Seminar XML und Datenbanken

Indexstrukturen in XML

Vanessa Schäfer

07.02.2003

Übersicht

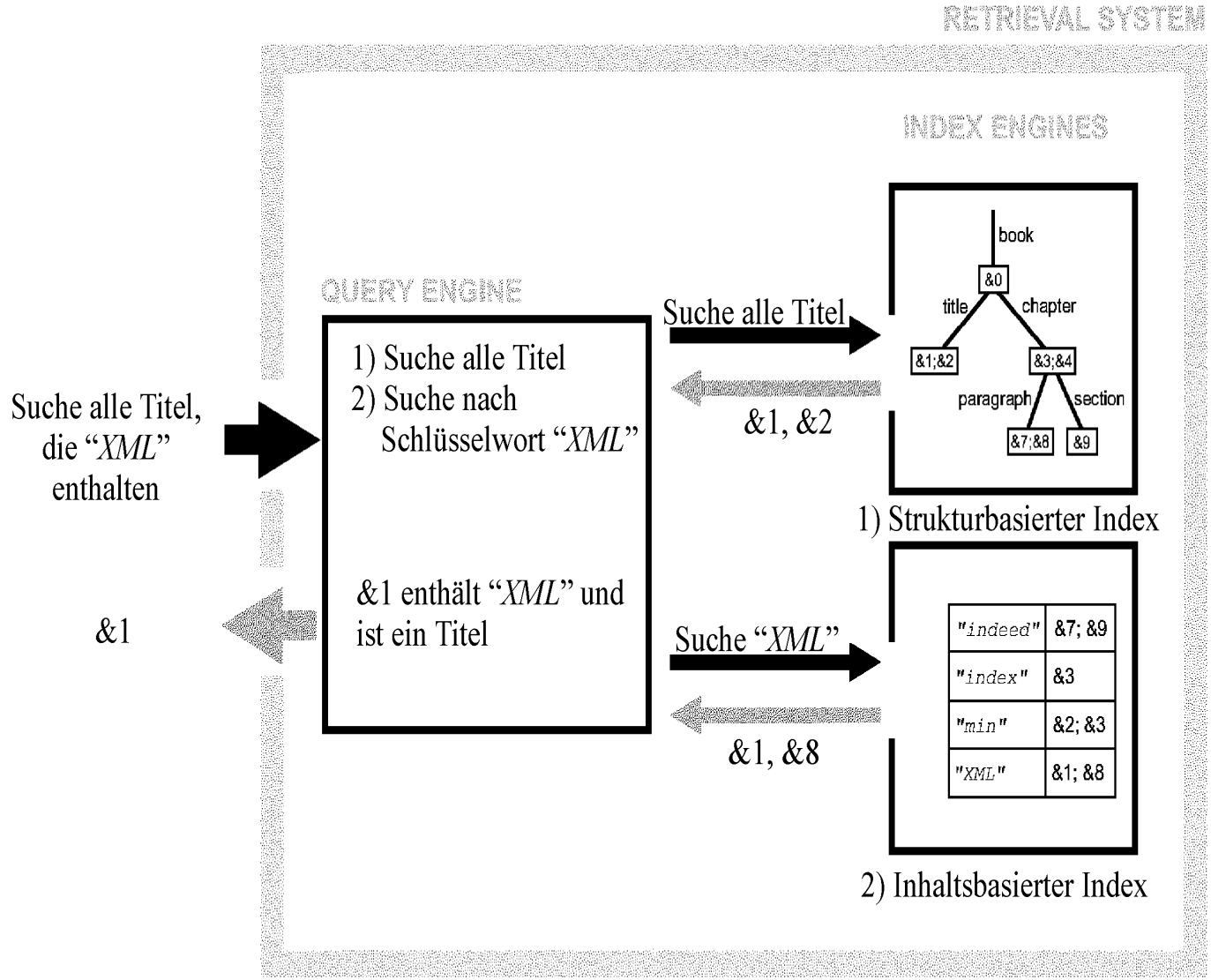
- Einführung
- Indexstrukturen in XML
- Ein Vergleich SphinX vs. Lore
- Zusammenfassung und Ausblick

Einführung – Was ist ein Index?

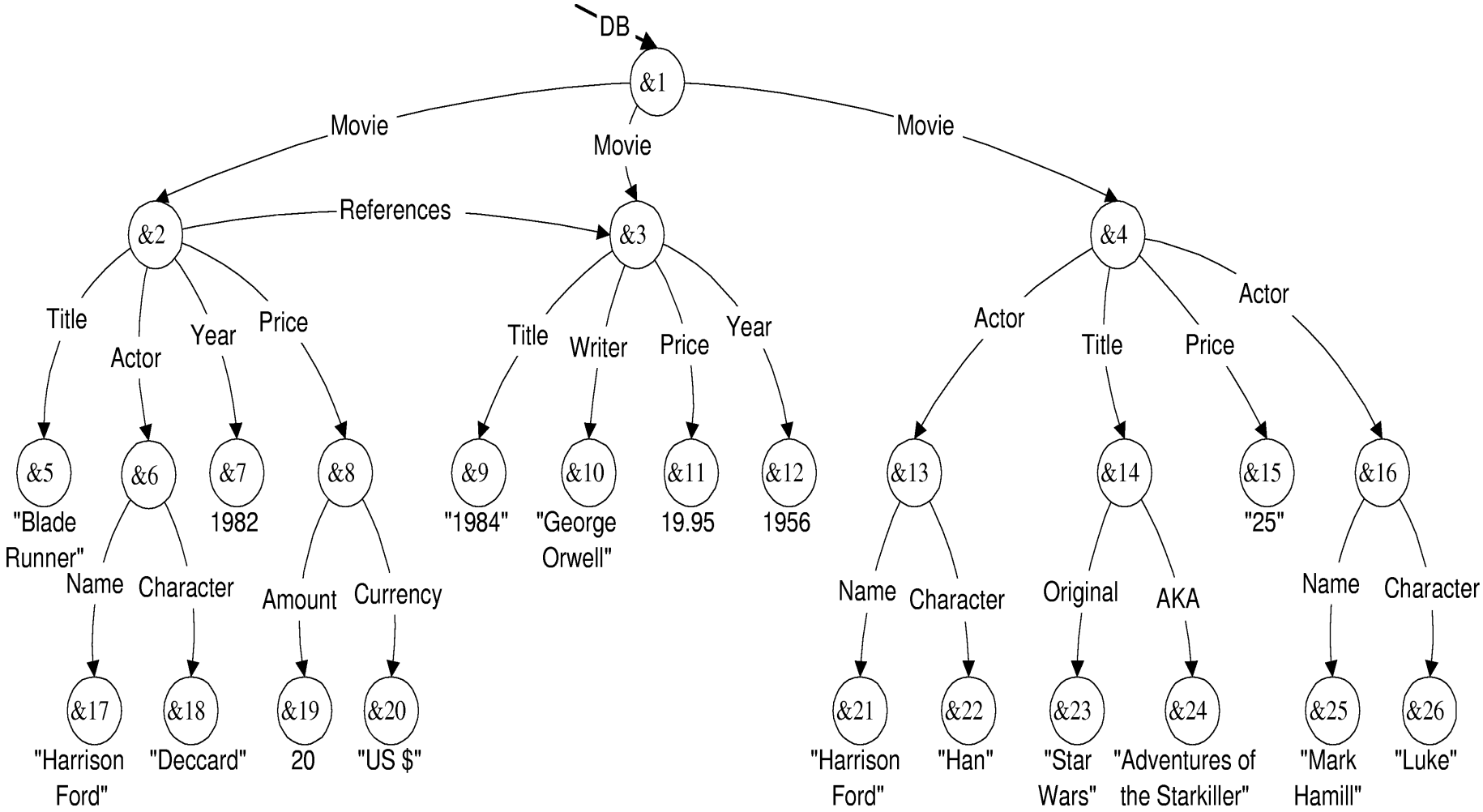
- Datenstruktur, die identifiziert, welcher Eintrag welchen Wert enthält
- Ziel: Beschleunigung der Anfrageverarbeitung
- Problem: Speicherbedarf

→Kompromiss erforderlich

Einführung – Wo wird der Index eingesetzt?



Einführung – OEM-Graph eines XML-Fragments



Einführung – Bewertungskriterien für Indizes

- Anwendbarkeit
- Erweiterbarkeit
- Skalierbarkeit
- Auswertungszeit
- Speicherplatz
- Updates

Indexstrukturen in XML

- Inhaltsbasierte Indexstrukturen
- Strukturbasierte Indexstrukturen
- Hybride Indexstrukturen
- SphinX

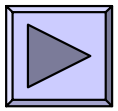
Inhaltsbasierte und Strukturbasierte Indizes nur in Kombination

Inhaltsbasierte Indexstrukturen

- Aus dem klassischen Information Retrieval
- Inverted Files
  - Invertierte Liste
  - Vokabular

Inhaltsbasierte Indexstrukturen – Text Index

- Bildet ein Schlüsselwort auf alle Knoten ab, die es enthalten
- Direkter oder indirekter Inhalt
- Reduzierung der Schlüsselwörter:
  - Stemming
  - Stoppwortlisten
  - Auswahl der Schlüsselwörter



Inhaltsbasierte Indexstrukturen – Value Index

- Generalisierung des Text Index
- Verschiedene Datentypen
- Selektionskriterien: >, <, =, contains

Inhaltsbasierte Indexstrukturen - Eigenschaften

Index	Text Index	Value Index
Input → Output	Keywords → NodeIDs	Predicates → NodeIDs
Indexstruktur	Inverted File	Inverted File
DB-Struktur	Graph	Graph
Inkrementelle Updates	Ja	Ja
Speicherbedarf	5-15% Storage Overhead (Angabe für Inverted Files)	5-15% Storage Overhead (Angabe für Inverted Files)
Einschränkungen	Keine Struktur	Keine Struktur
Besonderheiten		Kann Typecasting unterstützen

Inhaltsbasierte Indexstrukturen - Beispiel

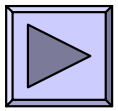
Lore (Lightweight Object REpository) Stanford University

- Tindex: Suche alle Namen, die Ford enthalten

  Ergebnis: {(&17, 2), (&21, 2)}

- Vindex: Suche Preise mit Preis > 15

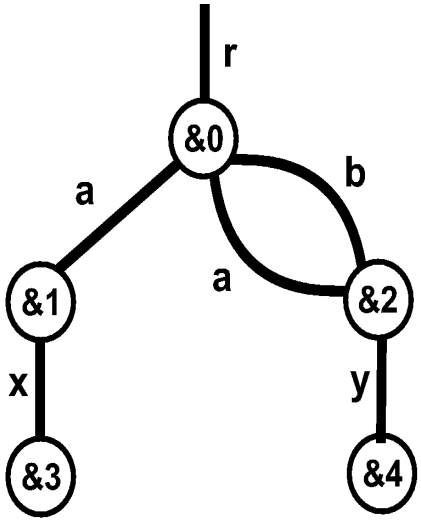
  Ergebnis: {&11, &15}



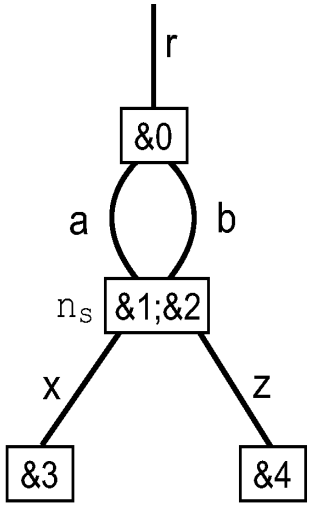
Strukturbasierte Indexstrukturen - DataGuide

- Graph, in dem jeder Pfad aus der Datenbank genau einmal vorkommt
- Für Indexstrukturen wird zusätzlich benötigt, dass kein Pfad auftritt, der nicht in der DB vorkommt (strong DataGuide).

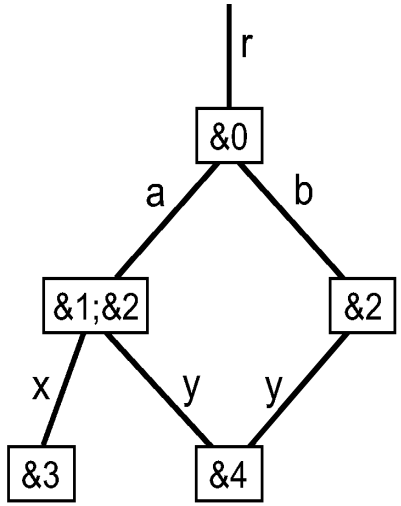
Strukturbasierte Indexstrukturen – DataGuide



a) Datenbank-Graph



b)weak DataGuide



c) strong DataGuide

Strukturbasierte Indexstrukturen - Eigenschaften

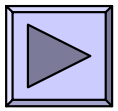
Index	DataGuide	T-Index
Input → Output	Simple Paths→ NodeIDs	1-Index: Simple Paths→ NodeIDs 2-Index: Relative Paths→ NodeIDs T-Index: Privileged Paths→ NodeIDs
Indexstruktur	Graph	Graph
DB-Struktur	Graph	Graph
Inkrementelle Updates	Ja	Ja
Speicherbedarf	Worst Case: exponentiell Im Experiment je nach DB-Struktur 2-317% Storage Overhead	Worst Case: 1-Index linear 2-Index quadratisch
Einschränkungen	Keine relativen Pfade, kein Inhalt	
Besonderheiten		Path Templates

Strukturbasierte Indexstrukturen - Beispiel

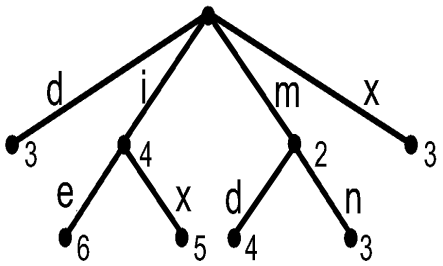
- Lore

  Pindex: Suche alle Titel

  Ergebnis: {&5, &9, &14}



Hybride Indexstrukturen – IndexFabric



PATRICIA Trie
für die Schlüsselwörter
die, indeed, index,
midi, min, xml

- Auf der Basis von PATRICIA Tries
- Raw Paths und Refined Paths
- Bezeichner zur Kodierung der Pfade

Hybride Indexstrukturen - IndexFabric

- Dokument 1:

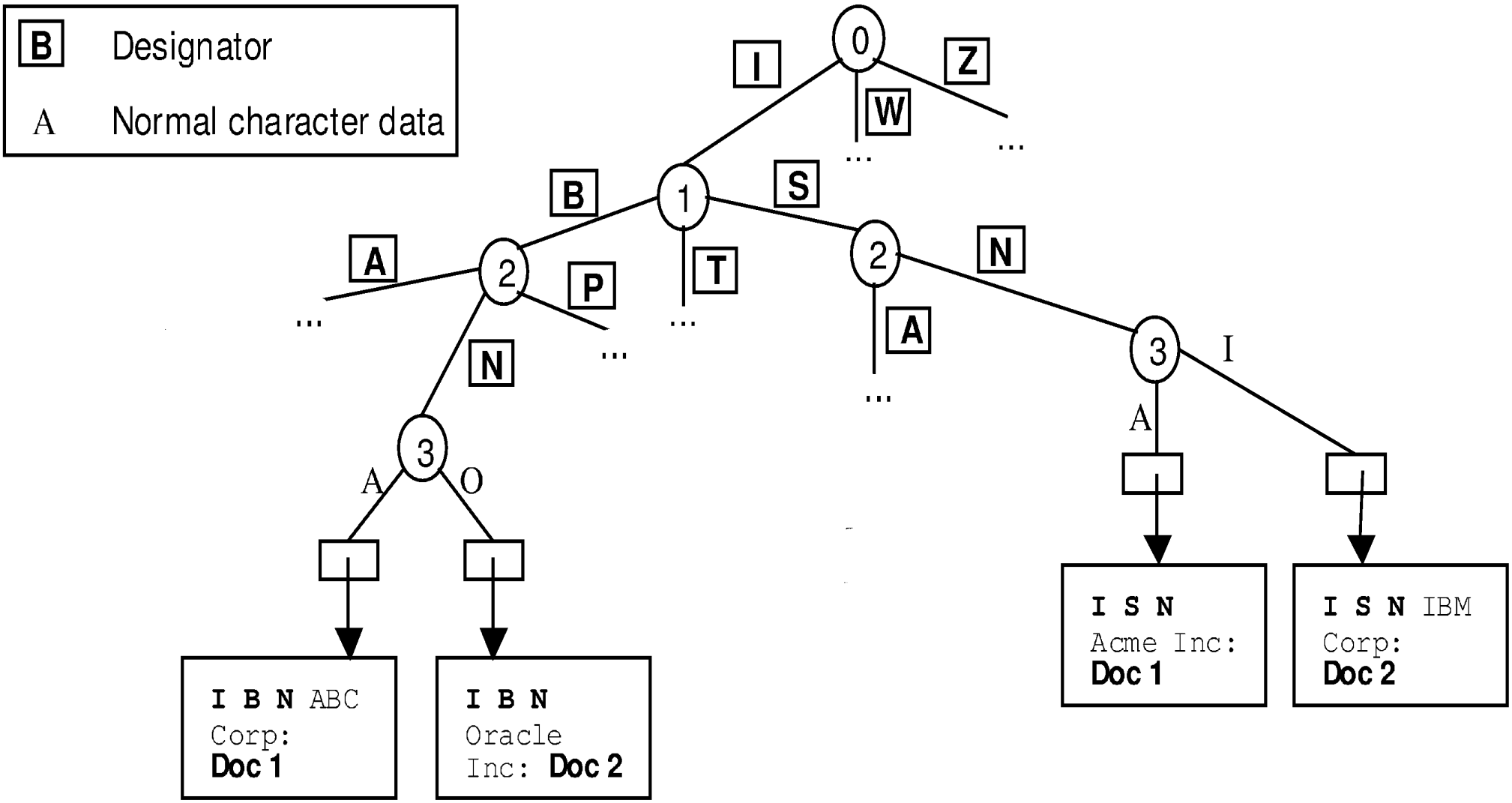
```
<invoice>
        <buyer><name>ABC Corp</name>
        </buyer>
        <seller><name>Acme Inc</name>
        </seller>
          ...
</invoice>
```

- Dokument 2:

```
<invoice>
        <buyer><name>Oracle Inc</name>
        </buyer>
        <seller><name>IBM Corp</name>
        </seller>
          ...
</invoice>
```

- Bezeichner:
  <invoice> = **I**
  <buyer> = **B**
  <name> = **N**
  <seller> = **S**
  ...
- Pfade:
  - Dokument 1:
    IBN ABC Corp
    ISN Acme Inc
    ...
  - Dokument 2:
    IBN Oracle Inc
    ISN IBM Corp
    ...

Hybride Indexstrukturen - IndexFabric



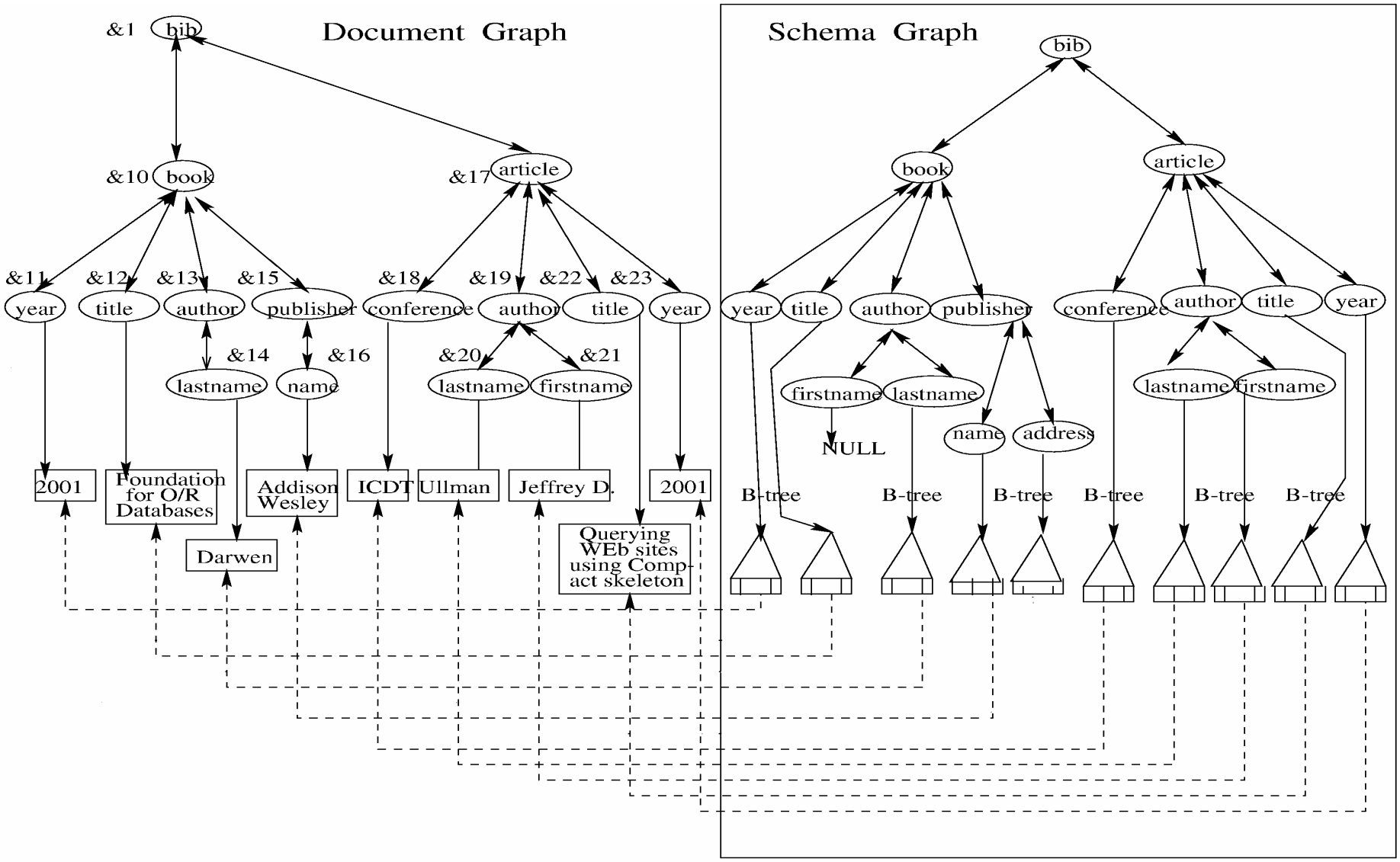
Hybride Indexstrukturen - Eigenschaften

Index	IndexFabric	BUS
Input → Output	Simple Paths → NodeIDs Privileged Paths → NodeIDs	Simple Paths x Keywords → DocIDs x NodeIDs x Weights
Indexstruktur	Baum	Baum
DB-Struktur	Baum	Baum
Inkrementelle Updates	Ja	Nur teilweise (MBM: Ja)
Speicherbedarf	In Experimenten bis 150% Storage Overhead	In Experimenten bis 240% Storage Overhead
Besonderheiten	Refined Paths	Ranking der Treffer

SphinX: Schema-conscious PatH INdexing of XML

- Verwendet DTD
- Besteht aus
  - Dokument-Graph
  - Schema-Graph
  - B+-Bäume

SphinX - Abbildung



SphinX - Anfrageverarbeitung

- Ermittlung der Pfade durch den Schema-Graph (Index-Path-Extraction-Algorithmus)
- Durchsuchen der B+-Bäume

  (Node-Identification-Algorithmus)
- Traversierung des Dokument-Graph

SphinX - Eigenschaften

Index	SphinX
Input → Output	Regular Path Expression → NodeIDs
Indexstruktur	Graph
DB-Struktur	Graph
Inkrementelle Updates	Ja
Speicherbedarf	160-300% Storage Overhead
Besonderheiten	DTD wird benutzt

Ein Vergleich – SphinX vs. Lore

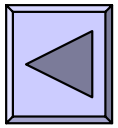
- einziger existenter direkter Vergleich

Eigenschaft	**Lore**	**SphinX**
DTD verwendet	Nein	Ja
Einzelner Index	Nein (Tindex, Lindex, Vindex und Pindex)	Ja
Spezialisierte Indexstrukturen	Ja (DataGuide)	Nein (B^+-Baum)
Getestet mit großen Dokumenten	Nein	Ja

SphinX vs. Lore – Anfrageverarbeitung

- Beispiel
  - A1: Suche /bib[//year=2001]//conference
  - A2: Suche /bib/article[year=2001]/conference

- Ergebnis:
  - SphinX: Beide Anfragen sehr effektiv, keine unproduktiven Pfade
  - Lore: Bei beiden Anfragen unproduktive Pfade



SphinX vs. Lore - Experimenteller Aufbau

- Pentium III 800MHz, 512 MB Speicher, 18 GB SCSI Plattenspeicher, Redhat Linux 7.1
- Daten und Index auf Platte gespeichert

Document	Size	Records	Depth	Elems	Attrs	Total E.	Total A.
Shakespeare	7.4MB	35	High	22	0	179619	0
Conference	40MB	104K	Low	25	2	1029494	159928
Journal	27MB	76K	Low	15	2	804176	89567
Swiss-Prot	158MB	80K	High	-	-	4243031	2898833
Ham-Radio	361MB	0.7 M	Low	24	0	14117198	0
Xmark	1126MB	1	High	77	16	16703249	3829772

SphinX vs. Lore – Ergebnisse 1/2

Index-Konstruktion

Dokument	SphinX	Lore
Shakespeare	15M	35M
Conference	85M	226M
Journal	63M	230M
Swiss-Prot	465M	877M
Ham-Radio	834M	Incomplete
Xmark	1900M	Incomplete

Index-Größe

Dokument	SphinX	Lore
Shakespeare	14s	41s
Conference	94s	1433s
Journal	73s	969s
Swiss-Prot	614s	20460s
Ham-Radio	1220s	Incomplete
Xmark	1930s	Incomplete

Erstellungszeit

SphinX vs. Lore – Ergebnisse 2/2

Zeitbedarf bei der Anfrageverarbeitung

Dokument	Query	SphinX	Lore
Shakespeare	QS1	0.7s	3.3 s
Conference	QS2	0.5 s	Incomplete
Conference	QS3	6.8 s	Incomplete
Journal	QS4	0.8 s	59.9 s
Journal	QS5	0.7 s	60.7 s
Ham-Radio	QS6	122.8 s	Incomplete
Xmark	QS7	234.2 s	Incomplete

a) absolute Pfadausdrücke

Dokument	Query	SphinX	Lore
Shakespeare	QG1	0.8 s	5.4 s
Conference	QG2	1.1 s	Incomplete
Conference	QG3	7.8 s	Incomplete
Journal	QG4	0.8 s	61.2 s
Journal	QG5	0.7 s	62.6 s
Ham-Radio	QG6	140.8 s	Incomplete
Xmark	QG7	237.2 s	Incomplete

b) allgemeine Pfadausdrücke

Zusammenfassung und Ausblick

- Zusammenfassung
  - XML stellt besondere Ansprüche an Indexstrukturen
  - Verschiedene Ansätze
  - Zum Teil deutliche Leistungsunterschiede
- Ausblick
  - Zur Zeit sehr intensiv erforschtes Gebiet
  - Vergleiche zwischen Indexstrukturen erforderlich