

ANALISIS DAN OPTIMASI PADA DATABASE RELASIONAL MENGUNAKAN INDEKS DAN NORMALISASI TINGKAT LANJUT: STUDI KASUS SISTEM INFORMASI AKADEMIK

ANALYSIS AND OPTIMIZATION OF RELATIONAL DATABASE USING ADVANCED INDEXES AND NORMALIZATION: A CASE STUDY OF ACADEMIC INFORMATION SYSTEMS

**Uus Firdaus^{1*}, Arinda Setyo Rini², Arjuna Aulia Ghifari³,
Ansar Abdi Dwi Saputra⁴, M. Nofal Syahwaludin⁵, Dimas Rizky Aliandy⁶**
¹⁻⁶Universitas Djuanda

***Email Correspondence: uus.firdaus@unida.ac.id**

Abstract

Implementing advanced normalization in relational databases is a fundamental standard to ensure data integrity in Academic Information Systems (SIKAD). However, normalized data structures often lead to increased query complexity and latency as the volume of student grade data expands. This study aims to analyze and optimize query performance in relational databases by applying the B-Tree Indexing technique. The research methodology employs a quantitative experimental approach, simulating workload on a dataset containing 10,000 grade history records. Testing focuses on comparing execution time before and after optimization. Results indicate that under standard table conditions without indexing, the data retrieval mechanism using a full table scan requires an average time of 0.0029 seconds. Following the application of indexing on key attributes, execution time decreased significantly to 0.0004 seconds. This represents an 84.5% increase in performance efficiency and a shift in the data access method to an index range scan. It is concluded that the combination of valid normalization design and appropriate indexing strategies is a crucial solution for handling high academic data access loads.

Keywords: *Relational Database, Normalization, Indexing, Query Optimization, Academic Information System.*

Abstrak

Penerapan normalisasi tingkat lanjut pada basis data relasional merupakan standar fundamental untuk menjamin integritas data Sistem Informasi Akademik (SIKAD). Akan tetapi, struktur data ternormalisasi seringkali berdampak pada peningkatan kompleksitas query dan latency saat volume data nilai mahasiswa membesar. Penelitian ini bertujuan untuk menganalisis dan mengoptimasi performa query pada basis data relasional melalui penerapan teknik B-Tree Indexing. Metode penelitian menggunakan pendekatan eksperimental kuantitatif dengan melakukan simulasi beban kerja pada dataset riwayat nilai sebanyak 10.000 record. Pengujian difokuskan pada perbandingan waktu eksekusi (execution time) sebelum dan sesudah optimasi. Hasil penelitian menunjukkan bahwa pada kondisi tabel standar tanpa indeks, mekanisme pencarian data menggunakan full table scan membutuhkan waktu rata-rata 0.0029 detik. Setelah penerapan indeks pada atribut kunci, waktu eksekusi menurun signifikan menjadi 0.0004 detik. Hal ini merepresentasikan peningkatan efisiensi performa sebesar 84,5% dan perubahan metode akses data menjadi index range scan. Disimpulkan bahwa kombinasi antara desain normalisasi yang valid dan strategi indexing yang tepat adalah solusi krusial untuk menangani beban akses data akademik yang tinggi.

Kata Kunci: Database Relasional, Normalisasi, Indexing, Optimasi Query, Sistem Informasi Akademik.

PENDAHULUAN

Sistem Informasi Akademik (SIKAD) merupakan infrastruktur teknologi vital di perguruan tinggi yang berfungsi mengelola data sensitif seperti biodata mahasiswa, Kartu Rencana Studi (KRS), dan riwayat nilai. Seiring dengan berjalannya waktu operasional kampus, volume data yang tersimpan akan terus mengalami pertumbuhan eksponensial. Tantangan utama yang sering muncul adalah penurunan performa sistem (*latency*), terutama saat ribuan mahasiswa mengakses fitur krusial secara bersamaan [1]. Kegagalan basis data dalam melayani permintaan (*request*) dengan cepat dapat menghambat proses administrasi akademik dan menurunkan kepuasan pengguna [2]. Oleh karena itu, strategi optimalisasi basis data menjadi aspek yang tidak bisa diabaikan dalam pengembangan perangkat lunak kampus [3].

Dalam paradigma perancangan basis data relasional (*Relational Database Management System*), standar baku yang wajib diterapkan adalah Normalisasi Tingkat Lanjut (seperti *Third Normal Form* atau BCNF). Normalisasi bertujuan untuk menjamin integritas data, mencegah anomali penyisipan (*insert anomaly*), dan menghilangkan duplikasi data (*redundansi*) yang tidak perlu, sebagaimana dijelaskan dalam konsep pencegahan *SQL Antipatterns* [4]. Namun, struktur tabel yang sangat ternormalisasi memiliki konsekuensi teknis tersendiri. Pemecahan data ke dalam banyak tabel relasional seringkali memaksa sistem untuk melakukan operasi penggabungan (*JOIN*) yang kompleks saat proses pengambilan data (*data retrieval*). Hal ini berpotensi membebani kinerja CPU dan memperlambat waktu respons *query*, khususnya jika tidak didukung oleh strategi pengindeksan yang tepat [5].

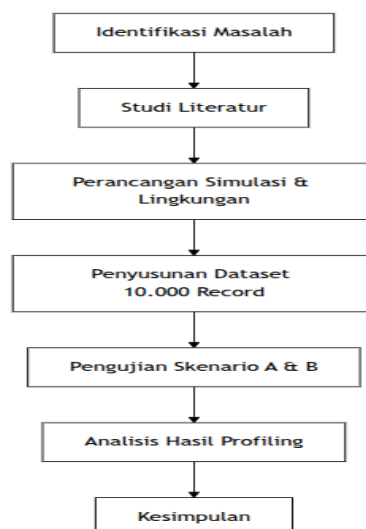
Untuk mengatasi kesenjangan antara kebutuhan integritas data (melalui normalisasi) dan kebutuhan kecepatan akses, diperlukan teknik optimasi basis data (*Database Tuning*). Salah satu metode yang terbukti efektif adalah penerapan Indeks (*Indexing*). Indeks didefinisikan sebagai struktur data tambahan—umumnya berbasis algoritma *B-Tree*—yang memungkinkan *database engine* untuk melompat langsung ke lokasi data yang dicari tanpa harus memindai seluruh isi tabel atau *full table scan* [6]. Penggunaan indeks yang tepat pada kolom pencarian utama dapat mengubah kompleksitas algoritma pencarian dari linear menjadi logaritmik, yang secara signifikan memangkas biaya I/O (*Input/Output cost*) pada server basis data [7].

Berdasarkan latar belakang tersebut, penelitian ini bertujuan untuk menganalisis efektivitas penerapan *Indexing* pada basis data relasional yang telah dirancang dengan kaidah normalisasi. Penelitian ini mengambil studi kasus spesifik pada data riwayat nilai mahasiswa dalam ekosistem Sistem Informasi Akademik. Fokus utama penelitian adalah mengukur signifikansi peningkatan waktu eksekusi *query* sebelum dan sesudah optimasi dilakukan.

METODE

1) Tahapan Penelitian

Metode yang digunakan dalam penelitian ini adalah metode eksperimental kuantitatif. Tahapan penelitian dirancang secara sistematis dimulai dari identifikasi masalah performa pada sistem akademik, studi literatur mengenai normalisasi dan *indexing*, perancangan lingkungan simulasi, pelaksanaan eksperimen, hingga analisis hasil komparasi data. Alur penelitian ini secara visual disajikan pada Gambar 1:



Gambar 1. Bagan Alur Penelitian

Pendekatan ini dipilih untuk mengukur secara empiris dampak penerapan teknik optimasi basis data terhadap kinerja waktu respons sistem.

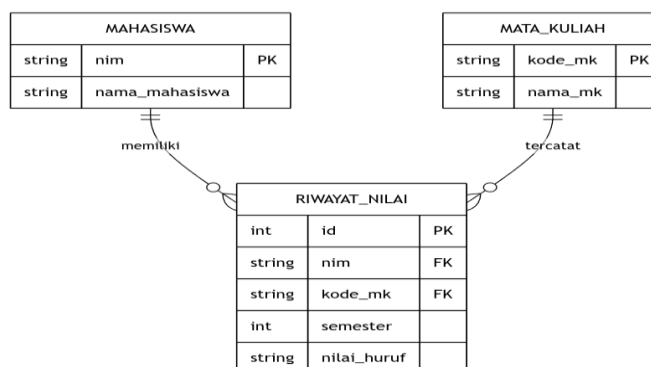
2) Instrumen Penelitian

Instrumen penelitian mencakup perangkat keras dan perangkat lunak yang digunakan untuk membangun lingkungan simulasi yang stabil:

- Perangkat Keras: Laptop dengan spesifikasi prosesor AMD Ryzen 7 dan RAM 16GB.
- Sistem Operasi: Linux Mint, dipilih karena stabilitas manajemen memori yang baik untuk server basis data.
- Database Management System (DBMS): MariaDB versi 10.4.32.
- Alat Pengukuran: Perintah *SHOW PROFILES* pada SQL digunakan untuk memantau durasi eksekusi *query* hingga satuan mikrodetik [8]

3) Perancangan dan Pengumpulan Data

Objek penelitian adalah tabel riwayat_nilai yang dirancang mengikuti prinsip Normalisasi Tingkat Lanjut (hingga 3NF). Penerapan bentuk normal ketiga ini dilakukan dengan memisahkan tabel untuk menghilangkan ketergantungan transitif pada atribut non-kunci, sekaligus memastikan atomisitas data. Rancangan struktur tabel yang telah dinormalisasi tersebut disajikan dalam skema relasional pada Gambar 2:



Gambar 2. Rancangan Skema Relasional 3NF

Struktur tabel terdiri atas atribut NIM, nama mahasiswa, mata kuliah, semester, dan nilai huruf. Data yang digunakan merupakan *dummy dataset* sebanyak 10.000 *record* yang dibangkitkan menggunakan metode *random sampling*. Jumlah tersebut merepresentasikan volume data nilai pada perguruan tinggi skala menengah [2].

4) Skenario Pengujian

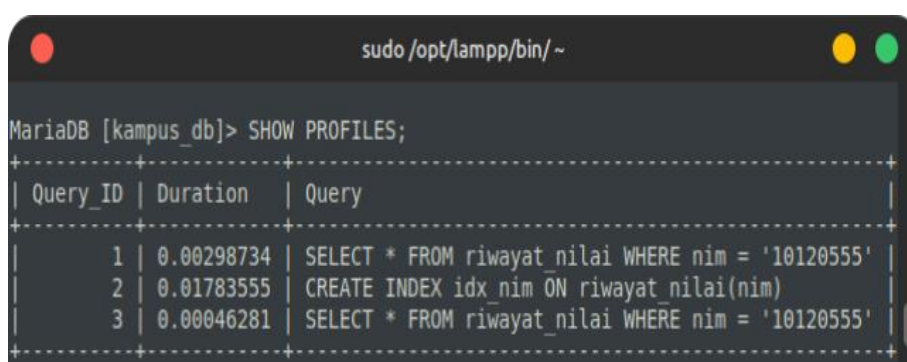
Prosedur pengujian kinerja dibagi menjadi dua skenario utama untuk mendapatkan data perbandingan (*before-after*):

- Skenario A (Tanpa Optimasi): Menjalankan *query* pencarian data nilai mahasiswa berdasarkan NIM pada tabel standar (*Heap Table*).
- Skenario B (Dengan Optimasi Indexing): Menerapkan struktur data *B-Tree Index* pada kolom NIM, kemudian menjalankan *query* yang sama.

HASIL DAN PEMBAHASAN

1) Hasil Pengukuran Waktu Eksekusi

Pengujian dilakukan dengan menjalankan perintah SQL untuk mencari rekap nilai mahasiswa berdasarkan Nomor Induk Mahasiswa (NIM) spesifik ('10120555'). Pengukuran waktu dilakukan menggunakan fitur *profiling* bawaan MariaDB.



```

sudo /opt/lampp/bin/~

MariaDB [kampus_db]> SHOW PROFILES;
+-----+-----+-----+
| Query_ID | Duration | Query |
+-----+-----+-----+
| 1 | 0.00298734 | SELECT * FROM riwayat_nilai WHERE nim = '10120555' |
| 2 | 0.01783555 | CREATE INDEX idx_nim ON riwayat_nilai(nim) |
| 3 | 0.00046281 | SELECT * FROM riwayat_nilai WHERE nim = '10120555' |
+-----+-----+-----+

```

Gambar 3. Hasil Profiling Query pada MariaDB

Berdasarkan Gambar 3, data hasil pengukuran dirangkum dalam Tabel 1 berikut:

Tabel 1. Perbandingan Performa Query

Skenario Pengujian	Metode Akses Data	Waktu Eksekusi (Detik)
Kondisi Awal (Tanpa Index)	Full Table Scan	0.00298734
Kondisi Akhir (Dengan Index)	Index Range Scan	0.00046281
Selisih Waktu	-	0.00252453
Peningkatan Performa	-	84,5 %

2) Pembahasan Analisis Kinerja

Hasil eksperimen menunjukkan perbedaan performa yang signifikan antara kedua skenario. Pada Kondisi Awal, basis data mencatat waktu eksekusi sebesar 0.0029 detik. Hal ini disebabkan oleh mekanisme *Full Table Scan*, di mana sistem dipaksa memindai seluruh 10.000 baris data satu per satu untuk mencocokkan NIM yang diminta. Dalam teori basis data, kompleksitas algoritma ini adalah $O(n)$, yang berarti waktu pencarian akan meningkat secara linear seiring bertambahnya jumlah mahasiswa.

Sebaliknya, pada Kondisi Akhir setelah penerapan perintah *CREATE INDEX*, waktu eksekusi turun drastis menjadi 0.0004 detik. Terjadi peningkatan efisiensi kecepatan sebesar 84,5%. Percepatan ini terjadi karena indeks *B-Tree* mengubah mekanisme pencarian menjadi *Index Range Scan* dengan kompleksitas waktu logaritmik $O(\log n)$. Sistem tidak lagi mencari data secara berurutan, melainkan menelusuri struktur pohon indeks yang langsung mengarah ke lokasi penyimpanan data NIM '10120555'.

Temuan ini mengonfirmasi teori [7] bahwa *Indexing* adalah solusi paling efektif untuk menangani beban *query* baca (*read-heavy*) pada sistem informasi. Dalam konteks Sistem Informasi Akademik, penerapan indeks pada atribut kunci seperti NIM terbukti mampu menjaga responsivitas sistem.

KESIMPULAN

Berdasarkan hasil analisis dan pengujian yang telah dilakukan pada simulasi Sistem Informasi Akademik, dapat disimpulkan bahwa penerapan teknik optimasi basis data sangat krusial pada tabel relasional yang memiliki volume data besar. Penerapan Normalisasi Tingkat Lanjut terbukti efektif dalam menjaga konsistensi data, namun berdampak pada peningkatan latensi pencarian. Solusi yang ditawarkan melalui penerapan *B-Tree Indexing* pada atribut kunci (NIM) terbukti mampu meningkatkan performa secara signifikan. Hasil eksperimen menunjukkan penurunan waktu eksekusi *query* sebesar 84,5%, yaitu dari 0.0029 detik (tanpa indeks) menjadi 0.0004 detik (dengan indeks).

DAFTAR PUSTAKA

- [1] S. Samidi, F. Fadly, Y. Virmansyah, R. Y. Suladi, and A. B. Lesmana, "Optimasi Database dengan Metode Index dan Partisi Tabel Database PostgreSQL pada Aplikasi E-Commerce: Studi pada Aplikasi Tokopintar," *Jurnal Pendidikan Tambusai*, vol. 6, no. 1, pp. 2094–2102, 2022.
- [2] E. Budiman, M. Fadli, D. Kurniawan, and E. R. Susanto, "Tinjauan Literatur dengan Pendekatan Systematic Literature Review untuk Optimasi Kueri dalam Basis Data," *Jurnal Ilmiah Teknik dan Ilmu Komputer*, vol. 4, no. 2, pp. 82–91, May 2025.
- [3] W. G. Mokodaser, M. Dwijayanti, and S. Samidi, "Implementasi Metode Indexing dan Penggunaan Subquery untuk Optimalisasi Database Rawat Jalan Rumah Sakit Menggunakan MySQL," *Cogito Smart Journal*, vol. 8, no. 2, pp. 335–345, Dec. 2022.
- [4] B. Karwin, *SQL Antipatterns: Avoiding the Pitfalls of Database Programming*. The Pragmatic Bookshelf, 2010.
- [5] E. Firmansyah, H. Firdaus, and S. Samidi, "Optimasi Performa Query Subsidi Debitur dengan Index and Table Partition, Subquery and Indexing, dan ParallelQueryExecution," *Jurnal Pendidikan dan Teknologi Indonesia (JPTI)*, vol. 5,

-
- no. 6, pp. 1769–1785, Jun. 2025.
- [6] D. Shasha and P. Bonnet, *Database Tuning: Principles, Experiments, and Troubleshooting Techniques*, 1st Edition. Morgan Kaufmann, 2002.
- [7] M. Winand, *SQL Performance Explained: Everything Developers Need to Know about SQL Performance*. 2012.
- [8] T. Connolly and C. Begg, *Database Systems: A Practical Approach to Design, Implementation, and Management*, 6th Edition. Pearson, 2015.