PERTIMBANGAN MELAKUKAN DENORMALISASI PADA MODEL BASIS DATA RELASI

Gandung Triyono

Fakultas Teknologi Informasi Universitas Budi Luhur
gandung.triyono@budiluhur.ac.id, gandungths@gmail.com

ABSTRAK

Pembentukan struktur basis data merupakan hal yang sangat penting dalam pengembangan sistem informasi. Tidak sedikit pengembangan sistem informasi gagal karena pemahaman kebutuhan data atas user tidak tercapai. Kebutuhan user inilah yang akan dituangkan dalam sebuah struktur basis data. Maka dari itu seorang perancang basis data tidak sekedar mampu membuat struktur yang bagus, struktur yang bagus tidak boleh hanya dilihat dari sisi teori belaka. Seorang perancang basis data juga harus mampu memahami kebutuhan tingkat operasional. Normalisasi merupakan parameter yang dapat digunakan oleh seorang perancang basis data untuk melakukan proses penyempurnaan dari tabel-tabel yang ada dalam basis data. Memang setelah diketahui tingkatan kenormalan dari setiap tabel, seorang perancang basis data dapat mengoptimalkan kinerjannya nanti sesuai dengan kebutuhan dengan melakukan denormalisasi. Denormalisasi merupakan proses untuk mengembalikan kembali sebuah tabel yang tadinya sudah normal ke sebuah tabel yang tidak normal. Tujuannya untuk mendapatkan kinerja yang bagus dari basis data. Denormalisasi ini sangatlah penting digunakan untuk acuan pembentukan struktur basis data yang sederhana dan bagus.

Kata kunci: Normalisasi, Denormalisasi, Basis Data

1. Pendahuluan

Pembentukan struktur basis data merupakan pekerjaan yang tidak mudah dalam pengembangan sistem informasi, dalam perancangan struktur basis data dituntut mampu membentuk sebuah struktur yang dapat memenuhi kebutuhan user pada saat ini maupun ke depan. Dalam pembentukan struktur yang sesuai dengan kebutuhan user seorang, perancang harus mampu melihat sampai dengan kebutuhan saat implementasin atau operasional.

Dalam kebutuhan operasional jelas beberapa yang harus dipertimbangkan, yaitu: kemanan data, kecepatan akses, *backup* dan *recovery,* dan juga dokumentasi yang baik. Beberapa kebutuhan operasional tersebut merupakan isu yang tidak boleh diabaikan. Salah satunya adalah kecepatan akses data, kecepatan akses data pada waktu sekarang ini merupakan tantangan bagi seorang perancang basis data bagaimana dapat membentuk struktur basis data yang kinerjanya cepat.

2. Normalisasi

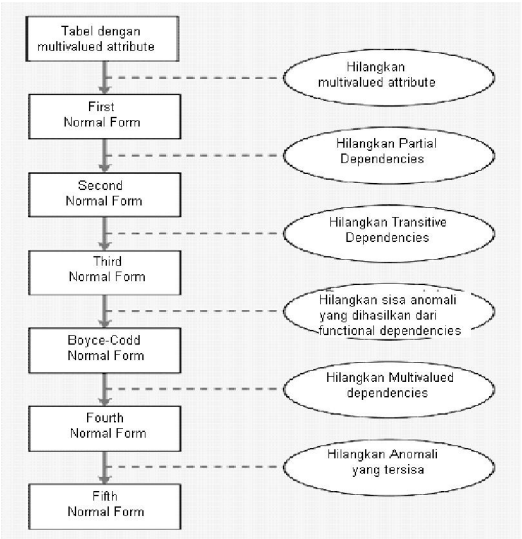
Normalisasi merupakan parameter digunakan untuk menghindarari duplikasi terhadap tabel dalam basi data dan juga merupakan proses mendekomposisikan sebuah tabel yang masih memiliki beberapa anomali atau ketidak wajaran sehingga menghasilkan tabel yang lebih sederhana dan struktur yang bagus, yaitu sebuah tabel yang tidak memiliki *data redundancy* dan memungkinkan user untuk melakukan *insert*, *delete*, dan *update* pada baris (*record*) tanpa menyebabkan inkonsistensi data. Tujuannya untuk menghindari beberapa *anomali*:

- *Insertion Anomaly* adalah proses melakukan penambahan *record* baru akan tetapi mempengaruhi *user* untuk terjadinya duplikasi data
- *Deletion Anomaly* adalah proses melakukan penghapusan *record* akan tetapi akan menyebabkan hilangnya data yang akan dibutuhkan pada *record* lain

- *Modification Anomaly* adalah proses merubah data pada sebuah *record* mempengaruhi perubahan pada *record* lain karena adanya duplikasi.

Proses penormalan tabel ada beberapa tahap yang harus dilakukan yaitu:

1. *First Normal Form* (1 NF)
   Sudah tidak ada *repeating group* yaitu pengulangan yang terjadi pada beberapa atribut atau kolom dalam sebuah tabel, dan juga setiap atribut harus bernilai tunggal. Atribut *multivalued, composite, derive* tidak tunggal. Setiap nilai dari atribut hanya mempunyai nilai tunggal.
2. *Second Normal Form* (2 NF)
   Untuk menjadikan tabel normal tingkat ke 2 maka sudah 1NF dan setiap atribut yang bukan *primary key* sepenuhnya secara *fungsional* tergantung pada semua atribut pembentuk primary key.
3. *Third Normal Form* (3 NF)
   Tabel sudah 2NF dan tidak memiliki *transitive dependencies*, *Transitive dependency* adalah ketika ada atribut yang secara tidak langsung tergantung pada *primary key* dan atribut tersebut juga tergantung pada atribut lain yang bukan *primary key*.
4. *Boyce-codd Normal Form* (BCNF)
   Tabel dalam BCNF jika sudah 3NF dan semua *determinants* adalah *candidate keys*. Perbedaan 3NF dan BCNF adalah untuk *functional dependency* A → B, 3NF memperbolehkan ketergantungan ada dalam relasi jika B adalah *Primary Key* dan A bukan merupakan *candidate key*. Sedangkan BCNF menuntut untuk ketergantungan tetap ada dalam relasi, A harus menjadi *candidate key*.
5. *Fourth Normal Form* (4 NF)
   Relasi berada pada bentuk normal keempat apabila memenuhi syarat BCNF dan tidak mempunyai *multivalue dependency*.
6. *Fifth Normal Form* (5 NF)
   Tabel bentuk normal kelima sering disebut PJNF (*Projection Join Normal Form*), penyebutan PJNF karena untuk suatu relasi akan berbentuk normal kelima jika tabel tersebut dapat dipecah atau diproyeksikan menjadi beberapa tabel dan dari proyeksi-proyeksi itu dapat disusun kembali (*join*) menjadi tabel yang sama dengan keadaan semula. Jika penyusunan ini tidak mungkin dilakukan dikatakan pada relasi itu terdapat ***join dependencies*** dan dikatakan bersifat ***lossy join***.



Gambar 1. Langkah-langkah dalam penormalan tabel

3. Denormalisasi

Denormalisasi adalah kebalikan dari proses normalisasi. Pada satu sisi denormalisasi menyebabkan adanya redudansi yang lebih besar, bahkan dapat mengurangi fleksibilitas basis data untuk perkembangan penggunan dimasa depan. Di sisi lain denormalisasi dapat mempercepat pemanggilan (retrieval) data, meskipun dapat memperlambat proses *update* data.

Secara prinsip, tidak ada metode khusus untuk melakukan denormalisasi

Beberapa situasi yang menjadi pertimbangan untuk melakukan denormalisasi sebagai upaya mempercepat transaksi sbb:

1. Mengkombinasikan relasi *One to One* (1:1)
2. Menduplikasi field bukan kunci dalam relasi *one to many* (1:M) untuk mereduksi proses join saat query.

3. Menduplikasi field FK dalam relasi *One to Many (*1:M) untuk mereduksi proses *join* saat *query*.
4. Membuat batasan pada multi atribut

Denormalisasi tabel adalah pelanggaran aturan normalisasi atau menjabarkan suatu tataan tabel dalam basis data yang telah normal untuk meningkatkan performa atau kinerja pengaksesan data pada basis data. Basis data yang telah normal disini dimaksudkan basis data yang redundansi datanya minim sehingga data yang disimpan tidak mengalami kerancuan dalam proses pengaksesan.

Perbedaan normalisasi dan denormalisasi adalah terletak pada redundansi data dan kompleksitas query. Pada redundansi data normalisasi lebih strik atau harus dihilangkan sebisa mungkin sehingga mengakibatkan apabila kita akan mengakses data dalam suatu database membutuhkan query yang kompleks. Berbeda dengan denormalisasi, denormalisasi disini tidak terlalu memikirkan tentang data yang redundan sehingga dalam mengakses data lebih cepat.

Pentingnya denormalisasi dalam database, Apabila kita menilik lebih lanjut tentang proses pengaksesan yang dilakukan basis data sewaktu data yang berada dalam suatu tabel ada 1000 baris dengan 100 juta baris. Hal itu akan terasa sangat beda proses kita menunggu untuk dapat melihat data. Itupun apabila kita mengaksesnya dari beberapa tabel yang setiap tabel berisikan jutaan data dan kita hanya menginginkan sebagian saja. Dari situ denormalisasi diperlukan, untuk menjaga kestabilan performa suatu sistem.

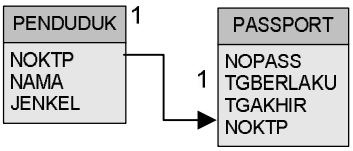
4. Pertimbangan Denormalisasi

Dalam implementasi dilapangan tabel-tabel yang terbentuk dalam basis data sangat dituntut dapat bekerja dengan optimal. Sedangkan struktur yang bagus dapat digambarkan bahwa tabel-tabel yang terbentuk lebih banyak dari pada jika basis data yang tidak normal. Dan jika tabel-table yang terbentuk banyak maka akan mengakibatkan beberapa efek yang kurang bagus. Melakukan implementasi basis data yang optimal kinerjanya dengan melakukan denormalisasi dengan berbagai pertimbangan. Di bawah ini merupakan gambaran yang dapat digunakan sebagai pedoman dalam melakukan denormalisasi.

4.1. Hubungan *One to one* (1 : 1)

Dalam relasi atau hubungan antar tabel terdapat tingkatan hubungan *one to one,* jika dalam implementasi terdapat 2 (dua) tabel yang memiliki tingkatan hubungan *one to one* boleh dijadikan satu (digabungkan) menjadi 1 (satu) tabel.

Contoh kasus tabel penduduk dengan tabel passport:



Gambar 2. Contoh relasi penduduk dan *passport*

Dari gambar di atas jika digabung menjadi satu relasi maka akan menjadi seperti pada tabel 1 di bawah ini:

Tabel penduduk → *primary key* **noktp**

Tabel 1. tabel data penduduk

noktp	nama	jenkel
P1	Yogie	L
P2	Vanie	P
P3	Zaky	L
P4	Zaya	P
P5	Naufal	L

Tabel passport → *primary key* **nopass**

Tabel 2. Data passport

nopass	tgberlaku	tgakhir	noktp
101	1-1-2010	1-1-2015	P1
102	1-2-2011	1-2-2016	P2

Bentuk tabel penduduk setelah dijadikan menjadi satu tabel.

Tabel 3. Tabel data penduduk

Noktp	nama	jenkel	nopass	tgberlaku	tgakhir
P1	Yogie	L	101	1-1-2010	1-1-2015
P2	Vanie	P	102	1-2-2011	1-2-2016
P3	Zaky	L			
P4	Zaya	P			
P5	Naufal	L			

a. Keuntungan

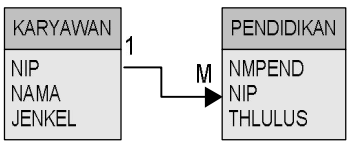
1) Perintah *Query* lebih sederhana, karena saat kita ambil data hanya menggunakan satu tabel.
2) Waktu yang dibutuhkan untuk akses lebih cepat.

b. Resiko
1) Tiga *field* akan bernilai *null* (kosong), jika Zaky, Zaya dan Naufal tidak memiliki passport.
2) Akan terjadi tambahnya ukuran dari tabel tersebut, yaitu sebanyak jumlah penduduk yang tidak memiliki passport. Sebagai pertimbangan untuk masalah ukuran sekarang ini sudah bukan masalah, karena mengingat harga dan ketersediaan media penyimpanan data sudah sangat besar dan murah.
3) Jika terjadi penghapusan atas data penduduk maka data passport juga akan ikut ke hapus, contoh jika nomor ktp P1 dihapus maka data passport dengan nomor 101 juga akan ikut terhapus.

4.2. Hubungan *One to Many* (1 : M) dengan Teknik Baris

Dalam relasi atau hubungan antar tabel terdapat tingkatan hubungan *one to many,* jika dalam implementasi terdapat 2 (dua) tabel yang memiliki tingkatan hubungan *one to Many* boleh dijadikan satu (digabungkan) menjadi 1 (satu) tabel. Dengan metode baris ini berarti *primary key* akan menjadi gabungan, lebih jelasnya lihat pembentukan tabel di bawah ini.

Contoh kasus tabel karyawan dengan tabel pendidikan:



Gambar 3. Relasi karyawan dan pendidikan

Contoh bentuk tabel dari karyawan dan pendidikan adalah:
Tabel Karyawan →*Primary Key* **nip**

Tabel 4. Tabel data karyawan

nip	nama	Jenkel
K1	Yogie	L
K2	Vanie	P

Tabel Pendidikan →*Primary Key* **nip** dan **nmpend**

Tabel 5. Tabel data pendidikan

nmpend	nip	thlulus
SD	K1	2004
SMP	K1	2007
SMA	K1	2010
SD	K2	1991
SMP	K2	1994
SMA	K2	1997
S1	K2	2001

Jika tabel karyawan disatukan maka akan terbentuk tabel karyawan yang terdiri dari nip, nama, jenkel, nmpend, dan thlulus, seperti terlihat pada tabel di bawah ini:

Tabel Karyawan →*Primary Key* **nip** dan **nmpend**

Tabel 6. Tabel karyawan setelah digabung

nip	**nama**	**jenkel**	**nmpend**	**thlulus**
K1	Yogie	L	SD	2004
K1	Yogie	L	SMP	2007
K1	Yogie	L	SMA	2010
K2	Vanie	P	SD	1991
K2	Vanie	P	SMP	1994
K2	Vanie	P	SMA	1997
K2	Vanie	P	S1	2001

a. Keuntungan

1) Perintah *Query* lebih sederhana, karena saat kita ambil data hanya menggunakan satu tabel.
2) Waktu yang dibutuhkan untuk akses lebih cepat.

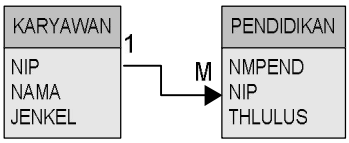
b. Resiko
1) Jika dilihat dari ketergantungan atribut maka tabel akan menjadi tidak normal ke 2 atau baru normal tingkat ke 1, karena jika diperhatikan **nama** dan **jenkel** akan tergantung sama **nip** padahal nip bukan **primary key** (hanya bagian dari primary key).

2) Resiko yang ditimbulkan adalah terdapat pengulangan beberapa atribut yaitu **nama** dan **jenkel.** Sehingga jika terjadi pencarian data karyawan seperti nama dan jenkel haris dilakukan teknik disnticnt (membuang kerangkapan).
3) Untuk primary *key* akan menjadi gabungan yaitu **nip** dan **nmpend**.
4) Akan terjadi tambahnya ukuran dari tabel tersebut, yaitu sebanyak jumlah nama dan jenkel yang terjadi kerangkapan.

4.3. Hubungan One to Many (1 : M) dengan Teknik Kolom

Dalam relasi atau hubungan antar tabel terdapat tingkatan hubungan *one to many,* jika dalam implementasi terdapat 2 (dua) tabel yang memiliki tingkatan hubungan *one to Many* boleh dijadikan satu (digabungkan) menjadi 1 (satu) tabel. Dengan metode kolom ini berarti harus melakukan pembatasan pada kolom yang bisa diisi yaitu akan membentuk kolom-kolom baru sejumlah baris yang ada di tabel pendidikan, lebih jelasnya lihat pembentukan tabel di bawah ini.

Contoh kasus tabel karyawan dengan tabel pendidikan:



Gambar 4. Relasi karyawan dan pendidikan

Contoh bentuk tabel dari karyawan dan pendidikan adalah:
Tabel Karyawan →*Primary Key* **nip**

Tabel 7. Tabel data karyawan

nip	nama	Jenkel
K1	Yogie	L
K2	Vanie	P

Tabel Pendidikan →*Primary Key* **nip** dan **nmpend**

Tabel 8. Tabel data pendidikan

nmpend	nip	thlulus
SD	K1	2004
SMP	K1	2007
SMA	K1	2010
SD	K2	1991
SMP	K2	1994
SMA	K2	1997
S1	K2	2001

Jika tabel karyawan disatukan maka akan terbentuk tabel karyawan yang terdiri dari nip, nama, jenkel, llsSD, llsSMP, llsSMA,lslsS1 dst, seperti terlihat pada tabel di bawah ini:
Tabel Karyawan →*Primary Key* **nip**

Tabel 9. Tabel karyawan setelah digabung

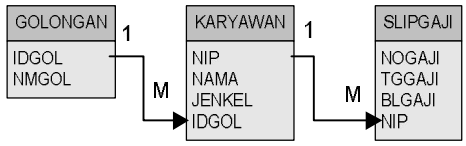
nip	**nama**	**jenkel**	**thllsSD**	**llsSMP**	**llsSMP**	**llsS1**
K1	Yogie	L	2004	2007	2010	
K2	Vanie	P	1991	1994	1997	2001

a. Keuntungan
1) Perintah *Query* lebih sederhana, karena saat kita ambil data hanya menggunakan satu tabel.
2) Waktu yang dibutuhkan untuk akses lebih cepat.
3) *Record* tidak terjadi duplikasi sepertihalnya jia melakukan dengan teknik baris.

b. Resiko
1) Karena terjadi pembatasan terhadap kolom pendidikan maka sulit untuk menambahkan kolom baru lagi.
2) Resiko sulit melakukan *query* untuk jika membutuhkan informasi yang menyangkut lulusan pendidikan.
3) Akan terdapat kolom akan bernilai *null (*kosong*)*, seperti terlihat pada tabel 4.7

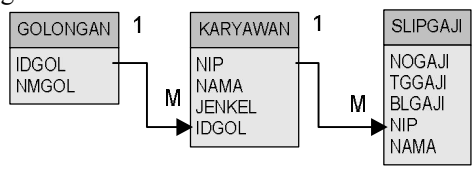
4.4. Duplikasi Atribut NonKey pada Hubungan One to Many (1 : M)

Dalam kasus ini diperbolehkan membuat duplikasi atau kerangkapan atribut, karena sangat dibutuhkan kecepatan.
Contoh kasus tabel karyawan dengan tabel slipgaji:



Gambar 5. Relasi karyawan dan slipgaji

Jika melakukan duplikasi terhadap atribut atau *field* nama, maka akan terlihat seperti gambar di bawah ini.



Gambar 6. Relasi karyawan dan slipgaji setelah diduplikasi

a. Keuntungan

1) Waktu yang dibutuhkan untuk akses lebih cepat.
2) Perintah *query*nya lebih sederhana karena tidak perlu baca tabel karyawan saat mencari data nama karyawan. Perhatikan perintah *query* sebelum diduplikasi dan setelah diduplikasi.

Query Sebelum duplikasi

Select noslip, blgaji, nama from karyawan, slipgaji where karyawan.nip = slipgaji.nip

Query sesudah duplikasi

Select noslip, blgaji, nama from slipgaji

b. Resiko

1) Karena untutukan integritas dan validitas data, maka jika dilakukan duplikasi kolom perlu melakukan singkronisasi. Jadi jika nama karyawan diedit pada tabel karyawan maka pada tabel slipgaji juga harus diedit.
2) Resiko akan muncul 2 (dua) model *query* jika membutuhkan informasi nama karyawan. *Query* pertama akan mengacu ke tabel karyawan dan yang ke dua akan mengacu pada tabel slipgaji untuk mendapatkan nama karyawan.
3) Ukuran dari basis data akan meningkat karena adanya duplikasi data sebelum diduplikasi dan setelah diduplikasi.

Query Sebelum duplikasi

Select noslip, blgaji, nama from karyawan, slipgaji where karyawan.nip=slipgaji.nip

Query sesudah duplikasi

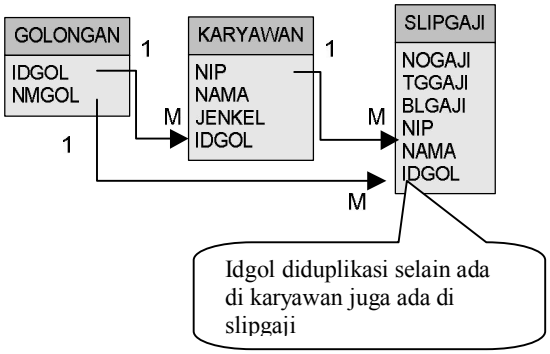
Select noslip, blgaji, nama from slipgaji

c. Resiko

1) Karena tuntutan integritas dan validitas data, maka jika dilakukan duplikasi kolom perlu melakukan singkronisasi. Jadi jika nama karyawan diedit pada tabel karyawan maka pada tabel slipgaji juga harus diedit.
2) Resiko akan muncul 2 (dua) model *query* jika membutuhkan informasi nama karyawan. *Query* pertama akan mengacu ke tabel karyawan dan yang ke dua akan mengacu pada tabel slipgaji untuk mendapatkan nama karyawan.
3) Ukuran dari basis data akan meningkat karena adanya duplikasi data.

4.5. Duplikasi Atribut FK (*Foreign Key*) pada Hubungan One to Many (1 : M)

Dalam kasus ini diperbolehkan membuat duplikasi atau kerangkapan atribut, karena sangat dibutuhkan kecepatan.
Contoh kasus tabel karyawan dengan tabel slipgaji:



Gambar 7. Relasi karyawan, golongan dan slipgaji

a. Keuntungan

1) Waktu yang dibutuhkan untuk akses lebih cepat. Karena saat dibutuhkan data nama golongan tinggal melakukan query seperti di bawah ini:
   Select noslip, nmgol from golongan, slipgaji where golongan.idgol = slipgaji.idgol
2) Perintah *query*nya lebih sederhana karena tidak perlu baca tabel karyawan saat mencari data nama karyawan. Perhatikan perintah *query* sebelum diduplikasi dan setelah diduplikasi.

Query sebelum duplikasi

Select noslip, blgaji, nmgol from karyawan, slipgaji, golongan where golongan.idgol = karyawan.idgol and karyawan.nip = slipgaji.nip

Query sesudah duplikasi

Select noslip, blgaji, nmgol from slipgaji, golongan where slipgaji.idgol = golongan.idgol

b. Resiko

1) Karena untutukan integritas dan validitas data, maka jika dilakukan duplikasi kolom perlu melakukan singkronisasi. Jadi jika nama karyawan diedit pada tabel karyawan maka pada tabel slipgaji juga harus diedit.
2) Resiko akan muncul 2 (dua) model *query* jika membutuhkan informasi nama golongan. Lihat contoh *query* yang dihasilkan dari *query-query* tersebut.

Query pertama

Select noslip, blgaji, nmgol from karyawan, slipgaji, golongan where golongan.idgol = karyawan.idgol and karyawan.nip = slipgaji.nip

Query kedua

Select noslip, blgaji, nmgol from slipgaji, golongan where slipgaji.idgol = golongan.idgol

5. Kesimpulan

a. Denormalisasi merupakan proses untuk melihat tabel-tabel yang ada pada basis data dapat bekerja lebih optimal.
b. Denormalisasi sebaiknya dilakukan setelah melakukan proses penormalan, Karena denormalisasi dilakukan jika dalam implementasi memang sangat dibutuhkan kecepatan akses yang tinggi. Akan tetapi jika tidak memerlukan kecepatan akses yang tinggi sebaiknya tidak melakukan denormalisasi.
c. Tabel dalam kondisi normal sangat mendukung dalam pengembangan aplikasi, jika terjadi perubahan atau pertumbuhan atas data.

Daftar Pustaka

[1] Date, C.J.,"*An Introduction to Database systems*", Sixth Edition, Addision Wesley Publishing Company Inc, 1995
[2] David M., Kronke, "*Data Processing, Fundamentals, Design & Implementation*", Pearson Prentice Hall, 2010
[3] McFadden, Fred R., Hoffer A., "*Modern Data Base Management*", Sixth Edition, Prentice Hall, 2002