DATABASE TERDISTRIBUSI
(DISTRIBUTED DATABASE= DDB)

PENDAHULUAN

CERI :

" A distributed DB is a collection of data which belong logically to the same system but are spread over the sites of a computer network"

OZSU :

"A distributed DB is a collection of multiple, logically interrelated DB distributed over a computer network"

KORTH :

"A DDB system consists of a collection of sites, each of which maintains a local database system"

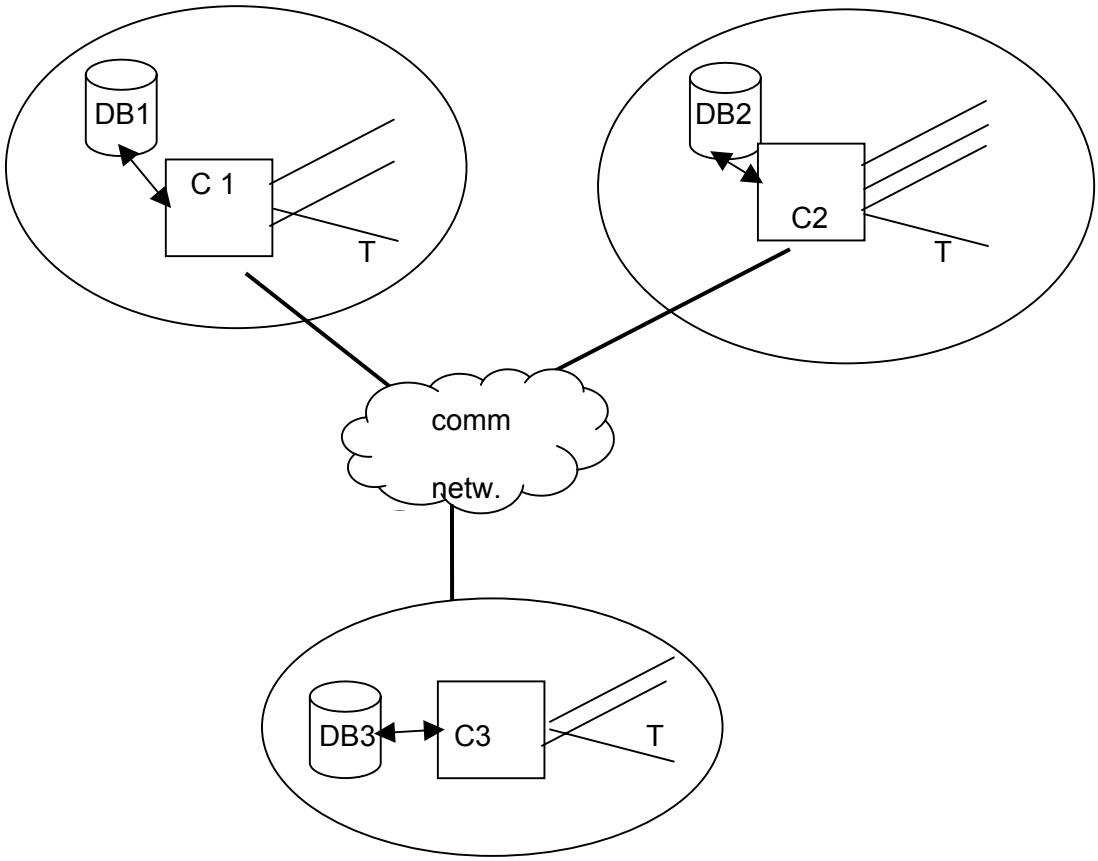
Dalam sebuah database terdistribusi, database disimpan pada beberapa komputer. Komputer-komputer dalam sebuah sistem terdistribusi berhubungan satu sama lain melalui bermacam-macam media komunikasi seperti high-speed buses atau telephone line.

Sebuah sistem database terdistribusi berisikan sekumpulan site, di mana tiap-tiap site dapat berpartisipasi dalam pengeksekusian transaksi-transaksi yang mengakses data pada satu site atau beberapa site. Tiap-tiap site dapat memproses transaksi lokal yaitu sebuah transaksi yang mengakses data pada satu site di mana transaksi telah ditentukan.

Sebuah site juga dapat mengambil bagian dalam mengeksekusi transaksi global yaitu transaksi yang mengakses data pada site yang berbeda di mana transaksi telah ditentukan, atau transaksi yang mengakses data pada beberapa site yang berbeda.

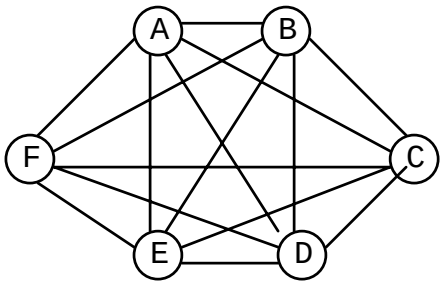
Ada 2 aspek penting dari DDB :

1. Distribusi : data tidak disimpan pada tempat (prosesor) yang sama, sehingga DDB dapat dibedakan dari database tunggal, sentralisasi
2. Korelasi logika : data memiliki property yang berhubungan sehingga DDB dapat dibedakan dari sekumpulan database local atau file yang disimpan pada tempat yang berbeda pada jaringan komputer.

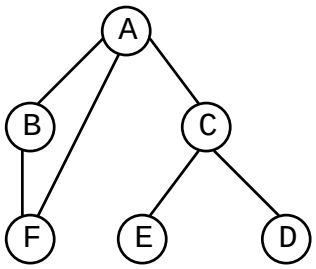


Gambar 1. Database terdistribusi secara geografis

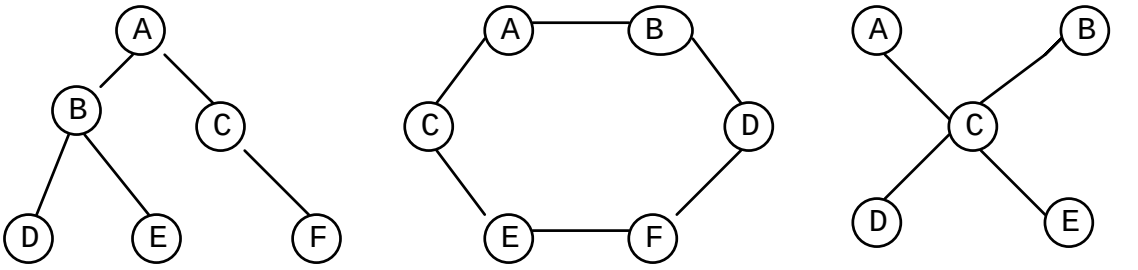
Site-site dalam database terdistribusi dihubungkan secara fisik dengan berbagai cara. Beberapa topologi digambarkan sebagai sebuah graph yang simpul-simpulnya bersesuaian dengan site. Sebuah edge dari simpul A ke simpul B bersesuaian dengan sebuah hubungan langsung antara dua site. Beberapa konfigurasi (bentuk) digambarkan sebagai berikut:



Fully connected network



Partially connected network



Tree structured network Ring network Star network

Gambar 2. Topologi network

Fully Connected network :
Keuntungan : kalau salah satu node rusak, yang lainnya masih dapat berjalan (tetapi biaya mahal).
Kerugian : control management tidak terjamin

Partially connected network :
Keuntungan : reliability rendah, biaya dapat ditekan
Kerugian : control management tidak terjamin

Tree structure network :
Keuntungan : bersifat sentral, control management lebih terjamin
Kerugian : kalau node pusat (A) rusak, semua akan rusak.
Cat : setiap proses dimulai dari bawah.

Ring Network (LAN) :
Keuntungan : rusak satu, yang lain masih berjalan
Kerugian : Control management kurang terjamin karena bersifat desentralisasi

Star Network (LAN) :
Keuntungan : - control management lebih terjamin, karena bersifat sentral
- reliability rendah
Kerugian : kalau pusat rusak, yang lainnya rusak

SistemManajemen Database Terdistribusi (Distributed DBMS) merupakan sistem software yang dapat memelihara DDBS dan transparan ke user.

DDBS bukan merupakan kumpulan dari file yang dapat disimpan tersendiri di setiap node dari jaringan komputer. Untuk membentuk DDBS, file tidak seharusnya berelasi secara logika saja, tetapi perlu ada struktur di antara file dan akses data bukan merupakan hal yang khusus.

Keuntungan dari DDBS

- Otonomi local : karena data didistribusikan, user dapat mengakses dan bekerja dengan data tersebut sehingga memiliki kontrol local.

- Meningkatkan kinerja : karena setiap site menangani hanya bagian dari DB, CPU dan I/ O tidak seberat seperti DB pusat. Data yang dipakai untuk transaksi disimpan dalam beberapa site, sehingga eksekusi transaksi dapat secara parallel.
- Meningkatkan reliability/ availability : jika satu site mengalami crash, dapat membuat beberapa site tidak dapat diakses. Jika data direplikasi ke banyak site, kerusakan hubungan komunikasi tidak menjadikan sistem total tidak dapat dioperasikan.
- Ekonomis : dari biaya komunikasi, baik membagi aplikasi dan memproses secara local di setiap site. Dari biaya komunikasi data, akan lebih murah untuk memelihara sistem komputer dalam satu site dan menyimpan data secara local.
- Expandibility : akan lebih mudah mengakomodasikan ukuran DB yang semakin besar. Ekspansi dapat dilakukan dengan menambah proses dan kekuatan penyimpanan ke jaringan.
- Shareability : jika sistem informasi tidak terdistribusi, akan sulit untuk berbagi data dan sumber daya. Sistem DB terdistribusi memungkinkan hal ini.

Kerugian dari DDBS
- Kurangnya pengalaman : sistem DB terdistribusi bertujuan umum (general-purpose) tidak sering digunakan. Yang digunakan adalah sistem prototype yang dibuat untuk satu aplikasi (misal : reservasi pesawat)
- Kompleksitas : masalah DDBS lebih kompleks dibandingkan dengan manajemen database terpusat
- Biaya : sistem terdistribusi membutuhkan tambahan hardware (untuk mekanisme komunikasi) sehingga biaya hardware meningkat. Yang terpenting pada biaya ini adalah replikasi. Jika fasilitas komputer dibuat di banyak site, akan memerlukan orang2 yang memelihara fasilitas tersebut
- Kontrol distribusi : sebelumnya menjadi keuntungan. Tetapi karena distribusi menyebabkan masalah sinkronisasi dan koordinasi, kontrol terdistribusi menjadi kerugian atau kekurangan di masalaha ini.
- Keamanan : akan mudah mengontrol database yang terpusat. Dalam sistem database terdistribusi, jaringan membutuhkan keamanan tersendiri.
- Perubahan yang sulit : tidak ada tool atau metodologi untuk membantu user mengubah database terpusat ke database terdistribusi.

PERANCANGAN DATABASE TERDISTRIBUSI

Alokasi data

Ada beberapa alternatif dasar untuk menyimpan atau menempatkan data : partisi dan replikasi. Dalam skema partisi, database dibagi ke dalam sejumlah partisi yang disjoin yang masing2 ditempatkan di site yang berbeda. Perancangan replikasi dibedakan atas fully replication atau fully duplicated dimana seluruh database ditempatkan di masing2 site, atau partially replicated yaitu masing2 partisi dari database disimpan di lebih dari satu site tetapi tidak di semua site.

Ada 2 perancangan dasar yaitu fragmentasi, pemisahan database ke dalam partisi2, disebut fragment, dan distribusi.

Fragmentasi
Relasi dibagi ke dalam beberapa fragment, masing2 disimpan di site yang berbeda. Ada 2 strategi, yaitu fragmentasi horizontal dan vertikal.

Fragmentasi horizontal
Fragmentasi berdasarkan tupel. Setiap fragment memilki subset dari tupel relasi.

Relasi r dibagi ke dalam sejumlah subset r_1, r_2,.., r_n, masing2 berisi dari sejumlah tupel relasi r. Masing2 tupel relasi r harus merupakan satu dari fragment2 tersebut sehingga relasi awalnya dapat dibentuk kembali. Suatu fragmen didefinisikan sebagai seleksi pada relasi global r. Sebuah predikat P_i digunakan untuk menyusun fragmen r_i :

$$r_i = \sigma_{Pi}(r)$$

Pembentukan kembali dilakukan dengan menggabungkan seluruh fragment :

$$R = \bigcup_{i=1}^{n} r_i$$

Fragmentasi vertikal
Fragmentasi vertikal dari r(R) melibatkan beberapa subset R_1, R_2, …, R_n dari R sedemikian sehingga

$$\bigcup_{i=1}^{n} R_i = R$$

Setiap fragment r_i dari r didefinisikan sebagai :

$$r_i = \Pi_{Ri}(r)$$

Pembentukan kembali dengan menggunakan join natural : $r = r_1 |x| r_2 |x| \ldots |x| r_n$
Fragmentasi vertikal dibuat dengan menambahkan atribut khusus yaitu tuple-id, yang merupakan alamat fisik atau logika untuk tupel dan menjadi kunci pada skema. Tetapi tuple-id tidak diperlihatkan ke user.

CONTOH

Deposit

Branch-name	Account-number	Customer-name	balance
Hillside	305	Lowman	500
Hillside	226	Camp	336
Valleyview	117	Camp	205
Valleyview	402	Kahn	10000
Hillside	155	Kahn	62
Valleyview	408	Kahn	1123
Valleyview	639	Green	750

Fragmentasi horizontal

$deposit_1 = \sigma_{\text{branch-name = "Hillside"}}$ (deposit)

Branch-name	Account-number	Customer-name	balance
Hillside	305	Lowman	500
Hillside	226	Camp	336
Hillside	155	Kahn	62

$deposit_2 = \sigma_{\text{branch-name = "Valleyview"}}$ (deposit)

Branch-name	Account-number	Customer-name	balance
Valleyview	117	Camp	205
Valleyview	402	Kahn	10000
Valleyview	408	Kahn	1123
Valleyview	639	Green	750

Fragmentasi vertikal

Deposit' = Deposit-scheme U tuple-id

Branch-name	Account-number	Customer-name	balance	Tuple-id
Hillside	305	Lowman	500	1
Hillside	226	Camp	336	2
Valleyview	117	Camp	205	3
Valleyview	402	Kahn	10000	4
Hillside	155	Kahn	62	5
Valleyview	408	Kahn	1123	6
Valleyview	639	Green	750	7

Deposit-scheme-3 = (branch-name, customer-name, tuple-id)
Deposit-scheme-4 = (account-number, balance, tuple-id)

$deposit_3 = \Pi_{\text{deposit-scheme-3}}$ (deposit')

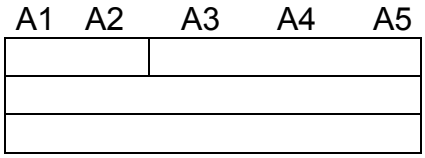
Branch-name	Customer-name	Tuple-id
Hillside	Lowman	1
Hillside	Camp	2
Valleyview	Camp	3
Valleyview	Kahn	4
Hillside	Kahn	5
Valleyview	Kahn	6
Valleyview	Green	7

$deposit_4 = \Pi_{deposit\text{-}scheme\text{-}4}$ (deposit')

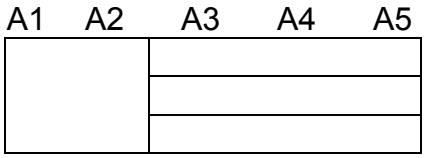
Account-number	balance	Tuple-id
305	500	1
226	336	2
117	205	3
402	10000	4
155	62	5
408	1123	6
639	750	7

Pembentukan kembalinya $\Pi_{deposit\text{-}scheme}$ ($deposit_3$ |x| $deposit_4$)

Fragmentasi campuran
Terdiri dari :
1. Mengaplikasikan fragmentasi horizontal terhadap fragment vertikal



2. Mengaplikasikan fragmentasi vertikal terhadap fragment horizontal



$Deposit_{3a} = \sigma_{branch\text{-}name = \text{“Hillside”}}$ ($deposit_3$)
$Deposit_{3b} = \sigma_{branch\text{-}name = \text{“Valleyview”}}$ ($deposit_3$)

Relasi r dibagi ke dalam 3 fragment $deposit_{3a}$, $deposit_{3b}$ dan $deposit_4$. Masing2 disimpan di site yang berbeda.

REPLIKASI
Jika relasi r direplikasi, salinan dari relasi r disimpan di 2 atau lebih site. full replication menyimpan salinan di setiap site dari sistem.

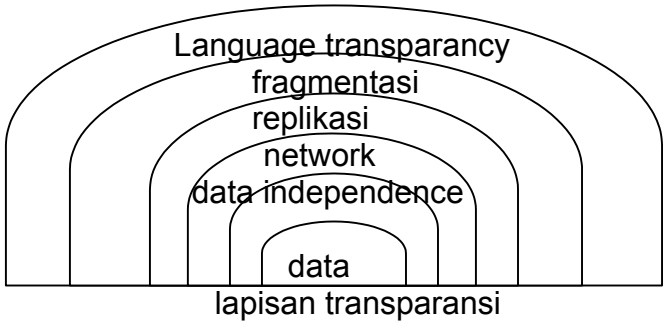
Fragmen dapat direplikasi. Misal ada sistem terdistribusi terdiri dari S_1, S_2, …, S_{10}. Salinan fragment $deposit_{3a}$ pada site S_1, S_3 dan S_7. Salinan $deposit_{3b}$ pada site S_7 dan S_{10} dan salinan $deposit_4$ ada pada site S_2, S_8 dan S_9.

Keuntungan dan kerugian dari replikasi
- Availability : jika satu site yang berisi relasi r gagal, relasi r masih didapat di site yang lain. Sistem dapat melanjutkan proses meskipun satu site mengalami kegagalan.

- Meningkatkan parallel : beberapa site dapat memproses query terhadap r secara parallel. Semakin banyak ada replikasi, semakin besar kesempatan data yang dibutuhkan ditemukan pada site dimana transaksi dijalankan. Replikasi data meminimalkan pergerakan data di antara site.
- Meningkatkan overhead update : sistem harus memastikan bahwa semua replikasi dari relasi r konsisten. Karena kalau tidak, akan terjadi kesalahan komputasi. Di mana pun r di-update, update ini harus disebar ke seluruh site. replikasi meningkatkan kinerja operasi baca dan meningkatkan availability pembacaan data. Transaksi update meningkatkan overhead. Masalah pengontrolan konkurensi update data yang direplikasi semakin kompleks dari pendekatan terpusat. Cara sederhana adalah membuat salinan utama dari r. Misal : di sistem perbankan, rekening dapat dihubungkan dengan site dimana rekening tersebut dibuka.

TRANSPARANSI PADA DDBMS

Merupakan pemisahan dari semantic level tingkat tinggi dari implementasi level rendah. Atau sistem transparansi menyembunyikan rincian implementasi dari user.



Tidak mudah dalam menggambarkan tingkat transparansi yang jelas. Dengan adanya transparansi bahasa sebagai lapisan generik, membuat user memiliki akses terhadap data bertingkat tinggi (4^{th} GL, GUI, akses bahasa natural, dll).

Bentuk2 transparansi

- *Data independence*

Kebebasan data menjadi bentuk dasar transparansi yang terlihat di DBMS. Kebebasan data berarti kekebalan dari aplikasi user untuk mengubah definisi dan organisasi data dan sebaliknya.

Definisi data dapat terjadi dalam 2 tahap. Pada satu level, struktur logika dari data dispesifikasikan (schema definition) dan pada level yang lain struktur fisik dari data didefinisikan (physical data description).

Ada 2 tipe kebebasan data

1. Kebebasan data secara logic : kekebalan aplikasi user untuk mengubah struktur logika database. Secara umum, jika aplikasi user dioperasikan pada sebuah subset atribut relasi, tidak ada pengaruh yang terjadi jika atribut baru ditambahkan ke relasi yang sama.
2. Kebebasan data secara fisik : berhubungan dengan penyembunyian rincian struktur penyimpanan dari aplikasi user. Saat aplikasi user ditulis, rincian dari

organisasi data secara fisik tidak perlu diperhatikan. Organisasi data apa pun dapat dipakai.

- *Transparansi jaringan/ terdistribusi*

Dalam sistem terpusat, hanya satu sumber yang dipelihara user yaitu data (sistem penyimpanan). Dalam manajemen DB terdistribusi, ada sumber kedua yagn perlu dipelihara yaitu jaringan. User perlu dilindungi dari detail operasi jaringan. Tidak perlu ada perbedaan antara aplikasi database yang berjalan di DB terpusat dan DB terdistribusi. Akan baik jika memiliki keseragaman dalam operasi yang diakses.
Jika ingin mengkopi sebuah file, perintah yang digunakan seharusnya sama untuk pengkopian dalam satu mesin atau antar mesin yang terhubung dengan jaringan. Sayangnya banyak OS untuk jaringan yang belum menyediakan transparansi ini.

cp <source file> <target file> ; satu mesin

rcp <machine-name:source file> <machine-name:target file> ; mesin berbeda

Transparansi lokasi merupakan transparansi terhadap perintah yang bebas digunakan pada lokasi data maupun pada sistem dimana operasi berjalan.

Transparansi penamaan (naming tranparancy) berarti nama yang unik diberikan ke setiap objek database. Caranya dengan menambahkan nama lokasi (identifier) sebagai bagian dari nama objek. Sistem yang bertanggungjawab memberikan penamaan terhadap objek agar menjadi unik.

- *Transparansi replikasi*

Data yang mudah diakses oleh user dapat ditempatkan pada mesin local user seperti mesin user lain dengan akses data yang sama. Jika satu mesin gagal, salinan data masih ada pada lokasi mesin yang lain dalam jaringan. Keputusan direplikasikan atau tidak, dan berapa banyak salinan dari objek database, bergantung pada tingkatan aplikasi user. Tetapi replikasi menyebabkan masalah dalam meng-update DB. Jika apllikasi user berorientasi pada update, sebaiknya tidak perlu ada banyak salinan data.

- *Transparansi fragmentasi*

Fragmentasi dapat mengurangi efek negatif dari repllikasi. Setiap salinan tidak merupakan salinan penuh tetapi hanya sebuah subset, jadi ruang yang dibutuhkan lebih sedikit dan item data lebih sedikit untuk dipelihara.
Saat objek DB difragment, masalah menangani queri user timbul. Masalah menangani strategi memproses queri berdasarkan fragment dibandingkan relasi meskipun query dibuat kemudian. Bentuk translasi ini disebut sebagai query global ke beberapa query fragment.

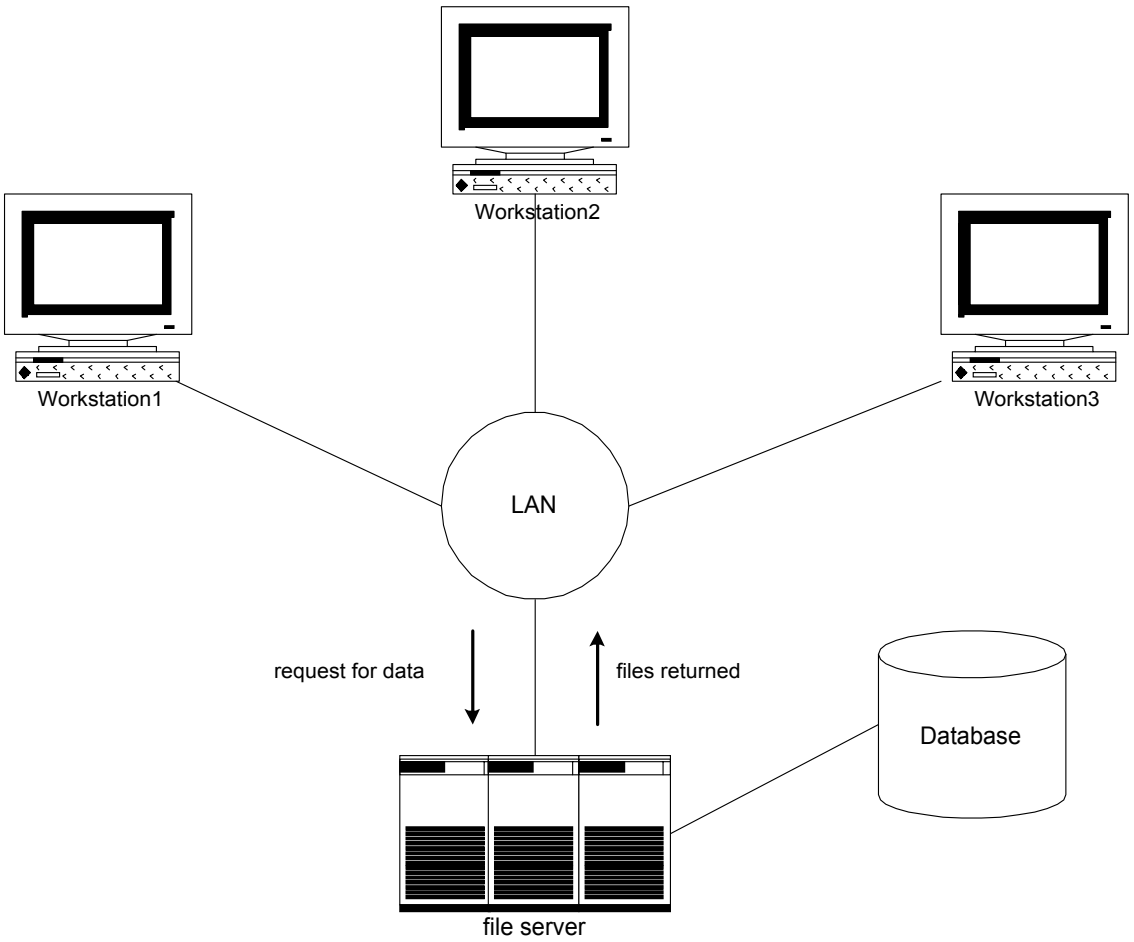
- *Siapa yang menyediakan transparansi*

Penyediaan transparansi perlu melihat lapisan transparansi. Ada 3 lapisan transparansi yang berbeda yang dapat saling menguntungkan secara eksklusif dalam penyediaan service meskipun lebih sesuai untuk ditinjau sebagai tambahan. Tanggungjawab dalam menyediakan akses data yang transparan

adalah akses lapisan. Transparansi dimulai dari transparansi bahasa yang menterjemahkan service yang diminta ke operasi yang dibutuhkan. Compiler atau interpreter mengambil alih tugas dan tidak ada service transparan disediakan untuk compiler atau interpreter. Lapis kedua adalah transparansi di level sistem operasi. Penyediaan akses transparan ke sumber daya di level OS diperluas ke distribusi, dimana manajemen jaringan diambil alih oleh OS terdistribusi. Sayangnya tidak semua OS memiliki manajemen jaringan ini. Lapis ketiga ada pada DBMS. DBMS bertindak sebagai operasi yang terintegrasi dan sistem manajemen DB. Yang khas adalah pembuatan DBMS pada komputer general-purpose yang berjalan di beberapa OS. Pada lingkungan ini, transparansi dan dukungan fungsi DB yang disediakan untuk perancang DBMS sangat minimal dan khas ke operasi dasar untuk menjalankan tugas khusus.

DBMS bertanggungjawab untuk membuat semua translasi yang berarti dari OS ke interface user yang lebih tinggi.

CLIENT-SERVER



Gambar 3. File Server

Di lingkungan file-server, pemrosesan didistribusikan ke jaringan yang Local Area Network (LAN). File-Server menunjang kebutuhan file dengan aplikasi-aplikasi dan DBMS. Aplikasi dan DBMS bekerja pada masing-masing workstation.

Permintaan file dari file-server diilustrasikan pada gambar di atas. File-server bertindak sebagai sebuah drive hard-disk yang dapat dipakai bersama. DBMS pada masing - masing workstation mengirimkan permintaan ke file-server untuk seluruh data yang dibutuhkan DBMS yang disimpan di disk.

Contoh :

```
SELECT fname, lname
FROM b, staff s
WHERE b.bno = s.sno AND b.street = '163 Main St' ;
```

File-server tidak mempunyai pengetahuan SQL, DBMS harus meminta file-file untuk berhubungan dengan relasi Branch dan Staff dari file-server.

Kerugian arsitektur file-server :

1. Adanya lalulintas network.
2. Salinan DBMS dibutuhkan pada masing - masing workstation.
3. Concurency, recovery dan integrity kontrol lebih kompleks karena pada multiple DBMS mengakses file yang dapat dipakai bersama.

Referensi

1. Ceri, Stefano & Pelagatti G, *Distributed Databases : Principles & Systems*, McGraw-Hill, Singapore, 1984
2. Korth, H.F & Siberschatz, *Database System Concepts*, McGraw-Hill, USA, 1986
3. Öszu, M.T & Valduriez, *Principles of Distributed Database Systems*, Prentice-Hall,New Jersey, 1991