IMPLEMENTASI INVERTED INDEX DENGAN SISTEM ORDBMS MENGGUNAKAN COLLECTION UNTUK MENDUKUNG MODEL PEMEROLEHAN BOOLEAN

JB Budi Darmawan
Jurusan Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Sanata Dharma
Kampus III Paingan Maguwohardjo Depok Sleman Yogyakarta
E-mail: b.darmawan@staff.usd.ac.id, jbbudi@yahoo.com

ABSTRAK
Inverted index yang diterapkan pada kebanyakan sistem pemerolehan informasi dan mesin pencari web terbukti sangat efisien untuk menjawab query. Implementasi sistem pemerolehan menggunakan sistem manajemen basisdata akan memperoleh kelebihan yang ditawarkan. Dalam paper ini peneliti mencoba melakukan penerapan inverted index ke dalam ORDBMS untuk mendukung model pemerolehan boolean untuk operasi dasar AND, OR dan NOT. Operasi SQL dengan operasi relational algebra dicoba diterapkan pada ORDBMS untuk mendukung query seperti yang diharapkan saat menggunakan inverted index. Ujicoba dengan menggunakan corpus 5336 dokumen berita teknologi dalam eksperimen di laboratorium menghasilkan hampir 51262 term untuk penerapan inverted index ke dalam RDBMS. Implementasi operasi boolean dasar AND, OR atau NOT menunjukkan bahwa peningkatan jumlah operator boolean yang digunakan dari nol sampai enam membutuhkan waktu yang meningkat secara linier dengan tingkat korelasi di atas 0,99. Dengan spesifikasi sistem yang digunakan, untuk query dengan kata yang dimiliki sekitar 1 sampai 2 dokumen, waktu yang dibutuhkan untuk penggunaan satu operator sekitar 0,073detik sampai sekitar 0,203 detik untuk enam operator. Sedangkan untuk query dengan kata yang dimiliki sekitar 5000 dokumen, waktu yang dibutuhkan untuk penggunaan satu operator sekitar 0,094 detik sampai sekitar 0,474 detik untuk enam operator.

Kata Kunci: inverted index, sistem pemerolehan boolean, dbms, ordbms

1. PENDAHULUAN

1.1 Latar Belakang Masalah

Sistem pemerolehan informasi menawarkan kemampuan menyediakan informasi yang dibutuhkan pemakai. *Inverted index* yang diterapkan pada kebanyakan sistem pemerolehan informasi dan mesin pencari web terbukti sangat efisien untuk menjawab *query* (Baeza-Yates, 1999). Implementasi *inverted index* dapat diterapkan ke dalam *Database Management System* (DBMS) dengan 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. Operasi merubah maupun menghapus banyak dokumen yang mahal dalam sebuah *inverted index* dapat dilakukan lebih efisien dalam *Relational Database Management System* (RDBMS. Dalam sistem pemerolehan informasi klasik, dengan DBMS perbedaan dan duplikasi indeks untuk menjawab *query* dan indeks untuk memperbarui 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* (Papadakos, 2008).

Disamping 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 sebuah RDBMS akan direpresentasikan dalam tiga tuple [t,d_1], [t,d_3], [t,d_5] 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* (ORDMS) dengan penerapan menggunakan *collection.* Hal ini karena data ($t,\{d_1,d_3,d_5\}$) dapat disimpan dalam sebuah *tuple* dengan data $\{d_1,d_3,d_5\}$ diletakkan dalam sebuah *collection*.

Akses *document-based* dapat dilakukan dengan lebih cepat dalam sistem RDBMS. Namun *Inverted index* menawarkan akses *term-based* yang lebih efisien untuk melakukan perhitungan jawaban dari sebuah indeks (Papadakos, 2008). Kelebihan *document-based* ini tidak dapat dimanfaatkan saat

menggunakan ORDMS dengan penerapan menggunakan *collection* karena struktur tuple dalam ORDMS menggunakan (t,{d_1,d_3,d_5}).

Struktur data *inverted index* berupa pasangan *term* dan *posting list* (t,{d_1,d_3,d_5}) yang direpresentasikan dalam RDBMS akan menjadi tiga tuple [t,d_1], [t,d_3], [t,d_5]. Struktur ini menuntut digunakannya operasi SQL yang sesuai untuk memenuhi kebutuhan *query* saat menggunakan RDBMS (Papadakos, 2008). Struktur data (t,{d_1,d_3,d_5}) yang diterapkan menggunakan ORDBMS juga menuntut penggunaan operasi SQL yang berbeda untuk mengakses *collection* (Connoly, 2005).

Model pemerolehan *boolean* (*boolean retrieval*) merupakan model yang menggunakan struktur data *inverted index*. Model ini merupakan model utama yang disediakan oleh penyedia informasi besar selama tiga dekade sampai awal 1990. Tetapi sistem ini tidak hanya menggunakan operasi *boolean* dasar (AND, OR, dan NOT) (Manning, 2008).

Kemampuan ORDBMS memanfaatkan *collection* dapat menghasilkan implementasi *inverted index* untuk mendapatkan beberapa kelebihan saat menggunakan DBMS seperti disebutkan di atas.

1.2 Tujuan Penelitian

Penelitian ini bertujuan untuk mengimplementasikan dan mengamati unjuk kerja penggunaan konsep *relational algebra* untuk menjawab *query* dalam ORDBMS seperti yang diharapkan saat menggunakan *inverted index* untuk model pemerolehan *boolean* dengan operasi *boolean* dasar. Penelitian ini bermanfaat sebagai alternatif penerapan *inverted index* ke dalam ORDBMS untuk memperoleh kelebihan yang ditawarkannya.

2. LANDASAN TEORI

2.1 Collection dalam 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).

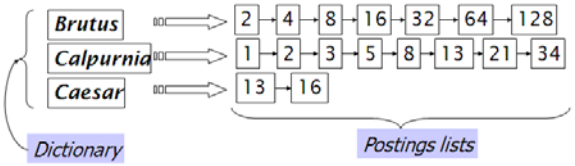
Multiset adalah sebuah tipe *collection* dengan elemen yang tidak terurut yang memungkinkan adanya duplikasi. Tidak seperti *array*, *multiset* tidak dibatasi oleh ukuran maksimum yang ditentukan di awal. *Operator* diperlukan untuk mengkonversi sebuah *multiset* ke sebuah tabel (Connoly, 2005).

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, 2010).

2.2 Relational Algebra untuk Inverted index

Representasi struktur data *inverted index* yang disajikan pada Gambar 1 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 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 1. Representasi *inverted index* (Manning, 2008)

Operasi model pemerolehan *boolean* dasar meliputi operasi AND, OR dan NOT. Untuk *inverted index* yang disajikan pada Gambar 1, 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 2. 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 2. Algoritma interseksi dari dua *posting list* p1 dan p2 (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 1, dapat memakai

operasi *intersection* (Connoly, 2005) menggunakan operasi dasar persamaan (1), operasi ini dapat diimplementasikan menggunakan *operator INTERSECT* pada operasi SQL.

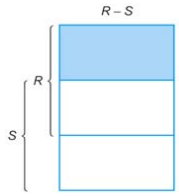
$$R \cap S \quad (1)$$

Dalam implementasi *operator* OR untuk representasi data dalam RDBMS dari *inverted index* seperti tersaji pada Gambar 1, operasi *boolean* dasar OR untuk model pemerolehan *boolean* dapat langsung diterapkan pada predikat dari perintah SQL. Namun untuk ORDBMS penerapan operasi boolean dasar OR melibatkan operasi *cartesian product* yang kurang efisien, sehingga operasi UNION yang disimbolkan pada persamaan (2) digunakan (Connoly, 2005).

$$R \cup S \quad (2)$$

Implementasi *operator* NOT untuk representasi data dalam RDBMS maupun *nested table* dari ORDBMS dari *inverted index* seperti tersaji pada Gambar 1, dapat dilakukan dengan menggunakan operasi *set difference relational algebra* seperti tersaji pada Gambar 3. Operasi *set difference* mendefinisikan suatu relasi yang berisi tuple dalam relasi R tetapi tidak di S. Operasi ini disimbolkan dengan persamaan (3) (Connoly, 2005). Implementasi dari operasi ini dapat menggunakan *operator* MINUS dari SQL.

$$R - S \quad (3)$$



Gambar 3. Representasi operasi *set difference* relational *algebra* (Connoly, 2005)

3. METODOLOGI

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

1. Studi pustaka penerapan konsep relational algebra untuk menjawab *query* dalam DBMS seperti yang diharapkan saat menggunakan *inverted index* untuk model pemerolehan *boolean* dengan operasi *boolean* dasar.
2. Pengumpulan dokumen-dokumen sebagai *corpus* diambil dari berita-berita Kompas Tekno (Kompas, 2010).
3. Implementasi penerapan konsep relational algebra yang telah dibahas dalam landasan teori menggunakan SQL ORDBMS untuk mendukung *inverted index*.
4. Pengamatan unjuk kerja waktu *query* dan jumlah hasil *query* sebagai implementasi pada langkah 3 untuk operasi AND, OR dan NOT dilakukan pada tiga kelompok kata berdasarkan jumlah dokumen yang memenuhi suatu kata atau *document frequencies (df)*. Ketiga kelompok kata ini adalah kelompok kata yang mempunyai *df* 1 sampai 2, *df* kurang lebih 2500 dan *df* kurang lebih 5000. Operasi AND, OR dan NOT dilakukan dengan menggunakan 1 sampai 7 *operand* kata atau dengan kata lain menggunakan 0 sampai 6 *operator*.

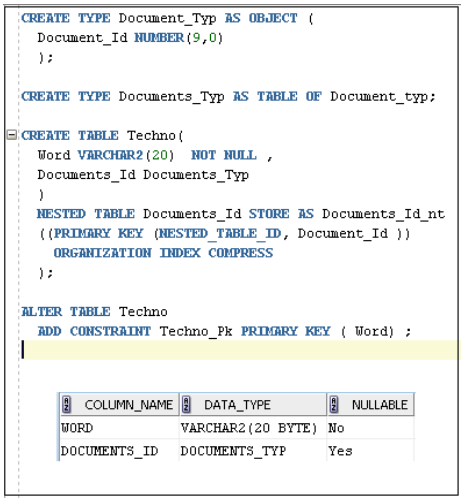
Penelitian ini dilakukan dengan menggunakan sebuah komputer dengan spesifikasi sebagai berikut:

a. Perangkat lunak
   1. Sistem operasi, Microsoft Windows XP SP2
   2. Oracle 10G Release 2
   3. Oracle SQL Developer (2.1.1.64)
   4. Java JDK 1.6.0 dan JDBC
   5. Netbeans 6.1
b. Perangkat keras
   1. Prosesor Intel Core 2 Quad 6600
   2. Memori RAM 2 GB/5300 DDR2
   3. Hardisk 160 GB SATA 2
   4. Motherboard chipset Intel DP35DP

4. HASIL DAN PEMBAHASAN

4.1 Data yang Digunakan

Struktur Tabel TECHNO yang digunakan pada percobaan ini disajikan pada Gambar 4. Tabel ini terdiri dari kolom WORD yang mewakili kata dan kolom DOCUMENTS_ID bertipe data *nested table* yang merepresentasikan *posting list* untuk menyimpan dokumen berupa id. Kolom WORD digunakan sebagai primary key pada tabel TECHNO yang sekaligus akan membangkitkan indeks. Sedangkan DOCUMENT_ID digunakan sebagai primary key pada nested table DOCUMENTS_ID.



```
CREATE TYPE Document_Typ AS OBJECT (
  Document_Id NUMBER(9,0)
  );

CREATE TYPE Documents_Typ AS TABLE OF Document_typ;

CREATE TABLE Techno(
  Word VARCHAR2(20)  NOT NULL ,
  Documents_Id Documents_Typ
  )
  NESTED TABLE Documents_Id STORE AS Documents_Id_nt
  ((PRIMARY KEY (NESTED_TABLE_ID, Document_Id ))
    ORGANIZATION INDEX COMPRESS
  );

ALTER TABLE Techno
  ADD CONSTRAINT Techno_Pk PRIMARY KEY ( Word) ;
```

COLUMN_NAME	DATA_TYPE	NULLABLE
WORD	VARCHAR2(20 BYTE)	No
DOCUMENTS_ID	DOCUMENTS_TYP	Yes

Gambar 4. Struktur Tabel TECHNO menggunakan Oracle SQL Developer

Dari 5336 dokumen Kompas Tekno, didapatkan 51262 kata yang berbeda dengan document frequency (df) antara 1 sampai 5336 dokumen yang merupakan hasil representasi *inverted index* ke dalam ORDBMS Tabel TECHNO. Panjang kata yang dihasilkan mempunyai jangkauan 2 sampai 19 huruf.

4.2 Hasil Percobaan

Untuk operasi AND dilakukan dengan operasi SQL *intersection* "(SELECT I.* FROM Techno T, Table(T.Documents_Id) I WHERE T.Word= 'term1') INTERSECT (SELECT I.* FROM Techno T, Table(T.Documents_Id) I WHERE T.Word='term2')". Pada operasi AND ini digunakan *operand* 'term1' dan 'term2'. Jumlah *operand* kata yang digunakan sampai sejumlah 7 *operand*. Hasil pengamatan waktu *query* yang diperoleh untuk operasi AND dengan jumlah *operand* satu sampai tujuh atau dengan kata lain jumlah *operator* boolean yang digunakan dari nol sampai enam untuk ketiga kelompok df disajikan pada Tabel 1. Representasi dalam bentuk grafik waktu *query* terhadap jumlah *operand* untuk *operator* AND untuk kelompok df kata 1-2, ±2500 dan ±5000 ini disajikan pada Gambar 5.

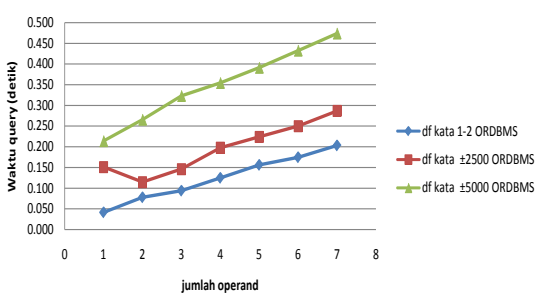
Dari hasil pengamatan waktu *query* pada Tabel 1, untuk kelompok df kata ±2500 saat jumlah operand satu atau tanpa operator AND terlihat waktu akses lebih lama dari pada saat jumlah operand dua, hal ini disebabkan karena terjadi perubahan hasil query yang menyolok dari 2547 untuk kondisi pertama menjadi kurang dari separuhnya sejumlah 1096. Karena jumlah hasil query ini juga berpengaruh pada waktu akses, maka dengan mengabaikan data saat jumlah operand satu untuk kelompok df kata ±2500 membuat korelasinya meningkat menjadi di atas 0,99.

Untuk ketiga kelompok *df* kata, penambahan jumlah *operand* untuk *operator* AND akan memperlama waktu akses secara linier dengan korelasi di atas 0,99. Peningkatan jumlah df kata dari 1-2, ±2500 sampai ±5000 yang digunakan sebagai *operand* untuk *operator* AND juga memperlama waktu akses seperti terlihat pada Gambar 5.

Untuk operasi OR dilakukan dengan operasi SQL *union* "(SELECT I.* FROM Techno T, Table(T.Documents_Id) I WHERE T.Word='term1') UNION (SELECT I.* FROM Techno T, Table(T.Documents_Id) I WHERE T.Word='term2')". Pada operasi OR ini digunakan *operand* 'term1' dan 'term2'. Jumlah *operand* kata yang digunakan sampai sejumlah 7 *operand*. Hasil pengamatan waktu *query* yang diperoleh untuk operasi OR dengan jumlah *operand* 1 sampai 7 untuk ketiga kelompok *df* disajikan pada Tabel 2. Representasi dalam bentuk grafik waktu *query* terhadap jumlah *operand* untuk *operator* OR untuk kelompok df kata 1-2, ±2500 dan ±5000 ini disajikan pada Gambar 6.

Tabel 1. Waktu *query* operasi AND dengan 1 sampai 7 *operand* kata untuk ketiga kelompok df kata

df kata	Jumlah *operand* untuk operasi AND	Rata- rata Waktu query	Hasil Query (dokumen)	Korelasi (r)
1-2	1	0,042	1	0,997324
	2	0,078	1	
	3	0,094	1	
	4	0,125	1	
	5	0,156	1	
	6	0,175	1	
	7	0,203	1	
±2500	1	0,151	2547	0,940363
	2	0,115	1096	
	3	0,146	631	Tanpa no 1
	4	0,198	379	
	5	0,224	188	0,994780
	6	0,250	181	
	7	0,287	181	
±5000	1	0,214	5336	0,996500
	2	0,266	5336	
	3	0,323	5336	
	4	0,354	5336	
	5	0,391	5336	
	6	0,432	5336	
	7	0,474	5336	



Gambar 5. Grafik waktu *query* terhadap jumlah *operand* untuk *operator* AND untuk kelompok df kata 1-2, ±2500 dan ±5000

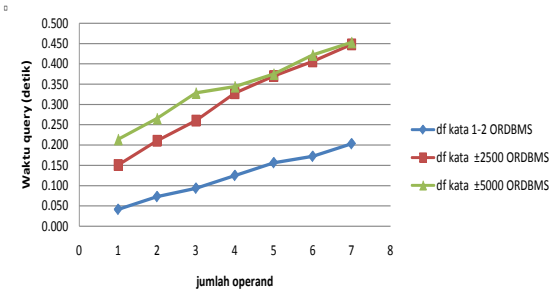
Dari hasil pengamatan waktu *query* pada Tabel 2, terlihat bahwa untuk kelompok kata ±2500 dan ±5000 saat jumlah operand 5, 6 dan 7 waktu aksesnya hampir sama hal ini disebabkan karena untuk ketiga kondisi mempunyai hasil query yang hampir sama yaitu 5133 dan 5336. Namun waktu akses untuk kelompok kata ±5000 secara umum lebih lama dari pada untuk kelompok kata ±2500.

Untuk ketiga kelompok *df* kata, penambahan jumlah *operand* untuk *operator* OR akan memperlama waktu akses secara linier dengan korelasi > 0,99 untuk ketiga kelompok *df* kata. Peningkatan jumlah df kata dari 1-2, ±2500 sampai ±5000 yang digunakan sebagai *operand* untuk *operator* OR juga memperlama waktu akses seperti terlihat pada Gambar 6.

 ISSN: 1907-5022

Tabel 2. Waktu *query* operasi OR dengan 1 sampai 7 *operand* kata untuk ketiga kelompok df kata

df kata	Jumlah *operand* untuk operasi OR	Rata- rata Waktu query	Hasil query (dokumen)	Korelasi (r)
1-2	1	0,042	1	0,997857
	2	0,073	1	
	3	0,094	1	
	4	0,125	1	
	5	0,156	1	
	6	0,172	1	
	7	0,203	2	
±2500	1	0,151	2547	0,995256
	2	0,211	3691	
	3	0,260	4433	
	4	0,328	4772	
	5	0,370	5133	
	6	0,406	5133	
	7	0,448	5133	
±5000	1	0,214	5336	0,990152
	2	0,265	5336	
	3	0,328	5336	
	4	0,344	5336	
	5	0,375	5336	
	6	0,422	5336	
	7	0,453	5336	



Gambar 6. Grafik waktu query terhadap jumlah *operand* untuk *operator* OR untuk kelompok df kata 1-2, ±2500 dan ±5000

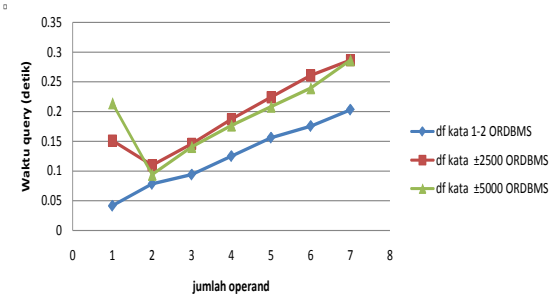
Untuk operasi NOT dilakukan dengan operasi SQL *minus* "(SELECT I.* FROM Techno T, Table(T.Documents_Id) I WHERE T.Word='term1') MINUS (SELECT I.* FROM Techno T, Table(T.Documents_Id) I WHERE T.Word=' term2')". Pada operasi NOT ini digunakan *operand* 'term1' dan 'term2'. Jumlah *operand* kata yang digunakan sampai sejumlah 7 *operand*. Hasil pengamatan waktu *query* yang diperoleh untuk operasi NOT dengan jumlah *operand* satu sampai tujuh untuk ketiga kelompok *df* disajikan pada Tabel 3. Representasi dalam bentuk grafik waktu *query* terhadap jumlah *operand* untuk *operator* NOT untuk kelompok df kata 1-2, ±2500 dan ±5000 ini disajikan pada Gambar 7.

Dari hasil pengamatan waktu *query* pada Tabel 3, untuk kelompok df kata ±2500 saat jumlah operand satu atau tanpa operator NOT terlihat waktu akses lebih lama dari pada saat jumlah operand dua, hal ini disebabkan karena terjadi perubahan hasil query yang menyolok dari 2547 untuk jumlah operand satu menjadi kurang dari separuhnya sejumlah 1451. Perubahan menyolok ini juga terjadi untuk kelompok df kata ±5000 dari hasil query 5336 untuk jumlah operand satu menjadi hasil query nol saat jumlah operand dua sampai tujuh. Dengan mengabaikan data saat jumlah operand satu untuk kelompok df kata ±2500 dan ±5000 membuat korelasinya meningkat menjadi di atas 0,99.

Dari Gambar 7 terlihat bahwa untuk kelompok df kata ±5000 waktu aksesnya hampir sama atau lebih baik dibandingkan untuk kelompok df kata ±2500. Hal ini disebabkan karena jumlah hasil query nol untuk kelompok df kata ±5000 sangat menyolok dengan jumlah hasil query untuk kelompok df kata ±2500 saat jumlah operand dua sampai tujuh.

Tabel 3. Waktu *query* operasi NOT dengan 1 sampai 7 *operand* kata untuk ketiga kelompok df kata

df kata	Jumlah *operand* untuk operasi NOT	Rata- rata Waktu query	Hasil query (dokumen)	Korelasi (r)
1-2	1	0,042	1	0,997370
	2	0,078	0	
	3	0,094	0	
	4	0,125	0	
	5	0,156	0	
	6	0,176	0	
	7	0,203	0	
±2500	1	0,151	2547	0,935918
	2	0,110	1451	
	3	0,146	849	Tanpa no 1: 0,998058
	4	0,187	479	
	5	0,224	297	
	6	0,261	297	
	7	0,286	297	
±5000	1	0,214	5336	0,697050
	2	0,094	0	
	3	0,141	0	Tanpa no 1: 0,997389
	4	0,177	0	
	5	0,208	0	
	6	0,239	0	
	7	0,286	0	



Gambar 7. Grafik waktu query terhadap jumlah *operand* untuk *operator* NOT untuk kelompok df kata 1-2, ±2500 dan ±5000

Secara keseluruhan untuk ketiga kelompok *df* kata, penambahan jumlah *operand* untuk *operator* NOT akan memperlama waktu akses secara linier dengan korelasi > 0,99 untuk ketiga kelompok *df*

kata. Peningkatan jumlah df kata dari 1-2 sampai ±5000 yang digunakan sebagai *operand* untuk *operator* NOT juga memperlama waktu akses seperti terlihat pada Gambar 7, kecuali untuk kelompok *df* kata ±2500 kelihatan waktu aksesnya lebih lama dari *df* kata ±5000 namun hal ini karena disebabkan perubahan jumlah hasil query yang menyolok menjadi nol untuk *df* kata ±5000 seperti yang disebutkan di atas.

Dengan spesifikasi sistem yang digunakan, untuk *query* dengan kata yang dimiliki sekitar 1 sampai 2 dokumen (*df kata*), waktu yang dibutuhkan untuk penggunaan satu *operator* sekitar 0,073 detik (Tabel 2) sampai sekitar 0,203 detik untuk enam *operator*. Sedangkan untuk *query* dengan kata yang dimiliki sekitar 5000 dokumen (*df kata*), waktu yang dibutuhkan untuk penggunaan satu *operator* sekitar 0,094 detik (Tabel 3) sampai sekitar 0,474 detik (Tabel 1) untuk enam *operator*.

5. PENUTUP

5.1 Kesimpulan

Penerapan *inverted index* ke dalam ORDBMS dengan kelebihan yang ditawarkan untuk mendukung model pemerolehan boolean dengan operasi dasar AND, OR dan NOT menunjukkan hasil di bawah ini.

Peningkatan jumlah *operator* boolean yang digunakan dari nol sampai enam *operator* membutuhkan waktu yang meningkat secara linier dengan tingkat korelasi di atas 99%.

Menggunakan spesifikasi sistem yang digunakan dengan ujicoba menggunakan corpus 5336 dokumen berita teknologi yang menghasilkan hampir 51262 *term* untuk penerapan *inverted index* ke dalam ORDBMS dengan *nested table collection*, untuk *query* dengan kata yang dimiliki sekitar 1 sampai 2 dokumen, waktu yang dibutuhkan untuk penggunaan satu *operator* sekitar 0,073 detik sampai sekitar 0,203 detik untuk enam *operator*. Sedangkan untuk *query* dengan kata yang dimiliki sekitar 5000 dokumen, waktu yang dibutuhkan untuk penggunaan satu *operator* sekitar 0,094 detik sampai sekitar 0.474 detik untuk enam *operator*.

Penerapan *inverted index* ke dalam ORDBMS menjadi salah satu alternatif penerapan *inverted index* yang dapat digunakan pada sistem yang sesuai dengan kebutuhan untuk mendapatkan kelebihan yang ditawarkan.

5.2 Penelitian Selanjutnya

Penelitan selanjutnya dapat dilakukan untuk membandingkan penerapan *inverted index* menggunakan teknologi ORDBMS dengan inverted index menggunakan struktur data klasik.

PUSTAKA

Baeza-Yates, R., dan Ribeiro-Neto, Berthier. (1999). *Modern Information Retrieval*. Addison Wesley.

Connoly, T., dan Begg, C. (2005). *Database Systems: A Practical Approach to Design, Implementation, and Management*. Pearson Education Limited, England.

Kompas. (2010). *Kompas Tekno*. Diakses terakhir pada 15 Februari 2010 dari http://tekno.kompas.com.

Manning, C.D., Raghavan, P., Schutze, H. (2008). *Introduction to Information Retrieval* , Cambridge University Press.

Oracle. (2010). *Oracle Documentation*. Oracle Corporation. Diakses terakhir pada 15 Februari 2010 dari http://www.oracle.com.

Papadakos, P., Theoharis, Y., Marketakis, Y., Armenatzoglou, N., Tzitzikas, Y. (2008). *Mitos: Design and Evaluation of a DBMS-based Web Search Engine*. IEEE.