*Lucene Workshop Stuttgart 2006*

# Einsatz von Lucene im BAM-Portal

*Dipl. Phys. Thomas Kirchhoff*
*Dipl. Inf. Christof Mainberger*
*Stuttgart 24.01.2006*

# Motivation – Wozu Lucene ?

- Herkömmliche Suchtechnik mittels RDBMS.
- Beispiel SWBPlus :
  - Zusätzlich zur normalen Suche in Metadaten werden Inhalte, wie zum Beispiel Klappentexte, Inhaltsverzeichnisse, oder Rezensionen zur Suche aufbereitet

  http://www.bsz-bw.de/recherche/swbplus/

  vs.

  http://linzgau.bsz-bw.de/bam-test/
- Enormer Geschwindigkeitsunterschied
- SQL Version nicht praktikabel

# Was genau ist Lucene ?

- Lucene ist eine Javabibliothek zur Konstruktion von Volltextsuchmaschinen (http://lucene.apache.org)
- Grundlegende Idee von Lucene:
  - Datenstruktur: Invertiertes File als Volltextindex
  - Vokabular des Textes wird extrahiert
  - Postingliste: zu jedem Wort des Vokabulars wird vermerkt, in welchem Dokument es an welcher Stelle vorkommt:

# Aufbau eines Lucene-Index

- Ein Index besteht aus Dokumenten
- Ein Dokument besteht aus Feldern
- Ein Feld hat einen Namen und einen Wert
- Der Wert des Feldes besteht aus Termen
- Ein Term ist im Prinzip ein Wort
- Bevor das Wort zum Term wird, wird es vorbehandelt
  - Punktuation entfernen
  - Wortstammbildung
  - Stoppwörter (der,die,das,ein,da,...) werden verworfen.
  - Umlaute und Akzente entfernen

# Suchmöglichkeiten mit Lucene

- Jedes Feld kann separat abgefragt werden
  - title:eisenbahn
- Abfragen können mit Booleschen Operatoren kombiniert werden
  - title:eisenbahn AND author:Maier
- Auch nach Phrasen kann gesucht werden
  - title:"Die Schwäbische Eisenbahn"
- Die Signifikanz eines gefundenen Wertes kann durch Angabe eine Boost-Faktors erhöht werden. Dies verändert die Rangreihenfolge der Trefferliste
  - title:eisenbahn^4 OR title:auto

# Suchraumerweiterung durch Synonyme

- Idee: Bei der Suche werden noch zusätzlich die Synonyme des gesuchten Terms benützt.
- Beispiel:
  - title:auto
  - title:(kraftwagen OR kraftfahrzeug OR auto)
- In BAM kommt die SWD zum Einsatz. Im Prinzip kann aber jeder Index in BAM seinen eigenen Thesaurus bekommen.

# Vorteile von Lucene

- Schnelligkeit: Im Vergleich zu SQL-Datenbanken ist die Volltextsuche mit Lucene extrem schnell
- Stabilität: Lucene ist sehr ausgereift und wird in vielen Projekten verwendet
- Skalierbarkeit: Indices lassen sich aufteilen und dann parallel abfragen. Dies lässt im Prinzip eine ähnliche Architektur zu wie Google: Für die Indexabfrage existiert hier eine Farm von ca **8000** Rechnern. Experimente mit **150 Mio** Dokumenten wurden erfolgreich durchgeführt.
- Eigene Experimente mit ca **33 Mio** Dokumenten

# BAM Suchmaschinen Framework

- Anwendungen sind von verwendeter Text-Retrieval-Engine unabhängig.
- Experimente mit anderer Technologie leicht möglich.
- Beispiele: Terrier, YACY, Approximus
- Indices und Pipelines per XML konfigurierbar
- Läuft in Tomcat oder JBoss
- Suche und Indexierung gleichzeitig möglich (Bei Massenindizierung aber nicht sinnvoll)
- Automatische Generierung von Suchformularen aus Indexkonfiguration => Rapid-Prototyping von HTML-Oberflächen.

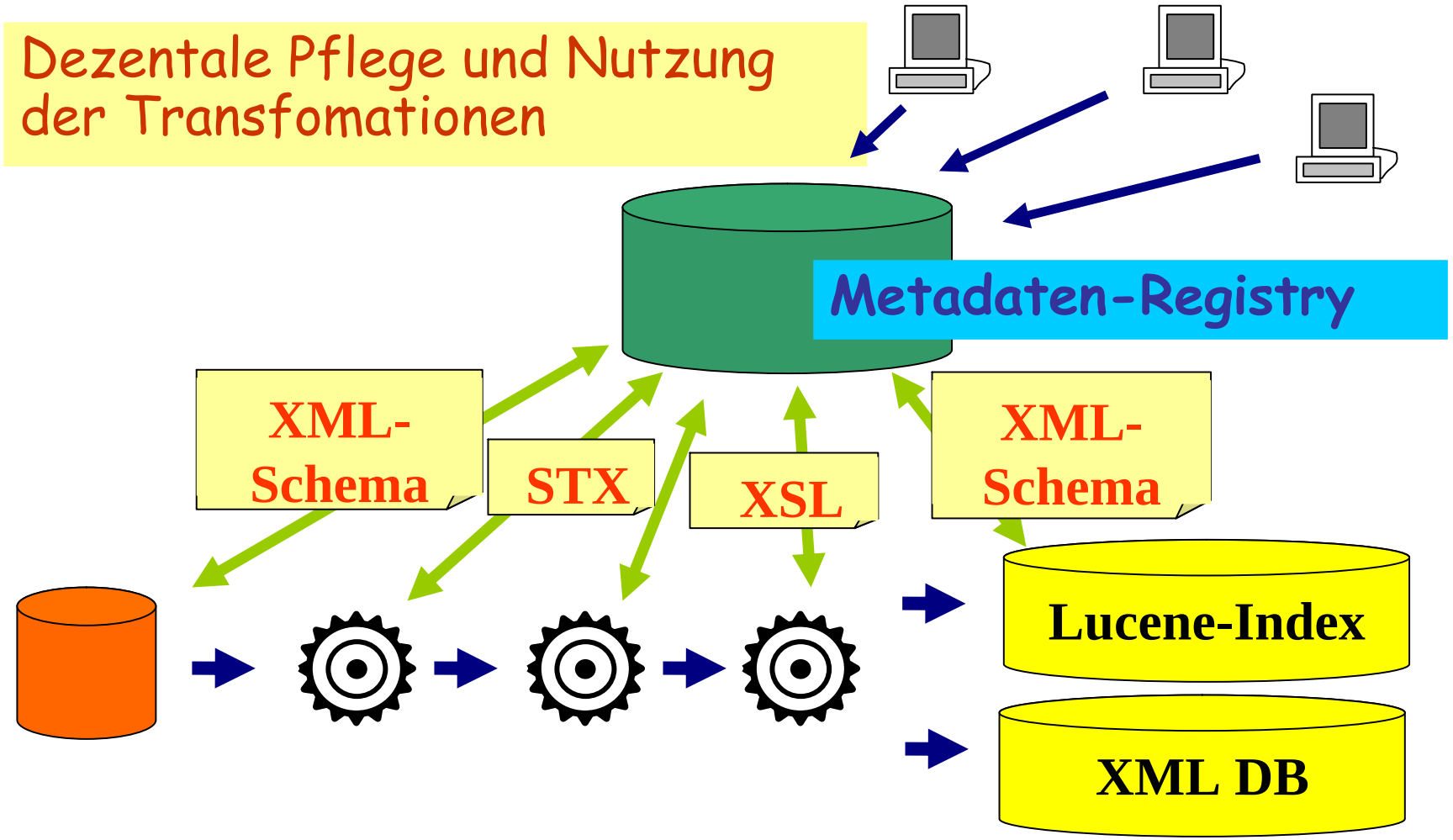
# Indexaufbau bei BAM

- Konfiguration per XML (Spring-Framework)
- Drei Indextypen:
  - Bibliothek (Datenquelle Pica/MAB2)
  - Archiv (Datenquelle SQL, geplant EAD)
  - Museum (Datenquelle SQL, geplant CRM)
- Gemeinsame Felder aller drei Indextypen:
  - Titel
  - Person/Körperschaft
  - Ort
  - Zeit (Zeitpunkt und Zeitintervall in Jahren gemessen)
  - Schlagwort

# Indexaufbau bei BAM

- Ferner werden gespeichert:
  - Alle Informationen, um auf die Website des Informationsanbieters zu verzweigen und dort das Dokument anzuzeigen (im Prinzip eine URL)
  - Zusätzliche Felder, die spezifisch für einen Indextyp ist, z.B. *Material* bei Museumsdaten
  - Die zusätzlichen Felder ermöglichen spezialisierte Suchen.
  - Verwaltungsinformation, wie *lastModified*, wichtig beim Indexupdate.

# Indexerzeugung in BAM



# Indexerzeugung in BAM

- Pipelinekonzept: es fliessen SAX Events
- Erfinder: Henry Ford, Cocoon, VDS, …
- Pipelines haben Anfang-, Mittelteile und ein Ende:
  - Generator: Beliebige Daten werden in XML Format gebracht.
  - Transformator: XML wird mittels XSL bzw. STX Tranformationen in Standardformate umgewandelt
  - Serialisierer: XML wird in einen neuen Aggregatzustand überführt (Luceneindex, XML-DB)
- Performance: ca **6000** Dokumente pro Minute auf 3 GHz PC. Sehr hohe Systembelastung.

# In BAM unterstützte Metadatenformate

- CSV/Excel
- Text, HTML, PDF
- MAB2, MAB-Diskette
- Pica, Pica+
- EAD
- Diverse nicht standardisierte XML Formate
- Neues Input-Format erfordert:
  - das Schreiben eines *Generators* (ca 2-3h)
  - ein oder mehrere *Stylesheets* (2-3h)
- Neues Outputformat erfordert:
  - Schreiben eines *Serialisieres*, meistens aber nicht nötig.

# Verteilte Suche

- Lose Kopplung:
  - Mehrere BAM-Suchmaschinen können mittels Opensearch-Schnittstelle verbunden werden.
  - Protokoll ist XML (Opensearchinitiative http://A9.com)
  - http://linzgau.bsz-bw.de/bam-test/opensearch.do?query=hund&category=library
- Enge Kopplung:
  - Per Java-RMI Schnittstelle
  - Für Cluster geignet

# Browsing Zugang zur Suchmaschine

- Recherche per hierarchischer Navigation.
  - Vorbild Google Directory http://directory.google.com
  - Zunächst geordnet nach Kategorie,Anbieter,Sammlung
  - Thematische Klassifikation (semiautomatisch ?)
  - Dies kann auch zur Präsentation genützt werden (Zum Beispiel für Museen, die sich keine eigene Homepage leisten wollen)
  - Nebeneffekt: Eine solche Hierarchie kann von Google indexiert werden.

# Administration der BSZ-Suchmaschine

- Verwaltung der BSZ-Suchmaschine per WebGUI
- Anlegen neuer Indices, Update bestehender
- Passiver Upload von Daten
- Aktives Harvesting mittels eines beim Informationsanbieter installierten Webservice (Wichtig vor allem bei großen Anbietern)

# BAM Portal: Personalisierung

- AAR via Shibboleth (Authentifizierung, Autorisierung, Rechteverwaltung)
- Quellenauswahl
- Suchhistorie
- Merkliste
- Anmerkungen zu einzelnen Objekten (siehe Amazon Reviews)
- Querverweise
- Virtuelle Sammlungen (siehe virtuelles Bücherregal Amazon)

# BAM Portal: Nutzerstatistik

- Herkunft des Nutzers
- Dauer der Session
- Suchanfragen
- Weiternavigation in eingebundene Fachinformationssysteme

# Fazit

- Es sind keine 2 Milliarden Euro nötig, um eine performante Suchmaschine zu bauen.
- Lucene heute viel **besser**, als Google 1998