BAB II

BASIS DATA RELASIONAL

Sistem basis data dirancang untuk menampung informasi data dengan kapasitas yang besar. Manajemen data meliputi struktur untuk penyimpanan informasi, mekanisme kelengkapan dan manipulasi informasi. Sistem basis data harus memberikan keamanan untuk penyimpanan informasi-informasi.

Suatu basis data yang memiliki relasi berupa skema atau tabel didefinisikan sebagai basis data relasional. Untuk model relasionalnya didefinisikan sebagai suatu cara untuk melihat data mulai dari struktur logika, integritas data sampai pada manipulasi data.

Struktur basis data didasari konsep model data. Kemudian basis data relasional dapat dirancang menurut konsep model data tersebut.

2.1 MODEL DATA

Model data menurut Naptalie Rishe (1992) adalah suatu ketentuan dari ketetapan konsep dunia nyata dalam bentuk yang dapat dimengerti oleh Sistem Manajemen Basis Data (SMBD).

Berikut macam-macam model data menurut N. Rishe (1992) dan D. M. Kroenke (1990) :

a. **Biner semantik**, yang mana informasi disajikan dengan relasi logik antara pasangan-pasangan obyek dan dengan klasifikasi obyek ke dalam kategori.

b. **Relasional**, yang mana informasi disajikan dengan kumpulan tabel. Model data ini didefinisikan pada tahun 1970, tetapi memiliki arti penting yang kecil sampai dihasilkan SMBD relasional yang cocok pada awal tahun 1980. Perkembangan model data relasional antara tahun 1970 dan 1980 ini telah memperbanyak pengertian dan teori dari model data. Yang kemudian, model data relasional merupakan hal penting untuk perancangan basis data dan implementasi dari basis data yang diproses secara langsung oleh pengguna.

c. **Network**, yang mana informasi disajikan dengan suatu graph dari record. Model data network dengan mudah menyajikan relasi one-to-many. Dan dapat digunakan secara langsung untuk menyajikan semua tipe obyek, kecuali gabungan obyek many-to-many. Setiap obyek harus diubah ke dalam relationship one-to-many.

d. **Hierarki**, yang mana informasi disajikan sebagai pohon (*tree*) dari record. Semua relationship data harus diubah ke dalam hierarki atau pohon sebelum dapat didefinisikan ke dalam basis data. Meskipun mungkin untuk mengubah beberapa struktur obyek ke dalam hierarki, transformasi kadang-kadang harus direncanakan.

2.2 ATRIBUT

Atribut didefinisikan sebagai relasi fungsional (m:1 atau 1:1) yang range-nya adalah kategori konkrit.

Contoh 2-1 : nama-depan – atribut dari ORANG, range: String (m:1)

Suatu atribut *A* dikatakan time-invariant jika terdapat satu kali obyek *x* direlasikan dengan *A* pada *y*, maka obyek *x* akan selalu direlasikan dengan *A* pada *y*, sepanjang *x* ada.

Sekalipun hukum alam atau sosial tidak memperbolehkan suatu atribut untuk berubah pada saatnya, atribut mungkin dirasakan berubah pada dunia nyata untuk menemukan kesalahan dalam persepsi yang lebih cepat. Sebagai contoh, Nomor KTP dapat dilaporkan dengan salah dan kemudian dikoreksi. Selanjutnya, time-invariant didefinisikan hanya pada pembatas implementasional. Seperti pembatas-pembatas yang tidak dapat dihindarkan pada rancangan basis data relasional. Metodologi rancangan skema relasional bertujuan untuk meminimalkan efek negatif dari pembatas implementasional.

Contoh 2-2 :

Berikut atribut yang diubah hanya ketika ditemukan kesalahan

tahun-lahir dari ORANG

tahun dan waktu dari SEMESTER

Berikut ini atribut yang mungkin kadang-kadang berubah menurut perubahan pada waktu. Kemudian sebagai time-invariant akan menjadi pembatas implementasional yang lebih kuat.

nama-akhir dan nama-awal dari ORANG

nama dari MATAKULIAH

Dan di bawah ini atribut mempunyai kemungkinan besar berubah menurut waktu; tidak ada kelayakan rancangan basis data akan dibatasi sebagai time-invariant.

alamat dari ORANG

kelas-akhir dari PENDAFTARAN

2.3 KUNCI (*KEY*)

Kunci adalah kumpulan dari satu atau lebih atribut yang secara kolektif memberikan identitas yang unik pada suatu entity dari kumpulan entity.

Kunci untuk keperluan kali ini, dibagi menjadi dua, yaitu :

1. Kunci Atribut-tunggal (*Single-attribute Key*)

Sebuah atribut time-invariant dari sebuah kategori dikatakan kunci jika 1:1 dan total. Ini berarti bahwa nilai dari atribut dapat digunakan untuk mengidentifikasikan obyek dari kategori.

Contoh 2-3 :

Jika diasumsikan suatu atribut nama dari MATAKULIAH adalah time-invariant, maka mungkin juga suatu kunci yang :

a. total, basis data ini tidak diinginkan berisi mata kuliah tanpa nama.

b. 1:1, tidak ada dua mata kuliah mempunyai nama yang sama dan tidak ada mata kuliah yang mempunyai dua nama.

Pada teks ini, nama atribut dibatasi menjadi kunci dengan akhiran *–kunci*.

Contoh 2-4 :

nama-*kunci* dari MATAKULIAH

Nama dari atribut nama-*kunci* secara mutlak mendefinisikan suatu pembatas atau pembatas implementasional "atribut ini merupakan kunci dari domainnya, kategori MATAKULIAH".

2. Kunci Atribut-ganda (*Multiattribute Key*)

Berikut diberikan definisi konsep kunci untuk kumpulan dari atribut.

Kunci dari kategori adalah kumpulan dari total atribut time-invariant $f_1, f_2, \ldots, f_n$ yang merupakan domain kategori tersebut dan memenuhi dua persyaratan berikut:

i. Untuk beberapa kumpulan nilai $x_1, \ldots, x_n$ yang tidak lebih dari satu obyek y dari kategori *s.t.*

$$x_1 = y.f_1 \text{ dan } x_2 = y.f_2 \text{ dan } \ldots \text{ dan } x_n = y.f_n$$

Ini berarti bahwa kumpulan atribut cukup untuk mengidentifikasikan setiap obyek dari kategori.

ii. Tidak ada sub-kumpulan yang tepat dari atribut ini selalu memenuhi i

Ini berarti bahwa kumpulan adalah minimal; jika satu dari atribut tidak diketahui maka sisa atribut tidak mungkin memberikan informasi yang cukup untuk mengidentifikasikan setiap obyek dari kategori.

Contoh 2-5 :

(tahun, waktu) hanya merupakan kunci dari kategori SEMESTER.

Atribut yang lain tidak mengidentifikasikan masing-masing obyek.

Pada teks ini, kategori dibatasi untuk mempunyai tepat satu kunci dan kunci diubah dari beberapa atribut, nama atribut ini diberi akhiran *–kunci*.

Contoh 2-6 :

tahun-*kunci* dan waktu-*kunci* dari SEMESTER

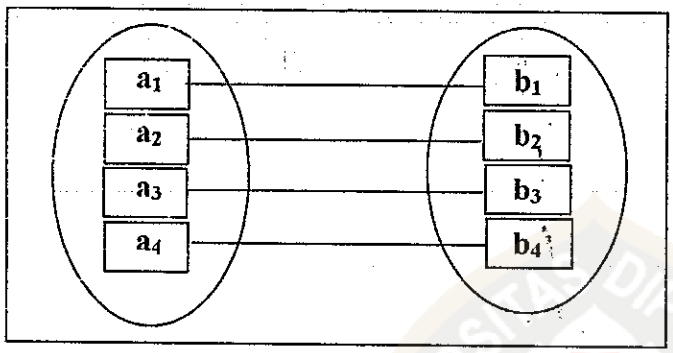
2.4 RELATIONSHIP

Relationship adalah gabungan antara beberapa entity. Himpunan relationship (*relationship set*) adalah kumpulan relationship dari tipe yang sama. Yang dimaksud dengan entity adalah suatu obyek yang nyata dan dapat dibedakan dari obyek-obyek yang lain. Karena ada kemungkinan himpunan entity dan himpunan relationship diantaranya terdapat nomor yang berbeda sehingga diperlukan berbagai atribut. Sehingga atribut dapat digunakan untuk membedakan sifat-sifat entity.

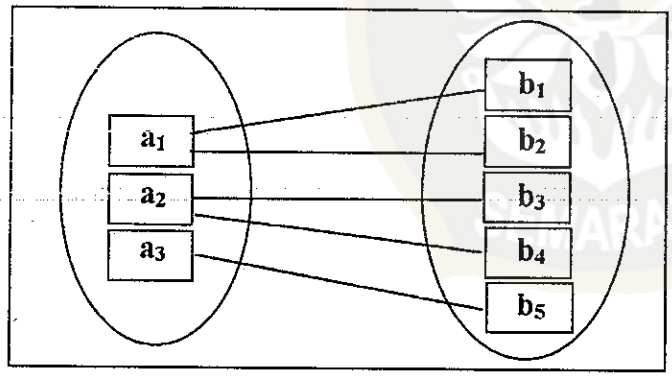
Untuk relationship biner himpunan *R* antara himpunan entity *A* dan *B*, kardinalitas pemetaannya harus salah satu dari yang tertera di bawah ini :

1. **One-to-one (1:1)**. Dikatakan 1:1 jika sebuah entity *A* dihubungkan dengan satu entity *B* dan sebuah entity *B* dihubungkan dengan satu entity *A*. (Gambar 2-1)

2. **One-to-many (1:m)**. Dikatakan 1:m jika sebuah entity *A* dihubungkan dengan beberapa nomor pada entity *B*, akan tetapi sebuah entity *B* dapat dihubungkan dengan satu entity *A*. (Gambar 2-2)

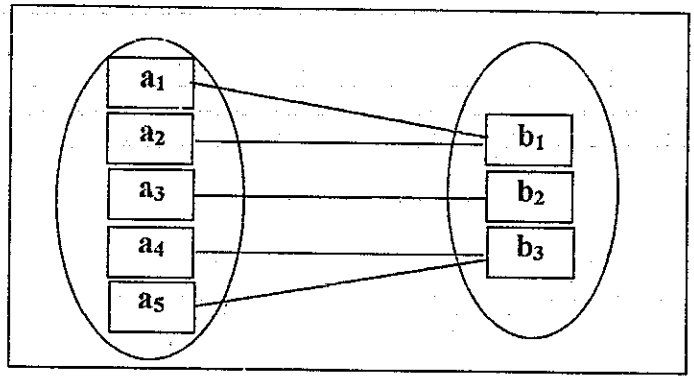


Gambar 2-1 Relationship one-to-one (1:1)



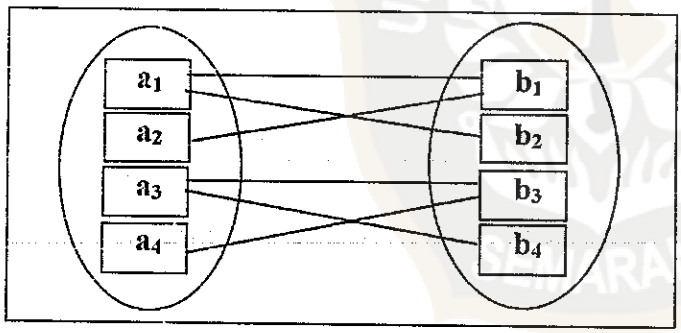
Gambar 2-2 Relationship one-to-many (1:m)

3. **Many-to-one (m:1)**. Dikatakan m:1 jika sebuah entity *A* dengan satu entity *B*, akan tetapi sebuah entity *B* dihubungkan dengan beberapa nomor pada entity *A*. (Gambar 2-3)



Gambar 2-3 Relationship many-to-one (m:1)

4. **Many-to-many (m:m).** Dikatakan m:m jika sebuah entity *A* dihubungkan dengan beberapa nomor pada entity *B* dan sebuah entity *B* dihubungkan dengan beberapa nomor pada entity *A*. (Gambar 2-4)



Gambar 2-4 Relationship many-to-many (m:m)

2.5. RANCANGAN BASIS DATA RELASIONAL

Basis data relasional adalah basis data yang berupa kumpulan relasi. Secara umum rancangan basis data relasional ditampilkan dalam bentuk skema relasi agar informasi dapat diaplikasikan dengan lebih mudah. Salah satu bentuk skema rancangan basis data relasional adalah *normal form*.

Dalam perancangan suatu basis data relasional diperlukan langkah-langkah untuk mempermudah perancangan (Barker, 1990). Langkah-langkah tersebut, menurut American National Standards Institute (ANSI) adalah :

1. Setiap entity dimasukkan kedalam tabel.
2. Setiap atribut dimuat dalam suatu kolom.
3. Identifier yang unik dari entity menjadi kunci unik pada tabel. Bila nama entity digunakan sebagai nama atribut untuk menjadi nama kolom yang unik maka digunakan sebagai kunci tamu.
4. Relasi m:1 dan 1:1 menjadi kunci tamu, jika dibawa kedalam duplikat identifier yang unik dari setiap entity dan berfungsi sebagai kunci kandidat.
5. Merancang indeks.

   Membuat indeks kandidat untuk :

   - Kunci unik (indeks unik)
   - Kunci tamu
   - Diturunkan sebagai beberapa fungsi : matriks atribut
6. Merancang sub-tipe

   Suatu sub-tipe entity secara sederhana merupakan entity yang memiliki atribut atau relationship tetapi juga mewarisi beberapa atribut dan/atau relationship dari induk entity (super-tipe).
7. Merancang basis data relasional dengan menggunakan relationship terpisah

Ada dua metode dasar yang menjadi pegangan untuk merancang basis data relasional dengan menggunakan relationship terpisah :

- Domain biasa

  Jika semua kunci tamu dalam domain yang sama (bentuk yang identik) maka dibuat dua kandidat kolom, yaitu:

  i. Relationship identifier

     Kolom relationship identifier akan digunakan secara berlainan antara relationship-relationship yang berbeda ditutup secara terpisah.

  ii. Entity identifier

     Kolom entity identifier akan digunakan untuk nilai identifier yang unik dari entity pada akhir dari relationship.

- Kunci tamu eksplisit

  Jika kunci tamu tidak dalam domain yang sama maka kolom kunci tamu eksplisit untuk setiap relationship ditutup secara terpisah.

Dengan langkah-langkah tersebut, akan membuat suatu basis data relasional menjadi mudah untuk dirancang.

Dalam merancang basis data relasional akan ditemui beberapa masalah, yaitu:

1. **Pengulangan informasi.**

   Misal pada rancangan basis data untuk suatu bank terdapat dua relasi, yaitu :

Cabang = (Nama_Cabang, Aset, Kota)

Pinjaman = (Nama_Cabang, Nomor_Pinjaman, Nama_Nasabah, Jumlah)

Kemudian pada alternatif perancangan terjadi pengulangan *Cabang* dan *Pinjaman* kedalam satu relasi :

Cab Pinj = (Nama_Cabang, Aset, Kota, Nomor_Pinjaman, Nama_Nasabah, Jumlah)

Pengulangan informasi seperti itu akan menggunakan banyak tempat.

2. **Ketidakmampuan menampilkan informasi yang tepat.**

   Pada relasi *Cab Pinj* tidak dapat ditampilkan informasi dengan tepat karena paling sedikit ada satu *Pinjaman* pada satu *Cabang*, sehingga pada relasi *Cab_Pinj* diperlukan nilai untuk Nomor_Pinjaman, Nama_Nasabah dan Jumlah

3. **Kehilangan informasi.**

   Misal *Pinjaman* dibuat menjadi dua relasi, yaitu :

   Jumlah = (Jumlah, Nama_Nasabah)

   No_Pinj = (Nama_Cabang, Nomor_Pinjaman, Jumlah)

   Jika terdapat beberapa pinjaman dengan jumlah yang sama maka tidak dapat dikatakan yang mana nasabah yang meminjam, dengan demikian telah terjadi kehilangan informasi.

Untuk mengatasi berbagai masalah tersebut maka diperlukan normalisasi. Normalisasi adalah pengelompokkan data item ke dalam struktur record tertentu

dan harus tidak terjadi kehilangan informasi. Normalisasi dibuat dengan menggunakan:

1. **Ketergantungan Fungsional (Fungsional Dependency)**

   Definisi :

   R suatu relasi, *x* dan *y* himpunan bagian yang berubah-ubah dari himpunan atribut *R*. *y* secara fungsional tergantung pada x, atau "$x \rightarrow y$" jika dan hanya jika setiap nilai x di dalam R berasosiasi tepat satu nilai *y* di dalam *R*.

2. **Ketergantungan Nilai Ganda (Multivalued Dependency)**

   Definisi :

   R suatu relasi. *A*, *B* atau *C* subset yang berubah-ubah dari himpunan atribut *R*. $A \twoheadrightarrow$(ketergantungan nilai ganda) jika dan hanya jika nilai *B* dapat dipasangkan dengan suatu pasangan (nilai *A*, nilai *C*), dalam arti tergantung pada nilai *A* dan bebas nilai *C*.

3. **Ketergantungan Berserikat (Join Dependency)**

   Definisi :

   R suatu relasi dan *A*, *B*, ..., *Z* suatu himpunan yang berubah-ubah dari *R*. *R* dikatakan memenuhi Ketergantungan Berserikat * (*A*, *B*, ..., *Z*) jika dan hanya jika *R* sama dengan join dari *R* pada *A*, *B*, ..., *Z*.