Model Hierarchie dan Network

Model data hierarchie, network dan relational adalah contoh dari model data berbasis record (record based model data). Model data ini bersifat logical yang merupakan model implementasi dari model Entity-Relationship yang bersifat konseptual (belum dapat diimplementasi). Pada bagian ini yang akan dibahas adalah model data hierarchie dan network.

1. **Model Hierarchie**

Pada model hierarchie data direpresentasikan dengan kumpulan records dan relationships diantara data representasikan dengan *links*. Model hierarchie adalah model data yang berbasiskan tree dimana keterhubungan antar data bersifat terbatas one-to-many, dimana seorang parent mempunyai banyak son. Misalkan pada system perbankan seorang customer mempunyai banyak account, dan suatu account hanya dimiliki oleh seorang customer.

**Struktur data**

Sebagai ilustrasi pada masalah perbankan tentang simpan, seorang customer mempunyai beberapa account dan sebuah account dapat dimiliki oleh beberapa customer dengan struktur record berikut:

|  |  |
| --- | --- |
| **type** *customer* = **record**  *customer name*: string;  *customer street*: string;  *customer city*: string;  **end** | **type** *account* = **record**  *account number*: string;  *balance*: integer;  **end** |

dimana *record type* diwakili dengan kotak yang berisi atribut dan garis diwakili *relationship*. Sebagai ilustrasi untuk masalah perbankan tersebut diberikan instances berikut:

|  |
| --- |
| Model dasar Hierarchie  Customer  Account  M  1  Nama  No\_Acc  Street  Kota  Balance |
| Instances data: |

Pada dunia nyata sering dijumpai keterhubungan many-to-many, misalkan aturan tentang customer-account tersebut di atas diperluas menjadi seorang customer mempunyai banyak account, dan suatu account dapat dimiliki oleh lebih dari seorang customer. Hal ini dapat terjadi pada persoalan join account diantara suami istri atau antar orang tua dan anak dst. Bagaimana model data hierarchie ini menangani keterhubungan many-to-

many juga akan menjadi bahasan pada bagian ini. Pada permasalahan nyata, pemodelan hierarchie akan berupa kumpulan pohon-pohon, seperti yang diilustrasikan di bawah ini.

**. . . .**

P

Q

M

1

P1

Q1

P2

Q3

Q2

C

1

A

B

M

1

A1

B1

A2

A3

B2

M

C1

C2

**. . . .**

**1.1 Transformasi model data E-R ke model Hierarchie**

1.1a) One-to-one atau one-to-many

|  |  |
| --- | --- |
| Model E-R | Model Hierarchie |
| R1  1  1  A  B  M  1  A  R1  B | atau  B  A  M  1  A1  A2  B1  B2  B3  B  A  M  1  A1  A2  B1  B2  B3  A  B  M  1  A1  A2  B1  B2  B3 |

1.1b) one-to-many dari entity set A ke B dengan descriptive attribute

|  |  |
| --- | --- |
| Model E-R | Model Hierarki |
| M  1  A  R1  B | B  R1  A  M  1  M  1  A1  A2  B1  B2  B3  Z |

1.1c) Many-to-many dari entity set A ke B

|  |  |
| --- | --- |
| Model E-R | Model Hierarki |
| R1  N  M  A  B | A  B  M  1  B1  B2  B3  A1  A2  B  A  M  1  A1  A2  B1  B2  B3  dan |

1.1d) Many-to-many dari entity set A ke B dengan descriptive attribute

|  |  |
| --- | --- |
| Model E-R | Model Hierarki |
| B  A  M  N  R1 | R1  A  B  M  1  M  1  B1  B2  B3  A1  A2  Z  R1  B  A  M  1  M  1  A1  A2  B1  B2  B3  Z  dan |

Tampak dari dua transformasi terakhir, dapat (dan sering terjadi duplikasi data) ketika memodelkan persoalan nyata yang sering cukup kompleks.

**Penggunaan virtual record**

Telah diperlihatkan ketika memodelkan persoalan yang cukup kompleks (misalkan many-to-many) secara model hierarchie akan memunculkan masalah duplikasi data yang masif. Hal ini bukan saja menghabiskan banyak ruang, tetapi juga memperumit pengendalian konsistensi. Pada operasi manipulasi data misalkan mengubah atau menghapus data pada suatu record type tertentu maka operasi tersebut harus dilakukan di semua duplikasinya. Pada prinsipnya virtual record menggantikan duplikasi fisik record, sehingga hanya akan ada satu record fisik untuk semua record type dan satu atau beberapa virtual records untuk record type yang diduplikasi. Sebagai ilustrasi gambar di bawah ini pada sebelah kiri diambil dari transformasi 1.1c), sedangkan yang sebelah kanan adalah penggunaan virtual records.

B1

B2

B3

B

A

B

M

1

B1

B2

B3

A1

A2

A2

A1

A

B

A

M

1

A1

A2

B1

B2

B3

1

1

A

M

M

B

Dengan virtual records maka perubahan data dilakukan hanya pada record fisiknya. sehingga memudahkan penjagaan konsistensi. Untuk masalah penghapusan data selain penghapusan pada record fisik juga mensetup pointer yang menunjuknya menjadi null, atau yang lebih sederhana adalah menggunakan perubahan status saja sebagai “deleted”.

1. **Model Network**

Pada model network, seperti halnya model hierarchie, data direpresentasikan dengan kumpulan records dan relationships diantara data representasikan dengan *links*. Bedanya dengan model hierarchie, pada model network berbasiskan graph dimana keterhubungan antar data tidak lagi bersifat terbatas pada hierarki one-to-many (1-M) tetapi sudah bersifat many-to-many (M-N) sehingga dalam pemodelan dunia nyata lebih natural.

Pada bagian ini akan dibahas hal-hal tentang

* Struktur data
* Penelusuran Data (retrieval)
* Manipulasi Data

dengan studi kasus masalah perbankan tentang simpanan.

**Struktur data**

Sebagai ilustrasi pada masalah perbankan tentang simpan, seorang customer mempunyai beberapa account dan sebuah account dapat dimiliki oleh beberapa customer dengan struktur record berikut:

|  |  |
| --- | --- |
| **type** *customer* = **record**  *customer name*: string;  *customer street*: string;  *customer city*: string;  **end** | **type** *account* = **record**  *account number*: string;  *balance*: integer;  **end** |

dimana *record type* diwakili dengan kotak yang berisi atribut dan garis diwakili *relationship*. Sebagai ilustrasi untuk masalah perbankan tersebut diberikan instances berikut:



Rectype LastAccess

Rectype Customer

Rectype Account

Tampak bahwa terjadi keterhubungan many-to-many antara customer dengan account, dalam contoh di atas Johnson memiliki account A-102 dan A-201 sedangkan account A102 dimiliki oleh Hayes dan Johnson. Record tentang tanggal diantara customer dan account adalah tentang misalkan kapan terakhir akses.

Untuk pembahasan selanjutnya pemodelan akan terlihat sebagai record type dan keterhubungan antar record type mengadopsi model yang direkomendasi oleh DBTG CODASYL yang merupakan salah satu Database Task Group yang ada saat itu. Aturan yang digunakan pada model CODASYL adalah tentang transformasi, DBTG sets dan data retrieval sebagai berikut.

* + 1. **Transformasi E-R ke model network:**

2.1.1a) One-to-one atau one-to-many dari entity set A ke B menghasilkan model network yang “sama”.

|  |  |
| --- | --- |
| Model E-R | Model Network |
| B  A  1  1  R1  M  1  A  R1  B | atau  B  A  M  1  A1  A2  B1  B2  B3  B  A  B  A  A1  A2  B1  B2  B3  1  M  M  1  A1  A2  B1  B2  B3 |

2.1.1b) one-to-many dari entity set A ke B dengan descriptive attribute

|  |  |
| --- | --- |
| Model E-R | Model Network |
| B  R1  A  1  M | B1  B2  B3  A2  A1  B  A  1  1  M  M  R1  Z |

2.1.1c) Many-to-many dari entity set A ke B

|  |  |
| --- | --- |
| Model E-R | Model Network |
| R1  N  M  A  B | B  A  M  1  M  1  A1  A2  B1  B2  B3    RLink |

2.1.1d). Many-to-many dari entity set A ke B dengan descriptive attribute

|  |  |
| --- | --- |
| Model E-R | Model Network |
| B  A  M  N  R1 | B  A  M  1  M  1  A1  A2  B1  B2  B3  Z  R1 |

Exercise

* Berikan contoh instances data dari model E-R dan model network pada transformasi 2.1.1d
* Transformasikan model data model E-R berikut ke model network

1

R1

N

M

A

B

C

**2.1.2. DBTG Set**

Jika diperhatikan pada aturan transformasi model data E-R ke model network maka akan tampak bahwa walaupun model network ini sudah mengakomodasi keterhubungan many-to-many secara natural namun pada model network DBTG hanya ada keterhubungan one-to-many. Artinya model data network DBTG mengimplementasi network dengan menggunakan prinsip hierarchie juga. Hal ini direalisasikan dengan tersedianya DBTG set. DBTG set adalah himpunan pasangan parent-son berikut:

Set name is <nama\_set>

owner is <record\_type>

member is <record\_type>

Contoh:



Jumlah set=2

Set name is depositor set name is account\_branch

owner is customer owner is branch

member is account member is account

**Penunjuk Kekinian:**

Seperangkat penunjuk ke berbagai catatan database yang paling baru diakses oleh program aplikasi; Beberapa jenis penunjuk kekinian adalah dari jenis berikut:

* Current dari record type:

Satu penunjuk kekinian untuk setiap jenis record direferensi oleh program aplikasi;

setiap penunjuk berisi alamat (lokasi pada disk) dari rekaman yang paling baru diakses dari typeT

* Current dari set\_type

Satu penunjuk kekinian untuk setiap set type yang direferensikan oleh program aplikasi; setiap penunjuk berisi alamat dari record yang paling baru diakses dari set type tersebut;

perhatikan bahwa penunjuk ini dapat menunjuk ke record dari owner atau member, tergantung pada apakah owner atau member paling baru diakses

* Current run unit

Satu penunjuk tunggal kekinian berisi alamat dari rekaman (terlepas dari jenisnya) yang terakhir diakses oleh program aplikasi

**Status flag**

Sekumpulan variabel yang digunakan oleh sistem untuk berkomunikasi dengan program aplikasi hasil dari operasi terakhir yang diterapkan ke database, yang paling sering digunakan adalah DB\_status, disetel ke 0 jika operasi terbaru berhasil dan sebaliknya disetel ke kode kesalahan. Variabel status tambahan (DB-set-name, DB-record-name, andDB-data-name) ditetapkan ketika operasi terakhir gagal, untuk membantu mengidentifikasi sumber kesulitan.

**2.1.3 Data Retrieval**

Dua perintah yang paling sering digunakan pada DBTG adalah:

* **find**,memposisikan pada record di database dan menyetel pointer2 kekinian
* **get**, mengcopy record dimana current of run unit menunjuk dari database ke template program work area

Pada data retrieval dapat disimpulkan dari struktur model data network adalah bahwa karena data terorganisasi dalam kumpulan hierarchie (tree) maka akan diperlukan penelusuran data:

* Penelusuran data pada suatu record type tertentu
* Penelusuran data antar record\_type pada tree dari suatu set (pasangan parent-son)

**Penelusuran pada suatu record type**

* find any <record\_type> using <field> : menempatkan pointer ke record yang memenuhi kondisi yang diinginkan dimulai dari awal
* find duplicate <record\_type> using <field>: menempatkan pointer ke record selanjutnya dari current pointer yang memenuhi kondisi yang diinginkan

**Penelusuran antar record type pada suatu tree**

* find first <record\_type> within <set\_type>: menempatkan pointer ke record type dari record\_type parent posisi current dari set\_type tertentu
* find next <record\_type> within <set\_type>: menempatkan pointer ke record type setelah posisi current yang merupakan <record\_type> son selanjutnya dari set\_type.

Penelusuran dari record type member ke parent

* find owner within*<*set-type*>*

Untuk penelusuran menyeluruh dari awal sampai habis dari suatu record type, diperlukan mekanisme tambahan berupa set dimana owner adalah system dan member adalah record type yang ingin ditelusuri.

Misalkan diinginkan menelusuri semua record dari customer

set name is *AllCust*

owner is *system*

member is *customer*

Selanjutnya dibuat perintah network retrieval berikut

find first *customer* within *AllCust*;

while *DB-status* = 0 do

begin

get *customer*

print (*customer*.*customer name*);

find next *customer* within *AllCust*;

end

Contoh

Diberikan suatu permasalahan perkuliahan yang disederhanakan seperti berikut:

|  |  |
| --- | --- |
| AmKlh  N  M  MHS  KLH |  |

* Transormasikan ke model network
* Berapa jumlah pointer
* Buat perintah network retrieval untuk:
  + Mendapatkan nim mahasiswa yang bernama “Muntu”
  + Mendapatkan kuliah yang diambil oleh “Amir”
  + Mendapatkan nim dan nama mahasiswa yang mendapatkan nilai “A” di matakuliah database.

Jawab:

* Hasil transformasi dari model E-R ke model network adalah
* Jumlah pointer:

Pointer record type + pointer set type + pointer run unit = 3+2+1=6

* Perintah network retrieval mendapatkan nim mahasiswa bernama “Muntu”

Mhs.nama = ”Muntu”

find any Mhs using nama

while DB\_status = 0 do

begin

get Mhs

print (Mhs.nim)

find duplicate Mhs using nama

end

* Mendapatkan kuliah yang diambil oleh mahasiswa ber-nim “NIM-7”

Mhs.nim = ”NIM-7”

find any Mhs using nim

while DB\_status = 0 do

begin

get (Mhs)

find first AmbilKlh within MhsAmbilKlh

while(DB\_status=0) do

begin

find owner within KlhAmbilKlh

get (Klh)

print (Klh.kd\_klh, nm\_klh)

find next AmbilKlh

end

find duplicate Mhs using nim ----tdk diperlukan

end

Kisi2 UTS

* + Menjawab pertanyaan yang sama dgn perintah yang berbeda, yaitu buatlah perintah penelusuran dimulai dari Klh
  + Dan lakukan analisis kinerjanya