Search Engine Technologies

Status and Directon

Ingvar.Aaberg@fast.no

February 1, 2006

Outline

- Background and introduction
- Anatomy of a search engine
- Some thoughts on the web search market
- Enabling technologies

FAST Background

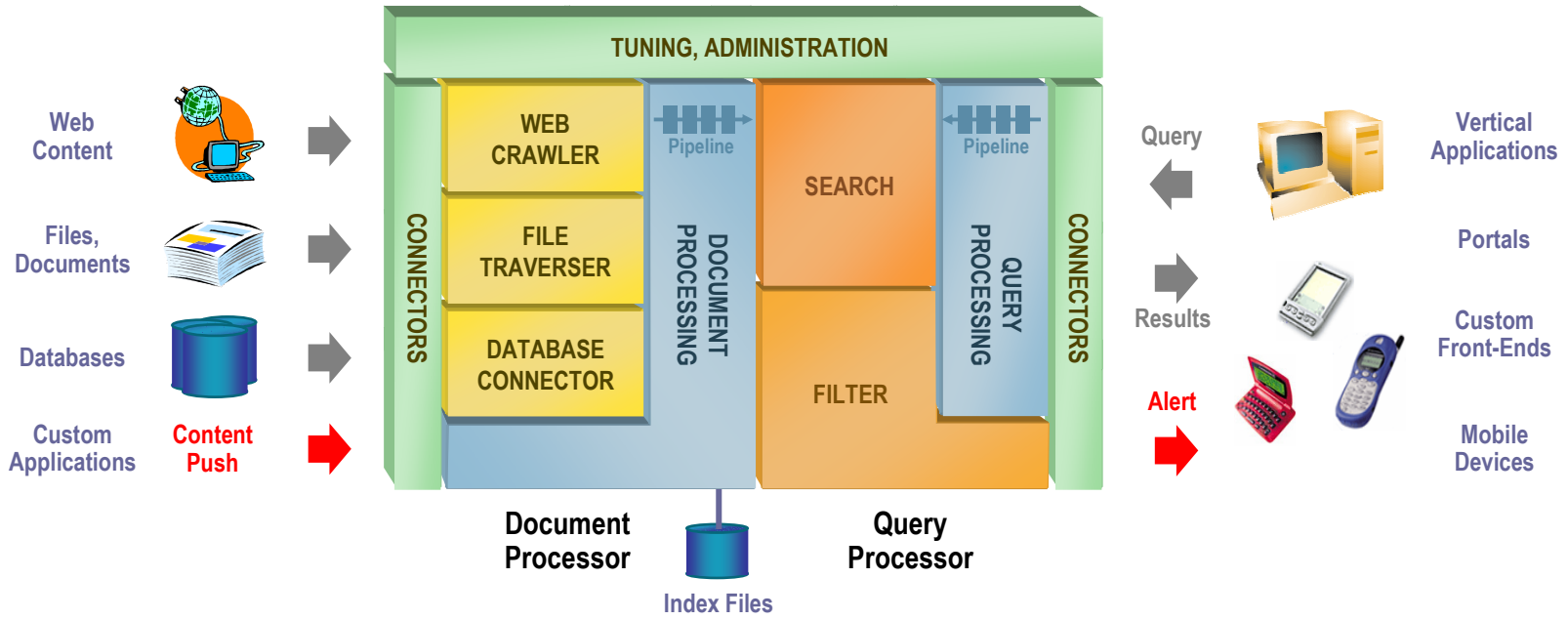
Corporate Overview

- **Leadership**
  - Founded in 1997
  - Public company (OSE: 'FAST')
  - Profitable and well capitalized
  - Revenue growth = 50%
  - > 3,000 customers
  - 40+ PhDs

- **Focus on Enterprise Search**
  - Sold internet business (alltheweb.com)
  - Acquiring enterprise search complements
  - Ranked market leader by Gartner

Search Engine
How It Works

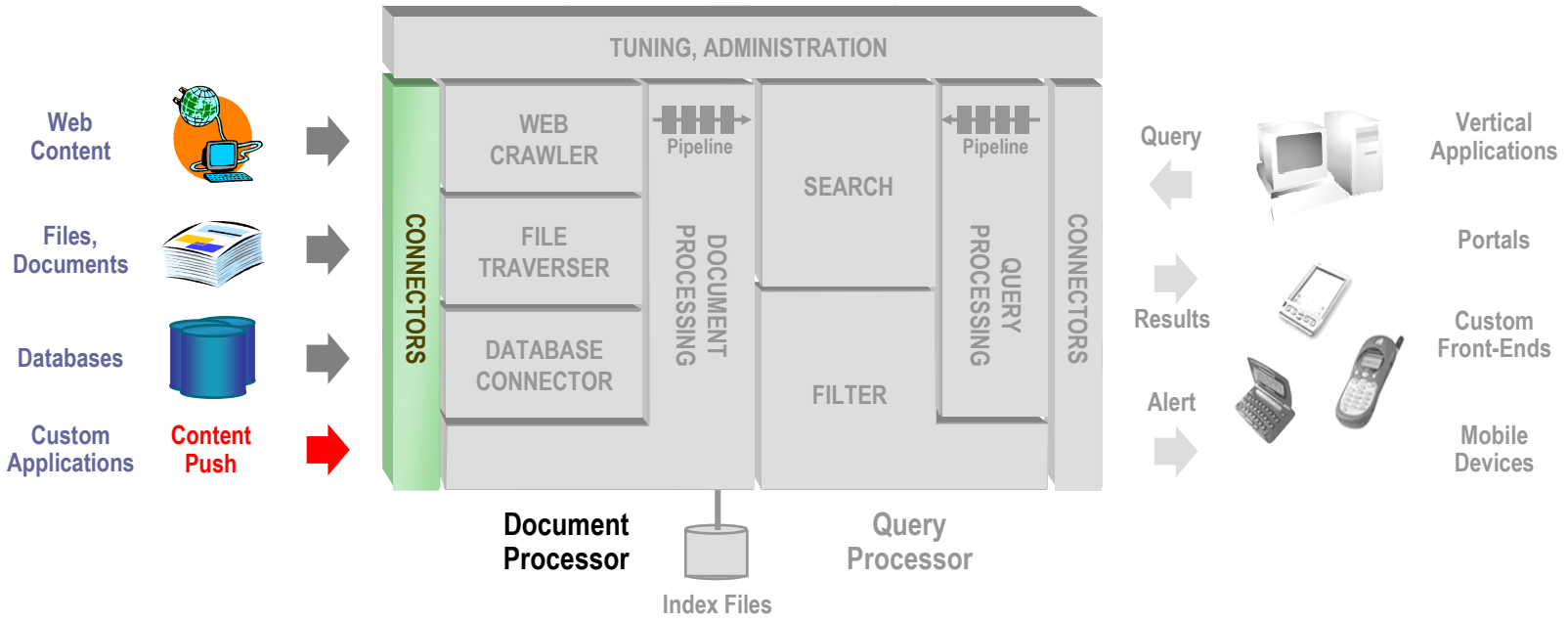
- **Two asynchronous independent components**
  - Document processor
  - Query processor



Search Engine
How It Works

1 Connect to content sources

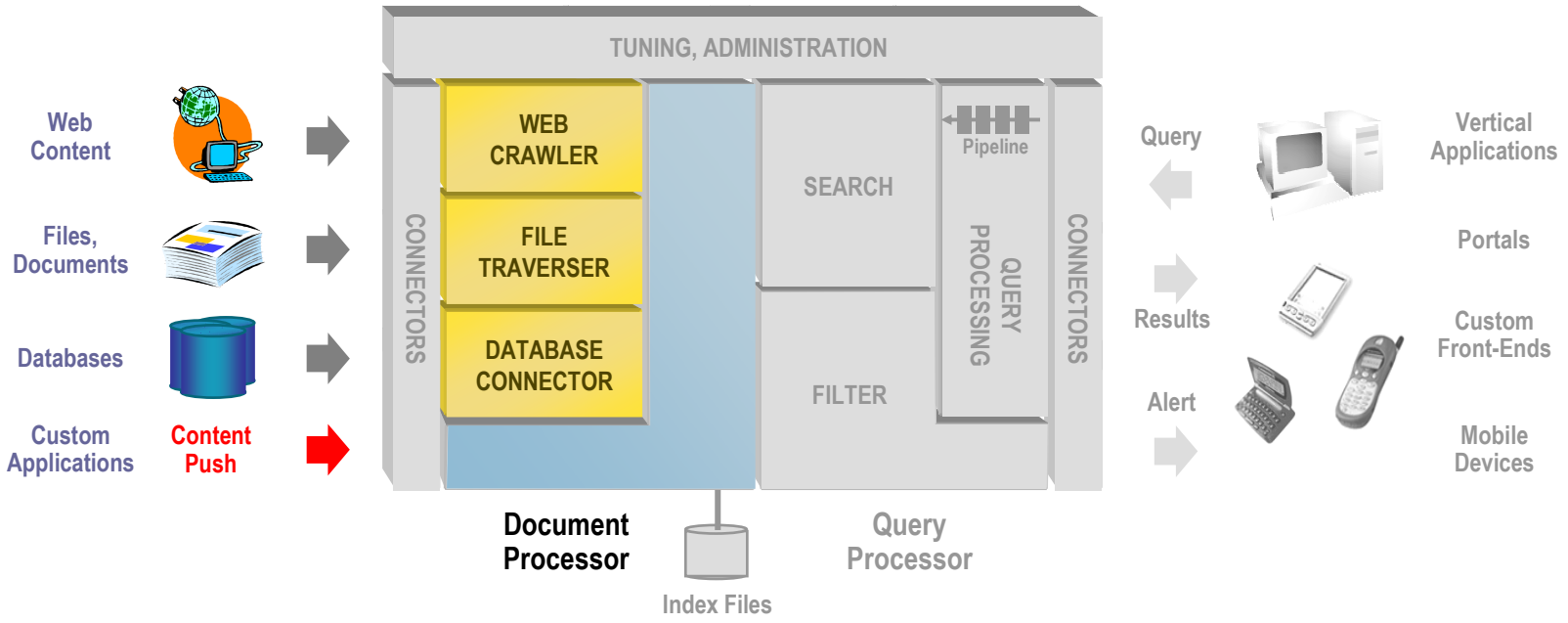
- Structured (RDBMS)
- Unstructured (Web, Office, etc.)
- Semi-structured (XML)



Search Engine
How It Works

2 Get content data

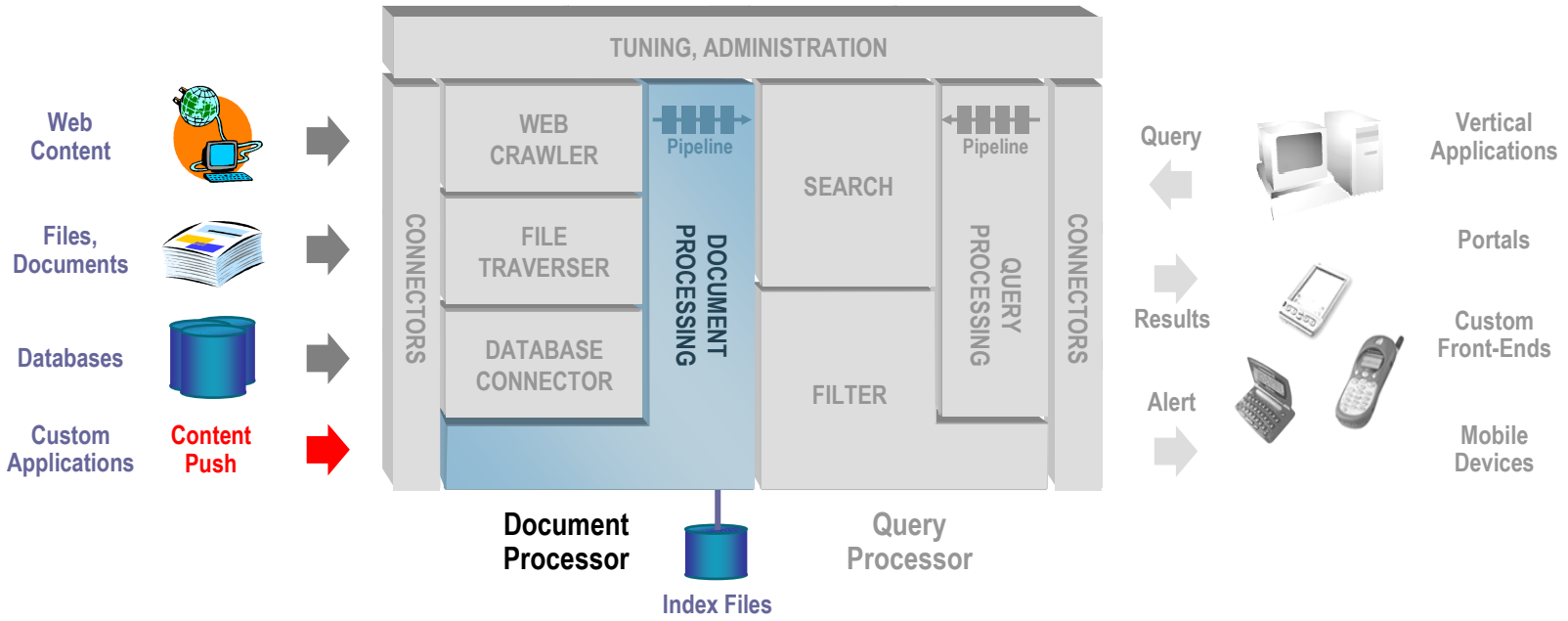
- Gather data wholesale by incrementally crawling links
- Push changed data content directly to processing stage



Search Engine
How It Works

3 Index content to make it searchable

- Convert and process content through pre-processing pipeline:
  - Linguistic processing, taxonomy classifying
  - Custom logic (e.g. adding special tags)
- Write content to index files



Document processing

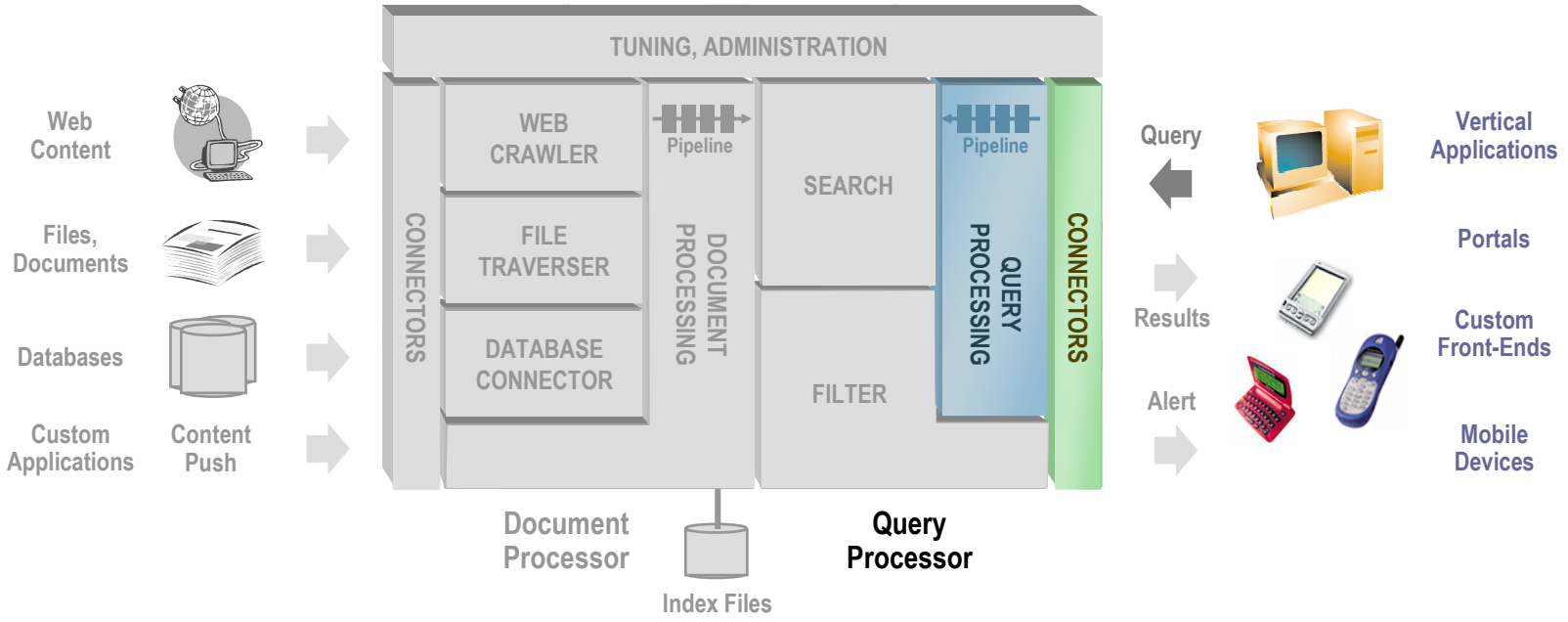
- **Processes the content before it gets indexed**
  - Documents flow through a pipeline of processing stages
  - Highly customizable

- **Example processing stages**
  - Format, language and encoding detection — pdf; english; iso-8859-1
  - Format and encoding normalization — pdf → html; iso-8859-1 → utf-8
  - HTML parsing
  - Entity extraction — venus williams; arthur ashe; ...
  - Vectorization — {(venus williams, 1.0), (wimbledon, 0.81), (center court, 0.65), ...}
  - Categorization — sports/tennis
  - Lemmatization and synonym handling — mouse ~ mice; car ~ automobile
  - Anchor text harvesting
  - ...

Search Engine
How It Works

Interpret query

- Use query language or query API
- Convert and process query through query pipeline:
  - Linguistic processing
  - Custom logic (e.g. query term modification/addition)



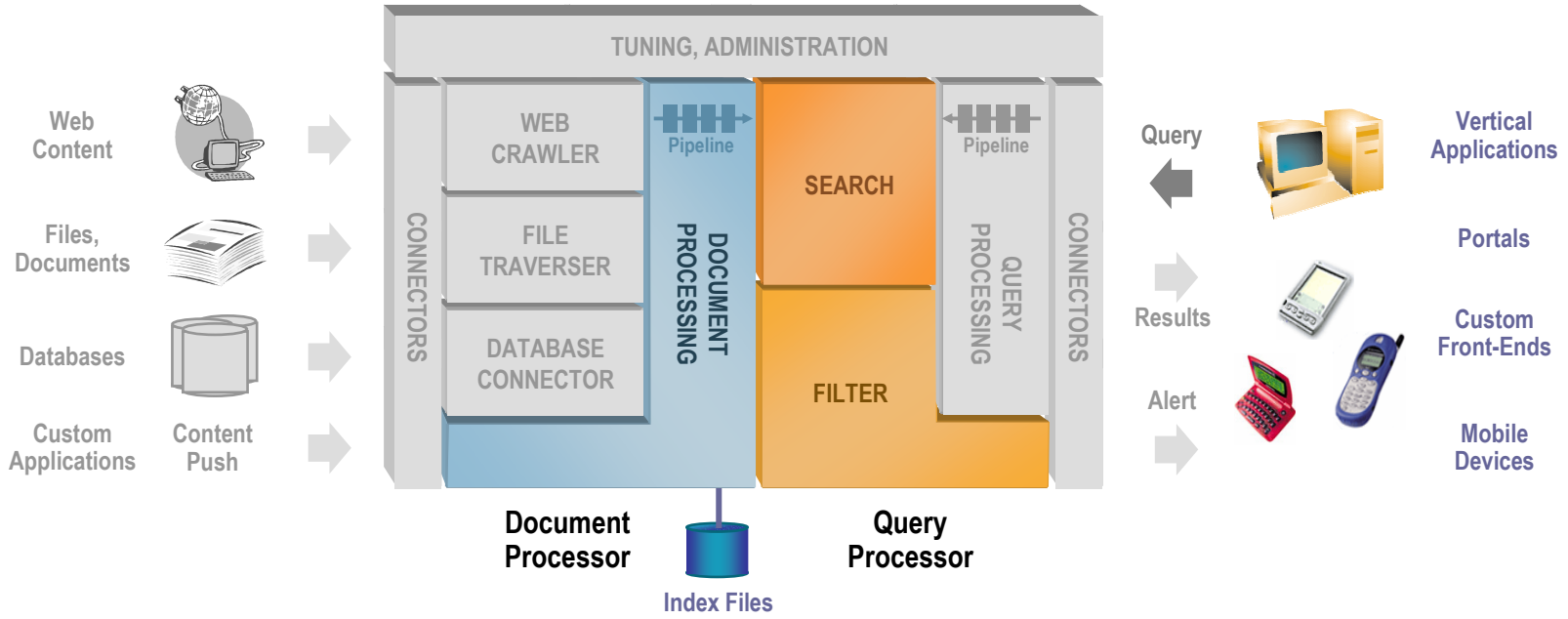
Content Enrichment

- **Processes the query before it gets sent to the search engine**
  - Queries flow through a pipeline of processing stages
  - Highly customizable

- **Example processing stages**
  - Stopword handling
  - Phrasing and antiphrasing — red cross → “red cross”; where can i find information about cars → cars
  - Spellchecking — brittany speers → britney spears
  - Natural language handling — television under 200 dollars → television AND price:<200
  - Lemmatization and synonym handling — mouse ~ mice; car ~ automobile
  - Ontologies — xsara < citröen < car < vehicle
  - ...

5 Get results

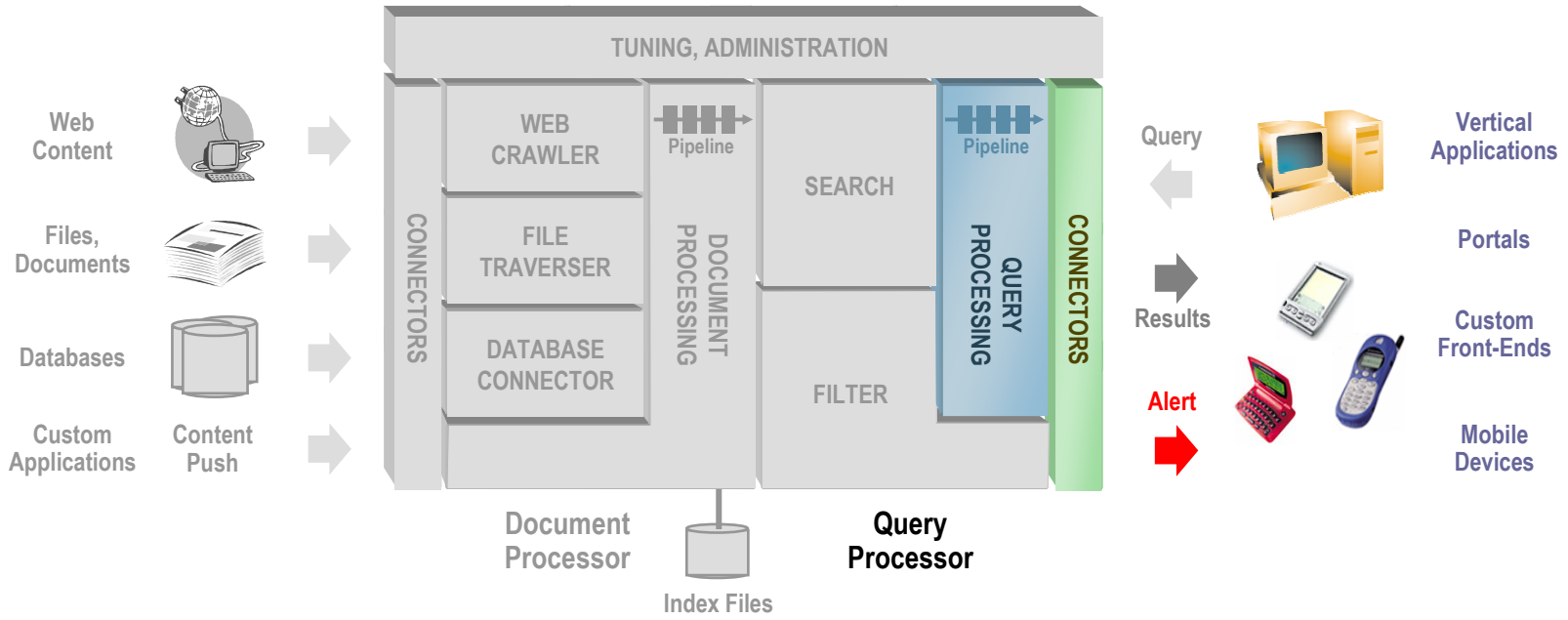
- Get crawled results from index files
- Get live results directly from processor



Search Engine
How It Works

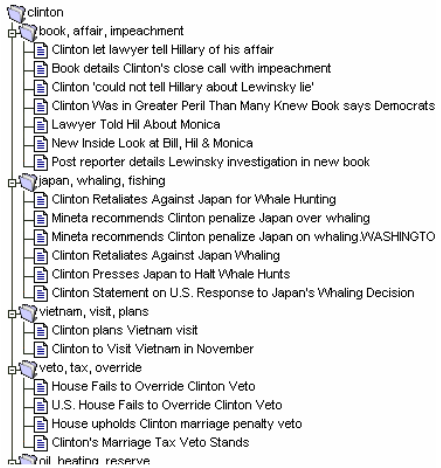
6 Return results to user

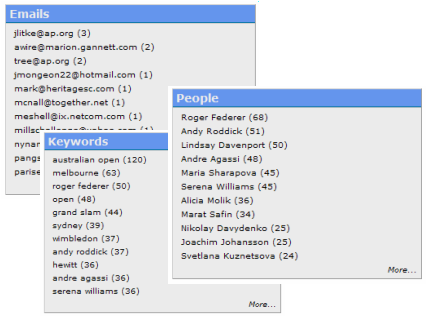
- Convert and process results through query pipeline:
  - Resort, filter for security, organize for dynamic drilldown
- Pass results on to application (generated or through API)
- Push results to alert engine and then external environment (e.g. mail, queue)



Result processing

- **Unsupervised clustering**
  - Analyzes on-the-fly similarities across matching documents
  - Group together documents having similar content
  - Provides a bird's-eye view of topical spread
- **Query refinement suggestions**
  - Examines the distribution of meta data
  - Builds a histogram of values
  - Provides a means for slicing and dicing the result set
- **Filtering**
  - E.g., dynamic duplicate detection



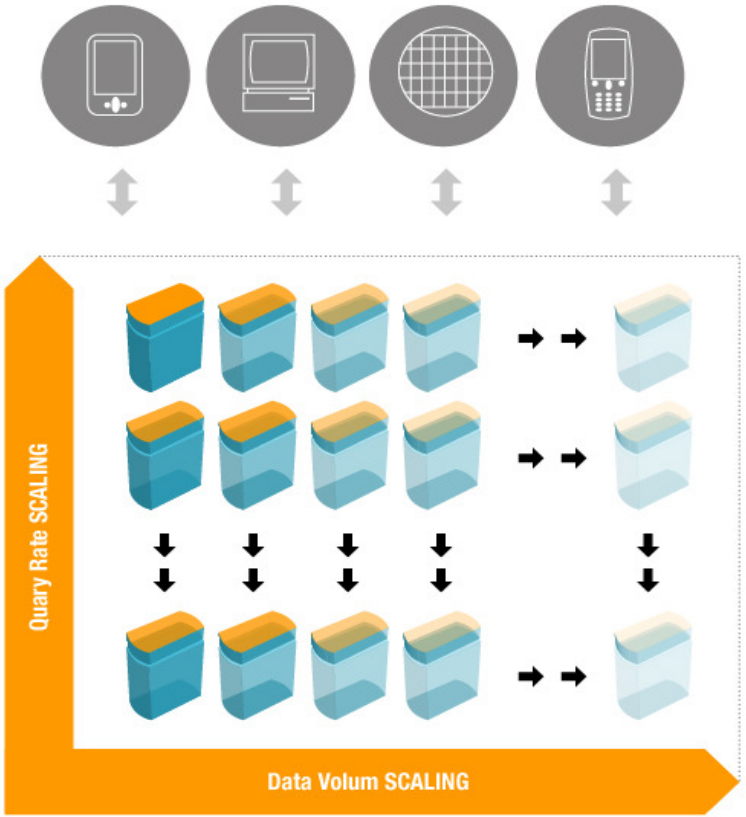


Relevancy

Completeness	How well does the query match superior contexts?
Authority	Is the document considered an authority for this query?
Statistics	How well does the content of this document match the query?
Quality	Is this a document of ”high quality”?
Freshness	How old is this document, when was it last updated?
Geography	Where are you querying from?

Engine architecture

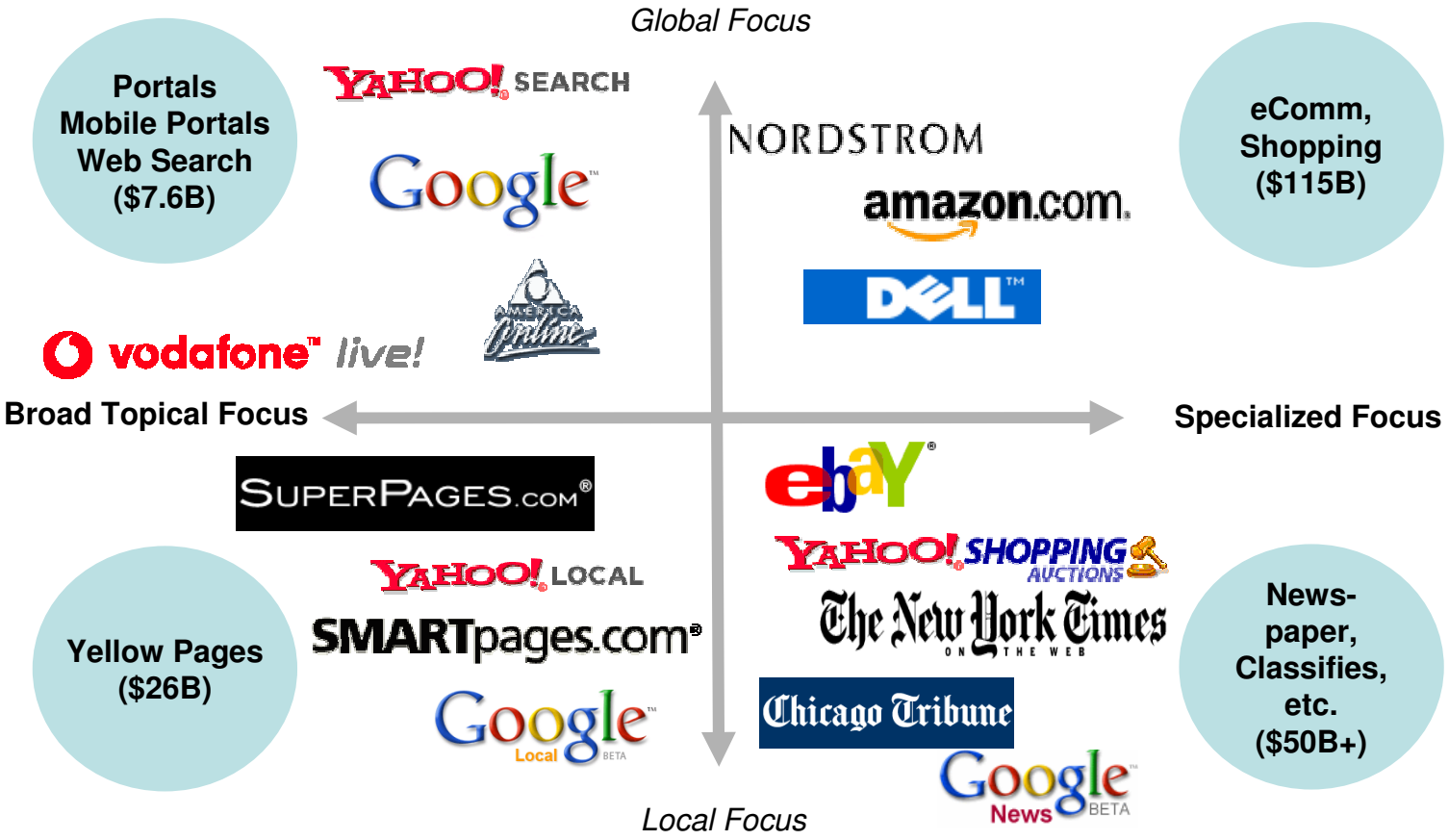
- **Scaling in data volume**
  - Add columns
  - Each column holds a partition of the data
  - Query the partitions in parallel
- **Scaling in query traffic**
  - Add rows
  - Replicate the partitions
  - Distribute the queries
- **Scaling in other dimensions**
  - Query complexity
  - Fault tolerance



Traditional Media Challenged

Entrance of Non-Traditional Competitors

Competitive Pressures Redefining the Business

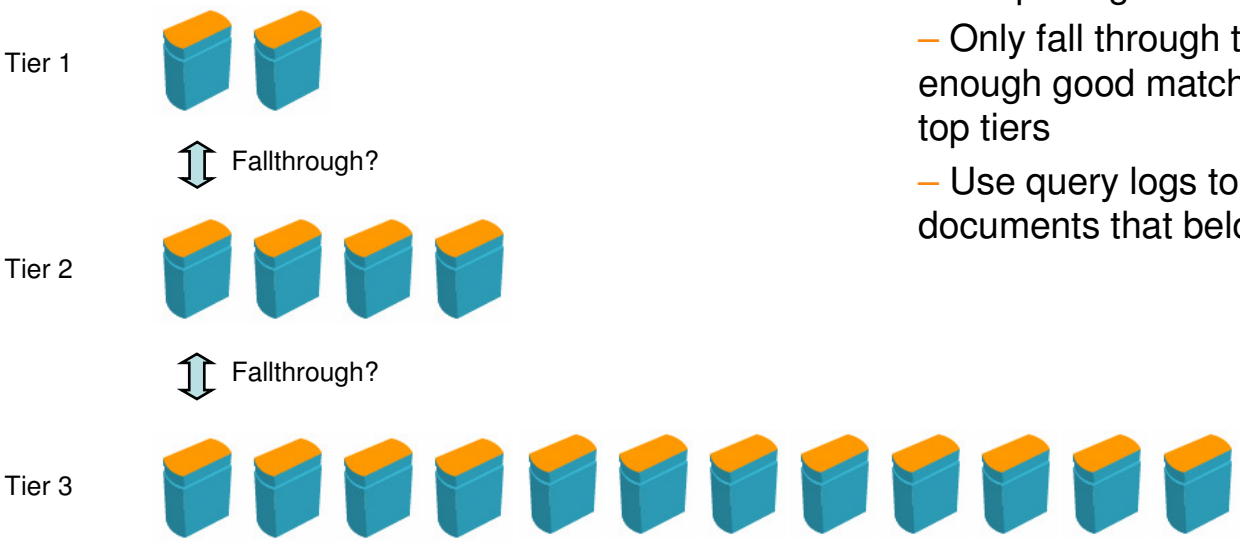


Web Search Challenges

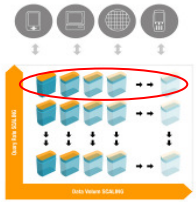
- **Business Model**
- **Architecture and operations**
  - Document volume, query volume and index freshness
- **Index quality**
  - Spam and offensive content detection
  - Duplicate detection
  - Algorithmic and/or editorial efforts
    - Characteristics must be defined
  - Filter or influence
- **Relevance**
  - Link cardinality
  - Anchor text
  - Freshness
  - Authorative sources
    - Editorial content

Engine architecture

- **For very big deployment scenarios**
  - Web scale, i.e., billions of documents



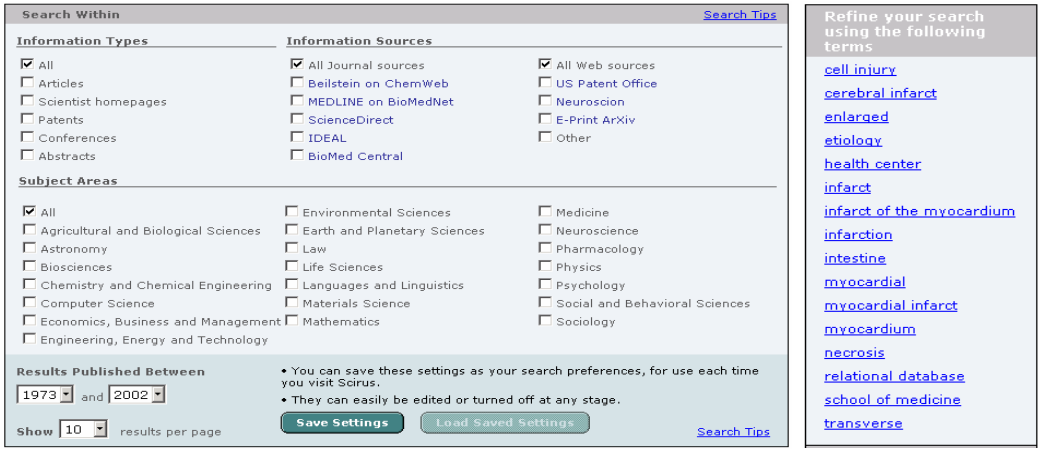
- **All search nodes are equal, but some are more equal than others**
  - Organize the search nodes into multiple tiers
  - Top tier nodes may have fewer documents and run on better hardware
  - Keep the good stuff in the top tiers
  - Only fall through to the lower tiers if not enough good matches are not found in the top tiers
  - Use query logs to decide which documents that belong in which tiers



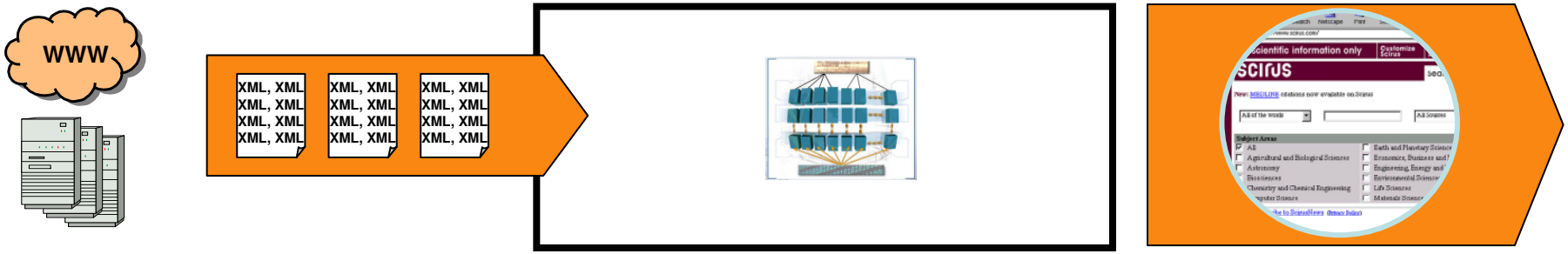
www.scirus.com

REED ELSEVIER
a world leading provider of information driven services and solutions

scirus
for scientific information only
infarct search



- **scirus.com – The Web's Science Search**
- **Scientific classification**
- **Combining web and corporate index**



90 M web pages
15 M Elsevier Science publications

- **Scientific classification**
- **Grouping and identification of related articles**

- **Leading science Index**
- **Understanding content**
- **Scientific navigation**

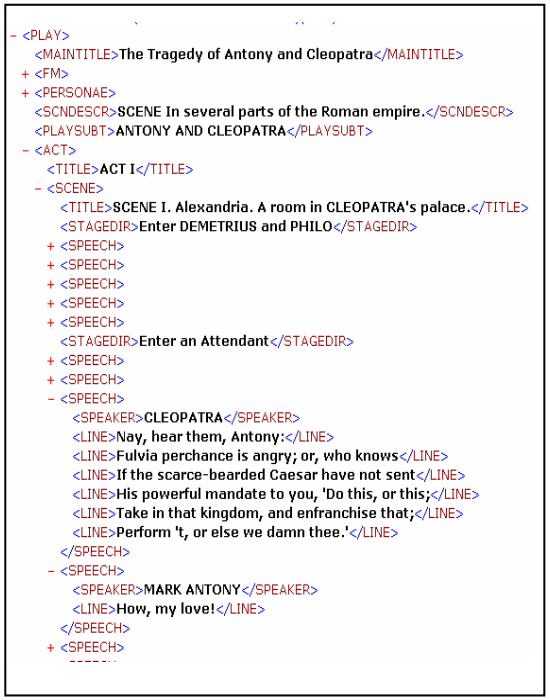
XML search

- **Moving beyond a flat document model**
  - From a simple (field, value) layout to complex, nested structure having scopes/tags
  - From a predefined index layout to schema flexibility

- **Some queries cannot be adequately handled without structure**
  - Flattening out the content won't quite work
  - False positives slip through

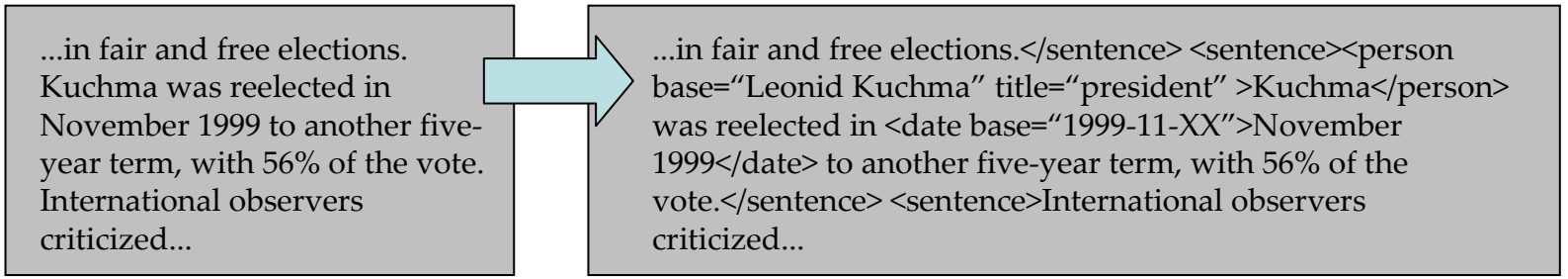
```
<authors>
  <author>John P. Brown</author>
  <author>George Smith</author>
</authors>
```

"Show me documents authored by John Smith"



Information extraction

- **Apply text mining techniques to identify entities of interest**
  - Structural and semantic regions
  - Makes unstructured data more structured
- **Mark them up in context**
  - Grammars as scope producers
  - Scopes can be annotated with meta data



- **Make it possible to act on the information!**
  - E.g., make it searchable in a way that preserves context

Scope

Scalable search

"Sentences where someone says something positive about Adidas."

xml:sentence:("adidas" and sentiment:@degree:>0)

"Paragraphs that contain quotations by George W. Bush, where he mentions a monetary amount."

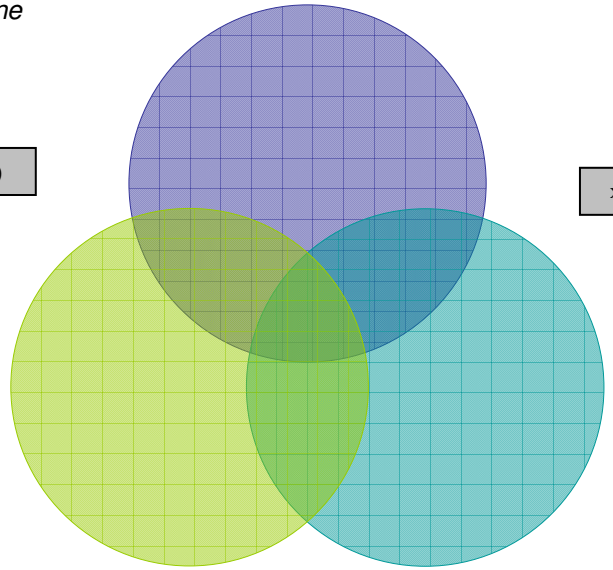
xml:paragraph:quotation:(@speaker:"bush" and scope(price)))

"Paragraphs that discuss a company merger or acquisition."

xml:paragraph:(string("merger", linguistics="on") and scope(company) and scope(price))

"Quotations where somebody says something about the Gaza Strip."

xml:quotation:("gaza strip")



XML technologies

Information extraction

"Sentences where the acronym 'MIT' is defined."

xml:sentence:acronym:(@base:"mit" and scope(@definition))

"Dates and locations related to the query 'd-day'."

xml:sentence:("d-day" and (scope(date) or scope(location))

fast

enabling information on demand™

XML search

- **Queries that combine structure and content**
  - Impose contextual constraints on the content
- **FAST Query Language (FQL)**
  - Partial overlap with XQuery
  - Linguistic extensions
- **Return matching scopes**
  - See the matching document fragments, including markup and annotations

xml:sentence:(“cancer” and scope(person))
≈
//sentence[fast:matches(., “cancer”) and .//person]

```
<matches>
  <match>
    <sentence>The publicist of <person>Robert De
Niro</person> announces that the actor has prostate
cancer.</sentence>
  </match>
  <match>
    <sentence><person>De Niro</person> was diagnosed
with cancer last week.</sentence>
  </match>
  <match>
    <sentence>”He’ll fight the cancer,” says
<person>John Barnes</person>, founder of his Welsh
fanclub.
  </match>
</matches>
```

Anatomy of a Search Platform