Jurnal ICT : Information Communication & Technology
Vol. 23, N0.1, Juli 2023, pp. 325-330
p-ISSN: 2302-0261, e-ISSN: 2303-3363, DOI:

Reverse Engineering Pada Basis Data Relasional Untuk Optimasi Performansi Query

Fajri Rakhmat Umbara[1*], Herdi Ashaury[2*], Puspita Nurul Sabrina[3*]
[1,2,3]Program Studi Teknik Informatika, Universitas Jenderal Achmad Yani Cimahi, Indonesia
Email: [1]fajri.rakhmat@lecture.unjani.ac.id, [2]herdi.ashaury@lecture.unjani.ac.id
,[3]puspita.sabrina@lecture.unjani.ac.id

INFORMASI ARTIKEL

Histori artikel:

Naskah masuk, 24 Asgustus 2023
Direvisi, 1 September 2023
Diiterima, 5 September 2023

Kata Kunci:

Basis data
Optimasi query
Reverse engineering
RDBMS

ABSTRAK

Abstract- Database implementation has been begun since 2nd generation of computer hardware has been found around 1960. Today, database has created and for variant purpose, along using modern technologies, database become more important than the past. The problem that can be solved using database is transactional data. Using relational scheme as a concept, Relational Database Management System (RDBMS) become more popular. Problems often encountered is the implementation of a relational database that is not good, especially in terms of relationships, cardinality, and the structure in general so if done a query process, the results are not optimal. This research evaluating benchmarks and profiling the results of relational database queries before redesigning and after redesigning using reverse engineering techniques. The results obtained are an average performance increase of about 82.2442% after improving the design structure and table implementation for the same query results in the Mysql RDBMS application.

Abstrak- Pemanfaatan basis data sudah dimulai sejak komputer generasi ke 2 diciptakan, yaitu sekitar tahun 1960. Semakin bertambahnya tahun, penggunaan basis data sudah memiliki perkembangan dan didukung dengan teknologi yang sudah canggih pada masa sekarang. Basis data yang sering digunakan dalam masalah transaksi adalah basis data dengan skema relasional atau sering disebut dengan *Relational Database* dan diimplementasikan dengan baik oleh aplikasi *Relational Database Management System* (RDBMS). Permasalahan yang sering dihadapi adalah pengimplementasian basis data relasional yang tidak baik, terutama dari sisi relasi, kardinalitas, dan strukturnya secara umum sehingga apabila dilakukan suatu proses *query*, maka hasilnya tidak optimal. Hal ini terjadi karena akses dari basis data yang secara paralel dapat diakses oleh banyak pengguna dalam satu waktu. Pada penelitian ini dilakukan evaluasi secara *benchmark* hasil *query* basis data relasional sebelum dilakukan desain ulang dan setelah dilakukan desain ulang menggunakan teknik *reverse engineering*. Hasil yang didapat adalah peningkatan performansi rata-rata sekitar 82.2442% setelah dilakukannya perbaikan struktur desain dan implementasi tabel untuk hasil *query* yang sama pada aplikasi RDBMS Mysql.

Penulis Korespondensi:

Fajri Rakhmat Umbara
Program Studi Teknik Informatika,
Universitas Jenderal Achmad yani
Jl. Terusan Jenderal Sudirman No 1, Cimahi, Indonesia
Email: fajri.rakhmat@lecture.unjani.ac.id

1. Pendahuluan

Basis data atau *Database* adalah suatu bentuk fisik dari tempat penyimpanan data secara digital serta banyak dipakai dan di implementasikan di berbagai aplikai, mulai dari aplikasi stand-alone, mobile, dan web-based. Basis data juga banyak diimplementasikan di berbagai arsitektur perangkat lunak seperti *multilayered architecture*, *data-centered architecture*, hingga arsitektur kekinian yaitu *microservices architecture* [1]. Basis data yang memiliki ketangguhan dalam memproses suatu transaksi adalah basis yang berbasis relasional, artinya memiliki bentuk *output* berupa tabel dan memiliki relasi yang di implementasikan dalam bentuk *Primary Key* dan *Foreign key* [1]. Konsep basis data relasional diimplementasikan dengan baik oleh *Structure Query Language* (SQL), sebagai Bahasa yang digunakan untuk mengeksekusi perintah yang diinginkan oleh pengguna atau oleh aplikasi yang menggunakannya. Sempat muncul bahasa – bahasa pemrosesan dan pengimplementasian dari basis data relasional lainnya, namun belum ada yang sebaik bahasa SQL. Bahasa SQL diadopsi sebagai bahasa baku disebagian besar aplikasi perangkat lunak *Relational Database Management System* (RDBMS) dan yang paling popular digunakan, selain gratis tetapi juga powerfull adalah Mysql. Mysql sendiri saat ini sudah berganti nama menjadi MariaDB, walaupun sekarang sudah banyak versi dan berganti nama, namun komunitas dari RDBMS ini tetap ada sampai saat ini [1].

Masalah umum yang terjadi dalam melakukan implementasi dengan konsep basis data relasional adalah kesalahan dari fase desain yang dilakukan oleh *software designer* [1]. Mereka seringkali mengabaikan elemen – elemen penting dalam merancang suatu basis data relasional, seperti diantaranya adalah masalah relasi, kardinalitas, penamaan tabel, dan masalah normalisasi [1]. Normalisasi adalah suatu ketentuan yang harus dimiliki oleh basis data relasional agar menjamin bahwa perintah – perintah dasar dari SQL itu sendiri, seperti *Data Definition Language* (DDL), *Data Manipulation Language* (DML), *Data Control Language* (DCL), dan *Transaction Control Language* (TCL) berjalan dengan baik, namun yang paling sering disoroti dalam hal memaksimalkan query adalah di perintah DML [1].

Beberapa penelitian pernah membahas mengenai pemaksimalan kinerja query, mulai dari melakukan pengaturan (*tunning*) secara *vertical* (*hardware*) maupun secara *horizontal* (*software*), dan bahkan ada yang melakukan keduanya. Setiap hasil penelitian pun berbeda hasil yang didapatkan., namun secara umum perbedaan yang ada adalah dari spesifikasi perangkat keras (*hardware*) yang digunakan, struktur table, relasi dan dari sisi optimasi penulisan *query* itu sendiri [2] .

Pada penelitian ini, penulis juga meneliti mengenai masalah optimasi *query*, selain itu peneliti juga menyoroti hasil performansi basis data yang dilihat dari evaluasi RDBMS yang digunakan dan memiliki permasalahan seperti pada variabel *opened tables*, *select join full*, *select range check*, *slow queries*, dan *table lock waited* yang teridentifikasi memiliki nilai yang mempengaruhi performansi dari sisi struktur, skema, dan relasi basis data serta penggunaan *query* yang buruk. Pada penelitian ini juga akan dilakukan desain ulang terlebih dahulu untuk menanggulangi masalah struktur dan skema yang buruk berdasarkan hasil performansi yang ditunjukkan oleh RDBMS tadi. Desain ulang disini berarti melakukan dan mengimplementasikan teknik *reverse engineering* terhadap implementasi basis data yang ada dan mendapatkan skema tabelnya, lalu di desain kembali dengan memperhatikan kaidah dan prinsip dari relasi, kardinalitas, dan normalisasi, setelah itu mengimplementasikan ulang ke RDBMS dan melihat hasilnya dengan evaluasi benchmark query saat sebelum dilakukan desain ulang dan sesudah dilakukan desain ulang menggunakan teknik *reverse engineering* tersebut.

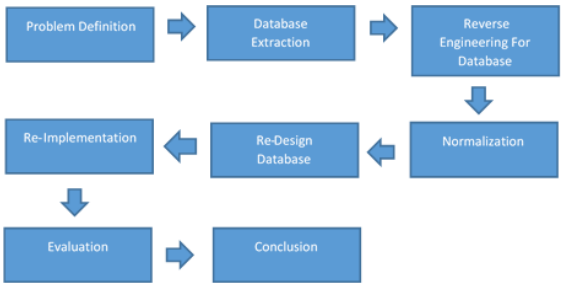
2. Penelitian terdahulu

Permasalahan dalam implementasi dan optimasi basis data relasional memang kerap kali terjadi, dan terdapat beberapa penelitian yang melakukan itu. Penelitian yang mengimplementasikan SPARQL dilakukan untuk mengoptimasi kinerja query pada mesin proses Hive dimana dari 14 uji *query* dihasilkan *loading time* yang cukup cepat untuk performansinya dan ini diimplementasikan dalam lingkungan berbasis web [3]. Penelitian yang mengubah skema umum basis data juga pernah dilakukan dengan mengubahnya ke *domain* yang spesifik dan menghasilkan suatu bentuk data yang sesuai dengan apa yang ingin dihasilkan sehingga pemrosesan query akan sesuai dan unik [4]. Lalu terdapat juga perubahan struktur aplikasi dan data

dari *monolith* ke *microservices* untuk menghasilkan sumber data yang sesuai dengan service yang ada sehingga menjaga data tetap memiliki kualitas dan performa yang baik [5]. Tidak hanya itu, optimalisasi dari penggunaan *shared memory* pernah dilakukan, menggunakan teknik *direct-memory access* (RDMA) untuk membuka dan mengeksekusi *query* secara cepat menghasilkan performansi yang lebih baik daripada konfigurasi memory standar yang diatur oleh sistem operasi [6]. Optimasi perangkat keras juga pernah dilakukan, dengan menggunakan memory dari GPU dan melakukan *benchmark* antara prosesing menggunakan CPU dan GPU menghasilkan *multi-core* GPU menghasilkan pemrosesan basis data lebih cepat daripada *multi-core* CPU dan menggunakan daya yang lebih hemat juga [7]. Beberapa penelitian juga secara bersamaan mengangkat masalah skema basis data yang terlalu kompleks dapat menyebabkan permasalahan performansi yang berbeda beda juga [2], [8-9], [10-12].

3. Metode Penelitian

Dari beberapa penelitian terdahulu, terdapat beberapa tahapan penelitian yang diadopsi dalam penelitian ini, namun dikembangkan sesuai dengan tujuan penelitian, yaitu untuk mengevaluasi performansi *query*.



Gambar 1. Metode Penelitian

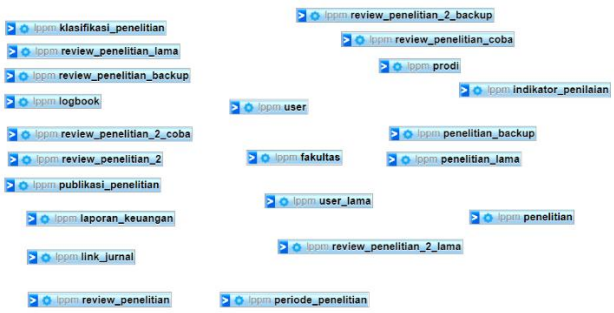
3.1 Definisi Masalah (*Problem Definition*)

Permasalahan performansi ditunjukkan pada tabel dibawah ini, dimana yang diamati adalah hasil yang mempengaruhi skema dan struktur basis data serta penggunaan *query* yang buruk, ditandakan dengan nilai dan deskripsi yang ada. Sampel basis data yang digunakan adalah basis data dari Lembaga Penelitian dan Pengabdian masyarakat (LPPM) di salah satu perguruan tinggi di Indonesia.

Tabel 1. Hasil Evaluasi Performansi RDBMS LPPM

Variable	Value	Description
Opened tables	14.9 M	*The number of tables that have been opened. If opened tables is big, your table cache value is probably too small.*
Select full join	1.2 M	*The number of joins without keys that do not uses indexes. If the value is not 0, you should carefully check the indexes of your table.*
Select range check	56.6 k	*The number of joins without keys that check the key usage after each row. If this is not 0, you should carefully check the indexes of tour tables.*
Slow queries	2.1 k	*The number of queries that have taken more than long_query_time seconds.*
Table locks waited	470.3 k	*The number of times that a table lock could not be aquired immediately and a wait has needed. If this is high, and you have performance problems, you should first optimized your queries, and then either split yout table or tables or use replication.*

Selain itu, dari struktur yang diamati, terdapat beberapa kesalahan struktur basis data yang tidak sesuai dengan kaidah pembangunan yang seharusnya, relasi yang tidak ada, dan kardinalitas yang tidak terimplementasikan dengan baik. Hal ini dapat mengakibatkan tidak optimalnya akses ke basis data dari sisi waktu eksekusi.



Gambar 2. Desain Skema Tabel Basis data LPPM yang kurang baik, tanpa implementasi relasi dan kardinalitas.

3.2 Ekstraksi Basis Data (*Database Extraction*)

Peneliti melakukan ekstraksi basis data diatas kedalam *localhost* untuk digunakan secara *offline* dan terpisah dari database aktif, jumlah tabel yang di

ekstraksi berjumlah 22 tabel. Namun disini, data asli dihapus untuk menjamin privasi data dan hanya diambil struktur tabelnya saja.

3.3 Reverse Engineering for Database

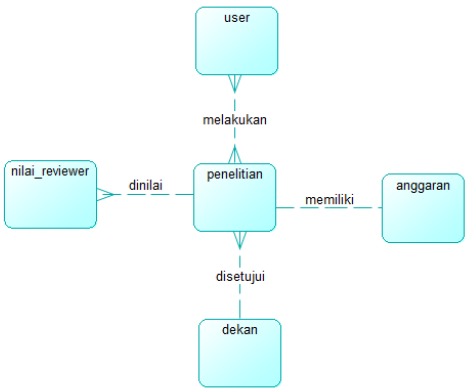
Teknik *Reverse Engineering* diimplementasikan ke dalam basis data relasional, yaitu dari tahapan ekstraksi basis data tadi, dilakukan perubahan ke bentuk desain struktur tabel. Salah satu dari 22 tabel yang telah di ekstraksi diubah bentuknya kedalam bentuk desain struktur table seperti yang ditunjukkan pada tabel dibawah ini.

Tabel 2. Salah Satu Desain Struktur tabel (Tabel Penelitian)

Nomor	Atribut	Tipe Data
1	id_penelitian	Number(11)
2	Tahun	Text(5)
3	Periode	Number(2)
4	Judul	Text
……	……	……
34	nilai_reviewer_1	Number(4)
35	nilai_reviewer_2	Number(4)
36	nid_dekan	Text(20)
37	nama_dekan	Text(250)

3.4 Normalisasi (*Normalization*)

Pada tabel penelitian yang ditunjukan oleh Tabel 2, dilakukan analisa dan implementasi Normalisasi, mulai dari *First Normal Form* (1NF), *Second Normal Form* (2NF) dan *Third Normal Form* (3NF), sehingga tabel penelitian ini terpecah menjadi 5 buah tabel yang saling berelasi dan sudah memenuhi bentuk normal ketiga (3NF). Untuk keseluruhan dari 22 tabel yang di ekstraksi dilakukan hal yang sama sehingga dihasilkan 31 tabel dan juga sudah memenuhi bentuk normal ketiga. Berikut contoh hasil Normalisasi yang dilakukan pada tabel penelitian, digambarkan dalam bentuk struktur entitas dan relasi.



Gambar 3. Desain Hasil Normalisasi Tabel Penelitian Dengan Syarat 3NF Terpenuhi

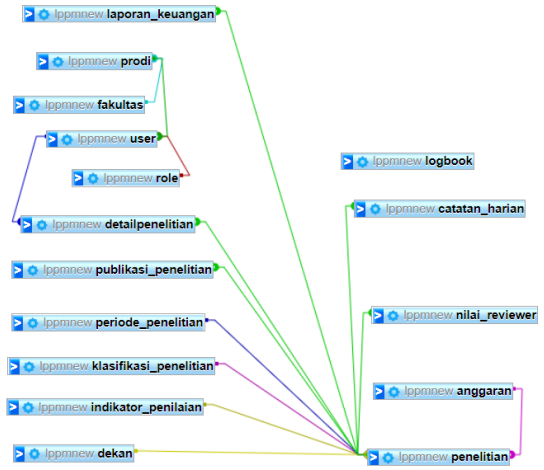
3.5 Desain Ulang basis Data (*Re-Design Database*)

Dari 31 tabel yang dihasilkan akan di desain ulang, kriteria yang digunakan untuk mendesain ulang adalah memperbaiki relasi dan kardinalitas relasi, menghilangkan duplikasi atribut dan duplikasi tabel, dan menyesuaikan type data dan panjang data. Terdapat banyak sekali duplikasi entiras yang terbentuk dari desain lama, seperti untuk entitas penelitian, tabel duplikasinya antara lain : penelitian_lama dan penelitian_backup. Entitas lain yang terpantau terjadi duplikasi adalah review_penelitian. Terdapat setidaknya 7 tabel yang duplikasi : review_penelitian, review_penelitian_lama, review_penelitian_backup, review_penelitian_2, review_penelitian_2_coba, review_penelitian_2_lama, dan review_penelitian_2_backup.

Dari pengurangan duplikasi tersebut, didapatlah 16 tabel yang sudah bersih dan sudah memenuhi semua syarat yang telah disebutkan sebelumnya.

3.6 Implementasi Ulang (*Re-Implementation*)

Hasil dari desain ulang databse diatas diimplementasikan menggunakan *Relational Database Management System* (RDBMS) Mysql. Relasi dan kardinalitas relasi serta tipe data dan panjang data menyesuaikan dengan implementasi. Pada tahapan ini ditambahkan juga relasinya secara fisik pada DBMS. Gambar dibawah ini menunjukan implementasi yang dilakukan untuk kemudian dilakukan evaluasi.



Gambar 4. Implementasi Basis Data LPPM Baru

3.7 Evaluasi dan Kesimpulan

Pada tahapan ini dilakukan pengujian secara *benchmark* menggunakan suatu *query* untuk mendapatkan hasil yang sama. *Benchmark* dilakukan dengan 2 kondisi, yaitu kondisi pertama adalah

proses *query* dilakukan dengan struktur basis data pada Gambar 2 atau kondisi *existing* dan kondisi yang kedua adlah proses *query* dilakukan dengan struktur basis data seperti pada Gambar 4 atau yang sesudah dilakukan desain ulang dengan teknik *reverse engineering*.

Pengujian yang dilakukan akan menggunakan *local environtment* dengan spesifikasi berikut :

- Processor intel i5 Gen 7
- Ram 8 GB DDR 5
- SSD 2.5 sata 512 GB
- Localhost XAMPP ver 7.2.10

4. Hasil Penelitian

Dalam penelitian ini, akan membandingkan kinerja *query* perintah *select* dalam satuan waktu *second* (s) sebelum dan sesudah proses *reverse engineering*. Dilakukan percobaan dengan melakukan proses *query* untuk informasi yang diinginkan. Data yang digunakan untuk tabel-tabel terkait adalah data sistesis, bukan data sebenarnya.

Tabel 3. Hasil Pengujian

No	Informasi	*Record*	Sebelum	Sesudah
1	Menampilkan semua penelitian yang ada	1327	0.019276	0.006152
2	Mencari peneliti berdasarkan nama	1	0.007654	0.001353
3	Menampilkan penelitian yang belum dilaporkan pada tahun 2020	11	0.005633	0.001134
4	Menampilkan penelitian yang dilakukan oleh masing – masing penulis	922	0.028553	0.003765
5	Menampilkan peneliti yang paling banyak meneliti, berikut penelitiannya	32	0.006654	0.000733
6	Menampilkan fakultas mana yang penelitiannya paling banyak dengan pengurutan *ascending*	10	0.006987	0.000881

Dari 6 pengujian diatas, dihasilkan selisih kecepatan eksekusi sebelum dan sesudah serta persentase kenaikan performansi pada tabel dibawah ini, kemudian dihitung rata-rata persentase kenaikan performansinya.

Tabel 4. Hasil Selisih Waktu dan Persentase Performansi

No	Selisih waktu (s)	Persentase(%)
1	0.013124	68.08466
2	0.006301	82.32297
3	0.004499	79.86863
4	0.024788	86.814
5	0.005921	88.98407
6	0.006106	87.39087
	Rata - rata	**82.2442**

5. Kesimpulan dan Saran

Optimasi *query* yang dilakukan menghasilkan performansi yang lebih baik, ini dapat terlihat dari hasil pengujian pada Tabel 3. Bila dihitung secara keseluruhan, dari 6 pengujian tersebut maka terjadi peningkatan kecepatan eksekusi dalam menampilkan informasi rata – rata keseluruhan sebesar 82.2442%. Hasil ini diharapkan dapat memperbaiki semua kendala yang ada pada Tabel 1, terutama untuk variabel *slow queries*.

Teknik *reverse engineering* sangat baik diterapkan apabila pada suatu aplikasi tidak terdapat dokumen yang memadai untuk memetakan kejadian yang *existing*. Sedangkan untuk basis data dengan *scalabilities* yang lebih besar, dapat menggunakan teknik *refactoring* untuk melakukan optimasi *query*.

Ucapan Terima kasih

Penulis mengucapkan Terima Kasih kepada Program Studi Informatika, Universitas Jenderal Achmad Yani yang telah membantu penulis dalam menyelesaikan penelitian ini baik itu dari teknis maupun dari pembiayaan.

Daftar Pustaka

[1] David M. Kroenke and D. J. Auer, *Database Processing, Fundamentals, Designing, and Implementations*, vol. 1, no. 2. 2015.

[2] T. Taipalus, "The effects of database complexity on SQL query formulation," *J. Syst. Softw.*, vol. 165, 2020, doi: 10.1016/j.jss.2020.110576.

[3] M. Banane and A. Belangour, "New approach based on model driven engineering for processing complex SPARQL queries on hive," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 4, pp. 604–609, 2019, doi: 10.14569/ijacsa.2019.0100474.

[4] S. Kordić, S. Ristić, M. Čeliković, and V. Dimitrieski, "Reverse Engineering of a Generic Relational Database Schema into a Domain-Specific Data Model," no. Dm, pp. 19–28, 2015.

[5] E. Djogic, S. Ribic, and D. Donko, "Monolithic to microservices redesign of event driven integration platform," *2018 41st Int. Conv. Inf. Commun. Technol. Electron. Microelectron. MIPRO 2018 - Proc.*, pp. 1411–1414, 2018, doi: 10.23919/MIPRO.2018.8400254.
[6] P. Fent, A. Van Renen, A. Kipf, V. Leis, T. Neumann, and A. Kemper, "Low-latency communication for fast DBMS Using RDMA and shared memory," *Proc. - Int. Conf. Data Eng.*, vol. 2020-April, pp. 1477–1488, 2020, doi: 10.1109/ICDE48307.2020.00131.
[7] E. Sitaridi, "Hardware acceleration of database analytics," *Proc. - Int. Conf. Data Eng.*, vol. 25, no. 5, p. 1616, 2017, doi: 10.1109/ICDE.2017.239.
[8] S. Link and H. Prade, "Relational database schema design for uncertain data," *Inf. Syst.*, vol. 84, pp. 88–110, 2019, doi: 10.1016/j.is.2019.04.003.
[9] H. Sneed and C. Verhoef, "Re-implementing a legacy system," *J. Syst. Softw.*, vol. 155, pp. 162–184, 2019, doi: 10.1016/j.jss.2019.05.012.
[10] S. Sbai, M. Reda Chbihi Louhdi, H. Behja, E. Moukhtar Zemmouri, and C. Rabab, "Using Reverse Engineering for Building Ontologies with Deeper Taxonomies from Relational Databases," *J. Softw.*, vol. 14, no. 3, pp. 138–145, 2019, doi: 10.17706/jsw.14.3.138-145.
[11] N. Iftekhar, M. R. Warsi, S. Zafar, S. Khan, and S. S. Biswas, "Reverse Engineering of Relational Database Schema to UML Model," *Proc. - 2019 Int. Conf. Electr. Electron. Comput. Eng. UPCON 2019*, pp. 1–6, 2019, doi: 10.1109/UPCON47278.2019.8980043.
[12] C. Raibulet, F. Arcelli Fontana, and M. Zanoni, "Model-driven reverse engineering approaches: A systematic literature review," *IEEE Access*, vol. 5, no. c, pp. 14516–14542, 2017, doi: 10.1109/ACCESS.2017.2733518.