Cara Praktis Merancang Database

Djoni Darmawikarta
djoni_darmawikarta@yahoo.ca

Latar Belakang

Ide tulisan ini berasal dari milis Ilmukomputer.com yang mendiskusikan bagaimana merancang database. Cara praktis dalam tulisan ini tidak menggunakan teknik normalisasi secara eksplisit atau formal, melainkan berdasar pengalaman penulis merancang berpuluh database berbagai ukuran dan kompleksitas, mulai dari yang modelnya memiliki kurang dari 25 tabel sampai dengan yang memiliki lebih dari 250 tabel. Database tersebut untuk berbagai macam sistem dan aplikasi, baik batch di mainframe, client/server, web-based, business oriented maupun real-time control system.

One Step at A Time

Kasus dalam tulisan ini adalah: Merancang database relasional untuk menyimpan data "stok barang" - ini adalah ruang-lingkupnya (scope)

Dari spesifikasi yang sudah diperoleh dari pemakai *(business user)*, kita ketahui bahwa yang dimaksud dengan stok barang adalah jumlah tersedia *(quantity on hand)* untuk setiap barang. Maka kita membutuhkan tabel stok_barang terdiri dari 2 kolom. Beberapa baris datanya sebagai berikut.

nama_barang	jumlah_tersedia
…	…
Barang A	100
Barang B	150
Barang C	175
Barang D	250
…	…

Perhatikan bahwa setiap barang hanya memiliki satu jumlah_tesedia. Dengan kata lain, nama barang unik didalam tabel stok_barang.

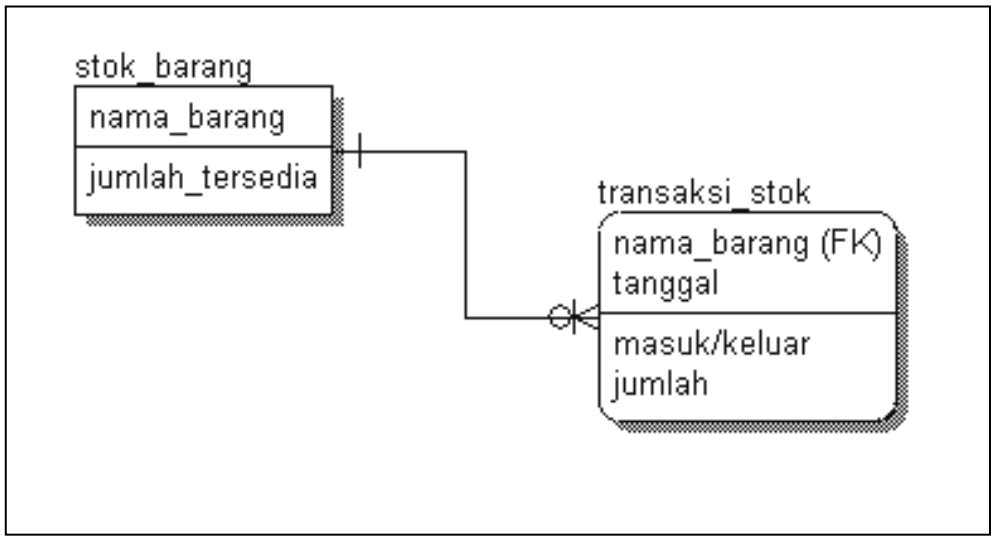
Spesifikasi juga meminta rancangan database kita untuk menangani transaksi pembelian dan penjualan barang, dan dampaknya pada stok barang. Tepatnya, bila ada barang masuk (pembelian) dan/atau keluar (penjualan), jumlah_tersedia perlu dimutakhirkan (update).

Ada dua pilihan cara melaksanakannya:
1. Jumlah barang masuk/keluar langsung ditambah/kurang-kan pada kolom jumlah_tersedia (update in place). Cara ini umumnya disebut sistem online. Data transaksi keluar/masuk barang disimpan, misalnya untuk keperluan audit atau rekonstruksi tabel stok_barang.
2. Data transaksi barang masuk dikumpulkan, setelah waktu tertentu, misalnya akhir hari, barulah diperhitungkan ke jumlah_tersedia. Cara ini disebut sistem batch.

Maka, untuk cara yang manapun dari kedua diatas, dibutuhkan satu tabel lagi untuk menyimpan data transaksi, misalnya sebagai berikut.

nama_barang	jumlah	masuk/keluar	tanggal
…	…	…	…
Barang A	1	m	2-Jan-04
Barang A	10	k	5-Jan-04
Barang B	5	k	10-Jan-04
…	…	…	…

Agar nama barang dikedua tabel sinkron, maksudnya semua nama barang ditabel transaksi_stok harus ada ditabel stok_barang, maka kedua tabel kita hubungkan: nama_barang ditabel stok_barang kita migrasikan ketabel transaksi_stok. Dengan kata lain, nama_barang ditabel transaksi_stok adalah foreign key dengan referensi nama_barang ditabel stok_barang. Diagram ER (Entity Relationship) model data sebagai berikut.

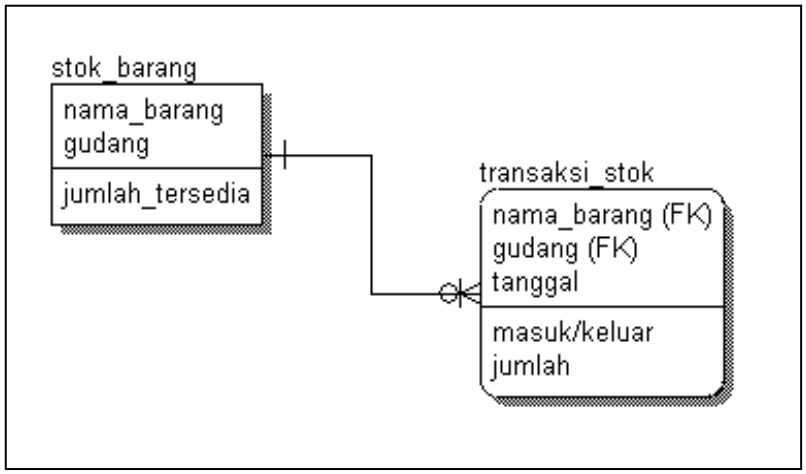


Perhatikan bahwa pada transaksi_stok, nama_barang dan tanggal adalah (composite) primary key. Ini berarti dalam satu hari (tanggal) untuk suatu barang hanya boleh ada satu transaksi.

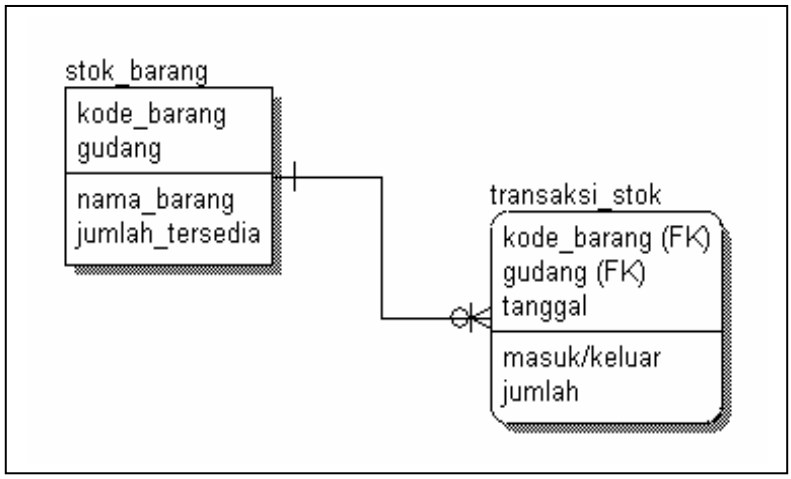
Lebih lanjut, spesifikasi menyatakan bahwa, ada beberapa gudang, dan suatu barang mungkin stoknya disimpan dibeberapa gudang (dan suatu gudang bisa menyimpan lebih dari satu barang) Maka perlu ditambahkan data gudang. Struktur tabel kita menjadi sebagai berikut.

nama_barang	gudang	jumlah_tersedia
…	…	…
Barang A	Gudang 1	100
Barang A	Gudang 2	10
Barang B	Gudang 1	150
Barang C	Gudang 2	175
Barang C	Gudang 3	20
Barang D	Gudang 3	250
…		…

Perhatikan bahwa setiap barang disuatu gudang hanya memiliki satu jumlah_tesedia. Dengan kata lain, nama_barang bersama gudang-nya unik didalam tabel stok_barang. Kini primary key tabel ini adalah nama_barang dan gudang. Dan, model data menjadi:



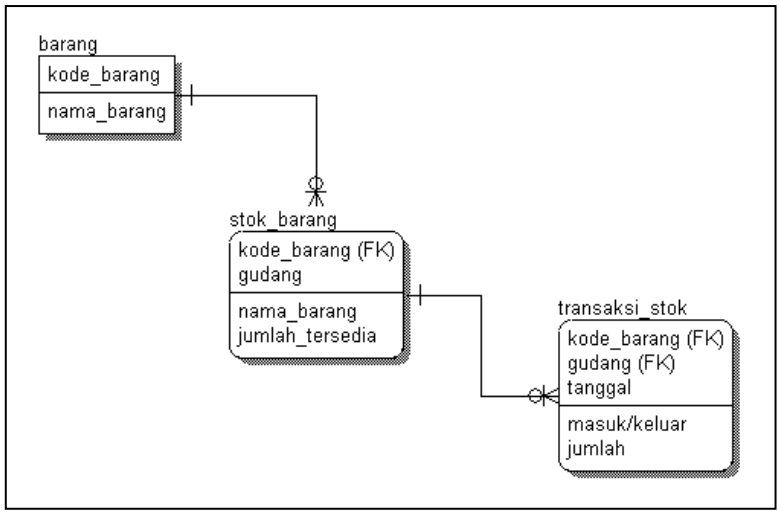
Karena oleh pemakai setiap barang sudah terbiasa diberi kode, maka dalam database kode barang juga diinginkan, selain namanya. Pemakai juga memastikan didalam spesifikasi, bahwa kode barang tidak pernah berubah, sedang nama barang kadang perlu diubah, maka kita gunakan kode_barang sebagai primary key. Berkembanglah model data kita menjadi:



Contoh isi tabel stok_barang sekarang adalah:

kode_barang	nama_barang	gudang	jumlah_tersedia
…	…	…	…
A01	Barang A	Gudang 1	100
A01	Barang A	Gudang 2	10
B01	Barang B	Gudang 1	150
C01	Barang C	Gudang 2	175
C01	Barang C	Gudang 3	20
D01	Barang D	Gudang 3	250
	…		…

Dapat dilihat, nama barang ikut diulang bersama kode barangnya. Akibatnya, bila ada perubahan nama, semua baris data barang bersangkutan harus seragam ikut dirubah dan barang ditabel ini sudah diwakili oleh kode_barang. Ini berbahayakan integritas data; maka sebaiknya dipisahkan, sebagai berikut.



Ringkasan

Laksanakan langkah-langkah demi langkah, jangan sekaligus menganalisa dan merancang semua data dalam spesifikasi.

1. Mulai dengan minimal, satu tabel, berdasar makna fungsi yang dibutuhkan. Dalam contoh kita, makna stok barang adalah quantity on hand untuk setiap barang.
2. Kembangkan struktur dari tabel ini, dengan makin menyertakan detil spesifikasi. Dalam contoh kita, quantity on hand disetiap gudang – maka perlu ditambahkan field "gudang". Demikian juga dengan penambahan kode_barang.
3. Bila ada duplikat data (data sama di lebih dari satu baris) dan sudah ada wakilnya pisahkanlah ditabel lain yang dihubungkan dengan tabel asalnya.
4. Fungsi berbeda biasanya memerlukan tabel terpisah; dalam contoh kita, fungsi "transaksi stok". Kita perlu tabel untuk menyimpan transaksi. Hubungkan dengan tabel utama (umumnya disebut master dalam system batch) agar data terkait dikedua tabel sinkron (integritas terjaga)

Makin besar dan rumit database yang harus kita rancang, teknik praktis ini makin lebih terbukti efektivitasnya, dibandingkan dengan menggunakan teknik normalisasi secara formal.