LAPORAN PENELITIAN

PENGGUNAAN SISTEM BASISDATA TERDISTRIBUSI TERHADAP APLIKASI INVENTORY DI GUDANG UNIVERSITAS TERBUKA

Oleh:
Drs. Agus Pratmoko, M. Kom.
NIP 131695127

UNIVERSITAS TERBUKA
Lembaga Penelitian
Pusat Studi Indonesia
2001

LEMBAR PENGESAHAN USULAN PENELITIAN BIDANG ILMU
Jurusan Matematika FMIPA UT

1. Judul Penelitian : **PENGGUNAAN SISTEM BASISDATA TERDISTRIBUSI TERHADAP APLIKASI INVENTORY DI GUDANG UNIVERSITAS TERBUKA**
2. Bidang Penelitian : Bidang Ilmu
3. Klasifikasi Penelitian : Penelitian Mandiri
4. Bidang Ilmu : Komputer
5. Peneliti
   a. Nama lengkap : Drs. Agus Pratmoko, M. Kom.
   b. NIP : 131695127
   c. Pangkat/golongan : Penata / III/c
   d. Jabatan : Lektor
   e. Fakultas/Jurusan : FMIPA / Matematika
   f. Bidang Keahlian : Komputer
6. Lokasi Penelitian : Jakarta
7. Lama penelitian : 4 bulan
8. Biaya Penelitian : Rp 2.050.000,00 (dua juta lima puluh ribu rupiah)

Pondok Cabe, 20 November 2001

Mengetahui,
Dekan FMIPA

Dr. Ir. D. Djokosetiyanto
NIP. 130536671

Peneliti,

Drs. Agus Pratmoko, M. Kom.
NIP: 131695127

Mengetahui,
Ketua Lembaga Penelitian UT

Dr. W.B.P. Simanjuntak
NIP 130212017

Menyetujui,
Kepala Pusat Studi Indonesia

Dr. Tian Belawati
NIP 131569974

ABSTRAK

Manajemen pergudangan yang ada di bagian Pusat Distribusi saat ini sudah ditangani menggunakan komputer yaitu dengan adanya aplikasi komputer untuk pergudangan. Dalam aplikasi tersebut digunakan basisdata terpusat, artinya semua data ada di UT-Pusat dan penjualan bahan belajar dilakukan di seluruh UPBJJ-UT yang tersebar di seluruh Indonesia.

Penelitian ini bertujuan untuk merancang suatu aplikasi pergudangan dengan basisdata terdistribusi, yaitu suatu basisdata yang secara lokasi terpisah di Jakarta, Surabaya, dan Makassar tetapi sebenarnya terintegrasi menjadi satu yaitu dalam aplikasi inventory di Universitas Terbuka. Dalam aplikasi ini basisdata yang dipakai dalam basisdata terpusat akan difragmentasi menjadi beberapa bagian dan dialokasikan di tiga tempat tersebut di atas, termasuk menentukan *entity-entity* beserta *attribute*-nya, dan *relationship* antar *entity*.

Hasil dari penelitian ini berupa rancangan *relation* untuk basisdata *relational*, rancangan *attribute* dan *tuple*-nya, rancangan fragmentasi data, dan rancangan alokasi data.

DAFTAR ISI

I PENDAHULUAN

1.1 Latar Belakang

Gudang dan pergudangan merupakan salah satu bagian penting dari seluruh proses pabrik atau badan usaha sejenisnya. Segala kegiatan di gudang ini tentu berpengaruh besar terhadap mekanisme kegiatan unit-unit lain dalam pabrik tersebut. Gudang dan pergudangan berfungsi menerima dan menyimpan bahan baku dari unit pembelian, kemudian menyalurkannya ke bagian unit produksi, selanjutnya menerima dan menyimpan barang jadi tersebut. Tahap berikutnya gudang mengeluarkan barang jadi tersebut ke bagian distribusi. Melihat kenyataan di atas, jelaslah bahwa apabila kegiatan di gudang dan pergudangan mengalami hambatan maka proses unit kerja pembelian, produksi, distribusi maupun pemasaran akan terhambat. Dengan kata lain, bila sistem manajemen pergudangan itu jelek, dengan sendirinya akan merugikan seluruh pabrik atau badan usaha sejenisnya, walaupun manajemen di unit-unit yang lain sudah baik.

Manajemen pergudangan sudah selayaknya ditangani menggunakan komputer, yang tentunya segala jenis informasi dan data yang ada di gudang dapat diproses dengan komputer yang akan menghasilkan suatu keluaran, baik itu di monitor atau dalam bentuk laporan di kertas. Untuk itu diperlukan suatu software (perangkat lunak) yang dapat mengelola masalah-masalah yang ada di gudang, sehingga semua masalah itu dapat diselesaikan. Software yang dimaksud di sini diharapkan dapat mencatat pengeluaran, penerimaan, dan sisa barang di gudang, dapat mencatat persediaan yang sedang dipesan dan tinggal menunggu penyerahannya, dan masih banyak lagi permasalahan yang dihadapi dalam pergudangan.

1.2 Ruang lingkup

Universitas Terbuka (UT) merupakan satu-satunya Perguruan Tinggi di Indonesia yang mahasiswanya tersebar di seluruh pelosok tanah air, mereka menggunakan modul (bahan belajar) sebagai bahan pengajaran terhadap mahasiswanya. Modul-modul ini dipesan oleh UT dari percetakan dan disimpan di dalam gudang sebelum disalurkan ke agen pemesan, dalam hal ini UPBJJ-UT yang ada di 31 tempat seluruh Indonesia. Oleh karena terlalu banyak modul yang dihasilkan, demikian juga banyaknya agen, ditambah lagi beberapa kendala dalam pengiriman, maka mata rantai jaringan ini begitu rumit. Dengan adanya komputerisasi pada masalah pergudangan ini diharapkan manajemen pergudangan akan jauh lebih baik dibandingkan dengan proses manual. Saat ini pengelolaan pergudangan di UT sudah memakai komputer, hanya saja basisdata terletak di Jakarta atau memakai *sistem basisdata terpusat*.

Pada penelitian ini akan diasumsikan bahwa basisdata yang digunakan akan terdistribusi di tiga lokasi, yaitu Jakarta, Surabaya, dan Makassar, dan masing-masing lokasi saling terhubung oleh suatu jaringan komputer sehingga dimungkinkan terjadinya pertukaran data dan informasi di antara masing-masing lokasi tersebut, sistem ini dikenal dengan *sistem basisdata terdistribusi*. Di tiga kota tersebut, masing-masing terdapat gudang sebagai tempat menyimpan bahan belajar. Data yang disimpan terdistribusi pada komputer-komputer yang ada pada jaringan komputer tersebut, sedangkan program aplikasi inventory yang dijalankan pada komputer-komputer tersebut dapat mengambil data dari lokasi-lokasi yang berbeda. Rancangan basisdata terdistribusi yang akan dibuat dalam penelitian ini meliputi fragmentasi *entity type* dan alokasi fragmen terhadap basisdata terpusat yang selama ini ada di UT.

1.3 Tujuan Penelitian

Penelitian ini mempunyai tujuan:

- Merancang sistem basisdata terdistribusi pada aplikasi inventory di UT yang selama ini menggunakan sistem basisdata terpusat.
- Merepresentasikan model data aplikasi sistem inventory basisdata terdistribusi.
- Merancang basisdata yang akan digunakan dalam aplikasi inventory, termasuk menentukan *entity-entity* beserta *attribute*-nya.
- Membentuk fragmentasi basisdata terpusat menjadi tiga basisdata lokal yang terhubung dengan jaringan komputer.
- Memberikan masukan kepada *programmer* berupa rancangan modul serta program-program yang terkait terhadap masing-masing modul pada aplikasi inventory.

1.4 Manfaat Penelitian

Dari hasil penelitian ini diharapkan dapat diperoleh cara lain dari cara yang selama ini dipakai, yang mungkin lebih menguntungkan, dalam mengelola basisdata pada aplikasi inventory. Sehingga dimungkinkan untuk mengganti aplikasi inventory yang selama ini dipakai oleh UT. Dengan demikian tidak ada lagi keterlambatan pengiriman bahan belajar UT ke UPBJJ-UPBJJ.

1.5 Metode Penelitian

- Studi literatur tentang sistem basisdata terpusat dan sistem basisdata terdistribusi, terutama yang berkaitan dengan perancangan basisdata.

- Menganalisis sistem basisdata yang dipakai pada aplikasi inventory di Universitas Terbuka.
- Melakukan fragmentasi terhadap *attribute-attribute* pada sistem basisdata terpusat.
- Melakukan perancangan sistem basisdata terdistribusi baik relasi global maupun relasi lokal yang akan ditempatkan pada beberapa lokasi.
- Pembuatan laporan.

II LANDASAN TEORI

2.1 Pendahuluan

Manajemen basisdata berkembang dari aplikasi komputer khusus menjadi komponen utama dalam suatu sistem informasi organisasi. Dalam periode 1970-an kita lihat banyak sekali komputer digunakan untuk mengembangkan sistem basisdata terintegrasi. Teknologi sistem basisdata telah banyak mengembangkan dasar-dasar teori mengenai sistem basisdata dan telah digunakan dalam banyak aplikasi. Pada saat yang sama, teknologi jaringan komputer juga berkembang dengan pesat, yang memungkinkan terhubungnya bermacam-macam komputer sehingga terjadi pertukaran data dan informasi di antara komputer-komputer tersebut.

Dalam beberapa tahun terakhir, tersedianya sistem basisdata dan jaringan komputer menyebabkan perkembangan baru yaitu basisdata terdistribusi. Basisdata terdistribusi merupakan suatu sistem basisdata terintegrasi yang dikembangkan di atas jaringan komputer. Data yang tersimpan di sini terdistribusi pada komputer-komputer yang ada pada jaringan tersebut, sedangkan program aplikasi yang dijalankan pada komputer-komputer tersebut akan mengambil data dari lokasi-lokasi komputer yang berbeda.

Beberapa alasan yang merupakan dasar perkembangan basisdata terdistribusi:

a. Alasan organisasi dan ekonomi

   Banyak organisasi yang bersifat desentralisasi, pendekatan melalui basisdata terdistribusi secara alamiah lebih tepat karena lebih sesuai dengan struktur organisasi yang ada. Perkembangan teknologi komputer juga menyebabkan penggunaan komputer besar atau mainframe computer secara terpusat cenderung untuk dihindari karena alasan ekonomi.

b. Menghubungkan beberapa sistem basisdata yang sudah ada

   Penggunaan sistem basisdata terdistribusi merupakan pemecahan yang terbaik bila dalam organisasi tersebut sudah ada beberapa basisdata yang letaknya terpisah, dan diperlukannya aplikasi global yang memerlukan diaksesnya data dari beberapa lokasi secara bersamaan.

c. Pertumbuhan di masa datang

   Bila suatu organisasi berkembang dengan menambahkan unit-unit organisasi baru yang relatif mempunyai otonomi sendiri, pendekatan dengan memanfaatkan basisdata terdistribusi menunjang terintegrasinya unit baru tersebut tanpa harus merubah sistem yang telah ada, atau dengan perubahan yang sangat minimal. Melalui sistem terpusat, penambahan di atas sulit untuk diintegrasikan baik untuk sistem yang baru maupun sistem yang sudah ada.

d. Mengurangi biaya komunikasi

Dalam basisdata terdidtribusi data-data tersebut tersebar dalam area geografis yang luas, kenyataan bahwa banyak aplikasi yang bersifat lokal jelas akan mengurangi komunikasi data. Dalam basisdata terpusat, jelas bahwa komunikasi data akan menjadi biaya tetap yang mahal.

e. Pertimbangan unjuk kerja

Adanya beberapa prosesor yang dapat dimanfaatkan secara bersama-sama jelas akan meningkatkan unjuk kerja yang merupakan akibat dari kerja paralel dari beberapa prosesor tersebut. Hal ini tidak hanya berlaku pada basisdata terdistribusi saja, melainkan juga berlaku untuk semua sistem yang mempergunakan banyak prosesor.

f. Reliabilitas dan availabilitas

Pendekatan basisdata terdistribusi, khususnya dengan duplikasi data, dapat digunakan untuk mencapai reliabilitas dan availabilitas yang lebih tinggi.

2.2 Perbedaan antara basisdata terdistribusi dan terpusat

Basisdata terdistribusi bukan merupakan implementasi sederhana dari suatu basisdata terdistribusi pada suatu basisdata terpusat. Basisdata terdistribusi mempunyai dasar perancangan yang berbeda dengan basisdata terpusat. Untuk mendapatkan pengertian yang lebih baik mengenai arsitektur basisdata terdistribusi, perlu kita lihat beberapa perbedaan penting antara kedua sistem basisdata tersebut.

a. Pengawasan terpusat

Salah satu motivasi terkuat dalam pengembangan basisdata adalah agar perusahaan atau organisasi pempunyai pengawasan terpusat terhadap seluruh sumber informasi. Hal ini merupakan pengembangan dari sistem informasi dimana setiap aplikasi mempunyai berkas-berkas yang dimiliki sendiri. Fungsi utama dari Administrator Basisdata (DBA) adalah untuk menjamin keamanan data, yang memerlukan tanggung jawab terpusat. Dalam basisdata terdistribusi, masalah pengawasan terpusat kurang begitu ditekankan. Secara umum dalam basisdata terdistribusi kita dapat lihat struktur pengawasan hirarki berdasarkan administrator basisdata global, yang bertanggung jawab terhadap basisdata secara keseluruhan, dan administrator basisdata lokal yang bertanggung jawab terhadap basisdata yang ada pada lokasi-lokasi lain. Perlu ditekankan di sini mengenai otonomi administrator basisdata lokal, dimana otonomi itu harus cukup tinggi sehingga bila tidak ada administrator basisdata global, maka administrator basisdata lokal akan mempunyai otonomi penuh terhadap basisdata yang ada pada

lokasinya, dan koordinasinya dengan administrator basisdata lokal lainnya. Ciri ini biasa disebut sebagai otonomi tempat (*site autonomy*).

b. Data bebas *(data independence)*

Data bebas merupakan motivasi utama dalam pendekatan sistem basisdata. Pada dasarnya data bebas mempunyai arti bahwa aplikasi tidak akan dipengaruhi oleh struktur data, sehingga perubahan-perubahan yang terjadi pada struktur fisik data tidak akan mempengaruhi program-program yang telah ada. Dalam basisdata terdistribusi, hal ini juga merupakan aspek penting. Akan tetapi terdapat aspek penting baru yang disebut distribusi transparan. Distribusi transparan berarti program aplikasi yang dibuat tidak menganggap bahwa basisdata sebenarnya terdistribusi. Jadi kebenaran dari suatu program aplikasi tidak dipengaruhi oleh perpindahan data antara lokasi satu ke lokasi lain, akan tetapi kecepatan pemprosesan yang dipengaruhi. Dalam basisdata terpusat, data bebas dicapai melalui arsitektur bertingkat di mana masing-masing tingkat mempunyai diskripsi data yang berbeda dan dengan pemetaan di antara tingkat-tingkat tersebut. Dalam basisdata terpusat, tingkat-tingkat ini dikenal dengan skema konseptual, skema tempat, dan skema eksternal.

c. Memperkecil duplikasi data *(reduction of redundancy)*

Dalam basisdata terpusat usaha-usaha memperkecil duplikasi data dilakukan karena, pertama, konsistensi data dapat dicapai dengan mempergunakan file tunggal, dan kedua, menghemat penggunaan media penyimpanan. Dengan file tunggal dan semua aplikasi melakukan akses pada file ini, maka konsistensi dapat dicapai. Dalam basisdata terdistribusi, duplikasi data kadang-kadang justru diperlukan. Terdapat dua alasan mengapa duplikasi data diperlukan, pertama, lokalitas aplikasi dapat ditingkatkan bila data yang sama ada pada semua lokasi, dan kedua, tidak berfungsinya salah satu lokasi tidak akan menyebabkan aplikasi terhenti. Terdapat kerugian dan keuntungan dalam penggunaan duplikasi data. Dalam aplikasi pengambilan data, duplikasi data jelas akan mempercepat proses karena transaksi yang dilakukan adalah lokal. Akan tetapi dalam proses *update*, penggunaan duplikasi data akan memerlukan proses yang lama karena *update* ini harus dilakukan terhadap semua lokasi yang mempunyai duplikasi data tersebut. Masalah duplikasi dalam basisdata terdistribusi pada dasarnya adalah bagaimana dapat dicapai suatu keadaan optimal.

d. Struktur fisik yang kompleks dan akses yang efisien

Dalam basisdata terpusat, penggunaan struktur fisik yang kompleks, misalnya penggunaan indeks kedua, diperlukan agar dapat dicapai efisiensi akses yang tinggi. Untuk basisdata

terdistribusi, struktur akses yang kompleks bukan merupakan alat yang tepat. Karena itu, sementara efisien akses merupakan masalah dalam basisdata terdistribusi, struktur fisik bukan merupakan isu teknologi yang relevan. Akses yang efisien pada basisdata terdistribusi tidak dicapai dengan menerapkan struktur fisik antar lokasi yang kompleks karena sulit sekali untuk mengembangkan dan memelihara struktur seperti ini. Dua masalah utama dalam efisien akses untuk basisdata terdistribusi yaitu, pertama, optimisasi global dan kedua optimisasi lokal. Optimisasi global berhubungan dengan penentuan mengenai data apa yang akan diakses pada lokasi tertentu, dan file mana yang harus dipindahkan dari lokasi yang satu ke lokasi yang lain. Parameter utama adalah biaya komunikasi disamping biaya yang diperlukan untuk akses lokalnya. Optimisasi lokal berhubungan dengan cara akses lokal pada setiap lokasi. Masalah ini sama dengan masalah untuk basisdata terpusat.

e. Integritas, membangun, dan konkuren kontrol *(integrity, recovery, and concurrency control)*

Ketiga masalah ini, walaupun berbeda mempunyai saling keterkaitan yang erat. Pada dasarnya cara pemecahannya adalah bagaimana suatu transaksi dilaksanakan. Transaksi merupakan suatu unit operasi yang terkecil yang terdiri atas urut-urutan operasi yang harus dijalankan seluruhnya atau tidak sama sekali. Dalam basisdata terpusat, masalah ini lebih mudah dipecahkan karena seluruh data terletak pada satu lokasi. Akan tetapi untuk basisdata terdistribusi, hal ini lebih sulit untuk diterapkan karena satu transaksi mungkin harus dilaksanakan untuk beberapa lokasi. Transaksi atomic jelas merupakan dasar untuk mencapai integritas data, karena hal ini akan mengubah data pada lokasi-lokasi di mana data tersebut berada, atau tidak mengubah sama sekali. Konkuren kontrol dipakai untuk menjamin operasi yang atomic pada saat terjadinya transaksi, sedangkan membangun dipakai untuk mengembalikan status data menjadi status data sebelum dilakukannya suatu transaksi yang gagal.

f. Privacy dan keamanan *(privacy and security)*

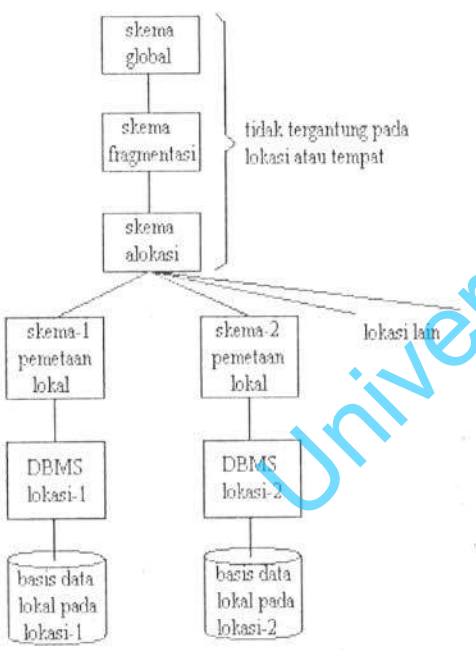
Dalam basisdata terpusat, administrator basisdata memiliki wewenang penuh yang menjamin bahwa pemakai data adalah pemakai aplikasi yang telah mendapatkan otoritas. Dalam basisdata terdistribusi, administrasi basisdata lokal mempunyai tanggung jawab penuh terhadap data lokal, sedangkan administrator basisdata global bertanggung jawab terhadap sistem basisdata tersebut secara keseluruhan. Masalah keamanan merupakan masalah penting, karena jaringan komunikasi merupakan point terlemah sehubungan dengan proteksi.

2.3 Arsitektur untuk basisdata terdistribusi

Salah satu hal penting dalam basisdata terdistribusi adalah bagaimana pemakai atau pemrogram aplikasi memandang basisdata terdistribusi tersebut. Salah satu konsep penting dalam basisdata terdistribusi adalah konsep data bebas (*data indepedence*), dimana program-program aplikasi seharusnya tidak dipengaruhi oleh perubahan-perubahan yang terjadi pada distribusi data.

Konsep ini dikenal sebagai distribusi transparan (*distribution transparancy*). Terdapat beberapa level dari distribusi transparan, yang akan kita lihat pada arsitektur untuk basisdata terdistribusi. Arsitektur ini memungkinkan kita untuk dengan mudah menentukan level-level yang berbeda untuk distribusi transparan. Level-level yang ada secara konsep relevan untuk mengerti basisdata terdistribusi, walaupun beberapa sistem mungkin tidak mengimplementasikan secara eksplisit.

Gambar 1 memperlihatkan arsitektur untuk basisdata terdistribusi. Arsitektur ini tidak secara eksplisit diterapkan pada semua sistem basisdata terdistribusi. Akan tetapi level-level yang ada secara konsepsi adalah relevan untuk mendapatkan pengertian yang lebih baik mengenai organisasi pada setiap basisdata terdistribusi.



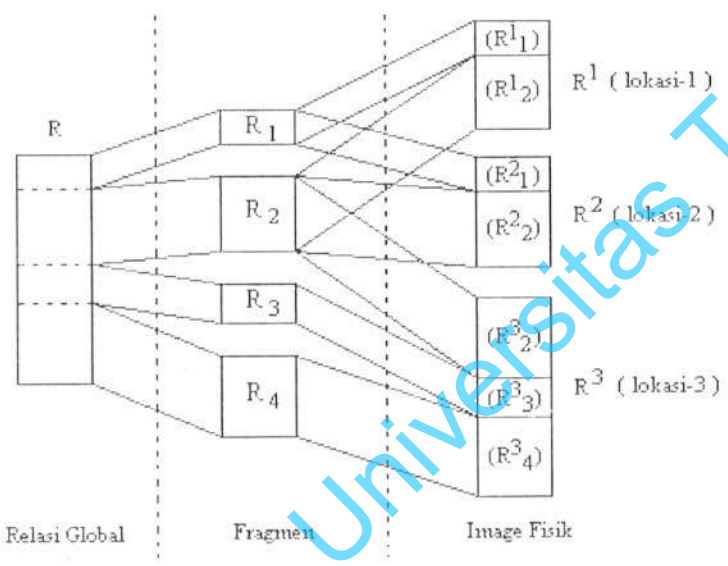
Gambar 1

Pada level paling atas, disebut skema global. Skema global di sini mendefinisikan data-data apa yang ada di basisdata terdistribusi ini, tetapi dapat dipandang seakan-akan basisdata ini terpusat. Karena itu skema global dapat didefinisikan seperti pada sistem basisdata terpusat. Dengan menggunakan model relasional, maka skema global ini terdiri atas definisi dari suatu set relasi global.

Setiap relasi global dapat dipecah menjadi beberapa bagian yang tidak saling tumpang tindih, yang disebut fragmen-fragmen. Terdapat beberapa cara untuk melakukan pemecahan ini yang akan dijelaskan kemudian. Pemetaan antara relasi global dan fragmen didefinisikan sebagai skema fragmentasi. Pemetaan ini bersifat satu ke banyak, yang berarti beberapa fragmen berhubungan dengan satu relasi global, tetapi hanya satu relasi global yang berhubungan dengan satu fragmen. Fragmen-fragmen ini ditunjukkan melalui nama relasi global dengan suatu indeks, sebagai contoh R_i menunjukkan fragmen yang ke-i bagi relasi global R.

Fragmen adalah bagian logik dari relasi global yang secara fisik terletak pada satu atau banyak lokasi dalam jaringan. Skema alokasi menentukan pada lokasi-lokasi mana suatu fragmen ditempatkan. Jenis pemetaan dalam skema alokasi juga menentukan apakah basisdata terdistribusi tersebut bersifat *redundant* atau *nonredundant*. *Redundant* berarti pemetaan bersifat satu ke banyak, sedangkan *nonredundant* berarti pemetaan bersifat satu ke satu. Fragmen-fragmen yang mempunyai hubungan pada satu relasi global R dan terletak pada lokasi yang sama j merupakan image fisik dari relasi global R pada lokasi J. Jadi terdapat pemetaan satu ke satu antara image fisik dan suatu pasangan <relasiglobal, lokasi>. Image fisik dapat ditunjukkan dengan nama relasi global dan indeks lokasi. Untuk membedakan dengan fragmen, disini digunakan superscript, sebagai contoh, R^j menunjukkan image fisik dari relasi global R pada lokasi j.

Suatu contoh hubungan antara jenis obyek yang didefinisikan di atas ada pada Gambar 2. Relasi global R dipecah menjadi 4 (empat) fragmen R_1, R_2, R_3, dan R_4. Keempat fragmen ini ditempatkan secara *redundant* pada tiga lokasi jaringan komputer, dan membuat 3 (tiga) image fisik R^1, R^2, dan R^3. Untuk menentukan copy dari suatu fragmen pada lokasi tertentu kita gunakan notasi dari nama relasi global dengan 2 (dua) indeks (indeks fragmen dan indeks lokasi). Jadi pada Gambar 2, notasi R^3_2 menunjukkan copy dan fragmen R_2 yang terletak pada lokasi 3. Perhatikan disini bahwa dua image fisik bisa identik. Dalam hal ini dapat kita katakan bahwa suatu image fisik adalah copy dari image fisik yang lain, R^1 adalah copy dari R^2.



Gambar 2

Kalau kita kembali pada arsitektur seperti apa yang terlihat pada Gambar 1, kita telah menggambarkan hubungan antara obyek-obyek yang ada pada tiga level paling atas dari arsitektur ini. Ketiga level ini tidak tergantung pada lokasi (*site independent*), jadi tidak tergantung pada data model dari basisdata lokal. Pada level yang lebih rendah, diperlukan pemetaan image fisik pada obyek-obyek yang akan dimanipulasi oleh basisdata lokal. Pemetaan ini disebut sebagai skema pemetaan lokal dan tergantung pada jenis data lokal. Dalam suatu

sistem yang heterogen (mempergunakan banyak sistem basisdata), kita akan mempunyai jenis pemetaan yang berbeda-beda pada lokasi yang berbeda-beda.

Arsitektur ini memberikan suatu konsep umum untuk basisdata terdistribusi. Tiga objektif utama yang merupakan dasar dari arsitektur ini, yaitu pertama, pemisahan antara fragmentasi data dan alokasi, kedua, kontrol terhadap *redundancy*, dan ketiga, bebas dari sistem basisdata lokal.

a. Pemisahan konsep fragmentasi data dari konsep alokasi data

Pemisahan ini memungkinkan kita untuk membedakan dua level berbeda untuk distribusi transparan, yaitu fragmentasi transparan dan lokasi transparan. Fragmentasi transparan merupakan tingkat transparan paling tinggi, sehingga pemakai atau pemrogram aplikasi bekerja pada relasi global. Transparan lokasi merupakan tingkat transparan yang lebih rendah, sehingga pemakai atau pemrogram aplikasi harus bekerja pada tingkat fragmen, akan tetapi pemakai ini tidak mengetahui dimana fragmen tersebut berada. Pemisahan ini sangat berguna dalam basisdata terdistribusi, karena penentuan bagian-bagian data yang relevan terpisah dan alokasi optimal.

b. Kontrol *redundancy* yang eksplisit

Arsitektur ini memberikan kontrol yang eksplisit pada level fragmen. Contoh pada Gambar 2 memperlihatkan bahwa dua image fisik R^2 dan R^3 adalah saling tindih. Definisi dari fragmen yang berbeda sebagai dasar dari image fisik memungkinkan kita untuk mengetahui secara eksplisit bagian mana yang saling tindih tadi.

c. Bebas dari sistem basisdata lokal

Hal ini yang disebut sebagai pemetaan lokal yang transparan, memungkinkan kita untuk mempelajari masalah-masalah dalam basisdata terdistribusi tanpa harus mempertimbangkan sistem basisdata lokal apa yang digunakan.

Jenis transparan lain yang berhubungan dengan lokasi transparan adalah replikasi transparan, hal ini berarti pemakai tidak perlu replikasi dan fragmen.

2.4 Jenis-jenis fragmentasi data

Pengaturan dan relasi global ke dalam fragmen-fragmen dapat dilakukan dengan mempergunakan dua jenis fragmentasi yaitu fragmentasi horisontal dan fragmentasi vertikal. Selain kedua jenis fragmentasi ini, kita dapat membuat fragmentasi yang lebih kompleks dengan mempergunakan kedua jenis fragmentasi ini bersama-sama.

Untuk semua jenis fragmentasi, suatu fragmen dapat didefinisikan melalui suatu ekspresi dengan mempergunakan bahasa relasional (dalam hal ini aljabar relasioanal), dimana kita

gunakan relasi global sebagai *operand* dan menghasilkan suatu fragmen. Sebagai contoh, bila relasi global berisi data mengenai pegawai, suatu fragmen yang berisi data pegawai suatu departemen tertentu dapat didefinisikan melalui operasi *selection* pada relasi global. Terdapat beberapa aturan untuk menentukan fragmen, yaitu:

a. Kondisi kelengkapan

   Semua data pada relasi global harus dipetakan ke dalam fragmen, tidak boleh terjadi suatu data yang ada pada relasi global tetapi tidak ada pada semua fragmen.

b. Kondisi pembentukan kembali

   Relasi global harus dapat dibentuk kembali berdasarkan fragmen-fragmen yang ada.

c. Kondisi terpisah

   Bila fragmen-fragmen tersebut terpisah, replikasi data dapat dikontrol pada level alokasi, akan tetapi kondisi ini hanya berguna pada fragmentasi horisontal. Untuk fragmentasi vertikal kondisi ini kadang-kadang tidak dapat dipenuhi.

2.4.1 Fragmentasi horisontal

Fragmentasi horisontal adalah mempartisi *tuple-tuple* yang ada pada relasi global ke dalam beberapa subset. Hal ini berguna dalam basisdata terdistribusi, dimana suatu subset dapat berisi data-data yang mempunyai ciri-ciri geografis yang sama. Sebagai contoh, kita memiliki relasi global seperti di bawah ini.

SUPPLIER(SNOMOR, NAMA, KOTA)

Fragmentasi horisontal dapat didefinisikan dengan cara sebagai berikut:

$$SUPPLIER_1 = SL_{KOTA = \text{"JKT"}}\ SUPPLIER$$

$$SUPPLIER_2 = SL_{KOTA = \text{"BDG"}}\ SUPPLIER$$

SL adalah operasi *selection* pada aljabar *relational*.

Fragmentasi di atas memenuhi kondisi kelengkapan kalau "JKT" (untuk Jakarta) dan "BDG" (untuk Bandung) hanya merupakan nilai-nilai yang ada pada *attribute* KOTA. Kondisi pembentukan kembali juga dipenuhi, karena kita dapat membentuk kembali relasi global SUPPLIER melalui operasi:

$$SUPPLIER = SUPPLIER_1\ UN\ SUPPLIER_2$$

UN adalah operasi *Union* pada aljabar *relational*.

Predikat yang digunakan pada operasi seleksi yang mendefinisikan suatu fragmen disebut sebagai qualifikasi. Pada contoh di atas qualifikasi yang kita gunakan adalah:

q_1: KOTA = "JKT"

q_2: KOTA = "BDG"

Secara umum, untuk memenuhi kondisi kelengkapan, set dari qualifikasi untuk semua fragmen harus lengkap, paling tidak dengan set dari nilai-nilai yang diperbolehkan. Kondisi pembentukan kembali harus selalu dipenuhi melalui operasi penggabungan (*union*) dan kondisi terpisah dipenuhi bila qualifikasi-qualifikasi tersebut bersifat *mutually exclusive*.

2.4.2 Fragmentasi horisontal yang diturunkan

Dalam kasus tertentu, fragmentasi horisontal dari suatu relasi tidak dapat langsung diperoleh dari *attribute-attribute* yang ada pada relasi tersebut, melainkan diperoleh melalui penurunan dari fragmentasi horisontal dari relasi yang lain. Perhatikan contoh berikut:

SUPPLY(SNOMOR, PNOMOR, DEPTNOMOR, JUML)

dengan SNOMOR adalah nomor dari supplier.

Partisi yang baik dari relasi ini jelas bila fragmen-fragmen yang ada berisi *tuple* untuk para supplier kota tertentu. Akan tetapi kota di sini bukan merupakan *attribute* dari relasi SUPPLY , melainkan merupakan *attribute* dari relasi SUPPLIER. Untuk itu kita perlu melakukan operasi semi-join untuk menentukan *tuple* dari SUPPLY yang berhubungan dengan SUPPLIER pada kota tertentu. Fragmentasi yang diturunkan untuk SUPPLY dapat didefinisikan sebagai berikut:

$$SUPPLY_1 = SUPPLY\ SJ_{SNUM = SNUM}\ SUPPLIER_1$$

$$SUPPLY_2 = SUPPLY\ SJ_{SNUM = SNUM}\ SUPPLIER_2$$

SJ adalah operasi Semi-Join pada aljabar *relational*.

Operasi semi join dipergunakan untuk mengambil dari SUPPLY *tuple-tuple* yang memenuhi kondisi join antara $SUPPLIER_1$ atau $SUPPLIER_2$ dengan SUPPLY, jadi menentukan *tuple-tuple* dari SUPPLY yang mengacu pada supplier di Jakarta atau Bandung.

Pembentukan relasi global SUPPLY dapat dilakukan melalui operasi *union* seperti pada SUPPLIER.

Kalau relasi global R mempunyai fragmentasi yang berasal dari penurunan, qualifikasinya tidak dapat diperlihatkan melalui predikat yang digunakan untuk *attribute* untuk R. Qualifikasi untuk contoh ini adalah:

q_1: SUPPLY.SNOMOR = SUPPLIER.SNOMOR AND SUPPLIER.KOTA = "JKT"

q_2: SUPPLY.SNOMOR = SUPPLIER.SNOMOR AND SUPPLIER.KOTA = "BDG"

Untuk setiap *tuple* dari $SUPPLY_1$ atau $SUPPLAY_2$, ada supplier dari Jakarta atau Bandung dengan nomor supplier yang sama. Kondisi kelengkapan di sini harus memenuhi suatu keadaan bahwa kalau ada nomor supplier dalam relasi SUPPLY, maka nomor itu juga harus ada pada relasi SUPPLIER.

Kalau kita lihat contoh di atas, setidaknya kunci harus direplikasi dalam seluruh fragmen untuk pembentukan kembali. Secara umum dalam fragmentasi vertikal masalah kondisi terpisah ini tidak sepenting dalam fragmentasi horisontal.

2.4.3 Fragmentasi vertikal

Fragmentasi vertikal dari suatu relasi global, adalah membagi *attribute-attribute* dari relasi tersebut ke dalam grup-grup. Fragmen diperoleh dengan mempergunakan operasi proyeksi terhadap relasi global tersbut. Hal ini berguna bila basisdata terdistribusi terdiri dari *attribute-attribute* yang berisi data yang mempunyai ciri-ciri geografis yang sama.

Sebagai contoh perhatikan relasi global berikut:

PEGAWAI(PNOMOR, NAMA, GAJI, PJK, MGRNOMOR, DEPTNOMOR)

Fragmentasi vertikal dapat didefinisikan sebagai berikut:

$$PEGAWAI_1 = PJ_{\text{PNOMOR, NAMA, MGRNOMOR, DEPTNOMOR}}\ PEGAWAI$$

$$PEGAWAI_2 = PJ_{\text{PNOMOR, GAJI, PJK}}\ PEGAWAI$$

PJ adalah operasi Projection pada aljabar *relational*.

Fragmentasi ini, sebagai contoh, dapat digunakan untuk suatu organisasi dimana gaji dan pajak diatur oleh satu unit terpisah. Pembentukan kembali relasi PEGAWAI diperoleh melalui:

$$PEGAWAI = PEGAWAI_1\ JN_{\text{PNOMOR=PNOMOR}}\ PEGAWAI_2$$

karena PNOMOR adalah kunci dari relasi PEGAWAI.

Secara umum memasukkan kunci dari relasi global ke dalam setiap fragmen merupakan jalan yang paling sederhana untuk menjamin pembentukan relasi global melalui operasi join. Alternatif lain adalah dengan mempergunakan identifikasi *tuple* yang digunakan untuk kunci dan akan diatur oleh sistem. Hal ini mungkin lebih baik untuk menghindari replikasi dari kunci yang panjang, dan keuntungan lain adalah kunci ini tidak dapat diubah oleh pemakai.

Pembentukan relasi global PEGAWAI di sini tidak memenuhi kondisi kelengkapan, karena hasil yang diperoleh dari penggabungan $PEGAWAI_1$ dan $PEGAWAI_2$ berisi dua kolom PNOMOR. Replikasi ini dapat dihilangkan melalui operasi proyeksi.

2.4.4 Fragmentasi campuran

Fragmen-fragmen yang diperoleh dari operasi-operasi di atas adalah relasi itu sendiri. Oleh karena itu kita dapat melakukan operasi-operasi fragmentasi secara rekursif, selama kondisi-kondisi kebenaran di atas tetap berlaku. Pembentukan kembali dapat dibuat melalui aturan pembentukan yang dilakukan secara terbalik. Ekspresi dalam mendefinisikan fragmen-fragmen tersebut menjadi lebih kompleks. Sebagai contoh, kita gunakan relasi global yang sama.

PEGAWAI(PNOMOR, NAMA, GAJI, PJK, MGRNOMOR, DEPTNOMOR)

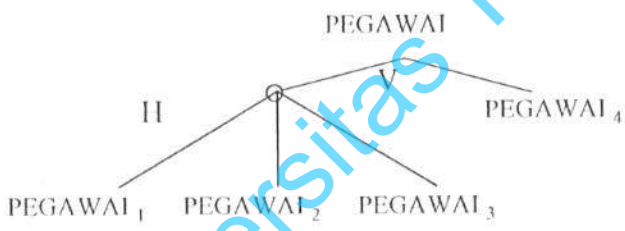
Berikut ini fragmentasi campuran yang diperoleh dengan melakukan fragmentasi vertikal seperti pada contoh sebelumnya dan diikuti oleh fragmentasi horisontal pada DEPTNOMOR seperti diperlihatkan pada Gambar3.

$$PEGAWAI_1 = SL_{DEPTNOMOR < 10}\ PJ_{PNOMOR, NAMA, MGRNOMOR, DEPTNOMOR}\ PEGAWAI$$

$$PEGAWAI_2 = SL_{DEPTNOMOR < 20}\ PJ_{PNOMOR, NAMA, MGRNOMOR, DEPTNOMOR}\ PEGAWAI$$

$$PEGAWAI_3 = SL_{DEPTNOMOR > 20}\ PJ_{PNOMOR, NAMA, MGRNOMOR, DEPTNOMOR}\ PEGAWAI$$

$$PEGAWAI_4 = PJ_{PNOMOR, NAMA, GAJI, PAJAK}\ PEGAWAI$$



Gambar 3

Pembentukan kembali relasi PEGAWAI diperoleh melalui:

$$PEGAWAI = UN(PEGAWAI_1, PEGAWAI_2, PEGAWAI_3)\ JN_{PNOMOR = PNOMOR}$$

$$PJ_{PNOMOR, GAJI, PAJAK}\ PEGAWAI_4$$

2.5 Merancang basisdata terdistribusi

Untuk merancang basisdata terdistribusi sebenarnya sangat sukar, hal ini disebabkan karena terlalu banyak masalah baik secara teknis maupun secara organisasi. Dari sudut teknis, masalah baru yang muncul adalah hubungan antar lokasi yang dilakukan oleh suatu jaringan komputer dan pendistribusian data dan aplikasi secara optimal ke lokasi-lokasi guna memenuhi kebutuhan data yang diperlukaan oleh aplikasi tersebut serta mengoptimisasikan penampilan. Dari sudut organisasi, masalah-masalah desentralisasi menjadi sangat penting karena sistem distribusi akan

menjadi sistem terpusat, dengan kata lain mendistribusikan aplikasi akan mempunyai akibat pada suatu organisasi.

2.5.1 Kerangka acuan untuk merancang

Dalam hal ini akan dikonsentrasikan pada aspek-aspek yang khas untuk basisdata terdistribusi, yang tentu saja ada kaitannya dengan basisdata terpusat. Pada rancangan basisdata terpusat, rancangan basisdata berarti:

- Merancang skema konseptual, yang menggambarkan basisdata secara terintegrasi yaitu semua data yang mana dipergunakan oleh aplikasi-aplikasi basisdata.
- Merancang basisdata phisik, yaitu pemetaan skema konseptual ke tempat penyimpanan dan menerangkan cara-cara akses yang tepat.

Dalam sistem basisdata terdistribusi, dua rancangan tersebut menjadi rancangan skema global dan rancangan basisdata phisik lokal pada masing-masing lokasi yang ada, teknik yang dipergunakan sama dengan yang dipergunakan pada basisdata terpusat.

Pada rancangan basisdata terdistribusi, di samping kedua rancangan pada basisdata terpusat, masih harus ditambah dengan rancangan yang dipergunakan untuk pendistribusian basisdata, yaitu:

- Merancang fragmentasi, yaitu melakukan fragmentasi suatu *relation* melalui fragmentasi horisontal, vertikal, atau campuran.
- Merancang pengalokasian fragmen yaitu menentukan bagaimana fragmen-fragmen hasil fragmentasi dipetakan ke image phisik.

Dalam merancang suatu basisdata terdistribusi, pengetahuan yang cukup tentang keperluan-keperluan aplikasi sangat dibutuhkan. Keperluan-keperluan tersebut adalah:

- Dari lokasi mana aplikasi tersebut dipakai.
- Seberapa sering aplikasi tersebut dipakai.
- Jumlah, tipe, dan statistik pendistribusian yang dilakukan oleh masing-masing aplikasi untuk masing-masing obyek data yang dibutuhkan.

2.5.2 Jenis rancangan pendistribusian data

Ada dua jenis rancangan yang dipergunakan untuk merancang pendistribusian data, yaitu secara *top-down* dan *bottom-up*. Dalam pendekatan *top-down*, dimulai dengan merancang skema global dan dilanjutkan dengan merancang fragmentasi basisdata, selanjutnya alokasikan fragmen

ke lokasi-lokasi yang ada dan buat image phisik, akhirnya buat rancanagan phisik dari data yang dialokasikan pada masing-masing lokasi. Jika basisdata terdistribusi yang akan dirancang dibangun berdasarkan basisdata yang sudah ada sebelumnya, maka tidaklah mudah apabila dipakai pendekatan *top-down*. Oleh karena itu untuk kasus ini dipakai suatu pendekatan *bottom-up* guna merancang pendistribusian data. Pendekatan *bottom-up* ini didasarkan pada pengintegrasian skema yang sudah ada ke dalam skema global yang tunggal.

2.5.3 Rancangan fragmentasi basisdata

Tujuan rancangan fragmentasi adalah menentukan agar supaya fragmen-fragmen, yang mengalokasikan unit-unit logik tidak saling tindih, dan akan didasarkan pada rancangan secara *top-down*. Kegiatan merancang fragmentasi terdiri dari kegiatan mengelompokkan *tuple* (dalam kasus fragmentasi horisontal) atau kegiatan mengelompokkan *attribute* (dalam kasus fragmentasi vertikal) yang mempunyai sifat-sifat yang sama dari sudut alokasinya. Masing-masing *tuple* atau *attribute* yang mempunyai sifat sama akan merupakan suatu fragmen.

2.5.3.1 Rancangan Fragmentasi horisontal

Menentukan fragmentasi horisontal suatu basisdata berarti menentukan sifat-sifat logik data dan sifat-sifat statistik data misalnya predikat-predikat fragmentasi dan jumlah aplikasi yang mengacu ke fragmen-fragmen. Dalam fragmentasi horisontal ini lapat kita bedakan antara fragmentasi horisontal yang utama dan fragmentasi horisontal yang diturunkan.

Untuk fragmentasi horisontal utama, masing-masing *tuple* dipisahkan dalam satu dan hanya satu fragmen, hal ini berarti bahwa menentukan fragmentasi horisontal utama suatu *relation* sama dengan menentukan sekumpulan predikat-predikat yang *disjoint* dan lengkap. Anggap P = (p_1, p_2, . . . , p_n) adalah himpunan predikat sederhana. Apabila akan digambarkan suatu fragmentasi yang benar dan efisien maka P haruslah lengkap dan minimal. Yang dimaksud dengan lengkap, jika dan hanya jika untuk sembarang dua *tuple* yang ada di suatu fragmen, akan direferensikan oleh suatu aplikasi dengan peluang sama, sedangkan yang dimaksud dengan minimal yaitu semua predikat relevan.

Contoh:

Anggap ada suatu *relation*:

PEMASOK(KODE,NO-LOKASI,NAMA,ALAMAT,NO-TELEPON, NAMA-PEJABAT)

Asumsikan ada aplikasi yang memerlukan informasi tentang pemasok yang berlokasi di sekitar Makassar, di samping itu ada aplikasi penting lainnya yang memerlukan data tentang pemasok-pemasok yang memproduksi suatu barang X. Aplikasi yang terakhir ini dapat ditempatkan pada suatu lokasi basisdata terdistribusi dan mereferensikan semua barang dengan peluang yang sama. Asusmsikan hanya ada dua status pemasok yaitu status 1 (swasta) dan status 2 (negeri), jadi STATUS = 1 sama dengan STATUS ≠1. Dua predikat sederhana misalnya STATUS = 1 dan produk sama dengan X. Semua predikat sederhana di atas adalah relevan, misalnya, produk dari pemasok itu lebih dari 25 jenis barang (PRODUK > 25).

Minterm untuk kedua predikat di atas adalah:

STATUS = 1 dan PRODUK = X

STATUS = 1 dan PRODUK ≠X

STATUS ≠ 1 dan PRODUK = X

STATUS ≠ 1 dan PRODUK ≠ X

Jadi P_1 = {STATUS = 1} tidak lengkap, karena aplikasi mereferensikan *tuple-tuple* barang dengan suatu peluang yang tidak sama dalam fragmen yang dihasilkan oleh P_1.

P_2 = {STATUS = 1, PRODUK = X} adalah lengkap dan minimal.

P_3 = {STATUS = 1, PRODUK = X, PRODUK > 25} lengkap tapi tidak minimal karena PRODUK > 25 adalah tidak relevan lagi.

2.5.3.2 Rancangan Fragmentasi horisontal yang diturunkan

Fragmentasi horisontal yang diturunkan suatu *relation* R tidak didasarkan pada sifat-sifat *attribute*-nya sendiri, tetapi diturunkan dari fragmentasi horisontal *relation* lainnya. Fragmentasi horisontal yang diturunkan dipergunakan untuk memudahkan hubungan antara fragmen-fragmen, jadi ide dasar adalah untuk menganggap fragmentasi yang diturunkan dari suatu *relation* merupakan alternatif (cadangan) terhadap fragmentasi horisontal yang utama.

2.5.3.3 Rancangan Fragmentasi vertikal

Apabila kita akan menentukan fragmentasi vertikal suatu *relation* R, maka perlu adanya pengelompokan *attribute-attribute* ke dalam suatu himpunan, yang mana *attribute-attribute* tersebut direferensikan dengan cara yang sama oleh suatu aplikasi. Fragmentasi vertikal dikelompokkan menjadi dua masalah, yaitu masalah partisi vertikal yang mana himpunan-

himpunan *attribute* tersebut harus saling disjoin, dan masalah pengelompokan vertikal yang mana himpunan-himpunan *attribute* tersebut dapat saling beririsan.

Syarat untuk fragmentasi vertikal adalah masing-masing *attribute* dari R dipunyai paling sedikit oleh satu himpunan dan masing-masing himpunan himpunan memasukkan kunci dari R atau suatu pengenal *tuple*. Tujuan fragmentasi vertikal adalah menentukan fragmen-fragmen sehingga beberapa aplikasi dapat dijalankan hanya dengan menggunakan satu fragmen. Anggap ada *relation* R dibagi secara vertikal ke dalam R_1 dan R_2, keuntungan akan didapat apabila suatu aplikasi dapat dijalankan hanya menggunakan R_1 atau R_2 saja, bukan dua-duanya. Disamping itu keuntungan akan didapat apabila beberapa aplikasi menggunakan R_1 dan beberapa aplikasi yang lain menggunakan R_2 yang dialokasikan pada lokasi yang berbeda.

Ada dua alternatif untuk mempartisi *attribute-attribute* yaitu:

- Pendekatan memisahkan, yang mana *relation* makin lama makin terpisah menjadi fragmen-fragmen.
- Pendekatan mengelompok, yang mana *attribute-attribute* makin lama makin mengumpul menjadi fragmen.

Pengelompokan vertikal menggunakan replikasi fragmen-fragmen karena nilai-nilai *attribute* yang saling tindih (*overlapping*) akan direplikasi. Terhadap aplikasi yang sifatnya hanya membaca saja dan aplikasi yang sifatnya memperbarui (*update*), replikasi akan memberikan efek yang berbeda. Aplikasi yang sifatnya hanya membaca saja akan memberikan keuntungan pada replikasi, karena ini lebih memungkinkan mereferensikan data secara lokal. Sedangkan untuk aplikasi yang sifatnya memperbarui, replikasi tidak cocok karena harus memperbarui semua replikasi yang ada.

2.5.3.4 Rancangan Fragmentasi campuran

Membangun fragmentasi campuran terdiri dari dua kegiatan, yaitu:

1. Menggunakan fragmentasi horisontal terhadap fragmen-fragmen yang vertikal, pada mulanya dilakukan fragmentasi vertikal dan selanjutnya dilakukan fragmentasi horisontal.
2. Menggunakan fragmentasi vertikal terhadap fragmen-fragmen yang horisontal, pada mulanya dilakukan fragmentasi horisontal dan selanjutnya dilakukan fragmentasi vertikal.

Walaupun operasi-operasi ini dapat diulang secara rekursif, tetapi kenyataannya tidak praktis jika harus dilakukan lebih dari dua tingkat fragmentasi.

2.6 Mengalokasikan fragmen-fragmen

Masalah alokasi fragmen pada basisdata terdistribusi dapat diselesaikan, cara berikut ini yang paling mudah, yaitu dengan menganggap bahwa masing-masing fragmen merupakan suatu *relation* yang terpisah.

2.6.1 Kriteria pengalokasian fragmen

Dalam mengalokasikan fragmen-fragmen, harus dibedakan apakah merancang suatu alokasi *nonredundant* atau *redundant*. Metode yang dipergunakan untuk menentukan suatu alokasi *nonredundant* adalah metode *best-fit* yaitu suatu ukuran yang dihubungkan dengan masing-masing alokasi yang mungkin, dan lokasi dengan ukuran yang paling bagus yang kita pilih. Untuk menentukan lokasi *redundant* fragmen-fragmen, salah satu dari metode berikut ini dapat dipergunakan.

- Metode *All beneficial sites*, menentukan himpunan semua lokasi yang mana benefit penempatan satu copy fragmen lebih besar dari cost, dan mengalokasikan satu copy fragmen ke masing-masing elemen himpunan ini.
- Metode *Additional replication*, menentukan solusi masalah tak terreplikasi dan memasukkan copy terreplikasi dimulai dari yang mempunyai benefit lebih besar.

2.6.2 Ukuran cost dan benefit

Akan diberikan rumus sederhana untuk mengevaluasi cost dan benefit pengalokasian fragmen suatu *relation* R, untuk itu diberikan beberapa definisi sebagai berikut.

i : indeks fragmen

j : indeks lokasi

k : indeks aplikasi

f_{ki} : frekuensi aplikasi k pada lokasi j

r_{ki} : jumlah aplikasi k yang mereferensikan pencarian ke fragmen i

u_{ki} : jumlah aplikasi k yang mereferensikan perubahan ke fragmen i

$n_{ki} = r_{ki} + u_{ki}$

2.6.3 Mengalokasikan fragmen hasil fragmentasi horisontal

Gunakan metode *best-fit* untuk suatu alokasi tanpa duplikasi, dengan penjelasan sebagai berikut:

Tempatkan R_i pada lokasi di mana jumlah referensi R_i maksimum, jumlah referensi lokal R_i pada lokasi j adalah $B_{ij} = \sum_k f_{kj}\, n_{ki}$

R_i ditempatkan pada lokasi j sehingga B_{ij} maksimum.

Gunakan metode *All beneficial sites* untuk alokasi duplikasi, dengan penjelasan sebagai berikut:

Tempatkan R_i pada semua loaksi j dimana cost aplikasi-aplikasi yang mereferensikan pencarian, lebih besar dari cost aplikasi-aplikasi pada lokasi lain yang mereferensikan perubahan ke R_i.

B_{ij} dihitung dengan cara sebagai berikut:

$$B_{ij} = \sum_k f_{kj}\, r_{ki} - cx \sum_k \sum_{j' \neq j} f_{kj'}\, u_{ki}$$

c adalah konstanta yang mengukur cost perubahan dan cost pencarian, mempunyai nilai sama atau lebih besar dari satu. R_i ditempatkan di semua lokasi j sehingga B_{ij} positif, jika B_{ij} negatif maka copy dari R_i ditempatkan pada lokasi yang membuat B_{ij} maksimum.

Gunakan metode *Additional replication* alokasi duplikasi, dengan penjelasan sebagai berikut:

Dapat diukur benefit penempatan suatu copy baru dari R_i dalam hubungan antara peningkatan ketahanan dan keberadaan sistem.

B_{ij} dihitung dengan cara sebagai berikut:

$$B_{ij} = \sum_k f_{kj}\, r_{ki} - cx \sum_k \sum_{j' \neq j} f_{kj'}\, u_{ki} + p(d_i)$$

dengan: $p(d_i) = (1 - 2^{1-di})\, F_i$

d_i : derajat *redundancy* R_i

F_i : benefit R_i, yang direplikasikan ke masing-masing lokasi

2.6.4 Mengalokasikan fragmen hasil fragmentasi vertikal

Akan diukur benefit partisi secara vertikal suatu fragmen R_i, yang dialokasikan pada lokasi r, ke dalam dua fragmen R_s dan R_t, yang dialokasikan pada s dan t.

Dengan pengaruh partisi seperti di bawah ini kita akan menghitung benefitnya.

- Ada dua himpunan aplikasi A_s dan A_t, dijalankan di lokasi s atau t, yang hanya menggunakan *attribute-attribute* R_s dan R_t dan menjadi aplikasi lokasi lokal untuk lokasi s dan lokasi t.

2.7.2 Jenis *attribute*

Attribute dapat digolongkan menjadi beberapa bagian, seperti dijelaskan berikut ini.

- *Attribute* atomic adalah *attribute* yang tidak dapat dibagi-bagi menjadi *attribute* yang lebih mendasar.
- *Attribute composite* adalah *attribute* yang terdiri dari beberapa *attribute* yang lebih mendasar.
- *Attribute single-value* adalah *attribute* yang mempunyai satuan harga untuk suatu *entity* tertentu.
- *Attribute* yang dapat terdiri dari sekumpulan harga untuk suatu *entity* tertentu.
- *Attribute null-value* adalah *attribute* yang tidak mempunyai nilai, atau *attribute* yang tidak diketahui harganya.

Identifier yang unik dari suatu *entity* disebut *attribute* kunci (*key attribute*), karena nilai dari *attribute* kunci ini akan berbeda untuk masing-masing *entity*. *Attribute* kunci dapat terdiri dari *attribute atomic* atau *composite*, tetapi sedapat mungkin diusahakan agar *attribute* kunci adalah *attribute atomic* disamping itu ada kemungkinan suatu *entity type* dapat memiliki lebih dari satu *attribute* kunci. *Attribute* kunci yang terdiri dari *attribute composite* yang kompleks, biasanya menunjukkan bahwa *entity type* harus dibagi menjadi beberapa *entity type* dengan jumlah *attribute* yang lebih sedikit.

Dalam aplikasi inventory ini ada beberapa *entity type*, *attribute*, dan *attribute key* yaitu sebagai berikut:

1. *Entity type* : AGEN
   Attribute : kode_agent, no_lokasi, nama_agent, kabu_agent, prop_agent, almt_agent
   Attribute key : kode_agent
2. *Entity type* : PEMASOK
   Attribute : kode_pmsk, no_lokasi, nama_ pmsk, almt_ pmsk, telepon_ pmsk, pjbt_ pmsk
   Attribute key : kode_pmsk
3. *Entity type* : BARANGKBL
   Attribute : no_srt_psn, no_agent, no_distr, kode_agent, kode_jenis, kode_barng, jumlah, tanggal, nm_tgjawab, jabatan, nip
   Attribute key : no_srt_psn, kode_agent
4. *Entity type* : BARANGLAKU
   Attribute : kode_agent, kode_jenis, kode_barng, jumlh_laku, tanggal

Attribute key : kode_agent

5. *Entity type* : UT
   Attribute : no_lokasi, nama_lokasi
   Attribute key : no_lokasi
6. *Entity type* : GUDANG
   Attribute : kode_jenis, kode_barng, blok, baris, rak, palet, kapasitas, jumlah, tanggal
   Attribute key : kode_barng
7. *Entity type* : MAHASISWA
   Attribute : nim_mhs, no_lokasi, nama_mhs, alamat_mhs
   Attribute key : nim_mhs
8. *Entity type* : PSKELUAR
   Attribute : kode_barng, ba
   Attribute key : kode_barng
9. *Entity type* : PSMASUK
   Attribute : no_srt_psn, nopesan, kode_jenis, kode_barng, kode_agent, jml_br_psn, tgl_sr_psn, jml_br_krm, no_srt_agn, tgl_sr_agn, notu_distr, tgl_distr, no_bauk, tgl_bauk, status, pstudi, smstr, kode_prg
   Attribute key : no_srt_psn
10. *Entity type* : TRANSMSK
    Attribute : no_srt_psn, no_srt_trs, kode_jenis, kode_barng, kode_pmsk, jml_br_msk, tgl_transk, no_faktur, tgl_trima, nama_opr.
    Attribute key : kode_pmsk, no_srt_psn
11. *Entity type* : TRANSKLR
    Attribute : no_srt_psn, kode_jenis, kode_barng, kode_agent, jml_pesan, jml_keluar, tgl_keluar, status.
    Attribute key : kode_agent, no_srt_psn
12. *Entity type* : FAKULTAS
    Attribute : kode_fak, nama_fsk
    Attribute key : kode_fak

2.7.3 Jenis *Relationship*

Apabila suatu *attribute* dari suatu *entity type* merujuk ke *entity type* lainnya, maka akan ada suatu hubungan atau yang biasa disebut dengan adanya suatu *relationship*. Untuk itu dapat didefinisikan suatu *relationship type* sebagai berikut:

Relationship type R di antara n buah *entity type* E_1, E_2, . . . , E_n adalah kumpulan dari *relationship* diantara *entity-entity* dari *entity type* tersebut. Batasan-batasan pada *relationship type* yang menyangkut *cardinality ratio*, akan dijelaskan sebagai berikut.

- *Cardinality ratio* 1 : 1 berarti satu *entity* pada *entity type* A berhubungan dengan satu *entity* pada *entity type* B dan juga sebaliknya.
- *Cardinality ratio* 1 : N berarti satu *entity* pada *entity type* A berhubungan dengan sejumlah *entity* di *entity type* B.
- *Cardinality ratio* M : N berarti sejumlah *entity* pada *entity type* A berhubungan dengan sejumlah *entity* pada *entity type* B.

III RANCANGAN BASISDATA

Istilah-istilah yang dipakai dalam aplikasi inventory ini mungkin agak berbeda dengan istilah yang dipakai dalam masalah inventory pada umumnya, istilah-istilah yang dipakai di sini hanya merupakan kesepakatan antara *end-user* dan *system analyst* saja. Dalam laporan penelitian ini akan dipakai beberapa istilah yang mana juga dipakai dalam program komputer serta tampilan yang dihasilkan.

Agen: lembaga, toko buku, perusahaan atau perorangan yang memasarkan modul bahan belajar ke mahasiswa atau individu lain. Agen dapat berlokasi di Jakarta, Surabaya, Makassar atau daerah sekitar ketiga lokasi di atas. Contoh: sampai saat ini hanya UPBJJ-UT di daerah yang berjumlah 31 dianggap sebagai agen.

Pemasok: badan usaha, perusahaan atau perorangan yang menyerahkan barang dalam hal ini modul bahan belajar ke Universitas Terbuka. Pemasok dapat berlokasi di Jakarta, Surabaya, Makassar atau daerah sekitar ketiga lokasi di atas. Contoh: percetakan ABC di Bandung, percetakan UT di Jakarta, percetakan Gramedia di Ambon, dsb.

Barang kembali: identitas dari setiap material bahan belajar yang dikembalikan ke UT karena sesuatu hal.

Barang laku: identitas dari setiap material bahan belajar yang telah laku terjual.

Jenis: identitas dari setiap material bahan belajar. Contoh: bahan belajar karunika, bahan belajar PGSD, dsb.

Gudang: kapasitas rak, palet, baris, dan blok suatu barang yang ada di gudang Jakarta, Surabaya, dan Makassar, yang mana basisdata berada.

Pesanan masuk: pesanan barang dari agen ke UT, dalam hal ini agen memesan ke lokasi UT yang terdekat. Dimungkinkan juga, bila keadaan terpaksa, agen di sekitar lokasi UT Jakarta memesan barang je lokasi Surabaya atau UT lokasi Makassar. Contoh: UPBJJ-UT Medan memesan modul Statistika Ekonomi I sebanyak 200 satuan ke UT Jakarta, UPBJJ-UT Yogyakarta memesan modul Kalkulus I sebanyak 50 satuan ke UT Surabaya.

Pesanan keluar: Pesanan barang dari UT ke pamasok sekitarnya. Dimungkinkan juga, bila keadaan terpaksa, UT lokasi Jakarta memesan barang ke pemasok yang ada di sekitar Surabaya atau UT lokasi Makassar, dan begitu juga sebaliknya. Contoh: UT lokasi Jakarta memesan 3000 modul Pancasila ke percetakan Merapi yang ada di Jakarta, UT lokasi Surabaya memesan 500 modul Pengantar Ekonomi Mikro ke percetakan Bumerang yang ada di Madiun.

Transaksi masuk: pengiriman dan pencatatan barang dari pemasok ke UT. Contoh: percetakan ABC mengirim modul Analisa Numerik sebanyak 150 satuan ke UT lokasi Surabaya, dsb.

Transaksi keluar: pengiriman dan pencatatan barang dari UT ke agen. Contoh: UT lokasi Surabaya mengirim modul Hubungan Masyarakat ke UPBJJ Surabaya sebanyak 45 satuan.

Surat pesan: surat yang berisi pesanan barang dari agen ke lokasi UT dimana saja atau dari UT ke pemasok.

Surat transaksi: surat yang berisi pengantar pengiriman barang dari UT ke agen atau dari pemasok ke UT.

Fakultas: fakultas yang ada di UT.

3.1 Rancangan *relation* untuk basisdata *relational*

Dapat dibuat suatu struktur *relation* yang mendukung aplikasi inventory ini, *relation* yang dihasilkan adalah sebagai berikut:

1. *Relation* agen, merupakan master *relation* agen, setiap *tuple* berisi identitas tetap agen serta relasi dengan perwakilan UT, mahasiswa, dan barang.
2. *Relation* pemasok, merupakan master *relation* pemasok, setiap *tuple* berisi identitas tetap pemasok serta relasi dengan perwakilan UT serta barang.
3. *Relation* barang, merupakan master *relation* barang, setiap *tuple* berisi identitas tetap barang serta relasi dengan fakultas.
4. *Relation* fakultas, merupakan master *relation* fakultas, setiap *tuple* berisi identitas tetap fakultas serta relasi dengan mahasiswa.
5. *Relation* mahasiswa, merupakan master *relation* mahasiswa, setiap *tuple* berisi identitas tetap mahasiswa serta relasi dengan fakultas dan agen.
6. *Relation* pesan masuk, merupakan *transaction relation*, setiap *tuple* berisi identitas tetap transaksi pemesanan barang dari agen ke UT.
7. *Relation* pesan keluar, merupakan *transaction relation*, setiap *tuple* berisi identitas tetap transaksi pemesanan barang dari UT ke Pemasok.
8. *Relation* lokasi, merupakan master *relation* lokasi, setiap *tuple* berisi identitas tetap lokasi yang ada di jaringan komputer.
9. *Relation* UT, merupakan master *relation* UT, setiap *tuple* berisi identitas tetap perwakilan UT yang menyimpan basisdata dan merupakan satu dari seluruh simpul yang ada di jaringan komputer.

10. *Relation* gudang, merupakan master *relation* gudang, setiap *tuple* berisi identitas tetap susunan barang dalam blok, baris, rak, dan palet tertentu yang terletak di gudang.

3.2 *Relation* dan *tuple*-nya

Ada beberapa *relation* yang mendukung aplikasi inventory ini, yaitu:

AGEN(kode_agent, no_lokasi, nama_agent, kabu_agent, prop_agent, almt_agent)

PEMASOK(kode_pmsk, no_lokasi, nama_ pmsk, almt_ pmsk, telepon_ pmsk, pjbt_ pmsk)

BARANGKBL(no_srt_psn, no_agent, no_distr, kode_agent, kode_jenis, kode_barng, jumlah, tanggal, nm_tgjawab, jabatan, nip)

BARANGLAKU(kode_agent, kode_jenis, kode_barng, jumlh_laku, tanggal)

FAKULTAS(kode, nama)

MAHASISWA(nim, no-lokasi, nama, alamat)

LOKASI(no_lokasi, nama_lokasi)

GUDANG(blok, baris, rak, palet, kode-barang)

UT(kode,no_lokasi, nama, alamat)

PESAN MASUK(no_srt_psn, nopesan, kode_jenis, kode_barng, kode_agent, jml_br_psn, tgl_sr_psn, jml_br_krm, no_srt_agn, tgl_sr_agn, notu_distr, tgl_distr, no_bauk, tgl_bauk, status, pstudi, smstr, kode_prg)

PESAN KELUAR(kode_barng, ba)

IV HASIL PENELITIAN DAN ANALISIS

Di bawah ini akan dibahas mengenai fragmentasi beberapa *relation* yang mendukung aplikasi inventory ini serta pengalokasian fragmen-fragmen hasil fragmentasi tersebut. Dalam mengalokasikan fragmen, ada beberapa hal yang perlu diperhatikan yaitu adanya beberapa asumsi. Asumsi tersebut yaitu, baik pemasok maupun agen akan berada di masing-masing lokasi atau tersebar pada sekitar masing-masing lokasi, dan tidak ada duplikasi data untuk *relation* yang difragmentasi pada masing-masing lokasi. Karena dalam aplikasi ini diasumsikan hanya mempunyai tiga lokasi dan program aplikasinya termasuk sederhana, maka teori alokasi yang dibahas di bab 2 tidak secara mendetail diterapkan di sini.

4.1 Fragmentasi data

Dalam rancangan aplikasi inventory ini digunakan fragmentasi horisontal, hal ini disebabkan karena *relation* dapat dipartisi menjadi beberapa subset *tuple* dan masing-masing subset mempunyai ciri yang sama secara geografis. Ciri yang sama dalam hal ini adalah nilai yang mungkin dari *attribute* lokasi hanyalah Jakarta (no_lokasi = 1), Surabaya (no_lokasi = 2), dan Makassar (no_lokasi = 3). Sehingga apabila dilakukan fragmentasi horisontal maka aturan kelengkapan, aturan rekonstruksi, dan aturan disjoint akan selalu dipenuhi.

Untuk *relation* PEMASOK, dilakukan fragmentasi horisontal yaitu masing-masing *tuple* atau *attribute* yang mempunyai sifat sama akan merupakan suatu fragmen. Dalam hal ini sifat yang sama yaitu keberadaan pemasok tersebut di sekitar suatu lokasi UT tertentu, oleh karena itu fragmentasi PEMASOK adalah sebagai berikut:

$$PEMASOK_1 = SL_{NO\text{-}LOKASI = 001}(PEMASOK)$$

$$PEMASOK_2 = SL_{NO\text{-}LOKASI = 002}(PEMASOK)$$

$$PEMASOK_3 = SL_{NO\text{-}LOKASI = 003}(PEMASOK)$$

Untuk *relation* AGEN, juga dilakukan fragmrntasi horisontal sebagaimana halnya *relation* PEMASOK, oleh karena itu masing-masing *tuple* atau *attribute* yang mempunayi sifat sama akan merupakan suatu fragmen. Dalam hal ini sifat yang sama yaitu keberadaan agen tersebut di sekitar suatu lokasi UT tertentu, oleh karena itu fragmentasi AGEN adalah sebagai berikut:

$$AGEN_1 = SL_{NO\text{-}LOKASI = 001}(AGEN)$$

$$AGEN_2 = SL_{NO\text{-}LOKASI = 002}(AGEN)$$

$$AGEN_3 = SL_{NO\text{-}LOKASI = 003}(AGEN)$$

Untuk *relation* UT perlu dilakukan fragmentasi horisontal sebagai berikut:

$$UT_1 = SL_{NO\text{-}LOKASI = 001}(UT)$$

$$UT_2 = SL_{NO\text{-}LOKASI = 002}(UT)$$

$$UT_3 = SL_{NO\text{-}LOKASI = 003}(UT)$$

Untuk *relation* MAHASISWA dilakukan fragmentasi horisontal sebagai berikut:

$$MAHASISWA_1 = SL_{NO\text{-}LOKASI = 001}(MAHASISWA)$$

$$MAHASISWA_2 = SL_{NO\text{-}LOKASI = 002}(MAHASISWA)$$

$$MAHASISWA_3 = SL_{NO\text{-}LOKASI = 003}(MAHASISWA)$$

Ada beberapa *relation* yang tidak perlu dilakukan fragmentasi karena masing-masing lokasi memerlukan *relation* tersebut. Apabila *relation* tersebut dilakukan fragmentasi dan ditempatkan pada lokasi yang ada maka akan terlalu banyak akses yang dilakukan oleh suatu lokasi ke lokasi yang lain. *Relation* tersebut adalah: BARANGKBL, BARANGLAKU, FAKULTAS, MAHASISWA, LOKASI, GUDANG, UT, PESAN MASUK, PESAN KELUAR.

Nilai yang mungkin dari *attribute* lokasi yaitu 001, 002, dan 003, untuk aturan pembentukan kembali dapat dilakukan operasi seperti di bawah ini:

$$PEMASOK = PEMASOK_1 \text{ UN } PEMASOK_2 \text{ UN } PEMASOK_3$$

$$AGEN = AGEN_1 \text{ UN } AGEN_2 \text{ UN } AGEN_3$$

$$UT = UT_1 \text{ UN } UT_2 \text{ UN } UT_3$$

$$MAHASISWA = MAHASISWA_1 \text{ UN } MAHASISWA_2 \text{ UN } MAHASISWA_3$$

4.2 Alokasi basisdata

Alokasi yang dimaksud di sini yaitu selain mengalokasikan fragmen-fragmen hasil fragmentasi *relation* yang ada juga akan dialokasikan *relation-relation* yang mendukung aplikasi inventory ini. Untuk itu akan ditulis kandungan *relation* untuk masing-masing lokasi. Pada aplikasi ini rancangan alokasi yang akan diterapkan adalah alokasi *nonredundant* karena tidak adanya duplikasi data pada masing-masing lokasi. Oleh karena itu pendekatan yang digunakan adalah pendekatan *best-fit*.

$$B_{ij} = \sum_{k} f_{kj}\, n_{ki}$$

Formula untuk pendekatan *best-fit* adalah sebagai berikut:

Sesuai dengan yang telah dijelaskan di bab2 yaitu:

i : indeks fragmen

j : indeks lokasi

k : indeks aplikasi

f_{kj} : frekuensi aplikasi k pada lokasi j

r_{ki}: jumlah aplikasi k yang mereferensikan pencarian ke fragmen i

u_{ki} : jumlah aplikasi k yang mereferensikan perubahan ke fragmen i

$n_{ki} = r_{ki} + u_{ki}$

Untuk masing-masing fragmen berlaku penjelasan sebagai berikut:

i = 1, 2, 3 dan j = 1, 2, 3; besarnya k sembarang karena berapapun besarnya k tidak akan berpengaruh pada masing-masing lokasi.

Contoh perhitungan B_{ij} untuk lokasi 1 (Jakarta):

$$B_{11} = f_{11}\, n_{11} + f_{21}\, n_{21} + \ldots + f_{k1}\, n_{k1}$$

$$B_{21} = f_{11}\, n_{12} + f_{21}\, n_{22} + \ldots + f_{k1}\, n_{k2}$$

$$B_{31} = f_{11}\, n_{13} + f_{21}\, n_{23} + \ldots + f_{k1}\, n_{k3}$$

f_{kj} adalah frekuensi aplikasi (proses) k pada lokasi j, contoh:

f_{11} adalah frekuensi aplikasi i pada lokasi 1 (Jakarta)

f_{13} adalah frekuensi aplikasi i pada lokasi 3 (Makassar)

Karena pada aplikasi inventory ini kita asumsikan bahwa masing-masing proses mempunyai frekuensi *run* yang sama dalam kurun waktu tertentu, maka jelas nilai fkj untuk j = 1, 2, 3 dan k = 1, 2, . . . , z besarnya akan sama untuk masing-masing lokasi. Karena diasumsikan bahwa di sekitar masing-masing lokasi terdapat beberapa pemasok dan agen, yang berarti bahwa pemasok dan agen tidak terpusat di suatu lokasi tertentu saja, maka u_{ki} dan r_{ki} untuk i = 1, 2, 3 dan k = 1, 2, 3, . . ., z akan mempunyai nilai yang sama. Karena $n_{ki} = r_{ki} + u_{ki}$, maka n_{ki} akan mempunyai nilai sama untuk masing-masing variabel i. Hal ini mengakibatkan:

$$B_{ij} = \sum_{k} f_{kj}\, n_{ki}$$

untuk i = 1, 2, 3 dan j = 1, 2, 3 akan mempunyai nilai yang sama terhadap variabel i, yaitu $B_{1j} = B_{2j} = B_{3j}$, maka jelas tidak ada B_{ij} yang maksimum. Oleh karena itu R_1 harus kita alokasikan di j =

1, R_2 harus kita alokasikan di j = 2, R_3 harus kita alokasikan di j = 3. Sebagai contoh, *relation* PEMASOK difragmentasikan menjadi PEMASOK1, PEMASOK2, dan PEMASOK3. PEMASOK1 ditempatkan di lokasi Jakarta, PEMASOK2 ditempatkan di lokasi Surabaya, PEMASOK3 ditempatkan di lokasi Makassar. Alokasi fragmen hasil fragmentasi dan *relation-relation* yang mendukung aplikasi inventory ini adalah sebagai berikut:

Lokasi 1: Perwakilan UT di Jakarta.

Di lokasi ini akan dialokasikan fragmen PEMASOK1, fragmen AGEN1, fragmen UT1, fragmen MAHASISWA1, *relation* BARANG, *relation* PESAN-MASUK, *relation* PESAN-KELUAR, *relation* LOKASI, *relation* FAKULTAS, *relation* GUDANG.

Lokasi 2: Perwakilan UT di Surabaya.

Di lokasi ini akan dialokasikan fragmen PEMASOK2, fragmen AGEN2, fragmen UT2, fragmen MAHASISWA2, *relation* BARANG, *relation* PESAN-MASUK, *relation* PESAN-KELUAR, *relation* LOKASI, *relation* FAKULTAS, *relation* GUDANG.

Lokasi 2: Perwakilan UT di Makassar.

Di lokasi ini akan dialokasikan fragmen PEMASOK3, fragmen AGEN3, fragmen UT3, fragmen MAHASISWA3, *relation* BARANG, *relation* PESAN-MASUK, *relation* PESAN-KELUAR, relation LOKASI, *relation* FAKULTAS, *relation* GUDANG.

V KESIMPULAN

Suatu aplikasi dengan basisdata terpusat dapat dibuat menjadi aplikasi dengan basisdata terdistribusi, yaitu dengan cara melakukan fragmentasi dan alokasi data pada basisdata yang sudah dibuat secara terpusat. Tidak ada jumlah lokasi ideal yang dapat dipakai untuk menyimpan basisdata secara terdistribusi, hal ini sangat tergantung pada biaya transportasi dalam jaringan dengan pengertian secara geografis. Rancangan aplikasi inventory dengan basisdata terdistribusi hanya sebagian dari dokumen yang harus dibuat dalam pengembangan suatu perangkat lunak, dokumen yang dimaksud adalah dokumen yang dihasilkan pada tahap perencanaan, analisis, perancangan, implementasi, testing, dan maintenance.

Basisdata yang dipergunakan dalam rancangan aplikasi inventory ini adalah basisdata terdistribusi, yang penting terhadap pertimbangan mengenai penghematan, hal-hal yang berhubungan dengan organisasi, dan juga secara teknologi. Basisdata terdistribusi dapat diimplementasikan dalam jaringan komputer berskala besar maupun kecil, juga merupakan pengembangan teknologi yang dipakai dalam sistem basisdata terpusat. Merancang basisdata terdistribusi memerlukan 4 tahapan, yaitu merancang entity type, merancang fragmentasi, merancang alokasi fragmen, dan merancang struktur physic pada masing-masing lokasi. Rancangan terhadap entity type dan rancangan struktur physic mempergunakan cara yang sama sebagaimana merancang skema konseptual dan skema physical dalam basisdata terpusat, dalam hal rancangan perbedaan itu terletak pada masalah fragmentasi dan alokasi fragmen.

Apabila sudah mengenal basisdata terpusat, maka dalam basisdata terdistribusi ada aspek baru yang penting yaitu disebut data terdistribusi transparan, yang berarti bahwa program aplikasi yang dibuat tidak menganggap bahwa basisdata yang ada dalam kenyataannya adalah terdistribusi. Jadi kebenaran suatu program aplikasi tidak dipengaruhi oleh perpindahan data antara lokasi yang satu ke lokasi yang lain, akan tetapi kecepatan pemroses n yang dipengaruhi. Tentu saja menyediakan distribusi transparan untuk aplikasi-aplikasi yang sifatnya melakukan perbaikan data, akan lebih rumit bila dibandingkan dengan aplikasi-aplikasi yang sifatnya hanya membaca saja. Hal ini disebabkan karena aplikasi-aplikasi yang sifatnya melakukan perbaikan dapat mengubah nilai-nilai *attribute*.

Dalam rancangan aplikasi inventory ini digunakan fragmentasi horisontal, karena *entity type* dapat dipartisi menjadi beberapa subset *tuple* dan masing-masing subset mempunyai ciri

yang sama secara geografis. Ciri yang sama dalam hal ini adalah nilai yang mungkin dari attribute lokasi hanyalah Jakarta, Surabaya, dan Makassar. Sehingga apabila dilakukan fragmentasi horisontal maka aturan kelengkapan, aturan rekonstruksi, dan aturan disjoint akan selalu dipenuhi.

Tujuan utama pengalokasian fragmen adalah untuk meminimalkan jumlah akses jarak jauh yang dilakukan oleh aplikasi-aplikasi yang mendukung aplikasi inventory ini. Pada aplikasi inventory rancangan alokasi yang akan diterapkan adalah alokasi *nonredundant* karena tidak adanya duplikasi data pada masing-masing lokasi, oleh karena itu pendekatan yang digunakan adalah pendekatan *best-fit*.

VI DAFTAR PUSTAKA

1. Ceri S., Pelagatti G., *Distributed database Principles And System*, McGraw-Hill, USA, 1984.
2. Elmasri R., Navathe S. B., *Fundamentals of Database Systems*, The Benjamin/Cummings Publishing Comp. Inc. California, 1989.
3. Korth H. F., Silberscathz A., Database System Concepts, McGraw-Hill, New York, 1989.
4. Whitten, J. L., Bentley, L. D., V. M., *Systems Analysis & Design Methods*, Irwin, Boston, MA, 1989.
5. P. H. Enslow, Jr., *What is a Distributed Data Processing System ?*, Computer, Vol. 11, No. 1, Jan. 1978, pp. 13-21.
6. B. H. Liebowitz and J. H. Carson, *Tutorial on Distributed Processing*, 2nd ed., IEEE Cat. No. EHO 127-1, pp. 1-3.
7. Muthalib A., Rakun R., *Arsitektur untuk basisdata terdistribusi*, PAU Ilmu Komputer UI, 1991.