Perancangan dan Implementasi *Query Data Extractor* Berbasis *Remote Method-Invocation*

Eddy Maryanto[1], Teguh Cahyono[2], Nur Chasanah[3]

***Abstract*— Distributed databases have been a research subject that attracted many researchers to study. Distributed databases have many advantages, such as reliability and data availability. There are still many problems in this subject area that need to be solved through researches. In this paper, query data extractor, a subsystem of distributed database system, is designed and implemented. Query data extractor reside at the heart of the distributed database system. The main task of the query data extractor is to extract query data issued by applications sitting at a site to result data needed by optimizer subsystem in fragmentation and allocation processes.**

***Intisari*— Basis data terdistribusi merupakan salah satu subjek penelitian yang menarik minat banyak peneliti. Basis data terdistribusi memiliki beberapa kelebihan, antara lain reliabilitas dan availabilitas data. Masih banyak masalah yang berkaitan dengan basis data teristribusi yang dapat diteliti dan diselesaikan Pada makalah ini dilakukan desain dan implementasi *query data extractor*. *Query data extractor* merupakan subsistem penting dari sebuah sistem basis data terdistribusi. Subsistem ini mengekstrak *query* yang diisukan oleh aplikasi dari sebuah *node* untuk menghasilkan data yang diperlukan untuk pengambilan keputusan yang optimal pada proses fragmentasi dan alokasi.**

***Kata Kunci*— basis data terdistribusi, *query data extractor*, fragmentasi, alokasi.**

I. PENDAHULUAN

Teknologi basis data telah mengubah paradigma pemrosesan data. Pada paradigma awal, setiap aplikasi mendefinisikan, memelihara, dan memiliki datanya sendiri-sendiri. Setelah dikembangkannya teknologi basis data, pendefinisian dan pemeliharaan data dilakukan secara terpusat dan aplikasi-aplikasi berbagi data yang ada (*data sharing*). Adanya teknologi basis data juga berdampak pada independensi data. Aplikasi-aplikasi tidak akan terpengaruh oleh adanya perubahan fisik maupun logika pada struktur data yang ada dan sebaliknya.

Sejalan dengan adanya kemajuan dalam teknologi jaringan komputer, teknologi sistem basis data telah dikembangkan ke arah teknologi sistem basis data terdistribusi atau *distributed database system* (DDBS) yang teknologi ini merupakan perpaduan dari dua teknologi, yaitu teknologi basis data dan teknologi jaringan komputer. Dengan adanya teknologi sistem basis data terdistribusi, paradigma pengelolaan data telah mengalami pergeseran dari pengelolaan yang terpusat menjadi pengelolaan data yang terintegrasi (*integrated data management*) [1].

Sistem basis data terdistribusi menjanjikan kinerja yang lebih baik dibandingkan sistem basis data terpusat. Peningkatan kinerja meliputi reliabilitas sistem yang lebih tinggi, eksekusi *query* secara paralel, dan kemudahan dalam ekspansi sistem. Di samping menjanjikan kinerja yang lebih baik, basis data terdistribusi memunculkan beberapa masalah yang sampai saat ini belum mempunyai solusi yang memuaskan dan perlu pengkajian lebih lanjut. Permasalahan-permasalahan tersebut antara lain proses fragmentasi, replikasi, dan alokasi pada suatu infrastruktur jaringan komputer dengan topologi dan karakteristik tertentu [2].

Permasalahan lainnya yang juga perlu dikaji adalah masalah yang terkait dengan impelementasi dari konsep atau model yang terkait dengan sistem basis data terdistribusi, seperti mengimplementasi server *data query*, pemrosesan *joint-query*, dan *caching* hasil *query intermediate* maupun final yang efisien. Pada makalah ini dikaji permasalahan membuat implementasi server *data query* yang efisien berbasis Java *Remote-Method Invocation* (RMI). RMI merupakan sebuah teknologi pemanggilan metode secara *remote* yang sangat berguna untuk implementasi sistem terdistribusi [3]. Penelitian ini bertujuan untuk: membuat desain arsitektural dari *query data extractor* untuk sebuah sistem basis data terdistribusi dengan menggunakan bahasa pemrograman Java; mengimplementasikan *query data extractor* berdasarkan desain yang sudah dibuat dengan berbasiskan teknologi RMI; dan melakukan pengujian terhadap subsistem *query data extractor* hasil implementasi.

Pada sistem terdistribusi, seringkali muncul kebutuhan akan pemanggilan metode dari objek yang ada pada komputer lain yang disebut juga dengan *remote object*. Pemanggilan metode dari objek yang ada pada komputer lain (dengan platform yang sama atau berbeda) dapat dilakukan dengan menggunakan RMI. RMI merupakan fitur utama pada bahasa pemrograman Java sejak bahasa pemrograman ini dirilis. Dengan menggunakan RMI, *remote object* menjadi transparan bagi pemrogram Java. Apabila referensi untuk *remote object* sudah diperoleh, maka pemanggilan metode dari *remote object* dapat dilakukan dengan cara yang sama seperti pemanggilan metode pada objek lokal. Struktur dari sebuah RMI disajikan pada Gbr. 1.

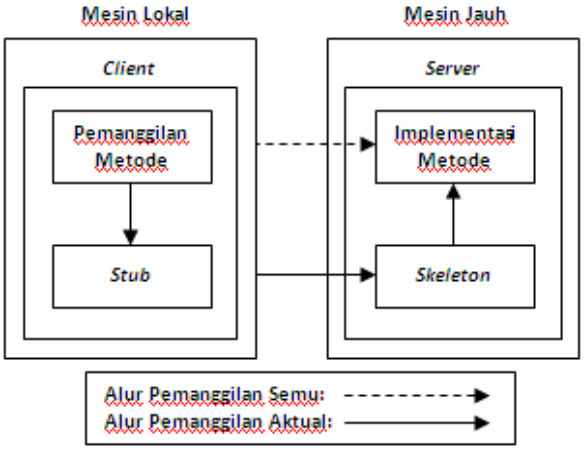
II. METODOLOGI

Penelitian dalam makalah ini dilaksanakan selama enam bulan. Metode yang digunakan adalah *action research*. Untuk

[1] *Dosen, Jurusan Teknik Informatika Fakultas Teknik Universitas Jenderal Soedirman Jl. Mayjen Sungkono KM 05 Purbalingga 53371 INDONESIA (tlp: 0281-6596700; fax: 0281-6596801; e-mail: eddy_maryanto@unsoed.ac.id)*

[2,3] *Dosen, Jurusan Teknik Informatika Fakultas Teknik Universitas Jenderal Soedirman Jl. Mayjen Sungkono KM 05 Purbalingga 55281 INDONESIA (telp: 0281-6596700; fax: 0281-6596801; e-mail: teguhsokaraja@yahoo.com, nurchasanah.ftunsoed@gmail.com*

menyelesaikan masalah yang dihadapi, langkah pertama yang dilakukan adalah membuat desain sistem basis data terdistribusi secara keseluruhan, lengkap dengan subsistem yang dibutuhkan. Selanjutnya, ditentukan spesifikasi dari salah satu subsistem, yaitu subsistem *query data server*. Berdasarkan spesifikasi yang sudah ditentukan, dibuatlah rancangan lebih detail untuk subsistem *query data server* tersebut. Hasil perancangan disajikan dalam bentuk diagram blok. Pada tahapan selanjutnya, dilakukan implementasi dari rancangan *query data server* yang sudah dibuat dan kemudian dilanjutkan dengan pengujian fungsionalitasnya. Metode pengembangan sistem yang digunakan adalah metode *Rapid Applicattion Development* (RAD). Jenis pengujian yang digunakan yaitu *system testing*, yang didahului dengan proses verifikasi dan validasi terhadap sistem yang sudah dibuat.



Gbr. 1 *Remote-Method Invocation* [3].

III. HASIL DAN PEMBAHASAN

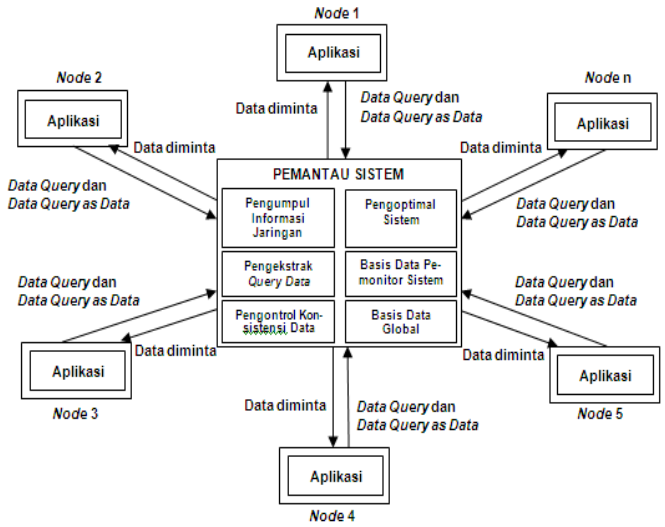
A. Desain Arsitektur Sistem Basis Data Terditribusi

Proses penciptaan sebuah sistem terdistribusi melibatkan dua proses utama, yaitu fragmentasi dan alokasi. Proses fragmentasi membutuhkan informasi yang berkaitan dengan aplikasi, yaitu *query*, dan informasi yang berkaitan dengan basis data, yaitu skema basis data. Sedangkan proses alokasi memerlukan informasi yang berkaitan dengan aplikasi, basis data, sistem komputer setiap *node*, dan jaringan komunikasi data. Semua informasi yang dibutuhkan oleh proses fragmentasi dan alokasi, seiring dengan perjalanan waktu, tentunya dapat berubah, sehingga tingkat *optimality* sebuah sistem terdistribusi pun dapat menurun. Berdasarkan fakta ini, maka dibutuhkan sistem terdistribusi yang adaptif.

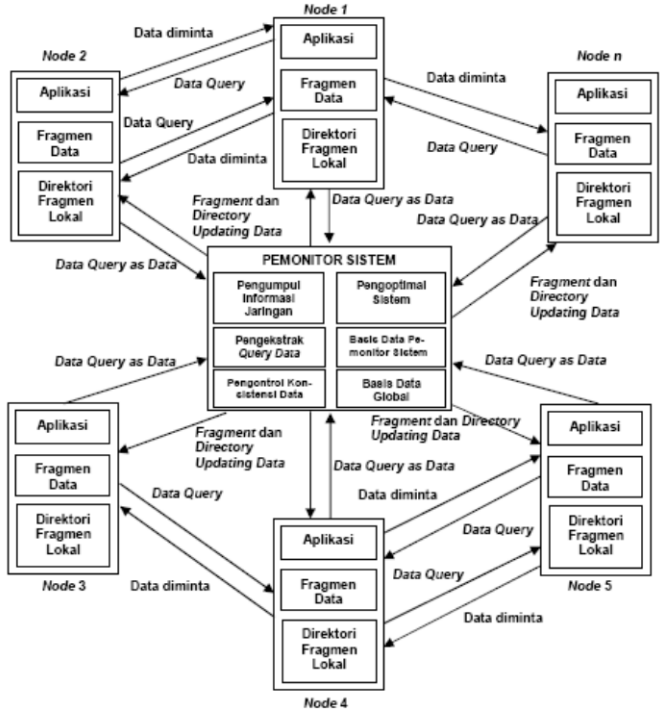
Jadi dua proses yang terkait dengan penciptaan basis data terdistribusi, yaitu fragmentasi dan alokasi, membutuhkan beberapa informasi yang salah satunya diperoleh dari hasil ekstraksi *query* yang diisukan oleh aplikasi/*node*. Proses fragmentasi dan alokasi dilakukan dengan tujuan untuk meningkatkan kinerja sistem, yaitu waktu respons. Untuk menentukan pola fragmentasi dan alokasi pada sistem basis data yang adaptif, dibutuhkan informasi yang disimpan dalam *log file* sistem. Dinyatakan dalam [4] bahwa sistem *log file* dari sebuah sistem basis data yang adaptif merupakan sumber informasi. Informasi yang tersimpan dalam sistem *log file* ini sangat dibutuhkan dalam melakukan pemantauan tingkat kinerja dari sistem basis data. Agar informasi yang ada pada *log file* tidak menjadi sangat besar jumlahnya, maka informasi yang dismpan dalam *log file* adalah informasi yang diperoleh dari hasil ekstraksi. Pada makalah ini, informasi yang disimpan dalam *log file* sistem merupakan informasi yang diperoleh dari hasil ekstraksi *query* yang diisukan oleh aplikasi pada *node*. Pada [4], penelitian difokuskan pada proses pengontrolan kinerja sistem basis data dengan indikator utama waktu respons. Sedangkan pada makalah ini, subjek penelitian difokuskan pada proses ekstraksi informasi dari *query* yang diisukan oleh aplikasi pada *node*. Proses ekstraksi ini memiliki arti yang sangat penting, karena proses pengontrolan kinerja sistem basis data sangat tergantung pada informasi yang dihasilkan pada proses ini. Pendekatan yang digunakan pada proses ekstraksi *query* pada makalah ini adalah pendekatan heuristik.

Pada sistem terdistribusi adaptif, proses fragmentasi dan alokasi dilakukan secara berulang dan otomatis apabila tingkat *optimality* sistem sudah menurun pada level tertentu yang disebabkan oleh perubahan yang terjadi. Oleh karena itu, pada sistem terdistribusi adaptif diperlukan sebuah *system monitor* yang berfungsi memonitor tingkat *optimality* sistem, serta melakukan proses refragmentasi dan realokasi apabila penurunan tingkat *optimality* sistem sudah melampaui batas nilai yang ditentukan. Pada sistem terdistribusi adaptif, sistem berevolusi secara otomatis dari sebuah sistem yang pada saat awal merupakan sebuah sistem dengan basis data terpusat menjadi sistem dengan basis data yang terdistribusi. Penelitian ini merupakan penelitian awal dari serangkaian penelitian yang akan dilaksanakan dengan tujuan untuk menciptakan sistem basis data terdistribusi yang adaptif. Sistem basis data terdistribusi adaptif yang akan diciptakan berbasis arsitektur jaringan *peer-to-peer* (P2P) dan menerapkan prinsip *reusable application*, yaitu multi DBMS.

Sistem basis data teristribusi adaptif terdiri atas beberapa komponen utama, antara lain *system monitor*, *application*, dan DBMS. *System monitor* terdiri atas beberapa subsistem, antara lain *query data extractor*, *network-related information collector*, *system optimizer*, *data consistency controller*, *global fragment data directory*, *system monitor database*, dan *global database*. *Global database* ada pada saat awal sebelum terjadi proses fragmentasi dan alokasi. Desain arsitektural sistem basis data terditribusi adaptif pada saat sebelum terjadi proses fragmentasi dan alokasi disajikan pada Gbr. 2, sedangkan setelah terjadi proses fragmentasi dan alokasi disajikan pada Gbr. 3. Perbedaan antara keduanya hanya terletak pada aliran data. Sebelum terjadi proses fragmentasi dan alokasi, semua *data query* (termasuk *data query as data*) mengalir dari semua *node* tempat aplikasi berada menuju *node* tempat *system monitor* berada, dan hasil *query* mengalir pada arah kebalikannya, yaitu dari *node* tempat *system monitor* berada menuju *node* tempat aplikasi berada (Gbr. 2).



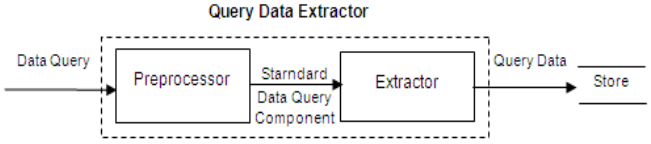
Gbr. 2 Struktur sistem sebelum fragmentasi dan alokasi.



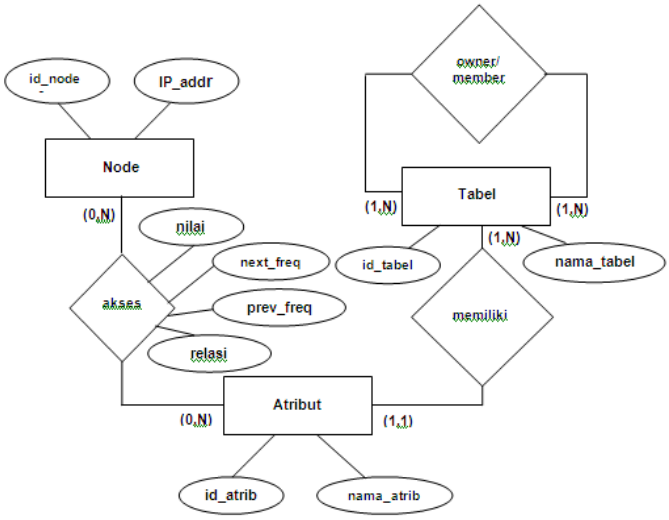
Gbr. 3 Struktur sistem setelah fragmentasi dan alokasi.

Setelah proses fragmentasi dan alokasi terjadi, aliran *data query* dan hasil *query* mengalami perubahan, yaitu sekarang *data query* mengalir dari *node* tempat aplikasi berada menuju *node* tempat *data fragment* yang diinginkan berada, sedangkan aliran hasil *query* terjadi pada arah kebalikannya. Selain itu, setelah proses fragmentasi dan alokasi terjadi, *global database* bisa tidak ada lagi. Yang ada sekarang adalah *local database* atau *data fragment*, serta diciptakannya *local fragment directory* pada masing-masing *node* tempat aplikasi berada dan *global fragment directory* pada *node* tempat *system monitor* berada.

Skema sistem yang disajikan pada Gbr. 2 merepresentasikan sistem aktual, sedangkan skema yang ditunjukkan pada Gbr. 3 merepresentasikan sebuah ilustrasi sistem, dengan jumlah *site* atau *node* pada realitanya dapat lebih banyak daripada yang ada pada Gbr. 3 tersebut dan pola alokasi *fragment* pada realitanya juga dapat berbeda dengan yang ada pada Gbr. 3, sehingga pola aliran *data query* dan *queried data* pada realitanya juga dapat berbeda dengan yang ada pada Gbr. 3 tersebut. Pola aliran *data query as data* yaitu aliran *data query* yang diperlakukan sebagai data untuk keperluan analisis *optimality* atau efisiensi sistem oleh *system monitor* baik sebelum maupun sesudah terjadinya fragmentasi dan alokasi adalah tetap, yaitu dari masing-masing *site*/*node* menuju *site*/*node* tempat *system monitor* berada.



Gbr. 4 Query data extractor.

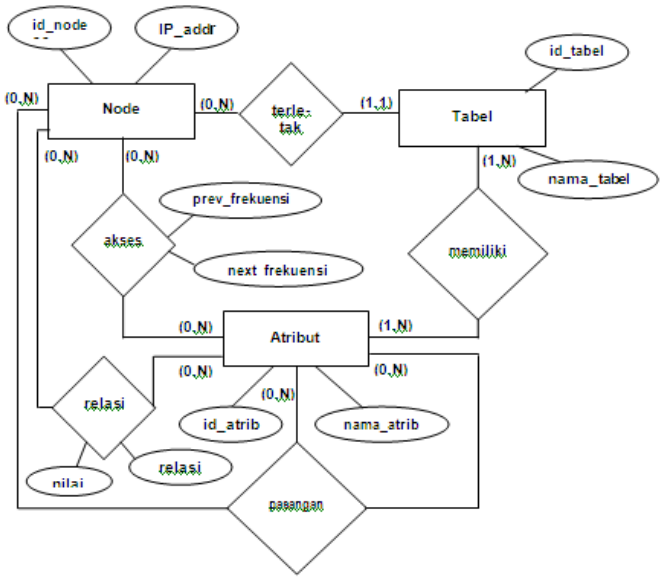


Gbr. 5 Desain konseptual data untuk fragmentasi horizontal.

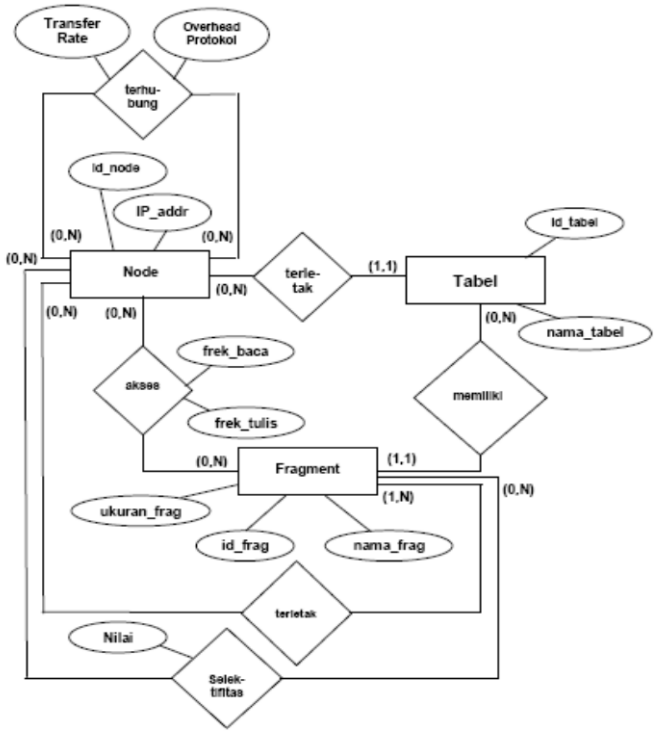
B. Desain Arsitektur Subsistem Query Data Extractor

Mengingat keterbatasan sumber daya, terutama waktu, maka pada makalah ini hanya dirancang dan diimplementasikan subsistem *query data server* atau *query data extractor* yang merupakan salah satu subsistem dari *system monitor*. *Query data extractor* berfungsi untuk mengekstrak informasi dari *query data* yang dikirim oleh *site*/*node* yang dibutuhkan oleh subsistem *system optimizer* untuk menentukan tingkat efisiensi sistem saat ini dan juga untuk melakukan fragmentasi/refragmentasi serta alokasi/realokasi apabila diperlukan.

Subsistem *query data extractor* terdiri atas dua bagian utama, yaitu *preprocessor* dan *extractor*, sebagaimana disajikan pada Gbr. 4. Tugas dari *tokenizer* adalah melakukan pemilahan bagian yang dibutuhkan dari *data query as data* dengan bagian lainnya. Data yang dihasilkan oleh *extractor* selanjutnya diklasifikasi sebelum disimpan ke dalam Basis Data Sistem Monitor (*System Monitor Database*) oleh Manajer Penyimpanan Data (*Data Storing Manager*).



Gbr. 6 Desain konseptual data untuk fragmentasi vertikal.



Gbr. 7 Desain konseptual data untuk alokasi.

C. *Desain Konseptual Basis Data Sistem Basis Data Terdistribusi*

Untuk memperoleh informasi yang akurat tentang karakteristik pengguna, dibutuhkan jumlah data yang mencukupi. Untuk itu, data harus dikumpulkan selama jangka waktu tertentu agar diperoleh data dalam jumlah tertentu, sehingga dibutuhkan basis data untuk menyimpan data yang dikumpulkan selama jangka waktu tersebut. Jenis basis data yang digunakan adalah basis data relasional.

TABEL I
NODE

Kolom	Tipe	PK/FK
id_node	*Integer*	PK
IP_addr	*Char*(19)	

TABEL II
TABEL

Kolom	Tipe	PK/FK
id_tabel	*Integer*	PK
nama_tabel	*Char*(20)	
id_node	*Integer*	FK

TABEL III
ATRIBUT

Kolom	Tipe	PK/FK
id_atrib	*Integer*	PK
nama_atrib	*Char*(20)	

TABEL IV
FRAGMENT

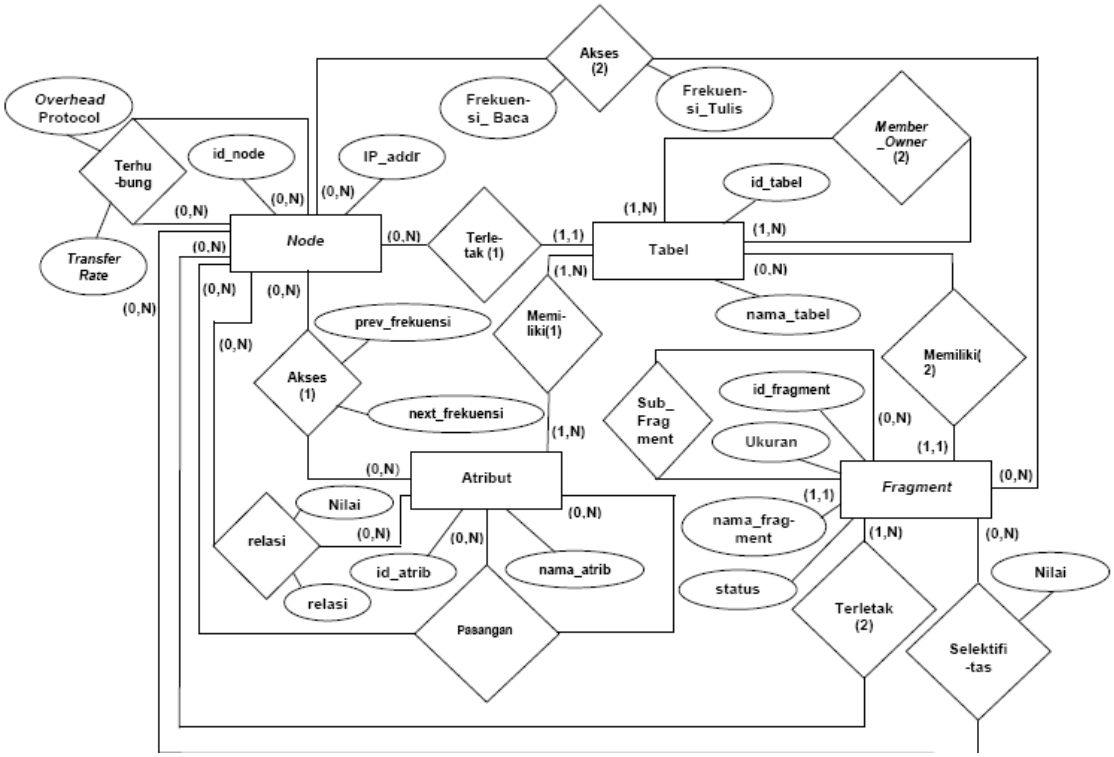
Kolom	Tipe	PK/FK
id_fragment	*Integer*	PK
nama_fragment	*Char*(20)	
ukuran	*Integer*	
id_tabel	*Integer*	FK
status V/H	*Char*(1)	
id_fragment	*Integer*	FK

TABEL V
AKSES_1

Kolom	Tipe	PK/FK
id_node	*Integer*	PK
id_atrib	*Integer*	PK
prev_frekuensi	*Integer*	
next_frekuensi	*Integer*	

1) Skema Basis Data dari Sudut Pandang Fragmentasi Horizontal: Rancangan konseptual untuk basis data dari sudut pandang fragmentasi horizontal disajikan pada Gbr. 5. Skema basis data ini dibuat berdasarkan kebutuhan data untuk proses fragmentasi horizontal, antara lain nama tabel, nama atribut, nilai atribut, logika relasi (yang menghubungkan atribut dan nilainya), hubungan antar objek basis data (*owner* dan *member*), serta identitas *node* (dapat berupa alamat IP dari *node* yang bersangkutan). Semua data ini diekstrak dari *query* yang diisukan oleh aplikasi yang ada pada suatu *node*.

2) Skema Basis Data dari Sudut Pandang Fragmentasi Vertikal: Rancangan konseptual untuk basis data dari sudut pandang fragmentasi vertikal disajikan pada Gbr. 6. Skema basis data ini dibuat berdasarkan kebutuhan data untuk proses fragmentasi vertikal, yaitu data yang terkait dengan aplikasi sebagai berikut: (1) data tentang atribut-atribut yang diakses oleh masing-masing aplikasi; (2) frekuensi akses dari masing-masing aplikasi pada situs-situs yang ada; dan (3) jumlah akses terhadap pasangan atribut (A_i,A_j) oleh masing-masing aplikasi pada situs-situs yang ada. Semua data ini diekstrak dari *query* yang diisukan oleh aplikasi yang ada pada suatu *node*.



Gbr. 8 Desain konseptual basis data.

TABEL VI
AKSES_2

Kolom	Tipe	PK/FK
id_node	*Integer*	PK
id_fragment	*Integer*	PK
Frekuensi_Baca	*Integer*	
Frekuensi_Tulis	*Integer*	

TABEL VII
TERLETAK_2

Kolom	Tipe	PK/FK
id_node	*Integer*	PK
id_fragment	*Integer*	PK

TABEL VIII
MEMBER_OWNER

Kolom	Tipe	PK/FK
id_tabel	*Integer*	PK
id_tabel	*Integer*	PK

TABEL IX
MEMILIKI_1

Kolom	Tipe	PK/FK
id_tabel	*Integer*	PK
id_atrib	*Integer*	PK

3) Skema Basis Data dari Sudut Pandang Alokasi: Alokasi merupakan proses penempatan *fragment-fragment* tabel dan replikasinya pada situs-situs. Proses alokasi dilakukan dengan tujuan untuk membuat kecepatan akses data maksimum dan biaya komunikasi minimum. Data yang dibutuhkan pada proses alokasi adalah (1) selektivitas *fragment*; (2) ukuran *fragment*; (3) jumlah *read access* dan *update access* oleh aplikasi pada *fragment*; (4) kapasitas penyimpanan dan pemrosesan dari tiap-tiap situs; dan (5) biaya komunikasi. Berdasarkan kebutuhan data untuk proses alokasi, disusun desain basis data untuk proses alokasi sebagaimana disajikan pada Gbr. 7.

TABEL X
MEMILIKI_2

Kolom	Tipe	PK/FK
id_tabel	*Integer*	PK
id_fragment	*Integer*	PK

TABEL XI
TERHUBUNG

Kolom	Tipe	PK/FK
id_node	*Integer*	PK
id_node	*Integer*	PK
Transfer_Rate	*Integer*	
Overhead_Protocol	*Integer*	

TABEL XII
SELEKTIFITAS

Kolom	Tipe	PK/FK
id_node	*Integer*	PK
id_fragment	*Integer*	PK
Nilai	*Integer*	

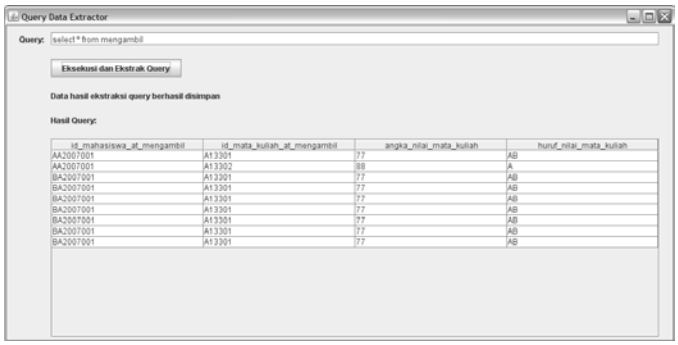
Basis data untuk sistem basis data terdistribusi diperoleh dengan mengintegrasikan desain konseptual basis data dari sudut pandang fragmentasi horizontal, vertikal, dan alokasi. Basis data untuk sistem secara keseluruhan ditunjukkan pada Gbr. 8.

TABEL XIII
PASANGAN

Kolom	Tipe	PK/FK
id_node	*Integer*	PK
id_atrib	*Integer*	PK
id_atrib	*Integer*	PK

TABEL XIV
RELASI

Kolom	Tipe	PK/FK
id_node	*Integer*	PK
id_atrib	*Integer*	PK
relasi	*Char*(2)	
Nilai	*Char*(30)	



Gbr. 9 Tampilan antarmuka subsistem *query data extractor*.

D. Rancangan Tabel Fisik

Dengan mentransformasikan desain konseptual basis data (Gbr. 8) diperoleh 14 buah rancangan tabel fisik yang disajikan dalam Tabel I sampai Tabel XIV.

E. Desain Antarmuka Subsistem Query Data Extractor

Desain antarmuka pengguna untuk subsistem *query data extractor* disajikan pada Gbr. 9. Antarmuka ini dibuat hanya untuk memudahkan proses pengujian subsistem ini, karena pada antarmuka ini ditampilkan data hasil *query* dan notifikasi terkait dengan berhasil tidaknya proses penyimpanan data hasil ekstraksi. Selain itu, melalui antarmuka ini penguji dapat memasukkan *SQL query* melalui sebuah *text field* yang ada pada bagian atas antarmuka ini. Pada bagian bawah *text field* ini terdapat sebuah komponen *button*. Pada komponen *button* ini ditambahkan sebuah *event* yang akan dieksekusi pada saat *button* diklik. Pada *event* ini ditambahkan kode program untuk mengeksekusi dan mengekstrak *query* yang dimasukkan melalui *text field*. Pada bagian bawah *button* terdapat dua buah label. Label atas digunakan untuk menampilkan notifikasi berhasil tidaknya proses penyimpanan data yang diperoleh dari hasil ekstraksi *query*, sedangkan label di bawahnya digunakan untuk menampilkan notifikasi terkait dengan hasil dari eksekusi *query*. Pada bagian bawah dari GUI sistem ini, terdapat sebuah *scroll pane* yang digunakan untuk menampilkan data hasil *query* dalam bentuk tabel.

IV. KESIMPULAN

Query extractor yang dirancang dan diimplementasi telah dapat digunakan untuk mengekstrak *query* baca maupun tulis yang berbasis tiga perintah dasar, yaitu *select*, *update*, dan *delete*, termasuk *joint selection*. *Query extractor* dapat dengan mudah dikembangkan sehingga dapat memproses *variant query* lainnya yang lebih kompleks yang melibatkan lebih banyak klausa seperti *having*, *group by*, dan lain-lain. Perlu juga dikreasikan metode ekstraksi yang nonheuristik untuk memperoleh proses ekstraksi yang efisien. Aspek lain dari sistem basis data terdistribusi yang direncanakan untuk diteliti adalah desain dan implementasi *data query extractor* yang berbasis *Common Object Request Broker Architecture* (CORBA), juga desain dan implementasi subsistem *optimizer*.

UCAPAN TERIMA KASIH

Diucapkan terima kasih yang sebesar-besarnya kepada Pimpinan UNSOED melalui LPPM yang telah memberikan dukungan dana, sehingga penellitian ini dapat terlaksana. Ucapan terimakasih juga disampaikan kepada tim reviewer yang senantiasa bekerja keras dalam penyeleksian proposal penelitian dalam upaya peningkatan kualitas penelitian secara berkelanjutan.

REFERENSI

[1] M.T. Özsu dan P. Valduriez, *Principles of Distributed Database Systems*, Third Ed., New York, USA: Springer, 2010.

[2] A. Patel, R. Hirapara, dan V. Dhamecha, "A Review on Fragmentation Techniques in Distributed Database," *International Journal of Modern Trends in Engineering and Research (IJMTER)*, Vol. 01, No. 3, Sept. hal. 5-10, 2014.

[3] J. Graba, *An Introduction to Network Programming with Java*, Revised Ed., New York, USA: Springer, 2007.

[4] S.F. Rodd dan U.P. Kulkrani, "Adaptive Tuning Algorithm for Performance Tuning of Database Management System," *International Journal of Computer Science and Information Security (IJCSIS)*, Vol. 8, No.1, hal. 121-124, Apr. 2010.