PEMEROLEHAN INFORMASI MENGGUNAKAN *INVERTED INDEX* DENGAN STRUKTUR DATA KLASIK VS ORDBMS

J.B. Budi Darmawan

Dosen Program Studi Teknik Informatika, Fakultas Sains dan Teknologi,
Universitas Sanata Dharma
Alamat korespondensi: Kampus III Paingan Maguwohardjo, Depok, Sleman, Yogyakarta
E-mail: *b.darmawan@usd.ac.id, jbbudi@gmail.com*

ABSTRACT

Engineering a Web search engine offering efficient information retrieval is a challenging task. The advances in DBMS for multicore and clustered database can transparently benefit information retrieval systems using inverted index that are built on top. A design choice is that inverted index is based on an object-relational database system using nested table collection. This paper discusses the performance of this choice compared to the classical inverted index. Based on the test results using 286 papers in Indonesian language and 25889 different terms with 1 to 4 keywords, the query time using ORDBMS inverted index is in 10^{-2} second time scala compared to the query time using classical inverted index is in 10^{-3} second time scala.

***Key words**: information retrieval, inverted index, ordbms, data structure.*

1. PENDAHULUAN

Seiring dengan perkembangan penelitian di Indonesia semakin banyak makalah ilmiah yang dihasilkan. Sistem pemerolehan informasi menawarkan kemampuan menyediakan informasi yang dibutuhkan pemakai. Berbagai teknik rekayasa sistem pemerolehan informasi yang menawarkan kemampuan menyediakan informasi yang relevan telah dicoba untuk mendapatkan kelebihan serta peningkatan efisiensi dan efektivitas.

Untuk menjawab *query, inverted index* terbukti efisien diterapkan pada kebanyakan sistem pemerolehan informasi dan mesin pencari web (Baeza-Yates, 1999). Implementasi *inverted index* menggunakan *Database Management System* (DBMS) menawarkan beberapa kelebihan selain kekurangannya (Papadakos, 2008). Menggunakan DBMS perluasan skema indeks dengan perluasan tambahan kolom maupun relasi untuk melebarkan spektrum dari fungsional yang ditawarkan dapat dilakukan dengan mudah. DBMS juga menangani lapisan fisik sehingga tidak diperlukan pembuatan dan penggabungan indeks partial untuk mengkonstruksi indeks dari sebuah *corpus* yang besar (Papadakos, 2008).

Dalam sistem pemerolehan informasi klasik, indeks untuk menjawab *query* dan indeks untuk memperbarui dibedakan, dengan DBMS perbedaan dan duplikasi indeks untuk menjawab *query* dan indeks untuk memperbarui ini tidak diperlukan. Indeks tunggal dapat digunakan karena tidak harus membuat indeks *partial*. Sistem pemerolehan informasi berbasis DBMS dapat memanfaatkan kemampuan DBMS yang dapat mendukung sistem *multicore* dan *cluster database* (Papadakos, 2008).

Di samping kelebihan diatas ada beberapa hal yang harus dipertimbangkan saat menggunakan *Relational Database Management System* (RDBMS) (Papadakos, 2008). Implementasi dalam sebuah RDBMS akan menempati lebih banyak ruang penyimpan daripada sebuah *inverted index*. *Inverted index* terdiri dari data dalam bentuk (t, occ) dimana t adalah *term* atau kata dan occ adalah *occurrence* atau dokumen (d) dari t dalam *corpus*. Sebagai contoh data $(t,\{d_1,d_3,d_5\})$ dalam *inverted index*, akan direpresentasikan dalam tiga tuple $[t,d_1]$, $[t,d_3]$, $[t,d_5]$ dalam sebuah RDBMS yang berakibat pemborosan ruang penyimpan. Sebagai bagian dari penggunaan ruang penyimpan yang lebih besar, waktu tanggapan *query* akan lebih tinggi untuk DBMS yang berdasarkan indeks, karena semakin banyak operasi I/O yang harus dilakukan. Namun pemborosan ruang penyimpan ini tidak terjadi saat menggunakan *Object Relational Database Management System* (ORDBMS)

dengan penerapan menggunakan *collections.* Hal ini karena implementasi *collections* dalam ORDBMS mempunyai struktur data yang mirip dengan *inverted index*, sehingga data (t,{d_1,d_3,d_5}) dapat disimpan dalam sebuah *tuple* dengan data {d_1,d_3,d_5} diletakkan dalam sebuah *collections* (Connoly, 2005).

Implementasi *inverted index* dengan *stemmer* bahasa Yunani menggunakan *Object Relational Database Management System* (ORDBMS) menawarkan kecepatan yang lebih baik dan penggunaan memori yang lebih efisien dibandingkan dengan *Relational Database Management System* (RDBMS) (Papadakos, 2008). Implementasi *inverted index* untuk mendukung model pemerolehan boolean dengan ORDBMS tebukti menawarkan kemampuan yang lebih baik dibandingkan dengan RDBMS untuk dokumen berbahasa Indonesia dengan kelompok df kata lebih besar 2500 (Darmawan, 2012).

Agar hasil dokumen pencarian sesuai dengan kebutuhan pemakai, maka dilakukan teknik pembobotan untuk membantu proses perangkingan. *Term frequency-inverse document frequency* (tf-idf) dengan rumus Savoy menawarkan teknik pembobotan yang telah dinormalisasi dan jika sebuah istilah mempunyai frekuensi kemunculan yang sama pada dua dokumen belum tentu mempunyai bobot yang sama. Teknik ini telah dicobakan dalam dokumen berbahasa Indonesia. (Hasibuan, 2001).

Mempertimbangkan kelebihan yang ditawarkan oleh sebuah ORDBMS, dalam penelitian ini penulis mengambil rumusan masalah: Bagaimana implementasi pemerolehan informasi dengan tf-idf untuk makalah ilmiah berbahasa Indonesia menggunakan *inverted index* dengan struktur data klasik vs *inverted index* dengan ORDBMS.

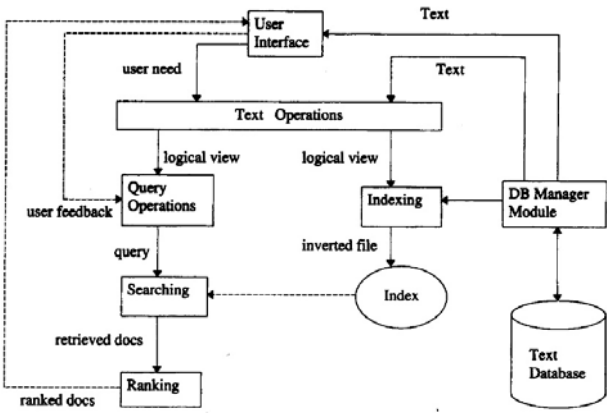
Penelitian ini bertujuan untuk mengimplementasikan dan mengamati unjuk kerja sistem pemerolehan informasi model tf-idf dengan *inverted index* menggunakan struktur data klasik vs *inverted index* menggunakan ORDBMS.

Penelitian ini bermanfaat sebagai alternatif penerapan sistem pemerolehan informasi model tf-idf *inverted index* ke dalam ORDBMS untuk memperoleh kelebihan yang ditawarkannya. Hasil penelitian ini dapat ditindaklanjuti agar dapat digunakan untuk menyimpan makalah-makalah ilmiah berbahasa Indonesia yang semakin banyak jumlahnya agar mudah dicari sesuai dengan kebutuhan pemakai.

2. LANDASAN TEORI

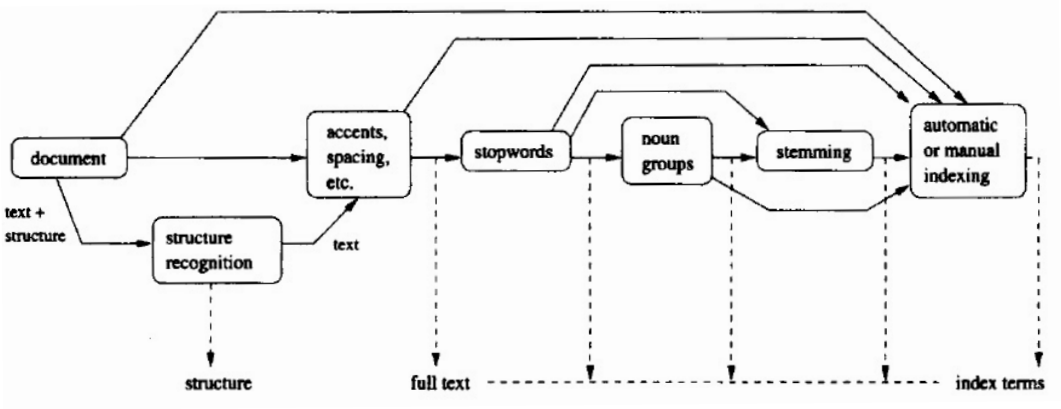
2.1 Pemerolehan Informasi

Tahap-tahap dalam proses pemerolehan informasi disajikan pada Gambar 1. Tahap pertama diperlukan pendefinisian basisdata teks (*database text*) dengan melakukan identifikasi terhadap koleksi dokumen asal (*corpus*) yang digunakan, operasi yang akan dilakukan pada teks dan model teks (struktur dan elemen teks yang dipakai). Operasi teks (*Text operation*) diperlukan untuk mendapatkan *logical view documents* dari dokumen asal. Setelah tahap ini selanjutnya dibuat *index term* dengan struktur yang banyak digunakan berupa *inverted file* (*inverted index*). Setelah indeks terbentuk maka proses pemerolehan (*retrieval*) dapat dimulai dengan penentuan kebutuhan pemakai yang kemudian dilakukan operasi teks seperti yang digunakan saat pembuatan indeks dari dokumen asal. Hasil operasi teks ini digunakan dalam operasi query untuk melakukan pencarian menggunakan *inverted file* yang telah dihasilkan sebelumnya sekaligus dilakukan perhitungan bobot untuk melakukan proses perangkingan agar dokumen yang ditemukan sesuai dengan kebutuhan pemakai (Baeza-Yates, 1999).



Gambar 1. Proses Pemerolehan Informasi
(Baeza-Yates, 1999)

Operasi teks yang diperlukan dalam *logical view documents* disajikan pada Gambar 2. Pada tahap pemrosesan awal ini dilakukan proses perubahan dari *full text* menjadi *set index term.* Pada tahap ini dilakukan operasi stopword untuk menghilangkan kata-kata umum dan operasi stemming untuk mendapatkan *root word.*



Gambar 2. Operasi Teks dalam *Logical View Documents* (Baeza-Yates, 1999)

2.2 Object Relational Database Management System

ORDBMS merupakan perluasan dari RDBMS dengan memiliki sifat berbasis obyek seperti *user-extensible type*. Fasilitas ini merupakan perluasan obyek dari SQL yang merupakan bagian dari standard SQL:2003 (Standard 1999 dan standard 2003). Salah satu tipe yang ada adalah *collection* yang dapat digunakan untuk menyimpan banyak nilai dalam sebuah kolom tunggal dari sebuah tabel yang akan menghasilkan *nested table* dimana sebuah kolom dalam sebuah tabel dapat berisi tabel yang lain (Connoly, 2005).

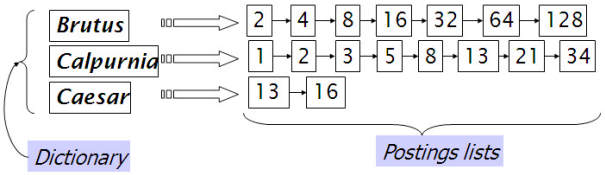
Multiset adalah sebuah tipe *collection* dengan elemen yang tidak terurut yang memungkinkan adanya duplikasi. Tidak seperti array, *multiset* tidak dibatasi oleh ukuran maksimum yang ditentukan di awal. Operator diperlukan untuk mengkonversi sebuah *multiset* ke sebuah tabel (Connoly, 2005).

Oracle mendukung *multiset* dengan menyediakan tipe data *nested table*. Untuk melakukan operasi DML, operator TABLE digunakan untuk memperlakukan sebuah *nested table* seperti tabel biasa. Meskipun elemen dalam sebuah *nested table* tidak urut, Oracle menyediakan fasilitas indeks pada *nested table* (Oracle, 2013).

2.3 Struktur Data *Inverted Index*

Representasi struktur data *inverted index* yang disajikan pada Gambar 3 menunjukkan *dictionary* yang berisi kumpulan *term* (t) dengan masing-masing *term* mempunyai *posting list* yang berisi kumpulan *document* (d). Sehingga setiap *posting list* dapat direpresentasikan t,{d_1,d_3,d_5}.

Struktur data *nested table* dari ORDBMS Oracle di atas sesuai dengan representasi *inverted index* ini (t,{d_1,d_3,d_5}) dibandingkan dengan implementasi menggunakan RDBMS. Hal ini karena dalam RDBMS, struktur data yang digunakan harus diubah menjadi tiga tuple [t,d_1], [t,d_3], [t,d_5] yang membutuhkan duplikasi t sebanyak d (Papadakos, 2008).



Gambar 3. Representasi *Inverted Index* (Darmawan, 2012)

Operasi model pemerolehan *boolean* dasar meliputi operasi AND, OR dan NOT. Untuk *inverted index* yang disajikan pada Gambar 3, dapat dilakukan operasi-operasi *boolean* dasar. Operasi AND dengan *n operand* akan melibatkan *n posting list*. Operasi Brutus AND Calpurnia dapat dilakukan dengan algoritma interseksi untuk kedua *posting list* Brutus dan Calpurnia seperti tersaji pada Gambar 4. Operasi ini menghasilkan dokumen 2 dan 8.

```
INTERSECT(p1, p2)
 1  answer ← ⟨ ⟩
 2  while p1 ≠ NIL and p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then ADD(answer, docID(p1))
 5             p1 ← next(p1)
 6             p2 ← next(p2)
 7        else if docID(p1) < docID(p2)
 8                then p1 ← next(p1)
 9                else p2 ← next(p2)
10  return answer
```

Gambar 4. Algoritma Interseksi dari Dua Posting List *p*1 dan *p*2 (Manning, 2008)

Salah satu alternatif implementasi *operator* AND untuk representasi data dalam RDBMS maupun *nested table* dari ORDBMS dari *inverted index* seperti tersaji pada Gambar 3, dapat memakai operasi *intersection* (Connoly, 2005) menggunakan operasi dasar persamaan (1), operasi ini dapat diimplementasikan menggunakan *operator INTERSECT* pada operasi SQL.

$$R \cap S \quad \ldots (1)$$

Struktur data klasik *Hash Table* dapat memetakan key term ke lokasi yang tepat dengan relatif cepat dengan kompleksitas O(1) (Lafore, 2003). Struktur data ini sesuai dengan struktur *dictionary* seperti tersaji pada Gambar 3. Sedangkan struktur data klasik *Linked List* bersifat dinamis karena ukurannya dapat bertambah dan berkurang mengikuti kebutuhan jumlah node (Lafore, 2003). Struktur data *Linked List* ini sesuai dengan kebutuhan untuk struktur *posting list* pada Gambar 3.

2.4 Pembobotan TF-IDF Savoy

Teknik pembobotan Savoy (1993) (Hasibuan, 2001) disajikan pada persamaan (2) dan (3) berikut ini:

$$W_{ik} = ntf_{ik} * nidf_k, \quad \ldots (2)$$

dimana

$$ntf_{ik} = \frac{tf_{ik}}{Max_j tf_{ij}} \text{ dan } nidf_k = \frac{\log\left[\frac{n}{df_k}\right]}{\log(n)} \quad \ldots (3)$$

Dimana:

- W_{ik} adalah bobot istilah k pada dokumen i.
- tf_{ik} merupakan frekuensi dari istilah k dalam dokumen i.
- n adalah jumlah dokumen dalam kumpulan dokumen.
- df_k adalah jumlah dokumen yang mengandung istilah k.
- $Max_j\ tf_{ij}$ adalah frekuensi istilah terbesar pada satu dokumen.
- Wd= bobot sebuah dokumen

Pada teknik pembobotan ini, dalam menentukan bobot yang ternormalisasi suatu istilah tidak hanya berdasarkan frekuensi *term*, tetapi juga berdasarkan frekuensi terbesar pada dokumen bersangkutan sehingga dapat ditentukan posisi relatif bobot dari *term* dibanding dengan *term-term* lain di dokumen yang sama. Teknik ini juga memperhitungkan jumlah dokumen yang mengandung *term* bersangkutan dan jumlah keseluruhan dokumen sehingga dapat ditentukan posisi relatif bobot *term* bersangkutan pada suatu dokumen dibandingkan dengan dokumen-dokumen lain yang memiliki *term* yang sama. Sehingga jika sebuah *term* memiliki frekuensi yang sama pada dua dokumen belum tentu memiliki bobot yang sama (Hasibuan, 2001).

Dengan kata lain rumus Savoy(1993) mempertimbangkan seberapa besarnya frekuensi istilah jika dibandingkan dengan frekuensi istilah terbesar pada dokumen, dan juga mempertimbangkan posisi dokumen yang mengandung istilah dimaksud terhadap jumlah dokumen keseluruhannya (Hasibuan, 2001).

3. METODOLOGI PENELITIAN

Dalam penelitian ini dilakukan tahap-tahap eksperimen dalam laboratorium sebagai berikut:

1) Studi pustaka penerapan konsep pemerolehan informasi menggunakan struktur data klasik vs ORDBMS.
2) Pengumpulan dokumen-dokumen makalah ilmiah berbahasa Indonesia sebagai *corpus* sejumlah 286 dokumen.
3) Implementasi penerapan konsep pemerolehan informasi menggunakan struktur data klasik vs ORDBMS.
4) Pengamatan unjuk kerja waktu *query* dan jumlah hasil *query* sebagai implementasi pada langkah 3 untuk operasi AND dilakukan pada empat kelompok kata berdasarkan jumlah dokumen yang memenuhi suatu kata atau *document frequencies (df)*. Ketiga kelompok kata ini adalah kelompok kata yang mempunyai *df* 1, *df* 2, *df* 144 dan *df* 286. Operasi AND dilakukan dengan menggunakan 1 sampai 4 *operand* kata atau dengan kata lain menggunakan 0 sampai 3 *operator*.
5) Hasil yang diperoleh dari implementasi *inverted index* menggunakan struktur data klasik dan ORDBMS pada langkah 4 akan dibandingkan.

Penelitian ini dilakukan dengan menggunakan sebuah komputer dengan spesifikasi sebagai berikut:

1) Perangkat lunak
   a) Sistem operasi Linux 64 bit Centos 59 di dalam Virtual Box(Oracle, 2013) v4.2.4 dengan RAM 3027MB

b) Oracle 11G Release 2 Express Edition (Oracle, 2013)
c) Oracle SQL Developer (2.1.1.64) (Oracle, 2013)
d) Java JDK 1.6.0 dan JDBC (Oracle, 2013)
e) Netbeans 6.9.1 (Netbeans, 2013)

1) Perangkat keras
a) Prosesor Intel Core 2 Quad 6600
b) Memori RAM 8 GB/5300 DDR2
c) Hardisk 160 GB SATA 2
d) Motherboard chipset Intel DP35DP

4. HASIL PENELITIAN DAN PEMBAHASAN

4.1 Data yang Digunakan

Penelitian ini menggunakan data dari 286 dokumen dari prosiding seminar di bidang teknologi informasi. Dokumen prosiding seminar yang tersedia masih berupa satu file besar untuk seluruh dokumen. Pengolahan dilakukan dengan melakukan pemecahan dokumen tersebut menjadi beberapa dokumen untuk setiap makalah. Dokumen sumber yang telah dipecah selanjutnya melewati pemrosesan teks seperti tersaji pada Gambar 2 untuk kemudian disimpan dalam basis data dengan perancangan logikal disajikan pada Gambar 5. Dari proses ini dihasilkan 25889 *term* dengan frekuensi dokumen untuk setiap *term* antara 1 sampai 286 dokumen.

4.2 Implementasi Struktur Data Klasik untuk Inverted Index

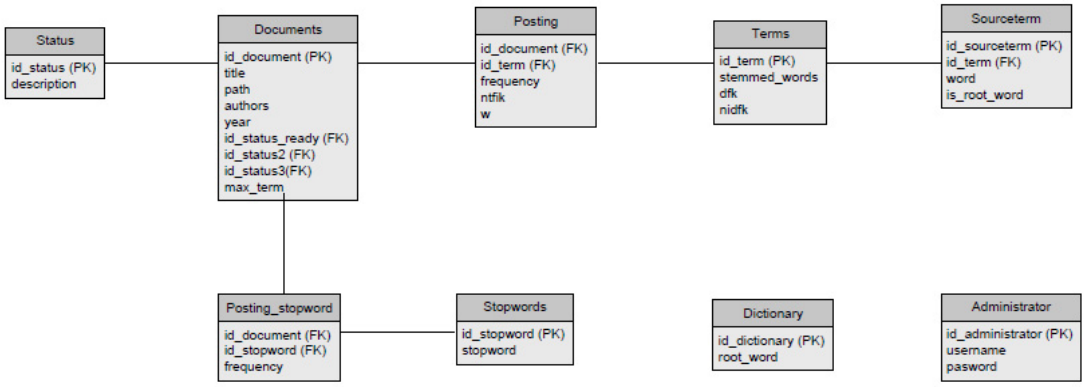
Dalam struktur data klasik, implementasi *inverted index* menggunakan kelas library java *Hashtable* untuk *dictionary* dengan *key* berupa *term* yang disimpan dan *value* berisi referensi ke LinkedList terurut untuk menyimpan posting list. LinkedList terurut dibuat dengan mengembangkan kelas LiskedList dalam library java.

4.3 Implementasi Basis Data ORDBMS untuk Inverted Index

Implementasi basisdata ORDBMS untuk *inverted index* disajikan pada Gambar 6. Data term akan disimpan dalam table term yang juga berisi posting list posting_nested_term bertipe *nested table* yang menyimpan dokumen yang mengandung *term* tersebut.

4.4 Implementasi Operasi AND dalam Inverted Index Menggunakan Struktur Data Klasik vs ORDBMS

Dalam struktur data klasik, operasi AND untuk kata kunci lebih dari satu dapat langsung menerapkan algoritma yang disajikan pada Gambar 4. Method untuk operasi AND *posting list* ini disajikan pada Gambar 7 di bawah ini. Dalam implementasi ini saat melakukan proses AND dapat sekaligus disisipkan operasi untuk menjumlahkan bobot dokumen yang berinterseksi.



Gambar 5. Perancangan Logikal basis data untuk pemrosesan teks

```
create or replace TYPE document_type AS OBJECT (
document_id NUMBER,
title VARCHAR2(256),
year NUMBER,
authors VARCHAR2(100),
maxjtfij NUMBER,
path VARCHAR2(256));
/
create or replace TYPE posting_type AS OBJECT (
tfik NUMBER,
ntfik DECIMAL(10,5),
w NUMBER,
document_posting REF document_type);
/
CREATE TYPE posting_nested_type AS TABLE OF posting_type;
/
create or replace TYPE term_type AS OBJECT (
term_id NUMBER,
term VARCHAR2(50),
dfk NUMBER,
nidfk NUMBER,
posting_nested_term posting_nested_type);
/
CREATE TABLE document of document_type (
document_id not null,
primary key (document_id));
/
CREATE TABLE term OF term_type (
term_id NOT NULL,
PRIMARY KEY (term_id))
NESTED TABLE posting_nested_term STORE AS posting_nested_term_table
((PRIMARY KEY (NESTED_TABLE_ID, document_id))
 ORGANIZATION INDEX COMPRESS
);
/
```

Gambar 6. Perancangan Fisikal ORDBMS untuk *Inverted Index*

```
// method untuk operasi AND dua posting
    public Term and(Term term1, Term term2) {
        ListIterator<Posting> termIterator1;
        ListIterator<Posting> termIterator2;
        // termIterator1 diisi posting dengan dokumen lebih sedikit
        if (term1.getPostingList().size() < term2.getPostingList().size()) {
            termIterator1 = term1.getPostingList().listIterator();
            termIterator2 = term2.getPostingList().listIterator();
        } else {
            termIterator1 = term2.getPostingList().listIterator();
            termIterator2 = term1.getPostingList().listIterator();
        }
        Term result = new Term("(" + term1.getTerm() + " AND " + term2.getTerm() + ")");
        result.setDfk(0);
        Posting bantu1 = null;
        if (termIterator1.hasNext()) {
            bantu1 = termIterator1.next();
        }
        Posting bantu2 = null;
        if (termIterator2.hasNext()) {
            bantu2 = termIterator2.next();
        }
        while (bantu1 != null && bantu2 != null) {
            //jika lokasi dokumen kata2 kedua sama dengan kata1 1
            if (bantu1.getDocument().getIdDocument() == bantu2.getDocument().getIdDocument())
            {
                bantu1.setW(bantu1.getW() + bantu2.getW());
                result.getPostingList().add(bantu1);
                result.incDfk();

                if (termIterator1.hasNext() && termIterator2.hasNext()) {
                    bantu1 = termIterator1.next();
                    bantu2 = termIterator2.next();
                } else {
                    bantu1 = bantu2 = null;
                }
            } else if (bantu1.getDocument().getIdDocument() <
                       bantu2.getDocument().getIdDocument()) {
                if (termIterator1.hasNext()) {
                    bantu1 = termIterator1.next();
                } else {
                    bantu1 = null;
                }
            } else if (bantu1.getDocument().getIdDocument() >
                       bantu2.getDocument().getIdDocument()) {
                if (termIterator2.hasNext()) {
                    bantu2 = termIterator2.next();
                } else {
                    bantu2 = null;
                }
            }
        }
        return result;
    }
```

Gambar 7. Implementasi Operasi AND dalam Inverted Index Menggunakan Struktur Data Klasik

Sedangkan implementasi operasi AND untuk kata kunci lebih dari satu dalam *inverted index* menggunakan ORDBMS dilakukan dengan melakukan dua operasi. Operasi pertama adalah operasi untuk melakukan operasi AND dengan menggunakan operator SQL INTERSECT dan operasi kedua untuk menjumlahkan bobot dokumen yang berinterseksi dengan menggunakan fungsi SQL *aggregate* SUM dan operator SQL GROUP BY. Operasi pertama menjadi sub query dari operasi kedua. Hal ini membuat AND untuk kata kunci lebih dari satu dalam inverted index menggunakan ORDBMS kurang efisien dibandingkan dengan struktur data klasik.

4.5 Hasil Percobaan

Sistem pemerolehan informasi ini telah berhasil dikembangkan dengan teknologi jsp dan servlet java berbasis web dengan Oracle Database Express Edition 11g. Tampilan hasil pencarian dapat dilihat pada Gambar 8.

kelompok kata *document frequencies (df)* disajikan pada tabel 1. Dari tabel 1 terlihat bahwa terjadi perbedaan skala waktu untuk sistem pemerolehan informasi menggunakan invented indek dengan struktur data klasik dan ORDBMS. Skala waktu saat menggunakan ORDBMS berada pada 10^{-2} detik sedangkan dengan struktur data klasik berada pada 10^{-3} detik. Jadi untuk keempat kelompok *df* 1, *df* 2, *df* 144 dan *df* 286 dengan kata kunci 1 sampai 4 terjadi perbedaan skala waktu 1 digit desimal.

Pada Gambar 9 terlihat bahwa pada penerapan inverted index dengan ORDBMS menunjukkan peningkatan waktu yang hampir linear sebanding dengan peningkatan jumlah kata kunci yang digunakan dalam pencarian. Untuk pencarian dengan kata kunci 1 sampai 4 yang memiliki *df* 1, *df* 2, *df* 144 dan *df* 286, waktu yang dibutuhkan berkisar antara 0,019 detik sampai 0,096 detik. Sedangkan pada penerapan inverted index dengan struktur data klasik peningkatan ini baru terlihat pada kelompok df 286 dengan jumlah

Gambar 8. Tampilan Sistem Pemerolehan Informasi *Search Engine* Makalah Ilmiah

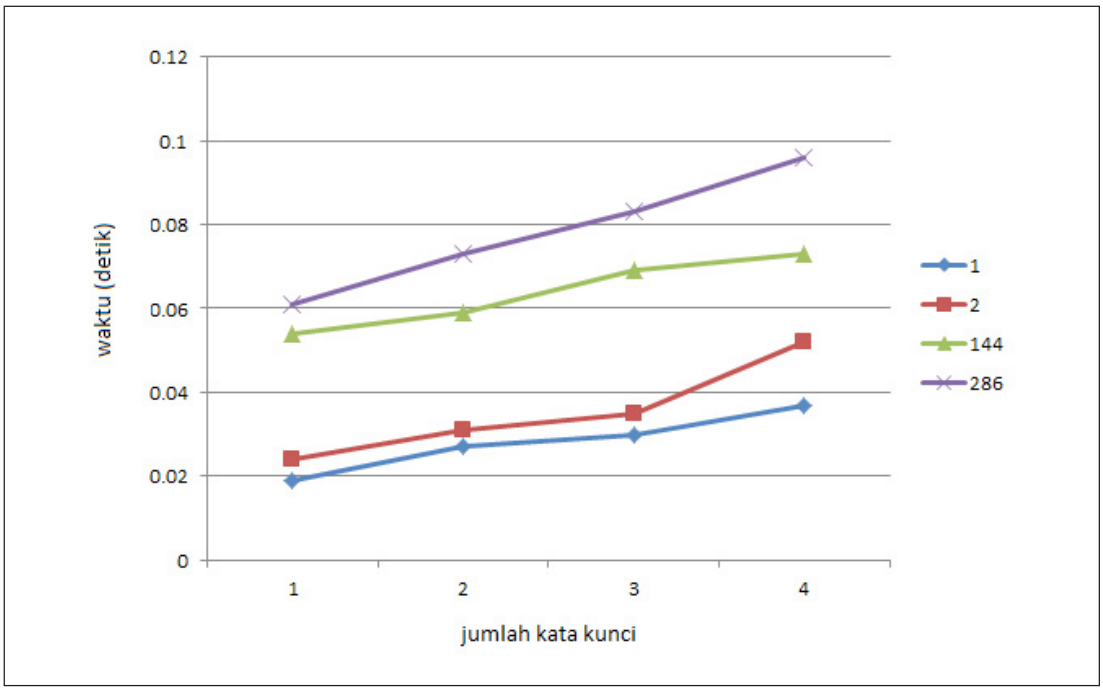
Pengamatan waktu diukur dimulai saat pemanggilan *method* untuk pencarian sampai dihasilkan nilai balik bertipe ArrayList dari dokumen yang ditemukan. Hasil pengamatan waktu untuk empat

kata kunci dari dua menjadi tiga. Hal ini kemungkinan karena keterbatasan kemampuan sistem operasi dan pemrograman java untuk melihat skala waktu yang lebih kecil.

Tabel 1: Waktu proses pencarian menggunakan struktur data klasik dan ORDBMS

Df kata	Jumlah Operand untuk Operasi AND	Rata-rata Waktu ORDBMS	Rata-rata Waktu Query Struktur Data Klasik
1	1	0,019	0,001
	2	0,027	0,001
	3	0,030	0,001
	4	0,037	0,001
2	1	0,024	0,001
	2	0,031	0,001
	3	0,035	0,001
	4	0,052	0,001
144	1	0,054	0,001
	2	0,059	0,001
	3	0,069	0,001
	4	0,073	0,001
286	1	0,061	0,001
	2	0,073	0,001
	3	0,083	0,002
	4	0,096	0,002

Pengamatan skala waktu 10^{-2} detik menggunakan ORDBMS dengan hasil pencarian dibawah 286 dokumen ini masih dibawah pengamatan skala waktu 10^{-1} detik untuk pencarian menggunakan Google. Namun hasil pencarian Google mencapai jutaan dokumen. Hal ini menunjukkan bahwa penggunaan ORDBMS untuk implementasi inverted index dapat dipertimbangkan untuk mendapatkan kelebihan yang ditawarkan teknologi basis data seperti penggunaan *cluster database* yang dapat membagi beban pada beberapa komputer untuk mendukung peningkatan skalabilitas data. Dengan struktur data klasik pemanfaatan *cluster* komputer untuk mendukung pencarian memerlukan pemrograman yang lebih komplek seperti pemrograman *distributed processing*.



Gambar 9. Grafik Peningkatan Waktu Pencarian Terhadap Jumlah Kata Kunci 1 sampai 4 untuk Kelompok *df* 1, *df* 2, *df* 144 *dan df* 286

5. PENUTUP

5.1 Kesimpulan

Pada penerapan *inverted index* ke dalam struktur data klasik dan ORDBMS untuk mendukung model pemerolehan informasi tfidf dengan operasi dasar AND menggunakan *corpus* 286 dokumen makalah ilmiah berbahasa Indonesia yang menghasilkan 25889 kata yang berbeda menunjukkan hasil di bawah ini.

Penerapan inverted index dengan ORDBMS menunjukkan peningkatan waktu yang hampir linear sebanding dengan peningkatan jumlah kata kunci yang digunakan dalam pencarian untuk pencarian dengan kata kunci 1 sampai 4 yang memiliki *df* 1, *df* 2, *df* 144 dan *df* 286, waktu yang dibutuhkan berkisar antara 0,019 detik sampai 0,096 detik.

Waktu pencarian dengan kata kunci 1 sampai 4 yang memiliki *df* 1, *df* 2, *df* 144 dan *df* 286 dengan kata kunci 1 sampai 4 mempunyai perbedaan skala waktu 1 digit desimal yaitu skala waktu 10^{-3} untuk struktur data klasik dan skala waktu 10^{-2} detik untuk ORDBMS. Sehingga penggunaan ORDBMS dapat dipertimbangkan untuk mengurangi kompleksitas implementasi *distributed processing* dengan menggunakan teknologi *cluster database* dengan memanfaatkan kemampuan beberapa komputer untuk mendukung peningkatan skalabilitas data.

5.2 Penelitian selanjutnya

Penelitan selanjutnya dapat dilakukan untuk menerapkan penggunaan beberapa komputer untuk mendukung peningkatan skalabilitas data dalam pemerolehan informasi dan membandingkan penerapan *inverted index* menggunakan teknologi ORDBMS dengan struktur data klasik dalam *distributed processing*.

DAFTAR PUSTAKA

Baeza-Yates, R. and Ribeiro-Neto, B. 1999. Modern Information *Retrieval*. Addison Wesley.

Connoly, T. and Begg, C. 2005. *Database Systems: A Practical Approach to Design, Implementation, and Management, 4th edition*. England: Pearson Education Limited.

Darmawan, J.B.B. 2012. "Implementasi Invertex Index untuk Mendukung Model Pemerolehan Boolean Menggunakan RDBMS vs. ORDBMS". *Prosiding Konferensi Nasional Sistem Informasi (KNSI)* STMIK STIKOM Bali.

Hasibuan, Zainal A.,& Andri, Yofi. 2001. "Penerapan Berbagai Teknik Sistem Temu-Kembali Informasi Berbasis Hiperteks". *Jurnal Ilmu Komputer dan Teknologi Informasi*. Volume 1, Nomor 2.

Lafore, R. 2003. *Data Structure & Algorithms in Java Second Edition*. Indiana: Sams Publishing.

Manning, C.D., Raghavan, P. and Schütze, H. 2008. Introduction to Information *Retrieval*. Cambridge University Press.

Netbeans. 2013. *http://netbeans.org*. 2013. Diakses Tanggal 15 Desember 2013.

Oracle. 2013. *http://www.oracle.com*. 2013. Diakses Tanggal 15 Januari 2013.

Papadakos, et. All. 2008. *Mitos: Design and Evaluation of a DBMS-based Web Search Engine*. IEEE.