PERBANDINGAN ANTARA DUA PENDEKATAN NORMALISASI DATA HINGGA KONDISI FOURTH NORMAL FORM (4NF)

Oleh: Michael Iskandar[1]

Abstract:

Data normalization is a technique used to create databases that are able to store data in an effective and efficient manner. This, in turn, will ensure the integrity of the data contained within those databases. The process of normalizing a database from completely unnormalized to its ultimate form is a gradual one, and in practical terms this is commonly thought of as necessitating at least four steps, each of which is marked with a specific "normalized form", usually annotated as 1NF, 2NF, 3NF, and 4NF.

This paper compares two different approaches to achieve 4NF. The first method comprises the four steps usually described in books and articles, while the second approach needs only two-and-a-half of those steps to attain 4NF.

1. Pendahuluan

Salah satu aspek penting dalam pengembangan dan perawatan sebuah sistem informasi berbasis komputer adalah perancangan basis data (*database*) yang baik. Database yang dirancang dengan benar akan sangat memudahkan *systems analyst* dan *database administrator* dalam pengembangan dan perawatan sistem informasi perusahaan, serta juga sangat memudahkan penggunaannya oleh *end-user*. Sebaliknya sebuah database yang dirancang secara salah atau kurang baik akan menyulitkan penggunaannya serta, lebih parah lagi, akan menjadi sumber terjadinya kesalahan dalam pengisian database tersebut. Pada akhirnya hal ini akan mempengaruhi kualitas data yang tersimpan, karena integritas datanya sendiri menjadi berkurang.

Ada berbagai pendekatan untuk mengembangkan database, misalnya dengan membuat *Entity-Relationship Diagram* (ERD), *Collaboration Diagram*, dan *Normalisasi Data*. Tulisan ini hanya akan membahas tentang normalisasi data saja.

2. Pengenalan Normalisasi Data

---

[1] Penulis adalah dosen honorer di Fakultas Ekonomi UNPAR.

Tabel 1 adalah sebuah contoh database yang dipergunakan pada Jurusan Sistem Informasi pada sebuah lembaga pendidikan tinggi tertentu. Tabel tersebut berada dalam kondisi *belum dinormalisasi* atau disebut *unnormalized form*, diberi kode *UNF* atau *0NF*.

npm	nama	kode_mk		pengarang	nilai	bobot
200301	Anto	PKD01	ter Mikro	Ir. Andi	A	4
200301	Anto	PKD01	Mikro	Ir. Budi	A	4
200302	Anti	PKL01	ter Mikro	Ir. Andi	C	2
200302	Anti	PKL01	Mikro	Ir. Budi	C	2
200302	Anti	SPK01	Systems	James Duff	D	1
200303	Yanto	PKD01	ter Mikro	Ir. Andi	E	0
200303	Yanto	PKD01	Mikro	Ir. Budi	E	0
200303	Yanto	PKL01	ter Mikro	Ir. Andi	B	3
200303	Yanto	PKL01	Mikro	Ir. Budi	B	3
200304	Yanti	SPK02		Irene Hand	A	4
200304	Yanti	SPK02	erhana	Dra. Maria	A	4

Keterangan:

Npm = nc

Kode_mk

Nama_ml

Dalam kondisi seperti ini maka database tersebut memiliki banyak kelemahan. Satu hal yang langsung tampak adalah banyaknya data yang harus dicatat berulang-ulang. Misalnya saja, pada Tabel 1 tersebut mahasiswa dengan NPM 200303 hanya menempuh dua mata kuliah saja, namun terpaksa nama mahasiswa itu, "Yanto", harus dicantumkan sebanyak empat kali.

Selain itu, kondisi UNF memudahkan sekali terjadinya sejumlah kesalahan (*anomalies*). Pertama adalah *insertion anomaly*, yaitu kejadian di mana kita tidak dapat mengisikan data tertentu karena data lain belum diketahui. Pada contoh di atas kita akan memperoleh kesulitan untuk mencatat data-data sebuah mata kuliah yang belum ditempuh mahasiswa mana pun, karena untuk mencatat data mengenai mata kuliah tertentu harus ada identitas mahasiswanya juga. *Anomaly* kedua adalah *update anomaly*, yaitu perubahan pada satu data tertentu menyebabkan *user* terpaksa mengubah sejumlah baris pada database yang bersangkutan. Misalnya, seandainya nama mata kuliah "Pengantar Komputer Lanjut" diubah menjadi "Pengantar Komputer 2" maka hal ini menyebabkan empat baris pada Tabel 1 yang mengalami perubahan. Terakhir adalah *deletion anomaly*, yaitu penghapusan sebuah data akan menyebabkan data lain hilang juga. Misalnya, seandainya mahasiswi bernama "Yanti" (NPM 200304) mengundurkan diri dari perguruan tinggi ini sehingga datanya dihapus, maka otomatis seluruh data mengenai mata kuliah yang ditempuhnya juga ikut terhapus. Karena kebetulan tidak ada mahasiswa lain selain "Yanti" yang menempuh mata kuliah "Sistem Pakar" maka tidak akan ada informasi apa pun yang tersisa dari mata kuliah ini.

Oleh karena adanya masalah-masalah tersebut di atas maka database tidak boleh dibiarkan dalam kondisi UNF, melainkan harus diubah sedemikian rupa sehingga menjadi *normalized form*. Proses perubahan tersebut dikenal dengan istilah *normalisasi data*. Normalisasi data terjadi dalam beberapa tahap, di mana setiap tahapan menghasilkan *normalized form* dalam gradasi yang semakin tinggi. Tingkat/gradasi *normalized forms* tersebut dapat dilihat pada Gambar 1.

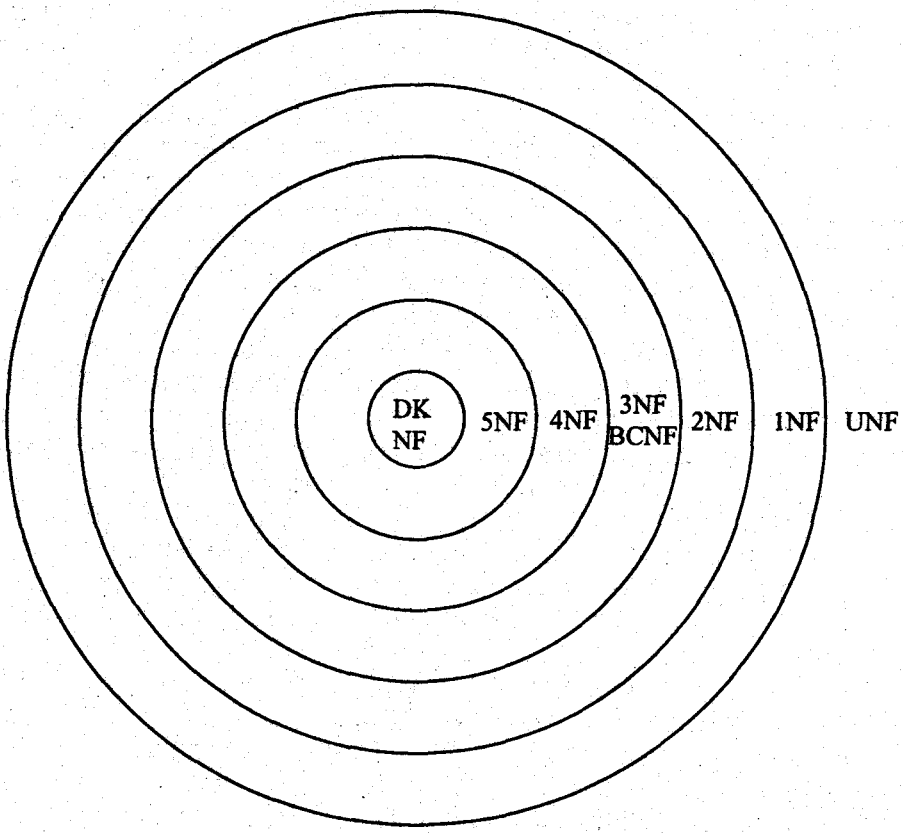
3. Langkah-langkah Normalisasi Data

Langkah pertama dalam melakukan normalisasi data adalah:

Tentukan kunci utama dan hilangkan kelompok berulang.

Yang dimaksud dengan *kunci utama* (*primary key*) adalah *field* (kolom) atau kombinasi *fields* yang mengidentifikasikan setiap *record* (baris) secara unik. Dalam contoh yang sedang dibahas nampak bahwa kuncinya adalah kombinasi dari *npm, kode_mk,* dan *isbn*. Sedangkan kelompok berulang yang perlu dihilangkan adalah nama mahasiswa serta nama mata kuliah. Caranya menghilangkan kelompok berulang adalah dengan memecah tabel tersebut menjadi dua. *Fields* yang membentuk kelompok berulang menjadi satu tabel, sedangkan *fields* sisanya membentuk tabel yang satu lagi. Jangan lupa untuk tetap menyertakan satu *field* penghubung antara kedua tabel tersebut.

Sebuah tabel yang telah memenuhi kondisi ini (jadi sudah tidak memiliki kelompok berulang) disebut berada dalam kondisi *First Normal Form* (1NF).



Gambar 1.
Tingkat-tingkat Bentuk Normal

Langkah kedua dalam melakukan normalisasi data adalah:

Hilangkan ketergantungan parsial.

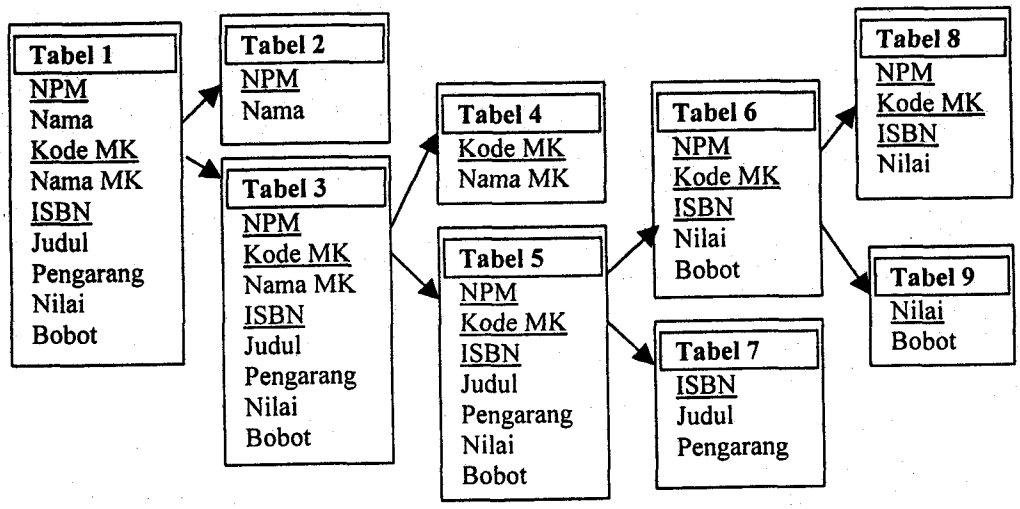
Yang dimaksud dengan ketergantungan parsial adalah *fields* tertentu yang tidak diidentifikasikan oleh semua *fields* kunci, melainkan hanya oleh salah satu/sebagian *field* kunci itu saja. Sama seperti pada langkah pertama, menghilangkan ketergantungan parsial juga dilakukan dengan memecah suatu tabel menjadi dua tabel yang baru. Tabel yang tidak lagi memiliki ketergantungan parsial disebut berada dalam kondisi *Second Normal Form* (2NF).

Langkah ketiga dalam melakukan normalisasi data adalah:

Hilangkan ketergantungan transitif.

Yang dimaksud dengan ketergantungan transitif adalah *fields* tertentu yang diidentifikasikan oleh *field* lain yang bukan merupakan *field* kunci. Tabel yang tidak lagi memiliki ketergantungan transitif dikatakan berada dalam kondisi *Third Normal Form* (3NF).

Sebagai contoh, berikut adalah proses normalisasi data atas Tabel 1, dari bentuk UNF hingga 3NF.



Gambar 2.
Proses Normalisasi Data Atas Tabel 1 Hingga 3NF

Seperti yang terlihat pada Gambar 2, Tabel 1 adalah tabel asli dari database yang masih dalam keadaan UNF. Langkah pertama normalisasi data mengharuskan kita menghilangkan kelompok berulang, dalam hal ini adalah data mahasiswa dan data mata kuliah. Hal ini terjadi pada pemecahan Tabel 1 menjadi Tabel 2 dan Tabel 3, serta pemecahan Tabel 3 menjadi Tabel 4 dan Tabel 5. Pada tabel-tabel 2, 4, dan 5 sudah tidak ada kelompok berulang sehingga tabel-tabel ini dikatakan telah 1NF.

Namun demikian, pada Tabel 5 *fields* Judul dan Pengarang tidaklah bergantung pada seluruh kunci, melainkan pada salah satu *field* kunci saja, yaitu *field* ISBN. Dengan perkataan lain, di sini terjadi ketergantungan parsial, yang harus dihilangkan jika hendak mencapai 2NF. Hal ini dilakukan dengan memecah Tabel 5 menjadi Tabel 6 dan 7. Kedua tabel yang terakhir ini, maupun Tabel 2 dan Tabel 4, tidak ada yang memiliki ketergantungan parsial lagi, sehingga telah mencapai kondisi 2NF.

Jika kita memeriksa Tabel 6 lebih jauh maka tampak bahwa di sini masih terdapat suatu ketergantungan transitif, yaitu suatu *field* bergantung pada *field* lain yang bukan merupakan kunci. Yang dimaksud di sini adalah *field* Nilai dan *field* Bobot. Baik Nilai maupun Bobot bukan merupakan kunci pada Tabel 6, namun *field* Bobot sebenarnya bergantung pada *field* Nilai.

Untuk mencapai 3NF, maka Tabel 6 dipecah lagi menjadi Tabel 8 dan Tabel 9. Pemeriksaan lebih lanjut atas Tabel 8 dan Tabel 9, maupun Tabel 2, 4, dan 7, menunjukkan tidak adanya ketergantungan transitif, sehingga semua tabel telah mencapai 3NF.

Bentuk "normal" 1NF, 2NF, dan 3NF telah dipresentasikan oleh Codd pada tahun 1972 dan sebenarnya sudah cukup baik untuk kebanyakan database. Namun kemudian Codd dan rekannya Boyce mengajukan sebuah bentuk normal yang baru, yang dikenal dengan istilah *Boyce-Codd Normal Form* (BCNF).

Peraturan BCNF adalah bahwa:

Sebuah tabel berada dalam kondisi BCNF jika setiap field yang menentukan fields lain adalah merupakan kunci.

Hasil dari aturan ini pada umumnya adalah sama dengan 3NF.

Sebagai contoh dari BCNF, dapat kita perhatikan kembali Tabel 1. Jika diperhatikan dengan seksama maka sebenarnya *field* nama_mk dapat juga dipergunakan sebagai kunci, dikarenakan nama_mk pasti adalah unik. Adalah tidak mungkin di dalam satu jurusan di perguruan tinggi tertentu terdapat dua mata kuliah berbeda dengan nama yang tepat sama. Jika hal ini dilakukan akan sangat menghambat kelancaran operasi lembaga ini, karena baik mahasiswa, dosen, maupun pihak tata usaha akan dibingungkan oleh kedua mata kuliah dengan nama yang sama tersebut.

Berangkat dari logika di atas maka sebenarnya field *nama_mk* dapat pula dipergunakan sebagai kunci Tabel 1. Jadi kunci yang mungkin adalah:

npm, kode_mk, isbn
npm, nama_mk, isbn

Yang manakah yang akan dipergunakan? Di sini dapat dilihat bahwa field *kode_mk* dibuat untuk mengidentifikasikan *nama_mk*, maka yang dipergunakan adalah kunci yang pertama (*npm, kode_mk, isbn*).

Jika aturan BCNF tersebut kita terapkan lebih lanjut untuk Tabel 1, maka akan diperoleh bahwa:

npm menentukan *nama*
kode_mk menentukan *nama_mk*
isbn menentukan *judul* dan *pengarang*
nilai menentukan *bobot*
npm, kode_mk, isbn menentukan *nilai*

Dengan demikian kita lihat bahwa dengan menggunakan definisi BCNF database ini bisa langsung mencapai kondisi 3NF.

Langkah normalisasi berikutnya, di atas 3NF/BCNF, adalah sebagai berikut:

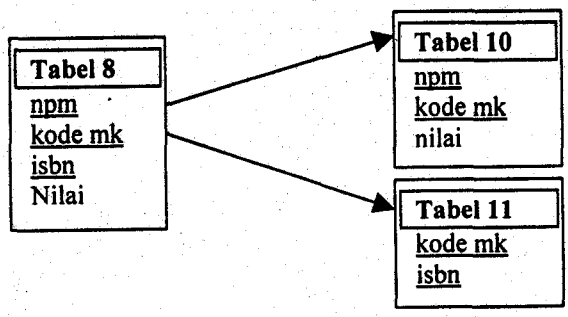
Hilangkan ketergantungan multi-nilai.

Tabel yang telah memenuhi peraturan ini dikenal dengan istilah *Fourth Normal Form* (4NF).

Peraturan ini sebenarnya bermaksud untuk mengatur hubungan *many-to-many* yang kadangkala muncul dalam sebuah database. Sebagai contoh, perhatikan kembali Tabel 1 di atas. Di sana terlihat bahwa antara mata kuliah dengan buku teks terdapat hubungan *many-to-many*, yaitu ada satu mata kuliah yang menggunakan beberapa buku teks (misalnya mata kuliah "Sistem Pakar" menggunakan dua judul buku), dan ada juga satu buku teks yang dipergunakan oleh beberapa mata kuliah (misalnya buku *Pengantar Komputer Mikro* karya Ir. Andi dipergunakan dalam mata kuliah "Pengantar Komputer Dasar" dan "Pengantar Komputer Lanjut").

Dalam proses normalisasi data yang telah digambarkan di atas, semua tabel yang telah mencapai 3NF ternyata juga telah mencapai 4NF, kecuali satu yakni Tabel 8. Apabila Tabel 8 tidak dinormalisasi lebih lanjut, maka akan terjadi perulangan yang mubazir, yakni setiap kali *kode_mk* tertentu muncul, isbn-isbn buku yang terkait akan menghasilkan sejumlah *records*. Hal ini menunjukkan bahwa normalisasi data belum tuntas bagi Tabel 8.

Apabila kita menerapkan peraturan 4NF atas tabel ini maka diperoleh:



GAMBAR 3.
Normalisasi Data Dari 3NF Ke 4NF

Seperti yang telah ditampilkan pada Gambar 1, di atas 4NF masih terdapat *Fifth Normal Form* (5NF) dan *Domain Key Normal Form* (DKNF). 5NF adalah sama dengan 4NF, dalam arti mengatur *many-to-many relationships*, namun dalam kondisi khusus sehingga sebuah tabel tidak dapat dipecah dua seperti contoh di atas, melainkan harus dipecah tiga. Sedangkan DKNF merupakan bentuk normal yang paling tinggi dan ideal. Pada prakteknya, baik 5NF maupun DKNF jarang sekali dibutuhkan, sehingga kedua bentuk ini lebih bersifat akademis daripada praktis, dan dalam tulisan ini baik 5NF maupun DKNF tidak dibahas lebih lanjut.

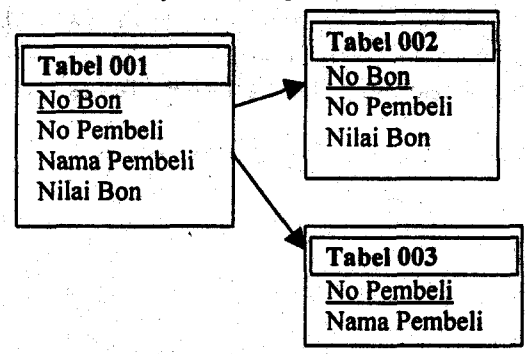
4. Metode Alternatif Pelaksanaan Normalisasi Data

Sebuah cara alternatif yang lebih pendek dijelaskan oleh Henk Jan Nootenboom dalam artikelnya berjudul *Normalisatiestappen*. Di sini hanya dibutuhkan 2½ langkah saja untuk mencapai 4NF, sebagai berikut:

a. Tentukan kunci utama
b. Hilangkan ketergantungan parsial
c. Hilangkan ketergantungan transitif

Perlu diperhatikan di sini bahwa pada langkah pertama Nootenboom hanya menentukan kunci utama dan tidak menghilangkan kelompok berulang, sehingga penulis menyebutnya sebagai hanya 2½ langkah saja jika dibandingkan dengan metode yang telah dijelaskan sebelumnya.

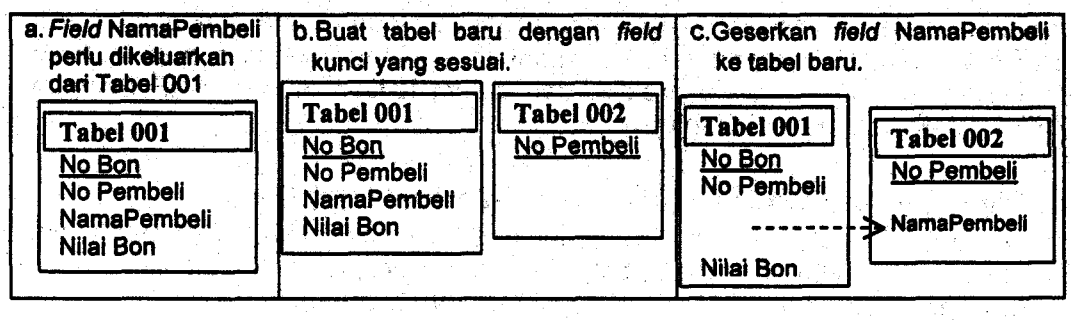
Perbedaan inti dari metode ini dengan metode sebelumnya adalah bahwa pemecahan sebuah tabel menjadi tabel baru dilakukan dengan *melepaskan* sebuah field dari tabel yang lama kemudian field itu *digeser* ke tabel yang baru. Jika ada field yang masih tersisa pada tabel yang lama, maka tabel tersebut tetap diperhitungkan. Sebagai contoh, misalkan ada tabel yang terdiri dari *fields* No Bon, No Pembeli, Nama Pembeli, dan Nilai Bon. Menurut metode pertama, normalisasi terjadi sebagai berikut:



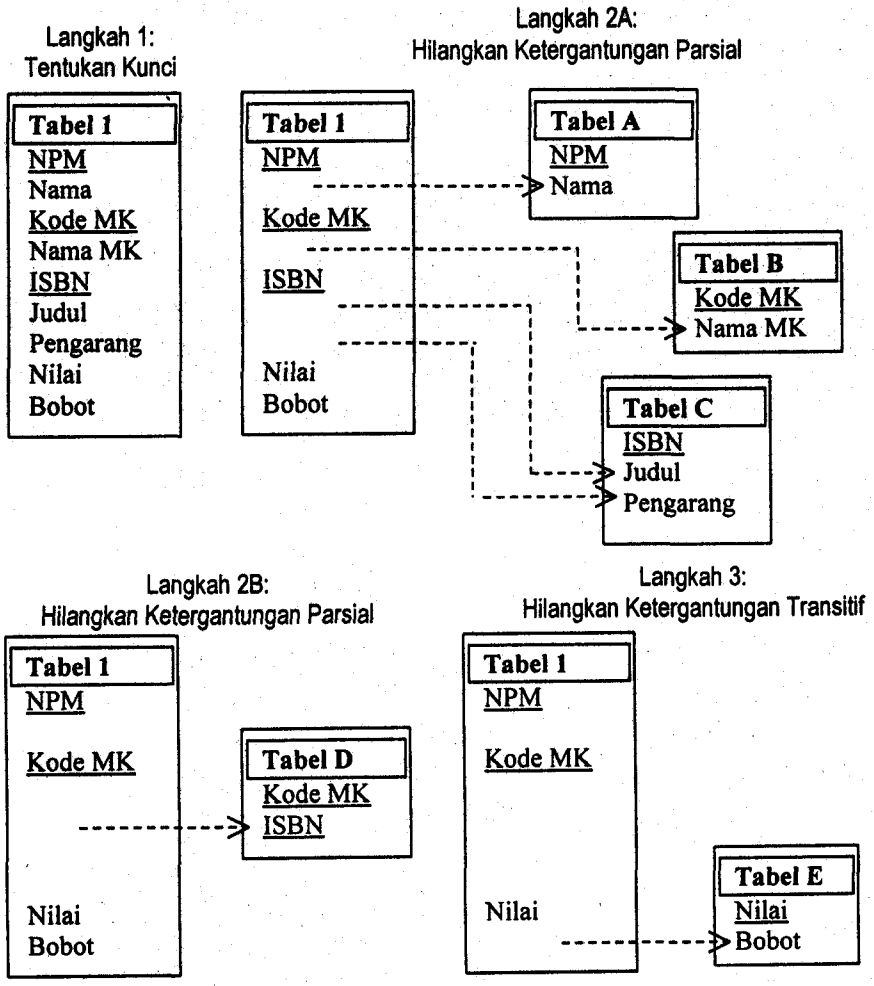
Sedangkan pada metode yang dijelaskan oleh Nootenboom, sambil memperhatikan 2½ langkah di atas, prosesnya adalah sebagai berikut:

a. Identifikasi *fields* yang perlu dikeluarkan dari tabel tertentu,
b. Buatlah sebuah tabel kosong yang baru yang hanya memiliki *field* kunci saja.
c. Geserkan/pindahkan *fields* yang hendak dikeluarkan dari tabel aslinya ke tabel kosong tersebut.

Sebagai contoh kecil, berikut ini sebuah database sederhana dinormalisasi menurut metode yang baru saja dijelaskan.



Proses penerapan metode ini atas database Jurusan Sistem Informasi (Tabel 1) adalah sebagai berikut:



Gambar 4.
Normalisasi Data Tabel 1 Dengan Menggeser Fields

Tabel 2
Perbandingan Hasil Dua Metode Normalisasi

Metode 1	Metode 2
Tabel 2 (npm, nama)	*Tabel A (npm, nama)*
Tabel 4 (kode_mk, nama_mk)	*Tabel B (kode_mk, nama_mk)*
Tabel 7 (isbn, judul, pengarang)	*Tabel C (isbn, judul, pengarang)*
Tabel 9 (nilai, bobot)	*Tabel E (nilai, bobot)*
Tabel 10 (npm, kode_mk, nilai)	*Tabel 1 (npm, kode_mk, nilai)*
Tabel 11 (kode_mk, isbn)	*Tabel D (kode_mk, isbn)*

Perhatikan bahwa hasil akhir yang diperoleh dengan menggunakan metode ini adalah tepat sama dengan yang diperoleh dengan metode pertama.

5. Perbandingan Antara Kedua Metode

Ada beberapa hal yang cukup menarik perhatian dalam pembandingan yang penulis lakukan antara kedua metode ini. Hal-hal tersebut adalah:

a. Pada metode yang dijelaskan Nootenboom, analis tidak perlu secara sadar (*consciously*) menghilangkan kelompok berulang seperti yang diharuskan oleh metode yang pertama. Penulis sempat menanyakan tentang hal ini melalui e-mail kepada Mr. Nootenboom, dan beliau menjawab sebagai berikut: *"Kies in 1NF een HELE composite key, voor een tabel, zodat geen enkele row meer een repeating group heeft, alle dataelementen een waarde hebben, afhankelijk van de sleutel. Die sleutel is dan te groot, maar 2NF lost dat op."* ("Pilihlah dalam 1NF satu *composite key* secara UTUH, untuk satu tabel, sehingga tidak ada baris yang memiliki *repeating group*, setiap elemen data hanya memiliki satu nilai, bergantung pada kunci. Kunci ini kemungkinan terlampau luas, namun 2NF akan mengatasi masalah itu.")

   Dari sini penulis menangkap bahwa beliau mengartikan tidak boleh adanya kelompok berulang adalah dalam kaitan dengan kunci utama yang telah diidentifikasi. Oleh karena sebuah kunci utama yang telah diidentifikasikan dengan benar akan otomatis menyebabkan tidak ada dua atau lebih *records* yang tepat sama, maka langkah menghilangkan kelompok berulang dapat diabaikan.

b. Pada metode kedua analis harus selalu memeriksa kembali tabel "sisa" setelah beberapa *fields* digeser ke tabel yang baru. Hal inilah yang menyebabkan "Tabel 1" masih *exist* pada saat normalisasi data selesai, sedangkan pada metode yang lebih umum, "Tabel 1" itu sudah lama tidak ada lagi, digantikan oleh tabel-tabel baru. Mr. Nootenboom menyebutkan dalam e-mailnya bahwa aspek ini, yaitu aspek memeriksa kembali "tabel sisa" (*resttabel*) untuk mengevaluasi kembali keberadaannya, adalah ciri khas dari pendekatan yang beliau pergunakan. Hal ini merupakan hal krusial pada contoh database di atas. Jika analis tidak jeli maka *Tabel D (kode_mk, isbn)* akan terlewat dan sama sekali tidak dibuat.

   Perbedaan yang terjadi di sini adalah akibat perbedaan interpretasi tentang 2NF dan 4NF. Karena pada metode pertama tabel yang sudah pernah dipecah dua tidak pernah dilirik lagi, maka dibutuhkan langkah keempat untuk menghasilkan 4NF. Sedangkan pada metode yang dijelaskan Mr. Nootenboom, 4NF dianggap tidak relevan lagi karena akan otomatis terjadi jika analis mengevaluasi "tabel sisa" tadi menggunakan langkah untuk 2NF ("hilangkan ketergantungan parsial").

Masih berkaitan dengan kedua hal di atas, Mr. Nootenboom juga menulis dalam e-mailnya, *"Splitsen van UNF naar 1NF lijkt me vreemd...Er zijn dan nog geen gronden om te splitsen. Kies eerst een compound key in 1NF,*

begin dan die key te splitsen in tabellen in 2NF." ("Memecah [tabel] dari UNF menjadi 1NF menurut saya adalah aneh ... [sebab] pada saat tersebut belum ada alasan untuk melakukan pemecahan. Pilih dulu sebuah *compound key* dalam 1NF, kemudian mulai memecah-mecah *key* tersebut dalam tabel-tabel dalam 2NF.") Dari sini tampak bahwa Mr. Nootenboom menekankan pemisahan/pemecahan *kunci* ke dalam tabel baru, sedangkan *non-key fields* hanya bergeser ke tabel baru karena secara logis terkait dengan tabel baru tersebut.

6. Kesimpulan

Dalam tulisan ini telah dijelaskan dua metode normalisasi data yang dapat dipergunakan analis untuk mengembangkan sebuah database yang baik. Menurut pendapat penulis masing-masing metode dapat dipertanggung-jawabkan dan memiliki keunggulannya sendiri-sendiri. Metode pertama, yang lebih umum dijelaskan dalam buku-buku teks, memiliki tahapan-tahapan yang lebih banyak sehingga dapat dianggap lebih "merepotkan". Namun keuntungan dari metode pertama ini adalah bahwa langkah-langkahnya itu lebih jelas sehingga lebih sedikit risiko analis membuat kesalahan. Di lain pihak, metode yang dijelaskan Mr. Nootenboom meskipun memiliki langkah-langkah yang lebih pendek, namun membutuhkan kejelian dan analisa yang lebih tajam.

Pada akhirnya yang juga sangat berpengaruh adalah pengalaman dan kenyamanan yang dirasakan analis itu sendiri dalam melakukan normalisasi data dengan kedua metode tersebut. Penulis hanya bisa menyarankan bahwa apabila seseorang sudah merasa "nyaman" dengan salah satu metode, maka agar metode tersebut yang dipergunakan dengan sebaik-baiknya.

Daftar Pustaka

1. Mannino, Michael V., *Database Application Development and Design*, 1st ed., Singapore, McGraw-Hill, 2001.
2. Kendall, Kenneth E. and Kendall, Julie E., *Systems Analysis and Design*, 2nd ed., New Jersey, Prentice-Hall, 1992.
3. Wyllys, R.E., *Database Management Principles and Applications*, The University of Texas at Austin, Graduate School of Library and Information Science, http://www.gslis.utexas.edu/~l384k11w/normover.html, 2002.
4. Nootenboom, Henk Jan, *Normalisatiestappen*, http: //www.sum-it.nl/cursus/dbdesign/hollands/logis010.php3 , 2003.
5. Nootenboom, Henk Jan, *4NF, Onzin Voor Wie Goed Normaliseert Tot Derde Normaal Vorm*, http://www.sum-it.nl/no200238.html, 16 Sept 2002.
6. Gupta, Gopal K., *Boyce-Codd Normal Form*, *http://www.cs.jcu.edu.au/Subjects/cp3020/1995/6/node12.html*, 5 Oct 1995.
7. Post, Gerald V., and Anderson, David L., *Management Information Systems: Solving Business Problems with Information Technology*, 2nd ed., Boston: Irwin/McGraw-Hill, 2000.