WebDBSearch – Eine Suchmaschine zum Auffinden relevanter Informationseinheiten in Web-Datenbanken

Gunnar Weber Andreas Heuer
{weber,heuer}@informatik.uni-rostock.de

Abstract: In Web-Datenbanken enthaltene Informationen können von aktuellen Suchmaschinen nur über einen aufwendigen Export der Inhalte in Hypertext-Dokumente gefunden werden. WebDBSearch ist ein System zur Suche nach Informationen in Web-Datenbanken auf der Basis von Web-Services, die einen geeigneten Zugriff auf die Datenbank gewährleisten. In einer Datenbank sind Informationseinheiten in der Regel über mehrere Tupel in verschiedenen Relationen verteilt. Ein Auffinden dieser Einheiten erfordert somit einen Verbund dieser Tupel "on the fly". Die auf der Grundlage der Datendefinition beschriebene Semantik der Datenbank ist für diese Art der Suche nicht ausreichend. Die fehlenden semantischen Informationen werden deshalb in einer zusätzlichen Ebene bereitgestellt, die als Basis für die Interpretation der Anfragen dient.

Zur Beantwortung einer Anfrage bestimmt WebDBSearch mit Hilfe lokaler Indexdaten zunächst alle Datenbanken, die relevante Informationen beinhalten. Anschließend erfolgt der Zugriff auf eine vom Nutzer selektierte Datenbank zur Bestimmung der relevanten Informationseinheiten auf der Grundlage von Anfragen, die aus den Indexdaten abgeleitet werden. Die Indexdaten umfassen nur die aktiven Domänen ausgewählter Attribute. Dieser Ansatz erlaubt eine ressourcenschonende Indexierung und berücksichtigt adäquat die Sicherheitsaspekte kommerzieller Datenbankanbieter.

1 Einleitung

Informationen, die in Web-Datenbanken enthalten sind, können im Allgemeinen nicht von den aktuellen Suchmaschinen gefunden werden. Diese Informationen werden damit zum Großteil ignoriert und deshalb häufig als "Hidden Web" bezeichnet. Zur Zeit existieren keine Protokolle, die es Suchmaschinen erlauben, auf die Inhalte einer Web-Datenbank zuzugreifen. Ein naiver Ansatz, der häufig genutzt wird, ist der Export der Inhalte von Web-Datenbanken in HTML-Dokumente. Ein wesentlicher Unterschied zwischen dem Dokumenten- und dem Datenbanken-Retrieval besteht aber in der Granularität der logischen Informationseinheiten. Während eine Informationseinheit beim Dokumenten-Retrieval aus einem Dokument besteht, wird diese bei relationalen Datenbanken aus einer Menge von Attribut-Wert-Paaren gebildet, die aus mehreren Tupeln verschiedener Relationen stammen können. Ein Export mehrerer Informationseinheiten in ein Dokument berücksichtigt daher nicht die eigentlichen logischen Einheiten. Auf der anderen Seite ist ein Export aller Informationseinheiten in separate HTML-Dokumente gerade bei umfangreichen Datenmengen viel zu aufwendig. Die Verwendung von XML als Austauschformat

kann zwar dieses Problem lösen, andere Probleme wie der hohe Aufwand für den Aufbau der Dokumente – alle in der Datenbank enthaltenen Informationseinheiten, die über mehrere Tupel in verschiedenen Tabellen verteilt sind, müssen generiert werden – und der große Umfang der dabei anfallenden Daten bleiben. Für viele Datenbankanbieter, die ihre Daten kommerziell verwenden, schließt sich außerdem eine komplette Duplizierung aus Sicherheitsgründen aus.

Web-Datenbanken haben typischerweise eine Anfrage-Schnittstelle in Form eines HTML-Formulars. Der Zugriff auf die Datenbankinhalte ist über dynamisch generierte Seiten möglich, die als Antwort auf die Nutzeranfrage, die im Formular angegeben ist, geliefert werden [RGM01]. Für jede Web-Datenbank lässt sich im Allgemeinen ein Gegenstand der realen Welt spezifizieren, der im Mittelpunkt der Anwendung steht. Dieser Gegenstand ist beispielsweise ein Verkaufsartikel bei einem Online-Shop oder ein Buch bei einer Bibliotheksdatenbank. Alle Informationen, die einem Gegenstand zugeordnet sind, bilden eine Informationseinheit. Als Ergebnis einer Anfrage über das Web-Formular wird eine Menge von relevanten Informationseinheiten – projiziert auf die wesentlichen Attribute – geliefert, aus der ein Nutzer eine konkrete Einheit zur Detaildarstellung auswählen kann. Die Navigation erfolgt im Allgemeinen auf der Basis eines eindeutigen Schlüssels für diese Informationseinheit.

WebDBSearch ist eine Datenbank-Suchmaschine, die einen geeigneten Zugriff auf die in Web-Datenbanken gespeicherten Informationen ohne Vorwissen über deren Struktur und Inhalte ermöglicht. Eine Anfrage an das System besteht aus einer Menge von Schlüsselwörtern, die in den gesuchten Informationseinheiten möglichst vollständig enthalten sein soll. Die Suche nach diesen Einheiten gestaltet sich als *dreistufiger Prozess*. In der ersten Stufe werden alle *Datenbanken* ermittelt, die relevante Informationseinheiten enthalten können. Eine geeignete Bewertung dieser Datenbanken auf der Basis klassischer Information-Retrieval-Techniken sorgt dafür, dass die Datenbanken, die mit einer höheren Wahrscheinlichkeit nützliche Informationen liefern, am Anfang der Ergebnisliste stehen. Bei der Ermittlung der relevanten Web-Datenbanken werden aus der Anfrage gleichzeitig ein oder mehrere plausible strukturierte Anfragen abgeleitet, die ebenfalls Einfluss auf die Bewertung einer Datenbank haben. Diese generierten Anfragen werden in der zweiten Stufe an die Web-Datenbank übermittelt, die der Nutzer aus der Liste der ermittelten Datenbanken ausgewählt hat. Als Ergebnis erhält der Nutzer eine bewertete Liste der relevanten *Informationseinheiten aus der gewählten Datenbank* – die Bewertung ist abhängig von der Anzahl der in der Informationseinheit enthaltenen Stichwörter aus der Originalanfrage. In der dritten Stufe erfolgt die *Detaildarstellung der gewünschten Informationseinheit* durch die Übergabe des zugehörigen Schlüssels an die Standard-Schnittstelle der Web-Datenbank.

Die Daten, die für eine Web-Datenbank indexiert werden, umfassen nur die aktiven Domänen bestimmter, vom Anbieter gewählter Datenbankattribute und keine kompletten Informationseinheiten. Dieser Ansatz ermöglicht somit eine *ressourcenschonende Indexierung* von Web-Datenbanken und vermeidet gleichzeitig die vollständige Veröffentlichung der Inhalte. Die Beantwortung der strukturierten Anfragen basiert auf Techniken für Universalrelationen-Schnittstellen, da durch die Normalisierung eines Schemas beim Datenbankentwurf logische Informationseinheiten im Allgemeinen fragmentiert und über

verschiedene Relationen verteilt sind. Die Ermittlung der zu einer gegebenen Menge von Schlüsselwörtern passenden Informationseinheiten erfordert somit den *Verbund verschiedener Relationen „on the fly"*.

WebDBSearch setzt kooperative Datenbankanbieter voraus, da für die Indexierung und die Anfragen Web-Services auf Anbieterseite vorausgesetzt werden. Durch die Bereitstellung fehlender semantischer Informationen in zusätzlichen Ebenen wird die Autonomie der eingebundenen Datenbanken nicht beeinflusst.

Der Artikel ist wie folgt gegliedert: Im zweiten Kapitel werden verwandte Arbeiten in den Bereichen Datenbank-Suchmaschinen und schlüsselwortbasierte Suche in Datenbanken vorgestellt. In Kapitel 3 erfolgt die Einführung eines geeigneten Datenmodells für die Suche in Web-Datenbanken. Kapitel 4 gibt einen kurzen Überblick über das System, die Details zu den einzelnen Phasen werden in den beiden nachfolgenden Abschnitten besprochen. Im siebten Kapitel wird kurz auf die Architektur des Systems eingegangen. Das letzte Kapitel fasst die Ergebnisse zusammen und gibt einen Ausblick auf geplante Arbeiten.

2 Verwandte Arbeiten

Die Datenbank-Suchmaschine *Jungle* [BBD99] nutzt JDBC-Verbindungen zu den Datenbanken zur Indexierung der Metadaten und der aktiven Domänen aller alphanumerischen Attribute sowie zur Berechnung von Anfragen. Die dabei genutzte Anfragesprache AQUA kombiniert die Suche nach passenden Attributwerten und Metadaten auf der Grundlage einer Stufenhierarchie ausgehend von der Datenbank bis hin zu den Werten innerhalb eines Attributs. AQUA-Anfragen auf Werte-Ebene werden unter Ausnutzung der Metadaten und der Indexinformationen in eine Menge von SQL-Anweisungen umgewandelt. Die Generierung dieser Anweisungen erfolgt für alle Datenbanken mit passenden Metadaten und Attributwerten. Der für die Indexierung und die Verarbeitung der Anfragen benötigte direkte Zugriff auf die Datenbank berücksichtigt jedoch nicht die Sicherheitsaspekte kommerzieller Datenbankanbieter. Ein weiteres Problem stellt der hohe Kommunikationsumfang und die starke Belastung der eingebundenen Datenbanken dar, da Anfragen im Allgemeinen immer durch das Weiterreichen einer Menge von SQL-Anfragen an eine Menge von Datenbanken beantwortet werden.

Im Folgenden werden verschiedene Ansätze zur schlüsselwortbasierten Suche in Datenbanken vorgestellt. Bei der Suche nach relevanten Informationen in einer Menge von Web-Datenbanken ausgehend von einer zentralen Suchmaschine müssen verschiedene Aspekte wie eine geeignete Datenbankauswahl und eine für den Zugriff über das Web adäquate Schnittstelle berücksichtigt werden, die bei der schlüsselwortbasierten Suche in einer Datenbank keine Rolle spielen.

Graphenbasierte Ansätze wie *Dataspot* [DEGP98] und *BANKS*[1] [BHNC02] basieren auf der Repräsentation der Datenbankinhalte in einem Graphen. Tupel bzw. deren Komponen-

[1] **B**rowsing **AN**d **K**eyword **S**earch

ten werden als Knoten im Graphen abgebildet, die Kanten repräsentieren die Beziehungen zwischen den Tupel verschiedener Relationen bzw. den Tupelkomponenten. Die Verbindungen zwischen den Tupeln aus verschiedenen Relationen werden über die definierten Fremdschlüssel abgeleitet. Gültige Antworten auf eine schlüsselwortbasierte Anfrage sind Teilgraphen, die verbunden sind und alle Schlüsselwörter in den Knoten enthalten. Die Einstiegspunkte für die Suche nach diesen Teilgraphen werden über einen Index ermittelt, der zu jedem Schlüsselwort die relevanten Knoten speichert. Der Nachteil der graphenbasierten Ansätze ist die Duplizierung der Daten durch den Aufbau großer Datengraphen, die bei Datenänderungen aktualisiert werden müssen.

Der *DBXplorer* [ACD02] findet für eine gegebene Menge von Schlüsselwörtern diejenigen Tupel aus einer Relation oder einem minimalen Verbund von Relationen, die alle Schlüsselwörter in beliebigen Attributen enthalten. Die im Vorverarbeitungsschritt aufgebaute Symboltabelle ist die wesentliche Datenstruktur zum Auffinden der Orte, an denen die Schlüsselwörter in der Datenbank vorkommen. Mit Hilfe dieser Ortsinformationen wird anschließend aus dem zuvor aufgebauten ungerichteten Schemagraphen S ein minimaler Teilgraph G ermittelt, der alle relevanten Relationen verbindet. G umfasst alle potentiellen Verbundbäume. Diese Verbundbäume sind Teilbäume von G, deren Blätter (i) Relationen mit passenden Attributwerten darstellen und (ii) zusammen alle Schlüsselwörter enthalten. Im nächsten Schritt erfolgt die Bestimmung aller Verbundbäume, die anschließend in SQL-Anfragen umgeformt werden. Die generierten Verbundbäume sind schemagebunden. Mögliche Beziehungen zwischen verschiedenen Tupeln derselben Relation werden daher nicht betrachtet.

Discover [HP02] basiert auf einem ähnlichen Ansatz wie der DBXplorer und liefert so genannte „Verbundnetzwerke von Tupeln". Jedes Netzwerk ist eine Menge von Tupeln, die über Fremdschlüssel in Beziehung stehen und zusammen alle Schlüsselwörter der Anfrage enthalten. Zur Verarbeitung einer Anfrage werden zunächst für alle Relationen diejenigen Tupel bestimmt, in denen ein Schlüsselwort als Teil eines beliebigen Attributwerts vorkommt. Anschließend erfolgt die Bestimmung der Kandidaten-Netzwerke – Verbundausdrücke zur Erzeugung der Verbundnetzwerke – auf der Basis des Schemagraphen und der ermittelten Tupelmengen. Die Kandidaten-Netzwerke sind nicht schemagebunden und berücksichtigen Beziehungen zwischen verschiedenen Tupeln derselben Relation. Die Semantik der sich in diesen Fällen ergebenden Verbundnetzwerke ist aber für den normalen Nutzer häufig schwer erkennbar. Die auf Tupelmengen basierende Anfragebearbeitung von Discover erfordert außerdem den Aufbau umfangreicher Indizes.

Discover und der DBXplorer betrachten keine zyklischen Datenbanken, die Beziehungen zwischen den einzelnen Relationen müssen durch Fremdschlüssel in der Datenbank definiert sein. In vielen Anwendungen werden solche Inklusionabhängigkeiten aber bereits in der Anwendung sichergestellt und sind deshalb in der Datenbank nicht spezifiziert. Die Forderung nach dem Enthaltensein aller Schlüsselwörter in einem Anfrageergebnis erlaubt zwar die Generierung effizient verarbeitbarer SQL-Anfragen, sie stellt aber für die Suche nach relevanten Informationseinheiten eine zu starke Einschränkung dar.

3 Datenmodell

3.1 Grundlagen

Den Ausgangspunkt für das Datenmodell bildet das lokal erweiterte Datenbankschema $S = \{(R_1, \mathcal{K}_1), \ldots, (R_n, \mathcal{K}_n)\}$ einer Datenbank D. Ein lokal erweitertes Datenbankschema ist eine Menge von Relationenschemata R_i, die jeweils um eine lokale Schlüsselbedingung $\mathcal{K}_i$ erweitert sind [HS00]. Das Universum U der Datenbank ergibt sich aus der Vereinigung aller Relationenschemata. Jedem Relationenschema R_i wird ein Objektschema O_i zugeordnet, das sich aus den Attributen von R_i und eventuellen Umbenennungen β_i ergibt. Die Objektschemata sind über gleichbenannte Attribute miteinander verbunden. Ein Objekt o_i ist eine Menge von Tupeln über O_i, die sich aus Tupeln der Relation r_i unter Berücksichtigung der Umbenennung β_i ergibt.

Die Menge der Objektschemata kann durch die Einführung weiterer Objektschemata zu einem Relationenschema R_i in Verbindung mit entsprechenden Umbenennungen zur Simulation von Selbstverbunden bzw. zum Auflösen von Zyklen erweitert werden. Die Verbindung zu anderen Objektschemata erfolgt durch die Gleichbenennung der gewünschten Verbundattribute. Für das lokal erweiterte Datenbankschema auf Objekt-Ebene $S_O = \{(O_1, K_1), \ldots, (O_m, K_m)\}$ mit $m \geq n$ gilt: (i) $O_i = \beta_i(R_i)$ und $K_i = \beta_i(\mathcal{K}_i)$ für alle $i \leq n$ und (ii) für alle (O_i, K_i) mit $n < i \leq m$ existiert ein Relationenschema R_j $(1 \leq j \leq n)$ mit $O_i = \beta_i(R_j)$ und $K_i = \beta_i(\mathcal{K}_j)$. Das Universum U_O ergibt sich aus der Vereinigung aller in S_O enthaltenen Objektschemata.

Das lokal erweiterte Datenbankschema auf Objekt-Ebene lässt sich in einem Hypergraphen darstellen. Ein *Hypergraph* H ist ein Paar (N, E) mit einer endlichen Knotenmenge N und einer Kantenmenge E, wobei jede Hyperkante $e \in E$ eine beliebige nicht leere Teilmenge von N ist [Fag83]. Der Hypergraph stellt eine Verallgemeinerung eines gewöhnlichen ungerichteten Graphen dar, wobei eine Kante nicht notwendigerweise zwei Knoten, sondern eine beliebige nicht leere Teilmenge von N enthält. Die Knotenmenge des Hypergraphen H für S_O entspricht dem Universum U_O, die Hyperkanten werden durch die Objekte aus S_O gebildet. S_O ist vollständig definiert, wenn H zusammenhängend ist und somit mindestens ein Knoten jeder Kante in einer anderen Kante vorkommt. Hypergraphen stellen die Grundlage für die Betrachtung der Azyklität eines Datenbankschemas und die einfache Berechnung von Verbundausdrücken dar.

$U_P \subset U_O$ ist die Menge aller Datenbank-Attribute, die beim Zugriff auf die Datenbank über das Web projizierbar oder selektierbar sind. Diese öffentlich zugänglichen Attribute müssen vom Datenbank-Administrator definiert werden. Jedes Attribut besitzt eine Domäne, die angibt, welche Einträge für das betreffende Attribut auftreten können. Für die Selektionsattribute werden einfache textuelle Domänen vorausgesetzt. Jedem Attribut A_i ist außerdem eine aktive Domäne $\mathcal{D}_i$ zugeordnet, die alle im aktuellen Datenbankzustand auftretenden Werte umfasst. Selektionsattribute werden im Folgenden mit $\mathcal{S}$, Projektionsattribute mit $\mathcal{P}$ bezeichnet. U_P ergibt sich aus der Vereinigung der beiden Mengen[2].

[2]Die beiden Mengen sind im Allgemeinen nicht disjunkt, d.h. verschiedene Attribute können sowohl projizierbar als auch selektierbar sein.

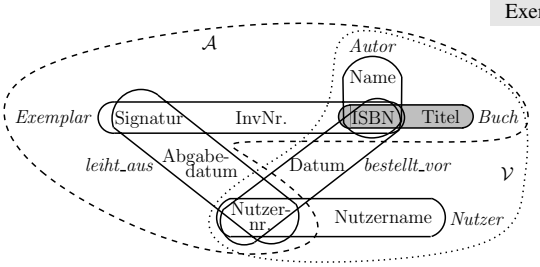

Abbildung 1: Hypergraph für das Datenbankschema S_O der Bibliotheksdatenbank auf Objekt-Ebene

Exemplar

Signatur	ISBN	InvNr.
S011-1	3-89319-175-5	001
S040-1	0-8053-1753-8	083
S040-2	0-8053-1753-8	084
S040-3	0-8053-1753-8	085
S110-1	3-929821-31-1	138
S110-2	3-929821-31-1	139
S110-3	3-929821-31-1	140
S110-4	3-929821-31-1	141
S444-1	3-89319-800-8	431

Autor

ISBN	Name
3-89319-175-5	Gottfried Vossen
3-89319-175-5	Witt
0-8053-1753-8	Elmasri
0-8053-1753-8	Navathe
3-929821-31-1	Andreas Heuer
3-929821-31-1	Gunter Saake
3-89319-800-8	Andreas Heuer

Buch

ISBN	Titel
3-89319-175-5	Das DB2-Handbuch
0-8053-1753-8	Principles of DBS
3-929821-31-1	Datenbanken
3-89319-800-8	Objektorientierte Datenbanken

Nutzer

Nutzernr.	Nutzername
100	Bodo Schulz
200	Otto Mayer

leiht_aus

Signatur	Nnr.[3]	Abgabedatum
S011-1	100	10.12.2004
S110-1	100	19.12.2004
S110-2	200	01.12.2004
S444-1	200	01.12.2004

bestellt_vor

ISBN	Nnr.[3]	Datum
3-89319-175-5	200	15.11.2004
3-89319-800-8	100	17.11.2004

Abbildung 2: Relationale Repräsentation der Bibliotheksdatenbank

Beispiel 1 Die Abbildung 1 zeigt den Hypergraphen für eine Bibliotheksdatenbank. Zu jedem Objekt, das im Hypergraphen enthalten ist, existiert genau eine Relation aus Abbildung 2. Das Objekt *Buch* ist z.B. über eine Hyperkante dargestellt, die beide Attribute des Objekts umfasst. Projektionsattribute sind *ISBN* und *Titel*, als Selektionsattribute werden *Titel*, *Name* und *Nutzername* definiert. □

Die Menge aller Terme T_i zu einem Attribut $A_i \in \mathcal{S}$ ergibt sich aus der Vereinigungsmenge aller Terme, aus denen die Werte der aktiven Domäne $\mathcal{D}_i$ zusammengesetzt sind. Das Vokabular V_D einer Datenbank D wird aus der Vereinigung aller T_i für alle Selektionsattribute $A_i \in \mathcal{S}$ gebildet.

Eine Informationseinheit in der Datenbank wird im Folgenden als Entity bezeichnet. Im Vergleich zum Entity-Relationship-Modell ist der Begriff Entity hier allgemeiner zu interpretieren, da für eine Datenbank genau ein Entitytyp definiert wird. D_E kennzeichnet die als Menge von Entities $\{e_1, \ldots, e_n\}$ betrachtete Datenbank. Das erweiterte Entity-Schema $S_E = (U_P, K_E)$ besteht aus dem Schema U_P und dem Universalschlüssel K_E. Das Schema wird aus allen öffentlichen Attributen gebildet. Der Universalschlüssel dient zur eindeutigen Identifizierung eines Entities – für alle $e_1, e_2 \in D_E$ gilt: $e_1(K_E) = e_2(K_E) \Rightarrow e_1 = e_2$ – und muss in der Menge der Projektionsattribute $\mathcal{P}$ enthalten sein. Ein Entity

[3]Nutzernr.

$e_i = \{(A_1, v_{1_i}), \ldots, (A_k, v_{k_i})\}$ ist eine Menge von Attribut-Wert-Paaren. Jedes A_j ist ein Attribut aus U_P und $v_{j,i}$ bezeichnet einen Wert aus der aktiven Domäne $\mathcal{D}_j$ von A_j. Ein Attribut $A_j \in (\mathcal{S} \setminus \mathcal{P})$, dem eine Menge von Werten zugeordnet ist, darf mehrfach auftreten. Hat A_j in einem Entity n verschiedene Werte, dann wird das Entity e_i in der Form $e_i = \{\ldots, (A_j, v_1), (A_j, v_2), \ldots, (A_j, v_n), \ldots\}$ repräsentiert. Für alle Projektionsattribute gilt, dass sie in einem Entity enthalten sein müssen und nicht mehrfach vorkommen dürfen. Das Vorhandensein eines Selektionsattributs in einem Entity ist dagegen nicht zwingend.

Für jedes reine Selektionsattribut $A_j \in \mathcal{S} \setminus \mathcal{P}$ muss spezifiziert werden, ob es mengenwertig ist und damit mehrfach in einem Entity auftreten kann. Dabei ist zu beachten, das (i) die Angaben für die Attribute aus demselben Objekt äquivalent sind und (ii) Attribute, deren Verbundpfad zum Universalschlüssel über ein Objekt mit einem mengenwertigen Attribut geht, ebenfalls mengenwertig sein müssen.

Beispiel 2 Der Universalschlüssel der Bibliotheksdatenbank ist $\{$*ISBN*$\}$. Die Selektionsattribute *Name* und *Nutzername* sind mengenwertig, das Attribut *Titel* nicht. □

3.2 Unstrukturierte und strukturierte Anfragen

Im Folgenden werden die Begriffe unstrukturierte und strukturierte Anfrage in Anlehnung an [CdSV+02] eingeführt. Die Menge DB umfasst alle in der Datenbank-Suchmaschine indexierten Datenbanken.

Definition 1 *Eine unstrukturierte Anfrage Q_U ist eine Menge von Schlüsselwörtern bzw. Termen $\{t_1, \ldots, t_n\}$.*

Die Antwort R_{Q_U} auf eine vom Nutzer spezifizierte unstrukturierte Anfrage Q_U ist eine Menge von Datenbanken, die mindestens einen Term $t_i \in Q_U$ im aktuellen Vokabular V_D enthalten, d.h. $R_{Q_U} = \{D \in DB \mid \exists\, t_x \in Q_U \ \wedge\ t_y \in V_D \ mit\ t_x = t_y\}$. Das zugrunde liegende Retrievalmodell ist das Vektorraummodell. Der Vektorraum wird durch die Terme aus der Vereinigung des Vokabulars V_D aller Datenbanken $D \in DB$ aufgespannt.

Definition 2 *Eine strukturierte Anfrage Q_S ist definiert als Menge von Attribut-Wert-Paaren $\{(A_1, v_{1q}), \ldots, (A_m, v_{mq})\}$, wobei jedes A_j ein Attribut aus der Menge der Selektionsattribute $\mathcal{S}$ und v_{jq} ein Wert aus der aktiven Domäne $\mathcal{D}_j$ von A_j ist. Ein reines Selektionsattribut $A_j \in \mathcal{S} \setminus \mathcal{P}$ darf mehrfach auftreten.*

Strukturierte Anfragen werden vom System auf der Basis des aktuellen Vokabulars V_D und der Kenntnis darüber, welche Attribut-Wert-Paare einer Datenbank $D \in DB$ zu einem Term passen, aus einer unstrukturierten Anfrage generiert. Das Ergebnis R_{Q_S} einer strukturierten Anfrage Q_S an eine Datenbank $D \in DB$ wird durch die Menge aller Entities gebildet, in denen mindestens ein Attribut-Wert-Paar aus Q_S enthalten ist, d.h. $R_{Q_S} = \{e_j \in D_E \mid \exists\, (A_x, v_{x_q}) \in Q_S \ \wedge\ (A_y, v_{y_j}) \in e_j \ mit\ (A_x = A_y \ \wedge\ v_{x_q} = v_{y_j})\}$.

Als Grundlage für die Bewertung dient derselbe Vektorraum wie bei der Bewertung einer Datenbank bzgl. einer unstrukturierten Anfrage.

Beispiel 3 Ein Beispiel für eine unstrukturierte Anfrage ist $\{$*Handbuch, Vossen*$\}$. Beide Terme sind im Vokabular der Bibliotheksdatenbank enthalten, als strukturierte Anfrage leitet das System $\{$*(Titel, 'Das DB2-Handbuch'), (Name, 'Gottfried Vossen')*$\}$ ab. □

Strukturierte Anfragen sind eng mit Anfragen an Universalrelationen (UR) verknüpft, da keine logische Navigation in Form von Verbundausdrücken innerhalb der Anfrage notwendig ist. Einzelheiten zum UR-Modell und zu den Annahmen, auf denen das Modell basiert, können in [MUV84] nachgelesen werden. Die wesentliche Voraussetzung für den Einsatz dieses Modells ist, dass die Attributnamen eine eindeutige Rolle spielen (Universal Relation Schema Assumption). Im vorgestellten Datenmodell kann dies durch geeignete Umbenennungen innerhalb der Objekt-Ebene sichergestellt werden. Im Gegensatz zum UR-Modell sind Mengenvergleiche bei Anfragen an mengenwertige Attribute durch die mehrfache Angabe des Attributs mit verschiedenen Werten formulierbar. Solche Mengenvergleiche sind innerhalb des UR-Modells nur über einen Selbstverbund der virtuellen Universalrelation möglich. Ein weiterer Unterschied zur UR-Anfragen ist der Wegfall der Spezifikation der Projektionsattribute: Ein Ergebnis-Entity enthält in diesem Modell standardmäßig alle Attribute aus der Menge $\mathcal{P}$ und weiterhin alle zu einer strukturierten Anfrage passenden Attribut-Wert-Paare.

3.3 3-Ebenen-Modell für die Datenbank-Schnittstelle

Das Datenmodell lässt sich in 3 Ebenen aufteilen, die in Abbildung 3 dargestellt sind. Die Relationale Ebene bildet die grundlegende Schicht, auf der die Objekt-Ebene aufsetzt. Jedem Objekt ist eine Relation zugeordnet. Das Objektschema ergibt sich aus dem Schema der zugeordneten Relation und eventuell vorgenommenen Umbenennungen dieser Attribute.

Die Objekt-Ebene repräsentiert die für die Anwendung notwendige Semantik der Datenbank, die sich aus dem Datenbankschema, den definierten Fremdschlüsselbeziehungen und zusätzlichen Angaben des Datenbank-Administrators ergibt. Fehlende Verbindungen können in dieser Ebene durch die Gleichbenennung von Attributen vom Administrator spezifiziert werden. Ungleichnamige Attribute, die in einer Fremdschlüsselbeziehung stehen, werden automatisch vom System gleich benannt. Das Datenbankschema ist um abgeleitete Attribute – in Verbindung mit einem zusätzlichen Objektschema – erweiterbar, die Anfragen mit einem explizit formulierten Selbstverbund ersetzen. Somit können Tupel aus derselben Relation auf Objekt-Ebene mit einem einfachen Verbund in Beziehung gebracht werden.

Beispiel 4 Sei eine Datenbank mit der Relation r_P über dem Schema R_P = *(Person, Elternteil)* zur Abspeicherung der Elternteile einer Person gegeben. Ein Selbstverbund wird benötigt, wenn festgestellt werden soll, ob eine Person ein Großelternteil einer anderen ist.

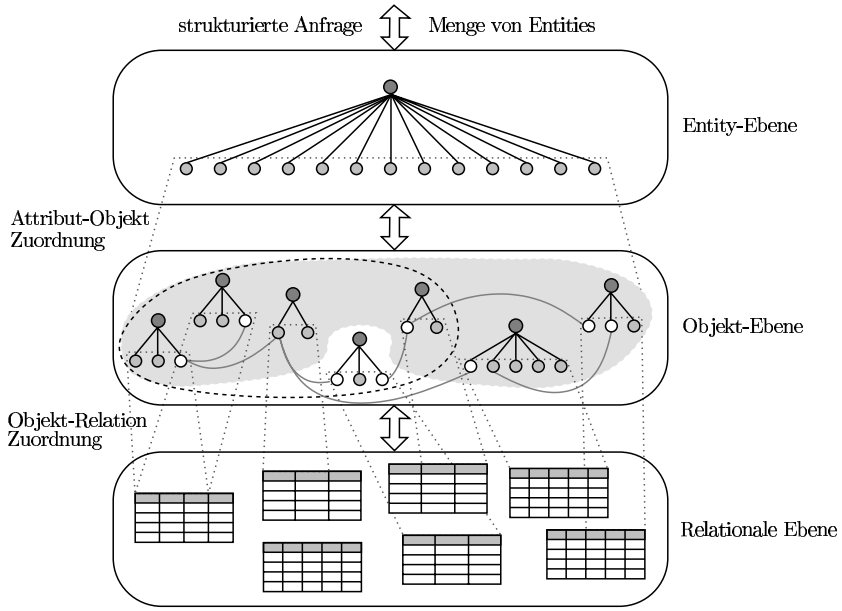

Abbildung 3: 3-Ebenen-Modell für die Datenbank-Schnittstelle

Durch die Spezifikation der beiden Objektschemata O_P = *(Person, Elternteil)* und O_{GE} = *(Elternteil, Großelternteil)* für R_P wird das abgeleitete Attribut *Großelternteil* definiert. Die Beziehung zwischen den Attributen *Person* und *Großelternteil* basiert auf einem einfachen Verbund der beiden Objekte. □

Zyklen, die innerhalb eines Schemas auftreten, können ebenfalls auf der Objekt-Ebene durch die Umbenennung von Attributen in den Objektschemata bzw. durch die Definition mehrerer Objektschemata für die verschiedenen Rollen einer Relation aufgelöst werden.

Für die Anfrageinterpretation wird das für die Berechnung der Projektionsattribute relevante Teilschema von S_O benötigt. Das Projektionsschema S_P lässt sich über die Tableauoptimierung bestimmen. Die Tableauoptimierung ist eine Methode zur Optimierung einer Anfrage mit Selektionen, Projektionen und Verbunden mit dem Ziel der kleinstmöglichen Anzahl von Verbunden. Die Ausgangsanfrage ist eine Projektion des Verbunds über allen Objekten auf die Attribute $A_i \in \mathcal{P}$. Einzelheiten zur Tableauoptimierung können in [ASU79] nachgelesen werden. Aus dem optimierten Tableau lassen sich alle Objekte ablesen, die zum Verbinden aller Projektionsattribute erforderlich sind.

Beispiel 5 Als Projektionsschema für die Bibliotheksdatenbank – in der Abbildung 1 grau hinterlegt – ergibt sich S_P = {*Buch* = {*ISBN, Titel*}} □

Die Entity-Ebene bildet die oberste Schicht des Systems und bietet eine universelle Sicht auf die Datenbank. Die Details der Konstruktion der Entities in der Objekt-Ebene wer-

den in dieser Ebene versteckt. Die Attribute der Entities umfassen alle Projektions- und Selektionsattribute.

3.4 Erweiterung des Datenmodells zur Behandlung zyklischer Datenbankschemata

Das erweiterte Datenbankschema S_O für die Objekt-Ebene kann Zyklen enthalten. Ein Zyklus hat zur Folge, dass verschiedene Verbundpfade zwischen dem Projektionsschema und einem Objekt existieren. Bei der Anfrageinterpretation ist in diesen Fällen nicht klar, welche Pfade zur Berechnung des Ergebnisses eingesetzt werden sollen. Details zu den verschiedenen Graden von Zyklen können in [Fag83] nachgelesen werden. Wichtig für die Anfrageverarbeitung ist vor allem das Auflösen der so genannten α-Zyklen. Ein *α-Zyklus* liegt vor, wenn die Graham-Reduktion des Hypergraphen, der aus dem Datenbankschema resultiert, nicht zum leeren Graphen führt. Die *Graham-Reduktion* beinhaltet die Nacheinanderausführung der folgenden Regeln in beliebiger Reihenfolge, bis keine Regel mehr angewendet werden kann: (i) Eliminiere alle Knoten, die nur in einer Hyperkante enthalten sind, und (ii) Eliminiere alle Hyperkanten, die Teilmenge einer anderen sind. Ein α-azyklisches Schema stellt aber nicht sicher, dass immer ein eindeutiger Verbundpfad für eine Attributmenge existiert. Für diese Fälle liefert aber die kürzeste minimale Verbindung – bezeichnet als kanonische Verbindung – im Allgemeinen sinnvolle Ergebnisse.

Ein Zyklus entsteht dadurch, dass ein beteiligtes Objekt o semantisch überladen ist und somit verschiedene Rollen spielt [MU83]. Er kann aufgelöst werden, indem für jeden Verbundpfad – und damit für jede Rolle von o – ein separates Objekt definiert wird. In vielen Fällen gestaltet sich aber die Bestimmung des semantisch überladenen Objekts schwierig. Eine Möglichkeit zur Behandlung von Zyklen ohne die Definition neuer Objekte ist die Verwendung maximaler Objekte. Ein *maximales Objekt* enthält alle Objekte, die maximal zur Berechnung des Verbunds zwischen einer Attributmenge genutzt werden dürfen. Maximale Objekte müssen α-azyklisch sein und können automatisch berechnet werden [MU83]. Die dafür benötigten eingebetteten mehrwertigen Abhängigkeiten lassen sich aus der angenommenen Verbundabhängigkeit über allen Objekten und den Schlüsselabhängigkeiten ableiten. Sind die zu einer Attributmenge gehörenden Objekte in mehreren maximalen Objekten enthalten, dann ergibt sich das Ergebnis aus der Vereinigung der Verbundausdrücke dieser Objekte. Die Einordnung der maximalen Objekte in das Modell für die Datenbank-Schnittstelle zeigt Abbildung 3. Der grau hinterlegte Bereich in der Objekt-Ebene stellt das eine maximale Objekt dar, ein Weiteres wird durch die Objekte innerhalb des gestrichelten Ovals definiert.

Beispiel 6 Die Bibliotheksdatenbank enthält einen α-Zyklus. Bücher und Nutzer sind dadurch über verschiedene Pfade verbunden. Ein Nutzer steht in Beziehung zu einem Buch, wenn er entweder (i) ein Exemplar dieses Buches ausgeliehen oder (ii) das Buch vorbestellt hat. Durch die Definition der beiden in Abbildung 1 dargestellten maximalen Objekte $\mathcal{A}$ (ausgeliehene Buchexemplare) und $\mathcal{V}$ (vorbestellte Bücher) werden beide Pfade separat ausgewertet. Die Vereinigung der beiden Teilergebnisse liefert alle Bücher, die ein Nutzer ausgeliehen oder vorbestellt hat. □

Die definierten maximalen Objekte werden in der Menge $\mathcal{M} = \{\mathcal{M}_1, \ldots, \mathcal{M}_k\}$ zusammengefasst. Als Voraussetzung für die in Abschnitt 6 beschriebene Anfrageinterpretation muss das Projektionsschema S_P im Durchschnitt aller definierten maximalen Objekte enthalten sein. Ist S_O α-azyklisch, dann existiert genau ein maximales Objekt, das alle Objekte O_i aus S_O umfasst. In diesem Fall wird zur Vereinfachung $\mathcal{M} = \emptyset$ definiert.

4 Überblick über das System

Der folgende Abschnitt gibt einen Überblick über die Arbeitsweise des Kern-Systems am Beispiel der Bibliotheksdatenbank. Eine Illustration der Abarbeitung einer unstrukturierten Anfrage findet sich in Abbildung 4. Die Navigation zur Detaildarstellung eines relevanten Entities wird hier nicht betrachtet.

Angenommen, ein Nutzer sucht nach Informationen, die Assoziationen zwischen den Termen “Heuer”, “Saake” und “Schulz” beinhalten. Im ersten Schritt werden über den IR-Index diejenigen Datenbanken, Attribute und Werte bestimmt, die einen Term aus der Anfrage enthalten. Außerdem wird für jede passende Attribut-Wert-Kombination ermittelt, wie häufig sie in der Datenbank als Bestandteil eines Entities auftritt. Der Term “Saake” ist beispielsweise in der Datenbank db_1 im Attribut *Name* als Teil des Attributwerts “Gunter Saake” mit der Vorkommenshäufigkeit f_3 enthalten.

Anschließend erfolgt die Ableitung strukturierter Anfragen für die ermittelten Attribut-Wert-Paare bezüglich der einzelnen Datenbanken mit Hilfe der gespeicherten Informationen über die Objektschemata dieser Datenbanken. Für die Datenbank db_1 wird zum Beispiel die strukturierte Anfrage {*(Name, 'Andreas Heuer'), (Name, 'Gunter Saake'), (Nutzername, 'Bodo Schulz')*} mit einer Bewertung s_1 generiert. Im Allgemeinen lässt sich eine Menge von strukturierten Anfragen für eine Datenbank ableiten. Die Gesamtbewertung r_i der Datenbank ergibt sich auf der Basis der am höchsten bewerteten strukturierten Anfrage.

Das Ergebnis der unstrukturierten Anfrage ist eine bewertete Liste von Datenbanken, die relevante Informationen bzgl. der Anfrage enthalten. In der zweiten Phase wird in der vom Nutzer gewählten Datenbank nach Entities gesucht, die möglichst viele Anteile einer strukturierten Anfrage, die aus der Menge $\mathcal{Q}_S$ der an die Datenbank übermittelten strukturierten Anfragen stammt, erfüllen. Für jedes in $\mathcal{Q}_S$ referenzierte Objekt O wird eine Anfrage abgeleitet, die das Objekt mit dem Projektionsschema in Verbindung bringt. Existieren für O mehrere Pfade zum Projektionsschema, dann wird für jedes maximale Objekt, das O enthält, eine Anfrage generiert. Im anschließenden Nachbearbeitungsschritt werden die Entities aus den verschiedenen Anfrageergebnissen mit Hilfe des Universalschlüssels zusammengesetzt und bewertet.

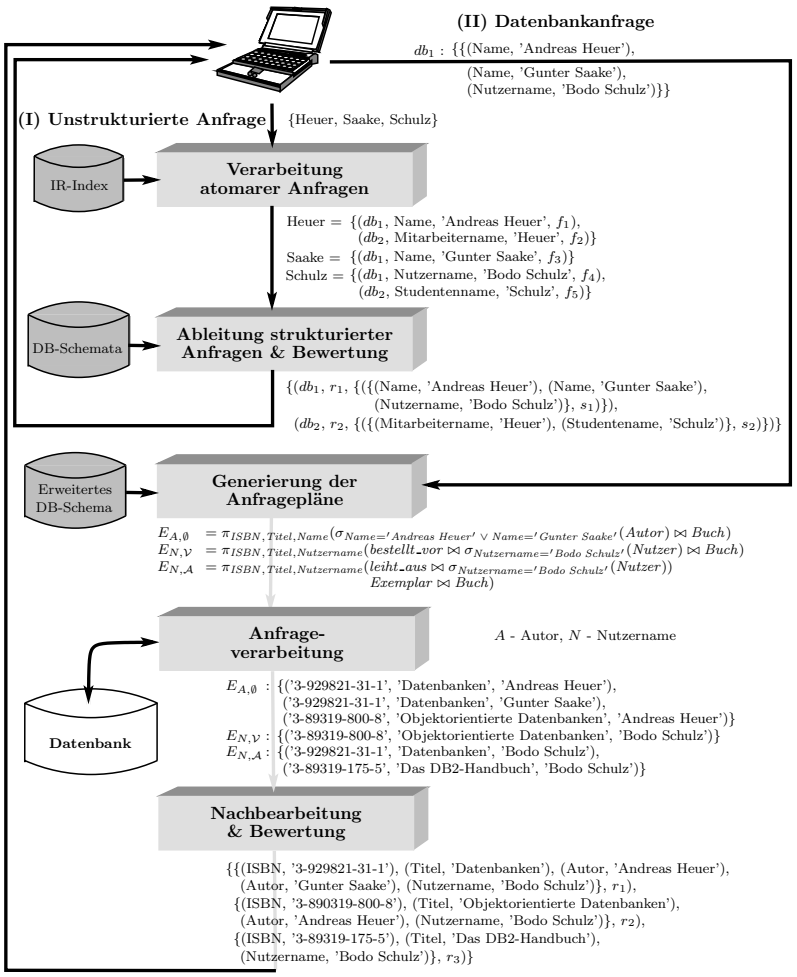

Abbildung 4: Abarbeitung einer unstrukturierten Anfrage durch die Suchmaschine

5 Ableitung und Bewertung strukturierter Anfragen

Innerhalb der Suchmaschine ist keine konkrete Aussage darüber möglich, wie viele Terme einer unstrukturierten Anfrage in einem Entity einer Datenbank D enthalten sind, da nur die aktiven Domänen der Selektionsattribute und keine kompletten Entities im Index der Suchmaschine gespeichert werden. Aus diesem Grund wird eine geeignete Bewertung für eine Datenbank benötigt, bei der nicht im Vordergrund stehen sollte, welche passenden Attribut-Wert-Paare in der Datenbank vorkommen. Die Bewertungsfunktion muss vielmehr berücksichtigen, welche Kombinationen dieser Paare innerhalb eines Entities überhaupt möglich sind und mit welcher Wahrscheinlichkeit diese auftreten können.

Voraussetzungen Jedem Term t_i einer unstrukturierten Anfrage Q_U ist die Menge $\mathcal{V}_{D,t_i}$ aller zu t_i passenden Attribut-Wert-Paare einer Datenbank D zugeordnet. Ein Paar (A_j, v_j) passt zu t_i, wenn t_i in v_j enthalten ist. Die Gesamtmenge $\mathcal{V}_D$ der passenden Paare einer Datenbank D wird durch die Vereinigung der Mengen $\mathcal{V}_{D,t_i}$ aller $t_i \in Q_U$ gebildet.

Zur Sicherstellung einer korrekten Beantwortung von Q_U und einer geeigneten Bewertung muss eine strukturierte Anfrage Q_S, die auf der Basis von Q_U und $\mathcal{V}_D$ abgeleitet wird, folgende Eigenschaften besitzen:

- *Gültigkeit*: Q_S ist gültig, wenn jedes mehrfach in Q_S enthaltene Attribut mengenwertig ist. Für jedes Attribut A_j in Q_S, das nicht mengenwertig ist, gilt: $\nexists (A_k, v_k) \in Q_S \; mit \; k \neq j \wedge \; A_k = A_j$.

  Beispiel 7 $\{$*(Titel, 'Das DB2-Handbuch'),(Titel, 'Datenbanken')*$\}$ stellt keine gültige strukturierte Anfrage dar. Q_S = $\{$*(Autor, 'Andreas Heuer'), (Autor, 'Gunter Saake')*$\}$ ist dagegen gültig, da es sich bei *Autor* um ein mengenwertiges Attribut handelt. □

- *Vollständigkeit*: Q_S ist vollständig, wenn kein Attribut-Wert-Paar $(A_j, v_j) \in \mathcal{V}_D$ existiert, so dass $Q_S \cup \{(A_j, v_j)\}$ eine gültige Anfrage ist, die mehr Terme der unstrukturierten Anfrage enthält. Die Menge $\mathcal{T}_{Q_S}$ der in Q_S vorkommenden Terme aus Q_U wird durch $\{t_i \in Q_U \mid \exists \, (A_j, v_j) \in Q_S \wedge t_i \; ist \; in \; v_j \; enthalten\}$ gebildet.

  Beispiel 8 Q_S = $\{$*(Autor, 'Andreas Heuer'), (Autor, 'Gunter Saake')*$\}$ stellt keine vollständige strukturierte Anfrage für die Bibliotheksdatenbank bzgl. Q_U = $\{$*Heuer, Saake, Datenbank*$\}$ dar. Die Erweiterung von Q_S um das Attribut-Wert-Paar *(Titel, 'Datenbanken')* liefert eine gültige Anfrage, in der alle Terme aus Q_U enthalten sind. □

- *Minimalität*: Q_S ist minimal, wenn zu jedem Term $t_i \in Q_U$ maximal ein Attribut-Wert-Paar $(A_j, v_j) \in Q_S$ existiert.

  Beispiel 9 Angenommen, "Boris Heuer" ist als weiterer Nutzer in der Bibliotheksdatenbank gespeichert. Die Anfrage $\{$*(Autor, 'Andreas Heuer'), (Titel, 'Datenbanken'), (Nutzername, 'Boris Heuer')*$\}$ ist bzgl. Q_U = $\{$*Heuer, Datenbank*$\}$ nicht minimal, da der Term *Heuer* in zwei Attribut-Wert-Paaren vorkommt. □

Eine strukturierte Anfrage Q_S ist eine Menge von Attribut-Wert-Paaren, für die gilt: Ein Entity, in dem eines dieser Attribut-Wert-Paare vorkommt, kann maximal alle in diesen Paaren enthaltenen Terme der unstrukturierten Anfrage Q_U beinhalten. Jedes zusätzliche Attribut-Wert-Paar in Q_S verringert die Wahrscheinlichkeit, dass ein Entity die Menge Q_S vollständig enthält. Dies ist der Grund für die Forderung nach der Minimalität von Q_S.

Ableitung strukturierter Anfragen Die Ableitung der Anfragen für eine Datenbank D erfolgt in einem iterativen Prozess: die Attribut-Wert-Paare $\mathcal{V}_{D,t_i}$ für die einzelnen Terme $t_i \in Q_U$ werden paarweise so kombiniert, dass diese Kombinationen gültig sind. Als

Ergebnis liegt die Menge $\mathcal{Q}_S$ von strukturierten Anfragen vor, wobei jedes Attribut-Wert-Paar aus $\mathcal{V}_D$ in mindestens einer strukturierten Anfrage enthalten ist. Dieser Grundalgorithmus stellt die Gültigkeit und die Vollständigkeit der ermittelten strukturierten Anfragen sicher, aber keine Minimalität. Der Grund dafür liegt im möglichen Auftreten von Attribut-Wert-Paaren, die mehrere Terme $t_i \in Q_U$ gleichzeitig erfüllen.

Beispiel 10 In Tabelle 1 werden die zu den einzelnen Termen der unstrukturierten Anfrage $Q_U = \{t_1, t_2, t_3\}$ angenommenen passenden Attribut-Wert-Paare in der Spalte $\mathcal{V}_{D,t_i}$ aufgelistet. Der Wert x des Paares (D, x) enthält sowohl den Term t_2 als auch t_3. Der

$t_i \in Q_U$	$\mathcal{V}_{D,t_i}$	aus $\mathcal{V}_D$ erzeugte Q_S bzgl. Q_U
t_1	$\{((A, w)\}$	$Q_{S_1} = \{(A, w), (C, y), (D, x)\}$
t_2	$\{(C, y), (D, x)\}$	$Q_{S_2} = \{(A, w), (D, x), (C, z)\}$
t_3	$\{(C, z), (D, x)\}$	$Q_{S_3} = \{(A, w), (D, x)\}$

Tabelle 1: Erzeugte strukturierte Anfragen bzgl. $Q_U = \{t_1, t_2, t_3\}$ für die vorgegebenen Attribut-Wert-Paare zu den einzelnen Termen

Grundalgorithmus generiert die in der rechten Spalte abgebildeten strukturierten Anfragen, für die gilt: $Q_{S_1} \supset Q_{S_3}$ und $Q_{S_2} \supset Q_{S_3}$. Die Anfragen Q_{S_1} und Q_{S_2} sind somit nicht minimal. □

Während des Erzeugungsprozesses müssen deshalb alle strukturierten Anfragen, die ein identisches Attribut-Wert-Paar zu verschiedenen Anfragetermen enthalten, in eine spezielle Menge $\mathcal{I}$ aufgenommen werden. Für alle erzeugten Anfragen ist abschließend zu prüfen, ob diese eine Obermenge einer Anfrage aus $\mathcal{I}$ darstellen. Ist dies der Fall, wird diese Anfrage verworfen.

Bewertung Für jede Datenbank, die mindestens ein passendes Attribut-Wert-Paar zu einem Term der unstrukturierten Anfrage enthält, wird die Menge $\mathcal{Q}_S$ abgeleitet. Die Bewertung der Datenbank erfolgt für jede strukturierte Anfrage $Q_{S_j} \in \mathcal{Q}_S$, die höchste Bewertung spiegelt die Güte der Datenbank bzgl. der unstrukturierten Anfrage Q_U wider. Die Datenbank wird somit nicht aufgrund aller in der Datenbank enthaltenen Anfrageterme, sondern auf der Grundlage der maximal in einem Entity vorkommenden Anfrageterme bewertet. Die Bewertung basiert auf einem klassischen tf-idf Maß (Termfrequenz tf - Inverse Dokumentfrequenz idf) zur Umsetzung des Vektorraummodells. Details zum Vektorraummodell sind u.a. in [Fer03] zu finden.

Sei $DB = \{D_1, \ldots, D_m\}$ die Menge der indexierten Datenbanken und $T = \{T_1, \ldots, T_n\}$ die Menge der Terme $T_j : DB \rightarrow R$, die in den aktiven Domänen der Selektionsattribute dieser Datenbanken vorkommen. Für jede Datenbank $D_i \in DB$ sei zu jedem Term $T_k \in T$ ein Gewicht $w_{D_i,k} \in R$ gegeben. Die Gewichte der Datenbank D_i lassen sich zum Vektor $w_{D_i} = \{w_{D_i,1}, \ldots, w_{D_i,n}\} \in R^n$ zusammenfassen, der als Datenbankvektor bezeichnet wird. Die unstrukturierte Anfrage Q_U wird ebenfalls als Vektor $q \in R^n$ dargestellt, wobei jedem Term $T_k \in Q_U$ ein Gewicht $q_k > 0$ zugeordnet wird. Für alle anderen Terme T_l, die nicht in der Anfrage vorkommen, ist $q_l = 0$.

Das Gewicht $w_{D_i,k}$ eines Terms T_k für die Datenbank D_i lässt sich über die Formel $w_{D_i,k} = log_e(1 + f_{D_i,k})$ berechnen, wobei $f_{D_i,k}$ die Vorkommenshäufigkeit des Terms T_k in der Datenbank D_i angibt. Für einen Anfrageterm T_k wird das Gewicht q_k über die Formel $q_k = \log_e(1 + m/f_{T_k})$ ermittelt – f_{T_k} bezeichnet die Anzahl der Datenbanken, die den Term T_k enthalten.

Jeder strukturierten Anfrage Q_{S_j} wird ein Vektor $s_j = \{s_{j,1}, \ldots, s_{j,n}\}$ zugeordnet, dessen Bestandteile $s_{j,k}$ auf der Basis der in Q_{S_j} enthaltenen Attribut-Wert-Paare gebildet werden:

$$s_{j,k} = \begin{cases} 0 & \textit{falls } T_k \notin Q_U \\ -w_{D_i,k} & \textit{falls } T_k \in Q_U \wedge \nexists\, (A_l, v_l) \in Q_{S_j} : T_k \textit{ kommt in } v_l \textit{ vor} \\ -w_{D_i,k} + w_{v_l} & \textit{falls } T_k \in Q_U \wedge \exists\, (A_l, v_l) \in Q_{S_j} : T_k \textit{ kommt in } v_l \textit{ vor} \end{cases}$$

Das Gewicht w_{v_l} des Attribut-Wert-Paares $(A_l, v_l) \in Q_{S_j}$ ergibt sich analog zu den Termgewichten über die Gleichung $w_{v_l} = log_e(1 + f_{v_l})$, wobei f_{v_l} die Anzahl des Auftretens von (A_l, v_l) in den Entities der Datenbank D_i bezeichnet[4]. Der Vektor s_j dient zum Verschieben des Datenbankvektors w_{D_i} bezüglich der in Q_{S_j} enthaltenen Attribut-Wert-Paare. Terme aus der unstrukturierten Anfrage, die nicht in Q_{S_j} vorkommen, sollen bei der Berechnung der Ähnlichkeit zwischen dem Datenbank- und dem Anfragevektor nicht berücksichtigt werden. Das Gewicht der in den Attribut-Wert-Paaren enthaltenen Terme aus Q_U wird entsprechend der Häufigkeit des zugehörigen Paares angepasst.

Das Cosinus-Maß zwischen den beiden Vektoren $w_{D_i} = (w_{D_i,1}, \ldots, w_{D_i,n}) \in R^n$ und $q = (q_1, \ldots, q_n) \in R^n$ unter Berücksichtigung des Vektors $s_j = \{s_{j,1}, \ldots, s_{j,n}\}$ für die strukturierte Anfrage Q_{S_j} ist eine Abbildung $R^n \times R^n \to R$, die durch

$$cos(w_{D_i} + s_j, q) = \frac{(w_{D_i} + s_j) \cdot q}{|w_{D_i} + s_j|\,|q|} = \frac{\sum_{k=1}^{n} (w_{D_i,k} + s_{j,k}) \cdot q_k}{\sqrt{\sum_{k=1}^{n} (w_{D_i,k} + s_{j,k})^2} \cdot \sqrt{\sum_{k=1}^{n} q_k^2}}$$

berechnet wird. Da die Änderung der euklidischen Länge w_{D_i} des Datenbankvektors zu aufwendig wäre und auch nur sehr geringe Auswirkungen auf das Ähnlichkeitsmaß hat[5], wird diese für die Bewertung der Datenbank bzgl. einer strukturierten Anfrage vernachlässigt.

Beispiel 11 Die unstrukturierte Anfrage Q_U = {*Heuer, Saake, Schulz*} (siehe Abbildung 4) soll von der Suchmaschine beantwortet werden, deren Index Inhalte aus der Bibliotheksdatenbank db_1 und neun weiteren Datenbanken enthält. In den indexierten Informationen der Datenbank existiert zum Term *Heuer* das passende Attribut-Wert-Paar *(Name, 'Andreas Heuer')* mit der Auftretenshäufigkeit 2. Der Anfrageterm *Saake* kommt im Paar *(Name, 'Gunter Saake')* mit der Häufigkeit 1 vor, der Term *Schulz* ist im Paar *(Nutzername, 'Bodo Schulz')* mit der Häufigkeit 3 enthalten. Weiterhin existieren in der Vorlesungsdatenbank

[4] Es wird davon ausgegangen, das ein Term nicht mehrfach in einem Attributwert enthalten ist.

[5] Im Allgemeinen gilt: $|w_{D_i}| \approx |w_{D_i} + s_j|$.

db_2 passende Attribut-Wert-Paare zu den Termen *Heuer* und *Schulz*. Für die Anfrageterme ergeben sich die Gewichte $q_{Heuer} = log_e(1 + 10/2)$, $q_{Saake} = log_e(1 + 10/1)$, $q_{Schulz} = log_e(1 + 10/2)$.

Für die Bibliotheksdatenbank wird die strukturierte Anfrage {*(Name, 'Andreas Heuer'), (Name, 'Gunter Saake'), (Nutzername, 'Bodo Schulz')* abgeleitet. Die Bewertung der Datenbank von ≈ 0.45 ergibt sich aus $log_e(1+2) * q_{Heuer} + log_e(1+1) * q_{Saake} + log_e(1+3) * q_{Schulz}$ dividiert durch das Produkt der euklidischen Längen des Datenbank- (≈ 3.89) und des Anfragevektors. Für die zweite Datenbank wird eine Bewertung von 0.20 ermittelt (bei einer angenommenen euklidischen Länge von 5.87). □

6 Abarbeitung strukturierter Anfragen durch die Web-Datenbank

Sei $\mathcal{Q}_S = \{Q_{S_1}, \ldots, Q_{S_n}\}$ die Menge der strukturierten Anfragen, die für die Datenbank D_i ermittelt wurde. Wählt der Nutzer die Datenbank D_i aus der Liste der Datenbanken aus, die als Ergebnis auf die Anfrage Q_U geliefert wurde, wird $\mathcal{Q}_S$ an die Datenbank D_i übermittelt.

Der folgende Algorithmus dient zur Ableitung von Anfrageplänen aus der Menge $\mathcal{Q}_S$, die als relationenalgebraische Ausdrücke dargestellt sind und anschließend unter Berücksichtigung der Zuordnung zwischen den Objekten und den Relationen in SQL-Anfragen umgewandelt werden.

Schritt 1 Gruppiere $\mathcal{Q}_S$ nach den Objekten, die in den Attribut-Wert-Paaren referenziert werden, und speichere das Ergebnis in $\mathcal{Q}_G$.

Beispiel 12 $\mathcal{Q}_S$ = {{*(Name, 'Andreas Heuer'), (Name, 'Gunter Saake'), (Nutzername, 'Bodo Schulz')*}} sei die Menge der strukturierten Anfragen (die in diesem Fall nur eine Anfrage enthält) an die Bibliotheksdatenbank. Für $\mathcal{Q}_G$ ergibt sich {*Autor:* {*(Name, 'Andreas Heuer'), (Name, 'Gunter Saake')*}, *Nutzer:* {*(Nutzername, 'Bodo Schulz')*}}. □

Schritt 2 $\mathcal{C}_{O_i}$ sei die Menge der Attribut-Wert-Paare, die einem Objekt O_i in $\mathcal{Q}_G$ zugeordnet sind. Die Menge aller Attribute, die in den Attribut-Wert-Paaren von $\mathcal{C}_{O_i}$ vorkommen, wird mit $\mathcal{A}_{O_i}$ bezeichnet. Für jedes in $\mathcal{Q}_G$ enthaltene Objekt O_i generiere einen Hypergraphen wie folgt:

- Ist $\mathcal{M} = \emptyset$ und damit S_O α-azyklisch, dann erzeuge einen Hypergraphen $H_{O_i,\emptyset}$ für S_O.
- Ansonsten bestimme zunächst alle maximalen Objekte $\mathcal{M}_j \in \mathcal{M}$, die O_i enthalten, und generiere für jedes maximale Objekt $\mathcal{M}_j$ einen Hypergraphen $H_{O_i,\mathcal{M}_j}$.

Markiere anschließend jeweils alle Attribute aus $\mathcal{A}_{O_i}$ und alle Projektionsattribute $\mathcal{P}$ als starre Knoten – Knoten, die bei der Reduktion nicht gelöscht werden dürfen – und nehme eine Graham-Reduzierung des Hypergraphen vor. Der reduzierte Hypergraph wird mit $R_{O_i,\emptyset}$ bzw. $R_{O_i,\mathcal{M}_j}$ gekennzeichnet. Sind die reduzierten Hypergraphen für verschiedene

maximale Objekte gleich, dann übernehme nur einen dieser Graphen. Bleibt am Ende nur ein reduzierter Hypergraph übrig, so wird er mit $R_{O_i,\emptyset}$ gekennzeichnet.

Beispiel 13 Da das Objekt *Autor* in den maximalen Objekten $\mathcal{A}$ und $\mathcal{V}$ vorkommt, werden die Hypergraphen $H_{Autor,\mathcal{A}}$ und $H_{Autor,\mathcal{V}}$ erzeugt. Die Knoten *Name*, *ISBN* und *Titel* werden als starr markiert und dürfen somit nicht bei der Reduktion entfernt werden. Die beiden reduzierten Hypergraphen werden zu $R_{Autor,\emptyset}$ zusammengefasst, da sie übereinstimmen. Das *Nutzer*-Objekt ist ebenfalls in beiden maximalen Objekten enthalten. Die Reduktion der beiden Hypergraphen $H_{Nutzer,\mathcal{A}}$ und $H_{Nutzer,\mathcal{V}}$ liefert unterschiedliche Graphen $R_{Nutzer,\mathcal{A}}$ und $R_{Nutzer,\mathcal{V}}$.

Schritt 3 Für jeden in Schritt 2 ermittelten reduzierten Hypergraphen $R_{O_i,\mathcal{M}_j}$ mit $\mathcal{M}_j \in \mathcal{M} \cup \{\emptyset\}$ erzeuge einen Ausdruck $E_{O_i,\mathcal{M}_j} = \pi_{\mathcal{P} \cup \mathcal{A}_{O_i}}(O_1 \bowtie \cdots \bowtie O_m)$, wobei jedes Objekt O_x mit einer Kante E_x aus $R_{O_i,\mathcal{M}_j}$ korrespondiert. Anschließend ersetze in $E_{O_i,\mathcal{M}_j}$ das Objekt O_y mit $O_y = O_i$ $(1 \leq y \leq m)$ durch den Ausdruck $\sigma_{S_i}(O_i)$. Die Selektionsbedingung S_i ergibt sich aus $\vee\, C_x$ für alle Attribut-Wert-Paare $C_x \in \mathcal{C}_{O_i}$.

Beispiel 14 Für das Objekt *Autor* lautet der Ausdruck:

$$E_{Autor,\emptyset} = \pi_{ISBN,Titel,Name}(\sigma_{Name='Andreas\ Heuer' \vee Name='Gunter\ Saake'}(Autor) \bowtie Buch)$$

Die folgenden Ausdrücke ergeben sich für das Objekt *Nutzer*:

$$E_{Nutzer,\mathcal{A}} = \pi_{ISBN,Titel,Nutzername}(\sigma_{Nutzername='Bodo\ Schulz'}(Nutzer) \; leiht\_aus \bowtie Exemplar \bowtie Buch)$$

$$E_{Nutzer,\mathcal{V}} = \pi_{ISBN,Titel,Nutzername}(\sigma_{Nutzername='Bodo\ Schulz'}(Nutzer) \bowtie bestellt\_vor \bowtie Buch)$$

□

Nachbearbeitung und Bewertung der Entities Die für die abgeleiteten Anfragen ermittelten Ergebnisse müssen in einem Nachbearbeitungsschritt zu Entities zusammengefasst werden. Sei $\mathcal{R}_{O_i,\mathcal{M}_j}$ das Ergebnis der Anfrage $E_{O_i,\mathcal{M}_j}$, das aus Tupeln über allen Projektionsattributen sowie den in $\mathcal{Q}_S$ enthaltenen Selektionsattributen des Objekts O_i besteht und nach dem Universalschlüssel K_E sortiert ist.

Zunächst wird jede Tupelmenge $\mathcal{R}_{O_i,\mathcal{M}_j}$ in eine Menge von Entities $\mathcal{E}_{O_i,\mathcal{M}_j}$ unter Beibehaltung der Ordnung umgeformt. Jedem Entity wird ein Vektor F zugeordnet, der die Basis für die Bewertung der Entities bildet und die Auftretenshäufigkeit der Terme der unstrukturierten Anfrage in einem Entity aufnimmt. In ein Entity werden für die reinen Selektionsattribute nur solche Attribut-Wert-Paare übernommen, die auch in der Menge $\mathcal{Q}_S$ enthalten sind. Steht ein Objekt in einer mengenwertigen Beziehung zum Universalschlüssel, dann erfolgt außerdem die Zusammenfassung aller Tupel mit gleichen Universalschlüsseln zu einem Entity. Die Ergebnismenge ergibt sich aus der Vereinigung aller ermittelten Entity-Mengen, wobei Entities mit gleichen Universalschlüsseln zu einem Entity zusammengefasst werden. Beim Zusammenfassen der Entities werden die zugeordneten Attribut-Wert-Paar-Mengen vereinigt und der Vektor F entsprechend angepasst.

Entities werden über demselben Vektorraum wie die Datenbanken bewertet. Dazu müssen die Gewichte der Terme in der unstrukturierten Anfrage bekannt sein. Die Bewertung der Entities erfolgt auf der Basis des Skalarprodukts zwischen dem gedachten Entity-Vektor

und dem Vektor für die unstrukturierte Anfrage. In die Berechnung des Skalarprodukts gehen nur die Gewichte der Terme ein, die in der unstrukturierten Anfrage Q_U enthalten sind. Für alle anderen Terme ist das Gewicht innerhalb des Anfragevektors 0. Die Gewichte der Terme $t_i \in Q_U$ für den gedachten Entity-Vektor basieren auf den ermittelten Vorkommenshäufigkeiten, die im Vektor F gespeichert sind.

7 Technische Umsetzung des Systems

WebDBSearch besteht wie bei Suchmaschinen üblich aus 2 Hauptkomponenten – dem Gatherer, der für die Indexierung der Datenbankinhalte verantwortlich ist, und der Anfrageverarbeitungskomponente, die hier als Broker bezeichnet wird. Auf der Seite des Datenbankanbieters müssen 3 Web-Services eingerichtet werden, die auf Java und JDBC basieren. Das Datenbank-Interface, das den Zugriff auf die Datenbank über das Web ermöglicht, wird zur Detaildarstellung einer Informationseinheit genutzt. Die Informationen für die

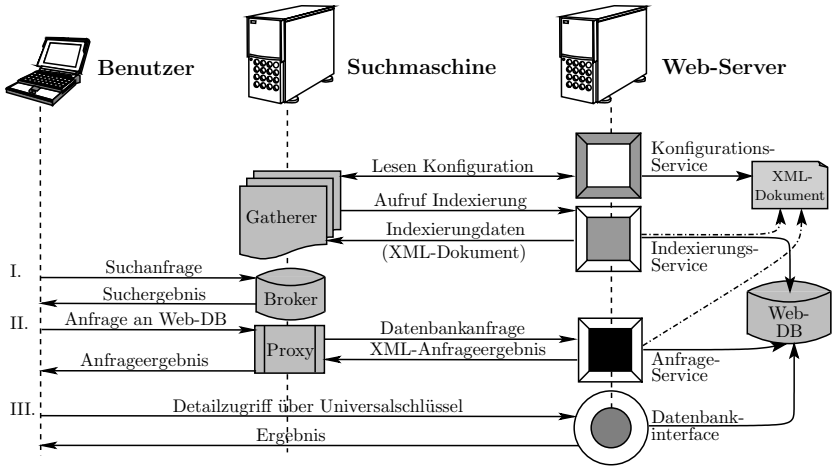

Abbildung 5: Architektur des Systems

einzelnen Ebenen des Datenmodells werden in einem interaktiven Prozess mit dem Datenbank-Administrator auf der Basis der Datendefinition ermittelt und in einer XML-Datei abgelegt. Über den Konfigurations-Service kann die Suchmaschine die notwendigen Informationen über die Web-Datenbank abfragen. Der Indexierungs-Service stellt die Schnittstelle für den Gatherer zur Ermittlung der aktiven Domänen der Selektionsattribute dar. Für jedes Selektionsattribut A_j wird eine Anfrage abgeleitet, die A_j mit dem Universalschlüssel verbindet und das Ergebnis anschließend nach A_j gruppiert. Damit kann ermittelt werden, in wie vielen Entities dieser Attributwert vorkommt. Existieren für das Selektionsattribut verschiedene Verbundpfade zum Universalschlüssel, werden separate Anfragen für die einzelnen Pfade generiert und die Ergebnisse anschließend zusammen-

gefasst. Der Anfrage-Service dient der Verarbeitung der strukturierten Anfragen an eine Web-Datenbank und wird über einen Proxy innerhalb der Suchmaschine aufgerufen.

Einen Überblick über das komplette System gibt Abbildung 5. Die Veröffentlichung einer Datenbank kann entweder über (i) die direkte Anmeldung der Datenbank bei der Suchmaschine oder (ii) die Nutzung der Techniken, die im Umfeld von Web-Services für die Diensteverwaltung mit UDDI und WS-Inspection zur Verfügung stehen, erfolgen. Das Eintragen der Web-Services in ein UDDI-Verzeichnis ermöglicht es, dass die Suchmaschine Web-Datenbanken mit passenden Web-Services für die Integration automatisch ermitteln kann.

8 Zusammenfassung und Ausblick

Das vorgestellte System erlaubt das Auffinden relevanter Informationseinheiten in Web-Datenbanken. Das in Kapitel 3 diskutierte Datenmodell liefert die Grundlage für die Spezifikation dieser Informationseinheiten und die Bereitstellung aller Informationen, die für das automatische Verbinden aller zu einer Informationseinheit gehörenden Tupel benötigt werden, ohne dabei die Autonomie der Web-Datenbank zu beeinflussen. Bei der Definition der Objekt- und der Entity-Ebene sowie den Beziehungen zwischen den Ebenen kann der Administrator weitgehend unterstützt werden.

Der mehrstufige Suchprozess stellt sicher, das die Mehrbelastung der in die Suchmaschine eingebundenen Datenbanken gering ausfällt. Es wird nur auf eine Datenbank zugegriffen, wenn sie der Nutzer auf der Basis der Bewertung und der k-besten strukturierten Anfragen, die ihm präsentiert werden, als relevant erachtet. Die Bewertung der relevanten Datenbanken auf der Basis des Cosinus-Maßes erlaubt eine einfache Kombination von Dokumenten- und Datenbanken-Retrieval.

Die in der Objekt-Ebene vorausgesetzte eindeutige Rolle jedes Attributs und die Verknüpfung der Objekte über gleichnamige Attribute ermöglichen den Einsatz von Hypergraphen zur Darstellung der Verbindungen zwischen den Relationen. Auf der Basis eines Hypergraphen können einfache Verfahren wie die Graham-Reduktion mit starren Knoten eingesetzt werden, um die notwendigen Verbundausdrücke abzuleiten.

Eine Erweiterung des Systems um Anfragen mit booleschen Verknüpfungen und Möglichkeiten zur attributierten Suche ist vorgesehen. Diese Anfragefunktionalitäten sind vor allem für Nutzer mit Vorkenntnissen über eine Datenbank interessant. Geplant ist außerdem die Berücksichtigung von Metadaten wie Attribut- und Relationennamen, die zu den Anfragetermen einer schlüsselwortbasierten Anfragen passen. Einen weiteren Schwerpunkt für zukünftige Arbeiten wird die Evaluierung der Effizienz des Systems bilden. Eine Bewertung der Güte der vorgestellten Anfragemethoden auf der Basis von Recall und Precision ist momentan leider nicht möglich, da keine geeignete Testumgebung – eine Menge von Datenbanken, für die eine Bewertung der Relevanz der einzelnen Informationseinheiten bzgl. verschiedener IR-Anfragen existiert – zur Verfügung steht.

Literatur

[ACD02] Sanjay Agrawal, Surajit Chaudhuri und Gautam Das. DBXplorer: A System for Keyword-Based Search over Relational Databases. In *Proceedings of the 18th International Conference on Data Engineering (ICDE'02), San Jose, California*, Seiten 5–16. IEEE Computer Society, Februar 2002.

[ASU79] A. V. Aho, Y. Sagiv und J. D. Ullman. Efficient Optimization of a Class of Relational Expressions. *ACM Transactions on Database Systems (TODS)*, 4(4):435–454, Dezember 1979.

[BBD99] Michael H. Böhlen, Linas Bukauskas und Curtis E. Dyreson. The Jungle Database Search Engine. In Alex Delis et al., Hrsg., *Proceedings ACM SIGMOD International Conference on Management of Data*, Seiten 584–586. ACM Press, Juni 1999.

[BHNC02] Gaurav Bhalotia, Arvind Hulgeri, Charuta Nakhe und Soumen Chakrabarti. Keyword Searching and Browsing in databases using BANKS. In *Proceedings of the 18th International Conference on Data Engineering (ICDE'02), San Jose, California*, Seiten 431–440. IEEE Computer Society, Februar 2002.

[CdSV+02] Pável Calado, Altigran S. da Silva, Rodrigo C. Vieira, Alberto H.F. Laender und Berthier A. Ribeiro-Neto. Searching web databases by structuring keyword-based queries. In *Proceedings of the 11th International Conference on Information and Knowledge Management*, Seiten 26–33. ACM Press, 2002.

[DEGP98] Shaul Dar, Gadi Entin, Shai Geva und Eran Palmon. DTL's DataSpot: Database Exploration Using Plain Language. In Ashish Gupta, Oded Shmueli und Jennifer Widom, Hrsg., *Proceedings of 24th International Conference on Very Large Data Bases*, Seiten 645–649. Morgan Kaufmann, 1998.

[Fag83] Ronald Fagin. Degrees of Acyclicity for Hypergraphs and Relational Database Schemes. *Journal of the Association for Computing Machinery*, 30(3):514–550, Juli 1983.

[Fer03] Reginald Ferber. *Information Retrieval – Suchmodelle und Data-Mining-Verfahren für Textsammlungen und das Web*. dpunkt.verlag, 2003.

[HP02] Vagelis Hristidis und Yannis Papakonstantinou. DISCOVER: Keyword Search in Relational Databases. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB 2002)*, August 2002.

[HS00] Andreas Heuer und Gunter Saake. *Datenbanken: Konzepte und Sprachen*. mitp-Verlag, 2. Auflage, 2000.

[MU83] David Maier und Jeffrey D. Ullman. Maximal Objects and the Semantics of Universal Relation Databases. *ACM Transactions on Database Systems*, 8(1):1–14, März 1983.

[MUV84] David Maier, Jeffrey D. Ullman und Moshe Y. Vardi. On the Foundations of the Universal Relation Model. *ACM Transactions of Database Systems*, 9(2):283–308, Juni 1984.

[RGM01] Sriram Raghavan und Hector Garcia-Molina. Crawling the Hidden Web. In Peter M. G. Apers et al., Hrsg., *Proceedings of 27th International Conference on Very Large Data Bases*, Seiten 129–138. Morgan Kaufmann, September 2001.