ISSN: 2614-6754 (print)
ISSN: 2614-3097(online)
Halaman 2297-2306
Volume 6 Nomor 1 Tahun 2022

Optimasi Database Menggunakan SQL Kueri Klausa IN dan EXIST pada Database Oracle 12c. Studi Kasus pada Aplikasi di Badan Nasional Pencarian dan Pertolongan (BASARNAS)

Samidi[1], Dika Andharu[2], Fajar Widyarto[3], Sutan Syahdinullah S[4], Samuel Yusach Eka W[5]
[1,2,3,4,5] Universitas Budi Luhur, Magister Ilmu Komputer, Jakarta Selatan
e-mail: samidi@budiluhur.ac.id[1], 2111600330@student.budiluhur.ac.id[2], 2111600652@student.budiluhur.ac.id[3], 2111600140@student.budiluhur.ac.id[4], 2111600405@student.budiluhur.ac.id[5]

Abstrak

Aplikasi ini merupakan sistem yang mengakomodir kegiatan pengumpulan Daftar Usulan Penetapan Angka Kredit (DUPAK) bagi para rescuer Badan Nasional Pencarian dan Pertolongan atau dikenal dengan BASARNAS. Sistem ini sudah berjalan beberapa tahun ke belakang dan memiliki banyak data sehingga performa dalam pengambilan atau pencarian data tentunya semakin berkurang. Dengan adanya permasalahan tersebut maka perlu adanya penelitian yang mendukung dalam meningkatkan performa dalam pengambilan atau pencarian data, salah satunya yaitu optimasi basis data. Metode optimasi basis data yang difokuskan pada penelitian ini dengan melakukan studi bunding dari berbagai efektivitas subkueri serta penggunaan indexing pada tabel. Dalam pengujiannya, metode subkueri klausa IN memiliki hasil rata – rata dari 3,114 detik sebelum dilakukan pengindeksan menjadi 2,972 detik setelah dilakukan pengindeksan dan metode subkueri klausa EXISTS memiliki hasil rata – rata dari 3,087 detik sebelum dilakukan pengindeksan menjadi 2,935 detik setelah dilakukan pengindeksan.

Kata kunci: *Optimasi Database, Oracle 12c, Subkueri, Pengindeksan*

Abstract

This application is a system that accommodates the collection of the Proposed List of Credit Scores (DUPAK) for rescuers from the National Search and Rescue Agency or known as BASARNAS. This system has been running for several years and has a lot of data so that the performance in data retrieval or search is indeed decreasing. With these problems, it is necessary to have research that supports improving performance in data retrieval or search, one of which is database optimization. The database optimization method focused on in this research is by conducting a bundling study of the effectiveness of various subqueries and the use of indexing in tables. In the test, the IN clause subquery method has an average result from 3,114 seconds before indexing is done to 2,972 seconds after indexing and the EXISTS clause subquery method has a moderate impact from 3,087 seconds before indexing is done to 2,935 seconds after indexing.

Keywords : *Database Optimization, Oracle 12c, Subqueries, Indexing*

PENDAHULUAN

Hampir semua aplikasi bisnis khususnya instansi pemerintahan membutuhkan basis data untuk tujuan integrasi data juga sebagai distribusi data antar sistem. Database Management System (DBMS) sebagai perangkat lunak yang mengelola data harus dikelola dan dioptimalkan untuk kinerja yang lebih baik. Sistem ini harus cepat menanggapi segala jenis ancaman yang mungkin terjadi. Performa sistem basis data dapat dipengaruhi dari

beberapa faktor yaitu: ukuran basis data yang tumbuh sebanding dengan data, peningkatan pengguna basis data, peningkatan proses pengguna, DBMS yang tidak disetel dengan benar dan tidak disetel. Semua faktor ini menurunkan waktu respons sistem yang akan mengantisipasi penurunan kinerja [1].

Salah satu masalah kritis utama yang sering terjadi di perusahaan khususnya instansi pemerintahan adalah kinerja kueri yang tidak memadai untuk melakukan output yang sesuai. Banyak faktor yang menjadi penyebab terjadinya ini, salah satunya adalah query masalah pemrosesan. Sejak itu, signifikan sejumlah penelitian dan observasi telah dilakukan untuk menemukan solusi yang efisien untuk memproses kueri. Sebuah query mungkin mahal dalam hal biaya eksekusi jika tidak dioptimalkan dengan baik [2]. Untuk waktu yang lama disini bisa berdampak negatif bagi instansi pemerintahan karena menurunnya kinerja performa. Deteksi penurunan kinerja akan terdeteksi oleh sistem pemantauan terus menerus dan berdampak pada parameter kinerja.

Dalam sistem basis data struktur generasi pertama, bahasa kueri prosedural tingkat rendah umumnya tertanam dalam bahasa pemrograman tingkat tinggi dan programmer harus memilih strategi eksekusi yang paling tepat. Berbeda dengan bahasa deklaratif seperti SQL, pengguna menentukan data apa yang diperlukan daripada bagaimana itu untuk diambil [3]. Pola ini mengubah pengguna tanggung jawab untuk menentukan, atau bahkan mengetahui apa metode untuk mendukung strategi eksekusi yang baik. Itu tujuan yang paling penting untuk dipertimbangkan dalam rangka untuk meningkatkan kinerja DBMS adalah: merancang skema data yang efisien, mengoptimalkan indeks, menganalisis rencana eksekusi, pemantauan akses ke data, dan mengoptimalkan kueri [4]. Untuk penelitian ini yang berfokus pada pengoptimalan kueri di Oracle Database 12c.

Pada penelitian ini akan membahas lebih lanjut terkait dengan aplikasi pada salah satu instansi pemerintahan yaitu Daftar Usulan Penetapan Angka Kredit (DUPAK) rescuer milik Badan Nasional Pencarian dan Pertolongan (BASARNAS). Proses bisnis utama dari aplikasi ini adalah untuk mendukung proses pengumpulan DUPAK para rescuer BASARNAS yang nantinya akan dicek oleh verifikator dari Direktorat Bina Tenaga. Pada penggunaannya beberapa tahun ke belakang, data yang disimpan cukup banyak dan data akan terus bertambah seiring dengan tuntutan pekerjaan dari para rescuer maupun penambahan personil rescuer itu sendiri. Dari sisi performa pun untuk saat ini agak menurun dibandingkan dengan awal mula aplikasi terbentuk. Maka dari itu, dengan adanya penambahan data yang diprediksi akan semakin banyak sehingga perlu adanya penelitian terkait dengan evaluasi performa query basis data.

Pada penelitian terdahulu, terdapat beberapa referensi jurnal yang fokus membahas komparasi penggunaan indexing dan subkueri, diantaranya adalah jurnal yang ditulis oleh Marlene, M., et al pada tahun 2014 [5]. Fokus penelitian ini adalah komparasi pada sebuah tabel yang belum memiliki index dan tabel yang sudah memiliki index dengan alat bantu (*tools*) *Database Engine Tuning Advisor* pada SQL Server 2008. Pengujian yang dilakukan pada penelitian ini sebanyak 10 (sepuluh) kali dengan hasil peningkatkan performa sebesar 11.6 % dari tabel yang belum memiliki index dengan tabel yang sudah dibuatkan index. Penelitian selanjutnya yaitu Indrajani pada tahun 2015 [6] yang membahas tentang tuning basis data. Adapun lingkup pada penelitiannya ini mulai dari restrukturisasi tabel, penggunaan ROWID, perubahan penggunaan data *type convertion, partitioning, dan indexing*. Dalam lingkup jurnal tersebut, hanya ada satu referensi yang terkait dengan penelitian ini yaitu indexing. Hasil dari penelitian ini secara keseluruhan lingkup penelitiannya cukup signifikan karena berawal butuh waktu minimal 4 jam untuk memproses 5.772.319 data dan setelah di tuning hanya membutuhkan waktu kurang lebih 5 menit. Referensi penelitian terakhir disusun oleh Oktavia, T pada tahun 2014 [7] yang membahas tentang perbandingan penggunaan index dan non-index. Selain itu, juga ditambahkan kasus komparasi dengan metode dari beberapa subkueri diantaranya IN, EXISTS, operasional (=) dan operasional (=) + TOP. Pengujian pada penelitian ini terdiri dari 10 (sepuluh) kali percobaan dan didapatkan hasil bahwa metode klausa IN dan EXISTS tidak memiliki perbedaan yang signifikan mulai yang belum diberikan index sampai dengan pengindeksan.

Selain itu, subkueri dengan operasional tidak direkomendasikan jika belum dilakukan pengindeksan karena akan memakan waktu yg lebih banyak. Namun, jika dengan pengindeksan maka hasilnya berbanding terbalik dan lebih direkomendasikan dibandingkan metode klausa IN dan EXISTS.

Beberapa jurnal diatas dapat dijadikan tolak ukur dari penelitian untuk memastikan apakah benar atau tidak, sehingga penulis disini akan melakukan pengujian khususnya dengan menggunakan metode subkueri klausa dan pengindeksan pada tabel.

Pada penelitian terdahulu telah dibandingkan hasil kinerja SQL basis data Microsoft SQL Server 2008 antara klausa IN dan klausa EXIST dalam table yang tidak dilakukan INDEX serta yang telah dilakukan INDEX dimana hasil nya adalah peningkatan sebesar 36,9% saat dilakukan pengindeksan pada klausa IN dan peningkatan sebesar 40,3 % saat dilakukan pengindeksan pada klausa EXISTS.

Adapun perbedaan pada penelitian ini dibandingkan dengan sebelumnya adalah data yang digunakan adalah data dari aplikasi DUPAK *Rescuer* BASARNAS dimana basis data yang digunakan yaitu Oracle 12c dan pengujiannya menggunakan metode subkueri klausa IN dan EXISTS serta adanya pengindeksan pada tabel sehingga penelitian ini bertujuan untuk memastikan apakah terdapat hasil yang berbeda dengan penelitian terdahulu.

METODE PENELITIAN

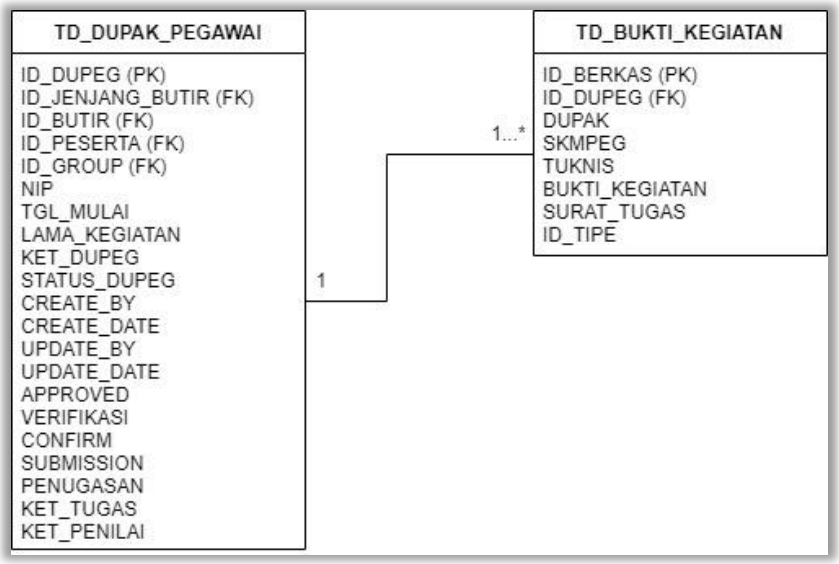
Secara umum, penelitian ini menguji untuk membandingkan total waktu eksekusi pada tabel yang memiliki index maupun tidak (non-index) dengan metode subkueri klausa IN dan EXISTS. Semua pemrosesan ini didukung oleh salah satu mesin pemroses SQL berbasis DBMS yaitu Oracle 12c. Adapun dalam mengakses basis data disini, penulis menggunakan GUI *database tools*-nya yaitu Navicat Premium Essentials 15. Penelitian ini merupakan proses iteratif untuk mengukur dampak metode dari masing-masing subkueri dan efektivitas dari penggunaan index pada tabel. Tentunya ini menentukan kinerja tetapi tergantung pada jumlah data maupun aktivitas pengguna yang mengakses aplikasi. Sehingga hal ini bisa dinilai masih banyak faktor yang harus dipertimbangkan untuk meningkatkan kinerja optimal kueri yang terhubung dengan aplikasi [8]. Performa yang mumpuni bisa dicapai dengan membangun kode yang efisien pada bahasa pemrograman dan desain basis data yang sesuai. Namun, tidak menutup kemungkinan bahwa indikator lain yang memengaruhi performa adalah penggunaan spesifikasi perangkat keras yang superior. Penulis melakukan langkah penelitian mulai dari duplikasi tabel sebanyak dua dimana masing-masing diberikan tanda penamaan tabel paling belakang dengan “INDEX” dan “NONINDEX”. Hal ini dilakukan agar tabel yang sedang beroperasi saat ini tidak terganggu dengan adanya pengujian ini. Selanjutnya, penulis melakukan identifikasi struktur basis data pada tabel, lalu mulai membuat kueri pengujian pada tabel transaksi dengan data yang paling banyak dan tentunya tabel ini akan dijadikan bahan pengujiannya. Kueri yang digunakan disini adalah join antara dua tabel dengan metode subkueri klausa IN dan EXISTS. Pada kesempatan berikutnya, penulis membuat index pada tabel dengan penamaan paling belakang “INDEX” agar dapat menjadi pembanding dengan tabel tanpa index yang penamaannya paling belakang “NONINDEX”. Saat semua persiapan telah dilaksanakan, penulis melaksanakan pengujian kueri yang telah dibuat dan mulai merekap berapa lama eksekusi yang dibutuhkan bagi subkueri IN dan EXISTS pada tabel dengan pengindeksan dan tidak dengan pengindeksan. Terakhir, penulis melakukan survey kepuasan bagi pengguna tertentu terhadap pengujian ini.

HASIL DAN PEMBAHASAN

Pada penelitian ini, penulis mengambil basis data dari aplikasi DUPAK BASARNAS dimana terdapat data pengajuan DUPAK yang diajukan para rescuer untuk dinilaikan oleh verifikator dari Direktorat Bina Tenaga. Adapun untuk pengujiannya, penulis menggunakan dua tabel utama dengan transaksi data paling banyak yaitu tabel TD_DUPAK_PEGAWAI dan TD_BUKTI_KEGIATAN. Berikut ini adalah gambaran struktur dari kedua tabel tersebut.

ISSN: 2614-6754 (print)
ISSN: 2614-3097(online)

Halaman 2297-2306
Volume 6 Nomor 1 Tahun 2022



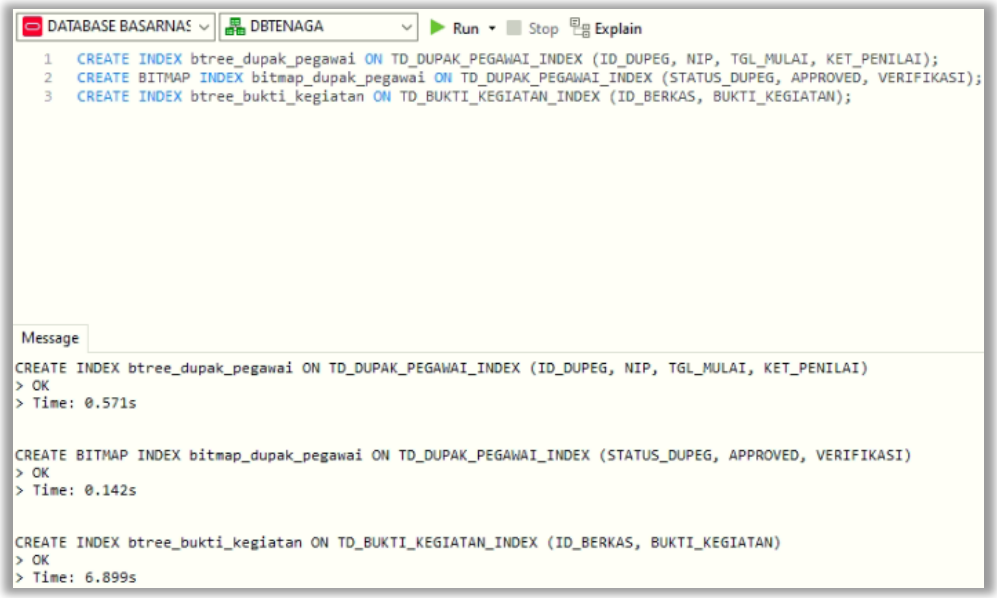
Gambar 1. Struktur Tabel

Tabel TD_DUPAK_PEGAWAI berisikan data pengajuan DUPAK bagi para *rescuer* dan untuk tabel TD_BUKTI_KEGIATAN merupakan tabel yang berisikan lampiran bukti kegiatan berupa nama file lampiran foto. Berdasarkan gambar diatas bisa dijelaskan bahwa 1 data pengajuan DUPAK bisa banyak data nama file lampiran foto.

Berdasarkan struktur tabel diatas, penulis melakukan duplikasi kedua tabel sebanyak dua rangkap dengan penamaan yang ditambahkan dibelakangnya dengan “INDEX” dan “NONINDEX”. Berikut hasil duplikasi dari tabelnya:

a. TD_DUPAK_PEGAWAI_INDEX
b. TD_BUKTI_KEGIATAN_INDEX
c. TD_DUPAK_PEGAWAI_NONINDEX
d. TD_BUKTI_KEGIATAN_NONINDEX

Dari keempat tabel tersebut, belum dilakukan pengindeksan, sehingga khusus untuk tabel TD_DUPAK_PEGAWAI_INDEX dan TD_BUKTI_KEGIATAN_INDEX akan dibuatkan indeksnya terlebih dahulu. Berikut kueri untuk membuatkan index pada kedua tabel tersebut.



Gambar 2. Kueri Pembuatan Index

Tabel TD_BUKTI_KEGIATAN_INDEX hanya menggunakan b-tree indexing sedangkan tabel TD_DUPAK_PEGAWAI_INDEX menggunakan b-tree indexing dan bitmap indexing. Beberapa sumber jurnal menjelaskan bahwa adanya perbedaan diantara pengindeksan dengan menggunakan b-tree dan juga bitmap. Berikut dibawah ini adalah perbedaan menurut salah satu jurnal.

	Bitmap Index	B- tree index
1.	Bitmap indexes are typically only a fraction of the size of the indexed data in the table.	Fully indexing a large table with a traditional B-tree index can be prohibitively expensive in terms of space because the indexes can be several times larger than the data in the table.
2.	Bitmap indexes store the bitmaps in a compressed way(effective for low-cardinality data). If the number of distinct key values is small, bitmap indexes compress better and the space saving benefit compared to a B-tree index becomes even better.	B-tree indexes are most effective for high-cardinality data: that is, for data with many possible values, such as *customer_name* or *phone_number*.
3.	In many cases, it may not be necessary to index these columns in a data warehouse, because unique constraints can be maintained without an index, and because typical data warehouse queries may not work better with such indexes.In general, bitmap indexes should be more common than B-tree indexes in most data warehouse environments.	.B-tree indexes are most commonly used in a data warehouse to index unique or near-unique keys B-tree indexes are more common in environments using third normal form schemas.
4.	Bitmap indexes on partitioned tables are always local.	B-tree indexes on partitioned tables can be global or local.
5.	Implements the OLAP (On-Line Analytical Processing).	Implements the OLTP (On-Line Analytical Processing).

Gambar 3. Perbedaan Bitmap Index dan B-Tree Index [9]

Menurut Hamad, M., M. (2008) secara umum menjelaskan bahwa bitmap indexes baik digunakan untuk kolom dengan perbandingan dari banyaknya nilai dengan jumlah baris dalam tabel yang kecil [9]. Sedangkan b-tree indexes baik digunakan untuk kolom dengan banyak jenis nilai yang beragam. Sehingga dapat disimpulkan bahwa pengindeksan dengan bitmap indexes baik digunakan pada kolom dengan ragam nilai data unik yang sedikit (low cardinality) seperti data jenis kelamin (gender) yang memiliki ragam nilai hanya sedikit, sedangkan b-tree indexes baik digunakan pada kolom dengan ragam nilai data unik yang beragam atau banyak (high cardinality) dan tidak terbatas seperti nama lengkap (full name), alamat (address), dan lainnya.

Setelah dilaksanakannya proses pembuatan index pada tabel, maka selanjutnya membuat kueri dengan informasi kolom tertentu dalam rentan waktu 7 (tujuh) hari. Berikut ini adalah hasil pembuatan kuerinya.

a. Kueri dengan Metode Klausa IN Tanpa Pengindeksan (Non-Index)

```
select a.ID_DUPEG, a.NIP, a.TGL_MULAI,a.STATUS_DUPEG,
a.APPROVED, a.VERIFIKASI,  a.KET_PENILAI, b.ID_BERKAS, b.BUKTI_KEGIATAN
from TD_DUPAK_PEGAWAI_NONINDEX a
inner join "TD_BUKTI_KEGIATAN_NONINDEX" b on a.ID_DUPEG= b.ID_DUPEG
where a.ID_DUPEG in (
select ID_DUPEG from TD_DUPAK_PEGAWAI_NONINDEX
where TGL_MULAI between to_date('2020-12-01 00:00:00','YYYY-MM-DD HH24:MI:SS')
and to_date('2020-12-07 23:59:59','YYYY-MM-DD HH24:MI:SS'))
```

b. Kueri dengan Metode Klausa EXIST Tanpa Pengindeksan (Non-Index)

```
select a.ID_DUPEG, a.NIP, a.TGL_MULAI,a.STATUS_DUPEG,
a.APPROVED, a.VERIFIKASI,  a.KET_PENILAI, b.ID_BERKAS, b.BUKTI_KEGIATAN
from TD_DUPAK_PEGAWAI_NONINDEX a
inner join "TD_BUKTI_KEGIATAN_NONINDEX" b on a.ID_DUPEG= b.ID_DUPEG
```

```
where exists (
select ID_DUPEG from TD_DUPAK_PEGAWAI_NONINDEX
where TGL_MULAI between to_date('2020-12-01 00:00:00','YYYY-MM-DD HH24:MI:SS')
and to_date('2020-12-07 23:59:59','YYYY-MM-DD HH24:MI:SS')
and ID_DUPEG = a.ID_DUPEG)
```

c. Kueri dengan Metode Klausa IN dan Pengindeksan (Index)

```
select a.ID_DUPEG, a.NIP, a.TGL_MULAI,a.STATUS_DUPEG,
a.APPROVED, a.VERIFIKASI,  a.KET_PENILAI, b.ID_BERKAS, b.BUKTI_KEGIATAN
from TD_DUPAK_PEGAWAI_INDEX a
inner join "TD_BUKTI_KEGIATAN_INDEX" b on a.ID_DUPEG= b.ID_DUPEG
where a.ID_DUPEG in (
select ID_DUPEG from TD_DUPAK_PEGAWAI_INDEX
where TGL_MULAI between to_date('2020-12-01 00:00:00','YYYY-MM-DD HH24:MI:SS')
and to_date('2020-12-07 23:59:59','YYYY-MM-DD HH24:MI:SS'))
```

d. Kueri dengan Metode Klausa EXIST dan Pengindeksan (Index)

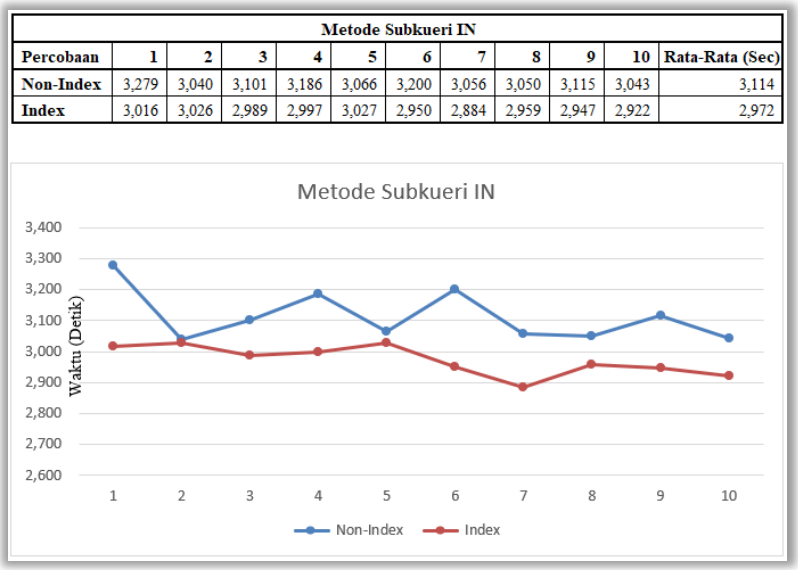
```
select a.ID_DUPEG, a.NIP, a.TGL_MULAI,a.STATUS_DUPEG,
a.APPROVED, a.VERIFIKASI,  a.KET_PENILAI, b.ID_BERKAS, b.BUKTI_KEGIATAN
from TD_DUPAK_PEGAWAI_INDEX a
inner join "TD_BUKTI_KEGIATAN_INDEX" b on a.ID_DUPEG= b.ID_DUPEG
where exists (
select ID_DUPEG from TD_DUPAK_PEGAWAI_INDEX
where TGL_MULAI between to_date('2020-12-01 00:00:00','YYYY-MM-DD HH24:MI:SS')
    and to_date('2020-12-07 23:59:59','YYYY-MM-DD HH24:MI:SS')
    and ID_DUPEG = a.ID_DUPEG)
```

Dari keempat kueri diatas, dapat digambarkan bahwa terdapat informasi yang terdiri dari data ID_DUPEG, NIP, TGL_MULAI, STATUS_DUPEG, APPROVED, VERIFIKASI, KET_PENILAI, ID_BERKAS, dan BUKTI_KEGIATAN dimana ada penambahan klausa where TGL_MULAI dengan rentang waktu 7 (tujuh) hari yang dimaksudkan untuk membatasi bahwa data yang ditampilkan hanya sebagian data (tidak keseluruhan data).

Selanjutnya, penulis melakukan pengujian pada keempat kueri diatas dengan masing-masing percobaan sebanyak 10 (sepuluh) kali. Berikut ini adalah hasil dari pengujian yang telah dilakukan pada rentang waktu mulai tanggal 1 Desember 2020 – 7 Desember 2020 dengan total data sebanyak 84.333 record.

a. Perbandingan Metode Subkueri Klausa IN dengan Index dan Non-Index

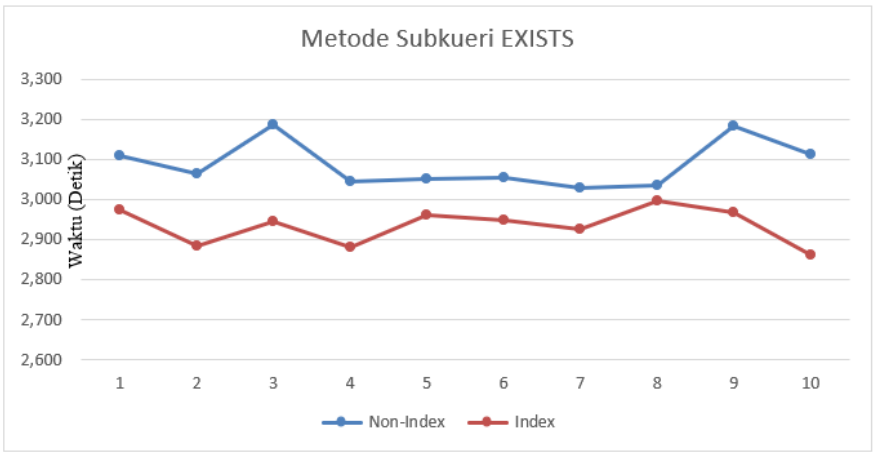
Metode Subkueri IN											
Percobaan	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Rata-Rata (Sec)**
Non-Index	3,279	3,040	3,101	3,186	3,066	3,200	3,056	3,050	3,115	3,043	3,114
Index	3,016	3,026	2,989	2,997	3,027	2,950	2,884	2,959	2,947	2,922	2,972



Gambar 4. Perbandingan Metode Subkueri Klausa IN dengan Index dan Non-Index

b. Perbandingan Metode Subkueri Klausa EXISTS dengan Index dan Non-Index

Metode Subkueri EXISTS											
Percobaan	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Rata-Rata (Sec)**
Non-Index	3,109	3,065	3,188	3,044	3,050	3,054	3,029	3,034	3,183	3,112	3,087
Index	2,974	2,884	2,945	2,880	2,961	2,950	2,927	2,997	2,968	2,860	2,935



Gambar 5. Perbandingan Metode Subkueri Klausa EXISTS dengan Index dan Non-Index

No	Peneliti	Objek	Metode	Hasil	Kesimpulan
1	Marlene, M., et al (2014)	Data AdventureWorks dengan database Microsoft SQL Server 2008	Metode pengindeksan pada tabel dan penempatan tabel pada file fisik yang berbeda maupun yang sama	Pengujian file fisik yang berbeda lebih cepat dengan file fisik yang sama dengan peningkatan sebesar 3,7 % dimana belum dilakukan pengindeksan. Setelah dilakukan pengindeksan, pengujian file fisik yang berbeda lebih cepat dengan file fisik yang sama dengan peningkatan sebesar 11,6 %	Pengindeksan lebih cepat dibandingkan tanpa pengindeksan. Selain itu, juga peningkatan terjadi lebih signifikan terhadap file fisik yang berbeda jika dibanding dengan file fisik yang sama.
2	Indrajani (2015)	Data bisnis perbankan	Metode mulai dari mulai dari restrukturisasi tabel, penggunaan	Pengujian pembentukan data mart sebelum dilakukan pengindeksan	Hasil dilakukannya pengindeksan jauh lebih cepat dengan hasil yang sangat

			ROWID, perubahan penggunaan data type convertion, partitioning, dan indexing.	dibutuhkan waktu mulai 4.58 jam sampai dengan 8.94 jam. Setalah dilakukan pengindeksan, diperoleh hasil yang sangat signifikan mulai dari 4 menit 40 detik sampai dengan 6 menit 20 detik	signifikan karena dari yang berawal membutuhkan minimal 4 jam menjadi 6 menit paling lama.
3	Oktavia, T (2014)	Data presensi Universitas Binus dengan database Microsoft SQL Server 2008	Pengujian dengan metode subkueri klausa IN, EXISTS, operasional (=) dan operasional (=) ditambah TOP	Pengujian dengan metode subkueri klausa IN didapatkan hasil 3003 detik sebelum *indexing* dan 1893,5 detik setelah *indexing*, subkueri klausa EXISTS dengan hasil 3147,3 detik sebelum *indexing* dan 1876,3 detik setelah *indexing*, subkueri klausa TOP dengan hasil 3505,9 detik sebelum *indexing* dan 26.3 detik setelah *indexing*, dan subkueri klausa EQUAL dengan hasil 7602,2 detik sebelum *indexing* dan	Pengujian pada semua metode subkueri klausa mengalami peningkatan setelah dilakukan pengindeksan. Adapun peningkatannya sebesar 36,9% klausa IN, 40,3% klausa EXISTS, 99,2% klausa TOP, dan 98,9% klausa Equal.

				82,9 detik sesudah *indexing*.	
4	Andharu, D., & Widyarto, F. (2022)	Data DUPAK rescuer BASARNAS dengan database Oracle 12c	Pengujian dengan metode subkueri klausa IN dan EXISTS	Pengujian dengan metode subkueri klausa IN didapatkan hasil 3,114 detik sebelum pengindeksan dan 2,972 detik setelah pengindeksan. Selain itu, untuk metode subkueri klausa EXISTS didapatkan hasil 3,087 detik sebelum pengindeksan dan 2,935 detik setelah pengindeksan.	Pengujian pada kedua metode tidak terlalu signifikan, adapun peningkatannya sebesar 0,045% setelah dilakukan pengindeksan pada metode subkueri klausa IN dan peningkatannya sebesar 0,049% setelah dilakukan pengindeksan pada metode subkueri klausa EXISTS.

Kesimpulannya adalah pengujian pada penelitian terdahulu lebih signifikan dibandingkan dengan pengujian pada penelitian ini. Referensi pada ketiga peneliti terdahulu lebih dominan menggunakan basis data Microsoft SQL Server 2008 sedangkan penelitian ini menggunakan basis data Oracle 12c. Dari hasil perbandingan ini, perlu dilakukan penelitian lebih lanjut mengapa hasil pengindeksan pada penelitian ini sangat kurang signifikan jika dibandingkan dengan penelitian terdahulu.

SIMPULAN

Berdasarkan pengujian diatas, dapat disimpulkan sebagai berikut:

a. Metode subkueri Klausa IN dan EXISTS tidak memiliki perbedaan yang signifikan.
b. Pada pengujian metode subkueri Klausa IN, percobaan yang kedua dan kelima hampir memiliki waktu (detik) yang sama dari keduanya (index maupun non-index).
c. Pada pengujian metode subkueri Klausa EXISTS, percobaan yang kedelapan hampir memiliki waktu (detik) yang sama dari keduanya (index maupun non-index).
d. Dari 2 (dua) poin diatas ini dapat disimpulkan bahwa perlu adanya penelitian dari sisi lain seperti performa server. Karena penulis disini melakukan remote ke server basis data menggunakan GUI database tools, sehingga penulis memiliki pendapat bahwa kemungkinan dampak dari jaringan maupun perangkat keras itu sendiri memberikan efek tersendiri saat GUI database tools ini menampilkan datanya.
e. Rata – rata dari pengujian waktu metode subkueri Klausa IN, memiliki hasil dari 3,114 detik sebelum dilakukan pengindeksan menjadi 2,972 detik setelah dilakukan pengindeksan.
f. Rata – rata dari pengujian waktu metode subkueri Klausa EXISTS, memiliki hasil dari 3,087 detik sebelum dilakukan pengindeksan menjadi 2,935 detik setelah dilakukan pengindeksan.

DAFTAR PUSTAKA

Verma, A. (2011). Enhanced Performance of Database by. IJCSMS International Journal of Computer Science & Management Studies.

Gupta, M. K., & Chandra, P. (2011). An Empirical Evaluation of LIKE Operator in Oracle. BVICAM's International Journal of Information Technology.

Connolly, T. M., & Begg, C. E. (2010). Database Systems : A Practical Approach to Design, Implementation, and Management. Boston: Pearson Education.

Mercioiu, N., & Vladucu, V. (2010). Improving SQL Server Performance. Informatica Economică

Marlene, M., et al. (2014). Analisis Alat Bantu Tuning Fisikal Basis Data pada SQL Server 2008.

Indrajani. (2015). Analisis dan Penerapan Metode Tuning pada Basis Data Funding.

Oktavia, T. (2014). Evaluation Subquery Methods in Microsoft SQL Server 2008.

Lungu, I., Mercioiu, N., & Vladucu, V. (n.d.). Optimizing Queries in SQL Server 3008. Scientific Bulletin – Economic Sciences, Vol. 9 (15).

Hamad, M., M. (2008). Bitmap Index: A Data Structure for Fast File Retrieval.