Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Julissa Cusi Juarez

Entwurf und Realisierung eines Empfehlungssystems
auf Basis von Skyline Queries

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Julissa Cusi Juarez

Entwurf und Realisierung eines Empfehlungssystems auf Basis von Skyline Queries

Masterarbeit eingereicht im Rahmen der Masterabschlussarbeit

im Studiengang Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Olaf Zukunft
Zweitgutachter : Prof. Dr. Stefan Sarstedt

Abgegeben am 16.04.2012

Julissa Cusi Juarez

Thema der Masterarbeit

Entwurf und Realisierung eines Empfehlungssystems auf Basis von Skyline Queries

Stichworte

Empfehlungssystem, Skyline Query, Vergleich von Zeichenketten, Book-Crossing

Kurzzusammenfassung

Die Verarbeitung von Skyline Queries ermöglicht die Suche nach Elementen bezüglich verschiedener Präferenzen. Die Ergebnisliste enthält Datenelemente bzw. Skyline Punkte die am besten sowohl alle Suchpräferenzen, als auch eine Untermenge davon, erfüllen. Um eine Skyline Query zu verarbeiten muss jedes Datenelement durch einen nummerischen Punkt in einem n-dimensionalen Raum repräsentiert werden, wobei die Raumdimensionen den Suchpräferenzen entsprechen. Der Skyline Query Ansatz kann an Empfehlungssystemen angewendet werden, in denen für den Benutzer interessante Items, in Bezug auf die Suchpräferenzen, geliefert werden. In dieser Masterarbeit wird ein Empfehlungssystem auf Basis von Skyline Queries am Beispiel von Büchern realisiert.

Julissa Cusi Juarez

Title of the paper

Design and Implementation of a Recommender System based on Skyline Queries

Keywords

Recommender system, Skyline Query, comparison of strings, Book-Crossing

Abstract

Skyline query processing allows search for elements with respect to different preferences. The results contain data elements called skyline points, which best satisfy all the preferences or a subset of them. In order to process a skyline query, each data element must be represented by a numerical point in a multidimensional space, whose dimensions represent the search preferences. The skyline query approach can be applied to recommender systems, where interesting items are presented to the user according to his search preferences. In this master thesis, a recommender system is developed based on skyline queries and applied to a book example.

Inhaltsverzeichnis

1 Einleitung

Die Suche nach Information ist ein wichtiges Untersuchungsfachgebiet geworden. Es gibt verschiedene Techniken, um die Information zu finden und dem Benutzer zu liefern. Die Auffindung geeigneter Informationen kann kompliziert werden, wenn der Benutzer verschiedene Präferenzen und Bedingungen bzw. Suchkriterien hat und es nicht Die Antwort gibt, welche alle Suchkriterien der Anfrage gleichzeitig erfüllt. Allerdings können mehrere Informationen ausgewählt werden, die Teile der Suchkriterien erfüllen. Der Benutzer muss dann entscheiden, welche Information er nimmt.

Empfehlungssysteme unterstützen den Vorgang der Entscheidungsfindung. Sie wählen adäquate Informationen aus einer Datenbank gemäß den Präferenzen und Bedingungen des Benutzers aus. Diese Informationen werden dem Benutzer als Empfehlung geliefert.

Die Verarbeitung von Skyline Queries kann eine gute Methode werden, um Empfehlungen zu finden, da sie für multikriterielle Entscheidungsunterstützung geeignet ist. Ein Verfahren, um Skyline Queries zu verarbeiten, wählt eine Menge von Daten, die am besten eine Anfrage mit mehreren Dimensionen erfüllen, aus. Die Dimensionen einer Skyline Query entsprechen den Suchkriterien des Benutzers.

Das Ziel der Masterarbeit ist zu beweisen, dass die Verarbeitung von Skyline Queries in der Suchmethode eines Empfehlungssystems angewendet werden kann. Dazu soll ein Anwendungsprototyp implementiert werden, wobei die Suchmethode auf Skyline Queries basiert. Der Anwendungsprototyp wird ein vordefiniertes Dataset, welches reale Daten aus der Book-Crossing Community[1], verwendet.

Die Masterarbeit ist in acht Kapitel unterteilt. Nach der Einleitung werden in Kapitel 2 die Grundlagen von Skyline Queries und Empfehlungssystemen erklärt. Diese Konzepte bilden die theoretische Basis der Arbeit. Anschließend wird die Problemstellung und das Szenario

1 http://www.bookcrossing.com/

des Empfehlungssystems in Kapitel 3 definiert. Kapitel 4 präsentiert die Analyse des Systems, wo die Anforderungen und Anwendungsfälle des Systems identifiziert werden und ein Entwurf der Architektur realisiert wird. Mit dem bis dahin aufbereiteten Wissen wird in Kapitel 5 die Konzeption des Systems realisiert. Es werden jeweils die Architektur, die Interaktion zwischen ihren Komponenten und die Entwurfsentscheidungen definiert. Auf der Grundlage der Konzeption wird die Implementierung des Anwendungsprototyps realisiert und in Kapitel 6 beschrieben. Nachfolgend soll eine Evaluierung des Prototyps durchgeführt werden, die Ergebnisse und Schlussfolgerungen der Evaluierung werden in Kapitel 7 behandelt. Schließlich werden die Zusammenfassung dieser Arbeit sowie ein Ausblick auf zukünftige Erweiterungen und Fortführungen in Kapitel 8 präsentiert.

2 Grundlagen

2.1 Skyline Query

Eine Skyline Query ist ein spezieller Anfragetyp, welcher nach einer Menge von Daten sucht, die verschiedene Suchkriterien erfüllen. Eine solche Datenbankanfrage kann mehrere Suchkriterien haben bzw. gewünschte Werte für bestimmte Felder. Jedoch kann es zumeist keine einzige Antwort geben, welche alle Suchkriterien der Anfrage gleichzeitig erfüllt. Hingegen können meist mehrere Antworten ausgewählt werden, die einen Teil der Suchkriterien erfüllen. Der Benutzer muss dann entscheiden, welche Information er nimmt.

Das Skyline Queries Processing ist für Top-k Queries Verarbeitung geeignet sowie für Multi-Kriterien Entscheidungsunterstützung [WANG, 2009] und wird für Benutzerpräferenzen Queries verwendet.

Ein Skyline Query Beispiel wäre Folgendes:

„Die preisgünstigsten Hotels, die sich am nächsten zum Meer befinden“

In dem Beispiel entsprechen „preisgünstig“ und „nah zum Meer“ den Attributen „Preis“ und „Abstand zum Meer“ der Anfrage. Beide Attribute werden die Dimensionen der Anfrage sein. Dadurch kann das Query Dataset in einem n-dimensionalen Raum repräsentiert werden. Eine Skyline Query hat die Eigenschaft nach Minimum oder Maximum Werten für ihre Dimensionen zu suchen.

Eine Skyline Query kann durch folgende Bezeichnung repräsentiert werden:

$$SQ = (q_1, q_2, \ldots, q_n)$$

Wobei „n“, die Anzahl der Dimensionen ist. q_i kann zwei Werte nehmen, entweder ‚Min‘ oder ‚Max‘ [WANG, 2009].

Die Repräsentation der Skyline Query für das oben angeführte Beispiel ist:

SQ = (‘Min’, ‘Min’)

SQ hat zwei Dimensionen (Preis und Abstand zum Meer) und beide suchen nach Minimum Werten. Die Anfrageergebnisse für das Beispiel sind die Daten, die beide Bedingungen am besten erfüllen. Sie sind in einem zweidimensionalen Raum durch Punkte repräsentiert und werden Skyline Punkte genannt. Die Skyline Punkte bilden zusammen eine Linie bzw. die Skyline.

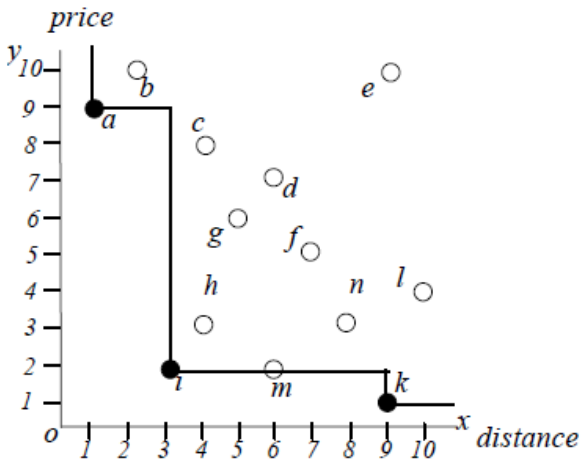

Abbildung 2.1. Beispiel Datenmenge und Skyline [PAPADIAS, 2003]

Abbildung 2.1 stellt Skyline Punkte für das Skyline Query Beispiel dar. Eine Menge von Hotels ist gegeben und jedem Punkt entspricht ein Hotel. Die Entfernung zum Meer ist in die *„distance“* Dimension, und der Preis ist in der *„price“* Dimension repräsentiert. Die Anfrage liefert die Hotels, die näher zum Meer und billiger sind. Die Punkte a, i, und k erfüllen die Anfrage besser als andere Punkte und bilden die Skyline für die Anfrage.

Die Skyline Punkte sind eine Menge von Punkten des Datasets, die von keinem anderen Punkt dominiert sind. Ein Punkt dominiert einen anderen Punkt, wenn er nicht schlechter als der andere in allen Dimensionen und in mindestens einer Dimension besser als der andere ist [WANG, 2009].

Punkt	Koordinate
a	(1,9)
b	(2,10)
c	(4,8)
d	(6,7)
e	(9,10)
f	(7,5)
g	(5,6)
h	(4,3)
i	(3,2)
k	(9,1)
l	(10,4)
m	(6,2)
n	(8,3)

Tabelle 2.1. Dataset Punkte in Koordinaten (angelehnt an [PAPADIAS, 2003])

In Tabelle 2.1 sind alle Punkte des Datasets von Abbildung 2.1 in Koordinaten repräsentiert. Die Skyline Query sucht nach Minimum Werten in der *„price“* Dimension sowie in der *„distance“* Dimension. Die Werte in beiden Dimensionen müssen daher möglichst klein sein. Der Punkt a dominiert den Punkt b, da die a-Werte in beiden Dimensionen kleiner sind als die b-Werte. Der Punkt a dominiert den Punkt c nicht, da der Punkt a nicht besser in der zweiten Dimension als der Punkt c ist. Ebenso dominiert c den Punkt a nicht, da der Punkt c nicht besser in der ersten Dimension als der Punkt a ist. Allerdings wird c von Punkt i dominiert, weil die i-Werte in beiden Dimensionen kleiner als die c-Werte sind. Ebenso i dominiert m, da der Punkt i besser in der ersten Dimension als der Punkt m ist, obwohl beide den gleichen Wert in der zweiten Dimension haben. Nach einer Analyse aller Punkte bleiben die Punkte a, i und k, da sie nicht dominante Punkte sind. Daher sind a, i, und k Skyline Punkte.

2.1.1 Algorithmen

Die ersten Algorithmen um Skyline Queries zu verarbeiten wurden im Jahre 2001 präsentiert. Die Algorithmen Block-Nested-Loops (BNL) [BÖRZSÖNYI, 2001]und Divide-and-Conquer (D&C) [BÖRZSÖNYI, 2001] sind die Basis für Skyline Query Forschungsarbeiten, da die Skyline Query Processing Algorithmen von ihnen entliehen wurden, mit der Verwendung von Indexstrukturen wie R-Tree und B-Tree .

BNL erstellt eine Liste von möglichen Skyline Punkten, die mit dem ganzen Dataset verglichen werden sollen. Die Liste fängt mit dem ersten Tupel des Datasets als einziges Element an und während des Scannens des Datasets können Punkte hinzugefügt oder gelöscht werden. Wenn ein dominierender Punkt gefunden ist, wird er am Anfang der Liste positioniert und dann mit den anderen verglichen, dadurch können mehrere dominierende Punkte schneller gefunden werden und die Anzahl der Vergleiche reduziert werden.

Der D&C Algorithmus partitioniert das ganze Dataset in mehrere Untermengen, innerhalb derer lokale Skyline Punkte für jede Untermenge errechnet werden. Um die finale Skyline Liste zu bestimmen, werden anschließend alle lokalen Skyline Punkte vereint. BNL hat den Nachteil viele unnötige Vergleiche zu realisieren, deswegen wird BNL üblicherweise mittels einer Sortierung der Input Daten vom Sort-Filter-Skyline (SFS) Algorithmus [CHOMICKI, 2003] optimiert.

Der erste Ansatz für die progressive Verarbeitung einer Skyline Query wurde in [TAN, 2001] behandelt, wo die Algorithmen Bitmap und Index präsentiert wurden. Bitmap repräsentiert alle Daten durch Bit-Strings bzw. Bit-Arrays. Somit können die Skyline Punkte durch die Berechnung von Bit-Operationen gefunden werden. Da die Daten zuerst in eine Bit-String Repräsentation umgewandelt werden sollen, sind die Kosten der gesamten Berechnungen ziemlich groß obgleich die Bit-Operationen schnell realisiert werden.

Index ist der erste index-basierende Algorithmus, verteilt die Daten in indexierte Listen, eine Liste für jede Dimension. Die Daten werden in der entsprechenden Liste zu ihrem Minimum Wert zugeteilt. Folgendes Beispiel. Der Punkt a(1,9) wird in die Liste für die erste Dimension eingetragen, und der Punkt b(2,10) in die Liste für die zweite Dimension. Danach werden die Listen gescannt, um die Skyline Punkte zu finden. Dank der Indexierung bzw. der Ordnung der Daten in den Listen wird die Anzahl unnötiger Vergleiche reduziert.

Der nächste Schritt in der Ansätze für progressive Verarbeitung ist die Umwandlung der Input Daten bzw. des Datasets in einen n-dimensionalen Raum. Ebenso wird die Skyline Anfrage durch den genannten Query Punkt in dem dimensionalen Raum repräsentiert. In Anbetracht des vorherigen Beispiels, sucht die Skyline Query nach Minimum Werten in den „price“ und „distance“ Dimensionen, von daher ist (0,0) der kleinste Wert, den diese Dimensionen haben können. Folglich wird (0,0) der Query Punkt sein.

Der Algorithmus Nearest Neighbors (NN) [KOSSMANN, 2002] versucht die Datenpunkte in Bezug auf den Query Punkt zu klassifizieren. NN such nach dem Punkt, der am nächsten zu dem Query Punkt steht. Dieser Punkt wird ein Skyline Punkt sein und wird die Teilung des Datasets ermöglichen. Anschließend wird eine neue NN Suche in den Subspaces durchgeführt und neue Teilungen des Raumes, bis alle Skyline Punkte gefunden sind.

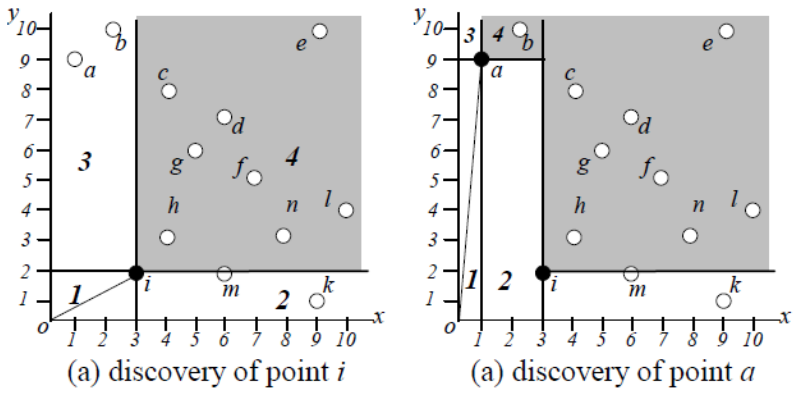

Abbildung 2.2. Skyline Punkte Suche nach NN Algorithmus [PAPADIAS, 2003]

In der Abbildung 2.2.a, wurde im ersten Schritt der Punkt i gefunden und dann vier Bereiche definiert. Der Bereich 4 wird beiseitegelassen, da alle seine Punkte von i dominiert sind. In Abbildung 2.2.b. wird in Schritt zwei, das gleiche Verfahren in den Bereichen 1 und 3 realisiert. Der Punkt a wurde gefunden und wieder werden vier Bereiche definiert, wobei der Bereich 4 beiseitegelassen wird.

Weitere Forschungen arbeiten in den Bereichen der Teilung des dimensionalen Raumes bzw. Subspaces, in der Steuerung von Skylines, besonders im Dataset Aktualisierungsfall, in der Skyline Query Verarbeitung über Data Streams, oder in Subspaces mit Einschränkungen oder auch in einem hohen dimensionalen Raum.

Es gibt auch Forschungsarbeiten in komplexeren Data-Umgebungen wie top-k Skyline [LEE, 2007], die nach der maximalen Anzahl der dominierenden Datenpunkte sucht und für höhe Dimensionalität geeignet ist. Sowie Spatial Skyline [SHARIFZADEH, 2006], die Skyline Punkte unter Berücksichtigung einer Gruppe von Query Punkten als Referenz bzw. Nachbarschaft sucht, unter anderem.

2.1.2 Skyline Operator

Der Skyline Operator [BÖRZSÖNYI, 2001] ist ein Ansatz, um Skyline Query Processing als Erweiterung des Datenbanksystems zu implementieren. Der Operator wird durch die Implementierung der Klausel „SKYLINE OF“ realisiert.

SELECT ... FROM ... WHERE ...
GROUP BY ... HAVING ...
SKYLINE OF [DISTINCT] d_1 [MIN | MAX | DIFF], ..., d_m [MIN | MAX | DIFF]
ORDER BY ...

Wobei d_1 bis d_m die Dimensionen der Anfrage sind. MIN, MAX und DIFF entsprechen den möglichen Werten einer Dimension, Minimum, Maximum und Unterschiedlichkeit. Die Klausel DISTINCT entfernt Duplikate. Die Skyline Operation wird danach kombinierbar mit anderen Operationen wie „join“ oder „top N“.

Für die Implementierung des Skyline Operators schlägt [BÖRZSÖNYI, 2001] sieben Varianten vor. Drei davon basieren auf dem Block-Nested-Loop (BNL) Algorithmus, drei auf dem Divide-And-Conquer (D&C) Algorithmus und die letzte Variante ist ein spezieller Fall einer zwei-dimensionalen Skyline Query.

Derzeit existiert eine öffentliche Implementierung des Skyline Operators: das Projekt „Random Dataset Generator for SKYLINE Operator Evaluation Project“. Dieses wurde vom Forschungsprojekt der PostgreSQL Community entwickelt [PGFOUNDRY, 2009a].

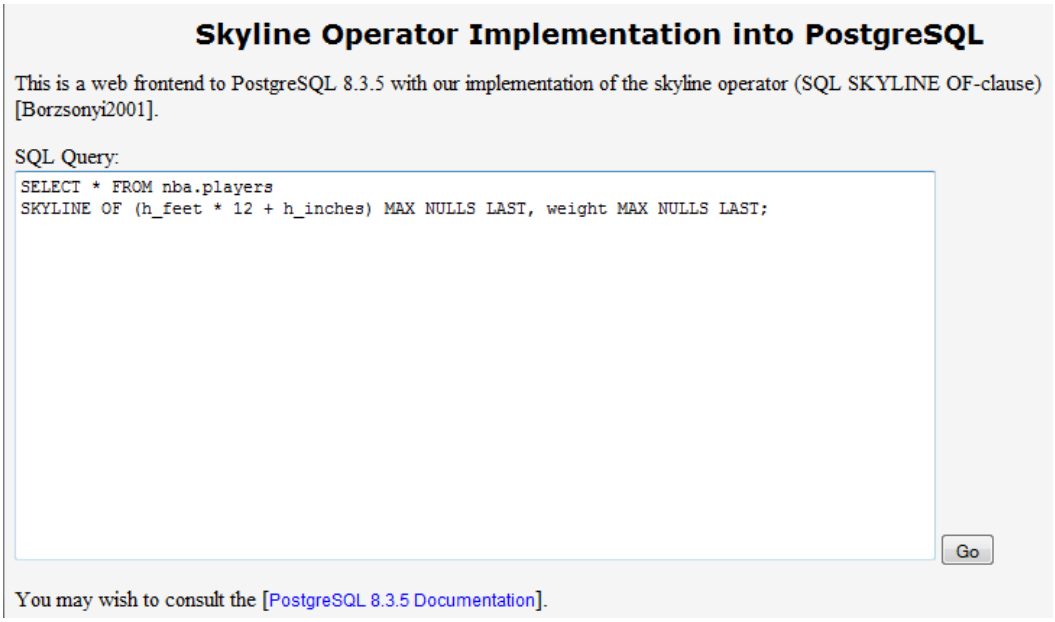

Abbildung 2.3. Implementierung des Skyline Operator [PGFOUNDRY, 2009a]

2.1.3 Skycube

Skyline Query Processing kann zu viele Resultate ergeben, insbesondere wenn das Dataset eine hohe Raumdimensionalität hat um den Bedarf einer großen Anzahl von Benutzern zu erfüllen. Dies kann unerwünschten Processing-Overhead bewirken.

Ein Skycube besteht aus Skylines von allen möglichen nicht-leeren Teilmengen einer gegebenen Dimensionen-Menge. Die Abhängigkeiten der verschiedenen Teile des Skycubes können genutzt werden, um die Berechnung der Skylines zu optimieren.

Um die Definition der Skycube besser zu verdeutlichen, wird nachfolgend das Konzept von Subspaces präsentiert.

- **Subspaces Skyline**

  In einem Dataset mit n-Attributen oder Dimensionen kann man Skyline Queries bezüglich aller Dimensionen bilden oder nur bezüglich einer Untermenge davon. Im Fall einer Untermenge der Dimensionen spricht man von Subspaces [XIA, 2006].

  Beispiel:
  Ein Hotel-Dataset mit drei Attributen bzw. Dimensionen: Preis, Abstand zum Meer und Anzahl der Sterne. Wir betrachten folgende Queries:

  1. **(3-dimensional)**
     Die preisgünstigsten Hotels, die sich am nächsten zum Meer befinden und am meisten Sterne haben.

  2. **(2-dimensional)**
     Die preisgünstigsten Hotels, die sich am nächsten zum Meer befinden.

  3. **(1-dimensional)**
     Die Hotels, die sich am nächsten zum Meer befinden.

  Das erste Beispiel nutzt alle Dataset-Dimensionen. Die nachfolgenden nutzen nur eine Untermenge der Dimensionen.

  Eine Untermenge wird Subspace, seine entsprechende Query wird Subspace Skyline Query und seine Skyline wird Subspace Skyline genannt.

Der Skycube für das vorige Beispiel kann wie folgt dargestellt werden.

Die Dimensionen:
Preis = A, Abstand zum Meer = B, Anzahl der Sterne = C

	A	B	C
t1	40	30	4
t2	50	10	5
t3	10	40	2
t4	30	50	1
t5	20	20	3

Tabelle 2.2. Dataset (angelehnt an [XIA, 2006])

In Tabelle 2.2 wird das Beispiel-Dataset dargestellt. A, B und C sind die Dimensionen und t1 bis t5 sind die Tupel, die die Anfrage am besten erfüllen bzw. Skyline Punkte.

Alle möglichen Kombinationen der verschiedenen Dimensionen sind Subspaces. Jeder Subspace wird im Folgenden „Cuboid“ genannt.

Cuboid	Skyline
ABC	{t1, t2, t3, t5}
AB	{t2, t3, t5}
AC	{t1, t2, t3, t5}
BC	{t2}
A	{t3}
B	{t2}
C	{t2}

Tabelle 2.3. Cuboids (angelehnt an [XIA, 2006])

In Tabelle 2.3 werden alle möglichen Cuboids des Beispiel-Datasets dargestellt. Außerdem werden die Tupel die am besten die Subspace Query erfüllen in der Spalte Skyline präsentiert.

Ein Skycube kann in einer Lattice-Struktur visualisiert werden, sowie ein Datacube in einem Datawarehouse visualisiert werden kann. In diesem Fall wird jede Ebene von unten bis oben inkrementell nummeriert.

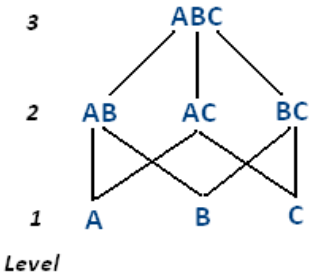

Abbildung 2.4. Skycube - Lattice Struktur (angelehnt an [XIA, 2006])

2.1.4 Compressed Skycube (CSC)

In Tabelle 2.3 wird dargestellt, dass ein Tupel oder Datenobjekt in mehrere Cuboids eines Skycubes gespeichert werden kann. Zum Beispiel ***t2***.

Falls das Dataset aktualisiert wurde, muss man den Cube neu berechnen. Eine neue Cube-Berechnung kann ein Query Processing Bottle-Neck bewirken [XIA, 2006].

Daher wurden das Konzept der Minimum Subspaces erfunden, womit Duplikate gelöscht werden können. Ein Minimum Subspace ist ein Cuboid der nur die Datenobjekte behält, die nicht in seinen eigenen Subspaces definiert sind [XIA, 2006].

Tabelle 2.4 stellt die zusammengefassten Informationen von Tabelle 2.3 ohne Duplikate dar. Zum Beispiel sind die beinhaltenden Tupel in Cuboid ABC gleichzeitig in ihren Subspaces gespeichert, folglich wird Cuboid ABC gelöscht.

Andererseits beinhaltet das Cuboid AB die Tupel t2, t3 und t5. Zwei davon, t2 und t3, sind jeweils in den Subspaces A und B gespeichert, daher wird das Cuboid AB nur die Tupel t5 behalten.

Cuboid	Skyline
AB	{t5}
AC	{t1, t3, t5}
A	{t3}
B	{t2}
C	{t2}

Tabelle 2.4. Cuboids (CSC) (angelehnt an [XIA, 2006])

Nach einer Umordnung der Tabelle 2.4 bezüglich der Tupel lassen sich die Minimum-Subspaces definieren. Die Minimum-Subspaces sind die Cuboids in denen ein Tupel gespeichert ist.

	Minimum Subspaces
t1	{AC}
t2	{B, C}
t3	{AC}
t5	{AC}

Tabelle 2.5. Minimum Subspaces (angelehnt an [XIA, 2006])

Ein Compressed Skycube besteht demnach aus Minimum Subspaces.

2.1.5 Personifizierte Top-k Skyline Queries

Das Ziel von personifizierten Top-k Skyline Queries ist, die idealen k-Anfrageergebnisse auf spezifischer Benutzerpräferenz basierend zu identifizieren. Die Top-k Ergebnisse können durch Navigation durch einen Skycube identifiziert werden. Der Aufbau des Skycubes kann aber Storage- und Berechnungsoverhead bewirken, hauptsächlich, wenn man eine hohe Dimensionalität und große Datenmenge hat [LEE, 2007].

Ein Ansatz, der den Storage- und Berechnungsoverhead eines Skycube zu reduzieren sucht, ist das Framework „Microscope“ [LEE, 2007]. Das Framework „Microscope“ verwendet einen Compressed Skycube, um die Daten zu speichern und eine Skyline Query zu beantwortet.

Microscope navigiert durch einen Compressed Skycube in einer Botton-up Weise. Es fängt in den kleinsten Subspaces an und schreitet dann voran bis zu den Größten. Man kann festhalten, dass ein Space nur nach seinen eigenen Subspaces besucht wird [LEE, 2007].

Abbildung 2.5 stellt eine Lattice-Struktur des CSC des vorherigen Hotel Beispiels dar. In einem Navigationsfall erfolgt der Besuch der Subspaces in folgender Reihenfolge: A, B, AB, C, BC, AC, ABC.

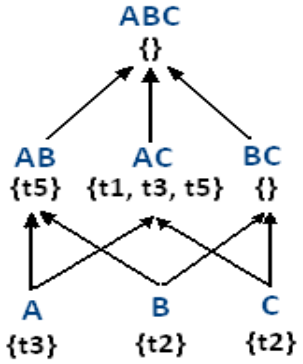

Abbildung 2.5. Lattice Struktur des CSC (angelehnt an [XIA, 2006])

Der Microscope-Algorithmus wird mit dem Besuch jedes Cuboid die Skylines zu seiner eigenen Ergebnismenge hinzufügen.

Die Verwendung eines Compressed Skycube ermöglicht es den Storage-Overhead und den Processing-Overhead in dem Aufbau des Skycube im Vergleich zu einem normalen Skycube zu reduzieren, weil die Daten eine komprimierte Struktur haben und dadurch weniger Operationen nötig sind. Ebenso ermöglicht der CSC den Processing-Overhead in der Verarbeitung einer Skyline Query zu reduzieren, weil Microscope die Skyline Punkte durch eine direkte Navigation durch den Compressed Skycube findet.

2.1.6 Spatial Skyline Queries (SSQ)

Eine Spatial Skyline Query [SHARIFZADEH, 2006] gibt als Ergebnis bezüglich mehrerer Anfragen eine Menge von nicht dominierten Datenpunkten. Eine Anfrage entspricht einem Querypunkt. Daten- und Querypunkte sind in einem n-dimensionalen Raum repräsentiert.

Beispiel:

- In der Suche eines Restaurants für ein Mitarbeitertreffen einer Firma mit mehreren Filialen, müssen eine Menge mögliche Restaurants ausgewählt werden. Die ausgewählten Restaurants müssen die günstigste Alternative für die Mitarbeiter aller Filialen bezüglich der Entfernung zwischen Filiale und Restaurant sein. Hierbei wird jede Filiale ein Query-Punkt sein und die Restaurants Daten-Punkte.

- Sowohl die Skyline Queries (Filialen) als auch die Datenpunkte (Restaurants) werden jeweils durch einen Punkt in dem zweidimensionalen Raum repräsentiert. Dieser Punkt entspricht einer Position im Koordinatensystem.

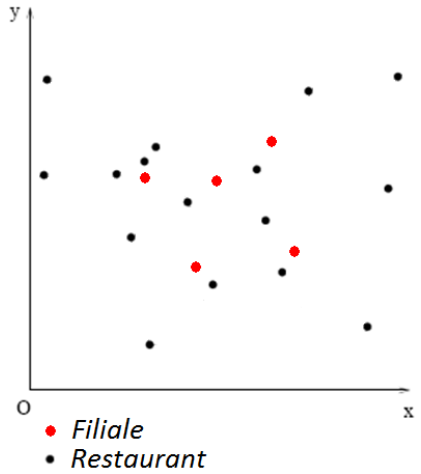

Abbildung 2.6. Beispiel (Spatial Skyline)

Die ausgewählten Restaurants bilden die Spatial Skyline.

Die Spatial Skyline wird in geometrische Strukturen gefasst um besser als SSQ verarbeitet zu werden. Man unterscheidet drei geometrische Strukturen.

- **Convex Hull.** Von mehreren Punkten in einen n-dimensionalen Raum wird die Convex Hull bzw. die konvexe Hülle, also das kleinste Polygon, das alle Punkte umfasst, erstellt.

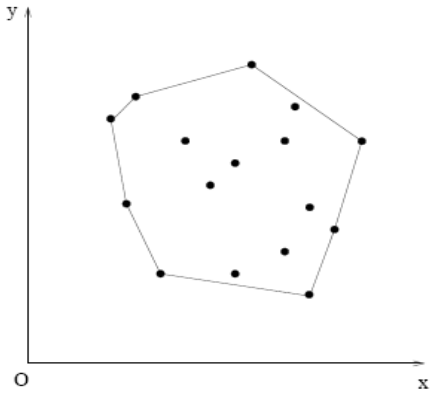

Abbildung 2.7. Convex Hull [LUO, 2004]

- **Voronoi Diagramm.** Das Voronoi Diagramm [JOSWIG, 2008] bildet eine besondere Art von Zerlegung eines Raumes in Regionen, bezüglich einer Menge von Punkten, die als Zentren bezeichnet werden. Jede Region bzw. Voronoi-Zelle wird durch ein

Zentrum bestimmt und umfasst alle Punkte des Raumes, die näher an diesem Punkt als an den anderen liegen. Die Entfernung eines Punktes zu einem Zentrum wird durch die Euklidische-Distanz definiert.

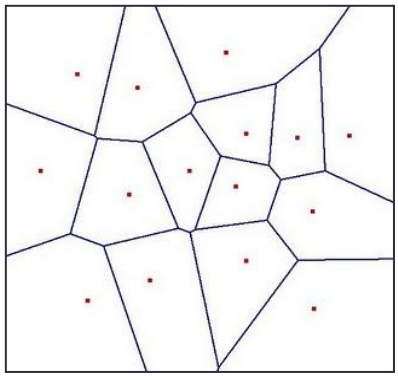

Abbildung 2.8. Voronoi Diagramm

- **Delaunay Triangulierung.** Die Delaunay-Triangulierung [JOSWIG, 2008] ist der duale Graph des Voronoi-Diagramms. Sie zerlegt die Convex-Hull einer Menge von Punkten in Simplizes, wobei jeder Punkt die Ecken der Simplizes[2] sind.

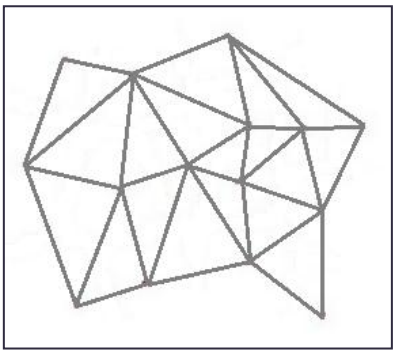

Abbildung 2.9. Delaunaygraph

Eine SSQ lässt sich durch verschiedene, auf den oben genannten geometrischen Strukturen basierende Algorithmen verarbeiten. Es wurde in [SHARIFZADEH, 2006] bewiesen, dass die Punkte, deren Voronoi-Zellen innerhalb der konvexen Hülle sind oder sich mit der konvexen Hülle schneiden, Skyline Punkte sind. Außerdem hat der Standort eines Querypunktes in der konvexen Hülle keinen Einfluss auf die finale Spatial Skyline.

2 Ein Simplex ist die n-dimensionale Repräsentation eines Dreiecks.

2.1.7 Skyframe

Skyframe ist ein Framework für effiziente Verarbeitung von Skyline Queries in P2P Systemen. Sein Ziel ist es den Zeitverlauf der Anfrageverarbeitung zu optimieren, die Netzwerkkommunikationskosten zu reduzieren und die Query-load durch die Peers zu balancieren [WANG, 2009].

Die folgende Abbildung stellt die Skyframe Architektur dar. Die Komponente „Query Processing Manager“ übernimmt die Verarbeitung von Skyline Queries. Die Komponente „Load Balancing Manager“ ist verantwortlich für die Lastverteilung der Queries unter den Knoten. Außerdem hat Skyframe noch weitere supplementäre Methoden, die von beiden Managern benutzt werden [WANG, 2009]. Anschließend wir der „Query Processing Manager“ genauer erläutert, da er wichtiger Bestandteil dieser Masterarbeit ist.

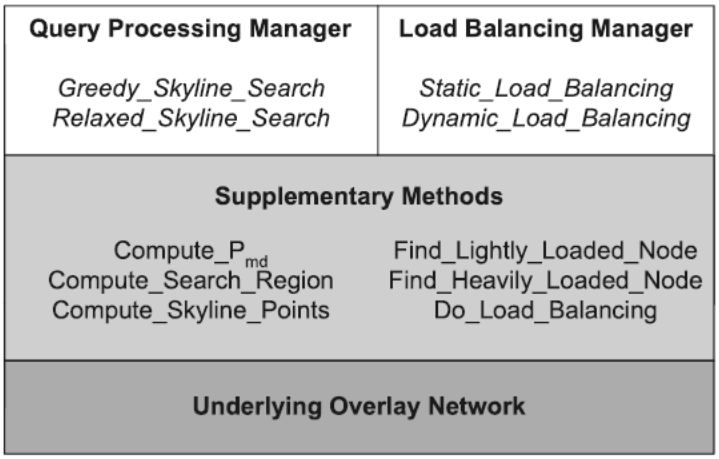

Abbildung 2.10. Skyframe [WANG, 2009]

- **Query Processing Manager**. Sein Ziel ist es den Suchraum zu beschneiden, um unnötigen Knotenzugriff zu vermeiden und die Skyline Punkte zu finden. Der Query Processing Manager hat zwei Suchmethoden, die sich gegenseitig ergänzen und simultan funktionieren: die Greedy und die Relaxed Skyline Search. Um den Greedy Suchbereich zu bestimmen, muss man einen Anfangsknoten bzw. SQ-Starter bestimmen und dort den dominantesten Punkt (pmd) finden. Dieser Punkt definiert den Greedy Suchbereich (s. Abb. 2.11). Diejenigen Knoten, die Daten im Greedy Suchbereich haben, sind SQ-Border Knoten (wie Knoten A, H, E und G in Abb. 2.12) und bilden den Relaxed Suchbereich.

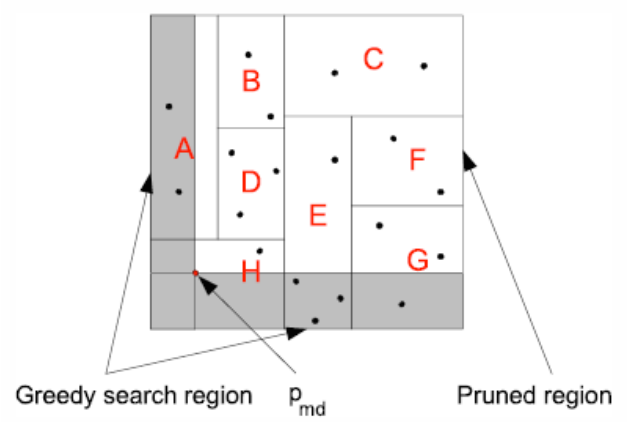

Abbildung 2.11. Greedy Skyline Search [WANG, 2009]

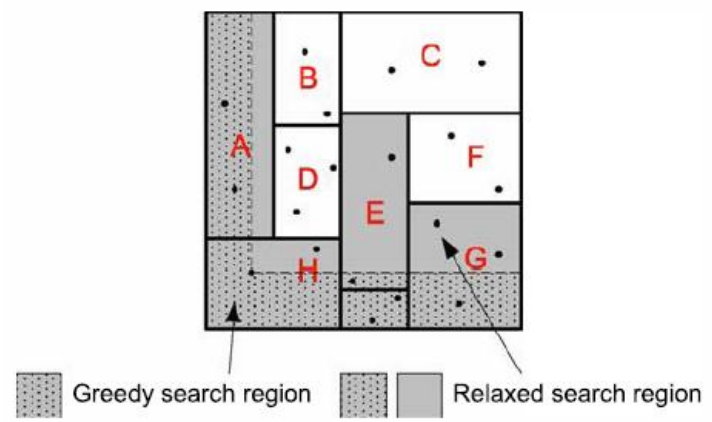

Abbildung 2.12. Relaxed Search Space [WANG, 2009]

2.2 Empfehlungssystem (RS)

Empfehlungssysteme oder Recommender Systeme sind Software Tools und Techniken, die dem Benutzer Items als Empfehlung liefern. Die Empfehlungen helfen in einem Entscheidungsprozess, wie zum Beispiel: welche Items kaufen, welche Musik hören, welche Bücher lesen, usw. [RICCI, 2011]. Ein RS fokussiert sich normalerweise auf ein spezifisches Item (CDs, Filme, Bücher, Hotels, Restaurants, usw.). Normalerwiese haben die Benutzer ihre eigenen Präferenzen, wenn sie etwas auswählen wollen. Von daher werden die Empfehlungen personalisiert, um einer Anfrage die passende Empfehlung zu geben. Folglich muss ein RS die Fähigkeit haben, Präferenzanfragen mit hoher Dimensionalität effizient zu beantworten [YANG, 2011].

Ein bekanntes RS Beispiel ist Amazon.com, in dem die Benutzer Vorschläge für ihren Einkauf bekommen. Die Vorschläge hängen vom Profil des Benutzers ab. Daher ist diese Art der Empfehlungen personalisiert. Eine nicht personalisierte Empfehlung ist einfach zu erzeugen und besteht aus einer Liste von Items, wie zum Beispiel einer Top Ten Liste. Eine nicht personalisierte Empfehlung wird nicht als RS bezeichnet [RICCI, 2011].

Um eine passende Empfehlung zu realisieren, braucht ein RS ein Benutzerprofil. Das Profil besteht aus verschiedenen Eigenschaften der Benutzer wie Alter, Geschlecht, Wohnort oder persönliche Präferenzen über das Item, wie gewünschter Preis, gewünschte Größe, gewünschte Qualität, Bewertungen (von anderen Benutzern), usw. Die Präferenzen können explizit für den Benutzer eingeben werden oder aus früherem Verhalten folgen.

Ein RS oder Empfehlungssystem System kann durch folgende Abbildung von [KLAHOLD, 2009] repräsentiert werden.

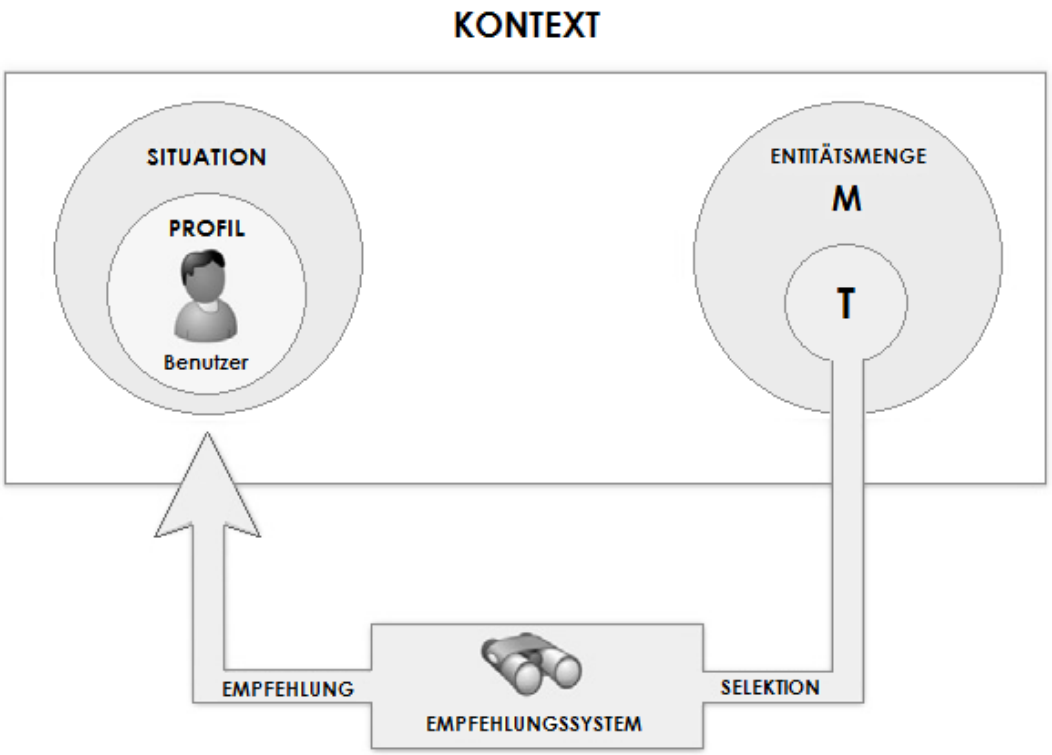

Abbildung 2.13. Empfehlungssystem (RS) [KLAHOLD, 2009]

Das RS empfiehlt einem Benutzer in einem gegebenen Kontext aus einer gegebenen Menge Entitäten bzw. Items eine Teilmenge „nützlicher“ Elemente. Der Kontext konstituiert sich dabei aus dem Benutzerprofil P, der Entitätsmenge M und der Situation S. Die Situation S

konstituiert sich aus Rahmenparametern der realen Welt (Datum, Uhrzeit, Geoinformation, verwendetes Endgerät des Benutzers, gerade angezeigter Text im Browser des Benutzers etc.) [KLAHOLD, 2009].

Die Qualität eines Empfehlungssystems ist primär vom Verfahren zur Selektion der Empfehlungen abhängig. Es sollte alle verfügbaren Informationen (Elemente der Entitätsmenge M) berücksichtigen und aus diesen nur die „nützlichsten“ (für den Benutzer im gegebenen Kontext) als Empfehlung anbieten [KLAHOLD, 2009].

Obwohl e-Commerce mehr und mehr Zuspruch gewinnt, ist der Onlineeinkauf einiger Produkte und Dienstleistungen immer noch eine anspruchsvolle Aufgabe. Nur wenige Plattformen bieten über eine einfache Anfrage-Schnittstelle die gesuchten Items, unter der Annahme, dass Kunden die technischen Details kennen. Empfehlungssysteme sind ein Versuch eine automatische Unterstützung solcher Anfragen zu geben. Ein RS ist nach [FELFERNIG, 2008] interpretiert also ein System, das einen Benutzer auf eine personalisierte Weise zu interessanten oder nützlichen Objekten einer großen Menge Möglichkeiten führt oder solche Objekte als Ausgabe erzeugt.

2.2.1 Knowledge-Quellen

Wie alle intelligenten Systeme verwenden RS unterschiedliche Formen des Wissens. Manchmal ist dieses Wissen implizit, wie die Verteilung der Benutzermeinungen durch Bewertungen oder in einem Algorithmus verschlüsselt. Anderseits kann es durch Ontologie explizit-codiert sein.

Die Quellen, aus denen das notwendige Wissen, um eine Empfehlung zu generieren, extrahiert werden kann, sind: Erstens der Benutzer selbst, Zweitens andere Benutzer des Systems, Drittens die Informationen über empfohlene Items selbst und Viertens schließlich die Empfehlungsdomain selbst bzw. wofür die empfohlenen Items angewandt werden und welche Bedürfnisse sie erfüllen [FELFERNIG, 2008].

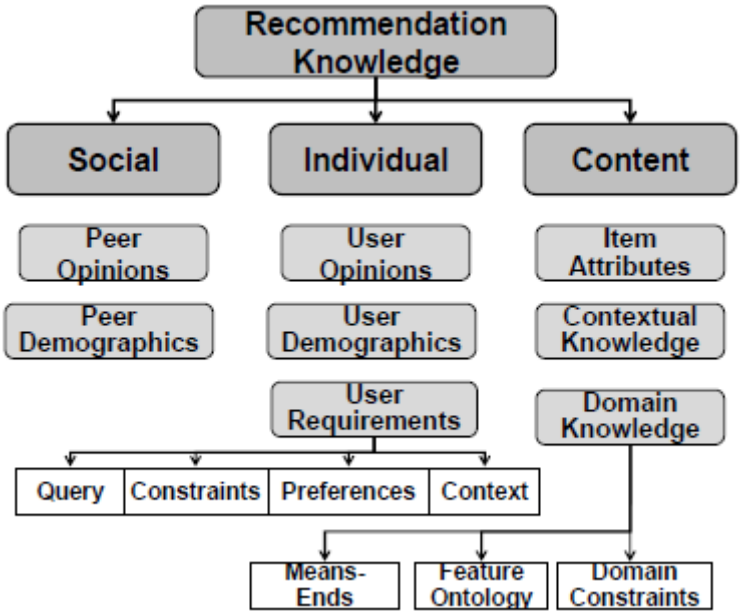

Abbildung 2.14. Knowledge-Quellen in Empfehlungssysteme [FELFERNIG, 2008]

Abbildung 2.14 stellt eine Taxonomie des Empfehlungs-Knowledge dar. Die Informationen, aus denen eine Empfehlung erstellt wird, sind in folgende Kategorien klassifiziert.

- **Social oder Collaborative**
  Das Knowledge entstammt anderen Benutzern (auch Peer-Benutzer genannt). Die Information umfasst sowohl Benutzer Meinungen, die normalerweise durch Bewertungen repräsentiert werden, als auch demografische Daten, die Ähnlichkeiten zwischen Benutzern oder mögliches Verhalten der Gruppe zeigen kann.

- **Individual oder User**
  Das Knowledge umfasst Informationen eines einzigen Benutzers, der die Anfrage stellt und auf eine Empfehlung wartet. Diese Information kann Social werden, wenn der Benutzer nicht mehr der aktuelle Benutzer ist. Ebenso können Präferenzen eines Benutzers historische Informationen werden. Solche Informationen können sehr nützlich sein, um Empfehlungen zu erstellen. Außerdem muss man die Anforderungen des Benutzers berücksichtigen.

- **Content**
  Das Knowledge umfasst Informationen über die zu suchenden Items, mit nur einigen oder vielen Attributen. Kontextuelle Informationen, aus denen sich Schlussfolgerungen ziehen lassen, gehören zu dieser Klassifikation. Ebenso

Informationen über die Domain, das heißt Informationen über das empfohlene Produkt oder die Dienstleistung und die Bedürfnisse.

2.2.2 Klassifikation

Je nach Art der angewendeten Informationen und Algorithmen, um eine Empfehlung zu erstellen, können Empfehlungssysteme in folgende Kategorien klassifiziert werden.

- **Content-based Empfehlungssystem**
  Dieses System verwendet Individual Knowledge. Die empfohlenen Items haben ähnliche Eigenschaften wie Items, die in der Vergangenheit für den gleichen Benutzer ausgewählt wurden. Zum Beispiel, ein Benutzer, der in der Vergangenheit ein Hotel am See ausgewählt hat, erhält eine Empfehlung über ein Restaurant welches am See liegt.

  Die angewendeten Techniken, um eine solche Empfehlung zu erstellen, sind:

  - TD-IDF (Information Retrieval)
  - Clustering
  - Bayes-Klassifikator
  - Entscheidungsbäume
  - Künstliche neuronale Netze

- **Collaborative Empfehlungssystem**
  Verwendet Social Knowledge. Die empfohlenen Items sind solche, die von anderen Peer-Benutzern mit ähnlichen Präferenzen und Vorlieben wie die des aktuellen Benutzers in der Vergangenheit ausgewählt wurden. Zum Beispiel, ein zwanzigjähriger Benutzer, der SciFi Filme sucht, bekommt als Empfehlung die SciFi Filme, die von anderen Benutzern in ähnlichem Alter ausgewählt wurden.

  Collaborative RS bzw. Collaborative Filtering (CF) verwendet verschiedene Techniken, die die Kollaboration von mehreren Agenten, Aussichtspunkten Datenquellen, usw. einbeziehen und normalerweise mit großen Datenmengen angewendet werden. Die zugrunde liegende Annahme des CF-Ansatzes ist, dass wenn der aktuelle Benutzer mit den Präferenzen anderer Benutzern in der Vergangenheit übereingestimmt hat, die Empfehlungen, die auf den Informationen dieser Benutzer basieren, auch für den aktuellen Benutzer relevant sein können CF kann mit verschiedenen Typen von Datasets angewendet werden, z.B. von E-Commerce Anwendungen, Web 2.0 Anwendungen, überwachten Daten (von Sensoren und Geräten), wirtschaftlichen Daten, usw.

Derzeit fokussieren sich Forschungsarbeiten in Bewertungsstrukturen, in denen es Benutzerbewertungen für verschiedene Items gibt. Dies kann durch eine Matrix repräsentiert werden.

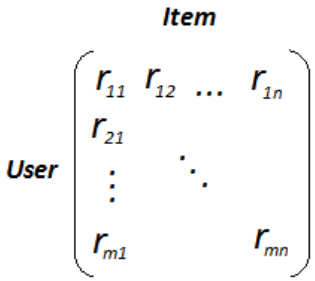

Abbildung 2.15. User x Item $_{(m \times n)}$

Jedes Element $r_{ij}|(1<i<m, 1<j<n)$ der Matrix wird einer Bewertung bzw. Rating eines Benutzers in Bezug auf ein Item zugeordnet. Das Element r_{ij} repräsentiert somit die Bewertung des Benutzers i für das Item j.

Die angewendeten Techniken, um eine solche Empfehlung zu erstellen, sind:

- Nearest neighbor
- Clustering
- Graphentheorie
- Bayessche Netze
- Künstliche neuronale Netze
- Lineare Regression
- Probabilistische Modelle

- **Knowledge-based Empfehlungssystem**
  In dieser Gruppe sind Systeme, die auch andere Informationsquellen außer Individual und Social verwenden. Sie berücksichtigen sowohl Benutzeranforderungen als auch Domaineneigenschaften. Deswegen sind die wichtigsten Merkmale eines Knowledge-based RS, die Benutzersituation und die Bedürfnisse, die die Empfehlungen erfüllen sollen.

  Ein Knowledge-based RS hat die Fähigkeit, neue Lösungen abzuleiten und dem Benutzer Vorschläge zu präsentieren. Auf diese Weise wird eine Art Gespräch bzw. Interaktion zwischen dem Benutzer und dem System etabliert, mit dem Ziel die Suche zu verfeinern und die optimalen Empfehlungen zu ergeben.

- **Hybride Empfehlungssystem**
  Hybride RS sind eine Mischung aus Collaborative und Content-based RS.

2.2.3 Multikriterielles Empfehlungssystem

Viele RS fokussieren sich auf nur ein Suchkriterium. Die erstellten Empfehlungen aus mehreren Suchkriterien (multicriteria decision analysis MCDA) werden kompliziert sein, weil die Wichtigkeit eines Kriteriums nicht für alle Benutzer gleich sein wird. Zum Beispiel ist in der Suche nach einem Hotel, für einen Benutzer der Preis wichtiger als die Lage und für einen zweiten Benutzer wird die Lage des Hotels wichtiger sein als der Preis.

Die meisten Ansätze, die mit dieser Situation umgehen, geben jedem Attribut ein Gewicht. Ein alternativer Ansatz ist die Verwendung von Skylines. Ein Skyline Query kann mehrere Kriterien bzw. Dimensionen haben und eine optimale Antwort bzw. Skyline Punkte geben. Die Skyline Punkte werden die Empfehlungen der Anfrage bilden.

2.2.4 Amazon.com Empfehlungssystem

Amazon.com verwendet die Empfehlungen als ein Marketing-Tool in der Werbung durch E-Mails und auf seiner Webseite.

Abbildung 2.16. Suche nach einer Digitalkamera in Amazon.de [http://www.amazon.de]

Amazon.com erstellt automatische Empfehlungen, die auf einem gesuchten Item basieren. In Abbildung 2.17 liefert das System als Empfehlung ein Angebot des gesuchten Items und dessen Zubehör. Diese Empfehlung basiert auf den Eigenschaften des Items selbst (Content Knowledge).

Abbildung 2.17. Empfehlungen bei Amazon.de [http://www.amazon.de]

In Abbildung 2.18 präsentiert das System ähnliche Items und Zubehör, die von anderen Benutzern gekauft wurden. Diese Empfehlung ist auf Informationen anderen Nutzer basiert bzw. Social Knowledge.

Abbildung 2.18. Empfehlungen bei Amazon.de [http://www.amazon.de]

Um die Empfehlungen zu erstellen hat Amazon.com seinen eigenen Algorithmus entwickelt, welcher Item-to-Item Collaborative Filtering genannt wird. Das Prinzip ist nicht einen Benutzer mit anderen ähnlichen Benutzern, sondern jedes gesuchte Item des Benutzers mit ähnlichen Produkten zu verbinden. Somit wird eine Liste ähnlicher Items erstellt, um dem Benutzer ein personalisiertes Shopping-Erlebnis zu ermöglichen. Dazu erstellt der Algorithmus eine Tabelle von einander ähnlichen Produkten, die normalerweise zusammen gekauft werden. Allerdings ist die Berechnung in Bezug auf Bearbeitungszeit und Speichernutzung aufwendig. Der Algorithmus ermittelt die Ergebnisse daher Offline. Die Bearbeitungszeit hängt somit nur noch von der Anzahl der Produkte, die gesucht werden

sollen, ab. Die Offline-Berechnung ermöglicht die Skalierbarkeit des Algorithmus für große Datenmengen von Produkten und Kunden. Die Online-Berechnung muss nur noch Benutzerspezifische Parameter wie gesuchte bzw. gekaufte Produkte berücksichtigen [LINDEN, 2003].

3 Szenario

In diesem Kapitel wird die Problemstellung des Systems beschrieben, sowie die Herausforderungen. Davon ausgehend und unter Berücksichtigung der Konzepte von Skyline Queries und der auf Book-Crossing vorhandenen Datenbasis[3], wird das Szenario des Empfehlungssystems definiert.

3.1 Bücher Empfehlungssystem

Derzeit sind Empfehlungssysteme sehr präsent. Viele Online Shops bieten dem Benutzer verschiedene Empfehlungen anderer Produkte, die für den Benutzereventuell auch interessant sein könnten. Im Rahmen dieser Arbeit wurde das Szenario eines Empfehlungssystems am Beispiel von Büchern gewählt.

Das Bücher Empfehlungssystem soll einem Benutzer interessante Bücher in Bezug auf seine Präferenzen empfehlen. Um die Empfehlungen zu realisieren, benötigt das System Informationen über die Bücher und die Benutzer.

Das System soll drei verschiedene Empfehlungen präsentieren:

3.1.1 Passende Bücher

Diese Suche soll die Eigenschaften des Buches berücksichtigen. Die ausgewählten Datensätze müssen diejenigen sein, die am besten mit den Suchparametern

3 Vordefiniertes Dataset, das aus der Book-Crossing Comunity (www.bookcrossing.com) stammt.

übereinstimmen. Hier werden die Empfehlungen nach Content Knowledge (s. 2.2.1) gesucht.

3.1.2 Bewertete Bücher in der Vergangenheit

Hier sollen Bücher gesucht werden, die in der Vergangenheit vom aktuellen Benutzer bereits bewertet wurden. Darüber hinaus müssen die ausgewählten Bücher am meisten mit den Suchparametern übereinstimmen. Die Empfehlungen werden nach Individual und Content Knowledge (s. 2.2.1) gesucht. Die Suche soll die Eigenschaften des Buches berücksichtigen sowie Informationen des aktuellen Benutzers.

3.1.3 Bewertete Bücher von anderen Benutzern

Es sollen Bücher gesucht werden, die von anderen Benutzern bewertet wurden. Diese Benutzer sollen möglichst mit dem aktuellen Benutzer und die Bücher am meisten mit den Suchparametern übereinstimmen. Die Empfehlungen werden nach Content und Social Knowledge (s. 2.2.1) gesucht. Die Suche soll Informationen des Buches berücksichtigen sowie Informationen anderer Benutzern.

3.2 Book-Crossing Dataset

Die Daten, die angewendet werden, gehören zu Book-Crossing Dataset aus der Book-Crossing Community. Book-Crossing (www.bookcrossing.com) ist eine weltweite Bewegung von Buchliebhaber zur kostenlosen Weitergabe von Büchern an bekannte, in der Regel aber an unbekannte Personen. Über eine zentrale Datenbank auf der Website des Projekts kann dabei der Weg des Buches von allen vorherigen Besitzern verfolgt werden.

In vier Wochen zwischen August und September 2004 wurden Daten von 278858 Mitgliedern von Book-Crossing und 1157112 Bewertungen, in Bezug auf 271379 unterschiedliche ISBNs gesammelt. Ungültige ISBNs wurden vom Dataset dieser Sammlung ausgenommen. Die Sammlung wurde von Cai-Nicolas Ziegler vom Institut für Informatik der Universität Freiburg realisiert [ZIEGLER, 2005].

Die Daten sind in http://www.informatik.uni-freiburg.de/~cziegler/BX/ erhältlich und sind in drei Tabellen geteilt.

- **BX-Books** hat Daten von Büchern wie ISBN, Buchtitel, Autor, Publikationsjahr, Verlag und Bild.

- **BX-Users** hat Daten von Benutzer wie ID, Wohnort und Alter.
- **BX-Book-Ratings** beinhaltet die gegebene Bewertung (von 1 bis 10) von einem Benutzer für ein bestimmtes Buch. Der Wert ist 0, wenn der Benutzer keine explizite Bewertung für das Buch eingibt.

3.3 Herausforderung

Das System soll dem Benutzer eine Schnittstelle bieten, um die Suche von Büchern zu realisieren. Die Suchparameter sollen den Eigenschaften des Buches wie Titel, Autor, Publikationsjahr und Verlag entsprechen, sowie der durchschnittliche Bewertung des Buches. Außerdem soll sich der Benutzer im System einloggen können, um so die Benutzerinformationen verwendbar zu machen.

Die Suchanfragen sollen mittels Skyline Query Verarbeitung gelöst werden. So entspricht jede Suchanfrage einer Skyline Query und die gefundenen Skyline Punkte entsprechen den Empfehlungen. Um eine Skyline Query zu verarbeiten, sollen alle Daten in der Lage sein, in einem n-dimensionalen Raum dargestellt zu werden, dass heiß, die Daten bzw. Elemente der Dimensionen müssen numerisch sein. Wobei die Dimensionen des Raumes die Suchparameter der Anfrage repräsentieren. Auf dieser Basis soll das System Gemeinsamkeiten zwischen den Daten des Datasets und den Suchparametern des Benutzers berechnen. Dem Resultat soll wiederum ein nummerischen Wert zugewiesen werden.

Der Vergleich der Suchparameter mit den Eigenschaften der Bücher entspricht einer Content-based Methode. Andererseits entspricht der Vergleich mit Berücksichtigung der Benutzereigenschaften einer Collaborative-based Methode (s.2.2.1).

Für die Evaluierung des Systems, sollen andere Skyline Query Ansätze (s.2.1) sowie bekannte Web-Empfehlungssysteme für Bücher wie amazon.de, buecher.de, books.google.de und bookcrossing.com, berücksichtigt werden.

3.4 Abgrenzung

Das System wird die Daten des Book-Crossing Datasets anwenden. Folglich:

- Wird kein Modul implementiert, um die Informationen zu sammeln.
- Die Suchparameter entsprechen nur den Datenfeldern der Tabellen des Book-Crossing Datasets und werden das gleiche Format verwenden.

4 Analyse

Das Ziel dieses Kapitels ist ein detailliertes Verständnis des Projektes zu schaffen. Unter Berücksichtigung des Szenarios das in Kapitel 3 beschrieben wurde, werden die Anforderungen des Systems identifiziert und ein konzeptuelles Model des Systems erstellt.

4.1 Fachliches Klassendiagramm

Anschließend wird das fachliche Klassendiagramm des Projektes dargestellt. Das Diagramm ist auf das Book-Crossing Dataset basiert.

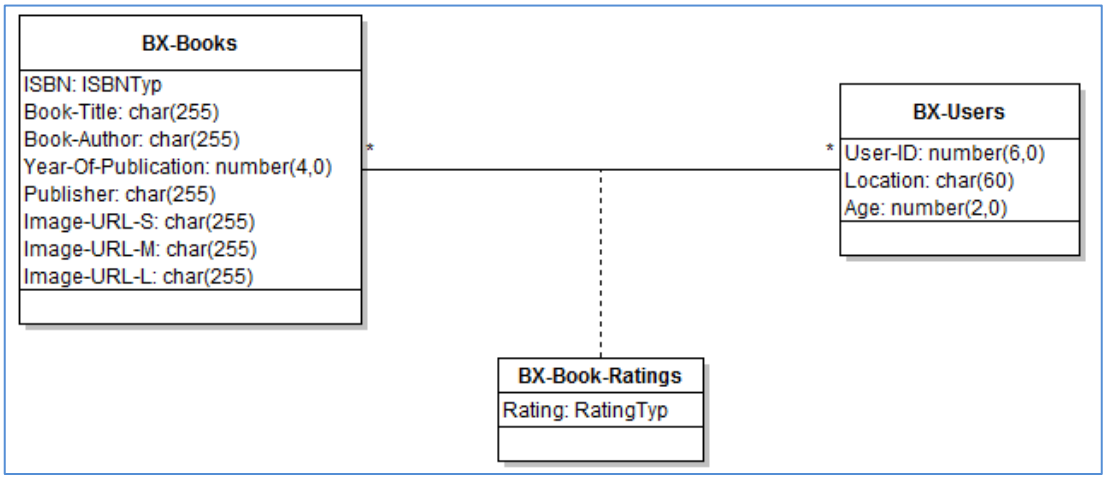

Abbildung 4.1. Fachliches Klassendiagramm

4.2 Funktionale Anforderungen

In diesem Abschnitt werden die wesentlichen Anforderungen, um die richtige Funktionalität des Systems zu garantieren, definiert.

- **FA1:** Das System soll ermöglichen, dass der Benutzer sich in das System einloggen kann.

- **FA2:** Die verfügbaren Suchparameter sollen den Eigenschaften des Buches wie Titel, Autor, Publikationsjahr, Verlag, sowie der durchschnittlichen Bewertung des Buches entsprechen.

- **FA3:** Nachdem der Benutzer Werte für die Suchparameter eingegeben hat, soll das System eine Empfehlungssuche von Büchern in Bezug auf die Suchparameter durchführen.

- **FA4:** Das System soll eine Empfehlungssuche nach „Passenden Büchern“ realisieren. Die Suche wird unter Berücksichtigung der Informationen der Bücher realisiert.

- **FA5:** Das System soll eine Empfehlungssuche nach „Bewerteten Büchern in der Vergangenheit“ realisieren. Die Suche wird unter Berücksichtigung der Informationen der Bücher und des aktuellen Benutzers realisiert.

- **FA6:** Das System soll eine Empfehlungssuche nach „Bewerteten Büchern von anderen Benutzern“ realisieren. Die Suche wird unter Berücksichtigung der Informationen der Bücher, des aktuellen Benutzers sowie anderer Benutzern realisiert.

- **FA7:** Das System soll einen Wert für die Ähnlichkeit zwischen den Daten des Datasets und den vom Benutzer eingegebenen Suchparametern berechnen.

- **FA8:** Das System soll die Suchanfragen mittels der Skyline Query Verarbeitung lösen.

- **FA9:** Das System soll eine Ergebnisliste der Empfehlungssuchen erstellen und dem Benutzer präsentieren.

4.3 Anwendungsfälle

In Bezug auf die Beschreibung des Szenarios (Kapitel 3) und die funktionalen Anforderungen (s. 4.2) wurden die Anwendungsfälle des Systems definiert. Sie werden mittels eines Anwendungsfalldiagramms aus der Modellierungssprache UML[4] in der Version 2.1 nachfolgend darstellen.

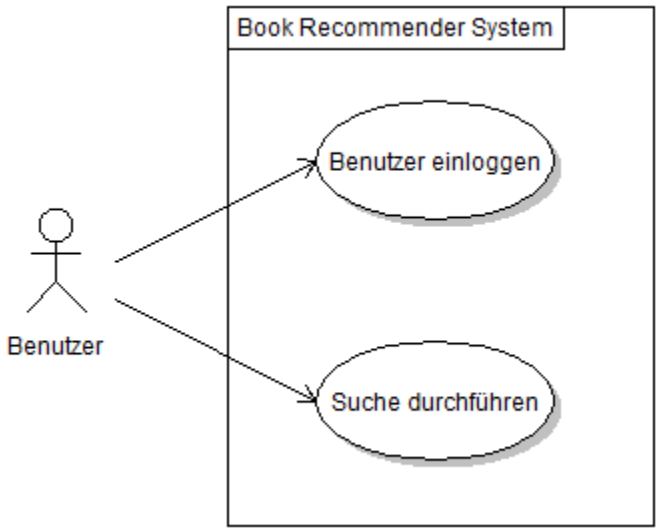

Abbildung 4.2. Anwendungsfälle des Book Recommender Systems

4.3.1 Anwendungsfall „Benutzer einloggen“

Der Anwendungsfall „Benutzer einloggen“ beschreibt den Vorgang des Einloggens eines Benutzers in das System. Dieser Vorgang wird mittels eines Aktivitätsdiagramms nachfolgend dargestellt.

4 Unified Modeling Language (www.uml.org)

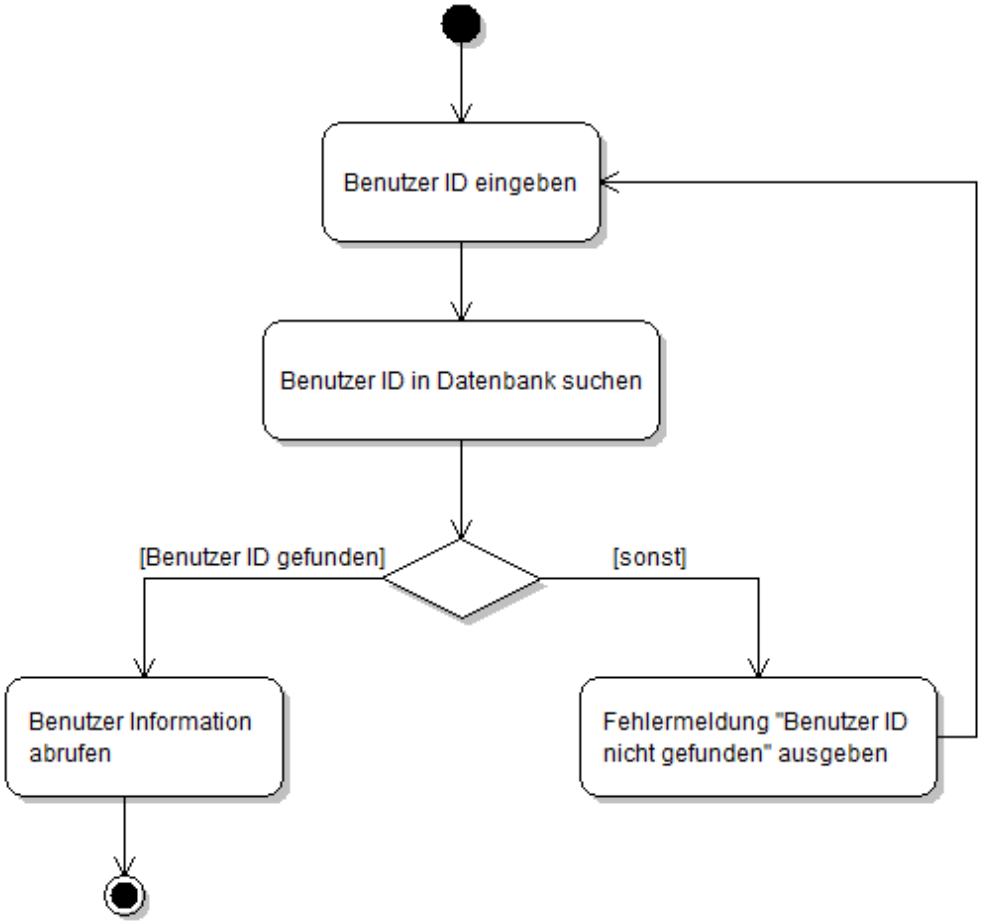

Abbildung 4.3. Aktivitätsdiagramm für Anwendungsfall „Benutzer einloggen“

Anschließend wird die Spezifikation des Anwendungsfalls „Benutzer einloggen“ präsentiert.

Name	Benutzer einloggen
Ziele	Einloggen des aktuellen Benutzers in das System und Abrufen seiner Informationen.
Akteure	Benutzer
Auslöser	Benutzer will sich in das System einloggen.
Vorbedingungen	Der Benutzer kennt seine Benutzer-ID.
Nachbedingungen	Der Benutzer ist eingeloggt und ist der aktuellen Benutzer des Systems. Das System kennt die Informationen des Benutzers.
Erfolgsfall	1. Der Benutzer gibt seine Benutzer ID ein und drückt auf „Login“. 2. Das System überprüft die Existenz der Benutzer ID. 3. Das System ruft die Informationen des Benutzers ab.
Alternative Abläufe	Keine
Fehlerfälle	1.1. Falls die Benutzer-ID ein ungültiger Wert ist, wird er vom System nicht akzeptiert. 2.1. Falls die Benutzer-ID in Datenbank nicht gefunden wird, wird eine Fehlermeldung „Benutzer-ID nicht gefunden“ ausgegeben.
Mengerüst/Häufigkeit	Nicht immer

Tabelle 4.1. Beschreibung Anwendungsfall „Benutzer einloggen“

4.3.2 Anwendungsfall „Suche durchführen“

Der Anwendungsfall „Suche durchführen“ beschreibt den Vorgang der Suche nach Empfehlungen. In diesem Vorgang wird das System mit einem Modul, der den Ähnlichkeitsgrad zwischen den Daten des Datasets und den Suchparametern berechnen wird, im Folgenden „Ähnlichkeitsbewertungs Modul“ genannt, und mit einem Modul, der die Verarbeitung einer Skyline Query realisieren wird, im Folgenden „Skyline Query Processing Modul“ genannt, interagiert. Das folgende Aktivitätsdiagramms beschreibt den Anwendungsfall.

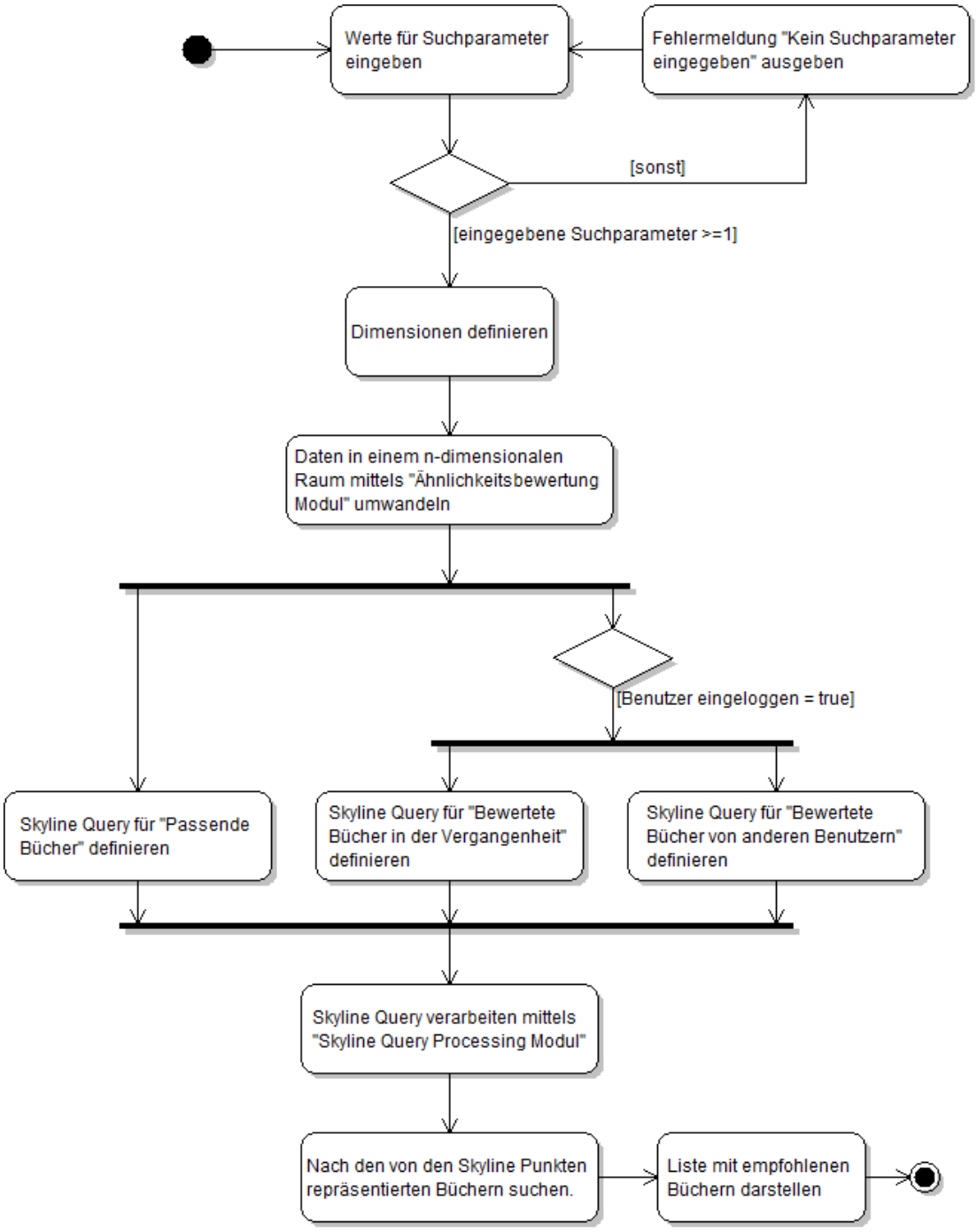

Abbildung 4.4. Aktivitätsdiagramm für Anwendungsfall „Suche durchführen“

Anschließend wird die Spezifikation des Anwendungsfalls „Suche durchführen“ präsentiert.

Name	Suche durchführen
Ziele	Durchführung der Suche.
Akteure	Benutzer
Auslöser	Benutzer will eine Suche durchführen.
Vorbedingungen	Mindestens ein Suchparameter wurde eingegeben.
Nachbedingungen	Die Dimensionen der Anfrage wurden definiert. Die gewünschten Werte wurden ins System übernommen.
Erfolgsfall	1. Der Benutzer gibt Werte für einen oder mehrere Suchparameter ein und drückt auf „Suchen“. 2. Das System überprüft, dass mindestens ein Suchparameter einen Wert hat. 3. Das System definiert die Dimensionen gemäß den Suchparametern. 4. Das System greift auf das Ähnlichkeitsbewertungs Modul zu, um die Elemente der Dimensionen in einem n-dimensionalen Raum umzuwandeln. 5. Das System definiert die Skyline Query für Anfrage „Passende Bücher“. 6. Falls der Benutzer eingeloggt ist, definiert das System die Skyline Query für Anfrage „Bewertete Bücher in der Vergangenheit“. 7. Falls der Benutzer eingeloggt ist, definiert das System die Skyline Query für Anfrage „Bewertete Bücher von anderen Benutzern“. 8. Das System greift auf den Skyline Query Processing Modul zu, um die Skyline Punkte zu finden. 9. Das System sucht die Bücher, die den Skyline Punkten entsprechen und erstellt eine Liste der gefundenen Bücher für die drei Empfehlungssuchen. 10. Das System präsentiert dem Benutzer die Empfehlungen.
Alternative Abläufe	Keine
Fehlerfälle	1.1. Falls die Suchparameter ungültige Werte beinhalten, werden sie vom System nicht akzeptiert. 2.1. Falls der Benutzer die Suche ohne Suchparameter durchführen will, wird eine Fehlermeldung ausgegeben „Kein Suchparameter eingegeben“.
Mengerüst/Häufigkeit	Immer

Tabelle 4.2. Beschreibung Anwendungsfall „Suche durchführen“

4.4 Technische Anforderungen

Angesichts anderer implementierte Skyline Query Ansätze und bekannten Web-Empfehlungssystemen für Bücher wie amazon.de, buecher.de, books.google.de und bookcrossing.com, sowie der angegebenen Ziele dieser Arbeit, werden in diesem Abschnitt die technischen Anforderungen des Systems definiert.

- **TA1:** Die Berechnung der Ähnlichkeit der vom Benutzer eingegebenen Werte und die Daten des Datasets für eine Anfrage mit maximal drei Dimensionen soll nicht mehr als 15 Sekunden dauern.

- **TA2:** Die Verarbeitung einer Skyline Query mit maximal drei Dimensionen soll nicht mehr als 15 Sekunden dauern.

- **TA3:** Die Funktionsweise des Systems darf sich auch unter Betrachtung von mehr als 3 Dimensionen nicht ändern.

- **TA4:** Die Ergebnisse müssen zu mindestens 50 % mit Ergebnissen anderer Bücher-Empfehlungssysteme übereinstimmen.

Da statistisch von den meisten Usern nach nicht mehr als drei Dimensionen gesucht wird, wurde TA1 und TA2 auf drei Dimensionen beschränkt. Um die maximale Laufzeit festzustellen wurden folgende Faktoren berücksichtig.

1. Durchschnittliche Laufzeit einer typischen Buchsuche
2. Technische Eigenschaften der Web-Empfehlungssysteme
3. Technische Eigenschaften der Umgebungen von Skyline Query Ansätze
4. Technische Eigenschaften der eigenen Umgebung

Der gewünschte Ähnlichkeitsgrad aus Anforderung TA4 liegt auf 50%, weil es berücksichtigt, dass sowohl jedes Web-Empfehlungssystem als auch das eigene System eigene Datenbanken verwendet, in denen viele Daten, aber nicht alle ähnlich sind. Der Ähnlichkeitsgrad wird durch Vergleichen der Ergebnisse des Web-Empfehlungssystems und den eigenen Ergebnissen ermittelt.

4.5 Entwurf der Architektur

Nachfolgend wird die Architektur des Systems beschreiben. Die Architektur umfasst die Komponenten: „Informations Sammlung“, „Knowledge“, „Benutzer Schnittstelle“, „Benutzer Verwaltung“, „Recommender Modul“, „Ähnlichkeitsbewertungs Modul“ und

„Skyline Query Processing Modul“. Die genannten Komponenten müssen die Erfüllung der technischen Anforderungen ermöglichen.

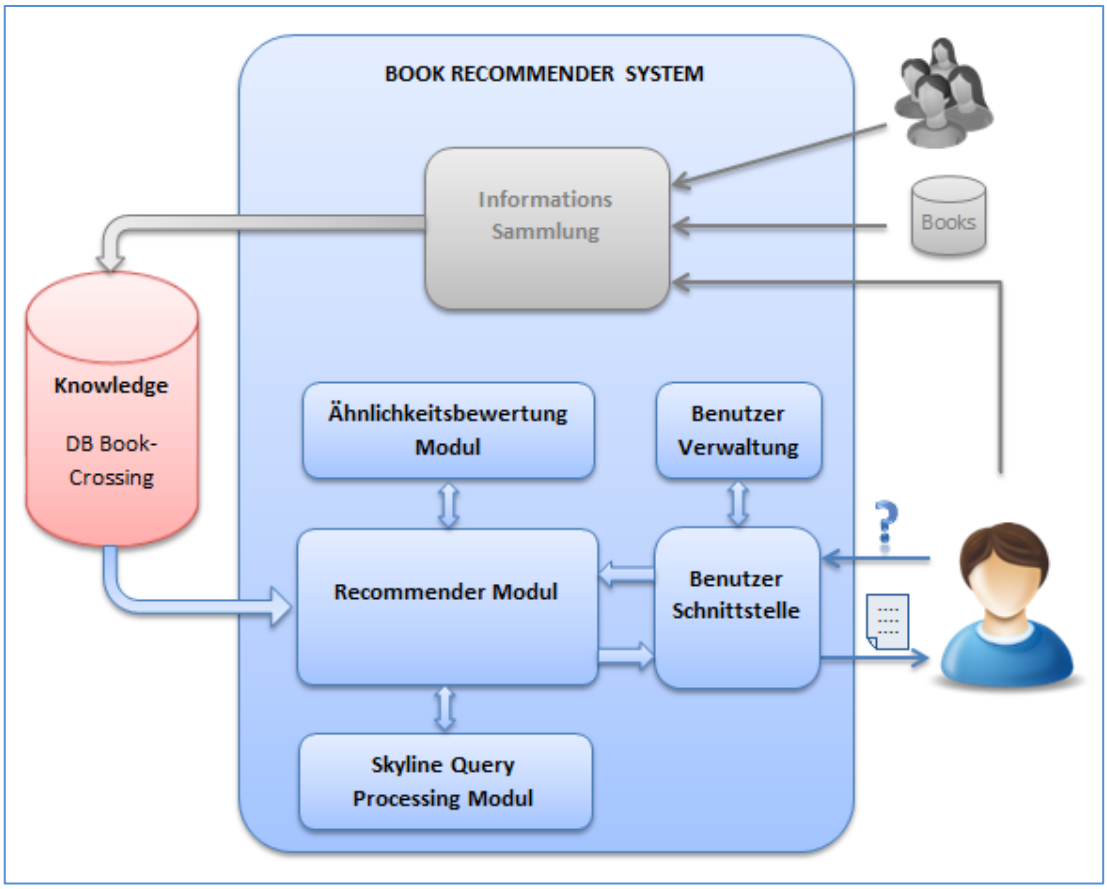

Abbildung 4.5. Architektur des Projektes

Abbildung 4.5 stellt den Entwurf der gesamten Architektur des Systems und die Interaktion zwischen ihren Komponenten dar.

4.5.1 Informations Sammlung

Das Ziel dieses Moduls ist Informationen zu sammeln, um das Knowledge des Systems aufzubauen. Die Informationen kommen aus zwei Quellen: Aus einer Bücherdatenbank und vom Benutzer. Die Benutzer Informationen umfassen die Daten vom Benutzer selbst sowie seine Meinung (Bewertung) bezüglich eines bestimmten Buches. Dieses Modul wird nicht implementiert, da das System das vordefinierte Book-Crossing Dataset verwendet.

4.5.2 Knowledge

Wie oben erwähnt, gehören die verwendeten Daten zum Book-Crossing Dataset (s. 3.2). Das Knowledge des Systems basiert daher auf den Daten des Book-Crossing Datasets.

4.5.3 Benutzer Schnittstelle

Diese Komponente repräsentiert die grafische Schnittstelle des Systems und ermöglicht die Kommunikation zwischen dem Benutzer und dem System. Der Benutzer kann sich durch diese Komponente in das System einloggen und eine Suchanfrage realisieren. Die verfügbaren Suchparameter (Buchtitel, Autor, usw.) werden in dieser Schnittstelle präsentiert. So kann der Benutzer Werte für die gewünschten Suchparameter eingeben. Die Parameter entsprechen den Tabellenfeldern des Datasets. Die Informationen der Benutzer werden zu der Benutzer Verwaltungs Komponente des Einlogg-Prozesses weitergeleitet. Ebenso werden die ausgewählten Parameter und ihre Werte zu dem Recommender Modul weitergeleitet.

Die Benutzer Schnittstelle wird nach der Anfrageverarbeitung das Ergebnis des Recommender Moduls bekommen und dem Benutzer die Empfehlungen präsentieren.

4.5.4 Benutzer Verwaltung

Dieses Modul verwaltet die Informationen der Benutzer und ermöglicht das Einloggen des Benutzers in das System.

4.5.5 Recommender Modul

Dieses Modul ist der Kern des Systems. Es kommuniziert mit der Benutzer Schnittstelle, dem Ähnlichkeitsbewertungs Modul, dem Skyline Query Processing Modul und der Knowledge Datenbank.

Das Recommender Modul erhält die Anfrage des Benutzers. Die Anfrage wird durch die eingegebenen Suchparameter repräsentiert und eine Skyline Query, deren Dimensionen die Suchparameter sind, generiert. Alle Elemente der Dimensionen müssen numerisch sein, um die Skyline Query zu verarbeiten. Aus diesem Grund wird das Recommender Modul zuerst auf das Ähnlichkeitsbewertungs Modul zugreifen, um numerische Werte für die Dimensionen zu definieren, und danach auf das Skyline Query Processing Modul, um die

Skyline Punkte zu finden. Schließlich wird das Recommender Modul die Skyline Punkte in eine Empfehlungsliste umwandeln und dem Benutzer das Ergebnis liefern.

4.5.6 Ähnlichkeitsbewertungs Modul

Um numerische Werte für eine Dimension zu ermitteln, soll dieses Modul die Ähnlichkeit zwischen den vom Benutzer eingegebenen Parametern und jedem Element des Datasets (Dimension), des entsprechenden Tabellenfeldes, berechnen.

Die Elemente können sowohl numerisch als auch textuell sein. Sie werden verglichen und basierend auf ihrer Ähnlichkeit einen numerischen Wert erhalten. Je kleiner der Wert, desto ähnlicher die Werte.

4.5.7 Skyline Query Processing Modul

Mit der Umwandlung der Daten in einem n-dimensionalen Raum kann die Skyline Query verarbeitet werden. Die gewünschten Suchparameter werden die Dimensionen der Skyline Query sein und die Ähnlichkeitswerte werden die Datenpunkte des n-dimensionalen Raumes sein. Nach der Verarbeitung der Skyline Query werden die erhaltene Skyline Punkte zu dem Recommender Modul weiterleiten.

5 Konzeption

Basierend auf der Beschreibung des Szenarios in Kapitel 3 und der Analyse in Kapitel 4, werden in diesem Kapitel eine genaue Beschreibung der Architektur des Systems und die Entwurfsentscheidungen für jede Komponente definiert.

5.1 Definition der Architektur

Die Architektur, die in Abschnitt 4.5 präsentiert wurde, lässt sich durch ein UML Komponentendiagramm wie folgt definieren.

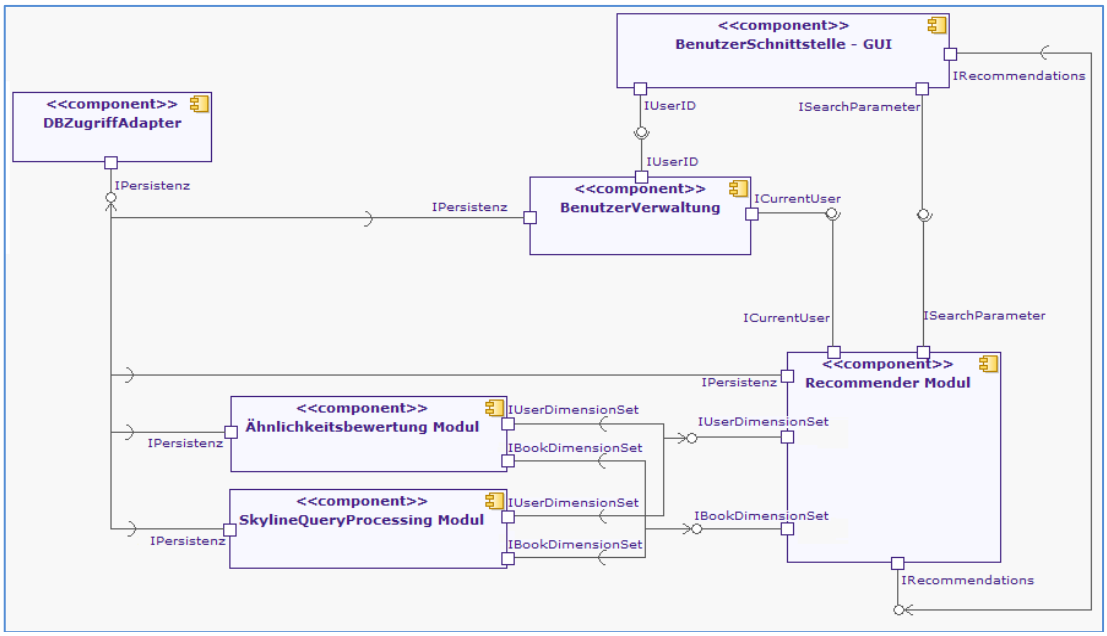

Abbildung 5.1. Komponentendiagramm des Book Recommender Systems

Abbildung 5.1 stellt die Komponenten des Systems und ihre Interaktion durch die Komponentenschnittstellen dar.

Das Empfehlungssystem interagiert durch die „Benutzerschnittstelle-GUI“ mit dem Benutzer, wer die Möglichkeit sich in das System einzuloggen hat. Gemäß der Definition des Empfehlungssystems (s.3.1) benötigen die Empfehlungssuchen „Bewertete Bücher in der Vergangenheit“ und „Bewertete Bücher von anderen Benutzern“ die Informationen des Benutzers. Folglich werden sie neben der Empfehlungssuche „Passende Bücher“, nur wenn der Benutzer eingeloggt ist, ausgeführt. Im umgekehrten Fall wird nur die Empfehlungssuche „Passende Bücher“ ausgeführt.

Die „Benutzerschnittstelle-GUI“ kommuniziert mit der Komponente „BenutzerVerwaltung“ durch die Schnittstelle „IUserID“. Die Komponente „BenutzerVerwaltung“ soll eine Suche des Benutzers realisieren. Wenn er gefunden wird, dann wird der Benutzer als aktueller Benutzer bezeichnet, er ist also am System angemeldet (eingeloggt). Die Informationen des Benutzers sind durch die Schnittstelle „ICurrentUser“ verfügbar.

Vom Benutzer wird erwartet Suchparameter in der „Benutzerschnittstelle-GUI“ einzugeben. Diese Information wird durch die Schnittstelle „ISearchParameter“ an das Recommender Modul weitergeleitet, um den Suchvorgang zu beginnen.

Das Recommender Modul ist verantwortlich für die Definition der Anfragedimensionen und die Bearbeitung der Suche. Ferner werden die Ergebnisse entgegengenommen und die Empfehlungsliste erstellt.

Um die Suchparameter in einen n-dimensionalen Raum umzuwandeln, kommuniziert das Recommender Modul mit dem Ähnlichkeitsbewertungs Modul durch die Schnittstellen „IUserDimensionSet“ und „IBookDimensionSet“. Nachfolgend soll die Ähnlichkeitsberechnung realisiert werden.

Um die Suche durchzuführen, kommuniziert das Recommender Modul mit dem Skyline Query Processing Modul auch durch die Schnittstellen „IUserDimensionSet“ und „IBookDimensionSet“. Das Skyline Query Processing Modul wird die numerischen Daten, die durch das Ähnlichkeitsbewertungs Modul bestimmt werden sollen, verwenden

Anschließend wird das Recommender Modul die Skyline Punkte interpretieren und durch die Schnittstelle „IRecommendations“ an die „BenutzerSchnittstelle-GUI“ weiterleiten, damit sie den Benutzer präsentieren kann.

Die Komponente „DBZugriffAdapter“ ist verantwortlich für den Datenbankenzugriff und Datenbanken Operationen. Sie kommuniziert mit den anderen Komponenten durch die Schnittstelle „IPersistenz“

5.2 Interaktion der Komponenten

Die Interaktion der Komponenten wird mithilfe von Sequenzdiagrammen repräsentiert. Dazu wurden zwei Sequenzdiagramme erstellt, für jeden Anwendungsfall eines.

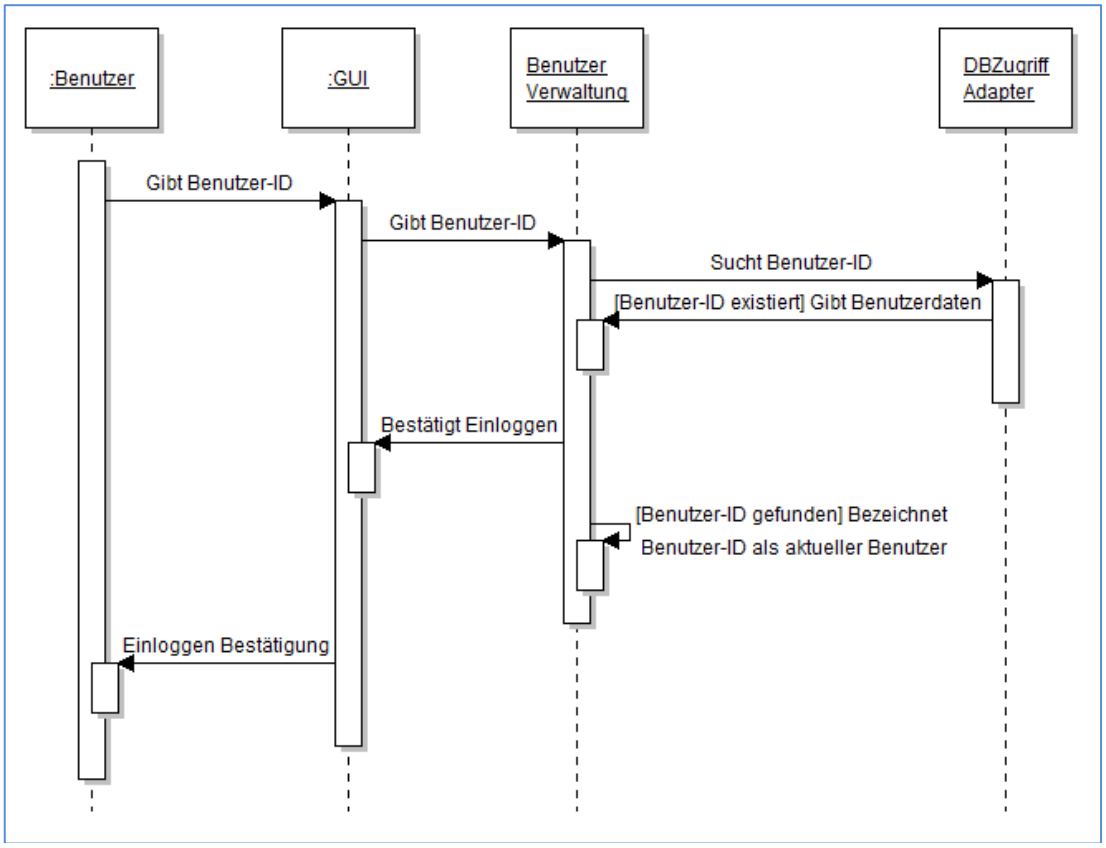

Abbildung 5.2. Sequenzdiagramm für Anwendungsfall „Benutzer Einloggen“

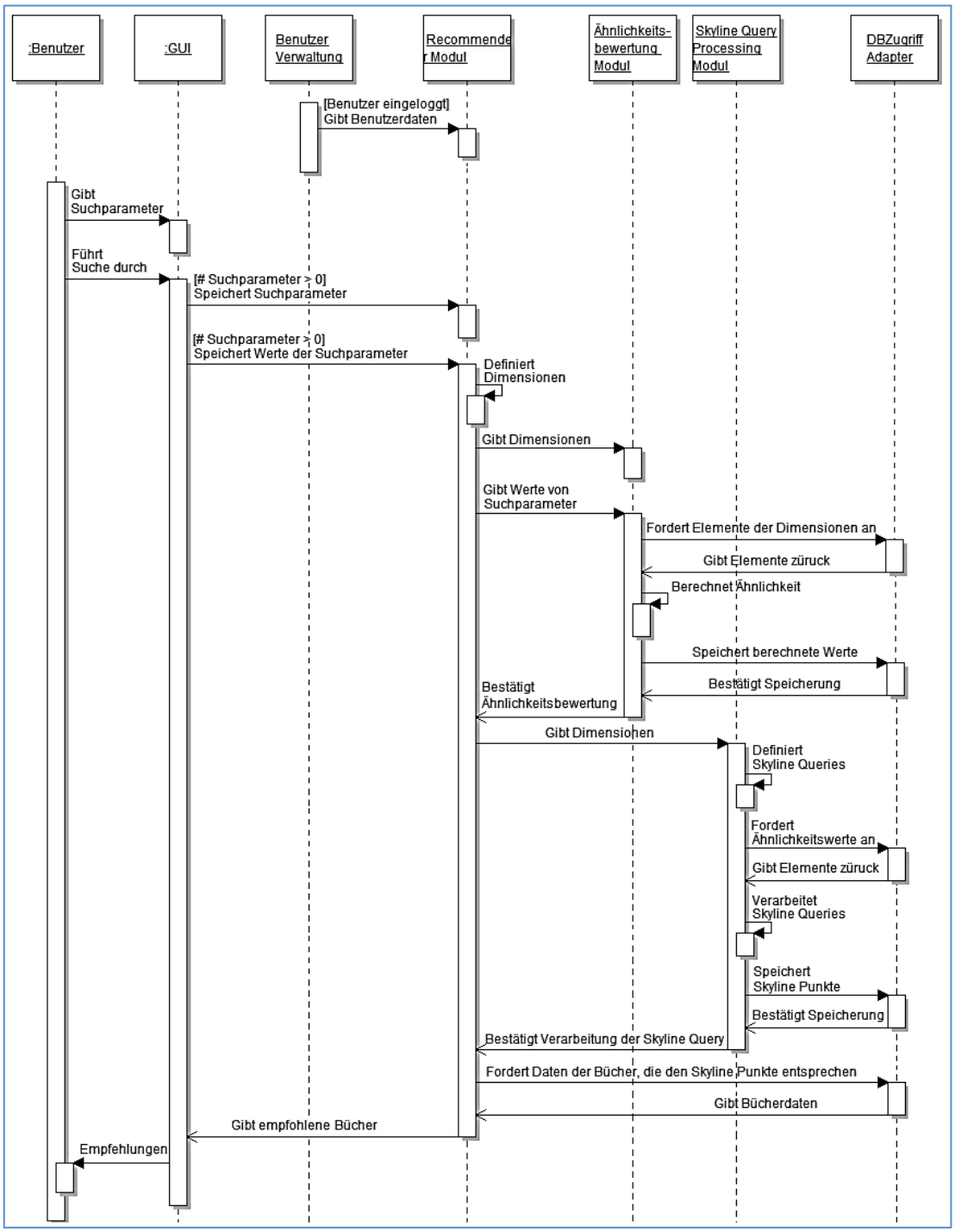

Abbildung 5.3. Sequenzdiagramm für Anwendungsfall „Suche durchführen“

5.3 Komponente Benutzer Schnittstelle

Das Empfehlungssystem benötigt bestimmte Informationen, um eine Suche zu realisieren. Die Informationen können in der Schnittstelle der Anwendung für den Benutzer eingegeben werden. Verantwortlich dafür ist die Komponente „Benutzerschnittstelle – GUI“. Sie übernimmt die Kommunikation mit dem Benutzer. Die Komponente soll Informationen des aktuellen Benutzers empfangen sowie Werte für die Buchsuche.

Abbildung 5.4. stellt das technische Klassendiagramm dieser Komponente dar. Die Komponente „Benutzerschnittstelle – GUI“ enthält die Klasse `BenutzerSchnittstelle`, die verantwortlich für die Kommunikation mit dem Benutzer sein soll. Die Klasse wird die für den Benutzer eingegebenen Informationen bekommen, ihre Korrektheit mittels der Methoden `isUserIdValid()` und `isSearchParameterValid()` überprüfen und sie an die nächsten Komponenten weiterleiten. Dazu soll die Klasse „Benutzerschnittstelle – GUI“ die Schnittstellen `IUserID` und `ISearchParameter` implementieren.

Darüber hinaus wird die Klasse den Login Status mithilfe der Methoden `showLogin()` und `showLogout()` beschreiben.

Schließlich wird die Klasse die gefundenen Empfehlungen durch die Schnittstelle `IRecommendations` erhalten und mittels der Methode `showRecommendations()` dem Benutzer präsentieren.

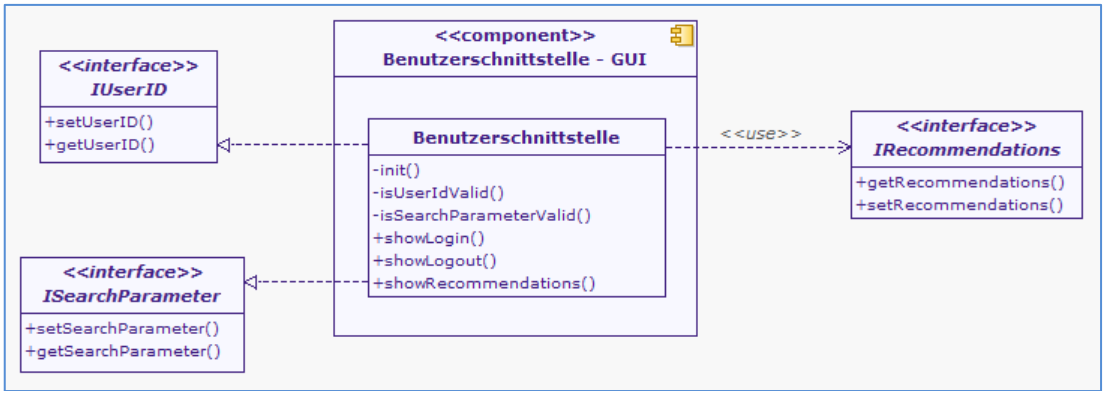

Abbildung 5.4. Technisches Klassendiagramm für Komponente „BenutzerSchnittstelle-GUI“

5.4 Komponente Benutzerverwaltung

Diese Komponente ist verantwortlich für die Verwaltung der Benutzerdaten. Sie bekommt die ID des Benutzers und anschließend soll dieser Benutzer in dem Dataset gesucht werden, um das Einloggen in das System zu ermöglichen und um seine Informationen abrufen zu können. Ein eingeloggter Benutzer wird als aktueller Benutzer bezeichnet.

Abbildung 5.5. stellt das technische Klassendiagramm der Komponente dar. Sie enthält die Klasse `CurrentUser`, die die Suche des Benutzers mithilfe der Methode `getUserRecord()` realisieren wird. Die Klasse wird die ID des Benutzers durch die Schnittstelle `IUserId` empfangen. Die Schnittstelle `ICurrentUser` ist implementiert, um die Daten des aktuellen Benutzers zurückzugeben. Der Zugriff auf die Datenbank wird durch die Komponente DBZugriffAdapter realisiert. Die Schnittstelle `IPersistenz` wird für Kommunikation zwischen beiden Komponenten verwendet.

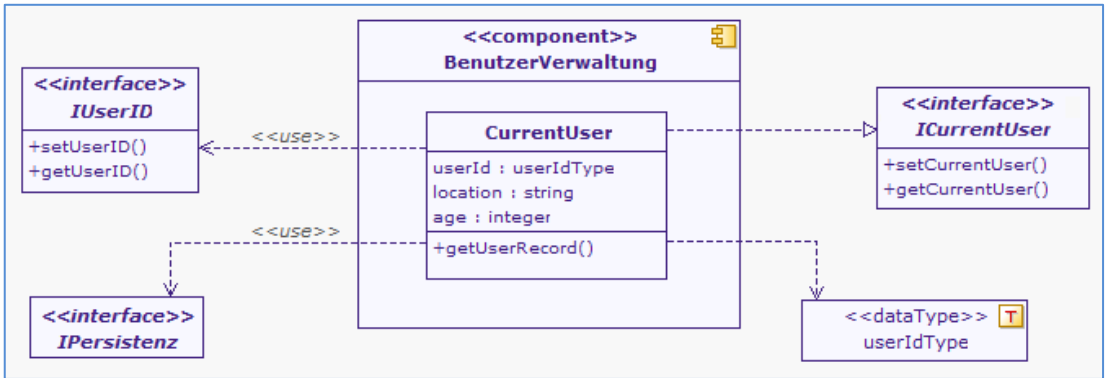

Abbildung 5.5. Technisches Klassendiagramm für Komponente „BenutzerVerwaltung“

5.5 Komponente Recommender Modul

Das Recommender Modul ist verantwortlich für die Definition der Anfragedimensionen. Sie werden aus den Daten des aktuellen Benutzers und den für den Benutzer eingegebenen Suchparametern, die durch die Schnittstellen `ICurrentUser` und `ISearchParameter` erhalten werden sollen, definiert. Das Recommender Modul wird die Schnittstellen `IUserDimensionSet` und `IBookDimensionSet` implementieren, um die definierten Dimensionen zur Verfügung zu stellen. Anderseits wird die Komponente nach der Verarbeitung der Skyline Query die Liste der Empfehlungen mithilfe der Methode `getBookInformation()` erstellen. Dafür implementiert sie die Schnittstelle `IRecommendations`, um die Empfehlungsliste zur Verfügung zu stellen.

Die folgende Abbildung stellt das technische Klassendiagramm für die Komponente „Recommender Modul“ dar.

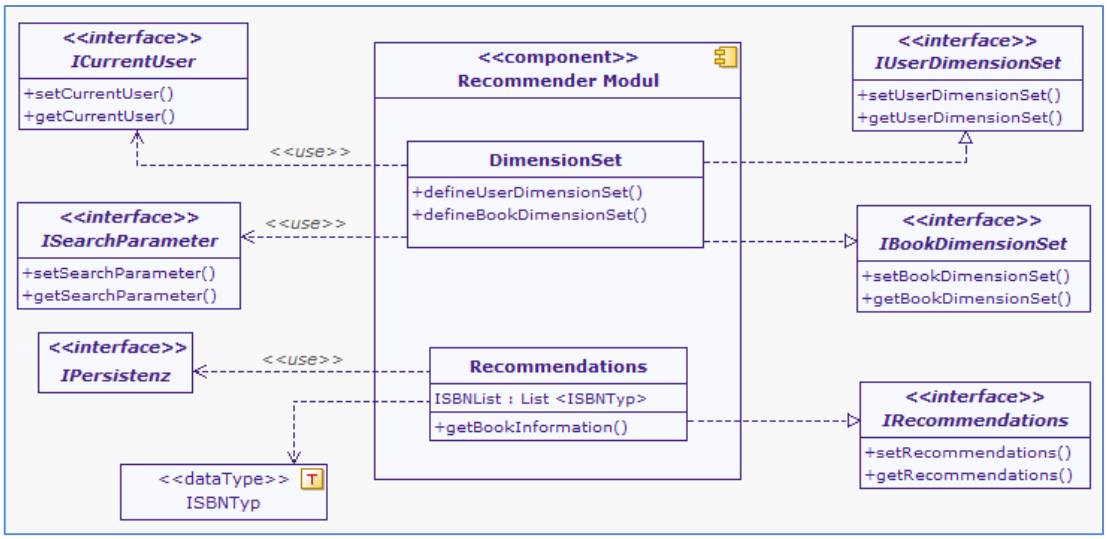

Abbildung 5.6. Technisches Klassendiagramm für Komponente „Recommender Modul“

5.6 Komponente Ähnlichkeitsbewertungs Modul

Dieses Modul bestimmt einen numerischen Wert für die Ähnlichkeit zwischen Elementen. Solche Elemente sind die Suchparameter einer Anfrage und alle Elemente des Datasets für das entsprechende Datenfeld.

Abbildung 5.7 stellt das technische Klassendiagramm für die Komponente dar. Sie enthält zwei Klassen `UserDimensionSet` und `BookDimensionSet`. Beide Klassen sind Spezialisierungen der Klasse `DimensionSet` und werden die Ähnlichkeit für die Daten der Benutzer und der Bücher jeweils mithilfe der Methoden `computeUsersimilarity()` und `computeBookSimilartity()` berechnen. Die Klassen werden die definierten Dimensionen durch die Schnittstellen `IUserDimensionSet` und `IBookDimensionSet`, die bei der Klasse `DimensionSet` verwendet werden, erhalten. Um auf die Datenbanken zu zugreifen wird die Schnittstelle `IPersistenz` verwendet.

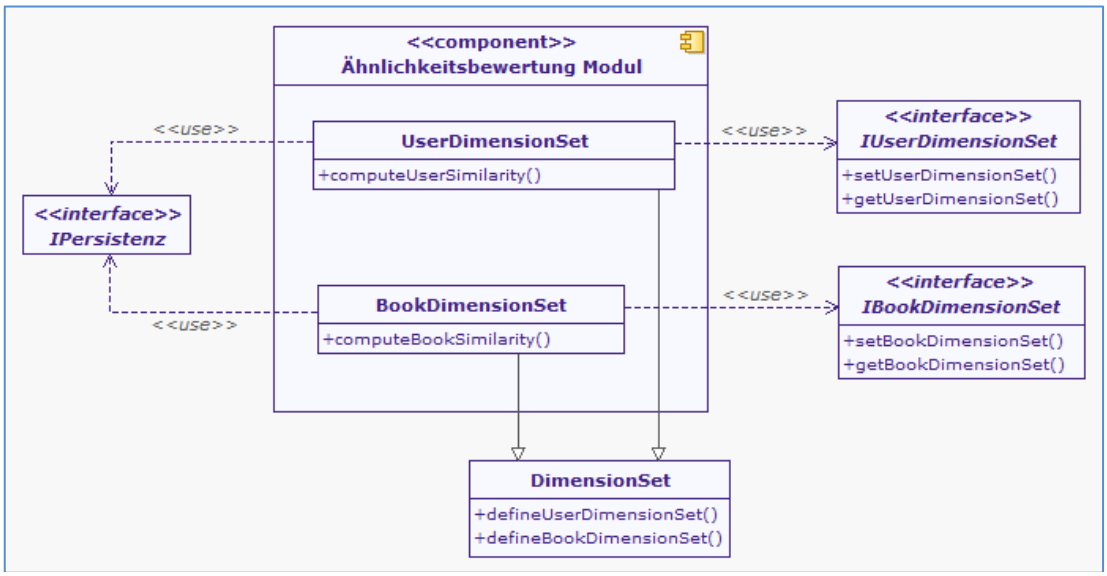

Abbildung 5.7. Technisches Klassendiagramm für Komponente „Ähnlichkeitsbewertungs Modul“

Die Suchparameter können entweder textuell oder numerisch sein. Das Ähnlichkeitsbewertungs Modul benötigt je nach Datentype einen unterschiedlichen Vorgang.

Der numerische Wert der Ähnlichkeit wird auf einer Skala von 0 bis 100, wobei 0 bedeutet dass beide Elemente gleich sind, und 100 bedeutet, dass sie komplett verschieden sind, angegeben.

Das Vorgehen des Ähnlichkeitsbewertungs Moduls, lässt sich in folgender Abbildung veranschaulichen:

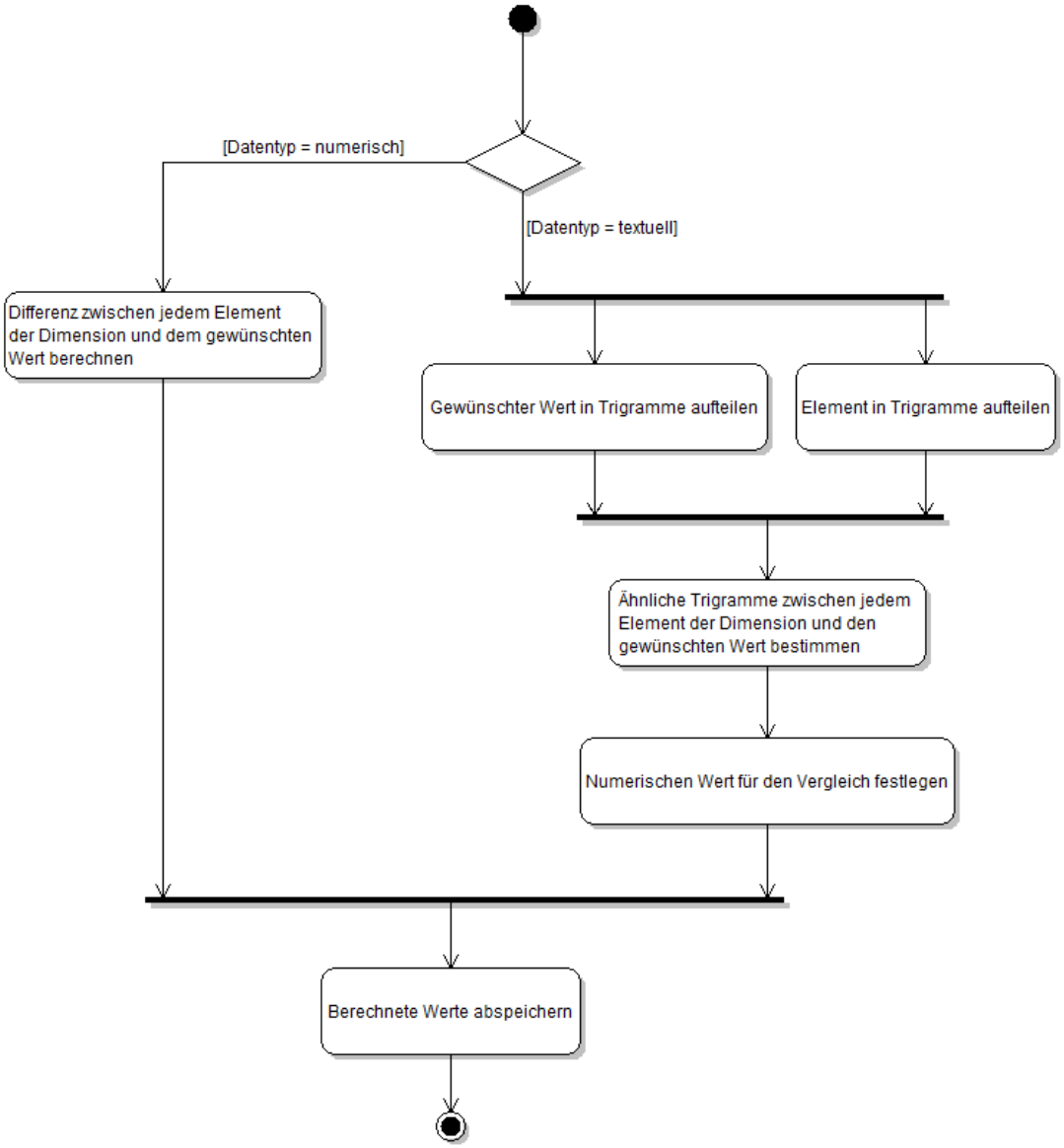

Abbildung 5.8. Aktivitätsdiagramm Ähnlichkeitsbewertungs Modul

5.6.1 Vergleich numerischer Daten

Um die Ähnlichkeit zwischen zwei numerischen Werten zu bestimmen, muss die Differenz zwischen beiden Elementen berechnet werden. Anschließend wird ein Beispiel der Ähnlichkeitsbewertung für Elemente der Dimension „Publikationsjahr“ dargestellt.

Gewünschtes Publikationsjahr	Publikationsjahr	Ähnlichkeit
2003	2001	2
2003	2004	1
2003	2007	4
2003	1998	5

Tabelle 5.1. Bewertung der Ähnlichkeit zweier Publikationsjahre

In Tabelle 5.1 repräsentiert die Spalte „Gewünschtes Publikationsjahr“ den Wert, der vom Benutzer in die Schnittstelle eingegeben wurde. Die Spalte „Publikationsjahr“ beinhaltet die Elemente des Datasets und die Spalte „Ähnlichkeit“ die Differenz.

5.6.2 Vergleich textueller Daten

Für die Dimensionen Titel, Autor und Verlag des Buches sowie für die Dimension Wohnort des Benutzers muss ein Textvergleich realisiert werden, um die Ähnlichkeit der Wörter zu bestimmen. Die folgende Tabelle stellt ein Textvergleich Beispiel für die Dimension „Autor“ dar.

Gesuchter Autor	Autor	Ähnlichkeit
„Charlotte Roche“	„Link Charlotte“	3
„Charlotte Roche“	„Tim Roche“	3
„Charlotte Roche“	„C. Roche“	1

Tabelle 5.2. Bewertung der Ähnlichkeit zwei Autornamen

Der Vergleich zweier Texte bzw. Zeichenkette der Buchdimensionen Titel, Autor und Verlag soll mittels der Definition von Trigrammen für jede Zeichenkette realisiert werden.

Ein Trigramm ist eine Gruppe von drei Buchstaben einer Zeichenkette. Folglich kann eine Zeichenkette in mehrere Trigramme zerlegt werden. Auf diese Weise, generiert die Zeichenkette „Charlotte“ folgende Trigramme: „ C“, „ CH“, „CHA“, „HAR“, „ARL“, „RLO“, „LOT“, „OTT“, „TTE“, „TE “, „E “

Der Vergleich textueller Daten wird in folgenden Schritten realisiert:

- Die Zeichenketten sollen nur aus die Zeichen a,… z, A,…Z, 0…9 und Leerzeichen bestehen. Das heißt, es erfolgt ein Ausschluss anderer Zeichen.
- Jedes Wort der Zeichenkette soll in Trigramme zerlegt werden. Die Zerlegung jedes Wortes ermöglicht den korrekten Vergleich für Texte mit gleichen Wörtern, die nicht in der gleichen Position sind. Z.B. „Charlotte Roche“ und „Link Charlotte“.

Original Zeichenkette	Vorbereitete Zeichenkette	Trigramme
„Charlotte Roche“	CHARLOTTE ROCHE	„ C“, „ CH“, „CHA“, „HAR“, „ARL“, „RLO“, „LOT“, „OTT“, „TTE“, „TE “, „E “, „ R“, „ RO“, „ROC“, „OCH “, „CHE“, „HE “, „E “
„C. Roche“	C ROCHE	„ C“, „ C “, „C “, „ R“, „ROC“, „OCH “, „CHE“, „HE “, „E “

Tabelle 5.3. Erzeugung der Trigramme einer Zeichenkette

Anschließend soll die Anzahl der Trigramme der ersten Zeichenkette, die in der Zweiten sind, und die Anzahl der Trigramme der zweiten Zeichenkette, die in der Erste sind, bestimmt werden.

Dann wird die Ähnlichkeitsrate wie folgendes berechnet:

> A1 = (Anzahl Trigramme von Zeichenkette 1 in Zeichenkette 2)/(Anzahl Trigramme von Zeichenkette 1)
>
> A2 = (Anzahl Trigramme von Zeichenkette 2 in Zeichenkette 1)/(Anzahl Trigramme von Zeichenkette 2)

Die Ähnlichkeit ist der Durchschnitt beider Werte A = (A1 + A2)/2. Schließlich soll die Ähnlichkeit auf einer Skala von 0 bis 100 eingestuft werden (100 – (A * 100)). Wobei 0 bedeutet, dass beide Zeichenketten identisch sind.

Die Daten der Dimension „Location“ bzw. Wohnort des Benutzers des Book-Crossing Datasets repräsentieren die Beschreibung eines Ortes mit drei Unterbereichen. Z.B. „Erfurt, Thueringen, Germany“ oder „San Francisco, California, USA“.

Der Textvergleich zwischen zwei Wohnorten soll die Anzahl der unterschiedlichen Unterbereiche zwischen beiden Werten bestimmen. Somit wird der maximal Wert für die Ähnlichkeit 3, wenn beide Wohnorte komplett unterschiedlich sind und 0, wenn sie gänzlich gleich sind.

Gesuchte Wohnort	Wohnort	Ähnlichkeit
„dresden, sachsen, germany“	„san francisco, california, usa“	3
„dresden, sachsen, germany“	„leipzig, sachsen, germany“	1
„dresden, sachsen, germany“	„erfurt, thueringen, germany“	2
„dresden, sachsen, germany“	„dresden, sachsen, germany“	0

Tabelle 5.4. Bewertung der Ähnlichkeit zwei Wohnorten

5.6.3 Speichern der Werte

Die Ähnlichkeitswerte werden in zwei temporäre Tabellen, die Tabelle „d_books“ für Bücher und die Tabelle „d_users“ für Benutzer, gespeichert. Die temporären Tabellen werden die gleichen Datenfelder der originalen Tabelle, aber mit numerischem Datentyp enthalten.

5.7 Komponente Skyline Query Processing Modul

Dieses Modul ist verantwortlich für die Verarbeitung einer Skyline Query und beinhaltet die Klasse `SkylineQuery`. Als ersten Schritt muss das Modul die Skyline Queries der Suche mithilfe der Methode `getDimensions()` definieren. Dafür benötigt das Modul die Dimensionen der Anfrage. Die Dimensionen sollen durch die Schnittstellen `IUserDimensionSet` und `IBookDimensionSet` geliefert werden

Anschließend kann der Algorithmus für die Verarbeitung einer Skyline Query durch den Abruf der Methode `processSkylineQuery()` durchgeführt werden. Der Algorithmus wird zuerst den Suchbereich definieren und seine Elemente bzw. Datenpunkte des Suchbereiches durch die Methoden `setDataRegion()` und `getSearchRegion()` in einer temporären Tabelle speichern. Dann wird mithilfe der Methoden `getnumberOfPoints()`, `getBoundaryPoint()`, `getPmd()`, `setGSS()` und `computeSkylinePoints()` die Suche der Skyline Punkte anfangen. Schließlich werden die gefundenen Skyline Punkte in einer anderen temporären Tabelle, mithilfe der Methode `saveSkylinePoint()`, gespeichert. Eine präzisere Beschreibung des Algorithmus wird in Abschnitte 5.7.2 und 5.7.3 präsentiert.

Der Zugriff auf die Datenbanken wird durch die Schnittstelle `IPersistenz` realisiert.

Abbildung 5.9 präsentiert das technische Klassendiagramm der Komponente.

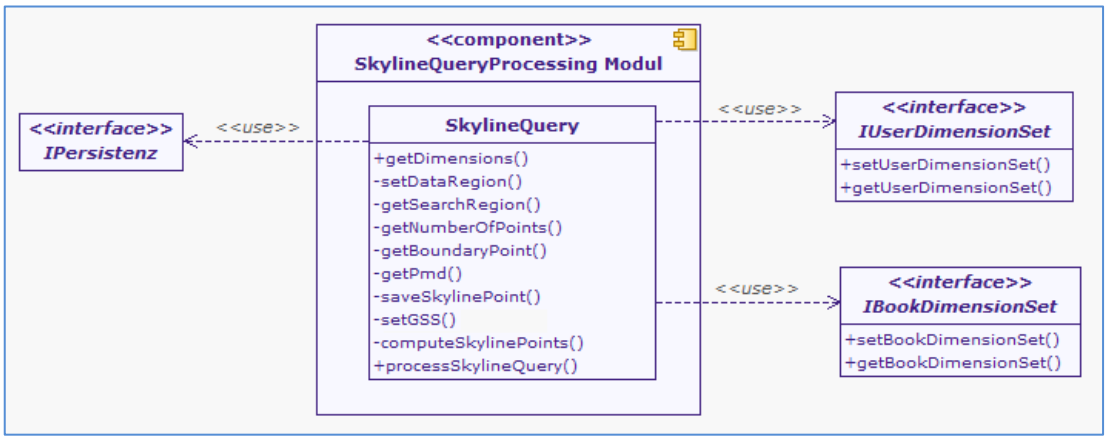

Abbildung 5.9. Technisches Klassendiagramm der „Skyline Query Processing Modul“ Komponente

5.7.1 Definition der Skyline Query

Bevor eine Skyline Query verarbeitet wird, sollen die Dimensionen der Anfrage definiert werden. Ebenso muss man festlegen, welche Art von Wert, entweder Minimum oder Maximum, in jeder Dimension gesucht werden soll. In diesem Abschnitt wird eine Skyline Query für jede Empfehlung („Passende Bücher“, „Bewertete Bücher in der Vergangenheit“ und „Bewertete Bücher von anderen Benutzern“) des Systems (s. 3.1) definiert.

Die Suchparameter, für die der Benutzer in der Benutzer Schnittstelle Werte eingegeben hat, werden die Dimensionen der Skyline Queries sein. Der Skyline Query Processing Modul verwendet als Elemente der Dimensionen die Daten, die von dem Ähnlichkeitsbewertungs Modul in den temporären Tabellen „d_books“ und „d_users“ gespeichert wurden. Auf diese Weise, wird jedes Datenelement der Tabellen durch einen Punkt in dem n-dimensionalen raum repräsentiert. Wie oben beschrieben, je kleiner der Ähnlichkeitswert ist, desto ähnlicher sind die Elemente. Aus diesem Grund werden in alle Dimensionen die Minimum Werte gesucht.

Anschließend wird die Definition der Skyline Queries des Systems beschrieben. Dafür werden die Informationen der Tabelle 5.5 als Referenz für eine Beispielsuche genommen, wobei die Werte für die Felder Benutzer-ID, Buchtitel, Publikationsjahr und Bewertung von dem Benutzer eingegeben wurden und die Werte für Wohnort und Alter von dem System abgerufen wurden.

Benutzer	Benutzer-ID:	85
	Wohnort:	London, England, United Kingdom
	Alter:	41
Buch	Titel:	best poems
	Autor:	
	Publikationsjahr:	2000
	Verlag:	
	Bewertung:	10

Tabelle 5.5. Suchparameterwerte einer Beispielsuche

5.7.1.1. Passende Bücher

Die Skyline Query für diese Anfrage verwendet folgende Dimensionen:

- Titel (Book-Title)
- Autor (Book-Author)
- Publikationsjahr (Year-Of-Publication)
- Verlag (Publisher)
- Durchschnittliche Bewertung (Book-Rating)

Die ersten vier Dimensionen gehören zu Tabelle „BX-Books“. Die fünfte Dimension (Bewertung) ist ein Mittelwert der im Feld „Book-Rating“ in Tabelle „BX-Book-Ratings“ im Book-Crossing Dataset gespeichert ist.

Für den Fall, dass der Benutzer für alle Dimensionen Parameter eingegeben hat und angesichts dessen, dass in allen Dimensionen die Minimum Werte gesucht werden sollen, ist die Definition der Skyline Query wie Folgt:

$$SQ_1 = (\text{‘Min’}, \text{‘Min’}, \text{‘Min’}, \text{‘Min’}, \text{‘Min’})$$

Die Skyline Query für das Beispiel der Tabelle 5.5 verwendet die Dimensionen Buchtitel, Publikationsjahr und durchschnittliche Bewertung. Die Definition der Skyline Query ist:

$$SQ_1 = (\text{‘Min’}, \text{‘Min’}, \text{‘Min’})$$

5.7.1.2. Bewertete Bücher in der Vergangenheit

Die Skyline Query für diese Suche verwendet folgende Dimensionen:

- Titel (Book-Title)
- Autor (Book-Author)
- Publikationsjahr (Year-Of-Publication)
- Verlag (Publisher)

Alle Dimensionen gehören zu Tabelle „BX-Books“ aus dem Book-Crossing Dataset. Außerdem wird das Feld „User-ID“ aus der Tabelle „BX-Users“ als Filter verwendet, da nur die Informationen des aktuellen Benutzers verwendet werden sollen.

Für den Fall, dass der Benutzer für alle Dimensionen Parameter eingegeben hat und angesichts dessen, dass in allen Dimensionen die Minimum Werte gesucht werden sollen, ist die Definition der Skyline Query wie Folgt:

$$SQ_2 = (\text{‘Min’}, \text{‘Min’}, \text{‘Min’}, \text{‘Min’})$$

Die Skyline Query für das Beispiel der Tabelle 5.5 verwendet die Dimensionen Buchtitel und Publikationsjahr. Außerdem wird die Benutzer-ID „85“ als Filter verwendet. Die Definition der Skyline Query ist:

$$SQ_2 = (\text{‘Min’}, \text{‘Min’})$$

5.7.1.3. Bewertete Bücher von anderen Benutzern

Die Skyline Query für diese Suche verwendet folgende Dimensionen:

- Titel (Book-Title)
- Autor (Book-Author)
- Publikationsjahr (Year-Of-Publication)
- Verlag (Publisher)
- Durchschnittliche Bewertung (Book-Rating)
- Wohnort(Location)
- Alter des Benutzers (Age)

Die ersten vier Dimensionen gehören zu der Tabelle „BX-Books“ aus dem Book-Crossing Dataset und die Fünfte ist ein Mittelwert aus dem Feld „Book-Rating“ in der Tabelle „BX-Book-Ratings“. Die sechste und siebte Dimension gehören zu der Tabelle „BX-Users“.

Für den Fall, dass der Benutzer für alle Dimensionen Parameter eingegeben hat und angesichts dessen, dass in allen Dimensionen die Minimum Werte gesucht werden sollen, ist die Definition der Skyline Query wie Folgt:

$$SQ_3 = (\text{‘Min’}, \text{‘Min’}, \text{‘Min’}, \text{‘Min’}, \text{‘Min’}, \text{‘Min’}, \text{‘Min’})$$

Die Skyline Query für das Beispiel der Tabelle 5.5 verwendet die Dimensionen Buchtitel, Publikationsjahr, durchschnittliche Bewertung sowie Wohnort und Alter des Benutzers „85“. Die Definition der Skyline Query ist:

$$SQ_3 = (\text{‘Min’}, \text{‘Min’}, \text{‘Min’}, \text{‘Min’}, \text{‘Min’})$$

5.7.2 Skyline Query Verarbeitung

Für die Verarbeitung der Skyline Query wurden die Nearest Neighbor (NN) und Skyframe Algorithmen, die in Kapitel 2 beschrieben wurden, berücksichtigt.

Nachfolgend wird der Vorgang zur Verarbeitung der Skyline Query beschreiben. Als Beispiel wird die Skyline Query für „Passende Bücher“ SQ_1 = (‘Min’, ‘Min’, ‘Min’) genommen. SQ_1 verwendet die Dimensionen „Titel“, „Publikationsjahr“ und „Durchschnittliche Bewertung“.

5.7.2.1. Suchbereich

Bevor die Skyline Query verarbeitet wird, soll der Suchbereich bestimmt werden. Der Suchbereich umfasst alle verfügbaren Datensätze für die Dimensionen der Anfrage.

5.7.2.2. Grenzpunkte

Ein Grenzpunkt repräsentiert den bestmöglichen Wert aller Dimensionen. Ein Skyline Punkt soll möglichst nah am Grenzpunkt liegen.

Gemäß den bisherigen Definitionen sind die Werte für die Dimensionen der Skyline Queries der Anwendung positive numerische Werte. Daher ist Der Minimum Wert einer Dimension gleich Null. Auf diese Weise wäre der Grenzpunkt für SQ_1 = (‘Min’, ‘Min’, ‘Min’) den Punkt (0, 0, 0).

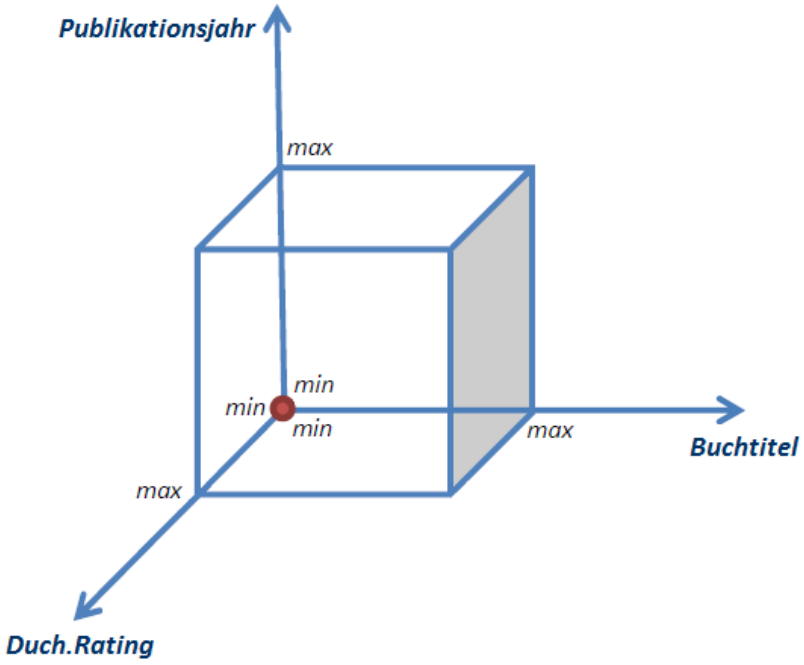

Abbildung 5.10. Datenraum und Grenzpunkt für SQ_1 = (‘Min’, ‘Min’, ‘Min’)

Die Abbildung stellt den drei-dimensionalen Raum für SQ_1 = (‘Min’, ‘Min’, ‘Min’) dar, wobei der Würfel die ganzen Datensätze bzw. den Suchbereich repräsentiert. Der Grenzpunkt der Skyline Query (0, 0, 0) wird durch den roten Punkt repräsentiert.

5.7.2.3. Dominantester Punkt

In diesem Schritt soll der dominanteste Punkt des ganzen Datenraumes bezüglich des Grenzpunkts gesucht werden. Der dominanteste Punkt ist nicht der der alle Punkte dominiert, sondern der, der die größte dominierende Region bestimmt. Eine dominierende Region umfasst die Region in der sich die meisten Skyline Punkte befinden.

Um den dominantesten Punkt zu finden, wird der euklidische Abstand von jedem Punkt zum Grenzpunkt berechnet. Der Punkt der dem Grenzpunkt am nächsten liegt ist der dominanteste Punkt der Skyline Query und wird automatisch ein Skyline Punkt sein.

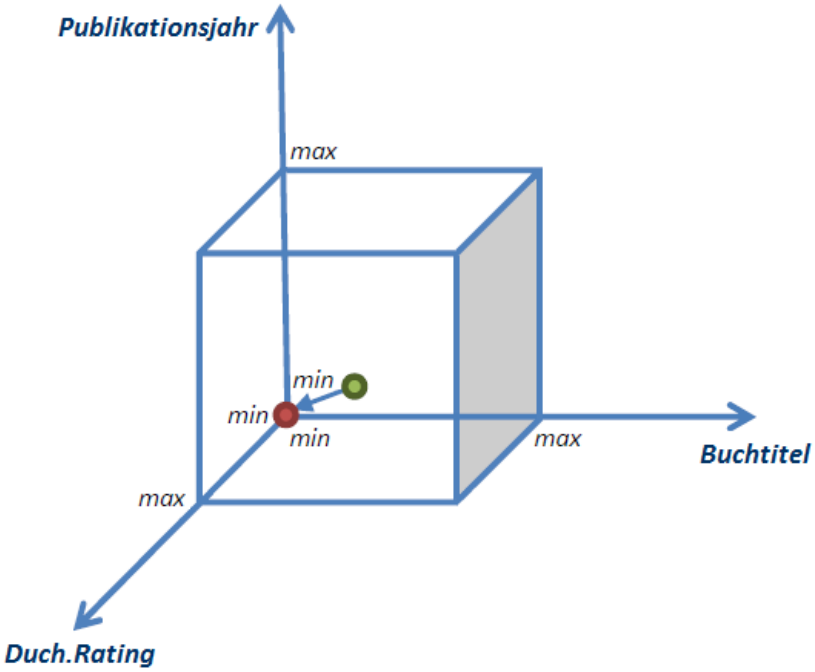

Abbildung 5.11. Definition des dominantesten Punktes für SQ_1 = (‘Min’, ‘Min’, ‘Min’)

Die Abbildung stellt die Definition des dominantesten Punktes für SQ_1 dar. Der rote Punkt repräsentiert den Grenzpunkt (0, 0, 0). Der grüne Punkt repräsentiert den Punkt der dem Grenzpunkt am Nächsten liegt, bzw. den dominantesten Punkt. Der Pfeil repräsentiert den euklidischen Abstand zwischen beiden Punkten.

5.7.2.4. Greedy Suchbereich

Der Greedy Suchbereich ist eine dominierende Region und wird vom dominantesten Punkt bestimmt. Der Greedy Suchbereich umfasst alle Punkte, die nicht vom dominantesten Punkt dominiert sind [WANG, 2009]. Das heißt, dass die dominierten Punkte aus dem Greedy Suchbereich ausgeschlossen werden. Aus diesem Grund ermöglicht die Definition des Greedy Suchbereiches die Verringerung des Anfangssuchbereiches. Somit soll die Suche der Skyline Punkte optimiert werden.

Die folgende Abbildung stellt den Greedy Suchbereich für die Dimension „Publikationsjahr“ für SQ_1 dar, wobei der grüne Punkt den Wert in der Dimension „Publikationsjahr“ des dominantesten Punkts repräsentiert und das blaue Feld den Greedy Suchbereich ist.

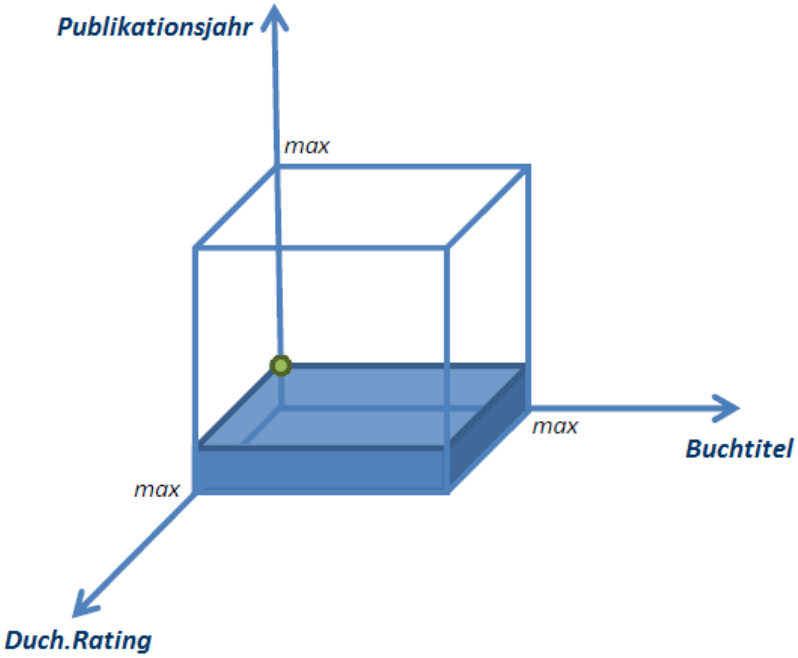

Abbildung 5.12. Greedy Suchbereich für Dimension „Publikationsjahr“ in SQ_1 = (‘Min’, ‘Min’, ‘Min’)

Der ganze Greedy Suchbereich für SQ_1 umfasst die Greedy Suchbereiche bezüglich der drei Dimensionen.

5.7.2.5. Skyline Punkte

In diesem Schritt müssen die Skyline Punkte gesucht werden. Die Skyline Punkte sind solche, die vom dominantesten Punkt nicht dominiert sind. Ein Punkt dominiert einen anderen Punkt, wenn er nicht schlechter als der andere in allen Dimensionen und in mindestens einer Dimension besser als der andere ist [WANG, 2009] (s.2.1). Alle Skyline Punkte befinden sich im Greedy Suchbereich. Allerdings sind nicht alle Punkte des Greedy Suchbereiches Skyline Punkte, deswegen wird es einige geben, die von anderen Punkten dominiert sind.

Um die Skyline Punkte zu finden, wird der Greedy Suchbereich geteilt und ein rekursiver Vorgang angewendet, der den Greedy Suchbereich in weitere Bereiche aufteilt, bis alle Skyline Punkte gefunden sind.

5.7.3 Algorithmen

Im Folgenden werden die Algorithmen für die Verarbeitung der Skyline Query in Pseudocode dargestellt.

5.7.3.1. getDimensions

Die Methode „getDimensions“ ist verantwortlich für die Definition der Dimensionen eine Skyline Query.

Algorithmus 1. getDimensions
Input: queryType QT, dimensionSet bookDimensionSet, dimensionSet userDimensionSet
Output: dimensionSet DS, filter F
Zweck: Dimensionen der Skyline Query bestimmen

```
        // Wenn es um die Anfrage „Passende Bücher“ geht
1:      if (QT = „Passende Bücher“) then
2:              for each dimension in bookDimensionSet do
3:                      DS.add(dimension)
4:              end for

        // Wenn es um die Anfrage „Bewertete Bücher in der Vergangenheit“ geht
5:      elseif (QT = „Bewertete Bücher in der Vergangenheit“) then
6:              for each dimension in bookDimensionSet do
7:                      DS.add(dimension)
8:              end for
                // Filter bestimmen
9:              F =“User-ID = currentUser(User-ID)“

        // Wenn es um die Anfrage „Bewertete Bücher von anderen Benutzern“ geht
10:     elseif (QT = „Bewertete Bücher von anderen Benutzern“) then
11:             for each dimension in userDimensionSet do
12:                     DS.add(dimension)
13:             end for
14:             for each dimension in bookDimensionSet do
15:                     DS.add(dimension)
16:             end for
                // Filter bestimmen
17:             F =“User-ID <> currentUser[User-ID]“
18:     end if
19:     return DS, F
```

5.7.3.2. setDataRegion

Die Methode „setDataRegion“ ist zuständig für die Definition des ganzen Suchbereiches einer Skyline Query. Dazu erstellt sie eine temporäre Tabelle, in der alle Datensätze des Suchbereiches gespeichert werden sollen. Darüber hinaus soll die Methode die temporäre Tabelle, in der die Skyline Punkte gespeichert werden sollen, erstellen. Es sei darauf hingewiesen, dass eine Suchbereich Tabelle und eine Skyline Punkte Tabelle für jede Anfrage des Systems (s. 3.1) erstellt werden soll.

Algorithmus 2. setDataRegion
Input: queryType QT, dimensionSet DS
Output: table temp_SkylineSearchRegion, table temp_Skyline
Zweck: Suchbereich der Skyline Query bestimmen

```
        // Temporäre Tabelle für Elemente des Suchbereiches von QT erstellen
1:      set SQL-Create Table–Clause for table temp_SkylineSearchRegion of QT

        // Datensätze des Suchbereiches selektieren
2:      for each dimension in DS do
3:              add dimension to SQL-Select-Clause
4:      end for

        //Wenn es um die Anfrage „Passende Bücher“ geht
5:      if (QT = „Passende Bücher“) then
6               add table d_books to SQL-From-Clause

        // Wenn es um die Anfrage „Bewertete Bücher in der Vergangenheit“ geht
7:      elseif (QT = „Bewertete Bücher in der Vergangenheit“) then
8:              add tables d_books and bx-book-ratings to SQL-From-Clause
9:              add filter to SQL-Where-Clause

        // Wenn es um die Anfrage „Bewertete Bücher von anderen Benutzern“ geht
10:     elseif (QT = „Bewertete Bücher von anderen Benutzern“) then
11:             add tables d_books, d_users and bx-book-ratings to SQL-From-Clause
12:             add filter to SQL-Where-Clause
13:     end if

        /* Temporäre Tabelle für Skyline Punkte von QT mit der gleichen Struktur der
         Tabelle temp_SkylineSearchRegion erstellen */
14:     set SQL-CreateTable–Clause for table temp_Skyline of QT like
        temp_SkylineSearchRegion of QT
```

```
        //SQL Klauseln zusammenfügen und durchführen
15:     concat(SQL-CreateTable-Clause for temp_SkylineSearchRegion, SQL-Select-Clause,
        SQL-From-Clause, SQL-Where-Clause)
16:     execute SQL-Query
17:     execute SQL-CreateTable-Clause for temp_Skyline
```

5.7.3.3. getSearchRegion

Die Methode „getSearchRegion“ hat das Ziel, die Elemente bzw. Punkte eines Suchbereiches zu selektieren. Es wird zwei Fälle geben. Im ersten Fall geht es um den gesamten Suchbereich. Hier werden alle Datensätze selektiert. Der zweite Fall ist eine Unterteilung eines Bereiches. In diesen Fall werden die Werte einer der Dimensionen begrenzt.

Algorithmus 3. getSearchRegion
Input: queryType QT, dimensionSet DS, dimensionName Dim, minimumPoint MinP, maximumPoint MaxP
Output: searchRegion SR
Zweck: Suchbereich bestimmen, entweder für den ganzen Datenbereich oder eine Untermenge davon.

```
        // Wenn es um die ganze Region geht
1:      if (Dim is null) then
                //SQL-Query bauen
2:              select all fields of table temp_SkylineSearchRegion of QT

        // Wenn es um eine Subregion geht
3:      else
                //SQL-Query bauen
4:              select all fields of table temp_SkylineSearchRegion of QT
                // SQL-Where-Clause bauen
5:              for each dimension in DS do
6:                      if (dimension = Dim) then
7:                              SQL-Where-Clause = SQL-Where-Clause +
                                “(point[dimension] between MinP[dimension] and
                                MaxP[dimension]) and”
8:                      else
9:                              SQL-Where-Clausel = SQL-Where-Clausel +
                                “(point[dimension] >= 0 and”
10:                     end if
```

```
11:             end for
12:     end if

        //SQL Klauseln zusammenfügen
13:     SR = concat(SQL-Select-Clause, SQL-From-Clause, SQL-Where-Clause)
14:     return SR
```

5.7.3.4. getBoundaryPoint

Die Methode „getBoundaryPoint“ bestimmt den Grenzpunkt des Suchbereiches. Der Grenzpunkt wird den Wert „0“ für alle Dimensionen bzw. (0, 0, …, 0) erhalten (die Anzahl der Werte „0“ der Notation (0, 0, …, 0) hängt von der Anzahl der Dimensionen ab).

Algorithmus 4. getBoundaryPoint
Input: dimensionSet DS
Output: boundaryPoint BP
Zweck: Grenzpunkt des Suchbereiches (0,0,..0) bestimmen.

```
// Boundary point initialisieren. BP = (0,0,..,0)
1:      for each dimension in dimensionSet do
2:              BP.add(0)
3:      end for

4:      return BP
```

5.7.3.5. getPmd

Die Methode „getPmd“ berechnet den dominantesten Punkt eines Suchbereiches. Der dominanteste Punkt wird der Punkt, der sich am Nächsten zum Grenzpunkt befindet. Dafür wird eine SQL-Anfrage erstellt deren Berechnung die euklidische Distanz in die SQL-OrderBy-Klausel einbezieht.

Algorithmus 5. getPmd
Input: dimensionSet DS, searchRegion SR, boundaryPoint BP
Output: mostDominatingPoint PMD
Zweck: Dominantester Punkt in Bezug auf den Grenzpunkt (boundary point) mittels der Berechnung der euklidischen Distanz bestimmen.

```
        //SQL-Query bauen
```

```
1:    set “select all fields from SR” in SQL-Select-Clause and SQL-From-Clause

      //Euklidische Distanz Berechnungsformel in SQL-OrderBy-Clause hinzufügen
2:    SQL-OrderBy-Clause = SQL-OrderBy-Clause  +
                  “(point[dimension] –BP[dimension])^2 +”
3:    add square root calculation to SQL-OrderBy-Clause

      //SQL Klauseln zusammenfügen und nur die erste Zeile selektieren
4:    concat(SQL-Select-Clause, SQL-From-Clause, SQL-OrderBy-Clause, limit 1)
5:    execute SQL-Query
6:    if SQL-Query Result is not null then
7:          PMD = SQL-Query Result
8:    end if

9:    return PMD
```

5.7.3.6. saveSkylinePoint

Die Methode „saveSkylinePoint“ ist zuständig für die Speicherung eines Skyline Punktes in der temporären Tabelle.

Algorithmus 6. saveSkylinePoint
Input: queryType QT, dimensionSet DS, mostDominatingPoint PMD
Output: table temp_Skyline
Zweck: Skyline Punkte in temporärer Tabelle speichern.

```
      //SQL-Query bauen
1:    set SQL-InsertInto-Clause for table temp_Skyline of QT
2:    set “select all fields from temp_SkylineSearchRegion of QT”  in  SQL-Select-
      Clause and SQL-From-Clause

      // Identische Punkte mit dem dominantesten Punkt selektieren
3:    for each dimension in DS do
4:          SQL-Where-Clause = SQL-Where-Clause +
            “(point[dimension] = PMD[dimension] and”
5:    end for

      //SQL Klauseln zusammenfügen
6:    concat(SQL-Select-Clause, SQL-From-Clause, SQL-Where-Clause)
7:    execute SQL-Query
```

5.7.3.7. setGSS

Die Methode „setGSS“ hat das Ziel, den Greedy Suchbereich zu bestimmen. Alle Punkte, die vom dominantesten Punkt (PMD) dominiert sind, sollen aus dem Suchbereich entfernt werden. Da der dominanteste Punkt ein Skyline Punkt ist und mithilfe der Methode „saveSkylinePoint“ gespeichert werden soll, kann er auch aus dem Greedy Suchbereich entfernt werden.

Algorithmus 7. setGSS
Input: queryType QT, dimensionSet DS, mostDominatingPoint PMD
Output: table temp_SkylineSearchRegion
Zweck: Suchbereich beschneiden in Bezug der dominanteste Punkt (PMD)

```
        //SQL-Query bauen
1:      set “delete from temp_SkylineSearchRegion of QT”      in SQL-Delete-Clause
        /* Identische Punkte mit dem dominantesten Punkt und dominierte Punkte
        selektieren */
2:      for each dimension in DS do
3:              SQL-Where-Clause = SQL-Where-Clause +
                “(point[dimension] >= PMD[dimension] and”
4:      end for

        //SQL Klauseln zusammenfügen
5:      concat(SQL-Delete-Clause, SQL-Where-Clause)
6:      execute SQL-Query
```

5.7.3.8. computeSkylinePoints

Die Methode „computeSkylinePoints“ hat das Ziel, die Skyline Punkte eines Bereiches zu finden. Aus dem Grenzpunkt (BP) wird der dominanteste Punkt (PMD) berechnet. Dann wird der Greedy Suchbereich als Suchbereich der Anfrage festgelegt. Und anschließend wird überprüft, ob es neben dem dominantesten Punkt, im Greedy Suchbereich, noch weitere dominante Punkte gibt. Falls dem so ist soll der Greedy Suchbereich weiter geteilt werden und rekursiv nach weiteren Skyline Punkten innerhalb der Subbereiche suchen.

Algorithmus 8. computeSkylinePoints
Input: queryType QT, dimensionSet DS, searchRegion SR, boundaryPoint BP
Zweck: Die Skyline Punkte eines Bereiches bestimmen.

```
        // Dominantester Punkt berechnen
1:      PMD = getPmd(DS, SR, BP)
```

```
        // Dominantesten Punkt als Skyline Punkt speichern
2:      saveSkylinePoint(QT, DS, PMD)
        // Greedy Suchbereich bestimmen
3:      SR = setGSS(QT, DS, PMD)
4:      if number of points in SR > 0 then
                // Suchbereich aufteilen
5:              for each dimension in DS do
                        //Untermenge des Suchbereiches bestimmen
6:                      SubSR = getSearchRegion(QT, DS, dimension, BP, PMD)
7:                      if number of points in SubSR > 0 then
8:                              computeSkylinePoints(QT, DS, SubSR, BP)
9:                      end if
10:             end for
11:     end if
```

5.7.3.9. processSkylineQuery

Die Methode „processSkylineQuery“ führt die Verarbeitung einer Skyline Query durch. Zunächst soll der Suchbereich innerhalb aller verfügbaren Datensätze bestimmt werden. Anschließend wird der Grenzpunkt definiert, welcher den Wert (0,0,…,0) erhalten wird. Schließlich werden die Skyline Punkte gesucht.

Algorithmus 9. processSkylineQuery
Input: queryType QT, dimensionSet DS
Zweck: Verarbeitung der Skyline Query durchführen.

```
        //Ganzen Suchbereich bestimmen
1:      setDataRegion(QT, DS)
        //Suchbereich initialisieren
2:      SR = getSearchRegion(QT, DS, “”, null, null)
        //Grenzpunkt bzw. boundary point initialisieren
3:      if number of points in SR > 0 then
4:              BP = getBoundaryPoint(DS)
                // Skyline Punkte suchen
5:              computeSkylinePoints(QT, DS, SR, BP)
```

6 Realisierung

In diesem Kapitel wird die Implementierung der verschiedenen Komponenten des Systems gemäß dem in Kapitel 5 realisierten Entwurf beschrieben. Zuerst wird der Aufbau der Datenbank beschrieben. Danach die Implementierung der Anwendung präsentiert und schließlich die Validierung der Anwendung realisiert.

6.1 Datenbank

Wie bereits erwähnt entstammt die Datenbasis dem Book-Crossing Dataset. Die Daten sind als MySQL-Dump Dateien verfügbar.

Um eine optimale Leistung des Systems zu erhalten, wurden in der Datenbank zusätzliche temporäre Tabellen erstellt. Zudem wurde die Tabelle “BX-Books“ um vier neue Datenfelder ergänzt.

6.1.1 Migration

Eine MySQL Datenbank unter dem Namen „BookCrossing“ wurde erstellt. Und die MySQL-Dump Dateien wurden dorthin migriert.

Die Dump Dateien enthalten SQL-Anweisungen zur Erstellung und zum Ausfüllen der Tabellen „BX-Books“, „BX-Users“ und „BX-Book-Ratings“. Nach der Migration wurden 271379 Datensätze für die Tabelle „BX-Books“ hinzugefügt. Für die Tabelle „BX-Users“ 278858 Datensätze und für die Tabelle „BX-Book-Ratings“ 1149780 Datensätze.

6.1.2 Temporäre Tabellen

Die Tabellen „d_books“ und „d_users“ wurden erstellt mit dem Ziel, dass der Ähnlichkeitsbewertungs Modul dahin die berechneten Daten für eine Anfrage speichern kann. Die Daten werden numerisch sein und werden die Elemente der Dimensionen für die Skyline Queries sein. Danach kann der Skyline Query Proccesing Modul diese Daten anwenden, um die Skyline Punkte zu finden.

Zusätzlich werden während der Laufzeit der Anwendung die Tabellen „temp_SkylineSearchRegion[n]“ und „temp_Skyline[n]“ für jede Empfehlung des Systems (s. 3.1) erstellt. Wobei „n“ ein numerischer Wert ist, der eine der drei Anfragen repräsentiert. Die Erstellung diese Tabellen ermöglicht einem einzigen Zugriff auf die originalen Tabellen, sowie die Ausführung von Joins und Anwendung von Filtern.

6.1.3 Berechnete Datenfelder

Um den Processing-Overhead zu reduzieren werden einige Offline-Berechnungen realisiert. Solche Berechnungen werden bei der Aktualisierung der Datensätzen durch die Ausführung von SQL-Trigger realisiert. Die Ergebnisse werden in neue Datenfelder der Tabelle „BX-Books“ gespeichert.

6.1.3.1. Durchschnittliche Bewertung

Das System verwendet die durchschnittliche Bewertung bzw. Rating der Bücher als Suchparameter des Systems. Die durchschnittliche Bewertung muss aus der Tabelle „BX-Book-Ratings“ für jeden Datensatz der Tabelle „BX-Books“ berechnet werden. Unter Berücksichtigung der großen Anzahl an Datensätze in den Tabellen „BX-Books“ (271379 Datensätze) und „BX-Book-Ratings“ (1149780 Datensätze), und angesichts dessen, dass die Daten beider Tabellen nicht verändern werden (s.4.5.1), wurde festgestellt, dass die durchschnittliche Bewertung durch eine Offline-Berechnung erhalten wird und sie in einem neuen Datenfeld, in der Tabelle „BX-Books“, gespeichert werden soll.

Das neue Datenfeld wurde „avgRating“ genannt und wird berechnet und gespeichert, wenn ein Datensatz der Tabelle „BX-Book-Ratings“ aktualisiert wurde. Dafür wurden eine SQL-Funktion „f_avgRating“ und der SQL-Trigger t_ComputeAvgRating, die in den folgenden Abbildungen präsentiert werden, für die Tabelle „BX-Book-Ratings“ erstellt.

```
DELIMITER $$
CREATE FUNCTION f_avgRating(book char(13)) RETURNS decimal(3,1)
BEGIN
	DECLARE avgRating decimal(3,1);

	select sum(`Book-Rating`)/count(`Book-Rating`) into avgRating
	from `bx-book-ratings`
	where `book-rating` <> 0 and isbn = book;

	RETURN avgRating;

END$$
DELIMITER ;
```

Abbildung 6.1. Skript der SQL-Funktion „f_avgRating“

```
DELIMITER $$
CREATE TRIGGER t_ComputeAvgRating
AFTER UPDATE ON `bx-book-ratings`
	FOR EACH ROW
	BEGIN

		UPDATE `bx-books`
		SET avgRating = ifnull(f_avgRating(isbn),0)
		WHERE isbn = NEW.isbn;

	END;
$$
DELIMITER ;
```

Abbildung 6.2. Skript des SQL-Triggers t_ComputeAvgRating

6.1.3.2. Vergleichbare textuelle Daten

Das Ähnlichkeitsbewertungs Modul realisiert den Vergleich textueller Daten mittels des Trigramm-Algorithmus [Adams, 1993]. Voraussetzung für den Algorithmus ist, dass die Daten bzw. Zeichenketten nur bestimmte Zeichen beinhalten. Konform sind Buchstaben von A bis Z, Nummern von 0 bis 9 und Leerzeichen. Aufgrund der großen Anzahl an Datensätzen in der Tabelle „BX-Books“ (271379 Datensätze), und angesichts dessen, dass die Daten der Tabellen nicht verändern werden (s.4.5.1), wurde festgestellt, dass Offline-Berechnungen in der folgenden Datenfelder der Tabelle „BX-Books“ realisiert werden soll.

Die Datenfelder, die textuelle Daten beinhalten, sind „Book-Title“, „Book-Author“ und „Publisher“, folglich wurden drei neue Datenfelder „compBookTitle“, „compBookAuthor“ und „compPublisher“ erstellt. Die neuen Datenfelder werden berechnet und gespeichert, wenn das Datenfeld „Book-Title“, „Book-Author“ oder „Publisher“ eines Datensatzes der Tabelle „BX-Books“ aktualisiert wird. Dafür wurden die SQL-Funktion „f_comparableString“ und der SQL-Trigger „t_ComputeComparableString“ für die Tabelle „BX-Books“ erstellt.

Die SQL-Funktion „f_comparableString“ wandelt eine Zeichenkette in eine andere großgeschriebene Zeichenkette um, die nur Buchstaben von A bis Z, Nummern von 0 bis 9 und Leerzeichen beinhaltet. Dafür muss die SQL-Funktion jedes Zeichen der Zeichenkette lesen und folgende Aktionen realisieren:

- Die Zeichen á, à, â, ä, ã, å, Á, À, Â, Ä, Ã und Å sollen durch den Buchstaben „A“ ersetzt werden.
- Die Zeichen é, è, ê, ë, É, È, Ê und Ë sollen durch den Buchstaben „E“ ersetzt werden.
- Die Zeichen í, ì, î, ï, Í, Ì, Î und Ï sollen durch den Buchstaben „I“ ersetzt werden.
- Die Zeichen ó, ò, ô, ö, õ, Ó, Ò, Ô, Ö, Õ sollen durch den Buchstaben „O“ ersetzt werden.
- Die Zeichen ú, ù, û, ü, Ú, Ù, Û und Ü sollen durch den Buchstaben „U“ ersetzt werden.
- Die Zeichen ñ und Ñ sollen durch den Buchstaben „N“ ersetzt werden.
- Die Zeichen ç und Ç sollen durch den Buchstaben „C“ ersetzt werden.
- Die Zeichen ÿ und Ÿ sollen durch den Buchstaben „Y“ ersetzt werden.
- Das Zeichen ß soll durch die Buchstaben „SS“ ersetzt werden.
- Die Zeichen -, _, ;, :, , ‘, ?, !, -, +, /, \, (,), [,], {, }, @, &, #, %, $, =, *, <, > sowie Leerzeichen, Punkt und Komma werden als Trennzeichen betrachtet und sollen durch ein Leerzeichen ersetzt werden.
- Wenn das Zeichen kein a..z, A..Z, 0..9 oder Leerzeichen ist, soll es gelöscht werden.

Anderseits soll es zwei Leerzeichen zwischen den Wörtern, sowie am Anfang und am Ende der Zeichenkette, geben. Das Ziel ist, die gleichen Trigramme zu erhalten, obwohl die Reihenfolge der Wörter der Zeichenkette anderes ist.

Zeichenkette	Trigramme
__CHARLOTTE__ROCHE__	„ C“, „ CH“, „CHA“, „HAR“, „ARL“, „RLO“, „LOT“, „OTT“, „TTE“, „TE “, „E “, „ R“, „ RO“, „ROC“, „OCH “, „CHE“, „HE “, „E “
__ROCHE__CHARLOTTE__	„ R“, „ RO“, „ROC“, „OCH “, „CHE“, „HE “, „E “,„ C“, „ CH“, „CHA“, „HAR“, „ARL“, „RLO“, „LOT“, „OTT“, „TTE“, „TE “, „E “

Tabelle 6.1. Trigramme

Tabelle 6.1 stellt die Trigramme für „CHARLOTTE ROCHE“ und „ROCHE CHARLOTTE“ dar. Wie zu erkennen ist ermöglichen die zwei Leerzeichen in der Mitte, am Anfang und am Ende, dass beide Zeichenketten die gleichen Trigramme generieren.

Der Trigger, der die Durchführung der SQL-Funktion „f_comparableString“ auslöst, wird in Abbildung 6.3 präsentiert.

```
DELIMITER $$
CREATE TRIGGER t_ ComputeComparableString
BEFORE UPDATE ON `bx-books`
        FOR EACH ROW
          BEGIN
            IF NOT (old.`Book-Title` = new.`Book-Title`)
            THEN
                SET new.compBookTitle = f_comparableString(new.`Book-Title`);

              ELSEIF NOT old.`Book-Author` = new.`Book-Author`
              THEN
                SET new.compBookAuthor = f_comparableString(new.`Book-Author`);

              ELSEIF NOT old.Publisher = new.Publisher
              THEN
                SET new.compPublisher = f_comparableString(new.Publisher);
              END IF;
        END
$$
DELIMITER ;
```

Abbildung 6.3. Skript des SQL-Triggers t_ComputeComparableString

6.2 Implementierung der Anwendung

Die Anwendung wurde in die Java Entwicklungsumgebung Eclipse 3.6.1 implementiert und die Datenbank wird von DBMS MySQL 5.5 gesteuert.

6.2.1 Benutzerschnittstelle

Das Empfehlungssystem benötigt bestimmte Informationen, um eine Suche zu realisieren. Die Informationen können in der Schnittstelle der Anwendung für den Benutzer eingegeben werden. Dies ist in der folgenden Abbildung präsentiert.

Abbildung 6.4. Schnittstelle der Anwendung

Die Schnittstelle der Anwendung präsentiert zwei Bereiche. Im linken Bereich kann der Benutzer sich in das System einloggen, sofern seine Informationen in der Datenbank gespeichert sind. Durch das Einloggen kann das System die Informationen des aktuellen Benutzers abrufen und somit alle drei Empfehlungen berechnen. Die verfügbaren Benutzer Informationen im „Book-Crossing“ Dataset sind „User-ID“, „Age“ und „Location“.

Der rechte Bereich präsentiert verschiedene Eigenschaften des Buches wie Titel, Autor, Publikationsjahr und Verlag. Diese Eigenschaften entsprechen den Attributen des Buches im „Book-Crossing“ Dataset, einschließlich der Bewertungen. Da ein Buch mehrere Bewertungen von verschiedenen Benutzern haben kann, wird ein durchschnittlicher Wert aller Bewertungen verwendet.

Sowohl die Informationen des aktuellen Benutzers als auch die gewünschten Eigenschaften des Buches und der Bewertung werden Parameter der Suche sein und werden später die Skyline Queries definieren.

Abbildung 6.1 stellt eine beispielhafte Suche dar. Der Benutzer „85“ sucht nach Büchern mit ähnlichen Titeln wie „best poems“, die ungefähr im Jahre „2000“ veröffentlicht wurden und möglichst die maximale Bewertung „10“ haben.

Die Benutzerschnittstelle beinhaltet die Klasse `BookRecommender_GUI`, die verantwortlich für die grafische Darstellung der Anwendung ist. Darüber hinaus implementiert die Klasse die Schnittstellen `I_UserIdentificator` und `I_SearchParameter`, um mit weiteren Komponenten zu kommunizieren.

```
public class BookRecommender_GUI extends JFrame implements
ActionListener, I_UserIdentificator, I_SearchParameter{
```

Abbildung 6.5. Deklaration der Klasse `BookRecommender_GUI`

Die Implementierung von `ActionListener` ermöglicht die Ausführung der Optionen Login, Logout und Suchen.

Um das Einloggen eines Benutzers zu realisieren, sollen zuerst die für den Benutzer eingegebenen Daten überprüft werden. Anschließend soll der Benutzer in der Datenbank mittels der Klasse `CurrentUser` gesucht werden. Wenn er gefunden ist, ruft das System seine Informationen ab und bezeichnet ihn als aktuellen Benutzer bzw. „Current User“.

```
if (command=="Login"){
      if (isUserIdValid()){
            CurrentUser objCurrentUser = new CurrentUser();
            objCurrentUser.userId = getUserId();
            if (objCurrentUser.getUserRecord() == null){
                  JOptionPane.showMessageDialog(null,
                  "Benutzer-ID existiert nicht");
                  jtUserId.setText("");
            }
            else{
                  objCurrentUser.setCurrentUser();
                  CurrentUser = new String[3];
                  CurrentUser[0]= objCurrentUser.userId;
                  CurrentUser[1]= objCurrentUser.location;
                  CurrentUser[2]= objCurrentUser.age;
                  showLogin();
            }
       }
}
```

Abbildung 6.6. Option Login

Die Option Logout entfernt die Bezeichnung „Current User“.

```
if (command=="Logout"){
      CurrentUser = new String[3];
      showLogout();
```

Abbildung 6.7. Option Logout

Die Option Suche überprüft zuerst, ob die für den Benutzer eingegebenen Suchparameter richtig sind, anschließend wird die Klasse `DimensionsSet` aufgerufen, um die Dimensionen der Anfrage zu definieren und die Ähnlichkeitsbewertung zu realisieren. Schließlich wird die Klasse `SkylineQuery` aufgerufen, um die Verarbeitung der Anfragen, durch Angabe des Anfragetyps und zwar Nummer „1“ für die Anfrage „Passende Bücher“, Nummer „2“ für „Bewertete Bücher in der Vergangenheit“ und Nummer „3“ für „Bewertete Bücher von anderen Benutzern“, zu realisieren.

```
if (command=="Suchen")
     if (isSearchParameterValid()){
           DimensionSet objDimensionSet = new DimensionSet();
           objDimensionSet.currentUser = CurrentUser;
           objDimensionSet.setUserDimensionSet();
           objDimensionSet.searchParameter = SearchParameter;
           objDimensionSet.setBookDimensionSet();
           objDimensionSet.ComputeSimilarity();

           SkylineQuery objSkylineQuery = new SkylineQuery();
           objSkylineQuery.userDimensionSet =
               objDimensionSet.userDimensionSet;
           objSkylineQuery.bookDimensionSet =
               objDimensionSet.bookDimensionSet;

           if (CurrentUser == null){
                 objSkylineQuery.getDimensions(1);
                 objSkylineQuery.processSkylineQuery(1);
                 .
                 .
                 .
```

Abbildung 6.8. Option Suche

Um dem Benutzer die Empfehlungen zu präsentieren, wird die Klasse `Recommendations` aufgerufen, um die Liste der empfohlenen Bücher zu erstellen, und anschließend wird die Methode `showRecommendations()` durchgeführt.

```
if (CurrentUser == null){
      .
      .
      .
      Recommendations objRecommendations =
            new Recommendations();
      objRecommendations.setRecommendations(1);
      DefaultTableModel modelRecommendations =
            objRecommendations.modelRecommendations;
      showRecommendations(1,modelRecommendation);
```

Abbildung 6.9. Darstellung der Empfehlungen

Abbildung 6.10 stellt die Ergebnisse einer Empfehlungssuche dar. Da der Benutzer eingeloggt ist wurden die drei möglichen Anfragen des Systems (s.3.1) durchgeführt.

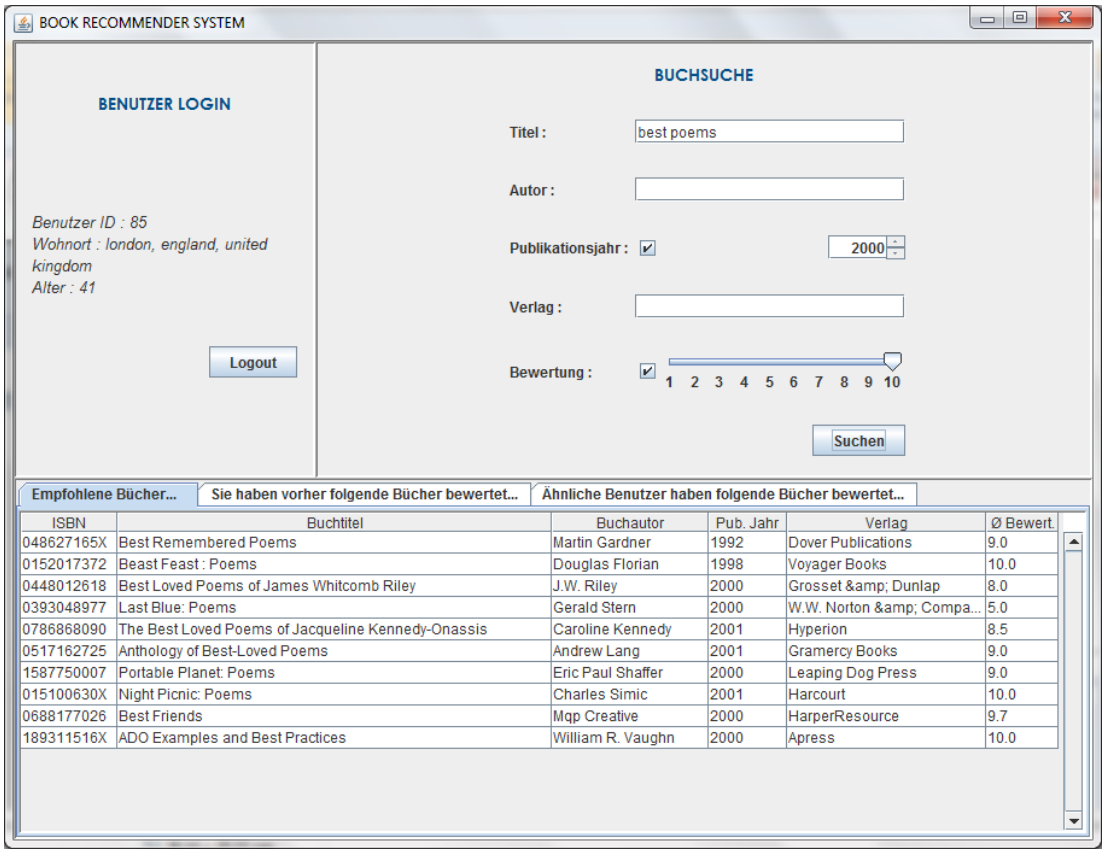

Abbildung 6.10. Darstellung der Empfehlungen in der Benutzer Schnittstelle

6.2.2 Benutzer Verwaltung

Für die Implementierung dieser Komponente wurde die Klasse `CurrentUser` erstellt. Sie implementiert die Schnittstelle `I_CurrentUser`.

```
public class CurrentUser implements I_CurrentUser {
```

Abbildung 6.11. Deklaration der Klasse `CurrentUser`

Die Klasse `CurrentUser` erhält die Benutzer Identifikation durch die Schnittstelle `I_UserIdentificator`, die in der Klasse `BookRecommender_GUI` implementiert wurde. Die Klasse `CurrentUser` sucht mittels der Methode `getUserRecord()` nach dem Benutzer. Wenn der Benutzer gefunden ist, stellt die Methode `setCurrentUser()` aus der Schnittstelle `I_CurrentUser` die Informationen zur Verfügung.

6.2.3 Recommender Modul

Die Komponente Recommender Modul enthält die Klassen `DimensionSet` und `Recommendations`. Die Klasse `DimensionSet` implementiert die Schnittstellen `I_UserDimensionSet`, `I_BookDimensionSet` und die Klasse `Recommendations` die Schnittstelle `I_Recommendations`.

```
public class DimensionSet implements I_UserDimensionSet,
I_BookDimensionSet {
```

Abbildung 6.12. Deklaration der Klasse `DimensionSet`

Die Klasse `DimensionSet` steuert die Dimensionen der Anfrage. Nach dem Empfang der Informationen des aktuellen Benutzers, bestimmt sie die Anfragedimensionen und stellt sie mittels der Methode `setUserDimensionSet()` aus der Schnittstelle `I_UserDimensionSet` und der Methode `setBookDimensionSet()` aus der Schnittstelle `I_BookDimensionSet` zur Verfügung.

Die Klasse `Recommendations` hat den Zweck, die Bücher Empfehlungsliste zu ermitteln. Dazu wird ein Datensatz der Tabelle „BX-Books“, für jeden Datensatz der Tabelle „temp_Skyline“, mittels der Methode `getBookInformation()` selektiert. Die Empfehlungsliste wird durch die Methode `setRecommendations()`, aus der Schnittstelle `I_Recommendations`, zur Verfügung gestellt.

```
public class Recommendations implements I_Recommendations {
```

Abbildung 6.13. Deklaration der Klasse `Recommendations`

6.2.4 Ähnlichkeitsbewertungs Modul

Die Komponente Ähnlichkeitsbewertungs Modul enthält die Klassen `UserDimensionSet` und `BookDimensionSet`.

Die Klasse `UserDimensionSet` ist verantwortlich für die Ähnlichkeitsbewertung der Benutzer Dimensionen. Nach dem Empfang der Benutzerdimensionen führt die Klasse die Ähnlichkeitsbewertung mithilfe der Methode `computeUserSimilarity()` durch. Dieser Methode greift auf die „BookCrossing“ Datenbank zu und führt die gespeicherte SQL-Prozedur „p_ComputeUserSimilarity“ aus.

```
-- CU_Location
if CU_Location <> ''
then
        SET @s1 = CONCAT(@s1, ', location');
        SET @s2 = CONCAT(@s2, ',
                        f_similarityLocation(\'',CU_location,'\',Location)');
end if;

-- CU_Age
if CU_Age IS NOT NULL
then
        SET @s1 = CONCAT(@s1, ', age');
        SET @s2 = CONCAT(@s2, ', (case when Age is not null then
                abs(',cast(CU_Age as char), ' - Age) else 100 end) as d_age');
end if;
```

Abbildung 6.14. Teil des Skripts der SQL-Prozedur p_ComputeUserSimilarity

Die gespeicherte SQL-Prozedur p_ComputeUserSimilarity vergleicht die Werte der Felder „location“ und „age“ der Tabelle „BX-Users“ mit den Werten der Benutzerdimensionen und speichert die erhaltenen Werte in der Tabelle „d_users“.

Der Vergleich für „Location“ wird mittels der SQL-Funktion „f_similarityLocation“ realisiert. Die Funktion wird nach dem in Abschnitt 5.6.2 beschriebenen Vorgang implementiert und in folgende Abbildung präsentiert.

```
-- Funktionsparameter speichern
set loc1 = location1;
set loc2 = location2;

-- Erster Location-Wert in drei Unterbereiche zerlegt
set pos = instr(loc1, ',');
set loc1_1 = substr(loc1, 1, pos-1);
set loc1 = substr(loc1, pos+1);
set pos = instr(loc1, ',');
set loc1_2 = substr(loc1, 1, pos-1);
set loc1_3 = substring(loc1, pos+1);
    .
    .
    .
-- Vergleich
set comparison = 3;
if (loc1_3 = loc2_3) then set comparison = comparison - 1; end if;
if (loc1_2 = loc2_2) then set comparison = comparison - 1; end if;
if (loc1_1 = loc2_1) then set comparison = comparison - 1; end if;

return comparison;
```

Abbildung 6.15. Teil des Skripts der SQL-Funktion f_similarityLocation

Für den Vergleich des Alters (age) wird die Differenz des Feldes „age“ mit dem Alter des aktuellen Benutzers berechnet. Es sei darauf hingewiesen, dass dieses Feld das einzige ist, in dem es Nullwerte geben kann. Aus diesem Grund muss die Ähnlichkeit in diesem Fall den schlechtesten Wert bekommen. Dieser wurde auf 100 gestellt.

Ebenso realisiert die Klasse `BookDimensionSet` einen ähnlichen Vorgang für Bücher. Nachdem einmal die Buchdimensionen erhalten wurden, wird die Ähnlichkeitsbewertung durch die Ausführung der Methode `computeBookSimilarity()` realisiert. Dieser Methode greift auf die „BookCrossing“ Datenbank zu und führt die gespeicherte SQL-Prozedur p_ComputeBookSimilarity aus.

Die gespeicherte SQL-Prozedur `p_ComputeBookSimilarity` vergleicht die Werte der Felder „bookTitle“, „bookAuthor“ und „publisher“ aus der Tabelle „BX-Books“ mit den für den Benutzer eingegebenen Suchparametern.

```
-- CU_Title
if CU_Title <> ''
then
        SET @s1 = CONCAT(@s1, ', booktitle');
        SET @s2 = CONCAT(@s2, ', f_valueOfTrigrammComparison(\'', CU_Title,'\',
                                                                compBookTitle)');
end if;
```

Abbildung 6.16. Teil des Skripts der SQL-Prozedur p_ComputeUserSimilarity für den Vergleich textueller Daten

Der Vergleich textueller Daten wird mittels der SQL-Funktion „f_valueOfTrigrammComparison“ realisiert. Die Funktion vergleicht zwei Zeichenketten, wobei in beiden nur Buchstabe von A bis Z, Nummern von 0 bis 9 und Leerzeichen beinhaltet sein dürfen. Die Suchparameter wurden vorher mittels der Java-Methode `comparableString()` der Klasse `BookRecommender_GUI` in dieses Format umgewandelt. Der Vergleich wird mit den berechneten Datenfeldern, die auch das benötige Format haben (s. 6.1.3.2), realisiert.

Die SQL-Funktion f_valueOfTrigrammComparison sucht die Anzahl ähnlicher Trigramme in beiden Zeichenketten. Der Vorgang wird nach dem in Abschnitt 5.6.2 beschriebenen Vorgang realisiert und in folgender Abbildung präsentiert.

```
-- Anzahl der Trigramme aus Str1 in Str2 bestimmen
set numCommonTrigramms1 = 0;
set i = 1;
while (i<=length(compStr1)-2) do
        set trigramm = substring(compStr1,i,3);
        if instr(compStr2,trigramm)>0
        then
                set numCommonTrigramms1 = numCommonTrigramms1 + 1;
        end if;
        set i:= i + 1;
end while;
.
.
.
-- Anzahl gemeinsamer Trigramme / Anzahl aller Trigramme
set equalityRate1 = numCommonTrigramms1/(length(compStr1)-2);
```

Abbildung 6.17. Teil des Skripts der SQL-Funktion f_valueOfTrigrammComparison

Für den Vergleich für „yearOfPublication“ und „rating“ berechnet die SQL-Prozedur p_ComputeBookSimilarity die Differenz zwischen den Werten der genannten Felder mit den für den Benutzer eingegebenen Werten.

Nach der Ähnlichkeitsbewertung der textuellen, sowie den numerischen Daten der Buchdimensionen, werden die gefundenen Werte in der Tabelle „d_books“ gespeichert.

6.2.5 Skyline Query Processing Modul

Diese Komponente ist verantwortlich für die Verarbeitung einer Skyline Query. Als ersten Schritt muss das Modul die Skyline Queries der Suche definieren. Danach kann der Algorithmus für die Verarbeitung einer Skyline Query durchgeführt werden. Schließlich werden Informationen der Bücher, die den Skyline Punkten entsprechen, abgerufen.

Die Komponente ist verantwortlich für die Verarbeitung einer Skyline Query und enthält die Klasse `SkylineQuery`. Nach dem Empfang der Anfragedimensionen wird die Methode `processSkylineQuery()` ausgeführt, um die Verarbeitung der Skyline Query zu realisieren. Die Klasse enthält außerdem die Methoden `getDimensions`(), `setDataRegion()`, `getSearchRegion()`, `getBoundaryPoint()`, `getPmd()`, `saveSkylinePoint()`, `setGSS()`, und `computeSkylinePoints()`, welche in Abschnitt 5.7.3 beschrieben wurden.

Die Elemente des Suchbereiches werden in den Tabellen „temp_SkyineSearchRegion1“, „temp_SkyineSearchRegion2“ und „temp_SkyineSearchRegion3“ gespeichert. Die gefundenen Skyline Punkte werden in den Tabellen „temp_Skyline1“, „temp_Skyline2“ und „temp_Skyline3“, wobei die Nummern 1, 2, und 3 den Anfragetyp entsprechen, gespeichert.

6.2.6 DBZugriffAdapter

Diese Komponente enthält die Klasse `DBManager`, deren Methoden den Zugriff auf die „BookCrossing“ Datenbank und die Ausführung von SQL-Anweisungen ermöglichen.

Die Klasse DBManager enthält die Methoden `ConnectDB()` und `DisconnectDB()`, um die Datenbankverbindung herzustellen oder zu trennen, die Methode `MakeQueryWithReturn()`, um SQL-Anweisungen, die Informationen zurückgeben sollten, auszuführen, und die Methode `MakeQueryWithoutReturn()`, um SQL-Anweisungen, die keine Information zurückgeben sollten, auszuführen.

6.3 Validierung der Anwendung

In diesem Abschnitt wird die Validierung der Anwendung vorgenommen. Als erstes wurde die Funktionalität der Anwendung als Ganzes überprüft. Anschließend wurden alle Bestandteile der Anwendung, und zwar Klassen und SQL-Methoden überprüft. Schließlich wurde ein Anforderungsabgleich mit den Funktionalen Anforderungen des Systems (s .3.4) realisiert

6.3.1 Eingabedaten

Die Eingabedaten entsprechen der Benutzer-ID, sowie den Bücher Suchparametern. Die Benutzer-ID sowie die Suchparameter Publikationsjahr und Bewertung dürfen nur numerische Werte enthalten. Im Fall der Benutzer-ID, wird diese Beschränkung bei einer `KeyListener` Methode der Klasse `BookRecommender_GUI` kontrolliert. Für die Parameter Publikationsjahr und Bewertung, wurden Java-Kontrollen, die nur numerische Werte annehmen, verwendet.

Anderseits müssen die Suchparameter Titel, Autor und Verlag gültige Werte beinhalten. Ein gültiger Wert muss mindestens aus einem Zeichen wie a..z, A..Z und 0..9 bestehen z.B. „best-poems“, „best & poems“, „poems (best)“. Die Zeichen, die nicht in dieser Gruppe sind, werden von der Methode `comparableString()` der Klasse `BookRecommender_GUI` entfernt. Somit werden die Suchparameter Titel, Autor und Verlag die gewünschte Format für den in Abschnitt 6.1.3.2 beschriebene Vergleich textueller Daten.

Um die Eingabedaten zu validieren wurde die Anwendung mit folgenden Werten überprüft.

	Eingabe	Erwarteter Wert
Benutzer-ID	0	0
	85	85
	‘a’	Nullwert
Titel	‘best poems’	‘best poems’
	‘best & poems’	‘best poems’
	‘#+&/’	‘ ’
Autor	‘dickinson’	‘dickinson’
	‘ *dickinson* ’	‘dickinson’
	‘#+&/’	‘ ’
Publikationsjahr	2000	2000
	1900	1900
	2010	2010

Verlag	‘harper’	‘harper’
	‘harper %&#’	‘harper’
	‘#+&/’	‘ ’
Bewertung	1	1
	7	7
	10	10

Tabelle 6.2. Test Eingabedaten

Die Ausführung der Anwendung nach Eingabe der Daten der Tabelle 6.2 hat in allen Fällen die erwarteten Antworten gegeben.

6.3.2 Ausgabedaten

Die Ausgabedaten werden von den Optionen „Login“ und „Suchen“ produziert. „Login“ gibt die Informationen des aktuellen Benutzers aus und „Suchen“ die empfohlenen Bücher.

Bei der Ausführung der Anwendung mit Angabe der gültigen Werte aus Tabelle 6.2 haben sich die erwarteten Antworten ergeben. Die Ausgabe des Wohnortes und Alter des Benutzers bei der Option „Login“ und die Listen der empfohlenen Bücher bei der Option „Suchen“. Die Korrektheit der Ausgabedaten wird im nächsten Abschnitt überprüft

6.3.3 Klassen

Für die Validierung der Klassen der Anwendung wurde ein Unit-Test[5], welcher die Überprüfung jedes Bestandteiles der Anwendung beziehungsweise seiner Klassen und Methoden ermöglicht, realisiert. Hierfür wurden auf das Testing-Framework JUnit[6] basierende Testmethoden für jede Methode der Anwendung implementiert.

6.3.3.1. Klasse BookRecommender_GUI

Die Methoden dieser Klasse sind verantwortlich für die Validierung der von dem Benutzer eingegebenen Daten, sowie der Darstellung der Ergebnisse der Suche. Die Funktionalität dieser Methoden wurden in der Validierung der Eingabedaten (s. 6.3.1) und Ausgabedaten (s.6.3.2) abgedeckt.

5 Auch Modultest oder Komponententest genannt.
6 Testing-Framework für Java-Programme. www.junit.org

6.3.3.2. Klasse CurrentUser

In dieser Klasse wird die Methode `getUserRecord()` überprüft. Die JUnit-Methode `test_getUserRecord()` prüft die richtige Ausführung der Methode.

```
@Test
	public void test_getUserRecord(){
		CurrentUser testClass = new CurrentUser();
		testClass.userId = "6";
		assertNotNull(testClass.getUserRecord());

	}
```

Abbildung 6.18. Testausführung der Methode `getUserRecord()`

Es gibt zwei Testfälle. Im ersten Fall existiert der Wert für die User-ID. Im zweiten Fall nicht. Die Testdaten für den ersten Fall sind nummerisch zwischen 1 und 278858, da das Feld „User-ID“ der Tabelle „BX-Users“ aus dem Book-Crossing Dataset Werte von 1 bis 278858 beinhaltet. Die Testdaten für den zweiten Fall sind positive Nummern außerhalb dieses Bereiches. Die Methode `assertNotNull()` beweist, dass der Methode `getUserRecord()` die Informationen der Benutzer „6“ zurückgibt.

6.3.3.3. Klasse UserDimensionSet

In dieser Klasse wird die Methode `computeUserSimilarity()` durch die Ausführung der JUnit-Methode `test_computeUserSimilarity()` überprüft. Da die Methode `computeUserSimilariy()` die gespeicherte SQL-Prozedur „p_ComputeUserSimilarity“ abruft und die gespeicherte SQL-Prozedur ihre Ergebnisse in der Tabelle „d_users“ speichert, werden, um ihre Korrektheit zu bestimmen, diese Ergebnisse durch eine Abfrage über die Tabelle „d_users“ in der MySQL-Konsole abgerufen.

Als Testdaten wurden die Datensätze von folgenden fünf Benutzern genommen.

User-ID	location	age
6	santa monica, california, usa	61
87	richardson, texas, usa	(null)
2900	cosenza, calabria, italy	18
17068	williamstown, victoria, australia	50
22952	harthausen, rheinland-pfalz, germany	30

Tabelle 6.3. Testdaten für Methode `computeUserSimilarity()`

Die folgende Abbildung stellt die Ausführung der Testmethode nach Angabe der Daten von Benutzer „6“ dar.

```
@Test
public void test_computeUserSimilarity(){
	UserDimensionSet testClass = new UserDimensionSet();
	//User-ID
	testClass.dimensionSet[0][0]= "6";
	//Location
	testClass.dimensionSet[0][1]= "santa monica, california,
	                               usa";
	//Age
	testClass.dimensionSet[0][2]= "61";
	testClass.computeUserSimilarity();
}
```

Abbildung 6.19. Testausführung der Methode `computeUserSimilarity()`

Die Ergebnisse der Ausführung der Methode `computeUserSimilarity()` befinden sich in der temporären Tabelle „d_users“. Anschließend werden die ersten fünf Datensätze der Tabelle „d_users“ für die Ausführung nach Angabe der Daten des Benutzers „6“ präsentiert.

User-ID	location	age
1	2	(null)
2	2	43
3	3	(null)
4	3	44
5	3	(null)

Tabelle 6.4. Datensätze in Tabelle “d_users”

Nach der Analyse der Datensätze der Tabelle „d_users“ wurde es festgestellt, dass die Tabelle die erwarteten Daten beinhaltet.

6.3.3.4. Klasse BookDimensionSet

Analog zum vorherigen Abschnitt wird in dieser Klasse die Methode `computeBookSimilarity()` durch die Ausführung der JUnit-Methode `test_computeBookSimilarity()` überprüft. Da die Methode `computeBookSimilarity()` die gespeicherte SQL-Prozedur „p_ComputeBookSimilarity“ abruft und die gespeicherte SQL-Prozedur ihre Ergebnisse in der Tabelle „d_books“ speichert, werden, um ihre Korrektheit zu bestimmen, diese

Ergebnisse durch eine Abfrage über die Tabelle „d_books“ in der MySQL-Konsole abgerufen.

Titel:	best poems
Autor:	„“
Publikationsjahr:	2000
Verlag:	„“
Bewertung:	6

Tabelle 6.5. Testdaten für Methode `computeBookSimilarity()`

Die Testdaten der Tabelle 6.5 werden in der Ausführung der Testmethode verwendet.

```
@Test
public void test_computeBookSimilarity(){
      BookDimensionSet testClass = new BookDimensionSet();
      //bookTitle
      testClass.dimensionSet[0][1]= "best poems";
      //bookAuthor
      testClass.dimensionSet[1][1]= "";
      //yearOfPublication
      testClass.dimensionSet[2][1]= "2000";
      //publisher
      testClass.dimensionSet[3][1]= "";
      //rating
      testClass.dimensionSet[4][1]= "6";
      testClass.computeBookSimilarity();
}
```

Abbildung 6.20. Testausführung der Methode `computeBookSimilarity()`

Die Variable `dimensionSet` beinhaltet die Suchparameter einer Testsuche. Die Ergebnisse der Ausführung der Methode befinden sich in der temporären Tabelle „d_books“. Anschließend werden die ersten fünf Datensätze der Tabelle „d_books“ präsentiert.

isbn	bookTitle	bookAuthor	yearOfPublication	publisher	rating
0195153448	100,0	(null)	2	(null)	0
0002005018	100,0	(null)	1	(null)	1,7
0060973129	100,0	(null)	9	(null)	1,5
0374157065	86,3	(null)	1	(null)	1,8
0393045218	94,3	(null)	1	(null)	0

Tabelle 6.6. Datensätze in Tabelle “d_books”

Da es kein Wert für die Suchparameter „Autor“ und „Verlag“ in der Tabelle 6.5 gibt, haben die Felder „bookAutor“ und „yearOfPublication“ nach der Ausführung der Methode `computeBookSimilarity()` Nullwerte bekommen.

Nach der Analyse der Datensätze der Tabelle „d_books“ wurde es festgestellt, dass die Tabelle die erwarteten Daten beinhaltet.

6.3.3.5. Klasse SkylineQuery

In dieser Klasse werden die Methoden `getDimensions()`, `setDataRegion()`, `getSearchRegion()`, `getBoundaryPoint()`, `getPmd()`, `saveSkylinePoint()`, `setGSS()`, und `getNumberOfPoints()` mittels der Ausführung von JUnit-Testmethoden überprüft. Da die Methoden `setDataRegion()`,`saveSkylinePoint()`, `computeBookSimilarity()` und `setGSS()` ihre Ergebnisse in der Tabellen „temp_SkylineSearchRegion“ und „temp_Skyline“ speichern, werden, um ihre Korrektheit zu bestimmen, diese Ergebnisse durch eine Abfrage über die Tabellen „temp_SkylineSearchRegion“ und „temp_Skyline“ in der MySQL-Konsole abgerufen.

```
@Test
public void test_setDataRegion(){
      SkylineQuery testClass = new SkylineQuery();

      //Benutzerdimensionen
      testClass.userDimensionSet[0][0]= "age";
      testClass.userDimensionSet[0][1]= 61";
      testClass.userDimensionSet[1][0]= "age";
      testClass.userDimensionSet[1][1]= "santa monica,
                                        california, usa"
      //Bücherdimensionen
      testClass.bookDimensionSet[0][0]= "bookTitle";
      testClass.bookDimensionSet[0][1]= "best poems";
      testClass.bookDimensionSet[1][0]= "bookAuthor";
      testClass.bookDimensionSet[1][1]= "";
      testClass.bookDimensionSet[2][0]= "yearOfPublication";
      testClass.bookDimensionSet[2][1]= "2000";
      testClass.bookDimensionSet[3][0]= "publisher";
      testClass.bookDimensionSet[3][1]= "";
      testClass.bookDimensionSet[4][0]= "rating";
      testClass.bookDimensionSet[4][1]= "6";
      //Ausführung für die Anfrage "Passende Bücher"
      testClass.getDimensions(1);
      testClass.setDataRegion(1);
}
```

Abbildung 6.21. Testausführung der Methode `setDataRegion()`

Für die Ausführung der Methode `setDataRegion()` wird es notwendig die Anfragedimensionen zu definieren. Dazu werden den Variablen `userdimensionSet` und `bookDimensionSet` der Klasse `SkylineQuery` die Suchparameter zugewiesen und die Methode `getDimensions()` ausgeführt. Die Anfrage „Passende Bücher“, die als Testfall genommen wurde (Abbildung 6.21), liefert ihre Ergebnisse der Methode setDataRegion() in die Tabelle „temp_SkylineSearchRegion1“. Für den Test wurden die ersten fünf Datensätze genommen.

isbn	bookTitle	yearOfPublication	rating
0002005018	100,0	1	1,7
0060973129	100,0	9	1,5
0374157065	86,3	1	1,8
0399135782	94,2	9	2,2
0425176428	75,7	0	2,0

Tabelle 6.7. Datensätze in Tabelle “temp_SkylineSearchRegion1”

Abbildung 6.22 stellt die Testausführung der Methode `getPmd()` dar. Für den Test wurden zuerst die Variablen `boundaryPoint` und `searchRegion` initialisiert. Nachfolgend wurde durch die Methode `assertEquals()` festgestellt, dass das Ergebnis dem erwarteten Wert entspricht.

```
@Test
public void test_getPmd(){
      SkylineQuery testClass = new SkylineQuery();
      .
      .
      .
      //Initialisierung von BoundaryPoint (0,0,0)
      Vector<String[]> boundaryPoint = new Vector<String[]>(1,1);
      boundaryPoint = testClass.getBoundaryPoint();

      //Suchbereich der ganzen Datenbereiche bestimmen
      String searchRegion = testClass.getSearchRegion(1, "",
                                                null, null);

      //Erwartete Werte für mostDominatingPoint
      Vector<String[]> pmd = new Vector<String[]>(1,1);
      pmd.add(testClass.element("bookTitle","75,7"));
      pmd.add(testClass.element("yearOfPublication","0"));
      pmd.add(testClass.element("rating","2,0"));

      assertEquals(pmd, testClass.getPmd(searchRegion,
                                                boundaryPoint));
}
```

Abbildung 6.22. Testausführung der Methode `getPmd()`

Anschließend wird in der Abbildung 6.23 die Testausführung der Methoden `saveSkyinePoints()` und `setGSS()` dargestellt. Darauf folgend wurden die Daten der Tabellen „temp_Skyline1“ und „temp_SkylineSearchRegion1“ analysiert, um die Korrektheit der Methode festzustellen.

```
@Test
public void test_setGSS(){
      SkylineQuery testClass = new SkylineQuery();

      //Initialisierung von mostDominatingPoint
      Vector<String[]> pmd = new Vector<String[]>(1,1);
      pmd.add(testClass.element("bookTitle","75,7"));
      pmd.add(testClass.element("yearOfPublication","0"));
      pmd.add(testClass.element("rating","2,0"));

      testClass.saveSkylinePoint(1, pmd);
      testClass.setGSS(1, pmd);
}
```

Abbildung 6.23. Testausführung der Methode `saveSkylinePoint()` und `setGSS()`

Tabelle 6.8 stellt die Daten der Tabelle “temp_Skyline1” nach der ersten Ausführung der Methode `saveSkylinePoint()` dar.

isbn	bookTitle	yearOfPublication	rating
0425176428	75,7	0	2,0

Tabelle 6.8. Datensätze in Tabelle “temp_Skyline1”

Tabelle 6.9 stellt die Daten der Tabelle “temp_SkylineSearchRegion1” nach der ersten Ausführung der Methode `setGSS()` dar.

isbn	bookTitle	yearOfPublication	rating
0002005018	100,0	1	1,7
0060973129	100,0	9	1,5
0374157065	86,3	1	1,8

Tabelle 6.9. Datensätze in Tabelle “temp_SkylineSearchRegion1” nach Ausführung der Methode `setGSS()`

Am Ende der Methode `processSkylineQuery()` befinden sich alle Skyline Punkte in der Tabelle „temp_Skyline1“.

isbn	bookTitle	yearOfPublication	rating
0425176428	75,7	0	2,0
0002005018	100,0	1	1,7
0060973129	100,0	9	1,5
0374157065	86,3	1	1,8

Tabelle 6.10. Skyline Punkte der Testausführung

Nach der Analyse der Datensätze der Tabellen „temp_SkylineSearchRegion1“ und „temp_Skyline1“ wurde es festgestellt, dass die Tabellen die erwarteten Daten beinhalten.

6.3.3.6. Klasse Recommendations

In dieser Klasse wird die Methode `getRecommendedBooks()` überprüft. Die Methode führt eine SQL-Abfrage über die Tabellen „temp_Skyline“ und „BX-Books“ durch, welche die Liste der empfohlenen Bücher ermittelt. Diese Liste wird in der Benutzerschnittstelle als Ausgabedaten präsentiert.

Für die Überprüfung der Methode wurde die Korrektheit der SQL-Abfrage durch Analyse ihre Ergebnisse überprüft und wurde es festgestellt, dass sie die erwarteten Daten beinhalten.

6.3.3.7. Klasse DBManager

Die Funktionalität der Methoden dieser Klasse wurde durch die Ausführung der Methoden anderer Klassen, die mit der Datenbank interagieren, überprüft.

6.4 Erweiterungen

Eine eventuelle Erweiterung des Systems ergibt sich bei Erhöhung der Benutzer- und Bücherdimensionen. In diesem Fall, müssen hauptsächlich Variablen und Parameter, die den Dimensionen entsprechen, modifiziert werden. Dazu werden folgende Veränderungen notwendig:

- Validierung der Eingabedaten.
- Definition der Variablen der Klasse `CurrentUser`.
- Definition der Variablen der Klasse `DimensionSet`.

- Abruf von gespeicherten SQL-Prozeduren in den Klassen `UserDimensionSet` und `BookDimensionSet`.
- Inhalt der gespeicherten SQL-Prozeduren p_`ComputeUserSimilarity` und p_`ComputeBookSimilarity`.
- Definition der Ausgabedaten in der Klasse `Recommendations`.

7 Evaluierung

In diesem Kapitel wird die Evaluierung der Leistung des implementierten Systems realisiert. Dafür sollen die funktionalen und technischen Anforderungen, welche in Abschnitt 4.2 und 4.4 beschrieben wurden, berücksichtigt werden.

Die Evaluierung wird in vier Bereiche aufgeteilt. Im ersten Teil wird die Leistung des Ähnlichkeitsbewertungs Moduls gemessen, im zweiten Teil die Leistung des Skyline Query Processing Moduls, im dritten Teil die Qualität der Empfehlungen und im vierten Teil schließlich die Leistung des Systems als Ganzes.

7.1 Testumgebung

Alle Tests wurden auf einer Intel Core i7 (1.6 GHz, 4 Cores) CPU, mit 4GB Arbeitsspeicher und dem 64-Bit-Betriebssystem Windows 7 Home Premium realisiert. Die Anwendung wurde in Java implementiert, die 175MB Datenbank ist vom DBMS MySQL 5.5 gesteuert und auf einer Seagate 500GB Festplatte mit 5400 rpm gespeichert. Sowohl die Anwendung als auch die Datenbank laufen auf derselben CPU.

Die Daten des Book-Crossing Datasets (s. 3.2) sind in drei Tabellen gespeichert: „bx-books“ mit 271065 Datensätzen (108 MB), „bx-users“ mit 278858 Datensätze (13,4 MB) und „bx-book-ratings“ mit 1149780 Datensätzen (54,2 MB). Während der Ausführung der Anwendung werden die Daten aus „bx-books“ und „bx-users“ in die Tabellen „d_books“ und „d_users“ umgewandelt und zwei temporäre Tabellen für jede Anfrage erstellt. Für die Optimierung der Komposition der Tabellen wurden die Primär- und Sekundärschlüssel der Tabellen „d_books“, „d_users“ und „bx-book-ratings“ indexiert.

7.2 Leistung des Ähnlichkeitsbewertungs Moduls

Wie in den vorherigen Kapiteln beschrieben wurde, besteht das Ähnlichkeitsbewertungs Modul aus zwei wichtigen Methoden `computeUserSimilarity()` und `computeBookSimilarity()`. Sie wandeln die Datensätze der Tabellen „bx-books“ und „bx-users“ in numerische Daten, die den Dimensionenwerten entsprechen und in den Tabellen „d_books“ und „d_users“ gespeichert werden sollen, um.

Um die Leistung des Ähnlichkeitsbewertungs Moduls zu messen, sollen die Methoden `computeUserSimilarity()` und `computeBookSimilarity()` mit verschiedenen Mengen an Datensätzen und Dimensionen ausgeführt werden.

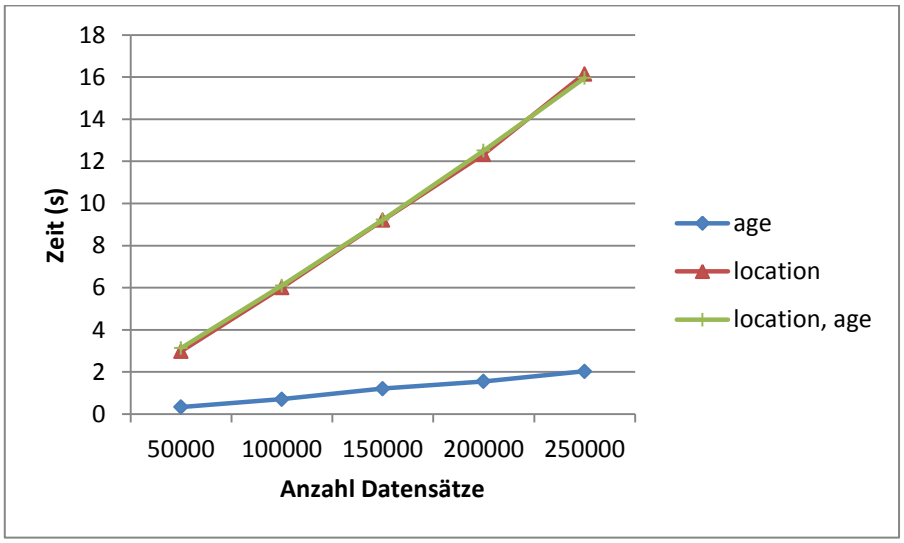

Abbildung 7.1. Leistung der Methode `computeUserSimilarity()` nach Anzahl der Datensätze

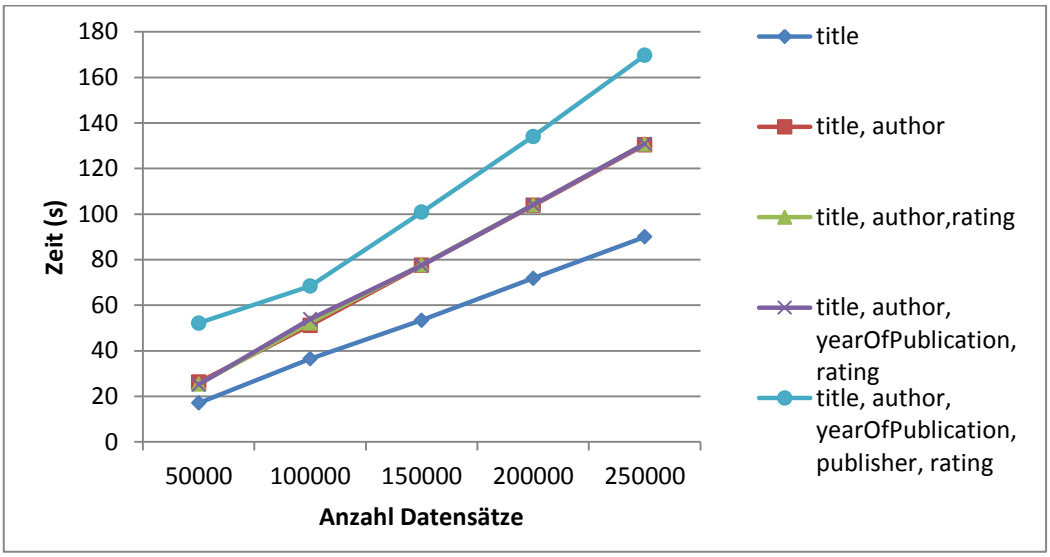

Abbildung 7.2. Leistung der Methode `computeBookSimilarity()` nach Anzahl der Datensätze

Die Abbildungen 7.1 und 7.2 stellen jeweils die Laufzeiten der Methoden `computeUserSimilarity()` und `computeBookSimilarity()` für verschieden viele Datensätzen dar. Es wurde festgestellt, dass die Laufzeit beider Methoden mit der Anzahl an Datensätzen linear steigt. Folglich liegt der Zeitaufwand der Methoden bei O(n).

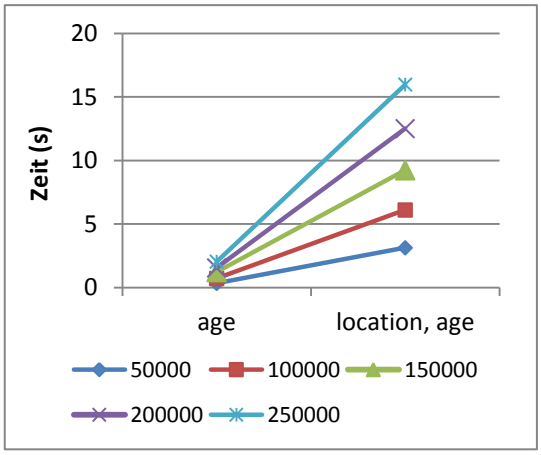

Abbildung 7.3.a. Leistung der Methode `computeUserSimilarity()` nach Dimensionen (age, location-age)

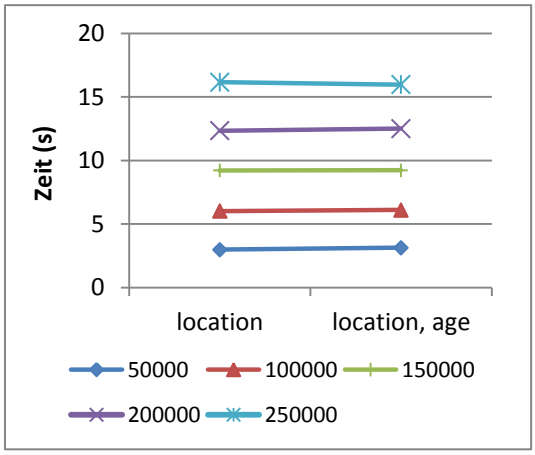

Abbildung 7.3.b. Leistung der Methode `computeUserSimilarity()` nach Dimensionen (location, location-age)

Die Abbildungen 7.3.a und 7.3.b stellen die Laufzeit der Methode `computeUserSimilarity()` für eine- und zwei Dimensionen dar. Es wurden fünf Ausführungen der Methode mit unterschiedlich vielen Datensätzen realisiert. Abbildung

7.3.a basiert in der ersten Dimension auf dem Wert „age“, welcher numerische Daten enthält, und Abbildung 7.3.b basiert auf der Dimension „location“, die textuelle Daten enthält. Aus diesen Informationen ist festzustellen, dass die Laufzeit mit der Anzahl der Dimensionen steigt. Allerdings nimmt die Verarbeitung textueller Daten mehr Zeit in Anspruch als die Verarbeitung numerischer Daten. So nimmt die Verarbeitung der Dimension „location“ ungefähr achtmal mehr Zeit in Anspruch als die Verarbeitung der Dimension „age“, und repräsentiert damit ca. 88% der gesamten Zeit. Dies liegt daran, dass für den Vergleich der Daten der Dimension „location“ die Wörter analysiert werden müssen (s. 6.1.3.2).

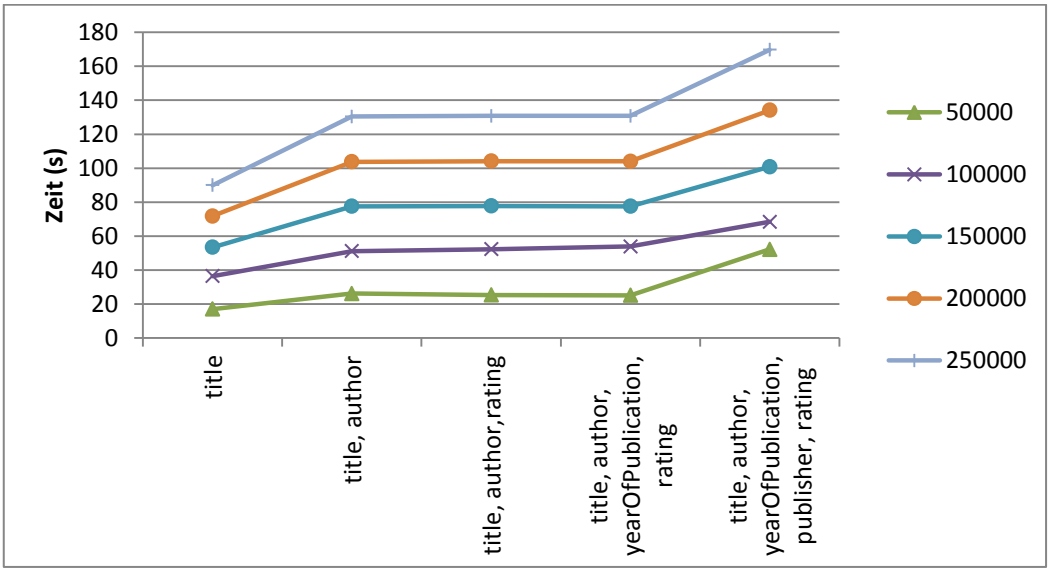

Abbildung 7.4. Leistung der Methode `computeUserSimilarity()` nach Dimensionen

Abbildung 7.4 stellt die Laufzeit der Methode `computeBookSimilarity()` für ein bis fünf Dimensionen, wobei nur die Dimensionen „rating“ und „yearOfPublication“ numerische Daten enthalten, dar. Wie beim Vergleich der Benutzer (`computeUserSimilarity()`), wurden fünf Ausführungen der Methode `computeBookSimilarity()` mit unterschiedlichen vielen Datensätzen realisiert. Dabei kann man beobachten, dass die Laufzeit mit der Anzahl der Dimensionen steigt und dass die Verarbeitung textueller Daten mehr Zeit in Anspruch nimmt als die Verarbeitung numerischer Daten. Die Dimensionen „title“, „author“ und „publisher“ beanspruchen jeweils durchschnittlich 48%, 23% und 28% der gesamten Zeit. Die Dimensionen „rating“ und „yearOfPublikation“ beanspruchen jeweils 0.5% der gesamten Zeit. Das bedeutet, dass die Verarbeitung der textuellen Daten ungefähr 99% der Zeit beansprucht. Der Grund dafür ist, dass jeder Buchstabe einer Zeichenkette analysiert werden muss, um die Ähnlichkeitswerte zu erhalten (Trigramme s. 1.3.2).

Wie zu sehen ist, ist der Zeitaufwand der Methoden `computeUserSimilarity()` und `computeBookSimilarity()` besonders hoch. Um dies genauer zu analysieren wurde die Leistung des Arbeitsspeichers, des Prozessors und des physikalischen Datenträgers des Testrechners (s.7.1) mit dem Windows Performance Monitor (Perfmon) gemessen. Dabei wurde festgestellt, dass die Laufzeit hauptsächlich von der Festplatten-Zugriffzeit bestimmt wird. Die Prozessorzeit beansprucht für beide Methoden nur ungefähr 12,5% der gesamten Ausführungszeit. Während die restliche Zeit für E/A-Operationen in Anspruch genommen wird. Im Fall der Methode `computeUserSimilarity()` generiert die Methode für die gesamten Benutzer-Datensätze (278858) einen durchschnittlichen E/A-Aufwand von 17,26 MB. Ebenso generiert die Methode `computeBookSimilarity()` für alle Bücher Datensätze (271065) einen durchschnittlichen E/A-Aufwand von 153,72 MB. Die Anzahl der Dimensionen aufweisen keinen signifikanten Einfluss auf den E/A-Aufwand.

Angesichts der Tatsache, dass die am häufigsten verwendeten Suchparameter Titel und Autor sind und eventuell die Ergebnisse nach ihrer Bewertung, wenn die Bewertungs Informationen zur Verfügung steht, sortiert werden müssen, werden die am häufigsten verwendeten Ausführungen für die Methode `computeBookSimilarity()` die ersten drei (title, title-author und title-author-rating) sein. Anderseits haben alle Datensätze der Tabelle „bx-users" einen Wert für das Feld „location“, so dass, wenn der Benutzer eingeloggt ist, die zweite und dritte Ausführung (location und location-age) der Methode `computeUserSimilarity()` die am häufigsten verwendeten Ausführungen sein werden.

Hinsichtlich der Qualität des Ergebnisses der Ähnlichkeitsbewertung ist zu analysieren, ob der Vergleich der Daten optimal ist. Im Fall von numerischen Daten ist der Vergleich 100% optimal, da nur arithmetische Berechnungen realisiert wurden (s. 5.6.1). Für den Vergleich textueller Daten (s. 5.6.2) kann man drei Fälle finden. Der erste Fall für die Dimension Buchtitel, der Zweite für Autor und Verlag und der Dritte für den Wohnort des Benutzers.

Für den Vergleich von Wohnorten, wird man nach ähnlichen Wörtern, die den Namen einer Region, einer Stadt oder einem Land entsprechen, suchen. Daher kann die Qualität des Vergleichs nur dadurch beeinträchtigen werden, dass der Ort falsch oder in andere Sprache geschrieben ist. Z. B. „Germany“ und „Deutschland“.

Der restliche Vergleiche textueller Daten basiert auf der Berechnung von Trigrammen (s. 6.2.4). Im Fall von Autor und Verlag, sucht man nach Wort-Elementen, die am ähnlichsten zu den eingegebenen Werten sind. Folglich ist die Verwendung von Trigrammen eine geeignete Methode dafür.

Der Vergleich von Buchtiteln wird folgenderweise analysiert. Tabelle 7.1 stellt die ähnlichsten Zehn Buchtitel zu „History of Europe“ dar. Wie zu sehen ist, hängen die gefundenen Buchtitel mit dem eingegebenen Wert eng zusammen.

History of Europe
A History of Pagan Europe
Europe: A History
The Penguin History of Europe
Medieval Europe 400-1500 (History of Europe)
The Oxford History of Medieval Europe
EUROPE CRISIS 1598-1648 (Fontana History of Europe)
The Oxford Illustrated History of Medieval Europe
A History of European Integration
Heart of Europe: A Short History of Poland (Oxford Paperbacks)

Tabelle 7.1. Die Zehn ähnlichsten Buchtitel zu “History of Europe”

Tabelle 7.2 stellt die ähnlichsten Zehn Buchtitel zu „Amazing places in the world“ dar. In diesem Fall hängen die gefundene Buchtitel mit dem eingegebenen Wert nicht so eng zusammen, wie es sein müsste.

Amazing Women: Amazing World
The World in the Evening
The amazing world of Kreskin
The World the World
In All the Wrong Places
World In the Evening
Dancing at the Edge of the World: Thoughts on Words, Women, Places
The Sacred Place: Witnessing the Holy in the Physical World
Living on the Edge : Amazing Relationships in the Natural World
Your Place In This World

Tabelle 7.2. Die Zehn ähnlichsten Buchtitel zu “Amazing places in the world ”

Der Unterschied der Qualität der Ergebnisse beider Testfälle liegt darin, dass die Zeichenkette „History of Europe“ spezifischer als die Zeichenkette „Amazing places in the world“ ist. Deshalb ist es wahrscheinlicher, mehr Wörter der ersten Zeichenkette zusammen in anderen Zeichenketten zu finden. Darüber hinaus ist die Länge der Zeichenkette relevant, denn je mehr Wörter in der Zeichenkette sind, desto schwerer ist es alle Wörter der Zeichenkette in einer anderen Zeichenkette zu finden.

Die Suchergebnisse für Titel sollen aber nicht nur Ergebnisse liefern, die ähnliche Wörter beinhalten, sondern vielmehr auch semantische Ähnlichkeit berücksichtigen. Da der Zeichenkettenvergleich des Algorithmus kein semantischer Vergleich ist, wird die Qualität

des Buchtitelvergleichs beeinträchtigt. Um einen semantischen Vergleich zu realisieren, wäre die Verwendung eines Thesaurus nötig.

7.3 Leistung des Skyline Query Processing Moduls

Das Skyline Query Processing Modul realisiert mehrere Schritte um die Skyline Punkte zu finden. In diesem Vorgang werden zwei temporäre Tabellen „temp_SkylineSearchRegion“ und „temp_Skyline“, in denen jeweils die Datenpunkte und die Skyline Punkte gespeichert werden sollen, für jede Anfrage erstellt.

Um die Leistung des Moduls zu messen, soll die Methode `processSkylineQuery()` mit verschiedenen Mengen von Datensätzen und Dimensionen ausgeführt werden. Die Testdaten der Ausführung werden in der folgenden Tabelle dargestellt.

Benutzer	Benutzer-ID:	85
	Wohnort:	London, England, United Kingdom
	Alter:	41
Buch	Titel:	best poems
	Autor:	Dickinson
	Publikationsjahr:	2000
	Verlag:	Harper
	Bewertung:	10

Tabelle 7.3. Testdaten für die Ausführung des Skyline Query Processing Moduls

Abbildung 7.5 stellt die Laufzeit der Methode `processSkylineQuery()` für eine verschiedene Anzahl an Datensätzen dar. Es wurden fünf Ausführungen mit jeweils eine bis fünf Dimensionen realisiert. Dieses Szenario entspricht einer Suche, in der der Benutzer nicht eingeloggt ist. Aus diesem Grund, wird die maximale Anzahl der Datensätze 271065 sein, da nur die Tabelle „d_books“ verwendet wird.

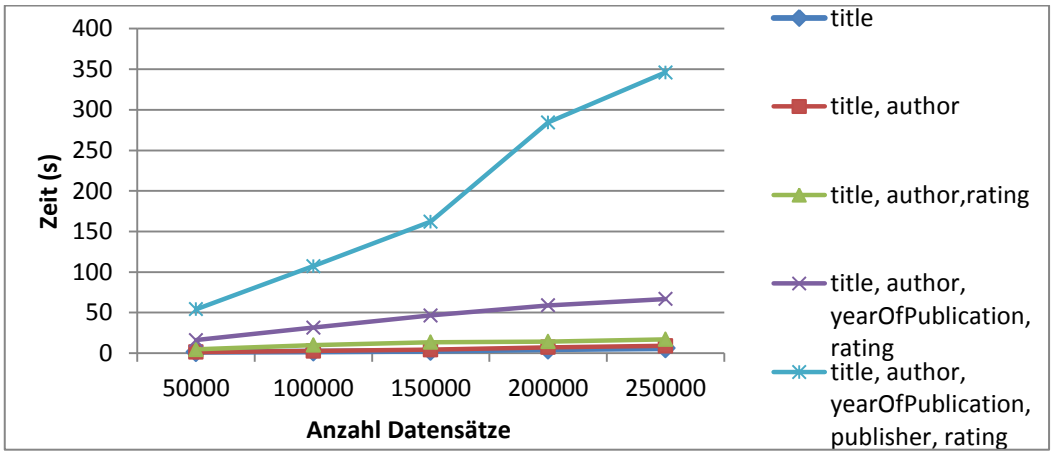

Abbildung 7.5. Leistung der Methode `processSkylineQuery()` für Buchdimensionen nach Anzahl der Datensätze

Abbildung 7.6 stellt die Laufzeit der Methode `processSkylineQuery()` für unterschiedlich viele Datensätzen dar. Es wurden fünf Ausführungen mit jeweils drei bis sieben Benutzer- und Buchdimensionen realisiert. Dieses Szenario entspricht einer Suche, in der der Benutzer eingeloggt ist. Folglich wird die maximale Anzahl der Datensätze 285437 sein, da die Datensätze aus einer Kombination der Tabellen „d_books“, „bx-book-ratings“ und „d_users“ kommen.

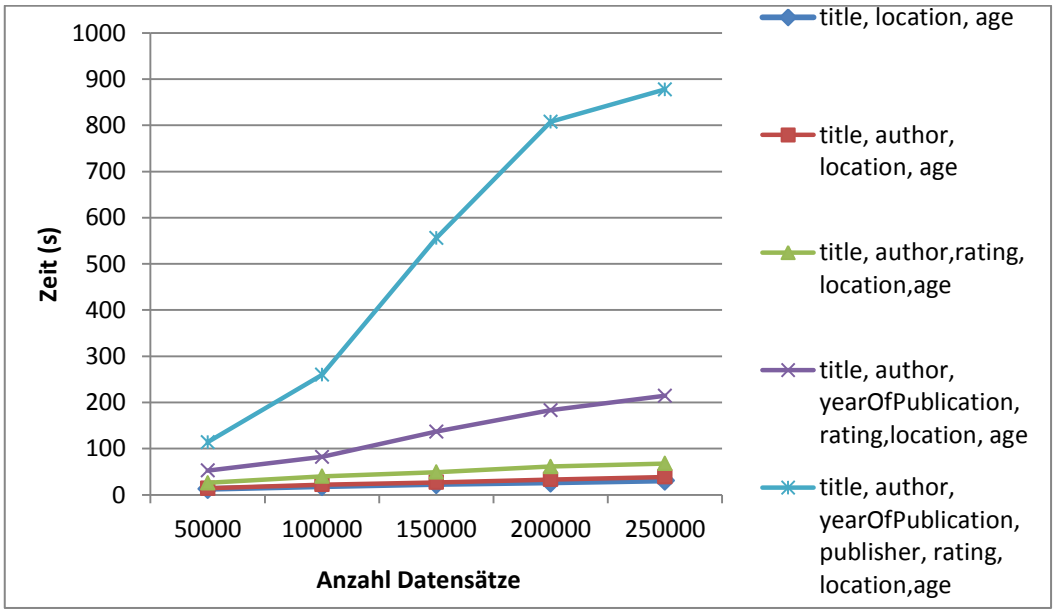

Abbildung 7.6. Leistung der Methode `processSkylineQuery()` für Benutzer- und Buchdimensionen nach Anzahl der Datensätze

Nach Analyse der in Abbildungen 7.5 und 7.6 dargestellten Daten, wurde festgestellt, dass die Laufzeit der Methode mit der Anzahl an Datensätzen linear steigt. Folglich liegt der Zeitaufwand der Methode bei O(n). Zudem erhöht die Einbeziehung der Benutzerdimensionen die Laufzeit auf das ungefähr 7.5-fache, da eine Join-Operation zwischen den Tabellen „d_books“, „bx-book-ratings“ und „d_users“ notwendig ist.

Anschließend wird die Leistung der Methode `processSkylineQuery()` nach Anzahl an Dimensionen in den Abbildungen 7.7 und 7.8 präsentiert. Wobei die Abbildung 7.7 die Ausführungen für nur Buchdimensionen, falls der Benutzer nicht eingeloggt ist, präsentiert und die Abbildung 7.8 die Ausführungen für Benutzer- und Buchdimensionen, fall der Benutzer eingeloggt ist.

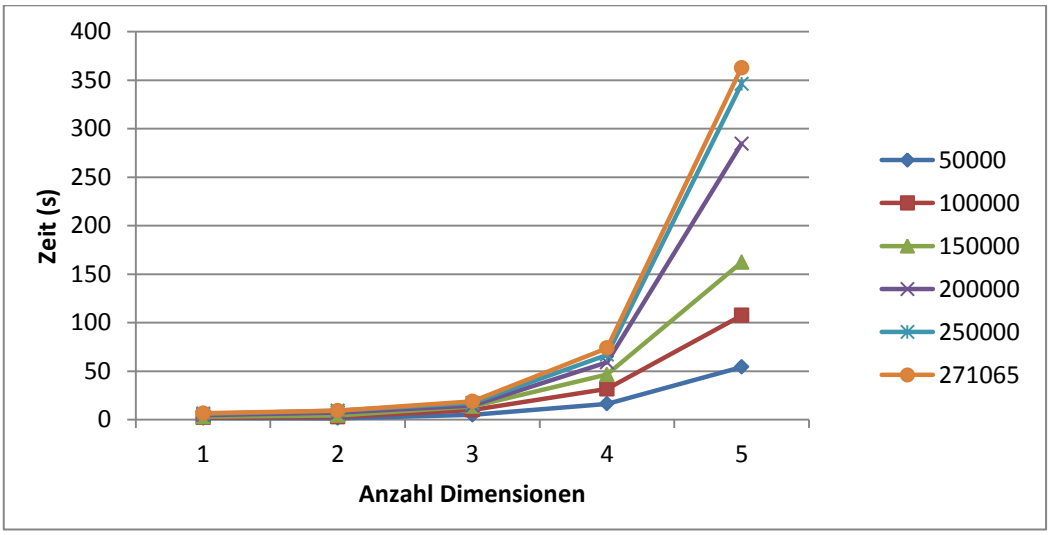

Abbildung 7.7. Leistung der Methode `processSkylineQuery()` für Buchdimensionen nach Anzahl an Dimensionen

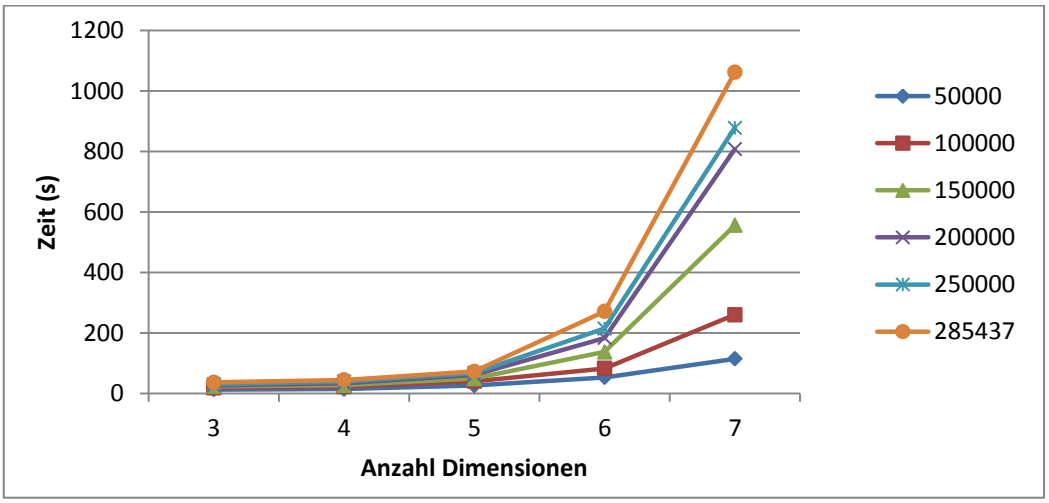

Abbildung 7.8. Leistung der Methode `processSkylineQuery()` für Benutzer- und Buchdimensionen nach Anzahl an Dimensionen

Wie in der Abbildungen 7.7 und 7.8 zu sehen ist, erhöht sich die Laufzeit der Methode zwischen mit der Anzahl an Dimensionen zwischen den ersten drei Dimensionen nur mäßig. Dann aber deutlich für die weiteren Dimensionen. So erreicht die Laufzeit in der ersten Abbildung im schlimmsten Fall ca. 360 Sekunden für fünf Dimensionen und 271065 Datensätze, und in der zweite Abbildung mehr als 1000 Sekunden für 7 Dimensionen und 285437 Datensätze. Dieses Verhalten zeigt, dass die Laufzeit exponentiell steigt. Folglich liegt der Zeitaufwand der Methode bei $O(c^n)$.

Genauso wie in der Methoden `computeUserSimilarity()` und `computeBookSimilarity()`, ist der Zeitaufwand der Methode `processSkylineQuery()` besonders hoch. Nach der Analyse der Leistung des Arbeitsspeichers, des Prozessors und des physikalischen Datenträgers des Testrechners (s.7.1), mithilfe des Windows Performance Monitors (Perfmon), wurde festgestellt, dass die Laufzeit auch in diesem Fall hauptsächlich von der Disk-Zugriffzeit bestimmt wird. So ist die Prozessorzeit für diese Methode nur ungefähr 19% der gesamten Zeit für Buchdimensionen und 22% für Benutzer- und Buchdimensionen zusammen. Die restliche Zeit werden für E/A-Operationen in Anspruch genommen.

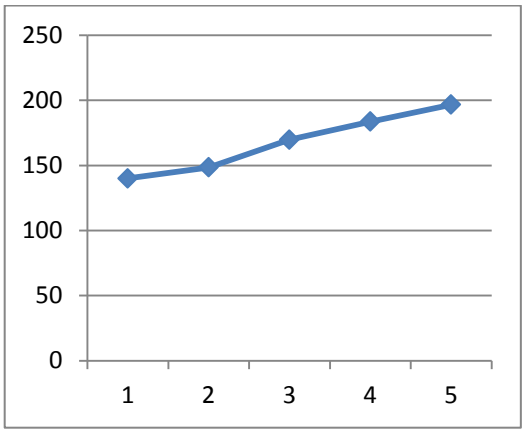

Abbildung 7.9. E/A-Aufwand (MB) der Methode `processSkylineQuery()` für Buchdimensionen

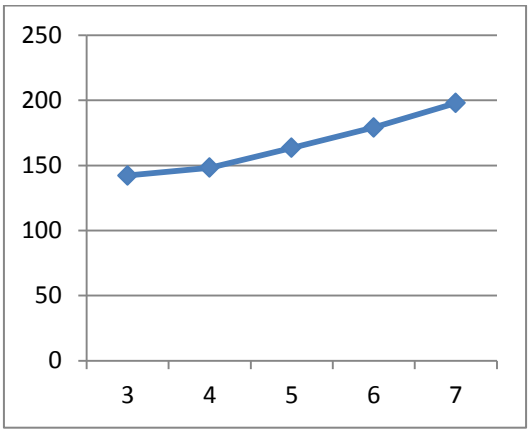

Abbildung 7.10. E/A-Aufwand (MB) der Methode `processSkylineQuery()` für Benutzer- und Buchdimensionen

In Abbildung 7.9 wird der E/A-Aufwand der Methode `processSkylineQuery()` für Buchdimensionen, wenn der Benutzer nicht eingeloggt ist, und 271065 Datensätze, dargestellt. Ebenso stellt die Abbildung 7.10 den E/A-Aufwand der Methode für Benutzer- und Buchdimensionen, wenn der Benutzer eingeloggt ist, und 285437 Datensätze, dar. In beiden Fälle kann man beobachten, dass der E/A-Aufwand linear mit der Anzahl an Dimensionen steigt und im schlimmsten Fall fast 200 MB erreicht.

Wie bereits gesagt wurde, sind Titel, Autor und Bewertung für eine Suche die am häufigsten verwendeten Dimensionen. Folglich werden die Ausführungen für „title“, „title-author“ und „title-author-rating“, wenn der Benutzer nicht eingeloggt ist, und die Ausführungen für „title-location-age“, „title-author-location-age“ und „title-author-rating-location-age“, wenn der Benutzer eingeloggt ist, die am häufigsten auftretenden Fälle sein.

Das Skyline Query Processing versucht Punkte zu finden, die am besten die Suchkriterien erfüllen. Von daher hängt die Anzahl der Skyline Punkte von der Anzahl der Dimensionen ab. Je mehr Dimensionen desto mehr Skyline Punkte, da die Punkte mehr Suchkriterien erfüllen sollen. Die Abbildung 7.11 stellt die Anzahl an Skyline Punkten, die in der Ausführung der Skyline Query für eine bis fünf Buchdimensionen und 271065 Datensätze erhalten wurden, dar. Genauso stellt die Abbildung 7.12 die Anzahl an Skyline Punkten, die in der Ausführung der Skyline Query für drei bis sieben Benutzer- und Buchdimensionen und 285437 Datensätze erhalten wurden, dar.

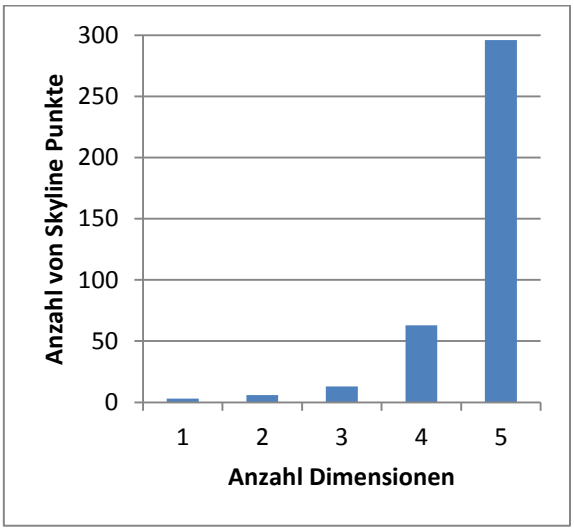

Abbildung 7.11. Anzahl an Skyline Punkten für Buchdimensionen

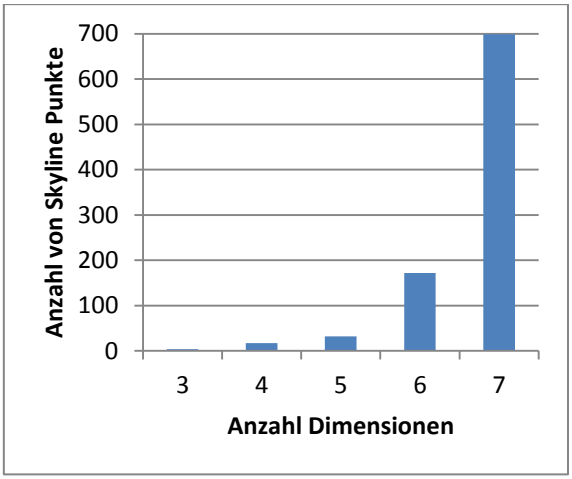

Abbildung 7.12. Anzahl an Skyline Punkten für Benutzer- und Buchdimensionen

Wie in Abbildungen 7.11 und 7.12 zu sehen ist, steigt die Anzahl der ermittelten Skyline Punkte mit der Anzahl an Dimensionen exponentiell. Unter Berücksichtigung, dass die Skyline Punkte die Empfehlungen des Systems sein werden, und das der Benutzer aus der Empfehlungsliste Elemente auswählt, kann man feststellen, dass die Skyline Query Verarbeitung nicht optimal für hohe Dimensionalität geeignet ist, da der Benutzer zu viele Elemente zur Auswahl erhält.

7.4 Leistung des Empfehlungssystems

Die Leistung des Empfehlungssystems wird durch die Leistung seiner Komponenten bestimmen.

Tabelle 7.4 stellt die Laufzeiten für das Ähnlichkeitsbewertungs Modul und das Skyline Query Processing Modul, für die am häufigsten verwendeten Ausführungsfälle und für die gesamten Datensätze des Book-Crossing Datasets, dar. Wie zu sehen ist, erreicht die Laufzeit einer Suche in der Testumgebung zwischen 113 bis 267 Sekunden insgesamt. Wenn der Benutzer nicht eingeloggt ist, wird, da keine Benutzerdaten verfügbar sind, nur die Anfrage nach „Passenden Büchern“ realisiert. Im umgekehrten Fall werden die Anfragen nach „Passenden Büchern“, „Bewerteten Büchern in der Vergangenheit“ und „Bewerteten Büchern von anderen Benutzern“ realisiert. Für die zweite Anfrage werden die Laufzeiten für 10000 Datensätze genommen, da die Datensätze nur dem aktuellen Benutzer gehören und die maximale Anzahl an bewerteten Büchern für einen Benutzer im Book-Crossing Dataset ungefähr 10000 ist. Die Methoden des Ähnlichkeitsbewertungs Moduls werden nur einmal während der Ausführung einer Suche ausgeführt, unabhängig davon, ob eine oder mehrere Anfragen verarbeiten werden müssen.

	Dimensionen	Ähnlichkeitsbewertungs Modul		Skyline Query Processing Modul			Gesamt (s)
		Compute User Similarity (s)	Compute Book Similarity (s)	Anfrage 1 (s)	Anfrage 2 (s)	Anfrage 3 (s)	
Ohne Login	title	0	106,89	6,523	0	0	**113,413**
	title-author	0	164,712	9,193	0	0	**173,905**
	title-author-rating	0	148,459	18,77	0	0	**167,229**
Mit Login	title-location-age	17,312	106,89	6,523	4,728	36,318	**171,771**
	title-author-location-age	17,312	164,712	9,193	5,476	44,226	**240,919**
	title-author-rating-location-age	17,312	148,459	18,77	10,844	72,509	**267,894**
Anfrage 1: Passende Bücher Anfrage 2: Bewertete Bücher in der Vergangenheit Anfrage 3: Bewertete Bücher von anderen Benutzern							

Tabelle 7.4. Laufzeit der häufigsten Ausführungsfälle einer Suche

Wie oben beschrieben kann die Laufzeit für große Datenmengen und hohe Dimensionalität sehr hoch sein. Der Grund dafür ist der hohe E/A-Aufwand, den die Ausführung der Methoden generiert. Tabelle 7.5 stellt die erreichten E/A-Werte für die häufigsten Ausführungsfälle, wobei der Gesamt-Wert zwischen 293 und 334 MB liegt, dar.

	Dimensionen	Ähnlichkeitsbewertungs Modul		Skyline Query Processing Modul (MB)	Gesamt (MB)
		Compute User Similarity (MB)	Compute Book Similarity (MB)		
Ohne Login	title	0	153,72	139,94	**293,66**
	title-author	0	153,72	148,44	**302,16**
	title-author-rating	0	153,72	169,72	**323,44**
Mit Login	title-location-age	17,26	153,72	142,24	**313,22**
	title-author-location-age	17,26	153,72	148,16	**319,14**
	title-author-rating-location-age	17,26	153,72	163,5	**334,48**

Tabelle 7.5. E/A-Aufwand der häufigsten Ausführungsfälle einer Suche

7.5 Qualität der Empfehlungen

Um die Qualität der Empfehlungen zu bestimmen, werden die Ergebnisse einer Suche analysiert.

Angenommen wurden folgende Suchparameter:

- Titel: „selected poems“
- Autor: „Dickinson“

Die gewünschten Daten müssen den Suchparametern so ähnlich wie möglich sein. Die folgende Tabelle stellt die Suchergebnisse des Empfehlungssystems für die Buchdimensionen „Titel“ und „Autor“ dar.

ISBN	Titel	Autor	Pub. Jahr	Verlag	Ø Bewertung
0812523385	Selected Poems of Emily Dickinson	Emily Dickinson	1995	Tor Books	7.0
1550651498	Selected Poems	Ralph Gustafson	2001	Vehicule Press	8.0
051709326X	Alfred, Lord Tennyson: Selected Poems	Alfred Tennyson, Baron Tennyson	1993	Random House Value Publishing	10.0
0060188065	Everest : Triumph and Tragedy on the World's Highest Peak	Matt Dickinson	2002	HarperResource	7.3
0786709456	Goodnight, Sweet Prince	David Dickinson	2002	Carroll & Graf Publishers	8.0
0806520922	Scarlett Slept Here : A Book Lover's Guide to the South	Joy Dickinson	2001	Citadel Press	10.0
0385750250	The Cup of the World	JOHN DICKINSON	2004	David Fickling Books	7.0

Tabelle 7.6. Erhaltene Ergebnisse der Testsuche

Wie zu sehen ist, erfüllt der erste Datensatz der Tabelle 7.6 beide Suchkriterien am besten. Die nächsten zwei Datensätze erfüllen den Titel und teilweise den Autor (z.B. haben „Dickinson“ und „Gustafson“ ähnliche Endungen). Und schließlich erfüllen die letzten vier Datensätze das Autoren Suchkriterium. Das Suchergebnis enthält nicht nur die Datensätze, die am besten alle Suchkriterien gleichzeitig erfüllen, sondern auch die Datensätze, die nur einen Teil der Suchkriterien erfüllen und auch interessant für den Benutzer sein können.

Die Ergebnisse der Testsuche erfüllen alle Kriterien der Skyline Query Verarbeitung (s. 2.1), allerdings gründet im nicht semantischen Vergleich der textuellen Daten für die Dimension Titel eine Begrenzung des Systems, die die Qualität der Ergebnisse beeinträchtigt.

7.6 Vergleich mit anderen Ansätzen

In diesem Abschnitt wird der Anwendungsprototyp mit anderen Ansätzen vergleichen. Da kein anderer Ansatz mit allen Eigenschaften des Anwendungsprototyps gefunden wurde, werden einzelne Aspekte der Anwendung analysiert.

7.6.1 Skyframe

Skyframe implementiert zwei Algorithmen, Greedy Skyline Search (GSS) und Relaxed Skyline Search (RSS), für die Verarbeitung von Skyline Queries in den P2P Systemen CAN

und BATON, und analysiert die Leistung der Algorithmen in dieser Umgebung. Skyframe verwendet für diese Evaluierung synthetisch erzeugte Datasets mit Normal- und Antikorrelationsverteilung der Punkte für 1638400 Datensätze. Skyframe wird in einem Cluster aus 20 Knoten ausgeführt, wobei jeder Knoten ein Intel Xeon 3.0Ghz Prozessor und 4GB RAM hat [WANG, 2009].

Da der Algorithmus für die Skyline Query Verarbeitung dieser Masterarbeit auf dem GSS Algorithmus basiert und die Daten zu einer realen Datenbank gehören, wird die Evaluierung der Leistung und zwar der Laufzeit des GSS Algorithmus von Skyframe für das normalverteilte Dataset analysiert. Diese Informationen werden in Tabelle 7.7 dargestellt.

	BATON (s)	**CAN (s)**
2 Dimensionen	0,43	0,51
3 Dimensionen	0,58	0,53
4 Dimensionen	0,63	0,52
5 Dimensionen	1,04	0,54

Tabelle 7.7. Laufzeiten des GSS Algorithmus in Skyframe für ein normalverteiltes Dataset [WANG, 2009]

Ein genauer Vergleich zwischen beide Algorithmen ist nicht möglich, da die Testumgebungen verschieden sind. Allerdings können die Leistungen beider Ansätze nach Anzahl an Dimensionen analysiert werden.

Wie zu sehen ist, steigt die Laufzeit bei Skyframe beim CAN System mit der Einbeziehung von mehreren Dimensionen kaum an. Anderseits, im Fall des BATON Systems, steigt die Laufzeit mäßig bis vier Dimensionen, aber ab fünf Dimensionen wird die Laufzeit um ca. 65% erhöht. Davon kann man eine exponentielle Steigerung der Laufzeit für mehr als fünf Dimensionen vermuten.

Wie oben erwähnt steigt die Laufzeit des Algorithmus dieser Masterarbeit exponentiell und präsentiert eine deutliche Zunahme der Laufzeit ab vier Dimensionen, wenn der Benutzer nicht eingeloggt ist, und ab 6 Dimensionen, wenn er eingeloggt ist. Von daher kann man eine Ähnlichkeit zwischen dem GSS Algorithmus für BATON mit dem normalverteilten Dataset und dem Algorithmus dieser Masterarbeit bezüglich des Zeitaufwands finden.

7.6.2 Bücher-Suchmaschinen

Um einen Vergleich zwischen den Ergebnissen des Anwendungsprototyps und den Empfehlungen bekannter Bücher-Suchmaschinen zu realisieren, wurden vier Bücher-

Suchmaschinen analysiert. Amazon.de, buecher.de, books.google.de und bookcrossing.com. Es wurde festgestellt, dass ein genauer Vergleich nicht möglich ist.

Ein Grund, der diesen Vergleich behindert, ist das Dataset, das jede Suchmaschine verwaltet. Obwohl es viele ähnliche Datensätze gibt, ist das ganze Dataset nicht gleich. Sogar bookcrossing.com verwaltet ein anderes Dataset, da der Anwendungsprototyp Daten von bookcrossing.com aus den Jahren vor September 2004 verwendet.

Ein zweiter Grund ist, dass die Bücher-Suchmaschinen andere Arten von Empfehlungen liefern. Normalerweise empfehlen die Suchmaschinen die am meisten gesuchten Bücher oder Bücher bzw. Produkte im Allgemeinen[7], die vorher vom aktuellen Benutzer schon gesucht bzw. gekauft wurden, obwohl sie nicht im Zusammenhang mit dem Suchelement stehen.

Ein Vergleich der Ergebnisse des Anwendungsprototyps mit den Ergebnissen der Bücher-Suchmaschinen ist nicht möglich. Der Grund liegt in dem Prinzip der Skyline Queries. Eine Suche nach Skyline Punkten liefert die Werte, die am ähnlichsten zu den eingegeben Werten sind. Andererseits gibt die Suche der analysierten Bücher-Suchmaschinen ähnliche aber nicht nur die ähnlichsten Werte zurück. Tabelle 7.8 stellt drei Buchtitel und ihre entsprechenden Ähnlichkeitswerte, die vom Ähnlichkeitsbewertungs Modul geliefert wurden, dar. Die Skyline Suche wird nur den ersten Titel zurückgeben, da er die anderen Titel dominiert. Anderseits würden die analysierten Suchmaschinen alle Titel zurückgeben, da sie die gesuchten Wörter beinhalten.

„best poems“	Ähnlichkeitswert
Best Remembered Poems	24,0
Best Loved Poems to Read Again and Again: The Most Moving Verses in the English Language	42,3
The Best Poems Ever: A Collection of Poetry's Greatest Voices (Scholastic Classics)	38,2

Tabelle 7.8. Ähnliche Buchtitel zu „best poems“

Für den Fall, dass die Suche mehrere Dimensionen verwendet, werden die analysierten Bücher-Suchmaschinen die Bücher, die alle Suchparameter genau erfüllen, auswählen. Auf diese Weise wäre folgender Datensatz für die Suche nach Büchern mit dem Titel „best poems“ und Autor „Dickinson“ nicht im Ergebnis, obwohl er die Dimension Titel genau erfüllt.

7 Im Fall von Amazon.de.

ISBN	Titel	Autor
048627165X	Best Remembered Poems	Martin Gardner

Tabelle 7.9. Datensatz der Tabelle „bx-books“

7.7 Anforderungsabgleich

Anschließend werden die funktionalen und nicht-funktionalen Anforderungen, die in Kapitel 4 definiert wurden, analysiert.

Funktionale Anforderung	Konzeption	Realisierung
FA1: Das System soll ermöglichen, dass der Benutzer sich in das System einloggen kann.	Ja. Komponente Benutzerverwaltung	Ja. Klasse CurrentUser
FA2: Die verfügbaren Suchparameter sollen den Eigenschaften des Buches wie Titel, Autor, Publikationsjahr, Verlag, sowie der durchschnittlichen Bewertung des Buches entsprechen.	Ja. Komponente Benutzer Schnittstelle	Ja. Klasse BookRecommender_ GUI
FA3: Nachdem der Benutzer Werte für die Suchparameter eingegeben hat, soll das System eine Empfehlungssuche von Büchern in Bezug auf die Suchparameter durchführen.	Ja. Komponente Benutzer Schnittstelle	Ja. Klasse BookRecommender_ GUI
FA4: Das System soll eine Empfehlungssuche nach „Passenden Büchern“ realisieren. Die Suche wird unter Berücksichtigung der Informationen der Bücher realisiert.	Ja. Komponente Recommender Modul	Ja. Klasse BookRecommender_ GUI
FA5: Das System soll eine Empfehlungssuche nach „Bewerteten Büchern in der Vergangenheit“ realisieren. Die Suche wird unter Berücksichtigung der Informationen der Bücher und des aktuellen Benutzers realisiert.	Ja. Komponente Recommender Modul	Ja. Klasse BookRecommender_ GUI
FA6: Das System soll eine Empfehlungssuche nach „Bewerteten Büchern von anderen Benutzern“ realisieren. Die Suche wird unter Berücksichtigung der Informationen der Bücher, des aktuellen Benutzers sowie	Ja. Komponente Recommender Modul	Ja. Klasse BookRecommender_ GUI

anderer Benutzern realisiert.		
FA7: Das System soll einen Wert für die Ähnlichkeit zwischen den Daten des Datasets und den vom Benutzer eingegebenen Suchparametern berechnen.	Ja. Komponente Ähnlichkeitsbe-wertungs Modul	Ja. Klasse DimensionSet
FA8: Das System soll die Suchanfragen mittels der Skyline Query Verarbeitung lösen.	Ja. Komponente Skyline Query Processing Modul	Ja. Klasse Skyline Query
FA9: Das System soll eine Ergebnisliste der Empfehlungssuchen erstellen und dem Benutzer präsentieren.	Ja. Komponente Recommender Modul	Ja. Klasse Recommendation

Tabelle 7.10. Abgleich der funktionalen Anforderungen

Tabelle 7.10 stellt die funktionalen Anforderungen (s.4.2) des Systems dar, sowie ihre Umsetzung in der Konzeptions- und Realisierungsphase. Die Erfüllung der funktionalen Anforderungen validiert die Funktionalität des Systems.

Anschließend wird eine Analyse der technischen Anforderungen realisiert.

- **TA1:** Die Berechnung der Ähnlichkeit der vom Benutzer eingegebenen Werte und die Daten des Datasets für eine Anfrage mit maximal drei Dimensionen soll nicht mehr als 15 Sekunden dauern.

  Die Berechnung der Ähnlichkeit wurde in zwei Teilen realisiert. Der erste Teil realisiert die Berechnung der Ähnlichkeit von Benutzerdaten und der zweite Teil der Ähnlichkeit von Bücherdaten. Nach einer Analyse der Laufzeit beider Methoden wurde festgestellt, dass nur weniger als 50000 Datensätze innerhalb der gewünschten Zeit berechnet werden können. Auch wenn der Benutzer nicht eingeloggt ist und die Berechnung der Ähnlichkeit von Benutzerdaten nicht stattfindet.

- **TA2:** Die Verarbeitung einer Skyline Query mit maximal drei Dimensionen soll nicht mehr als 15 Sekunden dauern.

  Um diese Anforderung zu erfüllen, werden zwei Fälle berücksichtigt. Im ersten Fall ist der Benutzer nicht eingeloggt. Im zweiten Fall ist er eingeloggt. Der Unterschied besteht darin, dass die Skyline Query für den ersten Fall nur die Daten der Tabelle „d_users“ verarbeiten muss und für den zweiten Fall eine Join-Operation zwischen den Tabellen „d_books“, „bx-book-ratings“ und „d_users“ realisiert werden muss.

Um die gewünschte Laufzeit zu erreichen muss das Verfahren für Fall eins mit einer Anzahl Datensätzen zwischen 200000 und 250000, und für Fall zwei zwischen 50000 und 100000 Datensätzen durchgeführt werden

- **TA3:** Die Funktionsweise des Systems darf sich auch unter Betrachtung von mehr als 3 Dimensionen nicht ändern.

  Die Ausführung der Anwendung wurde mit ein bis sieben Dimensionen und einer Datenmenge von 1000 bis 285437 Datensätze getestet. Für alle Testfälle wurde die Funktionalität der Anwendung validiert.

- **TA4:** Die Ergebnisse müssen zu mindestens 50 % mit Ergebnissen anderer Bücher-Empfehlungssysteme übereinstimmen.

  Ein Vergleich zwischen dem Ergebnis des Anwendungsprototyps und anderen Bücher-Empfehlungssystemen war, wie im Abschnitt 7.6.2 erklärt wurde, nicht möglich. Allerdings wurden einige Elemente des Ergebnisses der Anfrage „Passende Bücher“ in der Ergebnisliste der analysierten Bücher-Suchmaschinen gefunden. Die gefundenen Elemente ergeben 10% des gesamt Ergebnisses.

In der Realisierungsphase wurde ständig versucht, die Laufzeit zu optimieren. Berechnete Felder, die durch Trigger nach der Aktualisierung eines Datensatzes erstellt wurden, wurden verwendet, um Berechnungen während der Ausführung der Anwendung zu vermeiden. Die Schlüsselfelder der Tabellen „d_books“, „d_users“ und „bx-book-ratings“ wurden indiziert, um die Kombination der Tabellen zu optimieren. Außerdem wurden temporäre Tabellen erstellt, in denen nur die nötigen Datensätze für eine Anfrage gespeichert sein sollen, um den unnötigen Zugriff auf andere Daten zu vermeiden. Trotz der oben genannten Aktionen wird eine hohe Laufzeit in der Testumgebung erreicht, wenn es um hohe Dimensionalität und große Datenmengen geht.

7.8 Kritische Betrachtung

Normalerweise ist ein Suchelement wie ein Buch durch verschiedene Attribute beschrieben. Von daher muss ein Empfehlungssystem die Benutzerpräferenzen berücksichtigen bzw. sollte die Suchmethode in der Lage sein, multidimensionale Anfragen zu verarbeiten. Die Verarbeitung von Skyline Queries ist für multikriterielle Entscheidungsunterstützung geeignet, folglich kann ein Skyline Query Verfahren in der Suchmethode eines Empfehlungssystems angewendet werden.

Die Ergebnisse des Anwendungsprototyps sind kein Ersatz für die Suchergebnisse einer Bücher-Suchmaschine. Grund dafür ist, dass ein Benutzer häufig nur Stichwörter, also Wort-

Teile des Titels, eingibt, vielmehr aber den vollständigen Titel erwartet. Wenn der Benutzer Bücher mit einem bestimmten Titel sucht, ist der Skyline Query Ansatz geeignet. Folglich ist angemessen, dass das Ergebnis einer Bücher-Suchmaschine durch die Empfehlungen des Anwendungsprototyps ergänzt wird.

Die Verwendung des Book-Crossing Datasets ermöglicht die Beobachtung der Funktionalität der Anwendung mit realen Daten, um so Vorteile und Begrenzungen des Systems aufzufinden. Da die Skyline Punkte durch Punkte in einem n-dimensionalen Raum repräsentiert sind, kann die Qualität der Ergebnisse besser analysiert werden, wenn diese Punkte durch reale Daten dargestellt werden. Anderseits ermöglicht die Anzahl der Datensätze der Tabellen die Grenzen des Systems zu bestimmen. Schließlich zeigt die Verwendung der Book-Crossing Daten, dass der Skyline Query Ansatz auf reale Systeme anwendbar ist.

8 Zusammenfassung und Ausblick

In diesem Kapitel wird die ganze Arbeit zusammengefasst mit Betonung der wichtigsten Merkmale der Arbeit. Anschließen wird ein Ausblick gegeben, bezüglich möglicher Erweiterungen und Fortführungen dieser Arbeit.

8.1 Zusammenfassung

In dieser Arbeit wurden die Eigenschaften der Skyline Queries adressiert, indem sie in einem Empfehlungssystem Anwendung finden. Zu diesem Zweck wurde ein Anwendungsprototyp implementiert. Das Ziel des Anwendungsprototyps ist, eine Suche nach Büchern bezüglich verschiedener Suchparameter durch die Verarbeitung von Skyline Queries zu realisieren. Die Suche wird durch eine Skyline Query repräsentiert und die Ergebnisse bzw. die erhaltene Skyline Punkte entsprechen den Büchern, die dem Benutzer als Empfehlungen geliefert werden. Der Anwendungsprototyp verwendet das Book-Crossing Dataset, das reale Daten aus der Book-Crossing Comunity beinhaltet. Die Book-Crossing Comunity ermöglicht den Austausch von Büchern unter ihren Benutzern.

Um ein gutes Verständnis des Problems zu gewinnen wurden im ersten Schritt grundlegende Konzepte von Skyline Queries und Empfehlungssystemen aufgezeigt. Angesichts des theoretischen Wissens und der verfügbaren Datenbasis wurde die Problemstellung anschließend genauer beschrieben und dann das Szenario des Systems definiert.

Als Nächstes wurde eine Analyse des Systems durchgeführt, in dem die funktionalen und technischen Anforderungen, sowie deren Anwendungsfälle, identifiziert wurden. Von hier

aus wurde ein Entwurf der Architektur des Systems erstellt und seine Bestandteile bestimmt. Nachfolgend wurde die Konzeptionsphase gestartet. In dieser Phase wurden die Entwurfsentscheidungen jedes Bestandteils der Architektur definiert. Wobei die genannten Komponenten „ Ähnlichkeitsbewertungs Modul“ und „Skyline Query Processing Modul“ die wichtigsten sind.

Die Daten, die von einer Skyline Query verwendet werden, müssen numerisch sein und in einem n-dimensionalen Raum repräsentiert werden. Von daher muss, um eine Skyline Query zu verarbeiten, das ganze Dataset in einen n-dimensionalen Raum umgewandelt werden, wobei jeder Datensatz durch einen Datenpunkt in dem n-dimensionalen Raum repräsentiert wird und die Datenfelder die Dimensionen des Raumes sind. Um die Daten in den n-dimensionalen Raum umzuwandeln, wurde das „Ähnlichkeitsbewertungs Modul“ implementiert. Es realisiert die Berechnung zwischen den durch den Benutzer eingegebenen Suchparametern und den Werten jedes Datenfeldes. Auf diese Weise, bekommt jedes Datenfeld einen numerischen Wert, der dessen Ähnlichkeit repräsentiert. Je kleiner der Wert ist, desto ähnlicher sind die Werte. Um die Ähnlichkeit zwischen numerischen Daten zu berechnen, wurden arithmetische Operationen realisiert. Anderseits wurde, um die Ähnlichkeit zwischen textuellen Daten zu bestimmen, eine Methode, die auf dem Trigramm-Algorithmus für den Vergleich von Zeichenketten basiert, implementiert.

Das Skyline Query Processing Modul implementiert einen Algorithmus, um Skyline Punkte zu finden, der auf dem Nearest Neighbor Algorithmus (s. 2.1.1) und Skyframe [WANG, 2009] basiert. Für den Anwendungsprototyp wurden drei verschiedene Suchen realisiert „Passende Bücher“, „Bewertete Bücher in der Vergangenheit“ und „Bewertete Bücher von anderen Benutzern“. Jede Suche generiert eine andere Skyline Query mit verschiedenen Dimensionen. Dieses Modul verwendet die numerischen Ähnlichkeitswerte, von denen einige die Skyline Punkte sein werden. Jeder Skyline Punkt entspricht einem Buch Datensatz, der in der Empfehlungsliste präsentiert wird.

Sowohl das Ähnlichkeitsbewertungs Modul als auch das Skyline Query Processing Modul, sowie die anderen Komponenten des Systems, wurden in der Realisierungsphase in einen Anwendungsprototyp implementiert. Die Anwendung wurde in Java implementiert und die Datenbank wird mit dem DBMS MySQL 5.5 gesteuert. Nach der Implementierung wurde die Anwendung getestet, um ihre Funktionalität zu validieren.

Anschließend wurde die Evaluierung des Anwendungsprototyps realisiert. Als Erstes wurde die Leistung des Ähnlichkeitsbewertungs Moduls analysiert und festgestellt, dass die Laufzeit linear mit der Anzahl der Datensätze steigt. Außerdem wurde festgestellt, dass die Berechnung der Ähnlichkeit textueller Daten die meiste Zeit, ca. 88% im Fall von Benutzerdimensionen und ca. 99% im Fall von Bücherdimensionen, in Anspruch nimmt. Im Folgenden wurde eine Analyse der Leistung des Skyline Query Processing Moduls realisiert und festgestellt, dass die Laufzeit mit der Anzahl der Datensätze linear steigt, aber

exponentiell mit der Anzahl der Dimensionen. Darüber hinaus kann man eine Steigerung der Laufzeit, wenn der Benutzer eingeloggt ist, beobachten, da in diesem Fall eine Join-Operation realisiert werden muss. Anhand der Evaluierung der Komponente „Ähnlichkeitsbewertungs Modul“ und „Skyline Query Processing Modul“ wurde dann die Leistung der gesamten Anwendung berechnet. Angesichts der hohen Laufzeit für beide Komponenten wurden die Leistungen des Prozessors, des Arbeitsspeichers und des physikalischen Datenträgers analysiert. Dabei wurde entdeckt, dass die Disk-Zugriffszeit die Laufzeit der Methoden bestimmt. Da die Prozessorzeit nur 12.5% für die Ausführung des Ähnlichkeitsbewertungs Moduls und zwischen 19% und 22% in der Ausführung des Skyline Query Processing Moduls repräsentiert. Die restliche Zeit wird für E/A-Operationen verwendet.

Folgend wurde die Qualität der Empfehlungen analysiert und festgestellt, dass die Skyline Query Verarbeitung richtig funktioniert, dass allerdings der nicht semantische Vergleich der textuellen Daten für die Buchtiteln, die Qualität der Ergebnisse beeinträchtigt.

Als Nächstes wurden andere Ansätze analysiert, um einen Vergleich mit dem Anwendungsprototyp durchzuführen. Dafür wurden die Eigenschaften des Frameworks Skyframe, sowie die Eigenschaften der bekannten Bücher-Suchmaschinen berücksichtigt. Ein genauer Vergleich mit Skyframe war nicht möglich, da es eine andere Umgebung als die von dem Anwendungsprototyp verwendet. Allerdings konnte eine Ähnlichkeit zwischen dem Verhalten des Zeitaufwands von Skyframe und dem Anwendungsprototyp, trotz seiner hohen Laufzeit, gefunden werden. Ein genauer Vergleich mit den Suchergebnissen bzw. Empfehlungen der analysierten Bücher-Suchmaschinen war auch nicht möglich, weil sie verschiedene Datenbanken und verschiedene Suchprozesse verwenden. Dennoch konnte ein Anteil von 10% von gemeinsamen Ergebnissen gefunden werden.

Anschließend wurde ein Anforderungsabgleich bezüglich der in Kapitel 4 definierten funktionalen und technischen Anforderungen des Systems realisiert. Es wurde festgestellt, dass der Anwendungsprototyp die funktionalen Anforderungen richtig erfüllt, jedoch die technischen Anforderungen nur teilweise. Der Grund liegt an der hohen Laufzeit für große Datenmengen und hoher Dimensionalität.

Die Realisierung dieser Arbeit zeigt, dass der Skyline Query Ansatz auf reale Systeme anwendbar ist. Darüber hinaus ermöglicht die Verwendung realer Daten eine bessere Analyse der Funktionalität und Leistung des Systems. Ultimativ kann festgehalten werden, dass aufgrund des unterschiedlichen Suchprinzips die Ergebnisse einer Skyline Query die Ergebnisse einer Bücher-Suchmaschine nicht ersetzen, sie aber sehr wohl ergänzen können.

8.2 Ausblick

In dieser Arbeit wurde bewiesen, dass das Skyline Query Verfahren in einem Bücher Empfehlungssystem anwendet werden kann. Allerdings können Erweiterungen gemacht werden, um die Leistung und Qualität des Systems zu verbessern.

In Bezug auf das Ähnlichkeitsbewertungs Modul, werden ein semantischer Vergleich der textuellen Daten und die Verwendung eines Thesaurus für die Ähnlichkeitsbewertung der Buchtitel empfohlen. Auf diese Weise könnten Titeln wie „best poems“, „the best poems“ und „selected poetries“ die gleichen Ähnlichkeitswerte im Vergleich mit „best poems“ bekommen. Diese Erweiterung würde die Qualität der Ergebnisse des Systems erhöhen.

In Bezug auf das Skyline Query Processing Modul, wird die Verwendung von Benchmark-Daten empfohlen, um den Algorithmus für die Verarbeitung von Skyline Queries dieser Masterarbeit weiter zu evaluieren und einen genaueren Vergleich mit anderen Skyline Query Ansätzen zu realisieren.

Schließlich wird empfohlen die Anwendung in einer technisch besseren Umgebung als der hier verwendeten Testumgebung auszuführen, um die Leistung der Methoden zu verbessern.

A Inhalt der CD-ROM

Dieser Masterarbeit liegt eine CD-ROM mit folgender Verzeichnisstruktur bei:

- [Ausarbeitung] beinhaltet die Masterarbeit im PDF-Format.
- [Literatur] beinhaltet die verwendete Literatur.
- [Quellcode] beinhaltet den Quellcode des implementierten Anwendungsprototyps.
- [Datenbank] beinhaltet MySQL-Dump Dateien, die die Struktur der Tabellen, Trigger, gespeicherte Prozeduren, Funktionen sowie die Datensätze der Tabellen „bx-books“, „bx-users“ und „bx-book-ratings“ beinhaltet.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[ADAMS, 1993] E. S. Adams, A. C. Meltzer. *Trigrams as index elements in full Text Retrieval. Observations and experimental results*. In ACM 0-89791-558-5/93/0200/0433, 1993.

[BÖRZSÖNYI, 2001] S. Börzsönyi, D. Kossmann, K. Stocker. *The Skyline Operator*. In Proceedings of the 17th International Conference on Data Engineering IEEE Computer Society Washington, DC, USA, 2001.

[CELMA, 2010] O. Celma. *Music Recommendation and Discovery. The Long tail, Long Fail, and Long Play in the Digital Music Space*. Springer, 2010.

[CHOMICKI, 2003] J. Chomicki, P. Godfrey, J. Gryz, D. Liang. *Skyline with presorting*. In Proceedings of ICDE, 2003.

[CUSI, 2010a] J. Cusi Juarez. *P2P Datenbanken*. Ausarbeitung im Rahmen der Vorlesung Anwendungen 1 im Studiengang Informatik Master of Science am Studiendepartment Informatik der Fakultät. Technik und Informatik der Hochschule für Angewandte Wissenschaften Hamburg. Februar 2010.

[CUSI, 2010b] J. Cusi Juarez. *Skyline Query Processing*. Ausarbeitung im Rahmen der Vorlesung Anwendungen 2 im Studiengang Informatik Master of Science am Studiendepartment Informatik der Fakultät. Technik und Informatik der Hochschule für Angewandte Wissenschaften Hamburg. August 2010.

[CUSI, 2011] J. Cusi Juarez. *Skyline Query Processing in P2P Netze.* Ausarbeitung im Rahmen der Vorlesung Projekt 2 im Studiengang Informatik Master of Science am Studiendepartment Informatik der Fakultät. Technik und Informatik der Hochschule für Angewandte Wissenschaften Hamburg. Februar 2011.

[FELFERNIG, 2008] A. Felfernig, R. Burke. *Constraint-based Recommender Systems: Technologies and Research Issues*. In ACM 10th Int. Conf. on Electronic Commerce (ICEC) Innsbruck, Austria, 2008.

[JOSWIG, 2008] M. Joswig, T. Theobald. *Algorithmische Geometrie. Polyedrische und algebraische Methoden.* Vieweg, Wiesbaden, 2008.

[KHALEFA, 2008] M. E. Khalefa, M. F. Mokbel, J. J. Levandoski. *Skyline Query Processing for Incomplete Data*. In Proceedings of the 2008 IEEE 24th International Conference on Data Engineering IEEE Computer Society Washington, DC, USA 2008.

[KLAHOLD, 2009] A. Klahold. *Empfehlungssysteme. Recommender Systems – Grundlagen, Konzepte und Lösungen*. Vieweg+Teubner Verlag / GWV Fachverlage GmbH, Wiesbaden, 2009.

[KODAMA, 2009] K. Kodama, Y. Iijima, X. Guo, Y. Ishikawa. *Skyline Query Based on User Locations and Preferences for making Location-Based Recommendations*. In Proceedings of the 2009 International Workshop on Location Based Social Networks ACM New York, NY, USA 2009.

[KOSSMANN, 2002] D. Kossmann, F. Ramsak, S. Rost. *Shooting stars in the sky: an online algorithm for skyline queries*. In Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002.

[LEE, 2007] J. Lee, G. You, I. Sohn, S. Hwang, K. Ko, Z. Lee. *Supporting Personalized Top-k Skyline Queries using Partial Compressed Skycube*. In ACM WIDM‘07, 2007.

[LINDEN, 2003] G. Linden, B. Smith, J. York. *Amazon.com Recommendations. Item-to-Item Collaborative Filtering*. In IEEE Internet Computing, 2003.

[LUO, 2004] Y. Luo. *Multi-Dimensional Skyline Query Processing*. Dissertation submitted in fulfillment of the requirements for the degree of Master of Computer Science School of Computer Science and Engineering the University of New South Wales Sydney, Australia. August 2004.

[PAPADIAS, 2003] D. Papadias, Y. Tao, G. Fu, B. Seeger. *An Optimal and Progressive Algorithm for Skyline Queries*. In ACM SIGMOD 2003, June 9-12.

[PGFOUNDRY, 2009a] PostgreSQL Development Group - PGFOUNDRY. *Random Dataset Generator for SKYLINE Operator Evaluation Project*. http://randdataset.projects.postgresql.org/

[PGFOUNDRY, 2009b] PostgreSQL Development Group - PGFOUNDRY. *Implementation of Random Dataset Generator for SKYLINE Operator Evaluation Project.* http://skyline.dbai.tuwien.ac.at/

[RICCI, 2011] F. Ricci, L. Rokach, B. Shapira, P. B. Kantor. *Recommender System Handbook*. Springer, 2011.

[SHARIFZADEH, 2006] M. Sharifzadeh, C. Shahabi. *The Spatial Skyline Queries.* In Proceddings of the 32nd international conference on Very large data bases VLDB, 2006.

[TAN, 2001] K. L. Tan, P. K. Eng, B. C. W. Ooi. *Efficient progressive skyline computation.* In Proceddings of the 27th VLDB Conference, Roma, Italy, 2001.

[TENG, 2007] H. Lee, W. Teng. *Incorporating Multi-Criteria Ratings in Recommendation Systems.* In Information Reuse and Integration, 2007. IRI 2007. IEEE International Conference, USA, 2007

[WANG, 2009] S. Wang, Q. H. Vu, B. C. Ooi, A. K. H. Tung, L. Xu. *Skyframe: a framework for skyline query processing in peer-to-peer systems*. In VLDB Journal, Vol.18, No.1, 2009.

[XIA, 2006] T. Xia, D. Zhang. *Refreshing the sky: The compressing skycube with efficient support for frequent updates*. In SIGMOD, 2006.

[YANG, 2011] J. Yang, G. P. C. Fung, W. Lu, X. Zhou, H. Chen, X. Du. *Finding superior skyline points for multidimensional recommendation applications*. In World Wide Web Journal, 2011.

[YUAN, 2005] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, Q. Zhang. *Efficient Computation of the Skyline Cube*. In VLDB 2005, pages 241–252, 2005.

[ZIEGLER, 2005] C. Ziegler, S. M. McNee, J. A Konstan, G. Lausen. *Improving Recommendation Lists Through Topic Diversification*. In ACM 1595930469/05/0005, 2005.

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, den _____ _____