

## DATABASE TERDISTRIBUSI

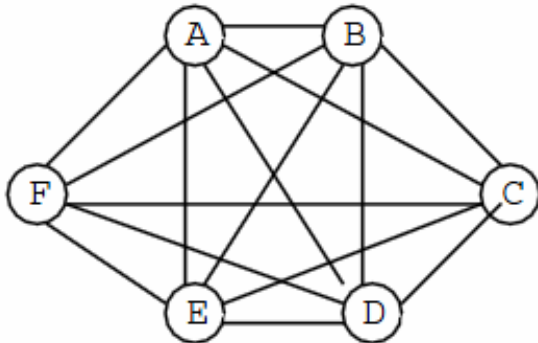
Yaitu kumpulan data yang digunakan bersama yang saling terhubung secara logic tetapi tersebar secara fisik pada suatu jaringan computer.

### Karakteristik database terdistribusi yaitu :

1. Kumpulan data yang digunakan bersama secara logic tersebar pada sejumlah computer yang berbeda
2. Komputer yang dihubungkan menggunakan jaringan komunikasi.
3. Data pada masing-masing situs dapat menangani aplikasi-aplikasi local secara otonom.
4. Data pada masing situs di bawah kendali satu DBMS.
5. Masing-masing DBMS berpartisipasi dalam sedikitnya satu aplikasi global.

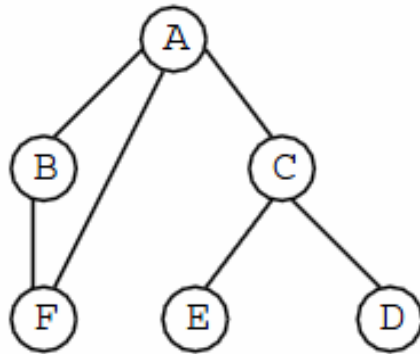
### Bentuk-bentuk Topologi Distribusi Data :

- a. Fully Connected Network



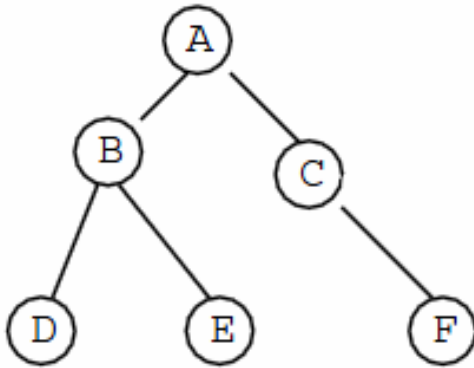
Kalau salah satu node rusak, yang lainnya masih dapat berjalan (biaya mahal), kontrol manajemen tidak terjamin.

b. Partially Connected Network



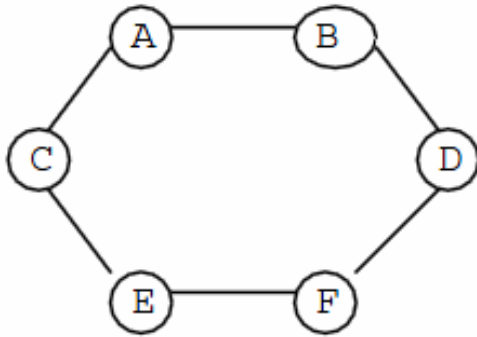
Reliability rendah, biaya dapat ditekan  
Kontrol manajemen tidak terjamin.

c. Tree Structured Network



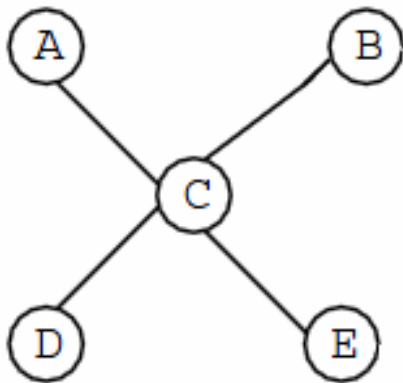
Bersifat sentral, control manajemen lebih terjamin  
Kalau node pusat rusak, semua akan rusak. (setiap proses dimulai dari bawah).

d. Ring Network



Rusak satu, yang lain masih berjalan  
Kontrol manajemen kurang terjamin karena bersifat desentralisasi.

e. Star Network



Rusak satu, yang lain masih berjalan  
Kontrol manajemen kurang terjamin karena bersifat desentralisasi.

### KEUNTUNGAN DATA BASE TERDISTRIBUSI

1. Pengawasan distribusi dan pengambilan data

Jika beberapa site yang berbeda dihubungkan, seorang pemakai yang berada pada satu site dapat mengakses data pada site lain.

Contoh : sistem distribusi pada sebuah bank memungkinkan seorang pemakai pada salah satu cabang dapat mengakses data cabang lain.

2. Reliability dan availability

Sistem distribusi dapat terus menerus berfungsi dalam menghadapi kegagalan dari site sendiri atau mata rantai komunikasi antar site.

3. Kecepatan pemrosesan query

Contoh : jika site-site gagal dalam sebuah sistem terdistribusi, site lainnya dapat melanjutkan operasi jika data telah direplikasi pada beberapa site.

4. Otonomi lokal

Pendistribusian sistem mengijinkan sekelompok individu dalam sebuah perusahaan untuk melatih pengawasan lokal melalui data mereka sendiri. Dengan kemampuan ini dapat mengurangi ketergantungan pada pusat pemrosesan.

5. Efisiensi dan fleksibel

Data dalam sistem distribusi dapat disimpan dekat dengan titik dimana data tersebut dipergunakan. Data dapat secara dinamik bergerak atau disain, atau salinannya dapat dihapus.

## **KERUGIAN DATABASE TERDISTRIBUSI**

1. Harga software mahal

Hal ini disebabkan sangat sulit untuk membuat sistem database distribusi.

2. Kompleksitas

Site-site beroperasi secara paralel sehingga lebih sulit untuk menjamin kebenaran dan algoritma. Adanya kesalahan mungkin tak dapat diketahui.

3. Biaya pemrosesan tinggi

Perubahan pesan dan penambahan perhitungan dibutuhkan untuk mencapai koordinasi antar site.

4. Sulit menjaga keutuhan data

Banyaknya pengaksesan data membuat kurangnya sekuritas terhadap data yang telah terdistribusi.

5. Kurangnya standar

tidak ada tool atau metodologi untuk membantu user mengubah database terpusat ke database terdistribusi.

6. Kurang pengalaman

sistem DB terdistribusi bertujuan umum (generalpurpose) tidak sering digunakan. Yang digunakan adalah sistem prototype yang dibuat untuk satu aplikasi (misal : reservasi pesawat)

7. Perancangan basis data lebih kompleks

Sebelumnya menjadi keuntungan. Tetapi karena distribusi menyebabkan masalah sinkronisasi dan koordinasi, kontrol terdistribusi menjadi kerugian atau kekurangan di masalah ini.

## FRAGMENTASI DATA

Adalah relasi dipartisikan ke dalam beberapa bagian, setiap bagian disimpan pada lokasi yang berbeda

Alasan-alasan diperlukannya fragmentasi, yaitu :

1. Penggunaan

umumnya aplikasi bekerja dengan tabel views dibandingkan dengan semua hubungan data. Oleh karenanya untuk distribusi data , yang cocok digunakan adalah bekerja dengan subset dari sebuah relasi sebagai unit dari distribusi.

2. Efisiensi

data disimpan dekat dengan yang menggunakan. Dengan tambahan data yang tidak sering digunakan tidak usah disimpan.

3. Pararelisme

dengan fragmen-fragmen tersebut sebagai unit dari suatu distribusi , sebuah transaksi dapat di bagi kedalam beberapa sub queri yang dioperasikan pada fragmen tersebut. Hal ini meningkatkan konkurensi atau paralelisme dalam sistem, sehingga memeperbolehkan transaksi mengeksekusi secara aman dan paralel.

4. Keamanan

data yang tidak dibutuhkan oleh aplikasi tidak disimpan dan konsukuen tidak boleh di ambil oleh pengguna yang tidak mempunyai otoritas.

Kerugian fragmetasi yaitu :

1. Kinerja

cara kerja dari aplikasi yang membutuhkan data dari beberapa lokasi fragmen di beberapa situs akan berjalan dengan lambat.

## 2. Integritas

pengawasan integritas akan lebih sulit jika data dan fungsional ketergantungan di fragmentasi dan dilokasi pada beberapa situs yang berbeda.

Beberapa peraturan yang harus diidentifikasi ketika mendefinisikan fragment :

### 1. Kondisi lengkap

Jika relasi contoh R di dekomposisi ke dalam fragment  $R_1, R_2, R_3, \dots, R_n$ , masing-masing data yang dapat ditemukan pada relasi R harus muncul paling tidak di salah satu fragmen. Aturan ini diperlukan untuk meyakinkan bahwa tidak ada data yang hilang selama fragmentasi

### 2. Rekonstruksi

Jika memungkinkan untuk mendefinisikan operasional relasi yang akan dibentuk kembali relasi R dari fragmen-fragmen.

Aturan ini untuk meyakinkan bahwa fungsional ketergantungan di perbolehkan

### 3. Disjointness

Jika item data  $d_i$  muncul pada fragment  $R_i$ , maka tidak boleh muncul di fragmen yang lain. Vertikal fragmentasi diperbolehkan untuk aturan yang satu ini, dimana kunci utama dari atribut harus diulangi untuk melakukan rekonstruksi. Aturan ini untuk meminimalkan redundansi.

Tiga Jenis Fragmentasi :

### 1. Fragmentasi horizontal

Fragmentasi berdasarkan tupel. Setiap fragment memiliki subset dari tupel relasi.

Relasi r dibagi ke dalam sejumlah subset  $r_1, r_2, \dots, r_n$ , 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  $r_i$  :

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

Pembentukan kembali dilakukan dengan menggabungkan seluruh fragment :

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

## 2. Fragmentasi vertical

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

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

Setiap fragment  $r_i$  dari  $r$  didefinisikan sebagai :

$$r_i = \pi_{R_i}(r)$$

Pembentukan kembali dengan menggunakan join natural :  $r = r_1 \bowtie r_2 \bowtie \dots \bowtie r_n$

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

$$\text{deposit}_1 = \sigma_{\text{branch-name} = \text{"Hillside"}} (\text{deposit})$$

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

$$\text{deposit}_2 = \sigma_{\text{branch-name} = \text{"Valleyview"}} (\text{deposit})$$

| 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

$$\text{Deposit}' = \text{Deposit-scheme U tuple-id}$$

| 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        |

$$\text{Deposit-scheme-3} = (\text{branch-name, customer-name, tuple-id})$$

$$\text{Deposit-scheme-4} = (\text{account-number, balance, tuple-id})$$

$$\text{deposit}_3 = \Pi_{\text{deposit-scheme-3}} (\text{deposit}')$$

| 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  $\Pi_{deposit\text{-}scheme} (deposit_3 \bowtie deposit_4)$

### 3. Fragmentasi campuran

Cara yang sederhana untuk membangun fragmentasi campuran adalah :

- a. Menggunakan fragmentasi horizontal pada fragmentasi vertical

| A1 | A2 | A3 | A4 | A5 |
|----|----|----|----|----|
|    |    |    |    |    |
|    |    |    |    |    |
|    |    |    |    |    |

- b. Menggunakan fragmentasi vertical pada fragmentasi horizontal

| A1 | A2 | A3 | A4 | A5 |
|----|----|----|----|----|
|    |    |    |    |    |
|    |    |    |    |    |
|    |    |    |    |    |

$Deposit_{3a} = \sigma_{branch\text{-}name = \text{"Hillside"}} (deposit_3)$

$Deposit_{3b} = \sigma_{branch\text{-}name = \text{"Valleyview"}} (deposit_3)$

Relasi r dibagi ke dalam 3 fragment deposit 3a, deposit 3b dan deposit 4, masing-masing disimpan pada site yang berbeda.