# Implementasi Metode Indexing dan Penggunaan Subquery untuk Optimalisasi Database Rawat Jalan Rumah Sakit Menggunakan Mysql

# *Implementation of Indexing Method and Use of Subquery for Optimizing Hospital Outpatient Database Using Mysql*

**Wilsen Grivin Mokodaser*[1], Monica Dwijayanti[2], Samidi[3]**
[1,2,3]Magister Ilmu Komputer, Fakultas Teknologi Informasi, Universitas Budi Luhur; Jl Ciledug Raya, Petukangan Utara, Jakarta Selatan, 12260. DKI Jakarta
e-mail: [1]**2111602104@student.budiluhur.ac.id**, [2]2111602260@student.budiluhur.ac.id, [3]samidi@budiluhur.ac.id

***Abstrak***

*Kebutuhan layanan rawat jalan rumah sakit yang dapat diakses dengan cepat oleh masyarakat merupakan aspek yang sangat penting. Dengan melihat keterpurukan yang disebabkan oleh pandemi Covid-19 yang melanda dunia sejak tahun 2019 dan berdampak pada berbagai aspek ekonomi termasuk layanan rumah sakit, digitalisasi sistem rumah sakit harus diterapkan untuk dapat menjadi solusi dalam memberikan layanan yang cepat bagi pasien yang datang berobat. sistem informasi tersebut bukannya tanpa kendala, seiring bertambahnya jumlah data sehingga pemilihan basis data yang tepat termasuk penggunaan kueri yang tepat dapat membantu memberikan output yang tepat dan cepat. metode indexing dapat diterapkan pada tabel dengan jumlah basis data yang besar. Dengan menggunakan Btree Indexing pada tabel dan melakukan pengujian menggunakan subquery Exist menghasilkan nilai rata-rata 0,52015 dibandingkan dengan tabel yang tidak dilakukan index yang membutuhkan nilai rata-rata 3.08008 dengan menggunkan kuery yang sama. sedangkan pengujian menggunakan subqeury IN menghasilkan nilai akses rata-rata pada tabel yang telah diindex 0.00213.*

***Kata kunci***—Subquery, Exist, IN, Btree Indexing.

***Abstract***

*The need for outpatient hospital services that can be accessed quickly by the community is a very important aspect. By looking at the downturn caused by the Covid-19 pandemic that has hit the world since 2019 and impacting various aspects of the economy including hospital services, digitalization of the hospital system must be implemented to be a solution in providing fast service for patients who come for treatment. the information system is not without problems, as the amount of data increases so that the selection of the right database including the use of the right query can help provide the right and fast output. indexing method can be applied to tables with a large number of databases. Using Btree Indexing on tables and testing using the Exist subquery produces an average value of 0.52015 compared to tables that are not indexed which requires an average value of 3.08008. while testing using the IN subqeury produces an average access value in indexed tables of 0.00213.*

***Keywords***—Subquery, Exist, IN, Btree Indexing.

## 1 PENDAHULUAN

Seiring dengan perkembangan wabah Covid-19 yang telah melanda sejak awal tahun 2020, membuat banyak orang membutuhkan akses pada layanan rumah sakit yang berkualitas baik. Salah satu layanan yang ditawarkan oleh rumah sakit adalah poliklinik rawat jalan pasien. Jika dibandingkan dengan pelayanan rawat inap maka pelayanan rawat jalan lebih cepat perkembangannya baik dari segi pendapatan maupun dari segi pilihan pasien jika membutuhkan perawatan [1]. Di Indonesia sendiri proses penerimaan pasien di poliklinik rawat jalan kebanyakan masih menggunakan sistem pendaftarann manual [2]. Itulah sebabnya ketika masyarakat berobat ke poliklinik, salah satu kendala yang dihadapi adalah lamanya antrian yang harus dilakukan mulai dari pendaftaran, konsultasi dokter, pembayaran dan jika ada kebutuhan periksa secara lebih lanjut di laboratorium atau radiologi hingga mengantri untuk mengambil obat di area farmasi. Hal ini terlihat dari *waiting time* yang dihasilkan dari data kunjungan pasien pada bulan Januari 2021 menunjukkan 1,357 pasien yang melakukan perawatan, ada 621 pasien yang mengalami keterlambatan berkunjung mencapai sebesar 45,76% dari total data jumlah kunjungan pasien. Pada proses pendaftaran juga mengalami keterlambatan pendaftaran pasien dengan rata-rata waktu yang dibutuhkan dari mengambil nomor antrian hingga menyelesaikan proses pendaftaran adalah 15,02 menit untuk satu pasien

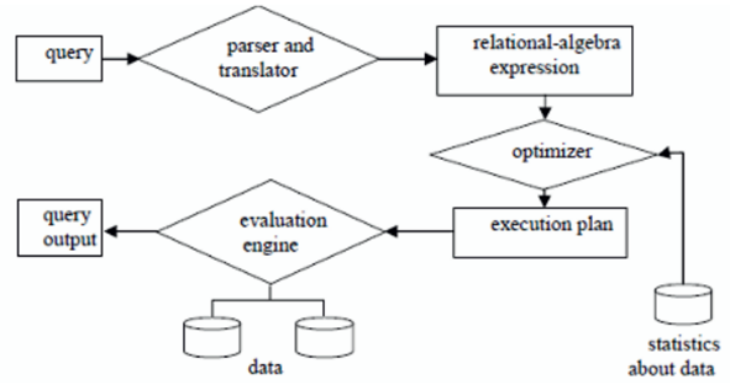
Salah satu terobosan yang dilakukan oleh rumah sakit adalah beralihnya penggunaan metode manual diimplementasikan ke dalam bentuk digitalisasi[3], yang juga mendukung proses transformasi dari sistem manual ke sistem yang terkomputerisasi dengan penggunaan *database* yang menjadi komponen penting dalam pengembangan teknologi informasi untuk menyimpan berbagai jenis data [4]. Penggunaan sistem *database* digunakan untuk menampung data dokter, *resume* medis, informasi terkait obat-obatan dan data pribadi pasien, pembangunan database yang baik dapat menghindari pengulangan (redudansi) yang tidak perlu untuk memenuhi kebutuhan dari proses pendaftaran pasien[5]. Dengan adanya pelayanan di area rawat jalan dapat berjalan optimal sebagai bentuk usaha untuk pasien yang akan berobat agar bisa terus memilih menggunakan jasa dari rumah sakit ini dan tidak beralih pada rumah sakit lainnya. Terkomputerisasi nya sistem rumah sakit dapat juga digunakan untuk memperoleh informasi pasien, mengatur jadwal dokter dan proses bisnis lainnya [6]. Proses digitalisasi yang digunakan oleh rumah sakit bukanlah tanpa kendala, seiring bertambahnya jumlah data dengan model formulasi kueri yang belum optimal tentu akan mengakibatkan kinerja akses data menjadi lambat dan kurang efektif [7]. Hal itu juga bisa menyebabkan *waiting time* yang harus ditempuh oleh pasien yang mengakses layanan rawat jalan menjadi lebih lama. Oleh sebab itu kinerja kueri yang tidak memadai untuk melakukan output yang sesuai menjadi faktor penyebab terjadinya keterlambatan dalam pemrosesan data [8].

Penelitian kali ini akan lebih membahas terkait optimalisasi *database* yang digunakan oleh rumah sakit untuk menyimpan data-data yang terkait dengan layanan rawat jalan. Pada penelitian terdahulu terdapat beberapa referensi jurnal yang juga membahas mengenai penggunaan *indexing* dan *subquery*. Seperti penelitian pertama yang ditulis oleh Novianty dkk pada tahun 2018 yang membahas tentang tentang perbandingan *query response time* pada model *query view* dengan jumlah data 100, 500 dan 1,000 mendapatkan selisih waktu sebesar 0,0012 detik lebih cepat daripada menggunakan *cross product* [9]. Penelitian kedua dilakukan oleh Edi & Witono pada tahun 2022 dengan memfokuskan tentang perbandingan *response time* penggunaan *index, view* dan *materialized view database MySQL* yang menghasilkan untuk penggunaan metode *index* dengan jumlah data 141.037 – 198.996 baris data diperoleh waktu 1,48-2,51 detik menggunakan *metode views* dan 0,01 detik menggunakan *materialized views* sehingga memperoleh kesimpulan bahwa penggunaan *materialized views* jauh lebih cepat dari pada *index* maupun *views* [10].

Setelah melihat kedua penelitian yang telah dibahas mengenai metode *index*, pada jurnal ketiga penelitian dilakukan oleh Samidi dkk pada tahun 2022 yang membahas mengenai penggunaan klausa Exist dan In dengan menggunakan *database* Oracle 12c dengan objek

penelitian pada aplikasi Badan Nasional dan Pertolongan (BASARNAS) dengan hasil pengujian didapatkan dari 10 kali percobaan pada data sebanyak 84.333 menyatakan bahwa metode *subquery* klausa IN dan EXIST tidak memiliki perbedaan yang signifikan pada tabel yang telah di *index* maupun belum di *index*. Penelitian tersebut berbanding terbaik dengan penelitian yang dilakukan oleh Oktavia dan Sujarwo yang melakukan percobaan pada SQL Server 2008 dengan hasil yang didapatkan terjadi peningkatan setelah dilakukan proses *indexing* dengan besar peningkatan yang terjadi sebesar 36,9% untuk klausa IN dan 40,3% untuk Klausa EXIST [11]. Penggunaan index pada database akan mempercepat kinerja dalam pengelolaan data, salah satunya menggunakan metode B tree indexing, Btree sendiri merupakan sebuah pohon pencarian dalam basis data yang dimana strukturnya memungkinkan data yang disimpan untuk melakukan searc, insert, delete yang dapat dilakukan secara terstruktur [12]. Btree sendiri dibuat untuk memungkinkan dapat dilakukan penyimpanan banyak data dalam satu node, jumlah sub pohonnya juga dapat sangat banyak itulah yang menyebabkan Btree sangat cocok untuk digunakan dalam pengelolaan data [13].

Dari beberapa jurnal terdahulu yang bisa dijadikan landasan atau tolak ukur terhadap penelitian yang akan dilakukan guna memastikan apakah efektif atau tidaknya metode yang digunakan, sehingga penulis menyimpulkan akan menggunakan metode *indexing* pada tabel dan menggunakan *subquery* klausa pada basis data menggunakan *tools MySQL* yang menjadi pembeda dari penelitian yang telah dilakukan sebelumnya. Dibutuhkan pengoptimalan *query* untuk mengakses data pada *database* dan entitas-entitas yang ada pada *database* tersebut. Penggunaan SQL dapat digunakan dalam berbagai *database* yang ada sehingga dapat mempermudah walaupun berpindah dari satu tabel *database* ke tabel *database* selanjutnya. Penggunaan *index* pada *database* memangkas waktu yang digunakan menjadi lebih cepat dalam proses mencari nilai kolom pada tabel tertentu [14]. Untuk meningkatkan kinerja dari DBMS (*database* management system) perancangan skema data yang efisien, optimalisasi indeks, perencanaan eksekusi, pemantauan akses data dan optimalisasi *query* dengan melihat proses yang terjadi pada kinerja eksekusi *query* pada gambar dibawah ini[15]:




Gambar 1 Proses Eksekusi Query

## 2 METODE PENELITIAN

### *2.1 Penelitian yang terkait*

Dalam tahapan pembuatan penelitian ini, tentu saja tidak lepas berdasarkan pendukung penelitian yang relevan. Penelitian yang relevan merupakan penelitian terdahulu yang berkaitan, sejenis, persis atau identik dengan menggunakan penelitian yang dilakukan. Hal ini bisa sebagai bahan acuan atau pembanding pada penelitian yang dilakukan.

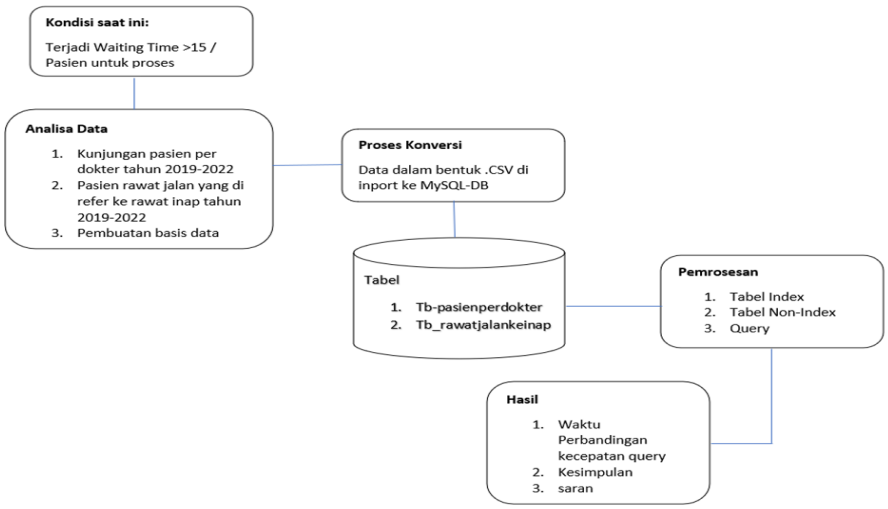
Tabel 1 Penelitian yang Relevan

| No | Peneliti - Tahun | Judul Penelitian | Hasil Penelitian | Variabel Penelitian |
|---|---|---|---|---|
| 1. | Samidi, S., Andharu, D., & Widyarto, F. (2022) | Optimasi Database Menggunakan SQL *Query* Klausa IN dan EXIST pada Database Oracle 12c. Studi Kasus pada Aplikasi di Badan Nasional Pencarian dan Pertolongan (BASARNAS) | Pengujian pada semua metode *subquery* 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. | Data*set*: 84,333 *record* data presensi Universitas Binus dengan database Microsoft SQL Server 2008. |
| 2. | Utomo, M. N. Y., Bastian, A., & Winursito, A. (2020) | *Improving Speed Performance of Select Random Query in SQL Database* | Didapatkan hasil percobaan metode FCO-Rand yang diusulkan memperoleh kecepatan proses terbaik dengan 0,074 detik pada 200.000 data, diikuti oleh SPO-Rand dengan 0,265 detik. Hasil ini jauh lebih cepat dari standar metode pilih acak (NO-Rand) yang memakan waktu hingga 7.035 detik untuk tugas yang sama. | Data*set*: Lima data acak dari beberapa set data, mulai dari 10.000 hingga 200.000 data. |
| 3. | James, Valerian, and Sixtus. (2018) | *Design and Implementation of a Hospital Database Management System (HDMS) for Medical Doctors* | Semua layanan masih dilakukan secara manual dengan adanya penelitian ini dirancang sebuah sistem rumah sakit untuk menangani Pusat Medis informasi seperti data pasien, persediaan (obat) manajemen, tagihan pasien dll yang dilakukan secara otomatis. | Data*set*: File *database* Rumah Sakit Our Lady of Mercy. |
| 4. | Tavares, O. M. I., Rangkoly, S. M., Bawan, S. B. D., Ahmad, K. B., Utami, E., & Mustafa, M. S. (2021) | Analisis Model Restrukturisasi Kueri dalam Mengoptimasi Waktu Respons Eksekusi Data pada Basis Data Relasional MySQL PHP 7.2.27 | Hasilnya terbukti dengan dilakukannya optimasi kueri didapatkan presentasi optimasi waktu yang signifikan yaitu sebesar 78% jauh lebih cepat dibandingkan dengan kueri awal. | Data*set*: Berjumlah 1000 data yang didapat dan diinputkan sebagai tambahan pada ketiga tabel uji. |

| | | | | |
|---|---|---|---|---|
| | | | Penelitian ini menghasilkan kesimpulan bahwa optimasi berperan penting dalam mengefisiensikan waktu respons kueri basis data. | |
| 5. | T. Oktavia and S. Sujarwo (2014) | Evaluation of Sub Query Performance in SQL Server | untuk aplikasi bisnis yang membutuhkan waktu pemrosesan yang cepat dalam menyimpan data dibandingkan waktu pemrosesan yang cepat dalam mengambil data, lebih baik tidak menerapkan pengindeksan dan untuk *subquery* lebih baik gunakan IN atau EXIST, karena jika pengindeksan diterapkan, pemrosesan waktu untuk menyimpan data akan meningkat di setiap INSERT operasi yang juga secara otomatis memperbarui indeks di tabel terkait. | Data*set*: data operasional kehadiran siswa di laboratorium yang terdiri dari 6 kolom dan 500.000 catatan**.** |

## *2.2 Kerangka Konsep*

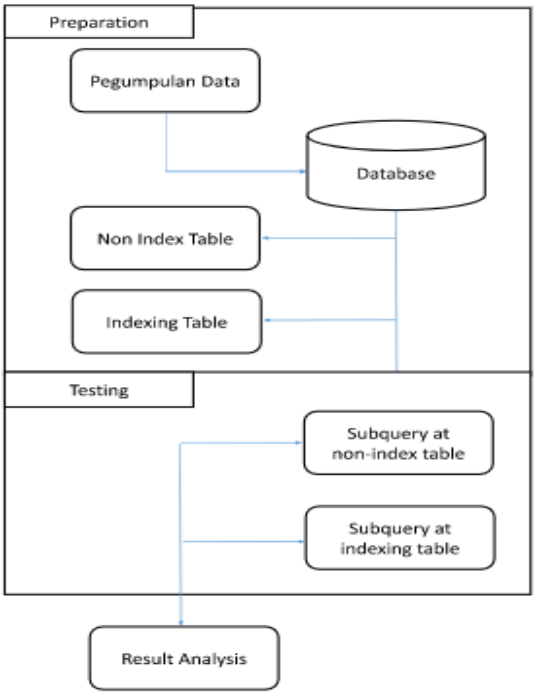
Setelah mengetahui studi literatur dan juga menggunakan data pendukung dari penelitian yang relevan, maka dibuatlah gambaran untuk penelitian ini sebuah kerangka konsep yang bisa dilihat pada gambar 2 berikut ini:




Gambar 2 Kerangka Konsep Penelitian

*2.3 Tahapan Penelitian*

Penelitian ini dilakukan dengan menggunakan Laptop V3B0LJL0 dengan spesifikasi prosesor AMD Ryzen 5 5600H dan RAM 8.00 GB, serta *operation system Windows 11 Home Single Language Version 21H2.* Untuk *tools* yang digunakan pada pengujian eksperimen ini menggunakan *MySQL* versi 1.6 dengan menggunakan log data kunjungan pasien pada setiap dokter dari tahun 2019 hingga 2022. Adapun tahapan dari penelitian ini dapat dilihat seperti pada gambar berikut:



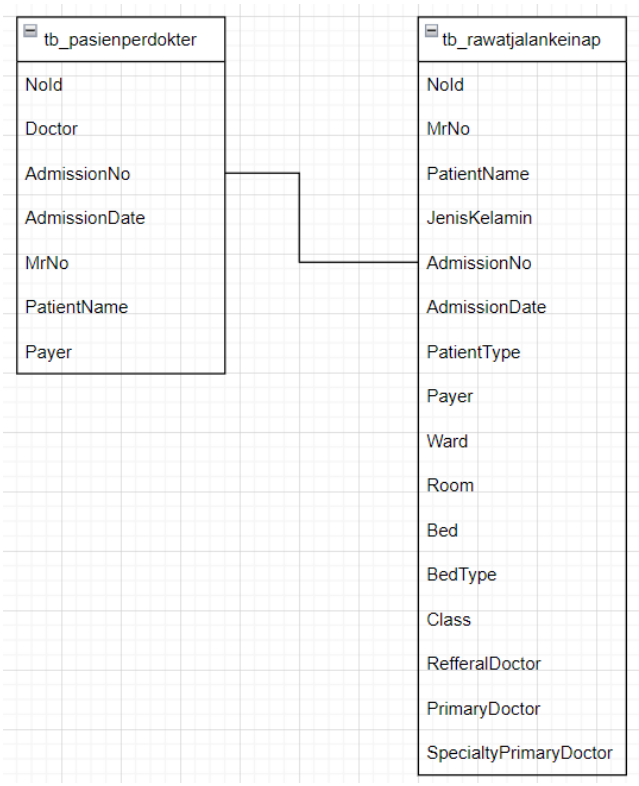

Gambar 3 Tahapan Penelitian

Gambar diatas menunjukkan tahapan yang akan dilakukan pada saat penelitian. Pada tahap awal dilakukan dengan mengambil data log pasien per dokter dan log pasien yang melakukan rawat jalan hingga masuk ke layanan rawat inap rumah sakit. Setelah tahap pengumpulan data selesai, maka data akan di *convert* ke dalam SQL *database.* Kemudian jika proses *create database* dan tabel selesai, tabel tersebut akan di duplikasi sehingga menghasilkan tabel yang telah di *index* maupun yang belum di *index. P*ada tahap *testing* dilakukan *subquery* pada masing-masing tabel baik yang belum di *index* maupun tabel yang telah di *index*. Hasil dari *subquery* tersebut akan dilakukan proses analisis data dan dilakukan perbandingan dari setiap percobaan yang rencananya akan diulang sebanyak 10 kali. Pengulangan *subquery* dilakukan lebih banyak untuk memberikan gambaran konsistensi *query* jika nantinya tabel yang telah terbentuk akan diakses oleh lebih banyak *user* dalam waktu yang bisa saja bersamaan.

## 3 HASIL DAN PEMBAHASAN

*3.1 Pembuatan basis data*

Pada penelitian ini penulis membuat basis data dari rumah sakit dalam bentuk .csv, kemudian di *import* ke dalam basis data *MySQL* sehingga menghasilkan dua tabel yaitu tb_pasienperdokter menjadi tabel dengan jumlah data yang lebih banyak dari tabel kedua yaitu tb_rawatjalankeinap. Bisa dilihat pada struktur tabel yang akan digunakan pada gambar berikut ini:




Gambar 4 Struktur Tabel

Gambar dibawah ini merupakan total jumlah data yang akan digunakan oleh penulis, yang mana dari kedua tabel tersebut belum dilakukan proses *indexing*.

| Table | Action | Rows |
|---|---|---|
| index_pasienperdokter | Browse Structure Search Insert Empty Drop | 222956 |
| index_rawatjalankeinap | Browse Structure Search Insert Empty Drop | 38551 |
| tb_pasienperdokter | Browse Structure Search Insert Empty Drop | 222956 |
| tb_rawatjalankeinap | Browse Structure Search Insert Empty Drop | 38551 |
| 4 table(s) | Sum | 523.014 |

Gambar 5 Gambar Jumlah Data

*Index* dan non-*index Table*

```
CREATE INDEX index_rawatjalankeinap ON index_rawatjalankeinap (AdmissionDate, AdmissionNo, Bed, BedType, Class, JenisKelamin, MrNo, NoId, PatientName, PatientType, Payer, PrimaryDoctor,
ReferralDoctor, Room, SpecialtyPrimaryDoctor,Ward) USING BTREE;

CREATE INDEX index_pasienperdokter ON index_pasienperdokter (AdmissionDate, AdmissionNo, Doctor, MrNo, NoId, PatientName, Payer) USING BTREE;
```

Gambar 6 Membuat *Index* pada Tabel

Pola index yang digunakan pada tabel adalah *BTree indexing*, yang mana menurut Samidi dkk [4] *index BTree* baik digunakan jika terdapat pola kolom dengan ragam nilai atau banyak (*High Cardinality*). Setelah selesai melakukan *indexing* pada tabel yang akan dilakukan pengujian selanjutnya adalah membuat *query* untuk kolom tertentu dan dilakukan proses *sorting* dengan rentan waktu tertentu. Berikut ini adalah *query* yang akan digunakan.

a. *Query EXIST* dengan tabel non-*index*

```
SELECT a.Doctor, a.AdmissionDate, a.MrNo, b.PatientName, b.AdmissionDate,
b.Class, b.Payer, b.Room
FROM tb_pasienperdokter a
LEFT JOIN tb_rawatjalankeinap b ON a.AdmissionNo>'OPA1908010522'
WHERE EXISTS (SELECT AdmissionNo FROM tb_rawatjalankeinap
        WHERE AdmissionDate BETWEEN '17-Jan-19'
        AND '25-Feb-19' AND a.AdmissionNo=b.AdmissionNo);
```

b. *Query* IN dengan Tabel Non-*Index*

```
SELECT a.PatientName, a.MrNo, b.AdmissionDate, b.Room, b.Class, b.Payer
FROM tb_pasienperdokter a
INNER JOIN tb_rawatjalankeinap b ON a.AdmissionNo=b.AdmissionNo
WHERE a.AdmissionNo IN (SELECT AdmissionNo FROM tb_pasienperdokter
                WHERE AdmissionDate BETWEEN '3/01/2020 12:50' AND '3/30/20 24:00' );
```

c. *Query* EXIST dengan tabel di *Index*

```
SELECT a.Doctor, a.AdmissionDate, a.MrNo, b.PatientName, b.AdmissionDate, b.Class,
b.Payer, b.Room
FROM index_pasienperdokter a
LEFT JOIN index_rawatjalankeinap b ON a.AdmissionNo>'OPA1908010522'
WHERE EXISTS (SELECT AdmissionNo FROM index_rawatjalankeinap
        WHERE AdmissionDate BETWEEN '17-Jan-19'
        AND '25-Feb-19' AND a.AdmissionNo=b.AdmissionNo);
```

a. Query IN dengan tabel di Index

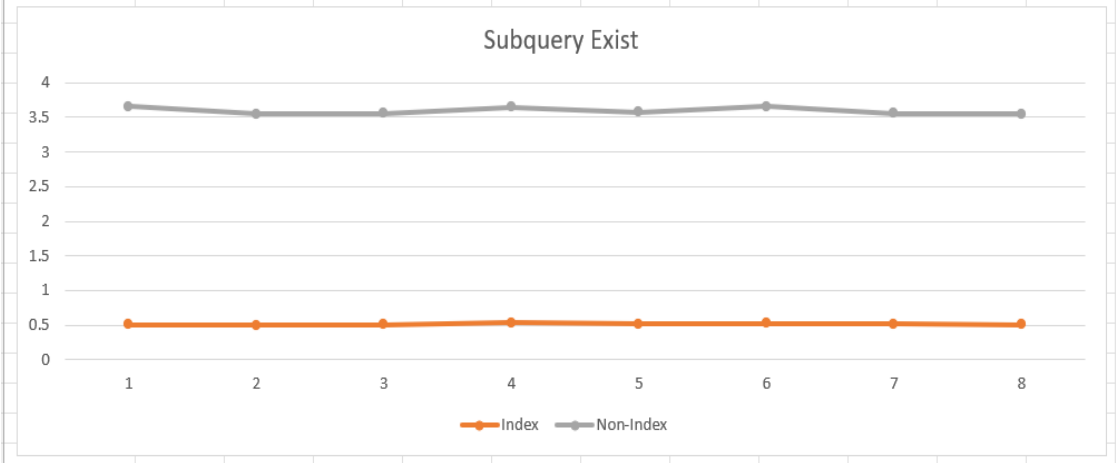
```
SELECT a.PatientName, a.MrNo, b.AdmissionDate, b.Room, b.Class, b.Payer
FROM index_pasienperdokter a
INNER JOIN index_rawatjalankeinap b ON a.AdmissionNo=b.AdmissionNo
WHERE a.AdmissionNo IN (SELECT AdmissionNo FROM index_pasienperdokter
                WHERE AdmissionDate
                BETWEEN '3/01/2020 12:50' AND '3/30/20 24:00' );
```

Dari keempat query diatas, terdapat informasi yang terdiri dari PatientName, MrNo, AdmissionNo, Room, Class, Doctor dan Payer dimana rentang waktu data yang diambil adalah kurang lebih 1-2 bulan dikarenakan untuk proses pasien yang melalui rawat jalan sampai ke rawat inap membutuhkan waktu yang cukup panjang mulai dari awal pendaftaran, diagnosa dokter, pemeriksaan radiologi dan laboratorium hingga keputusan dokter untuk melakukan proses rawat inap untuk pasien yang membutuhkan. dari hasil query tersebut diperoleh hasil sebagai berikut:

a. Pengujian Query Exist

Tabel 1 Hasil Subquery Exist

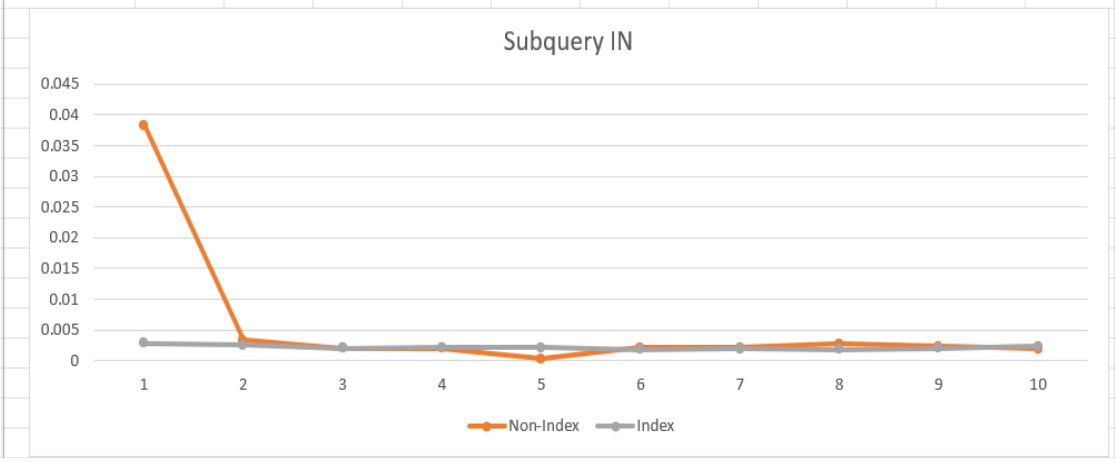
| Testing Subquery Exist | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Running ke | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | **Rata-rata** |
| Index | 0.5044 | 0.4958 | 0.5024 | 0.5326 | 0.5151 | 0.5207 | 0.5147 | 0.5015 | 0.5035 | 0.6108 | 0.52015 |
| Non-Index | 3.1510 | 3.0556 | 3.0553 | 3.1163 | 3.0609 | 3.1409 | 3.0396 | 3.0438 | 3.0860 | 3.0514 | 3.08008 |




b. Pengujian Query IN

Tabel 2 Hasil Subquery IN

| Testing Subquery IN | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Running ke | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | **Rata-Rata** |
| Non-Index | 0.0383 | 0.0033 | 0.0020 | 0.0020 | 0.0003 | 0.0021 | 0.0021 | 0.0027 | 0.0023 | 0.0019 | 0.0057 |
| Index | 0.0028 | 0.0025 | 0.002 | 0.0021 | 0.0021 | 0.0018 | 0.0019 | 0.0018 | 0.002 | 0.0023 | 0.00213 |




Dari hasil pengujian berupa grafik diatas kita dapat melihat beberapa perbedaan yang terjadi saat pengujian ini dilakukan. pada penelitian sebelumnya dilakukan dengan basis data Oracle 12c dan SQL Server 2008. Pada pengujian yang dilakukan dengan menggunakan Basis Data MySQL sekalipun jumlah data yang berbeda dengan penelitian terdahulu terdapat perbedaan tapi tidak signifikan.

## 4 KESIMPULAN

Berdasarkan pengujian diatas dapat disimpulkan sebagai berikut:

a. Sama seperti penelitian terdahulu metode Subquery IN dan Exist memiliki perbedaan akses data yang cukup signifikan
b. Percobaan Subquery IN yang pertama dengan tabel yang Non Index kemungkinan terjadi perbedaan waktu yang signifikan karena disebabkan oleh perangkat yang digunakan saat pengujian
c. Subquery IN terjadi perbedaan kecepatan akses antara tabel yang sudah diindex dan yang belum di index tapi perbedaan waktunya tidak terlalu signifikan
d. Pengujian Subquery Exist sangat terlihat kecepatan akses pada tabel yang belum dan yang sudah di index hal ini dapat dilihat dari rata-rata waktu yang dihasilkan saat mengakses data di tabel index dan non-index.

## 5 SARAN

Berikut saran bagi pengembangan penelitian selanjutnya:

1. Menambahkan penggunaan subquery lainnya selain IN dan EXIST.
2. Menggunakan lebih dari 2 tabel sehingga menghasilkan model query yang lebih kompleks untuk dapat mengetahui apakah akan terjadi perubahan kecepatan akses jika menggunakan subquery tersebut.

## DAFTAR PUSTAKA

[1] Reny Nugraheni and Yoanita Indra Kumalasari, 2020, "Evaluasi Sistem Informasi Pendaftaran Pasien Rwat Jalan di Rumah Sakit X Kota Kediri", *Jurnal Kesehatan*, vol. 8, no. 2, Hal 96-105.

[2] David Laksamana Caesar, 2019, "Studi Evaluasi Sistem Informasi Pendaftaran di Rumah sakit Umum Daerah dr. Loekmono Hadi Kudus", *Jurnal Kesehatan Masyarakat.*, vol. 7, no. 1, Hal 74-88.

[3] Robnah and Adi Widodo, 2022, "Tinjauan Sistem Informasi Pendaftaran Pasien Rawat Jalan di RSUD Tebet Jakarta Selatan", *Indonesian Journal of Health Information* Management *(IJHIM)*, vol. 2, no. 1.

[4] M. N. Y. Utomo, A. Bastian, and A. Winursito, 2020, "Optimasi Performa Kecepatan Query Select Random pada SQL Database", *INTEK J. Penelit.*, vol. 7, no. 1, Hal 26.

[5] Ellya Helmud, 2021, "Optimasi Basis Data Oracle Menggunakan Complex View Studi Kasus : PT. Berkat Opmimis Sejahtera (PT.BOS) Pangkalpinang", *Jurnal Informanika*, vol. , no. 1.

[6] A. J. C., A. V. C., and N. S. E., 2018, "Design and Implementation of a Hospital Database Management System (HDMS) for Medical Doctors", *Int. J. Comput. Theory Eng*, vol. 10, no. 1, hal 1–6.

[7] O. M. I. Tavares, S. M. Rangkoly, S. B. D. Bawan, K. B. Ahmad, E. Utami, and M. S.

Mustafa, 2021, "Analysis of Query Restructuration Models in Optimizing Data Execution Response Time on Mysql Php 7.2.27 Relational Databases", *J. Komput. dan Inform.*, vol. 9, no. 1, hal 1–10.

[8] S. Samidi, D. Andharu, and F. Widyarto, 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, *J. Pendidik. Tambusai*, vol. 6, no. 1, hal. 2297–2306.

[9] P. Noviyanti, A. Deolika, S. Hartinah, C. A. Haris, T. Maryana, and N. D. Sari, 2018, "Perbandingan Query Response Time pada Model Query View dan Cross Product Comparison of Query Response Time in the Query View and Cross Product", *e-Jurnal JUSITI*, vol. 7, no. 2, hal. 131–141.

[10] E. Witono, 2022, "Perbandingan Response Time Penggunaan Index , Views , dan Materialized Views Database Mysql", *Jurnal Sains Komputer & Informatika*, vol. 6, hal 499–506.

[11] T. Oktavia and S. Sujarwo, 2014, "Evaluation of sub query performance in SQL server, *EPJ Web Conf*", vol. 68, hal 1–5.

[12] Diva Putri Anasya, 2022, "Optimasi Hasil Query Pada Fitur Pencarian Platform Marketplace Penjualan dan Seea Menyewa Buku Online", *Seminar Nasional & Call Paper Fakultas Sains dan Teknologi (SENASAINS)*, Sidoarjo, 2 Juni.

[13] Dike Pradana Putra, 2015, "Implementasi Fulltext Indexing pada Dokumen Elektronik dengan Algoritma B-Tree", *e-Prodeeding of Enginering*, Vol 2, hal 1119.

[14] R. Pamungkas, 2018, "Optimalisasi Query Dalam Basis Data Mysql Menggunakan Index", *Res. Comput. Inf. Syst. Technol. Manag.*, vol. 1, no. 1, hal. 27.

[15] Indra Warman and Wildani, 2021, "Analisa Kinerja Query Stored Procedure Pada Database management System (DBMS) MYSQL", *Jurnal Sains dan Teknologi*, vol. 21, no. 1.