# Tight-Coupling: A Way of Building High-Performance Application Specific Engines

2003.3

**Kyu-Young Whang, Professor**
Department of Computer Science
Korea Advanced Institute of Science and Technology(KAIST)

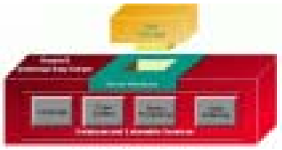
## Data in Internet Era

- The amount of data grows exponentially (10 times in 3~4 years)
  - Performance with such large DBs is an issue
- New database applications
  - Information Retrieval (IR)
  - Spatial Databases
  - Data Mining
  - OLAP
  - Data Streaming
  - Multimedia Databases
  - …

## Adding Application-Specific Features to DBMS

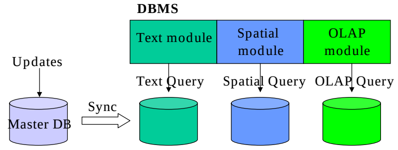
- Conventional techniques
  - Oracle's Cartridge
  - IBM DB2 Extender
  - Informix DataBlade

**Example**



⇒Performance not optimal due to relatively high-level interface with the main engine

- Database Module by Raghu Ramakrishnan
  - Build a suite of "data service" modules that can plug and play
  - Loose coupling across modules

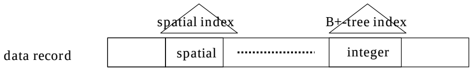


=> Performance and consistency issues

- New Technique: Tight Coupling (implementor's approach)
  - Direct implementation of a new feature at the level of records and indexes (e.g., at the same level of a B+-tree)
  - Examples
    - Tight Coupling of Spatial Features with DBMS
    - Tight Coupling of IR with DBMS
  - The engine code that must be modified to accommodate a new feature is not that big (approximately 20,000 lines)

## Tight-Coupling with Spatial Features

- Embedding spatial indexes



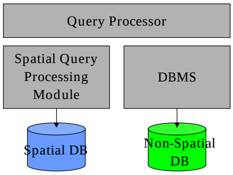
  - Integrated management of spatial and nonspatial attributes
- Queries

  Find the phone number of McDonald within 10km of Stanford University

```
SELECT s.phone_number
FROM universities u, stores s
WHERE DISTANCE(u.location, s.location)<10 AND
        u.name="Stanford" AND s.name="McDonald";
```

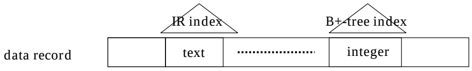
  - Integrated query against spatial and nonspatial attributes

- Conventional Systems (e.g., ArcInfo)



## Tight-Coupling with IR

- Embedding IR Indexes



  - Integerated management of text and nontext attributes

- Queries

  Find the papers that contain the word “database” in the title and that have been published after year 2000

```
SELECT p.oid
FROM paper p
WHERE MATCH(p.title, “database”)>0 AND p.pubyear>2000;
```

  - Integrated query against text and nontext attributes

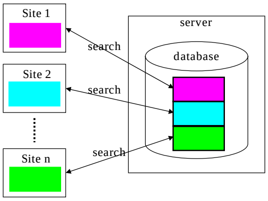
- Features

  - Crash recovery and fine-granularity locking and logging for the IR index
  - Bulk loading and bulk deletion for the IR index
  - Immediate update for the IR index

## Application of IR Index

- Site-limited Search (Hosted SiteSearch in Google)

  Find the web pages that include the word “SQL” from the site “Microsoft”



- Directory Search

  Find the web pages that include the word “park” from the directory “Recreation > Travel > Attractions”

## Implementation

- Example Schema

***siteInfo*** table

| column name | column type | description |
|---|---|---|
| siteId | integer | Site identifier |
| URL | varchar | Site URL |
| title | text | Site title |
| description | text | Site description |

***pageInfo*** table

| column name | column type | description |
|---|---|---|
| siteId | integer | Site identifier |
| siteIdText | text | Site identifier |
| title | text | Page title |
| URL | varchar | Page URL |
| content | text | Page content |

- Query

  Find the web pages that contain the word “Korea” from the site having siteId=6000 (Site-limited Search)

```
SELECT p.oid
FROM pageInfo p
WHERE MATCH(p.content, “korea”)>0 AND p.siteId = 6000;
```

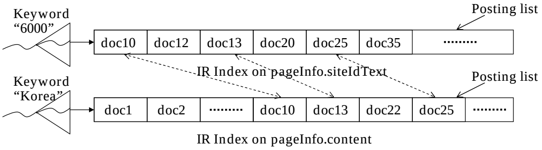
- Naïve Implementation

  - Find the record of the web page containing the word “Korea”
  - Access the records and select those whose siteId = 6000

  ⇒Very bad in performance

  (Tuples are scattered all over the database causing excessive random accesses)

- IR Index Join Method
  - We use ***siteIdText*** of type ‘text’ instead of ***siteId*** of type integer
  - Do join between the posting list (matching “Korea”) from the IR Index on the attribute ***content*** and the posting list (matching “6000”) from the IR Index on the attribute ***siteIdText***



Keyword “6000” → doc10 | doc12 | doc13 | doc20 | doc25 | doc35 | ·········  Posting list

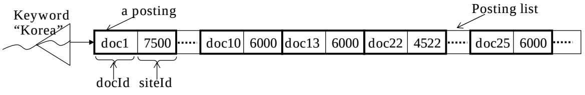
IR Index on pageInfo.siteIdText

Keyword “Korea” → doc1 | doc2 | ········· | doc10 | doc13 | doc22 | doc25 | ·········  Posting list

IR Index on pageInfo.content

⇒Performance is good since join is done in the indexes without accessing tuples

- Embedded Posting Method
  - Embedded Posting: Other attribute values of the record are embedded in the posting of another attribute
  - In this example, ***siteId*** (of type integer) is embedded in the posting of the attribute ***content***



Keyword “Korea” → doc1 | 7500 ····· doc10 | 6000 | doc13 | 6000 | doc22 | 4522 ····· doc25 | 6000 ·····

a posting

Posting list

docId siteId

IR Index on pageInfo.content

⇒Performance is good since query is resolved by accessing only one posting list without accessing the tuples (vs. space overhead)

## ODYSSEUS Object-Relational DBMS

- Being developed at KAIST for over thirteen years
- Tightly coupled with IR (IR Index – patented) and Spatial (MLGF) features
- A DBMS and, at the same time, a search engine
  - Concurrency control and recovery (coarse granularity and fine granularity)
  - IR performance is comparable or better than commercial search engines
  - Allows immediate updates
- An object-relational DBMS and, at the same time, a spatial DBMS
  - Integrated management of spatial and nonspatial attributes
- Many commercial applications
- Approximately 400,000 lines of C / C++ (precision) codes