisdarstellung der Jeffrey Divergence-Distanzfunktion in Abbildung 7(b) ist, aufrund der Ableitung, ähnlich der Kullback-Leibler-Abstandsfunktion. Dabei kann das bessere numerische Verhalten der Jeffrey Divergence, gegenüber der Kullback-Leibler-Abstandsfunktion, auch am Einheitskreis beobachtet werden, da sie sich besser an die Koordinatenachsen annähert. Durch die Verwendung der Logarithmus-Funktion verfügt jedoch auch die Jeffrey Divergence über eine vergleichsweise höhere Berechnungskomplexität.

J. Bhattacharyya

Die *Bhattacharyya*-Distanzfunktion gehört zur Klasse der Semi-Distanzfunktionen, da alle Eigenschaften außer die Dreiecksungleichung erfüllt werden [1]. Sie findet vor allem Einsatz in der Muster- und Spracherkennung und is als 2D-Kreis im 3D-Raum dargestellt. Formell ist die Abstandsfunktion wie folgt definiert, wobei p und q Verteilungen der Ereignismenge X repräsentieren [14], [15]:

$$d_{Bhat}(p,q) = -\log \sum_{x \in X} \sqrt{p(x) \cdot q(x)} \tag{19}$$

Die Einheitskreisdarstellung, ähnlich der von Kullback-Leibler und Jeffrey Divergence, repräsentiert einen dreidimensionalen Einheitskreis, der auf zwei Dimensionen abgebildet ist. Diese Darstellung begründet sich in der Forderung, dass die Summe der Wahrscheinlichkeiten 1 ergeben muss und damit bei drei Ereignissen die Wahrscheinlichkeiten nur von zwei dieser Ereignisse unabhängig sind [1]. Sie ist in Abbildung 7(c) graphisch dargestellt. Im Gegensatz zur Kullback-Leibler-Abstandsfunktion nähert sich die Bhattacharyya-Distanzfunktion näher an die Koordinatenachsen an, wodurch sie ein vergleichsweise besseres Verhalten aufweist. Jedoch ist auch hier, aufgrund der Verwendung von Logarithmus- und Wurzeloperationen, die Berechnungskomplexität wesentlich höher.



(a) Kullback-Leibler (b) Jeffrey-Divergence (c) Bhattacharyya

Abbildung 7. Distanzfunktionen basierend auf Histogrammen [1]

V. Konzept

Unsere Ziele für das DigiDak-Framework bestehen in der Umsetzung genannter Funktionen, der Erweiterung der grafischen Benutzungsoberfläche, um Funktionen auszuwählen und zu parametrisieren sowie der Bereitstellung eines intuitiven Programmierschnittstelle, die vor allem eine einfache Implementierung neuer Funktionen unterstützt.

Die Basis unserer Umsetzung besteht in der *DistanceMetric*-Klasse. Sie stellt über die statischen Funktionen *getInstance()* und *setInstance()* ein *Singleton-Pattern* bereit, mit dem die aktive Distanzfunktion geholt und gesetzt werden kann [16]. Mit Hilfe mit der beiden Instanzfunktionen *calculate(int[] x, int[] y)* und *calculate(double[] x, double[] y)* kann die Distanz zwischen zwei Feature-Vektoren von einer beliebigen Stelle im Framework berechnet werden[2]. Diese beiden Funktionen arbeiten intern wiederum ihren entsprechenden abstrakten *calculate*-Funktionen, welche in ihrer Signatur zusätzlich die untere und obere Schranke (*lower* und *upper*) für die Berechnung definieren.

Eine Erweiterung des Frameworks um weitere Funktionen gestaltet sich sehr einfach. Die entsprechende Funktion muss lediglich von der Klasse *DistanceMetric* erben und die abstrakten *calculate*-Funktionen implementieren. Für alle parametrisierbaren Funktionen (zum Beispiel: *Minkowski* und *Dynamical-Partial*) muss ein spezieller Konstruktor bereitgestellt werden, über den die jeweiligen Parameter zur Instanzierung übergeben werden können (siehe Abbildung 8). Durch die Nutzung von *Reflection*-Methoden kann auf diese Weise die Erstellung verschiedener Funktion-Instanzen dynamisch zur Laufzeit erfolgen.

```
public class MinkowskiMetric
      extends DistanceMetric
{
    public MinkowskiMetric(float norm)
    {
        ...
    }
}
```

Abbildung 8. Parametrisierbare Distanzfunktion

In der grafischen Benutzungsoberfläche werden implementierte Funktionen durch eine ComboBox bereitgestellt (siehe Abbildung 9). Die Kollektion verfügbarer Funktionen wird zur Laufzeit durch dynamisches Laden von *.class*-Dateien realisiert. Durch Selektion einer Distanzfunktion in der ComboBox, wird diese automatisch zur Berechnung innerhalb der verschiedenen Index-Strukturen verwendet.

[2]Die Entscheidung zur Bereitstellung zweier Berechnungsfunktionen liegt in dem schnelleren Laufzeitverhalten von *int* gegenüber *double* begründet.

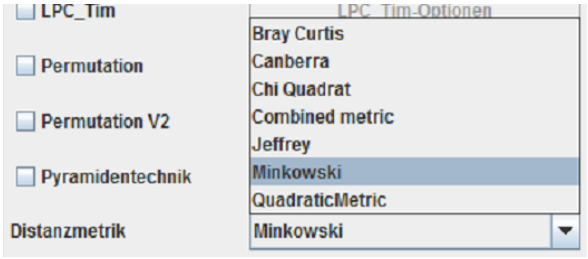


Abbildung 9. Auswahl der zu verwendenden Funktion

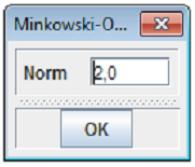


Abbildung 10. Eingabe der Parameter im Optionsdialog

Parametrisierbare Funktionen definieren zusätzlich einen eigenen Optionsdialog, über den die entsprechende Eingabe erfolgen kann (siehe Abbildung 10).

Gemäß der Beobachtung hinsichtlich der Nutzung von kombinierten Funktionen, sieht unsere Erweiterung unter anderem auch eine Klasse *CombinedMetric* vor. Eine Instanz der Klasse *CombinedMetric* setzt sich aus verschiedenen *RangedMetric*-Objekten zusammen. Jedes *RangedMetric*-Objekt wiederum stellt ihre untere und obere Schranke sowie die entsprechende Funktion durch Felder bereit. Beim Aufruf der Berechnungsfunktion eines *CombinedMetric*-Objekts werden alle *RangedMetric*-Objekte entsprechend ihrer Parameter aufgerufen. Im letzten Schritt werden die jeweils berechneten Teildistanzen aufsummiert und als Gesamtdistanz zurückgegeben.

VI. Evaluierung

Zur Evaluierung der vorgestellten Distanzfunktionen wurde ein Datensatz mit 1000 Datenpunkten und 1000 Dimensionen generiert. Der Wertebereich jeder Dimension war dabei auf den ganzzahligen, positiven Bereich zwischen 0 und 255 begrenzt. Des Weiteren wurden exemplarisch

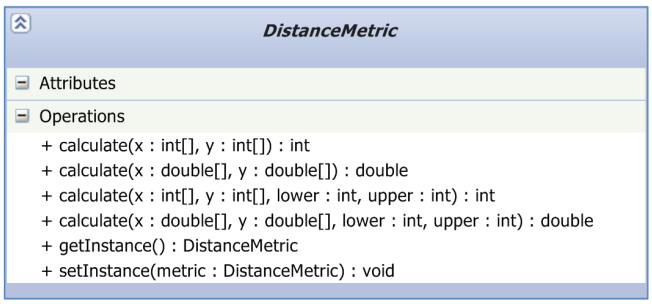


Abbildung 11. Abstrakte Basisklasse

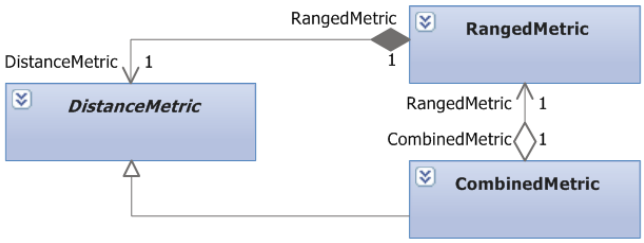


Abbildung 12. Aufbau einer Multi-Funktion

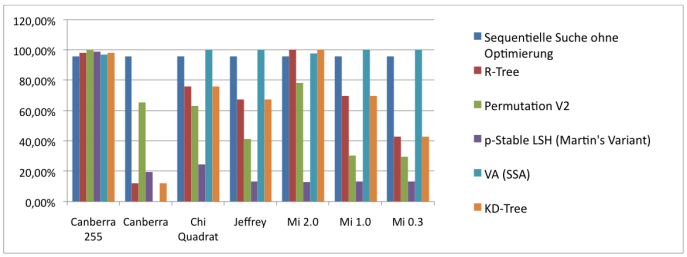


Abbildung 14. Präzision der Distanzfunktionen für k=10

verschiedene Indexstrukturen ausgewählt, mit denen die Distanzfunktionen getestet wurden. Zu Beginn wurden für die Tests die sequentielle Suche, die Permutation und der R-Baum verwendet, um die Performance für approximative und Baumverfahren zu betrachten. Im weiteren Testverlauf wurden weitere Indexstrukturen zur Evaluierung hinzugezogen, um Testergebnisse besser einordnen und interpretieren zu können. So wurden die Funktionen zusätzlich mit Hilfe von LSH (Locality Sensitive Hashing), VA (Vector Approximation)-File und KD-Baum evaluiert. Auf die Eigenschaften der Indexstrukturen wird in dieser Arbeit jedoch nicht genauer eingegangen.

Da in verwandten Arbeiten [2], [3] die Canberra-Funktion scheinbar am besten abschnitt, wurde auch diese mit in unsere Evaluierung einbezogen, um einen Vergleich zu ermöglichen. Des Weiteren wurden die Minkowski-Funktionen aufgrund ihrer häufigen Erwähnung in der Literatur [3], [13], die Jeffrey Divergence und die Chi-Quadrat-Distanzfunktion getestet. Dabei wurde im Fall der Minkowsi-Funktionen insbesondere auch die Gruppe der fraktionalen Funktionen betrachtet. Aufgrund der Erwähnung in der Literatur [8] wurde hierbei $m = 0.3$ als Startwert gewählt, da hiermit die besten Ergebnisse erzielt werden sollten.

Die Evaluation erfolgte nach Präzision, Laufzeit und Anzahl der Aufrufe der Distanzfunktion. Jede Kombination aus Distanzfunktion und Indexstruktur durchlief dabei je 25 Berechnungszyklen mit $k = 1$ und $k = 10$ für die *k-Nearest-Neighbor*-Suche. Die Testergebnisse hinsichtlich der Präzision der Distanzfunktionen sind in den Abbildungen 13 und 14 grafisch dargestellt.

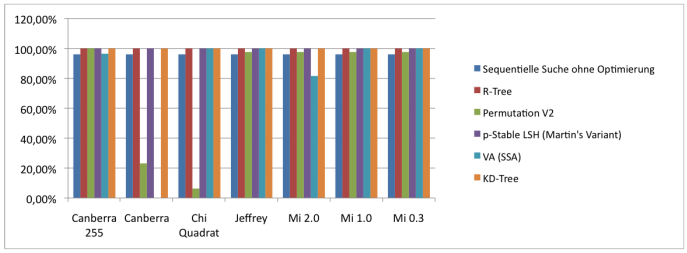


Abbildung 13. Präzision der Distanzfunktionen für k=1

Dabei wurde deutlich, dass sowohl die angewendeten Baumverfahren, als auch LSH keinen Einfluss auf die Distanzfunktionen hatten, da immer eine Präzision von 1 ausgegeben wurde. Die ersten Unterschiede zwischen den Distanzfunktionen wurden erst bei Verwendung der Permutation und der VA-File als Indexstruktur ersichtlich. Des Weiteren konnte bei den Testläufen bestätigt werden, dass die Minkowski-Funktionen, wie in der Literatur mehrfach beschrieben, eine hohe Präzision aufweisen und aufgrund dessen häufig Verwendung finden. Während der Evaluierung wurde eine Präzision von mehr als 97% für diese Funktionen erreicht. Im Vergleich mit den anderen Distanzfunktionen konnte nur die Jeffrey-Divergence ähnlich gute Ergebnisse aufweisen. In den Abbildungen 15 und 16 sind die Laufzeiten der Distanzfunktionen grafisch dargestellt.

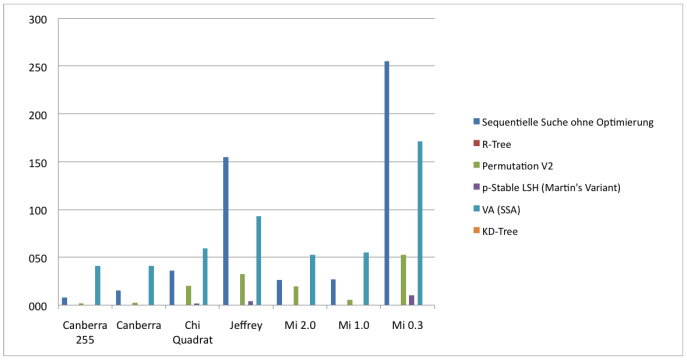


Abbildung 15. Laufzeit der Distanzfunktionen für k=1

Dabei wird deutlich, dass die Manhattan-Distanz die stabilsten Ergebnisse, bei zusätzlich geringer Laufzeit, liefert. Die Jeffrey-Divergence, die bei der Präzision auch sehr gute Ergebnisse lieferte, verfügt über eine vielfach höhere Laufzeit, da die Berechnungskomplexität im Vergleich um ein Vielfaches höher ist. Nur die fraktionale Minkowski-Funktion schneidet in diesem Fall noch schlechter ab. Bezüglich der Anzahl der Aufrufe der Distanzfunktionen wurden die folgenden, in den Abbildungen17 und 18 graphisch dargestellten, Ergebnisse erzielt.

Auffällig ist hierbei, dass die Anzahl der Distanzfunktionsaufrufe, bei Verwendung der Permutation als Indextruktur, für alle Distanzfunktionen am größten ist. Grund

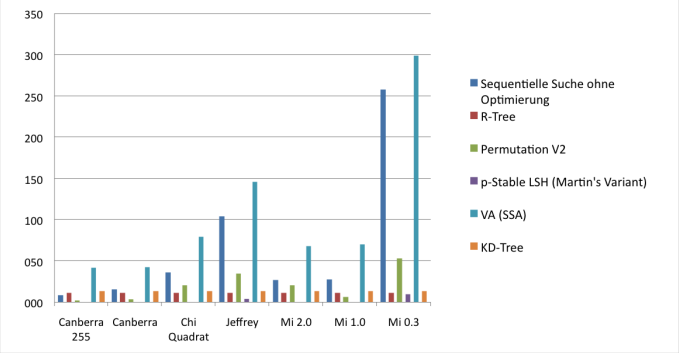


Abbildung 16. Laufzeit der Distanzfunktionen für k=10

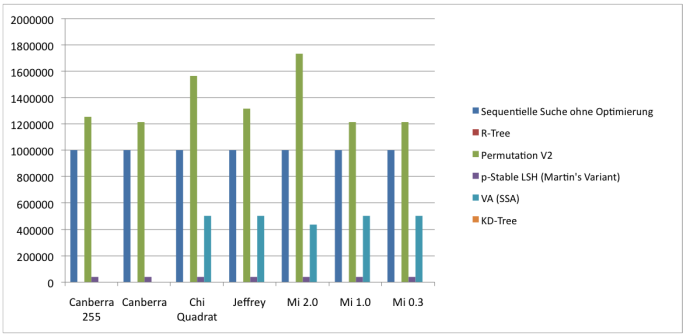


Abbildung 17. Anzahl der Aufrufe der Distanzfunktionen für k=1

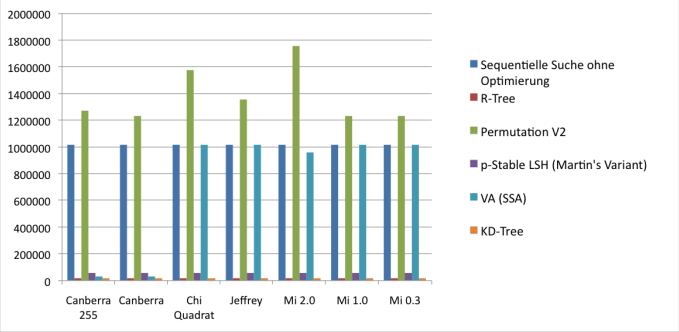


Abbildung 18. Anzahl der Aufrufe der Distanzfunktionen für k=10

dafür ist die höhere Anzahl an Vergleichen, die bei der Permutation notwendig sind. Im Vergleich dazu benötigte die sequentielle Suche für jede Funktion eine konstante Anzahl an Aufrufen, die aber vergleichsweise geringer ist, als bei der Permutation. Vor allem die verwendeten Baumverfahren benötigten hierbei nur eine sehr geringe Anzahl an Aufrufen, wodurch sich auch die geringe Laufzeit erklären lässt. Unterschiede konnten jedoch nicht nur bei alleiniger Betrachtung der drei Evaluationsparameter Präzision, Laufzeit und Distanzfunktionsaufrufe festgestellt werden. So konnten bei Betrachtung der gewählten Parameter $k = 1$ und $k = 10$ für die *k-Nearest-Neighbor*-Suche signifikante Unterschiede beobachtet werden. Für $k = 10$ konnte beispielweise festgestellt werden, dass mit keiner der gewählten Indexstrukturen eine konstant hohe Präzision erreicht wurde. Unter allen getesteten Distanzfunktionen lieferte die euklidische Distanz hierbei die vergleichsweise besten Ergebnisse.

Im Gegensatz dazu konnten für die *k-Nearest-Neighbor*-Suche mit $k = 1$ bessere Ergebnisse erzielt werden. Lediglich bei der Canberra-, Bray-Curtis- und Chi-Quadrat-Distanzfunktion konnten größere Unterschiede bei der Nutzung verschiedener Indexstrukturen beobachtet werden. So lieferte die Nutzung von Permutation und VA-File eine wesentlich geringere Präzision im Vergleich zu den anderen verwendeten Indexstrukturen, die relativ konstante Präzisionen aufweisen konnten.

Zusammenfassend für alle Testläufe kann konstatiert werde, dass die euklidische Distanz in Verbindung mit den verwendeten Baumverfahren präzise Ergebnisse, bei geringer Laufzeit, liefert. Des Weiteren zeigte die VA-File ein sehr stabiles Verhalten bei verschiedenen Distanzfunktionen und wechselnden Parametern für die k-nächste-Nachbarn-Suche. Als Ausnahmen sind hierbei nur die Bray-Curtis- und die Canberra-Funktion zu nennen, die Vergleich zu verwandten Arbeiten wesentlich schlechter abschnitten.

In einem sehr späten Abschnitt unseres Projekts, haben wir aufgrund der Beobachtung in Kapitel V eine Anpassung der Normalisierung der Canberra-Distanzfunktion vorgenommen. Diese sieht eine Ersetzung des Nenners durch die Kardinalität der Dimension vor:

$$d_{C_{255}}(X, Y) = \sum_{i=1}^{n} \frac{|x_i - y_i|}{255} \tag{20}$$

Dabei konnten wir beobachten, dass sich die Funktion vor allem in Hinblick auf die Präzision sehr stark verbessert. Die Ergebnisse für die angepasste Canberra-Funktion kann den jeweiligen Diagrammen entnommen werden.

VII. Zusammenfassung

In dieser Arbeit haben wir insgesamt zehn Distanzfunktionen formell definiert, informell beschrieben, nach ihren Eigenschaften klassifiziert und, sofern möglich, hinsichtlich Präzision und Laufzeitverhalten evaluiert. Die Tabelle aus Abbildung 19 fasst die wichtigsten Merkmale zusammen.

Distanzfunktion	Klasse	Komplexität	Impl./Eval.
Minkowski	D	gering bis hoch	● / ●
Gewichtete Minkowski	PD	gering bis hoch	● / ○
Quadratic-Form	PD	gering bis hoch	● / ○
Dynamical-Partial	SPD	gering bis hoch	● / ○
Bray-Curtis	SPD	gering	● / ○
Canberra	D	gering	● / ●
Chi-Quadrat	SPD	mittel	● / ●
Kullback-Leibler	-	hoch	○ / ○
Jeffrey Divergence	-	hoch	● / ●
Bhattacharyya	SD	hoch	○ / ○

Abbildung 19. Übersicht verwendeter Distanzfunktionen

Von den zehn untersuchten Distanzfunktionen wurden insgesamt acht von uns implementiert. Unser Klassendesign zielt vor allem auf eine einfache Erweiterung durch neue Distanzfunktionen. Diese brauchen lediglich von einer Basisklasse zu erben und ihre Berechnungsvorschrift in einer Funktion umsetzen. Zur Auswahl und Parametrisierung einer Distanzfunktion wurde das Framework durch eine *DropDown-Box* sowie einen individuellen Optionsdialog vervollständigt. Die Funktionalität der Multi-Distanzfunktionen wurde lediglich für die Nutzung durch Programmierer umgesetzt und könnte in zukünftigen Arbeiten eventuell durch eine grafische Benutzungsoberfläche ergänzt werden.

In der Evaluation konnten Abweichungen zu Ergebnissen verwandter Arbeiten beobachtet werden. Beispielsweise schnitt die Canberra-Distanzfunktion aufgrund der zugrundeliegenden Werteverteilung der Featuredaten wesentlich schlechter ab als erwartet. Aus diesem Grund wurde im Laufe unserer Arbeit eine Abwandlung der *Canberra*-Distanzfunktion untersucht, bei welcher durch die Kardinalität der entsprechenden Dimension geteilt wird. Im Gegensatz zur vorherigen Funktion wies diese Abwandlung ein sehr gutes Verhalten mit stabilen Präzisionen, auf. Abschließend konnte festgestellt werden, dass die Wahl des richtigen k, für die *k-Nearest-Neighbor*-Suche, ebenfalls Einfluss auf die Präzision der Distanzfunktionen hatte.

ANHANG A

Im Folgenden soll die Bray-Curtis- und Canberra-Distanzfunktion hinsichtlich der Eigenschaften Selbstidentität, Positivität, Symmetrie und Erfüllung der Dreiecksungleichung untersucht werden. Für die näheren Betrachtungen ist es dabei ausreichend, von eindimensionalen Vektoren auszugehen und die Eigenschaften dementsprechend im eindimensionalen Raum nachzuweisen.

A. Selbstidentität

$d_{BC/C}(x, x) = 0$ gilt für alle Werte $x \in \mathbb{R} \setminus \{0\}$, da im Zähler $x - x = 0$ resultiert. Wenn $x = 0$ liegt im Nenner eine nicht definierte Nulldivision vor. Die Selbstidentität ist somit für beide Distanzfunktionen erfüllt.

B. Positivität

Wenn $y > x$ ist der Nenner der Bray-Curtis-Distanzfunktion negativ und es folgt $d_{BC}(x, y) < 0$. Die Canberra-Distanzfunktion bildet, diesem Fall gegenwirkend, einen Betrag im Nenner, wodurch stets Positivität gegeben ist $\forall x, y \in \mathbb{R}, x \neq y : d_C(x, y) > 0$.

C. Symmetrie

Die Symmetrie folgt für beide Distanzfunktionen im Zähler aus der Betragsbildung sowie im Nenner durch die Kommutativität der Addition.

D. Dreiecksungleichung

Um auf die Erfüllung der Dreiecksungleichung zu prüfen, müssen die entsprechenden Ungleichungen der Bray-Curtis- und Canberra-Distanzfunktion bewiesen werden:

$$d_{BC} : \frac{|x - z|}{x + z} \leq \frac{|x - y|}{x + y} + \frac{|y - z|}{y + z} \tag{21}$$

$$d_C : \frac{|x - z|}{|x + z|} \leq \frac{|x - y|}{|x + y|} + \frac{|y - z|}{|y + z|} \tag{22}$$

Die Bray-Curtis-Distanzfunktion ergibt für die Dreiecksungleichung mit den Werten $x = 1, y = -5, z = 3$ eine falsche Aussage $0,5 \leq -5,5$.

Für die Canberra-Distanzfunktion muss eine Fallunterscheidung getroffen werden, wodurch die Betragsbildung der einzelnen Terme entfallen kann. Anschließend muss die Ungleichung durch Umformung derart vereinfacht werden, dass eine einfache Aussage über deren Wahrheit möglich ist. Die Erfüllung der Dreiecksungleichung wurde exemplarisch mit Hilfe eines Algebraprogramms [3] nachgewiesen.

LITERATUR

[1] I. Schmitt, *Ähnlichkeitssuche in Multimedia-Datenbanken*. Oldenbourg Verlag, 2006.

[2] M. Kokare, B. Chatterji, and P. Biswas, "Comparison of similarity metrics for texture image retrieval," in *Conference on convergent technologies for Asia-Pacific region*, ser. TENCON '03, vol. 2. IEEE, 2003, pp. 571–575.

[3] P. H. Bugatti, A. J. M. Traina, and C. Traina, Jr., "Assessing the best integration between distance-function and image-feature to answer similarity queries," in *Proceedings of the 2008 ACM symposium on Applied computing*, ser. SAC '08. New York, NY, USA: ACM, 2008, pp. 1225–1230.

[4] B. Bustos and T. Skopal, "Dynamic similarity search in multi-metric spaces," in *Proceedings of the 8th ACM international workshop on Multimedia information retrieval*, ser. MIR '06. New York, NY, USA: ACM, 2006, pp. 137–146.

[5] A. H. H. Ngu, Q. Z. Sheng, D. Q. Huynh, and R. Lei, "Combining multi-visual features for efficient indexing in a large image database," *The VLDB Journal*, vol. 9, pp. 279–293, April 2001.

[6] D. Jacobs, D. Weinshall, and Y. Gdalyahu, "Classification with nonmetric distances: Image retrieval and class representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 6, pp. 583–600, 2000.

[7] P. W. H. Kwan and J. Gao, "A multi-step strategy for approximate similarity search in image databases," in *Proceedings of the 17th Australasian Database Conference - Volume 49*, ser. ADC '06. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2006, pp. 139–147.

[3] http://reference.wolfram.com/mathematica/ref/CanberraDistance.html

[8] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional spaces," in *Proceedings of the 8th International Conference on Database Theory*, ser. ICDT '01. London, UK: Springer-Verlag, 2001, pp. 420–434.

[9] B. Li, E. Chang, and Y. Wu, "Discovery of a perceptual distance function for measuring image similarity," *Multimedia systems*, vol. 8, no. 6, pp. 512–522, 2003.

[10] S. Emran and N. Ye, "Robustness of canberra metric in computer intrusion detection," in *Workshop on Information Assurance and Security, West Point, NY, USA*. IEEE, 2001.

[11] W. Mendenhall, R. Beaver, and B. Beaver, *Introduction to probability and statistics*. Duxbury Pr, 2006.

[12] T. Cover, J. Thomas, J. Wiley *et al.*, *Elements of information theory*. Wiley Online Library, 1991, vol. 6.

[13] Y. Rubner, "Perceptual metrics for image database navigation," Ph.D. dissertation, Stanford university, May 1999.

[14] D. Kil and F. Shin, *Pattern recognition and prediction with applications to signal characterization*. Amer Inst of Physics, 1996.

[15] K. Fukunaga, *Introduction to statistical pattern recognition*. Academic Pr, 1990.

[16] J. Vlissides, R. Helm, R. Johnson, and E. Gamma, "Design patterns: Elements of reusable object-oriented software," *Reading: Addison-Wesley*, 1995.

Einfluss von multivariat schief normalverteilten Daten auf multidimensionale Indexstrukturen

Sebastian Dorok*, Martin Tobies†, Andre Zaske‡
Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg
D-39106 Magdeburg
*Email: sebastian.dorok@st.ovgu.de
†Email: martin.tobies@st.ovgu.de
‡Email: andre.zaske@st.ovgu.de

***Zusammenfassung*—In den letzten Jahren ist das Volumen an multidimensionalen Daten sowohl in Wirtschaft als auch in Wissenschaft stetig gewachsen. Der Einsatz multidimensionaler Indexstrukturen ist unerlässlich um einen effizienten Zugriff auf die Daten zu gewährleisten. Eine große Herausforderung besteht darin, die optimale Indexstruktur für einen Datenbestand auszuwählen. Allerdings hängt diese Auswahl von mehreren Kriterien ab. Ein Kriterium ist die Datenverteilung, die den Daten zu Grunde liegt. In der vorliegenden Arbeit untersuchen wir, welchen Einfluss unterschiedliche Datenverteilungen auf die exakte und kNN-Suche mit multidimensionalen Indexstrukturen haben. Wir konzentrieren uns bei der Untersuchung auf die Pyramidentechnik, den R*-Baum und die Prototyp-basierende Indexstruktur. Die verschiedenen Datenverteilungen werden mithilfe der multivariat schiefen Normalverteilung generiert, die unterschiedlich parametrisiert wird. Im Evaluationsergebnis beschreiben wir, welche Auswirkungen wir je Datenverteilung identifiziert haben und klären deren Ursache. Unsere Ergebnisse können als Grundlage dienen um eine multidimensionale Indexstruktur für eine bestimmte Datenbestand auszuwählen.**

***Keywords*-Datenverteilung, R*-Baum, Prototyp-basierende Indexstruktur, Pyramidentechnik, Multivariat schiefe Normalverteilung**

I. Einleitung

In vielen Bereichen von Wirtschaft und Wissenschaft steigt das zu verarbeitende Datenvolumen stetig an[1]. Je größer das Datenvolumen und je komplexer die Datenmodelle werden, desto schwieriger gestaltet sich der effiziente Umgang mit den Daten. Die primäre Herausforderung stellt dabei die effiziente Suche in den Daten dar. Um dieser Herausforderung zu begegnen, werden multidimensionale Indexstrukturen genutzt, die es ermöglichen sollen Daten schneller aufzufinden als es mit einer sequentiellen Suche möglich wäre.

Welche Indexstrukturen am Besten für die Suche geeignet sind, ist ein aktuelles Forschungsgebiet [10]. Dabei werden verschiedene multidimensionale Indexstrukturen gegeneinander evaluiert, indem sie auf unterschiedlich verteilte Daten angewandt und die resultierenden Laufzeiten für die Suche in den Daten miteinander verglichen werden.

Im Folgenden jedoch wollen wir multidimensionale Indexstrukturen isoliert betrachten und genauer untersuchen, welchen Einfluss die Datenverteilung auf die exakte und k-Nearest-Neighbour-Suche (kNN-Suche) hat. Daraus wollen wir allgemeine Rückschlüsse ziehen, wie sich die Indexstrukturen unter bestimmten Datenverteilungen verhalten. Außerdem können die Ergebnisse Aufschluss über das Verhalten der Indexstrukturen in multidimensionalen Räumen geben. Dieses Wissen kann bei der Bestimmung einer geeigneten Indexstruktur für eine Datenverteilung herangezogen werden.

Dazu werden in Kapitel II die Datenverteilungen und Indexstrukturen erläutert, die für die weiteren Betrachtungen relevant sind. In Kapitel III beschreiben wir das Evaluationskonzept zur Untersuchung des Einflusses verschiedener Datenverteilungen auf Indexstrukturen. Dabei geben wir an, auf welche Datenverteilungen die Indexstrukturen angewandt und wie diese Datenverteilungen erzeugt werden. Die daraus resultierenden Ergebnisse werden dann in Kapitel IV zusammengefasst und diskutiert. Nach der Diskussion der Ergebnisse gehen wir in Kapitel V auf weiterführende Untersuchungsansätze ein.

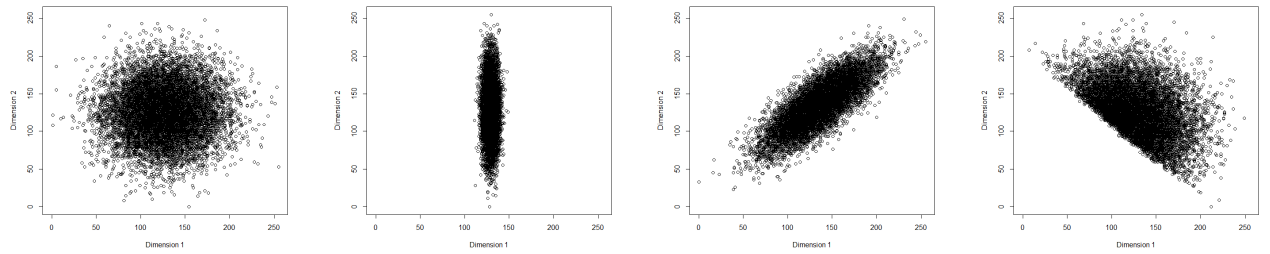
II. Grundlagen

Im Folgenden werden die Datenverteilungen und deren Eigenschaften vorgestellt, die für die Evaluation des Einflusses von Datenverteilungen auf Indexstrukturen verwendet werden. Außerdem wird auf die in der Evaluation untersuchten Indexstrukturen eingegangen und deren Funktionsweise kurz erläutert.

A. Datenverteilungen

Die Vorstellung einer neuen Indexstruktur geht immer mit der Evaluation dieser einher. Beispiele für die Evaluation multidimensionaler Indexstrukturen sind u.a. in [3], [4], [9], [11], [16] zu finden. Dabei werden die Indexstrukturen auf unterschiedlich verteilte Daten angewandt.

[1] http://smartdatacollective.com/gilpress/51849/how-data-became-big, Zugriff: 17.06.2012



(a) $\Omega=\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$; $\alpha=\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ (b) $\Omega=\begin{pmatrix} 1 & 0 \\ 0 & 50 \end{pmatrix}$; $\alpha=\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ (c) $\Omega=\begin{pmatrix} 1 & 0.8 \\ 0.8 & 1 \end{pmatrix}$; $\alpha=\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ (d) $\Omega=\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$; $\alpha=\begin{pmatrix} 10 \\ 10 \end{pmatrix}$

Abbildung 1. Wandelbarkeit der multivariat schiefen Normalverteilung

Jedoch werden selten genaue Angaben gemacht, wie die Datenverteilung erzeugt wurde. Das Fehlen der Angaben ist für die Evaluation einer Indexstruktur oder den Vergleich gegenüber anderen zulässig, da nur Aussagen über die Eignung getroffen werden. In der vorliegenden Arbeit wollen wir jedoch untersuchen, welchen konkreten Einfluss die Datenverteilung auf eine Indexstruktur hat. Dazu ist es notwendig die Verteilung der Daten präzise zu beschreiben um später Rückschlüsse auf das Verhalten der Indexstruktur ziehen zu können. Aus diesem Grund werden für die folgende Evaluation des Einflusses von Datenverteilungen auf Indexstrukturen synthetische Datenverteilungen verwendet. Die für die Evaluation benötigten Datensätze werden mit Hilfe von Wahrscheinlichkeitsverteilungen erzeugt. Die Datenverteilung kann dann anhand der Parametrisierung der Wahrscheinlichkeitsverteilung genau beschrieben werden.

Als Wahrscheinlichkeitsverteilung zur Erzeugung der Datensätze für die Evaluation haben wir die multivariat schiefe Normalverteilung gewählt, wie sie in [2] vorgestellt wird. Mit Hilfe dieser Verteilung können die Zusammenhänge mehrerer normalverteilter Zufallsgrößen beschrieben werden. Einerseits eignet sich die multivariat schiefe Normalverteilung zur Erzeugung von multidimensionalen Daten, die wir für die Untersuchung des Einflusses von Datenverteilungen auf multidimensionale Indexstrukturen benötigen. Anderseits werden in den Evaluationen von Indexstrukturen häufig normalverteilte Daten verwendet [3], [9], [16], u.a. auch deshalb, weil Daten, die in vielen natürlichen Prozessen generiert werden, als normalverteilt angenommen werden können [6], [13], [15]. Wir nutzen die multivariat schiefe Normalverteilung, indem wir jeder Dimension im multidimensionalen Datenraum eine der normalverteilten Zufallsgrößen der multivariat schiefen Normalverteilung zuweisen, die die Verteilung der Datenpunkte in der jeweiligen Dimension beschreibt.

Die multivariat schiefe Normalverteilung bietet drei Parameter, um die Verteilung der Daten zu beeinflussen:

1) Lagevektor ξ: Der Parameter ξ ist ein Vektor, dessen Komponenten die *Erwartungswerte* der Normalverteilung in den jeweiligen Dimensionen beschreiben. Der Erwartungswert ist der Schwerpunkt der Verteilung [12].

2) Kovarianzmatrix Ω: Der Parameter Ω ist eine symmetrische Matrix, deren Diagonalkomponenten $\omega_{X,X}$ die *Varianz* $V(X)$ der Normalverteilung in der jeweiligen Dimension X angibt. Die Varianz beschreibt die Stärke der Streuung um den Erwartungswert [12]. Die übrigen Matrixkomponenten $\omega_{X,Y}$ beschreiben die *Kovarianz* $C(X,Y)$ zwischen den jeweiligen Dimensionen X und Y. Die Kovarianz beschreibt den Zusammenhang zwischen zwei Zufallsgrößen [7]. Somit können Zusammenhänge zwischen zwei Dimensionen in den Datensätze für die Evaluation ausgedrückt werden. Im Kontext der Kovarianz wird auch von *Korrelation* $r(X,Y)$ gesprochen, die den Einfluss zwischen zwei Zufallsgrößen bzw. Dimensionen auf eine Skala von -1 bis 1 normiert. Korrelationen von -1 oder 1 beschreiben einen linearen Zusammenhang. Korrelationen von 0 drücken keinen linearen Zusammenhang aus [12]. Dabei gilt:

$$r(X,Y) = \frac{C(X,Y)}{\sqrt{V(X) * V(Y)}}$$

3) Schiefevektor α: Der Parameter α ist ein Vektor, dessen Komponenten die *Schiefe* der Normalverteilung in der jeweiligen Dimension beschreiben. Dabei können linksschiefe, rechtsschiefe und symmetrische Verteilungen unterschieden werden [7].

Mit Hilfe der drei Parameter ist es möglich völlig unterschiedlich verteilte Daten zu generieren, deren Dimensionen jedoch alle einer Normalverteilung unterliegen (Abbildung 1). In den einzelnen Abbildungen zeigen wir jeweils 10.000 Datenpunkte im zweidimensionalen Raum bei unterschiedlichen Parametrisierungen der multivariat schiefen Normalverteilung, wobei immer nur ein Parameter gegenüber der standard-multivariat schiefen Normalverteilung (Abbildung 1(a)) geändert wurde. Die Verteilungen wurden in den Datenraum [0, 255] in beiden Dimensionen skaliert um eine einheitliche Darstellung zu gewährleisten. Das ist auch der Datenraum in dem später die Evaluation des Einflusses der Datenverteilungen auf die Indexstrukturen stattfindet. Aufgrund der Skalierung der Daten befindet sich das Zentrum des Datenraumes der Datenverteilung stets bei $\xi = \begin{pmatrix} 128 & 128 \end{pmatrix}^T$ für zweidimensionale Daten und spielt keine Rolle bei der Parametrisierung.

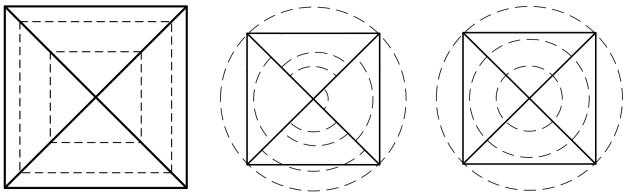
In Abbildung 1(a) zeigen wir standard-normalverteilte Daten. Dabei befindet sich im Zentrum des Datenraumes die höchste Konzentration an Daten. In Abbildung 1(b) illustrieren wir dagegen eine entartete Datenverteilung. Hier wurde das Verhältnis der Varianzen von Dimension 1 und 2 auf $1:50$ erhöht, so dass die Werte in Dimension 1 viel schwächer um das Zentrum des Datenraumes streuen als in Dimension 2. Dadurch entsteht eine achsenparallele Säule. In Abbildung 1(c) zeigen wir Daten, die auf einer Diagonalen im Raum liegen. Diese Datenverteilung wird durch eine hohe Korrelation von 0,8 der Dimensionen erzeugt. Je höher diese gewählt wird, desto linearer wird der Zusammenhang. In Abbildung 1(d) ist eine asymmetrische Datenverteilung zu erkennen, wobei die Verteilungen in beiden Dimensionen stark linksschief sind.

B. Indexstrukturen

Im Folgenden soll die Wahl der Indexstrukturen dargelegt und die zu Grunde liegenden Indexverfahren erläutert werden. Hierbei gehen wir soweit ins Detail wie es für das Verständnis in späteren Kapiteln notwendig ist. Für detaillierte Ausführungen verweisen wir auf die entsprechenden Referenzen.

Nach [17] lassen sich Indexstrukturen in datenraum- und datenaufteilende Verfahren einteilen. Die Pyramidentechnik haben wir als Vertreter der datenraumaufteilenden Indexstrukturen gewählt [4], [14], da sie in [10] ein gutes Laufzeitverhalten bei der exakten Suche zeigte. Bei den datenaufteilenden Verfahren setzen wir den R*-Baum ein [3], eine Variante des in [11] vorgestellten R-Baumes. Hintergrund der Wahl ist die häufige Anwendung in der Praxis, wodurch das Verfahren eine sinnvolle Referenz darstellt. Neben dieser Einteilung der Indexstrukturen findet sich aber auch die in [1] vorgeschlagene Untergliederung in exakte und approximierende Verfahren. Da sowohl die Pyramidentechnik als auch der R*-Baum exakte Verfahren sind, haben wir uns für die Prototyp-basierende Indexstruktur als Vertreter der approximierenden Indexstrukturen entschieden [9].

1) Pyramidentechnik: Das 1998 von Berchtold et al. vorgestellte Verfahren wird als stabiles Indexverfahren auch über höhere Dimensionen hinweg beschrieben [4], [14], d.h. die Suche artet mit steigender Dimensionalität nicht zur sequentiellen Suche aus (Fluch der Dimensionen). Erreicht wird dies, indem der gesamte d-dimensionale Datenraum in $2 * d$ Pyramiden aufgeteilt wird. Die Spitze jeder Pyramide liegt im Zentrum des Datenraumes. In einem zweiten Schritt wird jede einzelne Pyramide in mehrere Ebenen untergliedert, über die die Datenpunkte referenziert werden (Abbildung 2(a)). Die Anzahl der Ebenen ist parametrisierbar. Der zur Evaluation genutzten Implementation liegt die in [14] vorgestellte Variante der Pyramidentechnik zu Grunde, bei der die Pyramiden in Scheiben aufgeteilt werden. Somit wird es möglich kNN-Anfragen effektiv zu unterstützen (Abbildung 2(b)). Abschließend sei angemerkt, dass die Implementation dahingehend abweicht, dass die Scheiben eine fixe Höhe haben und die Anzahl je Pyramide konstant ist (Abbildung 2(c)).



(a) Nach [4] (b) Nach [14] (c) Implementation

Abbildung 2. Datenraumaufteilung der Pyramidentechnik

2) R-Baum:* Der R-Baum ist ein höhenbalancierter Baum, dessen Knoten minimal umgebende Rechteck (Minimal Bounding Rectangle MBR) im Datenraum beschreiben [11]. Das der Wurzel entsprechende MBR umfasst den gesamten Datenbestand mit minimalem Umfang. Mit jeder Stufe im Baum wird eine Untermenge des jeweiligen MBRs gewählt (Abbildung 3).

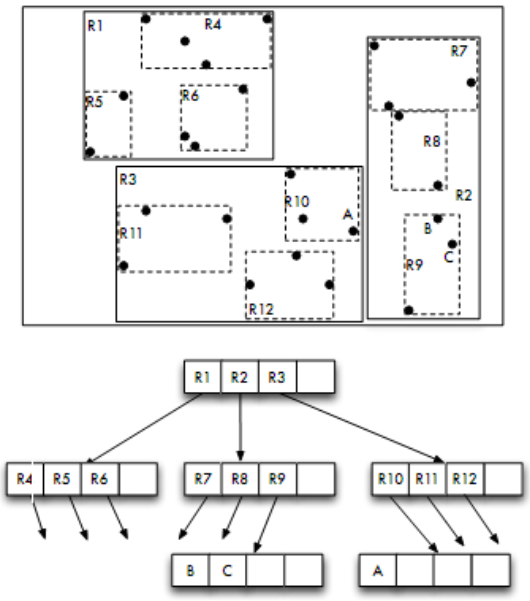


Abbildung 3. Datenraumaufteilung durch den R*-Baum [5]

Beim Einfügen, Ändern oder Löschen von Datenpunkten müssen die jeweiligen MBRs eventuell geteilt oder zusammengefügt werden. Die minimale und maximale Anzahl von Datenpunkten je MBR ist parametrisierbar. Die exakte Suche sollte in logarithmischer Zeit möglich sein, da nur bis zu dem jeweiligen Blatt traversiert werden muss, das den gesuchten Datenpunkt enthält. Mit steigender Anzahl der Dimensionen nimmt allerdings die Überlappung der MBRs zu. Dadurch existieren mehrere Pfade, die den gesuchten Punkt enthalten könnten und somit traversiert werden müssen. Im schlimmsten Fall führt das dazu, dass alle Blätter durchsucht werden

(sequentielle Suche). Das Problem der Überlappung der MBRs versucht der von Beckmann et al. in [3] vorgestellte R*-Baum durch angepasste Einfüge- und Splitalgorithmen zu lösen. So wird beim R*-Baum neben der auch im R-Baum vorhandenen Minimierung der leeren Flächen von Eltern- zu Kindknoten noch auf 3 weitere Kriterien geachtet. Der R*-Baum wird zusätzlich auf minimale Überlappungen, minimalen Umfang der Elternknoten und auf eine möglichst geringe Höhe des Baumes hin optimiert. Diese Variante wird zur Evaluation herangezogen.

3) Protoyp-basierende Indexstruktur: Bei der Protoyp-basierenden Indexstruktur werden eine definierte Anzahl an Datenpunkten zu Protoypen deklariert. Die Wahl der Prototypen aus der Datenmenge findet in der gewählten Implementation zufällig statt, wie dies auch von [9] gehandhabt wurde. Über die Entfernung zu den Prototypen wird die Lage der Datenpunkte im Datenraum approximiert.

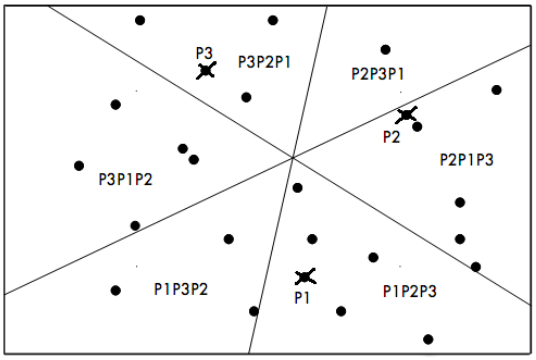


Abbildung 4. Datenraumaufteilung der Prototyp-basierenden Indexstruktur [5]

In Abbildung 4 wird gezeigt, wie sich ein 2-dimensionaler Raum aufteilt, wenn mit dem Prototyp-basierenden Indexverfahren gearbeitet wird. Für jeden Datenpunkt werden die Prototypen entsprechend der Distanz zwischen Prototyp und Datenpunkt aufsteigend sortiert. Die Reihenfolge der sortierten Prototypen wird als Permutation zur Lagebeschreibung des Datenpunktes im Datenraum abgespeichert. Zum Auffinden von Daten wird die zum Anfragepunkt gehörige Permutation bestimmt und diese mit dem Pool der Permutationen verglichen. Basierend auf der Distanz der Permutationen zueinander kann die Menge der Daten, die für das Ergebnis relevant sind, eingeschränkt werden. So sind Punkte mit gleicher oder ähnlicher Permutation wie der Anfragepunkt potentielle Resultate für kNN-Anfragen. Bei der exakten Suche sind natürlich nur Punkte gleicher Permutation von Interesse, die dann auf Gleichheit mit dem Anfragepunkt geprüft werden. Bei der kNN-Suche muss allerdings basierend auf dem geordneten Permutationspool die Distanzberechnung zwischen dem Anfragepunkt und den relevanten Datenpunkten stattfinden. Wie groß der Anteil der zu betrachtenden Datenpunkte ist, kann über einen Parameter definiert werden.

III. Evaluationskonzept

Dem Evaluationskonzept liegt die allgemeine Annahme zu Grunde, dass verschiedene Datenverteilungen die zu untersuchenden Indexstrukturen beeinflussen werden. Wir erwarten, dass der zu beobachtende Einfluss auf die einzelnen Indexstrukturen je Datenverteilung unterschiedlich stark ausfallen wird. Die verschiedenen Datenverteilungen werden mit Hilfe der multivariat schiefen Normalverteilung generiert. Die einzelnen Datenverteilungen unterscheiden sich hinsichtlich der Belegung der Parameter für die Varianz, die Schiefe und die Korrelation. Die Einflüsse der Datenverteilung auf die Indexstrukturen wird anhand des Laufzeitverhaltens bei exakter und kNN-Suche gemessen.

Für die Evaluation des Verhaltens der Indexstrukturen wurden 8 verschiedene Szenarien konzipiert. Zunächst werden im ersten Unterabschnitt die einzelnen Konfigurationen der Pyramidentechnik, des R*-Baum und der Prototyp-basierenden Indexstruktur dargelegt. Anschließend werden die 8 verschiedenen Testszenarien beschrieben und die Parameter benannt. Im Anschluss an die Beschreibung der Szenarien werden im Abschnitt Testdurchführung die Datengenerierung und die Testumgebung erläutert.

A. Indexstrukturen

Für die Verwendung von Pyramidentechnik, R*-Baum und Prototyp-basierende Indexstruktur bei der Evalierung müssen die Indexstrukturen konfiguriert werden. Die konkrete Konfiguration hat wiederum einen Einfluss auf das Laufzeitverhalten der Indexstruktur, so dass die Parameterwahl bedacht getroffen werden muss. Zur Festlegung der Parameter orientieren wir uns an der Parameterwahl in [10]. Für die Pyramidentechnik legen wir eine Partitionierung von 8 Scheiben je Pyramide festlegen. Beim R*-Baum wählen wir die Grenzen mit $min = 2$ und $max = 4$. Das Prototyp-basierende Indexverfahren initialisieren wir mit 8 Prototypen und brechen die kNN-Suche nach der Betrachtung von 4% aller Datenpunkte ab.

B. Testszenarios

Auf Grundlage der in Abschnitt II-A erwähnten verschiedenen Ausprägungen von Datenverteilungen, werden 8 verschiedene Testszenarien entwickelt. Mit den Szenarien sollen standard-normalverteilte, korrelierte, entartete und asymmetrische Datenverteilungen simuliert werden. In den jeweiligen Szenarien wird gezielt ein Parameter (Varianz, Schiefevektor, Korrelation) im Vergleich zu standard-normalverteilten Daten angepasst. Um Trends erkennen zu können wird der jeweilige Parameter in einem weiteren Szenario (im Fall der Korrelation sind es zwei) erhöht. Die 8 verschiedenen Szenarien sind in Tabelle I aufgelistet.

Zur Erzeugung von standard-normalverteilten Daten wird die multivariat schiefe Normalverteilungmit einer Varianz von 1 und ohne weitere Parametrisierung der Schiefe und Korrelation bei der Zufallszahlengenerierung eingesetzt. Die

Symbol	Szenario	Parameter für 2 Dimensionen	Beschreibung
Standard	Standardnormalverteilung	$\Omega=\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$; $\alpha=\begin{pmatrix} 0 \\ 0 \end{pmatrix}$	Referenzwert für die Laufzeitbetrachtung
Var=1:5	Varianz 1:5	$\Omega=\begin{pmatrix} 1 & 0 \\ 0 & 5 \end{pmatrix}$; $\alpha=\begin{pmatrix} 0 \\ 0 \end{pmatrix}$	Leicht entartete Datenverteilung
Var=1:50	Varianz 1:50	$\Omega=\begin{pmatrix} 1 & 0 \\ 0 & 50 \end{pmatrix}$; $\alpha=\begin{pmatrix} 0 \\ 0 \end{pmatrix}$	Stark entartete Datenverteilung
Skew=2	Schiefe 2	$\Omega=\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$; $\alpha=\begin{pmatrix} 2 \\ 2 \end{pmatrix}$	Leicht asymmetrische Datenverteilung
Skew=10	Schiefe 10	$\Omega=\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$; $\alpha=\begin{pmatrix} 10 \\ 10 \end{pmatrix}$	Stark asymmetrische Datenverteilung
Kor=0,2	Korrelation 0,2	$\Omega=\begin{pmatrix} 1 & 0.2 \\ 0.2 & 1 \end{pmatrix}$; $\alpha=\begin{pmatrix} 0 \\ 0 \end{pmatrix}$	Leicht korrelierte Daten
Kor=0,5	Korrelation 0,5	$\Omega=\begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$; $\alpha=\begin{pmatrix} 0 \\ 0 \end{pmatrix}$	Mäßig korrelierte Daten
Kor=0,8	Korrelation 0,8	$\Omega=\begin{pmatrix} 1 & 0.8 \\ 0.8 & 1 \end{pmatrix}$; $\alpha=\begin{pmatrix} 0 \\ 0 \end{pmatrix}$	Stark korrelierte Daten

Tabelle I
ZUSAMMENFASSUNG DER 8 TESTSZENARIEN MIT DEN PARAMETER FÜR 2 DIMENSIONEN

standard-normalverteilten Daten dienen als Referenzwert für die Auswertung um bei der Interpretation des Laufzeitverhaltens eine relative Aussage zu ermöglichen. Für den Fall von entarteten Daten sind zwei Testszenarien vorgesehen, wobei der Wert für die Varianz in der letzten Dimension erst 5 und dann 50 beträgt, während die anderen Dimension konstant den Wert 1 besitzen. Mit der Hilfe von entarteten Daten soll eine Stauchung der Datenverteilung um den Median in mehreren Dimensionen simuliert werden. Zur Simulation von asymmetrischen Datenverteilungen werden über die Anpassung des Schiefevektors zwei weitere Testszenarien generiert. Der Wert für die Schiefe wird dabei gleichmäßig in allen Dimensionen des Vektors erst auf 2 und im zweiten Szenario dann auf 10 festgelegt. Asymmetrische Datenverteilungen sorgen für Freiräume im Datenraum, so dass nicht der gesamte Datenraum mit Datenpunkten besetzt wird. Für die Evaluation des Einflusses von korrelierten Daten wurden drei Testszenarien konzipiert. Der Wert für die Korrelation der einzelnen Dimensionen wird dabei einheitlich über alle Dimensionen gesetzt und beträgt im jeweiligen Szenario erst 0,2 dann 0,5 und schließlich 0,8. Durch die Anwendung von Korrelation sollen lineare Zusammenhänge innerhalb der Datenverteilungen simuliert werden. Die generierten Daten werden abschließend einheitlich über alle Dimensionen in den Bereich [0,255] skaliert.

C. Testdurchführung

Neben den Einflüssen der verschiedenen Parameter wird auch das Laufzeitverhalten der multidimensionalen Indexstrukturen mit steigender Dimension betrachtet. Zu diesem Zweck werden die Dimensionen 2, 5, 9, 12, 15 und 30 betrachtet und Datenverteilungen gemäß den Dimensionen für jedes Testszenario erstellt. Einen weiteren Einfluss auf das Laufzeitverhalten der Indexstrukturen stellt die Datensatzgröße dar. Zur Minimierung des Einflusses der Datensatzgröße und zur Erhöhung der Aussagekraft der Evaluationsergebnisse werden für die 8 Testszenarien und die 6 Dimensionen 3 verschiedene Datensatzgrößen mit jeweils 10.000, 50.000 und 100.000 Datenpunkten erzeugt.

Die Datenpunkte der einzelnen Datensätze werden mit Hilfe von Zufallszahlen unter Verwendung der multivariat schiefen Normalverteilung generiert. Die Generierung der Datenpunkte wurde mit der freien Softwareumgebung R für statistische Berechnungen und Grafiken[2] unter Verwendung der R-Bibliothek für die multivariat schiefe Normalverteilung[3] erstellt. Die Zufallszahlengenerierung kann in R über die Angabe eines Seeds parametrisiert werden. Damit zufällige Verteilungsanomalien die Evaluationsergebnisse nicht verfälschen, werden die 8 Testszenarien in den 6 Dimensionen unter Verwendung von 3 Datensatzgrößen mit 2 verschiedenen Seeds generiert. Damit ergeben sich insgesamt 288 Datensätze. Zur effektiven Generierung der Testdaten haben wir in Java ein Datengenerator entwickelt, welcher das Ausführen von R-Code in einer R-Console aus Java heraus erlaubt.

Für die Evaluation der Indexstrukturen wird das Java-Framework QuEval[4] verwendet, welches so konfiguriert wurde, dass für jeden von der jeweilgen Indexstruktur benötigten Datenpunkt ein Festplattenzugriff durchgeführt wird. Ausgeführt wurde das Framework auf einem Dell Optiplex 980 (Core i5 660, 8 GB RAM) mit installierten Windows 7 SP1 (64bit) und Java 7. Um mögliche Nebeneffekte durch

[2]http://www.r-project.org/, Zugriff: 03.07.2012
[3]http://cran.r-project.org/web/packages/sn, Zugriff: 04.07.2012
[4]http://wwwiti.cs.uni-magdeburg.de/iti_db/research/iJudge/index_en.php, Zugriff: 03.07.2012

Betriebssystem und die virtuelle Maschine von Java zu minimieren, wurde vor der eigentlichen Testdurchführung eine Warmlaufphase durchgeführt. Diese Warmlaufphase bestand darin, für jede Indexstruktur die exakte und kNN-Suche 5-mal für die zweidimensionale, 10.000 Datenpunkte umfassende Standardnormalverteilung durchzuführen. Im Anschluss wurde mit jedem Datensatz und jeder Indexstruktur die exakte und kNN-Suche ($k = 5$) 30-mal durchgeführt, so dass sich für die Ausführung insgesamt 8640 Testdurchläufen ergeben.

IV. Ergebnisse und Diskussion

Die mit Hilfe des QuEval-Frameworks erzeugten Messdaten wurden für die exakte und kNN-Suche pro Datensatzgröße über alle Seeds konsolidiert, um die Einflüsse der unterschiedlichen Datenverteilungen besser untersuchen zu können. Hierzu wurde der Mittelwert über die 30 Durchläufe und zwei Seeds gebildet. Zur besseren Vergleichbarkeit wurden die Laufzeiten in Relation zu den Laufzeiten mit standard-normalverteilten Daten gesetzt, so dass sich die folgenden Angaben über prozentuale Abweichungen immer auf die Laufzeit mit standard-normalverteilten Daten bezieht. Die Auswertung der Ergebnisse erfolgte zunächst anhand der Messdaten für 50.000 Datenpunkte. Hier wurden Trends und Besonderheiten bezüglich der Laufzeiten der Indexstrukturen bei der exakten und kNN-Suche gesucht. Dabei wurde zunächst darauf geachtet, ob eine Datenverteilung über die verschiedenen Dimensionen hinweg zu schlechteren oder besseren Laufzeiten im Vergleich zu den anderen Datenverteilungen führt. Außerdem wurde überprüft, wie sich die Laufzeiten bei verschiedenen Belegungen eines Parameters der multivariat schiefen Normalverteilung verändern. Die gewonnen Erkenntnisse wurden anschließend auf die Messergebnisse für 10.000 und 100.000 Datenpunkte übertragen, um diese zu bestätigen oder zu widerlegen. Im Folgenden werden die haltbaren Erkenntnisse pro Indexstruktur präsentiert und diskutiert.

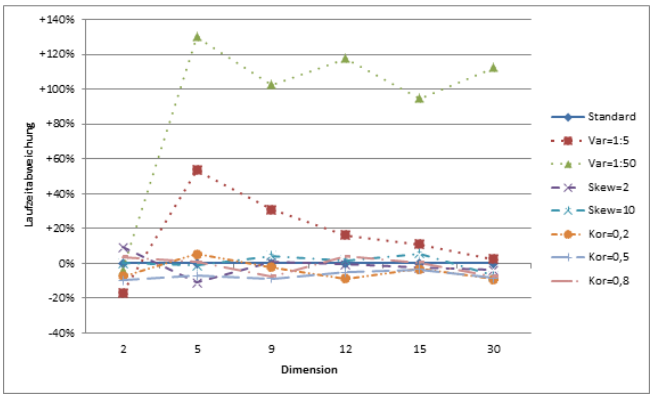
A. Pyramidentechnik

1) Exakte Suche: Die exakte Suche mit Hilfe der Pyramidentechnik führt durchweg zu sehr guten Ergebnissen bezüglich der Laufzeit. Im Falle von standard-normalverteilten Daten beträgt die Laufzeit bei 100.000 Datensätzen unabhängig von der Dimension nie mehr als 0,3 Millisekunden. Damit lässt sich die von Berchthold et al. beschriebene Unempfindlichkeit der Pyramidentechnik gegenüber dem Fluch der Dimensionen bestätigen. Die Laufzeiten bei anderen Datenverteilungen mit Ausnahme der stark entarteten Daten weichen nie mehr als 10% ab. In Abbildung 5(a) zeigen wir die Unterschiede der Laufzeiten über alle Dimensionen in Prozent. Dabei ist deutlich zu erkennen, dass die prozentuale Abweichung der Laufzeit bei stark entarteten Daten unabhängig von der Dimension mindestens 100% beträgt. Als Ursache vermuten wir, dass sich die entarteten Daten vorwiegend in zwei Hyperpyramiden einordnen. Da die Größe der Scheiben der evaluierten Implementation der Pyramidentechnik statisch ist, führt das zu einer Überfüllung von wenigen, einzelnen Scheiben. Dadurch steigt der Aufwand bei der sequentiellen Suche innerhalb einer solchen Scheibe. Das Verhalten lässt sich auch bei weniger stark entarteten Daten erkennen. Jedoch nähert sich die prozentuale Abweichung mit steigender Dimension der Laufzeit mit standardnormalverteilten Daten an, so dass davon auszugehen ist, dass die Datenpunkte besser über die Pyramiden und deren Scheiben verteilt werden. Ähnliches Verhalten zeigt sich auch für asymmetrische und korrelierte Daten.

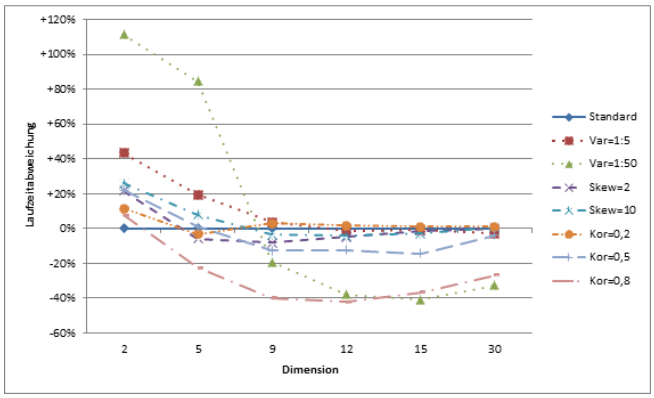
2) k-Nearest-Neighbour-Suche: Bei der Untersuchung der Laufzeiten für die kNN-Suche fällt auf, dass die kNN-Suche mit stark entarteten und korrelierten Daten ab Dimension 9 zu besseren Laufzeiten führt als mit standardnormalverteilten Daten. Die Suche für $k = 5$ mit standardnormalverteilten Daten in 9 Dimensionen dauert bei 100.000 Datenpunkten bis zu 2075 Millisekunden. Im Falle von stark entarteten Daten sind es 1676 Millisekunden und bei stark korrelierten Daten 1251 Millisekunden.In Abbildung 5(b) werden die prozentualen Laufzeitunterschiede bei unterschiedlichen Datenverteilungen gegenüber der Laufzeit mit standard-normalverteilten Daten gezeigt. Wir vermuten, dass die Datenverteilung der stark entarteten und korrelierten Daten dazu führt, dass die Wahrscheinlichkeit, die benötigte Anzahl k Datenpunkten in einer Scheibe zu finden, steigt, da diese besonders dicht beieinander liegen. Dadurch ergibt sich eine Zeitersparnis, da weitere Scheiben nicht durchsucht werden müssen.

B. R*-Baum

1) Exakte Suche: Die exakte Suche mit Hilfe des R*-Baum dauerte bei 100.000 Datenpunkten maximal 28,81 Millisekunden. Die Laufzeiten der anderen Datenverteilungen weichen im Mittel um 10% ab. In Abbildung 6(a) illustrieren wir die Laufzeitabweichungen der einzelnen Datenverteilungen von der Laufzeit mit standard-normalverteilten Daten in Prozent. Als Ausreißer lassen sich korrelierte und stark korrelierte Daten identifizieren. Bei den stark korrelierten Daten beträgt die Abweichung zur Laufzeit mit standard-normalverteilten Daten bis zu 742% bei 100.000 Datenpunkten in 30 Dimensionen. Dabei ist ein Trend über die Dimensionen und über die Stärke der Korrelation erkennbar. Auch bei einer Korrelation von 0,5 sind stärkere Abweichungen als bei den anderen Verteilungen sichtbar. Die großen Laufzeitunterschiede können damit begründet werden, dass die Korrelation der Daten dazu führt, dass die MBRs häufiger überlappen. Damit steigt der Aufwand bei der Suche nach dem gesuchten Datenpunkt. Dieses Verhalten ist bei entarteten Daten nicht erkennbar, was darauf hindeutet, dass der R*-Baum mit dieser Art der Datenverteilung besser umgehen kann. Ursache dafür kann sein, dass die

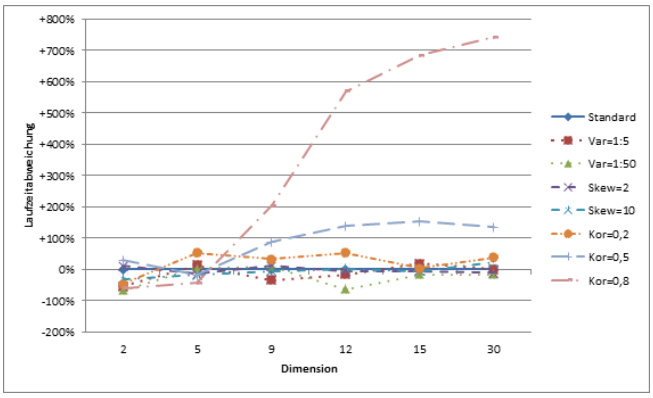


(a) Exakte Suche

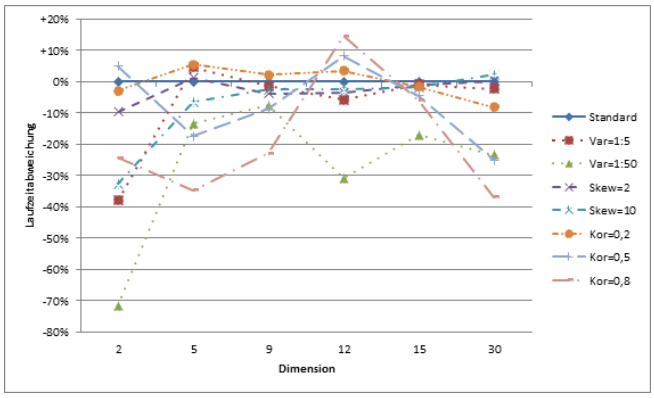


(b) k-Nearest-Neighbour-Suche

Abbildung 5. Pyramidentechnik: Prozentuale Laufzeitabweichungen bei je 100.000 Datenpunkten verschiedener Datenverteilungen gegenüber standard-normalverteilten Daten



(a) Exakte Suche



(b) k-Nearest-Neighbour-Suche

Abbildung 6. R*-Baum: Prozentuale Laufzeitabweichungen bei je 100.000 Datenpunkten verschiedener Datenverteilungen gegenüber standard-normalverteilten Daten

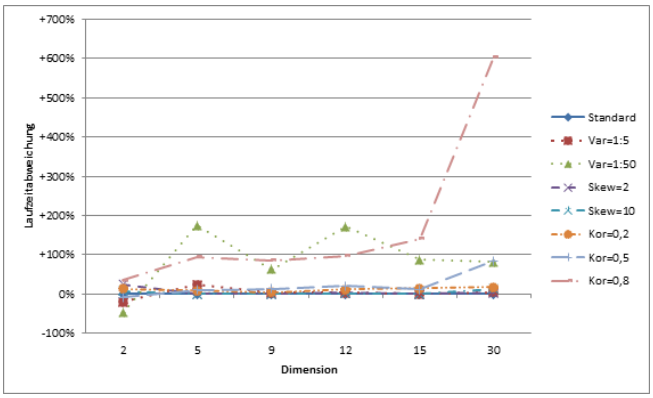
Daten im entarteten Fall achsenparallel im Datenraum liegen und deshalb die MBRs, die auch achsenparallel angeordnet sind, weniger überlappen.

2) k-Nearest-Neighbour-Suche: Für die kNN-Suche lassen sich keine Trends ableiten, da die Abweichungen zu gering sind oder großen Fluktuationen unterliegen um sie eindeutig der Datenverteilung zuzuschreiben (Abbildung 6(b)). Auffällig ist aber dennoch, dass die angenommene Überlappung der MBRs bei korrelierten Daten, die bei der exakten Suche zu teilweise extrem großen Abweichungen von der Laufzeit mit standard-normalverteilten Daten führt, bei der kNN-Suche scheinbar keine Auswirkung hat. Die Begründung liegt darin, dass bei der kNN-Suche mehrere MBRs durchsucht werden müssen, um die gewünschte Anzahl k zu finden. Bei korrelierten Daten müssen bereits wegen der Überlappungen eine Vielzahl MBRs durchsucht werden. Im Falle von nicht korrelierten Daten führt der Verlust der Aussagekraft des Distanzmaßes in hohen Dimensionen dazu, dass ebenfalls eine Vielzahl MBRs durchsucht werden muss. Die beiden Effekte führen zu ähnlichem Aufwand bei der Suche und somit zu vergleichbaren Laufzeiten.

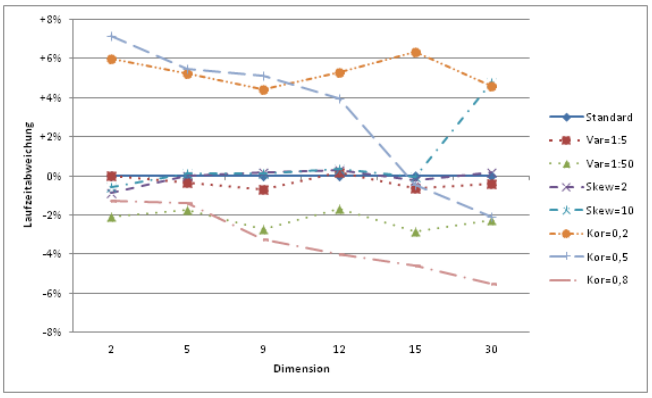
C. Prototyp-basierende Indexstruktur

1) Exakte Suche: Die exakte Suche mit Hilfe der Prototyp-basierenden Indexstruktur dauert bei standard-normalverteilten Daten im Mittel 25 Millisekunden unabhängig von der Dimension. Bis auf stark korrelierte und stark entartete Daten weichen die Laufzeiten mit anderen Datenverteilungen nicht signifikant ab. Für stark korrelierte Daten ergibt sich jedoch bei 30 Dimensionen eine Abweichung von über 600%. In Abbildung 7(a) wird die Laufzeitabweichungen gegenüber standard-normalverteilten Daten in Prozent gezeigt. Wir erklären die auffälligen Abweichung der stark entarteten und stark korrelierten Daten damit, dass viele Datenpunkten der gleichen Permutation an Prototypen zugeordnet werden. Damit teilen die Permutationen die Datenpunkte nicht optimal auf, so dass der Aufwand für den Test auf Gleichheit steigt, da mehr Datenpunkte verglichen werden müssen.

2) k-Nearest-Neighbour-Suche: Bei der k-NN-Suche lassen sich keine signifikanten Abweichungen zur Laufzeit mit standard-normalverteilten Daten feststellen (Abbildung 7(b)). Die Laufzeit im Falle von standard-



(a) Exakte Suche



(b) k-Nearest-Neighbour-Suche

Abbildung 7. Prototyp-basierende Indexstruktur: Prozentuale Laufzeitabweichungen bei je 100.000 Datenpunkten verschiedener Datenverteilungen gegenüber standard-normalverteilten Daten

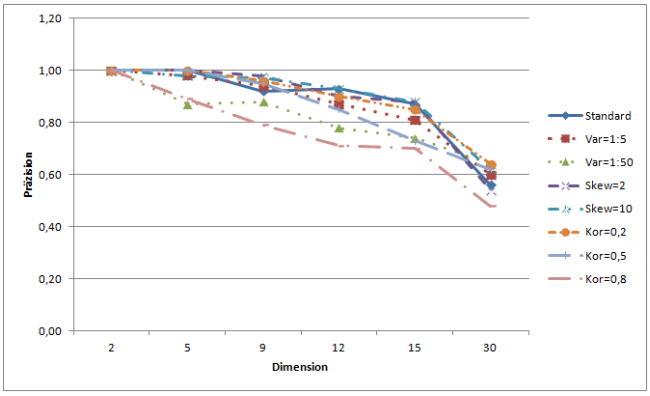


Abbildung 8. Prototyp-basierende Indexstruktur: Präzision der k-NN-Suche bei je 100.000 Datenpunkten verschiedener Datenverteilungen

normalverteilten Daten bei 100.000 Datenpunkten beträgt im Mittel 540 Millisekunden unabhängig von der Dimension. Da nur 4% der Datenpunkte durchsucht werden (Kapitel III-A), ist bei allen Datenverteilungen der Aufwand für die kNN-Suche nahezu gleich. Allerdings ist ein Abfall der Präzision über die Dimensionen festzustellen. In Abbildung 8 illustrieren wir diesen Abfall über die Dimensionen. Da es sich um ein approximierendes Verfahren handelt, ist eine Präzision kleiner 1 nicht überraschend. Allerdings sinkt die Präzision in der vorliegenden Evaluation auf bis zu 0.48 ab. Das deutet daraufhin, dass die Prototyp-basierende Indexstruktur für den Einsatz in Abhängigkeit von der Dimension kalibriert werden sollte. In [10] wurde beispielsweise für jedes Testszenario eine andere Konfiguration der Prototyp-basierenden Indexstruktur angewandt, wodurch die Präzision auch in hohen Dimensionen mindestens 0.9 betrug.

V. Fazit und Ausblick

Das Ziel dieser Arbeit ist es die Auswirkung verschiedener Datenverteilungen auf ausgewählte multidimensionale Indexstrukturen zu überprüfen. Wir konnten in unserer Evaluation feststellen, dass die Laufzeiten für die exakte und kNN-Suche von datenraum- und datenaufteilenden Indexstrukturen durch korrelierte und entartete Daten beeinflusst werden. Stabiles Verhalten zeigen dagegen alle betrachteten Indexstrukturen nur bei asymmetrischen Daten. Bezüglich des Einflusses korrelierter und entarteter Daten wird ein weiterer Trend deutlich. So führen sowohl die korrelierten als auch die entarteten Daten bei der exakten Suche über alle Indexverfahren hinweg zur Verschlechterung der Laufzeit, wohingegen die kNN-Suche mit den gleichen Daten besser abschneidet. Als abschließendes Ergebnis dieser Arbeit sei zudem erwähnt, dass besonders das Prototyp-basierende Verfahren auf zum Datensatz passende Parameter angewiesen ist. An dieser Stelle könnten weitere Nachforschungen betrieben werden, indem neben der Datenverteilung auch die Indexstrukturen angepasst werden. Neben dieser existierenden Einschränkung der Parametrisierung der Indexstrukturen gibt es noch weitere Anknüpfpunkte an diese Arbeit, die betrachtet werden können. So wäre ein insgesamt größer angelegter Testumfang wünschenswert, indem die Anzahl an unterschiedlichen Seeds, Indexstrukturen oder Datenver-

teilungen erhöht wird. Weiterhin sollte die Möglichkeit in Betracht gezogen werden auf eine hardwarenähere Testumgebung zu wechseln, da Java-spezifische Faktoren wie Garbage Collection, Just-in-Time-Compiler und weitere Optimierungen der JVM die Ergebnisse beeinflussen können [8]. Weitere Faktoren die Einfluss auf die Ergebnisse haben, sind das Betriebssystem, die vorliegenden Implementierungen der Indexstrukturen und die Skalierung des Datenraumes.

LITERATUR

[1] A. Andoni, P. Indyk. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *Communications of the ACM*, 51(1):117-122, 2008.

[2] A. Azzalini, A. Dalla Valle. The Multivariate Skew-Normal Distribution. *Biometrika*, 83(4):715-726, 1996.

[3] N. Beckmann,H.P. Kriegel,R. Schneider,B. Seeger. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In *SIGMOD*, pages 322-331, 1990.

[4] S. Berchtold, C. Böhm, H.-P. Kriegel. The Pyramid-Technique: Towards Breaking the Curse of Dimensionality. *SIGMOD Rec.*, 27:142-153, 1998.

[5] D. Broneske. Visuelle Analyse der Raumaufteilung und Bucketauslastung von permutationsbasierten Indexverfahren. University of Magdeburg. 2012.

[6] C.F. Dormann, I. Kühn. Angewandte Statistik für die biologischen Wissenschaften. 2nd Edition. UFZ Leipzig-Halle. 2009.

[7] L. Fahrmeir. Statistik. *Springer-Lehrbuch*, 6th Edition, Springer. 2007.

[8] A. Georges, D. Buytaert, L. Eeckhout, Lieven. Statistically Rigorous Java Performance Evaluation. *Proceedings of the 22nd annual ACM SIGPLAN conference on Object-oriented programming systems and applications*, pages 57-76, 2007.

[9] E.C. Gonzalez, K. Figueroa, G. Navarro. Effective Proximity Retrieval by Ordering Permutations. *IEEE Trans. Pattern Analysis Machine Intelligence*, 30(9):1647-1658, 2008.

[10] A. Grebhahn, M. Schäler, R. Schröter, S. Schulze, V. Köppen,G. Saake. Quantitative Comparisons and Evaluations of High-Dimensional Index Structures.

[11] A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *SIGMOD*, pages 47-57, 1984.

[12] N. Henze. Stochastik für Einsteiger. Vieweg+Teubner. 2008.

[13] J.F. Lawless. Statistical Models and Methods for Lifetime Data. *Wiley Series in Probability & Mathematical Statistics*, John Wiley & Sons. 1982.

[14] D.H. Lee, H.J. Kim. An Effcient Technique for Nearest-Neighbor Query Processing on the SPY-TEC. *IEEE Transactions on Knowledge and Data Engineering*, 15(6):1472-1486, 2003.

[15] W.B. Nelson. Applied Life Data Analysis. Wiley-Interscience. 2005.

[16] R. Weber, S. Blott. An Approximation-Based Data Structure for Similarity Search. Technical Report ESPRIT project, no. 9141, 1997.

[17] R. Weber, H.J. Schek, S. Blott. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In *VLDB*, volume 24, pages 194-205, 1998.