Indexierung von Datenbankinhalten durch Suchmaschinen

Gunnar Weber
Lehrstuhl Datenbank- und Informationssysteme
Fachbereich Informatik, Universität Rostock

Zusammenfassung

In Datenbanken enthaltene Informationen machen einen Großteil des im WWW vorhandenen Wissens aus. In den meisten Fällen sind diese Informationen in dynamisch generierte WWW-Seiten eingebunden, so daß sie von den aktuellen Suchmaschinen nicht indexiert werden können. Im Projekt SWING wurden verschiedene Lösungen zur Integration von Datenbanken entwickelt. Ein bereits realisierter Ansatz nutzt lokal beim Datenbank-Anbieter vorhandene Summarizer, um Index-Fragmente, die aus den Datenbankinhalten generiert werden, an den Gatherer zu liefern. Eine weitere Möglichkeit ist die Beschreibung der Funktionalität des Anfrageformulars und des Aufbaus der Ergebnisseiten in Metadaten, die vom Gatherer zur automatischen Generierung von Anfragen und zur Extraktion der Informationen aus den Ergebnisseiten genutzt werden können.

1 Einleitung

Ein Großteil des Inhalts im WWW wird dynamisch generiert. Allgemein gibt es drei Gründe, Web-Inhalte dynamisch zu gestalten: zeitabhängige Informationen, Anpassung des Inhalts an den Client (Browser) bzw. an den Nutzer (Personalisierung) sowie die Reaktion auf Nutzereingaben. Dynamische Inhalte können auf unterschiedliche Art und Weise erzeugt werden:

- *Serverseitige Programme*: Ein Programm, das auf dem Web-Server läuft, generiert die komplette HTML-Seite, die dann an den Client übermittelt wird (CGI, Java-Servlets).
- *Eingebetter Code mit serverseitiger Ausführung*: Die Web-Seite enthält sowohl statischen HTML-Text als auch eingebetteten Programmcode. Bei der Anforderung der Seite wird dieser Code ausgeführt und das Ergebnis zusammen mit dem HTML-Text an den Client geliefert (PHP, W3-SQL).
- *Eingebetter Code mit clientseitiger Ausführung*: Die Web-Seite enthält neben dem HTML-Text eingebetteten Programmcode. Im Gegensatz zum vorhergehenden Fall wird dieser Code mit dem HTML-Text an den Client übertragen und dort ausgeführt (Java-Applets, ActiveX).

Ein genauere Einordnung dynamischer Web-Inhalte kann in [RGM00] nachgelesen werden.
Die Gatherer der aktuell verfügbaren Suchmaschinen sammeln nur Daten ein, die im sogenannten öffentlich indexierbaren Web [LG98] vorhandenen sind. Dieses umfaßt auch einen geringen Anteil von dynamischen Inhalten. Dazu zählen beispielweise Seiten, die mit Hilfe von Content Management Systemen erzeugt werden und die nach außen wie statische HTML-Seiten wirken. Informationen, die hinter Suchmasken verborgen sind, oder Seiten, die eine Autorisierung bzw. vorherige Registrierung erfordern, werden ignoriert. Dadurch wird bei der Recherche

im Internet eine gewaltige Menge hochqualitativer Informationen aus großen, durchsuchbaren elektronischen Datenbanken nicht berücksichtigt. Solche Datenbanken haben typischerweise ein Web-Anfrageinterface, das ein HTML-Formular enthält. Der Zugriff auf die Datenbank ist dann nur über dynamisch generierte Seiten möglich, die als Antwort auf die Nutzeranfrage geliefert werden.
Nach der obigen Klassifikation dynamischer Web-Inhalte lassen sich Web-Datenbanken wie folgt einordnen: es wird auf Nutzereingaben reagiert und die Generierung des Ergebnisses geschieht gewöhnlich auf der Serverseite. Eine clientseitige Ausführung von eingebettetem Programmcode ist zwar denkbar, wird aus Sicherheitsgründen aber kaum realisiert. Aus diesem Grund wird dieser Mechanismus im folgenden nicht berücksichtigt.
Es gibt verschiedene Varianten für die Integration von Datenbanken in Suchmaschinen. Diese unterscheiden sich in der Möglichkeit des Zugriffs auf die lokale Datenbank:

- *Kooperative Datenbankanbieter* ermöglichen den strukturierten Zugriff auf den Datenbankinhalt (oder spezielle Sichten) über JDBC oder ähnliche Mechanismen. In diesem Fall kann die volle Funktionalität einer Anfragesprache auch in der Datenbankanbindung ausgenutzt werden.

- Bei *nicht-kooperativen Datenbankanbietern* können Anfragen nur über ein Anfrageformular, das im WWW über Parameter versorgt wird, gestellt werden. Das Schema der Datenbank und Anfragefunktionen, die darüberhinaus nutzbar wären, sind nicht bekannt bzw. können nicht an die darunterliegende Datenbank übergeben werden.

In diesem Artikel wird unter anderem ein Extraktionsmechanismus vorgestellt, bei dem vom Anbieter bereitgestellte Metadaten zur Indexierung herangezogen werden. Da der Anbieter im Gegensatz zum nicht-kooperativen Fall aktiv werden muß, um die Indexierung zu ermöglichen, wird die Klasse der *eingeschränkt kooperativen Anbieter* eingeführt: Hier erfolgt der Zugriff auf die Datenbank zwar wieder über das Web-Formular, das Schema der Datenbank und die Anfragefunktionalität des Formulars werden aber über spezielle Metadaten bekannt gegeben.
Bei der Indexierung von Inhalten, die hinter Anfrageformularen in Datenbanken "versteckt" sind (oft auch als *Hidden Web* bezeichnet), treten drei grundlegende Probleme auf.
Das erste ist das Problem der Skalierung. Nach einer Studie [B+00] ist das Hidden Web bis zu 500 mal größer als das statische Web. Aus diesem Grund ist eine umfassende Indexierung unmöglich und häufig auch nicht sinnvoll. So ist es z.B. unsinnig, eine Artikelnummer zu indexieren. Die aktuell verfügbaren Suchmaschinen berücksichtigen keine Zusammenhänge zwischen den Informationen eines Dokumentes[1]. Somit reicht es aus, nur unterschiedliche Werte für ein Attribut und keine kompletten Tupel in den Index aufzunehmen. Dies führt zu einer deutlichen Reduzierung der Größe des Indexfragmentes für eine Datenbank.
Das zweite Problem liegt in der automatischen Erfassung der Funktionalität des Anfrageformulars, das für den menschlichen Nutzer konzipiert ist. In diesem Beitrag werden deshalb Metadaten vorgeschlagen, mit denen Anbieter diese Funktionalität in maschinenverständlicher Art und Weise beschreiben können.
Die Aufgabe des Anfrageformulars besteht in der Filterung der für einen Nutzer relevanten Datenbankinhalte. Diese Filterung führt zum dritten Problem, da ein Gatherer alle wichtigen Inhalte einsammeln soll. Somit muß der Gatherer beim Zugang über eine Web-Schnittstelle mehrere Anfragen nacheinander an die Datenbank stellen. Mit zunehmender Komplexität des

[1] außer der Entfernung zwischen Wörtern

Anfrageformulars und zunehmender Anzahl möglicher Werte für die Elemente steigt der Zeit- und Ressourcenaufwand für das Abfragen der Datenquelle sehr stark an.
Im kooperativen Fall spielen die letzten beiden Probleme keine Rolle, da direkt auf die Datenbank zugegriffen wird.
Für den nicht-kooperativen Fall sei auf zwei interessante Ansätze verwiesen. An der Stanford Universität wurde ein aufgabenspezifischer Web-Crawler mit der Bezeichnung *Hidden Web Exposer* [RGM00] entwickelt, der beim Bestücken der Formulardaten auf eine endliche Menge von Konzepten bzw. Kategorien zugreift. Die Menge der möglichen Werte pro Konzept wird dynamisch um neu extrahierte Informationen erweitert. Die Zuordnung eines Formularelements zu einem Konzept wird über Ähnlichkeiten zwischen der textuellen Beschreibung des Elements und der Konzeptbeschreibung vorgenommen. In [Bru01] wird das Ausnutzen von kombinierten Analysetechniken des Text Mining vorgeschlagen, um Formularelemente zu Konzepten einer Ontologie zuzuordnen.
In Abschnitt 2 wird zunächst ein in der Suchmaschine SWING realisierter Ansatz vorgestellt [HW00], der die Einbindung von Datenbanken kooperativer Anbieter erlaubt. Dieser Mechanismus setzt momentan noch das separate Anmelden einer Datenbank bei der Suchmaschine voraus. Deshalb wird in Abschnitt 3 die Nutzung von Metadaten im Anfrageformular vorgeschlagen, um den Indexierungsprozeß anzustoßen. Der Hauptteil des Artikels beschäftigt sich mit der Integration von Inhalten teilweise kooperativer Anbieter. Die Grundidee dieses Ansatzes ist es, Metadaten so zu definieren, daß der Gatherer die Datenbankinhalte über die Web-Schnittstelle ermitteln kann.

2 Einbindung von Datenbanken in SWING

Basisbestandteil der Suchmaschine SWING ist das Informationsmanagementsystem *Harvest* [HSW96]. Zur Einbindung von Datenbanken wurde die Gatherer-Komponente[2] des Harvest-Systems modifiziert. Der Gatherer ermittelt aus den Dokumenten Informationen, die den Inhalt betreffen (z.B. Titel des Dokuments), und legt sie zusammen mit verschiedenen Meta-Informationen, zu denen beispielsweise URL, Größe, und Datum der letzten Änderung zählen, im Index ab. Die Grundidee des Ansatzes besteht in der Verwendung lokaler Summarizer, die leicht konfigurierbar sind und vorhandene Standards für den Datenbankzugriff nutzen (DBPerl, JDBC).
Die offenen Schnittstellen des Gatherers erlauben die Einbindung weiterer Filterprogramme (Summarizer) für beliebige Dokumenttypen. Dadurch ist es möglich, eigene Filter für die Integration von Datenbankinhalten zu realisieren. Für die Informationen, die vom Gatherer extrahiert werden, definiert Harvest das *Summary Object Interchange Format* (SOIF). Das SOIF besteht aus einer Liste von Attribut/Wert-Paaren, gebräuchliche Attribute sind beispielsweise *author*, *description*, *keywords* und *title*. Andere Attribute können frei hinzugefügt werden. Die uneingeschränkte Erweiterbarkeit der Attribute wird für die strukturierte Abspeicherung der Datenbankinhalte genutzt.
Damit der Gatherer erkennen kann, daß es sich um eine einzubindende Datenbank handelt, wurde der neue Dokumenttyp `DB-Content` eingeführt. Solch ein Dokument muß auf dem WWW-Server, der den Zugriff auf die einzubindende Datenbank ermöglicht, vorhanden sein und enthält

[2] Die zwei Hauptkomponenten des Harvest-Systems werden als *Gatherer* und *Broker* bezeichnet. Gatherer sammeln Informationen über Dokumente, die im Internet verfügbar sind. Anschließend werden diese Informationen vom Broker indexiert und als Ergebnis auf Suchanfragen geliefert.

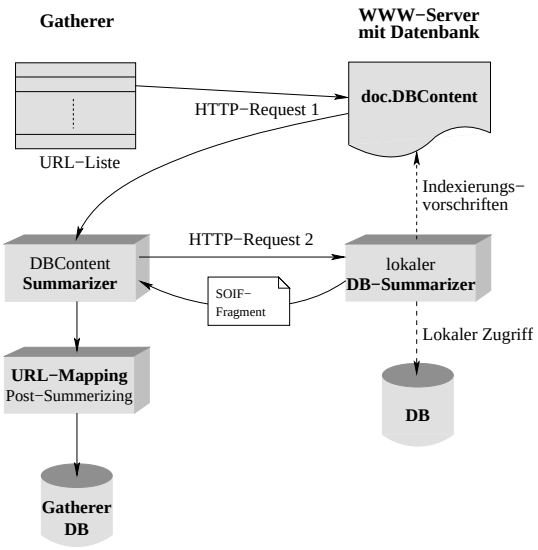

Abbildung 1: Integration von Datenbanken

die für die Integration notwendigen Informationen:

- Die Angabe *Summarizer* gibt Auskunft darüber, wo das Programm zur Extraktion der Datenbankinhalte auf dem Server zu finden ist.
- *URL* enthält die Adresse für das Anfrageformular.
- *Title*, *Description* und *Keywords* enthalten Informationen, die dem Anfrageformular zugeordnet werden.
- *Database* gibt den Namen der Datenbank an, die eingebunden werden soll.
- Über die Angaben *Table* und *Attribut* kann der Datenbank-Anbieter steuern, welche Attribute aus welcher Relation indexiert werden sollen. Dadurch wird sichergestellt, daß nur Inhalte, die öffentlich sein sollen, an die Suchmaschine weitergegeben werden.

In der Abbildung 1 sind die Schritte graphisch dargestellt, die vom Gatherer durchgeführt werden, um Datenbankinhalte in das Retrievalsystem der SWING-Suchmaschine einzuspeisen.
Die URLs für die Konfigurationsdateien müssen dem Gatherer explizit bekannt gemacht werden. Stößt der Gatherer bei der Abarbeitung der *URL-Liste* auf solch eine Datei, wird der *DBContent-Summarizer* aufgerufen. Dieser parst die Konfigurationsdatei und ruft dann die URL auf, die für den *lokalen Summarizer* angegeben wurde. Als Parameter wird der Pfad der Konfigurationsdatei relativ zum lokalen Summarizer übergeben. Über diesen Pfad kann der lokale Summarizer auf die Datei zugreifen, um die datenbankspezifischen Informationen zu lesen (Name der Datenbank, zu indexierende Tabellen und Attribute). Anschließend werden SQL-Anfragen zum Ermitteln der Werte für die zu indexierenden Attribute an die Datenbank gestellt. Im letzten Schritt faßt der lokale Summarizer alle Ergebnisse in einem *SOIF-Fragment* zusammen und schickt es an den Gatherer zurück. Dort wird es dann mit der URL für das Anfrageformular und weiteren Metainformationen (Größe des Fragmentes usw.) in der *Gatherer-Datenbank* gespeichert.
Nach dem Einsammeln der Daten liegen die Datenbankinhalte in strukturierter Form im Suchmaschinen-Index vor, so daß jederzeit feststellbar ist, in welchem Attribut ein Suchbegriff enthalten ist.

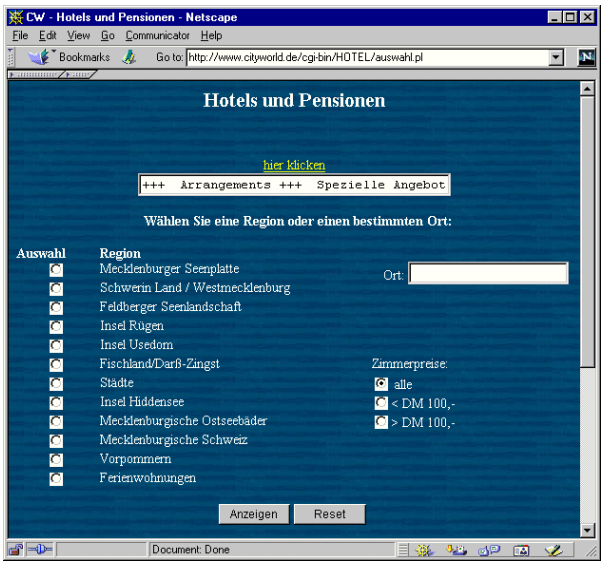

Abbildung 2: DB-Formular

Abbildung 3: Anfrageergebnis

In die SWING-Suchmaschine wurde beispielsweise der CityWorld-Hotelführer integriert, der auf einer Oracle-Datenbank basiert. Dieser Führer enthält eine Liste von Hotels und Pensionen in Mecklenburg-Vorpommern mit deren Adressen. Das Anfrageformular (siehe Abbildung 2) gestattet die Suche nach einer Region oder einem Ort, wobei das Ergebnis noch auf bestimmte Preiskategorien eingeschränkt werden kann. Als Ergebnis wird eine Tabelle (Abbildung 3) mit allen Hotels oder Pensionen geliefert, die den Suchkriterien entsprechen. Auf dem WWW-Server, über den der Zugriff auf diese Datenbank erfolgt, ist das Dokument `cityworld.DBContent` vorhanden, das folgende Informationen bereitstellt:

```
Summarizer:    http://www.cityworld.de/cgi-bin/Ora-DB.pl
URL:           http://www.cityworld.de/cgi-bin/auswahl.pl
Title:         CityWorld - Hotels und Pensionen
Description:   Hotelführer M-V
Keywords:      Hotel, Zimmer, Unterkunft, Pension, Motel, Preis
Database:      Cityworld
Table:         hotel
Attribute:     name
Attribute:     ort
Table:         region
Attribute:     region
```

Der Index für den Hotelführer enthält somit 3 Attribute, denen alle vorhandenen Hotels, Orte bzw. Regionen zugeordnet sind.
Jedes `DBContent`-Dokument muß explizit bei der Suchmaschine angemeldet werden[3], da der Gatherer den Zusammenhang zwischen der Web-Seite mit dem Anfrageformular und der Konfigurationsdatei nicht kennt. Damit der Gatherer solche Beziehungen erkennen kann, bieten sich Metadaten an, die im folgenden Abschnitt vorgestellt werden sollen.

[3]Ein Verlinken des Dokuments auf einer anderen Seiten ist zwar denkbar, aber nicht sinnvoll.

3 Metadaten für kooperative Datenbankanbieter

Metadaten dienen zur Definition bestimmter Eigenschaften eines HTML-Dokuments, die vom WWW-Browser bei der Darstellung nicht anzeigt werden. Suchmaschinen können diese Informationen interpretieren und sie beispielsweise bei der Bewertung des Dokuments bezüglich der Suchanfrage oder bei der Ausgabe des Suchergebnisses verwenden. Metadaten werden mit Hilfe von *Meta-Tags* angegeben, die folgende Syntax haben:

```
<META NAME="Meta-Name" CONTENT="Meta-Wert">
```

Diese Meta-Tags müssen im Kopf eines HTML-Dokuments stehen. Es existiert eine ganze Reihe von Meta-Tags, im Umfeld von Suchmaschinen sind die Angaben *description* und *keywords* am verbreitesten: *description* enthält eine Beschreibung der HTML-Seite, die bei der Ausgabe des Dokuments im Suchergebnis verwendet wird, und *keywords* nimmt Stichworte auf, unter denen das Dokument gefunden werden soll.
Das in Abschnitt 2 beschriebene Verfahren läßt sich durch die Definition folgender Metadaten im Anfrageformular noch weiter vereinfachen:

```
<META NAME= "DB.Summarizer" CONTENT="URL Summarizer">
<META NAME= "DB.ConfigFile" CONTENT="URL Konfigurationsdatei">
```

In diesem Fall wird kein eigener Dokumenttyp für Web-Datenbanken benötigt und das URL-Mapping entfällt ebenfalls. Die Angaben *URL* und *Summarizer* sind damit auch überflüssig. Für das CityWorld-Beispiel müßten somit folgende Metadaten definiert werden:

```
<META NAME="DB.Summarizer" CONTENT="http://www.cityworld.de/cgi-bin/ROra-DB.pl">
<META NAME="DB.ConfigFile" CONTENT="http://www.cityworld.de/cgi-bin/cityworld.txt">
```

Neben der erwähnten Vereinfachung des Verfahrens ergibt sich ein weiterer Vorteil: Die URL des Anfrageformulars braucht nicht mehr notwendigerweise bei der Suchmaschine angemeldet zu werden[4].

4 Datenbankeinbindung eingeschränkt kooperativer Anbieter

In vielen Fällen haben Anbieter von Web-Datenbanken Bedenken, fremde Programme auf ihren Rechnern zu installieren, die Inhalte nach außen geben. Deshalb soll im folgenden ein Mechanismus vorgestellt werden, der die normale Web-Schnittstelle für die Indexierung nutzt. Dieser Ansatz geht davon aus, daß der Anbieter mit Hilfe von Metadaten beschreibt, welche Funktionalität das Anfrageformular bietet und wie die Ergebnisseiten aufgebaut sind.
Im folgenden Unterabschnitt wird zunächst auf den Aufbau von Web-Formularen und die Probleme bei der automatischen Verarbeitung dieser Formulare eingegangen. Anschließend erfolgt dann eine detaillierte Beschreibung des Gatherers, der die Integration von Datenbanken über Metadaten gestattet. Zur Bestimmung der relevanten Inhalte auf den Ergebnisseiten sind Extraktionsregeln notwendig, die in Abschnitt 4.3 vorgestellt werden. Der letzte Unterabschnitt gibt einen Überblick über alle Metadaten, die zur Integration notwendig sind.

[4] Hier wird davon ausgegangen, das ein Link auf das Anfrageformular auf einer Seite existiert, die vom Gatherer eingesammelt wird.

4.1 Modellierung von Formularen

Ein Formular F ist eine Menge von *(Element, Domäne)*-Paaren

$$F = (E_1, D_1), (E_2, D_2), ..., (E_n, D_n),$$

wobei das Element E_i eines der folgenden Standardeingabeobjekte sein kann: Auswahlliste, ein- oder mehrzeiliges Textfeld, Radiobutton oder Checkbox. Die Domäne D_i bezeichnet die Menge aller Werte, die das zugehörge Element E_i annehmen kann. Einige Elemente haben *bestimmte Domänen*, d.h. die gültigen Werte sind in der HTML-Seite eingebettet (Auswahllisten, Radiobuttons, Checkboxen). Andere Elemente wie Textfelder haben *unbestimmte Domänen*, d.h. sie können beliebige Werte annehmen. Zusätzlich werden die meisten Elemente mit einem beschreibenden Text versehen, der dem Nutzer die Semantik des Elements verständlich machen soll. Dieser Text wird im folgenden als *Label* bezeichnet. Jedes Element hat einen internen Bezeichner, der bei der Übertragung der Formulardaten verwendet wird. Bei Radiobuttons bzw. Checkboxen korrespondiert zusätzlich jeder nach außen sichtbare Wert mit einem internen Bezeichner, bei Auswahllisten sind diese optional.
Für die Indexierung der Datenbankinhalte sind die Labels und Domänen relevant, bei der automatischen Ausführung der Formulare werden dagegen die internen Bezeichner benötigt.
Das automatische Extrahieren der Labels und Domänen (bei Radio- und Checkboxen) ist ein schwieriges Problem, da ihre Beziehung zum Formularelement nicht fest vorgegeben ist. Für den menschlichen Nutzer ist diese Zuordnung einfach durch die Entfernung des Labels bzw. des Wertes vom Formularelement zu erkennen, wenn das Anfrageinterface durch den Browser dargestellt wird. Im HTML-Text ist dieser Abstand nicht bestimmbar, da Formularelemente, Domänenwerte, Label und Layoutinformationen willkürlich ineinander geschachtelt werden können. Das Formular müßte somit visualisiert werden, damit ein Gatherer die richtigen Beziehungen über die Abstände ermitteln kann.
Für die Beantwortung von Anfragen sind einige Elemente wesentlich, d.h. die Angabe eines Wertes ist unbedingt erforderlich. Es gibt aber auch Elemente, die nicht spezifiziert werden müssen und nur eine Reduzierung der Ergebnismenge bewirken. Die automatische Bestimmung der wesentlichen Attribute ist kein triviales Problem, auf das hier auch nicht näher eingegangen werden soll.

4.2 Extrahieren der Datenbankinhalte

Die grundlegenden Aktionen des Gatherers, der das Einsammeln von Datenbanken eingeschränkt kooperativer Anbieter erlaubt, ähneln denen eines traditionellen Gatherers: Auswahl einer URL, Anforderung der Seite und anschließend die Verarbeitung dieser Seite mit der Extraktion neuer Links. Diese Aktionen laufen innerhalb einer Schleife ab, die dann beendet ist, wenn keine URLs mehr abzuarbeiten oder bestimmte Abbruchbedingungen (z.B. Erreichen der maximalen Anzahl von Dokumenten, die eingesammelt werden sollen) eingetreten sind. Unterschiede in der Abarbeitung treten erst auf, wenn der Gatherer auf ein Dokument stößt, das datenbankspezifische Metadaten enthält.
Die Abbildung 4 zeigt die grundlegende Arbeitsweise des Gatherers. Der *Gathering-Manager* steuert den gesamten Prozeß der Datensammlung. Er entscheidet, welcher Link als nächstes bearbeitet werden soll und fordert dann das zugehörige Dokument an. Dieses Dokument wird an den *Parser* weitergegeben, der inhaltliche Informationen ermittelt und diese zusammen mit Informationen über das Dokument selbst (URL, Größe, usw.) in strukturierter Form (z.B. SOIF)

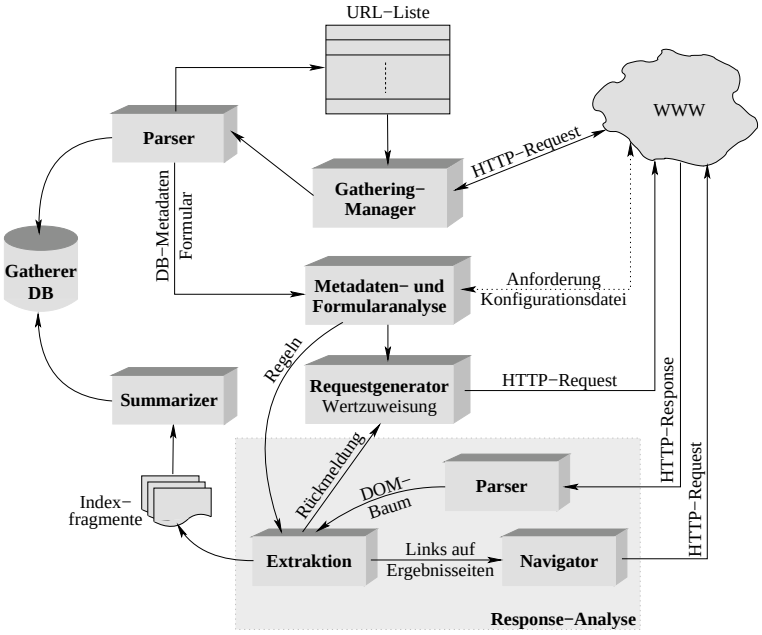

Abbildung 4: Arbeitsweise des Gatherers

in der *Gatherer-Datenbank* ablegt. Die extrahierten Links werden in die *URL-Liste* eingefügt und stehen somit dem Gathering-Manager zur Verfügung. Enthält das Dokument datenbankspezifische Metadaten, handelt es sich um ein Anfrageformular für eine Web-Datenbank. Der Gatherer unternimmt folgende Schritte, um die Inhalte der Datenbank zu bestimmen:

- *Metadaten- und Formularanalyse*: Die vom Parser übergebenen DB-Metadaten und das Anfrageformular werden zunächst zerlegt und analysiert. Wird in den Metadaten auf eine Konfigurationsdatei verwiesen, dann muß diese über einen zusätzlichen HTTP-Request angefordert werden. Für das Formular sind folgende Information zu bestimmen:
  - die URL des Skripts, das die eingegebenen Daten entgegennimmt und die Antwort generiert sowie die Methode (GET oder POST), mit der die Parameter übergeben werden,
  - alle Elemente und deren interne Bezeichner sowie der Typ des Elements und
  - für alle Elemente mit bestimmten Domänen (Auswahllisten, Radiobuttons und Checkboxen) die internen Bezeichner[5] der möglichen Werte, bei Auswahllisten zusätzlich die nach außen sichtbaren Werte (Domäne).

  Nach der Analyse des Formulars kann ein Abgleich mit den Metadaten vorgenommen werden, wobei folgende Bedingungen zu prüfen sind: die in den Metadaten spezifizierten Anfrageelemente müssen im Formular vorkommen, bei Anfrageelementen mit unbestimmten Domänen muß diese definiert sein und die Extraktionsregeln müssen der vorgegebenen Syntax entsprechen.

- *Request-Generation*: Bei der Erzeugung der Anfragen werden nicht alle Formularelemente berücksichtigt, sondern nur diejenigen, die vom Datenbankanbieter als wesentlich spezifiziert wurden. Ist nur das Element E_i mit der Domäne D_i wesentlich, dann werden

[5] Wurden für die Elemente einer Auswahlliste keine internen Bezeichner definiert, dann werden die nach außen sichtbaren Werte genommen.

$|D_i|$ Anfragen erzeugt. Bei $n > 1$ Elementen wird das kartesische Produkt der Domänen $D_1 \times D_2 \times ... \times D_n$ gebildet, so daß insgesamt $|D_1| * |D_2| * ... * |D_n|$ Anfragen an die Datenbank gestellt werden müssen. Bei unbestimmten Domänen, die mit Hilfe regulärer Ausdrücke definiert wurden (siehe Abschnitt 4.4), ist zusätzlich noch eine Erzeugung der möglichen Werte notwendig.

Der *Request-Generator* ist weiterhin dafür verantwortlich, das die erzeugten Anfragen nacheinander an die Datenbank geschickt werden. Dies geschieht solange, bis alle Anfragen abgearbeitet wurden oder eine bestimmte Anzahl von Requests (Abbruchkriterium) erreicht ist.

- *Response-Analyse*: Die Bestimmung der Inhalte auf der Ergebnisseite erfolgt mit Hilfe der in den DB-Metadaten enthaltenen Extraktionsregeln. Einzelheiten zu diesen Regeln können in Abschnitt 4.3 nachgelesen werden. Die Ergebnisseite dient als Eingabe für einen *HTML-Parser*, der daraus eine abstrakte Darstellung (HTML-Baum oder Analysebaum) erzeugt. Innerhalb dieses Analysebaums kann jedes Element im Dokument anhand einer Extraktionsregel eindeutig identifiziert werden. Für jede Antwortseite entsteht so ein Indexfragment, das die extrahierten Inhalte zusammenfaßt. Ist das Ergebnis über mehrere HTML-Seiten verteilt, müssen diese ebenfalls angefordert werden. Dafür ist der *Navigator* zuständig, der alle Links auf Ergebnisseiten sammelt und nacheinander abarbeitet. Für jede Ergebnisseite muß dann der gesamte Prozeß der Response-Analyse wieder durchlaufen werden.

- *Zusammenfassung*: Der *Summarizer* faßt die Indexfragmente, die pro Anfrage erzeugt wurden, zusammen. Doppelte Werte für ein Attribut werden dabei eleminiert.

Die extrahierten Datenbankinhalte werden zum Schluß gemeinsam mit dem Anfrageformular in der Gatherer-Datenbank abgelegt.
Im folgenden Abschnitt soll näher auf die Extraktionsregeln eingegangen werden, die eigentlich für das Wrapper-Toolkit *W4F* entwickelt wurden. Es existiert eine ganze Reihe solcher Werkzeuge wie *WebL* oder *JEDI*, die ebenfalls Mechanismen zur Extraktion von Daten anbieten [Goh00]. Im Gegensatz zu W4F werden hier aber Skriptsprachen eingesetzt, die das Beschreiben der Ergebnisseiten erschweren.

4.3 W4F-Extraktionsregeln

W4F[6] [SA98, SA00] ist ein Werkzeug zur Generation von Wrappern für Web-Datenquellen. Zur Definition robuster Extraktionregeln stellt W4F mit *HEL* (***H****TML* ***E****xtraction* ***L****anguage*) eine deklarative Sprache zur Verfügung. Ein *Extraktions-Wizard* hilft bei der Erstellung dieser Regeln. Er generiert beispielsweise den identifizierenden Pfadausdruck für ein Textelement der HTML-Seite, wenn es vom Nutzer im Browser selektiert wird. Daraus leitet sich das *wysiwyg* (**w**hat **y**ou **s**ee textbfis **w**hat **y**ou **g**et) im Namen ab. Durch den deklarativen Charakter der Sprache und die Unterstützung bei der Erstellung der Regeln eignet sich HEL sehr gut für die Analyse der Ergebnisseiten.
Als Grundlage für die Definition der Regeln dient der HTML-Baum oder Analysebaum, der aus dem HTML-Dokument erstellt wird. Die Zuordnung von HTML-Dokument und Analysebaum ist eineindeutig, d.h. dasselbe HTML-Dokument ist wieder aus dem Analysebaum ermittelbar und umgekehrt.

[6] WysiWyg Web Wrapper Factory

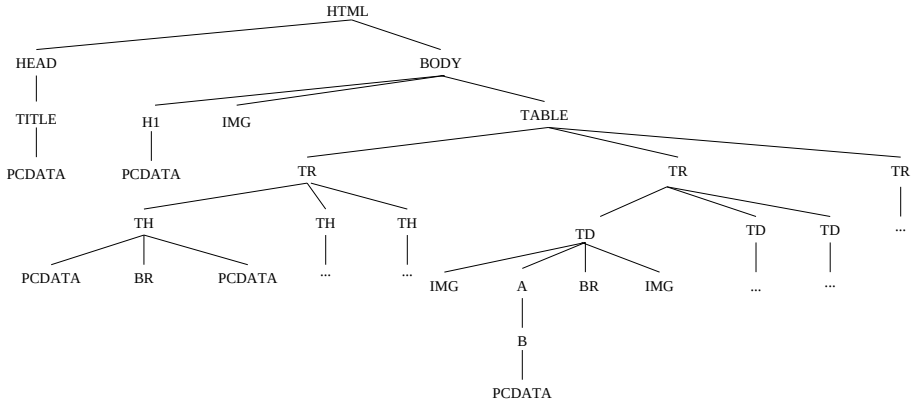

Abbildung 5: Beispiel eines HTML-Baums

Der Baum besteht aus einer Wurzel (Bezeichner `html`), internen Knoten und Blättern. Jeder Knoten korrespondiert mit einem HTML-Tag oder Textstück (Bezeichner `PCDATA`). Interne Knoten repräsentieren die geschlossenen HTML-Tags[7], und haben Kinder, auf die über den Bezeichner (Name des zugehörigen HTML-Tags) und den Index[8] (die Reihenfolge des Auftretens) zugegriffen werden kann. Blätter sind entweder offene HTML-Tags (z.B. `img` oder `br`) oder `PCDATA`-Knoten.
In Abbildung 5 ist zur Veranschaulichung der Analysebaum für das folgende HTML-Dokument abgebildet:

```
<HTML>
 <HEAD><TITLE>Text</TITLE> </HEAD>
 <BODY>
  <H1>Text</H1>
  <IMG>
  <TABLE>
   <TR>
    <TH>Text<BR>Text</TH>
    <TH> ... </TH>
    <TH> ... </TH>
   </TR>
   <TR>
    <TD><IMG><A><B>Text</B></A><BR><IMG></TD>
    <TD> ... </TD>
    <TD> ... </TD>
   </TR>
   <TR>
    ...
   </TR>
  </TABLE>
 </BODY>
</HTML>
```

[7]Geschlossene HTML-Tags sind alle HTML-Sprachelemente, die durch eine Start- und Endmarke gebildet werden, z. B. `<table>...</table>`.

[8]Die Vergabe der Indizes geschieht nach dem *left-depth-first-Prinzip*, d.h. die Durchnummerierung gleichnamiger Knoten beginnt zuerst links absteigend im Baum.

Die Navigation entlang des abstrakten Baumes geschieht über Pfadausdrücke. Zur Bildung dieser Ausdrücke stehen zwei Navigationsansätze zur Verfügung:

- Der *Punkt-Operator* nutzt die Baumhierarchie aus, d.h. hinter jedem Element wird getrennt durch den Punkt-Operator ein unmittelbarer Kindknoten im Baum angegeben. Damit läßt sich zu jedem Knoten genau ein Pfadausdruck angeben. Der Pfad `html.head.title` führt z.B. zu dem Knoten, der mit dem `<Title>`-Tag im Kopf des Dokuments korrespondiert.

- Der *Pfeil-Operator* orientiert sich am Dokumentfluß. Das Durchlaufen des Baumes geschieht hier nach dem *depth-first-Prinzip*. Beispielsweise führt der Pfad `html->table[0]` zur ersten Tabelle im Dokument. Die Navigation erfolgt hier über die Hierarchie des Baumes hinweg, so daß z.B. von einem Blatt zu einem anderen Knoten innerhalb des Baumes gesprungen werden kann. Der Operator erweitert die Ausdrucksfähigkeit von HEL beträchtlich, da so auch irreguläre Strukturen verarbeitet werden können.

Als Index für ein Pfadelement können auch Intervalle oder Wildcards verwendet werden, so daß mehrere Knoten das Ergebnis bilden. Beispielsweise liefert `html.body->a[*]` alle Anker des HTML-Dokuments.
Die Extraktionsregeln sind nicht nur auf die Knoten selbst beschränkt, sondern sie können auch auf die Informationen zugreifen, die diese tragen. Jeder Knoten hat ein zugeordnetes Textattribut `.txt`. Bei Blättern liefert dieses Attribut im Falle von `PCDATA` das zugehörige Textstück. Bei internen Knoten wird der Wert des Textattributs aus der rekursiven Verkettung aller Unterknoten gebildet. Der Ausdruck `html.body.table[0].tr[0].th[0].txt` ermittelt z.B. die Überschrift der ersten Tabellenspalte. Auf die Eigenschaften der Knoten wie die Werte der Attribute oder die Anzahl der Kinder kann über die Funktionen `getAttr` oder `numberOf` zugegriffen werden. So liefert der Pfadausdruck `html.body->a[0].getAttr(href)` beispielsweise die URL des ersten Links im HTML-Dokument.
Zur Extraktion von Informationen reicht die HTML-Struktur, die durch den Analysebaum bereitgestellt wird, nicht immer aus. Dieser Fall tritt z.B. auf, wenn auf einzelne Elemente zugegriffen werden soll, die innerhalb einer Tabellenspalte aufgezählt sind. HEL stellt für solche Fälle die Operatoren `match` und `split` bereit, die mit regulären Ausdrücken arbeiten.
Für die Extraktion von Strukturen steht der fork-Operator `#` zur Verfügung. Er erlaubt die Aufspaltung eines Pfadausdruckes in mehrere Unterausdrücke und gruppiert die Teilergebnisse zu einem Gesamtergebnis. Dies ist beispielsweise nützlich, wenn Vor- und Nachname einer Person aus verscheidenen Tabellenspalten extrahiert und zum kompletten Namen zusammengefaßt werden sollen. Die Regel `html.body->a[*](.txt#.getAttr(href))` ermittelt z.B. alle Links in einem HTML-Dokument mit Beschreibung und URL.
Zur Definition robuster Extraktionsregeln können Bedingungen formuliert werden. Statt fester Indizes stehen dann im Pfadausdruck Variablen, deren Wert jeweils durch eine Bedingung bestimmt wird. Für jede eingeführte Variable muß genau eine Bedingung in der `WHERE`-Klausel angegeben sein, die Verknüpfung erfolgt konjunktiv über das Schlüsselwort `AND`. Die Bedingungen können nicht die Knoten selber einbeziehen, sondern nur deren Werte und verschiedene Vergleichsoperatoren, die in HEL erlaubt sind.
Zur Illustration der Ausdruckskraft von HEL-Bedingungen soll folgendes Beispiel dienen: In der in Abbildung 6 gezeigten Währungsumrechnungstabelle sollen Informationen extrahiert werden, die über eine Spalte und eine Zeile identifizierbar sind. Speziell wird nach einer Zelle gesucht, die mit der Spalte “DM” und der Zeile “US” korrespondiert, um den Umtauschkurs zwischen

Currency	U.S. $	Aust $	U.K. Pound	DMark	FFranc
U.S. $	1	0.6244	1.657	0.6024	0.1796
Aust $	1.602	1	2.654	0.9648	0.2877
U.K. Pound	0.6034	0.3768	1	0.3635	0.1084

Abbildung 6: Umrechnungstabelle für Beispiel-Extraktionsregel

D-Mark und Dollar zu ermitteln. Nachstehende Extraktionsregel bestimmt diesen Kurs:

```
value = html.body->table[1].tr[i].td[j].txt
WHERE   html.body->table[1].tr[0].th[j].txt = 'DM'
AND     html.body->table[1].tr[i].th[0].txt = 'U.S.';
```

Zeile und Spalte werden in dieser Regel variabel gehalten, damit sich Änderungen an der Tabelle nicht auf das Ergebnis auswirken, wie z.B. das Hinzufügen einer neuen Währung. Die erste Spaltenüberschrift, die den String "DM" enthält, legt die Variable `j` fest, der erste Zeilenkopf mit der Zeichenkette "U.S." die Variable `i`.

4.4 Notwendige Metadaten

In der Tabelle 7 sind alle Metadaten aufgelistet, die zur Einbindung von Datenbanken über Web-Formulare benötigt werden. *Kursiv* hervorgehobene Anteile stehen für Terminale, Nichtterminale sind durch die Zeichen < und > geklammert. Geschweifte Klammern dienen zur Kennzeichnung von Bestandteilen, die null- oder mehrmalig auftreten können und Alternativen werden durch einen vertikalen Strich getrennt.

Bezeichner	Wert
ConfigFile	<URL_Konfigurationsdatei>
QueryScript	<URL_Script>, *GET* | *POST*
QueryElement *QueryElements*	<Element> <Element>, <Element> {, <Element> }
QueryElement.<Element>	/<Einfacher_regulaerer_Ausdruck>/ oder <Wert> {, <Wert> }
MaxQueries	<Anzahl>
MaxPages	<Anzahl>
Result	<gemeinsamer_Teil_einer_W4F_Extraktionsregel>
Result.<Attributname>	<W4F_Extraktionsregel> { <Attributwert> {, <Attributwert> } }
NavigationURL	<URL_mit_Platzhalter_fuer_Parameter>

Abbildung 7: Notwendige Metadaten für die Datenbankeinbindung

Zur Unterscheidung von anderen Metadaten müssen datenbankspezifischen Informationen immer mit den Initialen `DB` beginnen. Im folgenden werden die einzelnen Metadaten detailliert beschrieben:

- *ConfigFile*: Alle datenbankspezifischen Metainformationen können in einer Konfigurationsdatei zusammengefaßt werden, die unter der angegebenen URL abgelegt ist. Dies hat

folgenden Vorteil: das Verzeichnis, in dem diese Datei liegt, kann nur für spezielle Suchmaschinen frei gegeben werden (htaccess-Datei), so daß ein kontrollierter Zugriff auf diese Informationen gewährleistet ist.

- *QueryScript*: Hier kann der Anbieter ein Skript für die Abarbeitung der Anfragen angeben, das anstelle des im Formular spezifizierten Programms verwendet werden soll. Dieses Skript könnte beispielsweise einfachere Anfragen zulassen oder Ergebnisseiten liefern, die leichter zu analysieren sind. Außerdem werden dem Anbieter so Layoutänderungen erleichtert: er kann die alten Extraktionsregeln weiter nutzen und einfach auf das Skript verweisen, das die alten Ergebnisseiten generiert.

- *QueryElement(s)*: Ein *QueryElement* ist ein für die Anfrage wesentliches Formularelement. Sind mehrere Elemente für eine Anfrage notwendig, dann können diese durch Kommata getrennt in *QueryElements* angegeben werden. Für die Elemente ist der interne Bezeichner zu verwenden. Bei mehreren Elementen ergeben sich die möglichen Anfragebelegungen aus dem kartesischen Produkt der zugeordneten Domänen. Deshalb sollten die notwendigen Anfrageelemente so gewählt werden, daß der Inhalt der Datenbank mit einer minimalen Anzahl von Anfragen bestimmbar ist. Sind z.B. in einem Formular 2 Elemente A und B mit bestimmter Domäne enthalten und B schränkt das Anfrageergebnis von A ein, dann ist nur A anzugeben[9]. Ansonsten müßten alle Kombinationen von A und B angefragt werden, um den gleichen Inhalt zu ermitteln. Weiterhin sollten möglichst Elemente mit bestimmter Domäne verwendet werden, da unbestimmte Domänen nur sehr eingeschränkt beschreibbar sind.

- *QueryElement.<Element>*: Innerhalb einer Domnen knnen allgemeinere und speziellere Attributwerte auftreten. Die Verwendung eines spezielleren Werts macht in diesem Fall keinen Sinn, da die Ergebnismenge eine Untermenge des Ergebnisses ist, das auf eine Anfrage mit dem zugehrigen allgemeineren Wert geliefert wird[10]. Über diesen Bezeichner können die allgemeineren Attributwerte für eine bestimmte Domäne spezifiziert werden, da eine automatische Erkennung dieser Werte ohne lexikalische Analyse nicht möglich ist.

  Außerdem dient dieser Bezeichner zur Beschreibung von Formularelementen mit unbestimmter Domäne, die in Anfragen benötigt werden. Die meisten unbestimmten Domänen lassen sich leider nicht über reguläre Ausdrücke beschreiben. Die Zuhilfenahme von Ontologien zur Bestimmung möglicher Werte ist aber sehr problematisch, so daß in diesem Ansatz versucht wird, zumindest einen Teil unbestimmter Domänen mit regulären Ausdrücken zu erfassen. Viele Web-Anfrageschnittstellen lassen bei der Eingabe Teilausdrücke zu, etwa bei der Suche nach einer Stadt, wo nur die Anfangsbuchstaben angegeben werden müssen, z.B. 'Ros'. Im Ergebnis werden dann alle Städte berücksichtigt, die mit diesen Buchstaben beginnen, wie 'Rosenheim', 'Rostock' usw.. Diese Teilmuster können über reguläre Ausdrücke dargestellt werden.

  Ein weiteres Problem ist, das hier kein Vergleich von vorhandenen Zeichenketten mit den regulären Ausdrücken stattfindet, sondern das mit Hilfe dieser Ausdrücke mögliche Werte erzeugt werden sollen. Aus diesem Grund sind hier auch nur sehr einfache reguläre Ausdrücke gestattet, mit denen eine endliche Menge von Werten beschrieben werden kann. Bei der Definition von Mustern dürfen folgende Zeichen auftreten:

[9] Voraussetzung ist, daß die Web-Schnittstelle das Weglassen des Wertes für Element B erlaubt.
[10] Hier wird davon von einer vollstndigen Ergebnismenge ausgegangen.

?	null- oder einmaliges Auftreten eines Teilausdrucks
+	ein- oder mehrmaliges Auftreten eines Teilausdrucks
*	null- oder mehrmaliges Auftreten eines Teilausdrucks
{N,M}	Minimum und Maximum des Auftretens (zwingend für +, *)
()	Gruppierung von Teilausdrücken
[]	Klasse von Zeichen (entweder Aufzählung oder Zeichenbereiche)
|	Alternativen

Der reguläre Ausdruck '[A-Z]%' wird vom Gatherer beispielsweise expandiert zu 'A%', 'B%', ..., 'Z%', wobei % in diesem Fall für einen erlaubten Platzhalter für das Formularelement steht. Eine weitere Schwierigkeit mit unbestimmten Domänen tritt auf, wenn eine Anfrageschnittstelle zwar Teilausdrücke bei der Suche erlaubt, aber im nächsten Schritt erwartet, daß der Suchbegriff präzisiert wird. Dieses Problem wird hier aber nicht berücksichtigt[11].

Für unbestimmte Domänen ist auch ein Verweis auf eine andere Seite, die alle möglichen Werte der Domäne enthält, denkbar: In diesem Fall wird dann aber auch eine Extraktionsregel wie bei der Beschreibung der Ergebnisseiten notwendig.

- *MaxQueries*: Der Datenbankanbieter kann über diesen Bezeichner die maximale Anzahl der Anfragen, die vom Gatherer durchgeführt werden sollen, vorgeben. Ist die maximale Anzahl kleiner als die vom Gatherer eigentlich benötigte Anzahl von Anfragen, dann kann der Inhalt der Datenbank nicht vollständig bestimmt werden.

- *MaxPages*: Ein Anfrageergebnis kann über mehrere Seiten verteilt sein, die dann vom Gatherer nacheinander eingesammelt werden müssen. Mit *MaxPages* kann ein Anbieter die Anzahl der Ergebnisseiten, die vom Gatherer pro Anfrage maximal geholt werden sollen, spezifizieren.

- *Result*: Kommen mehrere Ergebnisattribute an einer bestimmten Stelle im Dokument vor, dann kann der gemeinsame Teil der Pfadausdrücke in den einzelnen Extraktionsregeln weggelassen und in diesem Bezeichner angegeben werden. Dies ist z.B. nützlich, wenn alle Attribute in einer Tabelle vorkommen: der Pfad für die Tabelle wird mit *Result* definiert, so daß die Regeln für die Attribute dann alle relativ zu dieser Angabe gelten. Der gemeinsame Teil gilt für die Regeln aller nachfolgenden Ergebnisattribute, die nicht mit dem Wurzelelement `html` beginnen.

  Der Bezeichner kann mehrfach angegeben werden, der neue Wert überschreibt dann einfach die alte Angabe.

- *Result.<Attributname>*: Für jedes Attribut, dessen Werte bestimmt werden sollen, ist eine W4F-Extraktionsregel anzugeben. In Abschnitt 4.3 befindet sich eine detaillierte Beschreibung dieser Regeln. Für bestimmte Domänen, bei denen sich die nach außen sichtbaren Werte schlecht ermitteln lassen (Radiobuttons, Checkboxen), können auch alle möglichen Werte durch Kommata getrennt aufgelistet werden.

- *NavigationURL*: Die Angabe der URL, über die eine Navigation durch die Ergebnisseiten erfolgt, mit dem Platzhalter * für die sich ändernden Parameter gewährleistet das korrekte Bestimmen der Ergebnismenge. Ansonsten muß der Gatherer alle Links auf der

[11] In diesem Fall müßten mit Hilfe von Metadaten auch noch Arbeitsabläufe beschrieben werden.

Ergebnisseite verfolgen oder über Heuristiken[12] versuchen, die weiteren Ergebnisseiten zu ermitteln.

In Abschnitt 2 wurde die Einbindung des CityWorld-Hotelführers für den kooperativen Fall gezeigt. Dieser soll jetzt über Metadaten eingebunden werden. Das Anfrageformular des Hotelführers (siehe Abbildung 2) hat drei Eingabefelder: Region (interner Bezeicher `region`), Ort (`ort`) und Zimmerpreis (`preis`). Die Felder Region und Preis haben eine bestimmte Domäne, für den Ort können beliebige Angabe gemacht werden. Das wesentliche Anfrageelement ist die Region. Der Ort wäre zwar auch ein Kandidat, kommt aber durch die unbestimmte Domäne nicht in Frage. Die Angabe eines Zimmerpreises schränkt nur die jeweilige Ergebnismenge ein. Somit läßt sich der gesamte Datenbankinhalt durch das Abfragen der einzelnen Regionen ermitteln. Aus diesen Inhalten sollen die Attribute Region, Ort und Hotel- bzw. Pensionsname extrahiert werden. Die automatische Bestimmung der Domäne für das Feld Region im Anfrageformular ist nicht möglich, da es sich um Radiobuttons handelt. Für die Ermittlung dieser Domäne sind 2 Varianten möglich: entweder werden die Werte aus den Ergebnisseiten extrahiert oder es erfolgt eine Auflistung der Werte im entsprechenden Meta-Tag. Bei den Attributen Ort und Name gibt es keine Wahl: hier ist nur die Angabe von Extraktionsregeln erlaubt. Zur Einbindung des Hotelführers müssen folgende Metadaten angegeben werden:

```
<META NAME="DB.QueryElement" CONTENT="region">
<META NAME="DB.Result.region" CONTENT="Fischland,..">
<META NAME="DB.Result.ort" CONTENT='html.body->table[1].tr[i:*].td[1].txt
 where html.body->table[1].tr[i].td[1].font.getAttr(color) = "#FF8000"'>
<META NAME="DB.Result.name" CONTENT='html.body->table[1].tr[j:*].td[1].b.txt
 where html.body->table[1].tr[j].td[0].small.input.getAttr(name) = "hotel_id"'>
<META NAME="DB.Navigation" CONTENT="NO">
```

oder:

```
<META NAME="DB.QueryElement" CONTENT="region">
<META NAME="DB.Result.region" CONTENT="html.body->A[i]->pcdata[1].txt
 where html.body->A[i].getAttr(name) = 'Seitenanfang'">
<META NAME="DB.Result" CONTENT="html.body->table[1].">
<META NAME="DB.Result.ort" CONTENT="tr[j:*].td[1].txt
 where tr[j].td[1].font.getAttr(color) = '#FF8000'">
<META NAME="DB.Result.name" CONTENT="tr[k:*].td[1].b.txt
 where tr[k].td[0].small.input.getAttr(name) = 'hotel_id'">
<META NAME="DB.Navigation" CONTENT="NO">
```

Bei der ersten Variante werden die Werte für die Region aufgelistet, bei der zweiten erfolgt die Bestimmung der Domäne aus dem Anfrageergebnis. Die Extraktionsregeln sind so definiert, daß sie bei kleineren Layoutänderungen nicht geändert werden müssen. Eine Ortsangabe im Ergebnis ist farbig hervorgehoben. Dies wird bei der Bestimmung von Orten ausgenutzt, da nur Angaben in der 2. Spalte als Ort anerkannt werden, die eine solche Hervorhebung haben. Hotels und Pensionen sind im Ergebnis mit einer Checkbox versehen, um Reservierunganfragen an diese zu schicken. Bei der Extraktion von Hotels und Pensionen werden deshalb solche Zeilen gesucht, die in der ersten Spalte eine Checkbox enthalten. Das Ergebnis umfaßt den Namen eines Hotels bzw. einer Pension und weitere Informationen. Da der Name fett hervorgehoben ist, kann er von den übrigen Informationen getrennt werden.

[12] Das verfolgen aller Links, die mit Zahlen bezeichnet sind, wäre eine mögliche Heuristik.

5 Zusammenfassung und Ausblick

In diesem Artikel wurden zwei Ansätze zur Indexierung von Datenbankinhalten durch Suchmaschinen vorgestellt, die von einer unterschiedlichen Kooperationsbereitschaft des Datenbankanbieters ausgehen. Der erste in der Suchmaschine SWING realisierte Ansatz setzt die volle Kooperationsbereitschaft des Anbieters voraus, da ein lokaler Summerizer installiert werden muß. Dieser Mechanismus ist sehr effizient, da nur zwei HTTP-Requests für die Indexierung notwendig sind. Viele Anbieter scheuen sich aber aus unterschiedlichen Gründen davor, ein solches Programm auf ihrem Server zu installieren. Aus diesem Grund wurde ein zweiter Ansatz vorgeschlagen, der die Indexierung der Datenbankinhalte über die normale Anfrageschnittstelle erlaubt. Hier beschreibt der Anbieter über Metadaten, welche Funktionalität diese Schnittstelle bietet und wie die Ergebnisseiten aufgebaut sind. Der Gatherer nutzt die Metadaten für die Generierung von Anfragen und die Extraktion der Attribute, die indexiert werden sollen. Zwei Hauptprobleme treten bei diesem Mechanismus auf: zum einen die Beschreibung unbestimmter Domänen und zum anderen die Vielzahl der Anfragen, die an die Datenbank gestellt werden müssen. Zur Definition unbestimmter Domänen wird momentan auf reguläre Ausdrücke zurückgegriffen. Somit lassen sich nur Domänen beschreiben, die ein bestimmtes Format voraussetzen oder bei denen in der Anfrage einfache Teilmuster verwendet werden können. Weitere ungelöste Probleme sind Interaktionen im Anfrageprozeß bzw. abgeschnittene Ergebnislisten bei zu vielen Treffern.
Als nächster Schritt ist die prototypische Implementierung des Gatherers vorgesehen, um Aussagen über die Leistungsfähigkeit des Ansatzes geben zu können. Dabei ist vor allem interessant, wie sich das Zeitverhalten bei steigendem Datenvolumen und bei steigender Komplexität der Anfrageschnittstelle entwickelt. Dann kann auch die Frage geklärt werden, ob die Einbindung von Datenbanken überhaupt Sinn macht, wenn Formularelemente mit unbestimmten Domänen beim Abfragen der Datenbankinhalte berücksichtigt werden müssen.

Literatur

[B+00] Michael K. Bergmann et al. The Deep Web: Surfacing Hidden Value. Whitepaper, BrightPlanet.com LCC, July 2000. Available at `http://www.completeplanet.com/tutorials/deepweb/`.

[Bru01] Ilvio Bruder. Zugriff auf dynamische Web-Dokumente mittels Web-Mining-Analysetechniken. Preprint CS-04-01, Fachbereich Informatik, Universität Rostock, March 2001.

[Goh00] Rolf Gohla. Integrierte WWW-Anfragesichten. Master's thesis, Fachbereich Informatik, Universität Rostock, 2000.

[HSW96] Darren R. Hardy, Michael F. Schwartz, and Duane Wessels. Harvest User's Manual. Technical Report CU-CS-743-94, University of Colorado, Boulder, January 1996. Available at `http://www.tardis.ed.ac.uk/harvest/docs/`.

[HW00] Andreas Heuer and Gunnar Weber. Swing: Eine Suchmaschine mit Datenbankanschluß. In *Workshop Internet-Datenbanken*, number 12 (Preprint), Magdeburg, September 2000. Fakultät für Informatik, Otto-von-Guericke Universität.

[LG98] Steve Lawrence and C. Lee Giles. Searching the World Wide Web. *Science*, 280(5360):98–100, 1998.

[RGM00] Sriram Raghavan and Hector Garcia-Molina. Crawling the Hidden Web. Technical report, Stanford University, November 2000.

[SA98] Arnaud Sahuguet and Fabien Azavant. WysiWyg Web Wrapper Factory (W4F). Technical report, University of Pennsylvania und Telecom Paris (E.N.S.T.), 1998. Available at `http://db.cis.upenn.edu/W4F/`.

[SA00] Arnaud Sahuguet and Fabien Azavant. *World Wide Web Wrapper Factory: User Manual*, 2000. Available at `http://db.cis.upenn.edu/W4F/`.