OPTIMALISASI DATABASE TRANSAKSI TELEKOMUNIKASI DENGAN NOSQL MEMCACHED

Riki Ramdani Saputra*[1]
[1]Universitas Budi Luhur; Jakarta Selatan, Telp: 021-3928688-89 Fax: 021-3161636
[3]Magister Ilmu Komputer, Fakultas Teknologi Informasi, Jakarta
e-mail: ***[1]2111601882@student.budiluhur.ac.id**

Abstrak

Semakin bergantungnya masyarakat pada aplikasi teknologi informasi saat ini maka developer dituntut untuk dapat memberikan kemudahan dalam menyediakan layanan, termasuk didalamnya adalah layanan transaksi yang dapat dilakukan dari berbagai sarana telekomunikasi. Hal ini menyebabkan tabel transaksi memiliki beban kerja sangat berat karena diakses dari berbagai aplikasi baik aplikasi berbasis web maupun aplikasi smartphone. Dengan besarnya transaksi melalui aplikasi berbasis web, maka dituntut untuk efisiesi dari metode web transaksi dalam mengakses database. Dengan hadirnya metode noSql membuat hasil pemaggilan data (query) dari client dapat disimpan pada memori server sebagai penyimpanan sementara (cache), tujuannya untuk membuat web transaksi tetap responsif walupun banyak diakses dan melakukan pemanggilan data (query) berulang. Dengan penggunaan metode NoSql, kinerja database transaksi dapat ditekan hingga 5%.

Kata kunci— database, memcache,nosql, optimasi, telekomunikasi, website

Abstract

Nowadays, more people depend on application computer; therefore, developers are required to be able to provide convenience in providing services. Those including transaction services that can be carried out from various telecommunication facilities. These issues causing the transaction table to have a very-heavy workload because it is accessed from various applications, both web-based applications and smartphone applications. With the large number of transactions through web-based applications, it is required for the efficiency of the web transaction method in accessing the database. With the presence of the noSql method, the results of data calls (query) from the client can be stored on the server memory as temporary storage (cache). The goal is to make the transaction website remains responsive, even though it is accessed a lot and having repeated data calls (queries). By using the NO Sql method, transaction database performance can be suppressed by up to 5%.

Keywords— database, memcache,nosql, optimization, telecommunication, website

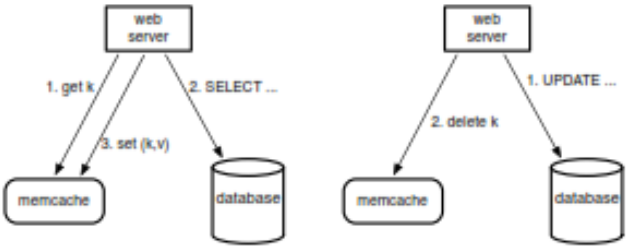
1. PENDAHULUAN

Berawal dari makin membengkaknya biaya pengembangan infrastruktur sistem informasi membuat perusahaan harus membuat langkah agar biaya pengembangan dapat ditekan, sementara biaya pengembangan tersebut dihitung dari bertambahnya pengguna dan perkiraan kinerja dari infrastruktur tersebut. Oleh karena itu perusahaan memberi tugas kepada seluruh *stakeholder* IT untuk melakukan efisiensi agar tercipta biaya pengembangan yang lebih kecil namun tidak mengurangi performa layanan ke pengguna. Ini menjadi tantangan tersendiri ke setiap divisi IT

untuk mencari cara agar hal tersebut dapat terwujud karena pada hakikatnya seluruh pekerjaan yang dilakukan seluruh divisi saling terkait dan ketergantungan.

Layanan IT merupakan motor utama pada dunia telekomunikasi dan menjadikan teknisi IT development dan Database Administrator sosok paling depan dengan disokong oleh tim infrastruktur dan tim lainnya. Divisi IT diwajibkan menjadi satu kesatuan yang tangguh dalam memberikan seluruh layanan secara optimal kepada pengguna. Salah satu layanan yang diberikan adalah layanan transaksi dimana didalamnya tercakup data yang sangat banyak dan diakses dari berbagai platform, baik melalui aplikasi berbasis web, aplikasi *smartphone*, maupun aplikasi aplikasi internal lain serta untuk data *reporting*. Oleh karena itu database transaksi menjadi sangat sibuk dan kadangkala tidka bisa dihindari bebeban kerja database menjadi tidak otpimal saat performa server digunakan secara penuh untuk melayani seluruh perintah kerja dari pengguna. Setelah ditelaah salah satu penyebab nya adalah banyaknya proses pemanggilan data (*query*) dari aplikasi bebrbasis web yang masih banyak digunakan terutama di *gallery* dan *call center*, masalah yang terjadi adalah banyaknya pemanggilan data (*query*) berulang dan bahkan bisa menyebabkan status perintah pemanggilan data (*query*) yang menggantung dimana aplikasi berbasis web sudah terputus *(timeo out)* namun dtabase tetap menjalankan proses tersebut walau sudah tidak dibutuhkan.

Merujuk pada apa yang telah dijelaskan oleh Chang Bao Rong dkk. dalam jurnal "Integration and optimization of multiple big data processing platforms" [1] dan dikuatkan oleh Abdullah Talha Kabakus, and Resul Kara dalam jurnal "A performance evaluation of in-memory databases" [2] serta penjelasan tentang NoSQL Data Stores Based on Key Design oleh Makris, Antonios dalam jurnal " A Classification of NoSQL Data Stores Based on Key Design Characteristics" [3] dapat ditarik satu kesimpulan bahwa salah satu cara untuk membantu meringankan beban kerja server database dari sisi pengembangan aplikasi berbasis web adalah dengan memanfaatkan fungsi dari cache dan mengkombinasikannya dengan database No Sql, dimana cara kerjanya adalah menyimpan sementara data yang didapat dari hasil *query* database dan menyimpannya beberapa waktu sehingga saat ada *query* yang sama maka database No Sql tinggal menampilkan data yang masih ada di dalam cache sehingga tidak perlu lagi mengambil data dari database. Metode ini pun rupanya telah digunakan oleh Facebook seperti yang ditelaah oleh Nishtala R dkk dalam jurnal "Scaling Memcache at Facebook"[4]. Selain hal tersebut Memcached sudah dapat digunakan juga karena platform framework yang saat ini digunakan sudah mendukung dengan library yang sudah disediakan.



Gambar 1. Proses Memcached

Database No SQL yang mendukung fungsi dalam optimasi aplikasi berbasis web adalah No SQL Memcached, dengan metode menyimpan data sementara pada memory dalam rentan waktu tertentu, jadi alur aplikasi mendapatkan data yang tadinya langsung melakukan *query*ke database dialihkan dulu untuk mengecek memcached, jika data ada dalam memcached maka data langsung dikirim ke client namun jika tidak ada maka proses query akan dilanjutkan ke database dan data yang dihasilkan akan dikirim ke client dan juga dikirim ke memcached untuk disimpan.

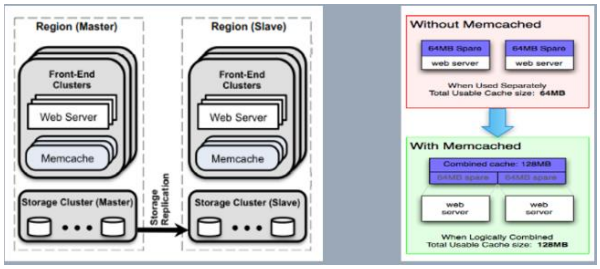
Metode dalam pengujian ini seperti dijabarkan diatas, jadi saat pengguna pertama kali memanggil data maka data akan diproses oleh server aplikasi kemudian server aplikasi melakukan pengecekan apakah ada data berdasarkan *key* yang dicari dalam memory, jika hasilnya tidak ada

maka diteruskan ke server Database selanjutnya hasil proses dikirim kembali ke server aplikasi yang kemudian ditampilkan ke pengguna dan dikirim pula ke memory dalam bentuk *array* sesuai *key* yang dibuat dan akan disimpan dalam *memory* sesuai dengan waktu yang ditentukan, dan jika pengguna melakukan panggilan data yang sama dan data tersebut masih disimpan di memory maka server aplikasi langsung menampilkan data dari memory tapi meminta server database untuk melakukan pemrosesan data yang tentunya membutuhkan waktu yang lebih lama, seperti yang telah dijabarkan oleh Kulvinder Singh, and Ajit Singh dalam jurnal "Memcached DDoS Exploits: Operations, Vulnerabilities, Preventions and Mitigations" [5] serta diperrkuat oleh pemaparan oleh Yoann Ghogof dkk dalam 18th USENIX Symposium on Networked Systems Design and Implementation dengan judul "Accelerating Memcached using Safe In-kernel Caching and Pre-stack Processing" [6]tentang pemahaman tentang cara kerja Memcached.

Pemilihan NO SQL dalam penelitian ini merujuk pada penelitian sebelumnya yang dituangkan dalam sebuah tulisan yang dipublikasikan menjadi sebuah jurnal oleh Danny Kriestanto, Alif Benden Arnado dalam jurnal berjudul "Implementasi Website Pencarian Kos Dengan Nosql" [7] dan jurnal milik dari Erwin Aprliyanto dkk dalam judul "Perbandingan Kinerja Database NoSql Dengan Database Sql Server Pada Pemesanan Tiket Pesawat Online" [8] telah memberikan gambaran nyata dalam penerapan No SQL dalam aplikasi yang sudah terimplementasi dan didapatkan kelebihan kelebihan dari No SQL, selanjutnya Ari Fadli dkk dalam jurnal "Analisis Perbandingan Unjuk Kerja Database SQL dan Database NoSQL Untuk Mendukung Era Big Data " [9] menguatkan kelebihan No SQL dalam optimasi pada big data. Setelah mengetahui lebih dalam tentang kelebihan No SQL penulis memutuskan menggunakan Memcached karena dirasa cukup sesuai dengan data yang akan diproses serta aplikasi serta infrasturktur yang digunakan saat ini merujuk pada jurnal "Metode-MetodeOptimasiMemcachedsebagaiNoSQLKey-value MemoryCache" yang ditulis oleh Mandahadi Kusuma [10], serta pada jurnal "Scaling Memcache at Facebook" [4]oleh Rajesh Nishtala, yang jadi pertimbangan adalah jika data sebesar Facebook saja dapat dioptimalisasi oleh Memcached maka dengan data yang lebih kecil dapat sangat optimal hasil otpmilasisasi dari penggunaan Memcached. Namun pada jurnal jurnal yang menjadi rujukan belum menyentuh dari optimilasiasi terhadap resource infrastruktur, karena tujuan akhir dari penelitian ini adalah otpimasi dan efisiensi resource infrastruktur IT pada sebuah perusahaan provider Telekomunikasi.

2. METODE PENELITIAN

Memcached memberikan solusi untuk memanfaatkan memory yang masih tersedia dimana anda dapat memanfaatkan lebih banyak *memory* dari ukuran *memory* yang telah dialokasikan untuk anda. Seperti memungkinkan anda untuk memiliki atau menggunakan bagian memori sistem dialokasikan ke server lain di mana sistem memiliki lebih dari memori yang dibutuhkan, kemudian mengalokasikan untuk operasi server memiliki memori kurang dari yang dibutuhkan [5]. Dengan penggunan NoSql Memcached memberikan dampak pengurangan dalam proses pemanggilan data (*query*) ke Database Server karena data data yang sudah diakses akan dipindahkan ke *memory memory* berlebih pada sistem server komputer.



Gambar 2. Alur kerja penerapan Memcached

Dalam Pengujian kali ini menggunakan server test Bed yang memiliki spesifikasi sama persis dengan server sesungguhnya yang bertujuan agar apapun yang dikerjakan oleh developer nantinya akan mendapatkan hasil yang sama jika sudah berjalan di server production, dan untuk bahasa pemrograman menggunakan PHP dengan farmework CI dimana framework ini sudah menyediakan library untuk penggunaan Memcached serta didukung oleh database Oracle sebagai penyimpanan datanya.

Data yang digunakan dalam pengujian ini adalah merupakan data yang sama persis dengan data yang ada pada database dengan melakukan penarikan data pada server database production dan menyalinnya di server database *test bed*, data transaksi ini sangat besar dan melibatkan banyak tabel, sehingga sekali proses pemanggilan data banyak digunakan penggabungan data antar tabelnya agar data yang diminta pengguna dapat ditampilkan seutuhnya, sehingga memakan waktu dalam pencarian datanya. Disamping itu database transaksi sendiri tidak hanya diakses oleh aplikasi transaksi berbasis web namun diakses juga oleh semua aplikasi yang membutuhkan data tersebut, dan di waktu tertentu data tersebut ditarik juga untuk keperluan reporting dan proses *back up* data. Jadi nanti data yang didapat akan di rubah menjadi sebuah array dengan menentukan key sebagai penanda pencarian dan akan disimpan dalam memory dalam waktu tertentu sehingga selama data tersebut ada dalam *memory* maka server aplikasi tidak akan meminta sever database untuk melakukan proses pencarian data yang diminta

```
class CI_Cache_memcached extends CI_Driver {

    private $_memcached;    // Holds the memcached object

    protected $_memcache_conf    = array(
                    'default' => array(
                        'default_host'      => '127.0.0.1',
                        'default_port'      => 11211,
                        'default_weight'    => 1
                    )
                );

    // ------------------------------------------------------------------------

    /**
     * Fetch from cache
     *
     * @param   mixed       unique key id
     * @return  mixed       data on success/false on failure
     */
    public function get($id)
    {
        $data = $this->_memcached->get($id);

        return (is_array($data)) ? $data[0] : FALSE;
    }
```

Gambar 3. Proses alur program dengan memcached

Dalam pengujian ini untuk mengetahui perbedaan yang dapat dihasilkan oleh penggunaan Memcached dalam aplikasi transaksi berbasis web dengan PHP sebagai bahasa pemrogramannya, Untuk merekam waktu eksekusi masing-masing operasi dasar tersebut digunakan fungsi PHP *microtime*. Fungsi ini akan mencatat waktu dari awal runtime skrip dan sampai selesai [9]. Selanjutnya akan dibuat suatu keadaan dimana banyak aplikasi transaksi berbasis web yang

melakukan permintaan data (*query*) dengan menggunakan teknik *cron job* di jam yang sama persis di hari yang berbeda untuk melihat kondisi traffic pada server database. Pada saat pengujian menggunakan waktu 300 detik atau 5 menit dalam penyimpanan data sementara di Memcached.

```
public function save($id, $data, $ttl = 300)
{
    if (get_class($this->_memcached) == 'Memcached')
    {
        return $this->_memcached->set($id, array($data, time(), $ttl), $ttl);
    }
    else if (get_class($this->_memcached) == 'Memcache')
    {
        return $this->_memcached->set($id, array($data, time(), $ttl), 0, $ttl);
    }

    return FALSE;
}
```

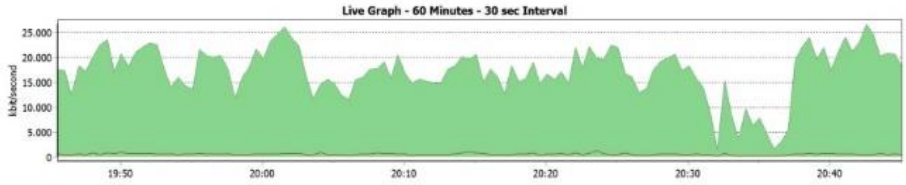
Gambar 4. Source code menyimpan data hasil pemanggilan data selama 300 detik ke dalam Memcached

Gambar 5 menunjukan perbedaan hasil waktu eksekusi proses pemanggilan data hingga data tampil dimana percobaan pertama masih langsung melakukan pemanggilan data dari database server, dan percobaan kedua diambil tiga puluh detik setelah proses pertama dan data hasil pemanggilan data pertama masih disimpan dalam memcached. Dari hasil pengujian didapatkan hasil efisiensi sebesar 61% dalam proses penarikan data (*query*).

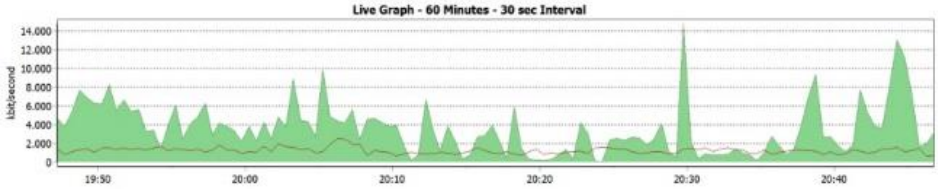
```
php > echo microtime();
0.95489000 1469455565
php > echo microtime();
0.58843800 1469455569
php >
```

Gambar 5. Hasil pengujian pemanggilan data

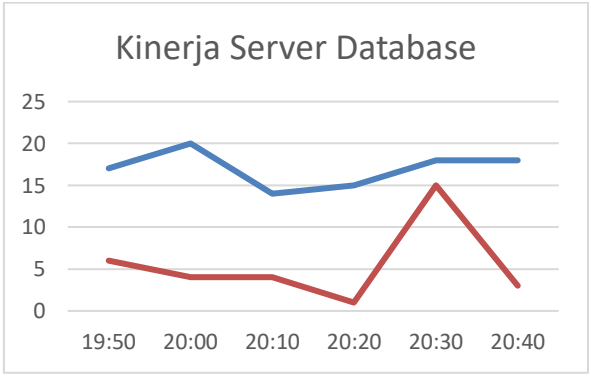
Gambar 6 menunjukan hasil waktu proses pemanggilan data dimana di hari pertama tidak menggunakan memcached, dimana semua proses pemanggilan data dan proses penarikan data hingga data ditampilkan pada pengguna. Gambar 7 menunjukan hasil penggunaan Memcached dengan menggunakan perintah Cron Job yang sama persis dengan percobaan di hari yang pertama. Gambar ketujuh menunjukan penurunan traffic dari transaksi aplikasi berbasis web yang berhubungan dengan proses penanganan proses pemanggilan data hingga tahap data ditampilkan kepada pengguna. Untuk skema pengujian dengan *cron job* ini dilakukan dengan komposisi pada pukul 19:50dilakukan Hit dengan 500 dengan tarnsaksi baru kemudian hingga 19:59 dilakukan Hit dengan 400 transaksi baru dan 100 transaksi berulang tiap menitnya, pada pukul 20:00 hingga 20:10 dilakukan Hit dengan 300 transaksi baru dan 200 transaksi berulang, pada pukul 20:10 hinga, pada pukul 20:11 hingga pukul 20:20dilakukan Hit transaksi baru dengan 250 data baru dan dikuarngi 50 transaksi baru tiap menitnya serta 250 transkasi berulang dengan ditambah 50 transaksi berulang tiap menitnya, pada pukul 20:21 hingga pukul 20:30 dilakukan percobaan kebalikan di 10 menit sebelumnya dan di pukul 20:31 hingga pukul 20:40 dilakukan percobaan *Hit* dengan data transaksi baru dan berulang secara random. Dari penjelasan di atas yang dimaksud transaksi baru adalah data yang belum ada ata tersimpan di Memcached sedangkan transaksi berulang adalah data yang sudah tersimpan di Memcached dan belum terhapus atau masih dalam rentang waktu 600 detik dari awal disimpan. Gambar 8 menunjukan grafik pengujian dengan tidak menggunakan Memcached dan menggunakan Memcached dengan berbagai skenario yang telah dijelaskan.



Gambar 6. Hasil waktu proses pemanggilan data dimana di hari pertama tidak menggunakan Memcached.



Gambar 7. Hasil penggunaan Memcached dengan menggunakan perintah Cron Job.



Gambar 8. Grafik pengujian dengan tidak menggunakan Memcached dan menggunakan Memcached.

3. HASIL DAN PEMBAHASAN

Dari pengujian yang dilakukan membuktikan hasil kajian dari Abdullah Talha Kabakus, and Resul Kara dalam jurnal "A performance evaluation of in-memory databases"[2] bahwa database NoSql dimana Memcached sebagai *key-value stores* memiliki 3 keunggulan utama, (1) memiliki *latency* yang rendah dan memiliki performa yang tinggi, (2) struktur data yang lebih fleksibel (bebas skema), dan, (3) waktu proses sejumlah besar data lebih cepat daripada sistem manajemen basis data relasional dengan mengambil keuntungan dari arsitektur yang sangat skalabel.

Di samping kelebihan yang dimiliki, terdapat kelemahan karena menjadi salah satu celah untuk masuknya serangan cyber seperti yang telah dikemukakan oleh Mishra Nivedita dkk dalam jurnal "Memcached: An experimental study of ddos attacks for the wellbeing of iot applications" [11]. Selain itu, juga akan memberi dampak yang tidak kecil bagi performa dari sistem secara keseluruhan sehingga perlu diperhatikan dan dilakukan juga pencegahan seperti yang dipublikasikan oleh Kulvinder Singh, and Ajit Singh dalam jurnal "Memcached DDoS Exploits: Operations, Vulnerabilities, Preventions and Mitigations" [5].

4. KESIMPULAN

Dalam optimalisasi sebuah sistem yang menggunakan database, selain optimalisasi database yang telah dijabarkan oleh Iskandar Daniel dkk. dalam jurnal " Database Tuning in Hospital Applications Using Table Indexing and Query Optimization"[12] dapat dilakukan optimalisasi dari aplikasinya itu sendiri yang dampaknya tidak terlihat langsung di sisi optimalisasi pemrosesan data secara langsung, tetapi meringankan kinerja server database itu sendiri serta menghindari aktivitas operasi pemrosesan data yang sebetulnya tidak diperlukan serta operasi pemrosesan data yang tidak pernah selesai (menggantung).

5. SARAN

Penulis mengahrapkan untuk penulis selanjutnya dapat mengkaji lebih jauh kelemahan dan faktor keamanan yang berdampak pada kinerja system keseluruhan baik dari sisi *system* dan *server* serta infrastrukturnya.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada keluyarga saya yang telah menciptakan suasana yang mendukung dalam pmbuatan tulisan dan penelitian ini.

DAFTAR PUSTAKA

[1] B. R. Chang, H. F. Tsai, Y. C. Tsai, C. F. Kuo, and C. C. Chen, "Integration and optimization of multiple big data processing platforms," *Engineering Computations (Swansea, Wales)*, vol. 33, no. 6, pp. 1680–1704, Aug. 2016, doi: 10.1108/EC-08-2015-0247.

[2] A. T. Kabakus and R. Kara, "A performance evaluation of in-memory databases," *Journal of King Saud University - Computer and Information Sciences*, vol. 29, no. 4, pp. 520–525, Oct. 2017, doi: 10.1016/j.jksuci.2016.06.007.

[3] A. Makris, K. Tserpes, V. Andronikou, and D. Anagnostopoulos, "A Classification of NoSQL Data Stores Based on Key Design Characteristics," in *Procedia Computer Science*, 2016, vol. 97, pp. 94–103. doi: 10.1016/j.procs.2016.08.284.

[4] R. Nishtala *et al.*, "Scaling Memcache at Facebook."

[5] K. Singh and A. Singh, "Memcached DDoS Exploits: Operations, Vulnerabilities, Preventions and Mitigations," in *Proceedings on 2018 IEEE 3rd International Conference on Computing, Communication and Security, ICCCS 2018*, Dec. 2018, pp. 171–179. doi: 10.1109/CCCS.2018.8586810.

[6] Y. Ghigoff, J. Sopena, K. Lazri, A. B. Gandi, and G. Muller, *Open access to the Proceedings of the 18th USENIX Symposium on Networked Systems Design and Implementation is sponsored by BMC: Accelerating Memcached using Safe In-kernel Caching and Pre-stack Processing BMC: Accelerating Memcached using Safe In-kernel Caching and Pre-stack Processing*. [Online]. Available: https://www.usenix.org/conference/nsdi21/presentation/ghigoff

[7] D. Kriestanto, A. Benden Arnado, J. Teknik Informatika, S. AKAKOM Yogyakarta Jl Raya Janti, and K. Yogyakarta, "IMPLEMENTASI WEBSITE PENCARIAN KOS DENGAN NOSQL," 2017. [Online]. Available: https://www.openshift.com/

[8] "38956-110714-1-PB".
[9] A. Fadli, M. I. Zulfa, A. W. Widhi Nugraha, A. Taryana, and M. S. Aliim, "Analisis Perbandingan Unjuk Kerja Database SQL dan Database NoSQL Untuk Mendukung Era Big Data," *JURNAL NASIONAL TEKNIK ELEKTRO*, vol. 9, no. 3, Nov. 2020, doi: 10.25077/jnte.v9n3.774.2020.
[10] M. Kusuma and M. Eng, "Metode-Metode Optimasi Memcached sebagai NoSQL Key-value Memory Cache," 2019.
[11] N. Mishra *et al.*, "Memcached: An experimental study of ddos attacks for the wellbeing of iot applications," *Sensors*, vol. 21, no. 23, Dec. 2021, doi: 10.3390/s21238071.
[12] D. Iskandar, M. Fachruroji, W. Adi Septyo Wibowo, A. A. Khaerani, F. Teknologi Informasi, and U. Budi Luhur, "Database Tuning in Hospital Applications Using Table Indexing and Query Optimization," 1960.