CHAPTER 5

LEBIH LANJUT TENTANG DEPENDENSI DAN KUNCI RELASI

5.1 PENDAHULUAN

Pada chapter sebelumnya telah dideskripsi semua yang perlu diketahui tentang relasi untuk metodologi desain database yang paling praktis. Desain databases praktis memulai dengan model level paling tinggi yang dikonversikan untuk relasi yang dekat dengan bentuk normal tertinggi. Periksa seperti yang telah dideskripsikan pada chapter sebelumnya digunakan untuk menguji relasi-relasi ini dan jika perlu dekomposisikan mereka kedalam bentuk BCNF. Dalam banyak kasus, sangat sedikit dekomposisi yang dibutuhkan karena kebanya relasi yang diproduksi oleh analisis level tinggi adalah BCNF.

Dependensi fungsional dan kunci relasi mempunyai banyak properti (sifat) tambahan yang tidak perlu digunakan dalam metodologi praktis kebanyakan. Properti ini berkenaan dengan turunan dari kunci relasi dari dependensi fungsional dan turunan dari dependensi fungsional dari setiap yang lain. Properti ini dibutuhkan dalam metodologi desain yang hanya menggunakan analisis relasional. Properti jua memberikan latar belakang yang berguna dalam rancangan praktis melalui pemahaman yang lebih baik dari proses relasional dan pemahaman yang lebih baik untuk proses normalisasi.

Chapter ini meliputi beberapa properti tambahan tentang dependensi fungsional, hubungannya dengan kunci relasi dan bagaimana kunci relasi dapat diturunkan dari dependensi fungsional. Chapter berikutnya menggunakan properti ini untuk mendeskripsikan metoda desain yang hanya menggunakan analisis relasional. Akan tetapi, jika anda tertarik pada metodologi desain praktis berdasarkan model semantic level tinggi, anda boleh melewati chapter ini dan chapter berikutnya tanpa kehilangan kontinuitas.

5.1 DEPENDENSI FUNGSIONAL REDUNDANT

Dalam desain database relasional, relasi dibangun dari dependensi fungsional. Sembarang dependensi fungsioanl redundant menyebabkan redundansi dalam relasi. Untuk menghindari redundansi ini, dependensi fungsional redundasi harus dibuang sebelum relasi dibangun.

Dependensi fungsional redundansi adalah dependensi yang dapat diturunkan dari dependesi lain. Sebagai contoh, jika diketahui bahwa:

REG-NO $\rightarrow$MODEL, dan

MODEL $\rightarrow$ NO-CYLINDERS

Maka kita dapat menurunkan:

REG-NO $\rightarrow$NO-CYLINDERS

Ini menyatakan bahwa jika kita mengetahui model dari mobil untuk sebuah nomor registrasi (REG-NO) dan jumlah silinder untuk model tersebut, maka kita dapat menurunkan jumlah silinder untuk REG-NO yang diberikan. Suatu himpunan non-redundant dari dependensi akan memasukan

sebarang dependensi yang dapat diturunkan dari dependensi lain dalam himpunan tersebut. Himpunan non-redundant demikian tidak akan memasukan dependensi REG-NO → NO-CYLINDERS. Redundansi dapat dibuang masing-masing oleh:

- Pengujian diagram dependensi untuk dpendesni redundant menggunakan sebuah himpunan aturan penilaian, atau dengan
- Menggunakan algoritma keanggotaan.

5.3 MENGHAPUS DEPENDENSI DENGAN PENGUJIAN

Menghapus dependensi redundant melalui pengujian menggunakan aturan yang menspesifikasikan bagaimana satu dependensi dapat diturunkan dari dependensi yang lain. Ditunjukan bahwa terdapat himpunan minimum seperti aturan untuk tujuan ini. Aturan-aturan ini dikenal dengan aturan penilaian. Jika kita lihat bahwa satu dependensi fungsional, f1, dalam sebuah himpunan dapat diturunkan dari dependesi fungsional lain dalam himpunan menggunakan penilaian lain, maka f1 adalah redundant.

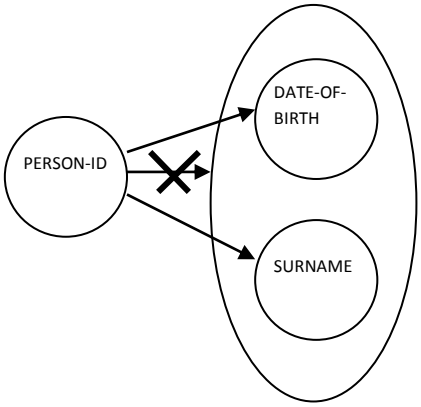
Aturan Penilaian

Terdapat sejumlah aturan penilaian. Aturuan penilaian didefinisikan dalam istilah himpunan atribut. Definisi merujuk pada himpunan menggunakan huruf capital seperti X, Y, Z dan seterusnya. Kita mulai dengan pendefinisian aturan-aturan dengan lebih langsung dan kemudian mengikuti definisi yang lebih komplek. Aturan pertama adalah aturan penggabungan

Penggabungan (Union)

If X→Z dan X→Y maka X→ZY

Gambar 5.1 Aturan Penggabungan



Aturan ini cukup jelas dan diilustrasikan secara diagram dalam Gambar 5.1. Disini kedua dependensi fungsional:

PERSON-ID →DATE-OF-BIRTH, dan

PERSON-ID →SURNAME

Digantikan oleh satu dependensi fungsional

PERSON-ID →SURNAME, DATE-OF-BIRTH

Dependensi yang terakhir adalah redundant seperti yang ditunjukan tanda silang dalam Gambar 5.1.

Decomposisi

If X→ZY maka X→Z dan X→Y

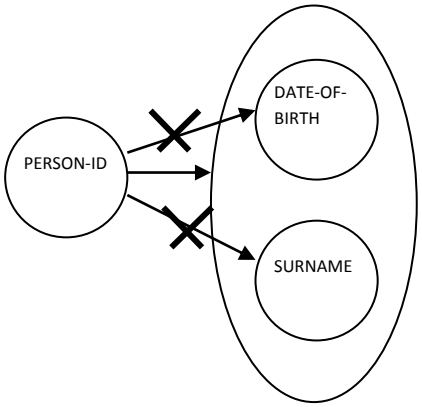
Aturan dekomposisi berlawanan dengan aturan penggabungan. Aturan ini menyatakan bahwa sebuah dependensi fungsional dengan sejumlah atribut pada sisi kanan dapat digantikan oleh beberapa dependensi fungsional.

Sebuah contoh diberikan dalam Gambar 5.2. Contoh ini jelas merupakan kebalikan dari Gambar 5.1. Disini dependensi fungsional

PERSON-ID →SURNAME, DATE-OF-BIRTH

Didekomposisikan menjadi dua dependensi

Gambar 5.2 Aturan dekomposisi



PERSON-ID →SURNAME, dan

PERSON-ID → DATE-OF-BIRTH

Dua dependensi fungsional yang terakhir yang disilangi adalah redundant dalam Gambar 5.2

Juga terdapat sebuah kata peringatan bagi pemula untuk hati-hati dan tidak memperluas dekomposisi pada determinant. Hal yang menggoda untuk memperluas aturan dekomposisi dan menyatakan bahwa jia:

AX→Y maka

A→Y dan X→Y.

Hal ini tidaklah benar. Sebagai contoh jika

PERSON-ID, PROJECT-ID →TIME-SPENT

Maka tak satupun dari pernyataan dibawah ini yang benar

PERSON-ID →TIME-SPENT atau

PROJECT-ID →TIME-SPENT

Tak satupun dari dependensi fungsional ini yang benar karena boleh jadi terdapt banyak nilai yang berbeda dari TIME-SPENT untuk sebuah project. Boleh jadi terdapat perbedaan nilai TIME-SPENT untuk setiap orang yang berbeda untuk sebuah proyek yang diberikan. Sama halnya orang yang sama mungkin menghabiskan waktu yang berbesa untuk setiap proyek. Perancang harus ingat bahwa dekomposisi diaplikasi pada bagian sebelah kanan dan tidak pada bagian sebelah kiri dan dependensi fungsional.

Reflexitas

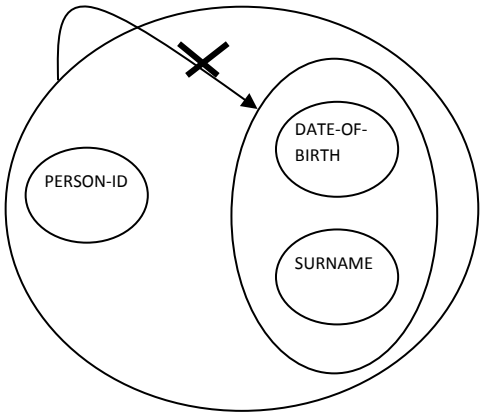
Hukum reflexitas menyatakan bahwa:

Jika Y adalah subset dari X maka X→ Y

Diagram dependensi fungsional yang mewakili aturan ini terlihat agak asing. Seperti diilustrasikan dalam Gambar 5.3 terdapat kembali ganda kedirinya sendiri, sehingga disebut refleksitas. Contoh dalam Gambar 5.3 menunjukan bahwa beberapa atribut dari PERSON dapat ditentukan oleh atribut ini dikombinasikan dengan atribut PERSON lain. Jelaslah ini redundant.

Aturan refleksitas jika menyatakan X→X.

Gambar 5.3 Refleksitas



Augmentasi

Aturan augmentasi menyatakan bahwa:

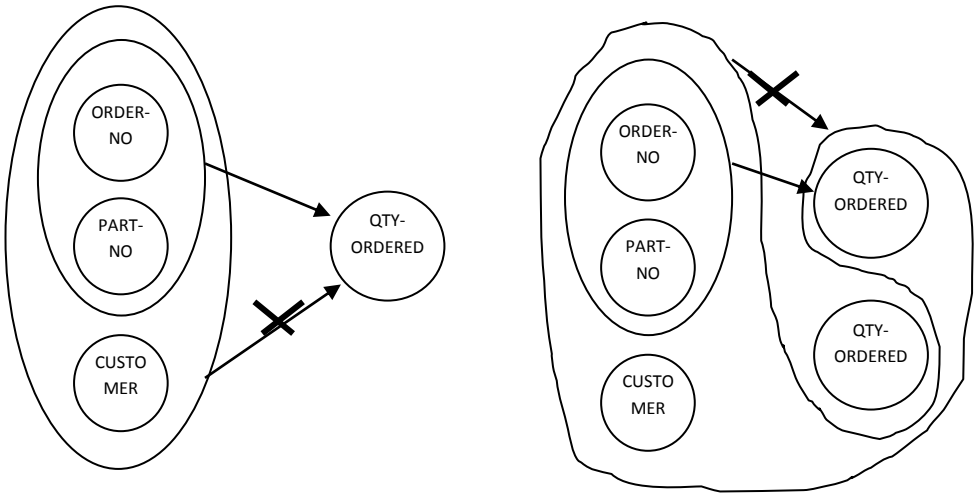
Jika Z adalah sebuah subset W dan X →Y THEN XW→YZ

Aturan augmentasi sangat mirip dengan aturan refleksivitas, kecuali bahwa subset dikombinasikan dengan dependensi fungsional lain. Sebuah contoh sederhana dari aturan augmentasi diberikan dalam Gambar 5.4. disini kita diberikan bahwa:

ORDER-NO, PART-NO $\rightarrow$ QTY-ORDERED

Contoh yang lebih kompleks juga ditunjukan pada Gambar 5.4:

W={CUSTOMER, CUSTOMER-ADDRESS} dan Z={CUSTOMER}



Dengan augmentasi kita mendapatkan:

ORDER-NO, PART-NO, CUSTOMER, CUSTOMER-ADDRESS$\rightarrow$QTY-ORDERED, CUSTOMER-ADDRESS

Aturan berikutnya adalah transivitas

Transivitas

Aturan transivitas menyatakan bahwa:

If X$\rightarrow$Y AND Y$\rightarrow$Z THEN X$\rightarrow$Z

Ide dibalik transivitas diilustrasikan dalam Gambar 5.5 dalam hal contoh yang kongrit. Disini:

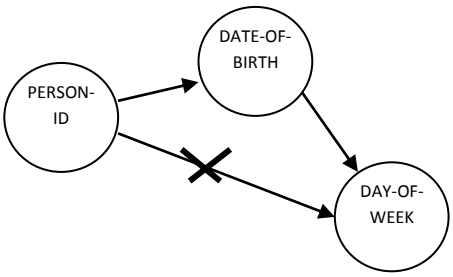
PERSON-ID $\rightarrow$DATE-OF-BIRTH

DATE-OF-BIRTH $\rightarrow$DAY-OF-WEEK

Hal ini dapat digunakan untuk menurunkan:

PERSON-ID$\rightarrow$DAY-OF-WEEK

Gambar 5.5 Transivitas



Dependensi fungsional PERSON-ID →DAY-OF-WEEK adalah redundant karena ketika kita mengetahui tanggal kelahiran seseorang maka kita dapat menerangkan nama hari dia lahir.

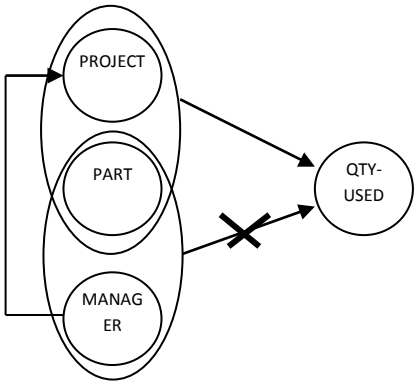
Aturan terakhir adalah pseudo-transivitas

Pseudo-Transivitas

Aturan pseudo-transivitas menyatakan bahwa:

Jika A →B AND BC→X THEN AC→X

Gamber 5.6 Pseudo-Transivitas



Pseudo-transivitas diilustrasikan dalam Gambar 5.6 menggunakan contoh yang lebih kongkrit. Disini:

PROJECT, PART → QTY-USED

(sebuah proyek menggunakan sejumlah barang (part) yang diberikan)

Dan

MANAGER →PROJECT

(proyek tersebut adalah hanya sebuah proyek yang diatur oleh seorang manejer)

Maka

MANAGER, PART →QTY-USED

(manejer bertanggun jawab untuk menggunakan jumlah barang yang diberikan)

Beberapa Aturan Turunan

Anda harus mencatat bahwa sangat mungkin untuk membuat aturan penilaian tambahan dari yang telah diberikan di atas. Sebenarnya, itu hanya diperlukan untuk memulai dengan tiga aturan di atas, yaitu refleksitas, augmentasi dan transivitas, dan turunkan aturan baru dari mereka. Sebagai contoh, kita dapat menurunkan aturan pseudo-transivitas sebagai berikut:

Misalkan kita diberikan A $\rightarrow$ B and BC$\rightarrow$X, maka dari A$\rightarrow$B kita mendapatkan AC$\rightarrow$BC dengan aturan augmentasi, dan dari AC $\rightarrow$ BC dan BC$\rightarrow$X kita mendapatkan AC$\rightarrow$X dengan transivitas.

Refleksivitas, augmentasi, dan transivitas kadang-kadang diketahui sebagai axioma sebagai mereka dapat digunakan untuk menurunkan aturan-aturan yang lain. Aturan-aturan lain yang berguna meliputi:

- Jika A$\rightarrow$B dan AB$\rightarrow$X maka A$\rightarrow$X dengan pseudo-transivitas, dan
- Jika A$\rightarrow$X dan B$\rightarrow$Y dan XY $\rightarrow$Z, maka AB$\rightarrow$Z

Kedua aturan ini diturunkan oleh sebuah aplikasi ganda aturan pseudo-transivitas sebagai berikut:

- Dari A $\rightarrow$X dan XY$\rightarrow$Z kita mendapatkan AY$\rightarrow$Z dengan pseudo-transivitas, dan kemudia dari B$\rightarrow$Y dan AY$\rightarrow$Z kita mendapatkan AB$\rightarrow$Z dengan pseudo-transivitas.

Aturan ketiga yang umum digunakan adalah dapat diturunkan dari aturan penilaian :

- Jika Z$\rightarrow$A dan B$\rightarrow$X, maka ZB$\rightarrow$AX

Aturan nyata sekali tapi dapat diturunkan dari aturan penilaian sebagai berikut.

Jika Z$\rightarrow$A (diberikan) maka ZB $\rightarrow$AB (dengan augmentasi)

Jika B$\rightarrow$X (diberikan) maka AB$\rightarrow$AX (dengan augmentasi)

Dari ZB$\rightarrow$AB dan AB$\rightarrow$AX kita mendapatkan ZB$\rightarrow$AX (dengan transitivitas)

Akan tetapi, ada peringatan. Jika anda diberikan AB$\rightarrow$AX maka anda tidak dapat membatalkan A untuk memberikan B$\rightarrow$X.

Sebagai contoh, jika:

PERSON-ID, DEPT-NAME$\rightarrow$PERSON-ID, DATE-STARTED

Maka tidak benar bahwa:

DEPT-NAME$\rightarrow$DATE-STARTED

Aturan-Aturan Penilaian dan Dependensi Banyak Nilai

Adalah hal yang penting pada tahapan ini untuk menunjukan bahwa aturan penilaian pada dependensi fungsional tidak berlaku untuk dependensi banyak nilai. Sebagai contoh, tidaklah benar bahwa jika:

PERSON-ID $\rightarrow\rightarrow$PROJECT, SKILL-USED maka

PERSON-ID→→PROJECT, dan

PERSON-ID→→ SKILL-USED

Disini dependensi banyak nilai pertama menggambarkan skill yang digunakan pada proyek oleh person. Adalah hal yang penting untuk sisi sebelah kanan untuk menyertakan pasangan SKILL-USED dan PROJECT karena kita menentukan skill apa yang dibutuhkan seseorang (person) pada proyek apa. Dengan demikian kita dapat mempunyai:

Jill menentukan (PROJ1, ACCOUNTING) dan (PROJ2, COMPUTING)

Dekomposisi menerangkan apa proyek seseorang dan skill apa yang mereka miliki. Pada contoh kita, dekomposisi dapat menunjukan bahwa:

Jill mempunyai skill COMPUTING dan ACCOUNTING dan

Jill bekerja pada PROJ1 dan PROJ2.

Informasi apa skill seseorang pada proyek sekarang hilang.

Juga terdapat himpunan aturan-aturan yang dikembangkan untuk dependensi banyak nilai tapi tidak dibicarakan disini karena mereka tidak digunakan dalam praktis.

Pengujian Dependensi Redundant

Ide umum dibelakang pengujian himpunan dependensi untuk redundasi adalah untuk melihat apakah dependensi fungsional dalam sebuah himpunan dapat diturunkan dari dependensi fungsional lain menggunakan aturan penilaian. Jika itu dapat dilakukan maka dependensi fungsional adalah redundant.

Contoh

Misalkan kita diberikan himpunan dependesi fungsional diilustrasikan dalam Gambar 5.7. Himpunan ini adalah:

J→K V→J VY→W KZ→W Y→Z

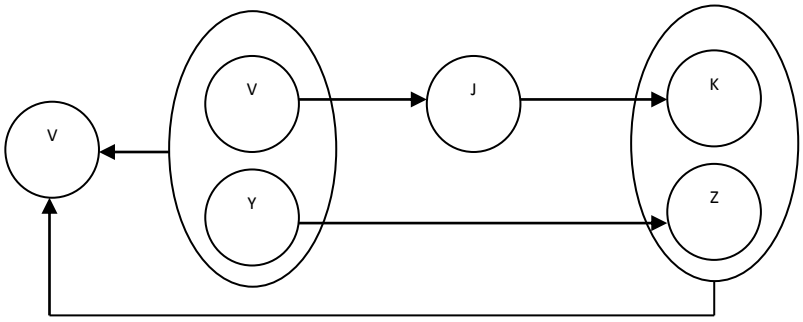
VY→W adalah redundant karena dapat diturunkan dari fungsional dependensi yang lain sebagai berikut:

V→J dan J →K memberikan V→K dengan hukum transitivitas

V→K dan KZ→W memberikan VZ→W dengan pseudo-transitivitas

Y→Z dan VZ→W memberikan VY →W dengan pseudo-transitivitas

Gambar 5.7 Penerapan hukum penilaian



Secara diagram dapat diperdebatkan bahwa kita tidak membutuhkan dependensi fungsional VY→W karena terdapat sebuah alternative alur dari VY ke W. Alur ini berangkat dari V ke K melalui J dengan transifitivitas dan dari VY ke Z. Satu kali KZ dicapai maka W mengikuti.

ALGORITMA MEMBERSHIP

Untuk melihat apakah sebuah dependensi fungsional X→Y (dimana X adalah himpunan atribut) adalah redundant, gunakan langkah berikut:

Step 1: Letakan variabel T pada X. T adalah sebuah variabel yang memuat sembarang jumlah atribut.

Step 2: Periksa setiap Dependensi Fungsional untuk melihat apakah determinantnya dalam T

> Jika iya, maka masukan RHS dari dependensi fungsional kedalam T. Jika tidak, maka pergi ke langkah 4.

Step 3: Setiap kali T dirubah, ulangi Step 2.

Step 4: Untuk melihat jika X→Y redundant, periksa jika Y dalam T. Jika iya, maka X→Y adalah redundant.

Algoritma ini harus diulangi untuk setiap dependensi fungsional dalam himpunan tersebut.

Contoh

Periksa sembaran dependensi redundant dalam setiap himpunan berikut:

Z→A, ZB→X, AX→Y, ZB→Y

Setiap dependensi fungsional harus diuji.

Apakah Z→A redundant?

Step 1: T= {Z}
Step 2: Tidak ada Dependensi fungsional lain yang mempunyai determinan dalam T
Step 3: Pergi ke Step 4 karena T tidak berubah
Step 4: A tidak dalam T dan sehingga Z→A tidak redundant.

Apakah ZB $\rightarrow$X redundant?
Step 1: T={Z,B}
Step 2: ZB $\rightarrow$Y mempunyai determinan dalam T. Dengan demikian tambahkan Y ke T dan T={Z,B,Y}.
Step 3: Ulangi langkah 2 karena T telah berubah
Step 2: Z$\rightarrow$A mempunyai penentu dalam T. Sehingga tambahkan A pada T dan T={Z,B,Y,A}
Step 3: Ulangi langkah 2 karena T telah berubah
Step 2: Tidak ada lagi dependensi fungsional penentu dalam T
Step 3: Lanjut ke langkah 4 karena T tidak berubah
Step 4: X tidak dalam T dan dengan demikian ZB$\rightarrow$X tidak redundant

Apakah AX $\rightarrow$Y redundant?
Step 1: T={A,X}
Step 2: Tidak ada dependesi fungsional lain mempunyai determinan dalam T

Apakah ZB$\rightarrow$Y redundant?
Step 1: T={Z,B}
Step 2: Z$\rightarrow$A mempunyai determinan dalam T. Sehingga tambahkan A pada T dan T={Z,B,A}
Step 3: Ulangi langkah 2 karena T telah berubah
Step 2: ZB$\rightarrow$X mempunyai determinan dalam T. Sehingga tambahkan X pada T dan T={Z,B,A,X}
Step 3: Ulangi langkah 2 karena T telah berubah
Step 2: AX$\rightarrow$Y mempunyai determinan dalam T. Sehingga tambahkan Y pada T dan T={Z,B,A,X,Y}
Step 3: Ulangi langkah 2 karena T telah berubah
Step 2: Tidak ada lagi dependensi fungsional dalam list
Step 3: Pergi ke langkah 4 karena T tidak berubah
Step 4: Y dalam T dan sehingga ZB$\rightarrow$Y adalah redundant

Beberapa hal penting tentang Algoritma Membership

1. Catat bahwa setelah T berubah anda harus melihat semua dependensi fungsional yang tertinggal lagi – walaupun anda telah melihat sebelumnya.
2. Urutan dimana dependensi fungsional diujikan tidaklah penting
3. Anda dapat mengakhiri ketika sisi sebelah kanan dari dependensi fungsional di bawah test menjadi bagian dari T
4. Ingat bahwa jika anda mempunyai dependensi fungsional dengan dua atribut pada sisi kanan, anda harus memisahkannya menjadi dua dependensi fungsional dan menguji mereka satu persatu.
   Dengan demikian A$\rightarrow$XY
   Menjadi A$\rightarrow$X dan
   A$\rightarrow$Y
   dan masing-masingnya diuji sebagai sebuah dependensi fungsional terpisah.
5. Setelah itu anda harus dapat menerapkan algoritma membership dengan cepat menggunakan metoda terpendek.

HIMPUNAN-HIMPUNAN MINIMAL

Sering terdapat lebih dari satu dependensi fungsional redundan dalam sebuah himpunan dependensi fungsional. Akan tetapi, adalah yang tidak mungkin untuk membuang masing-masing dependensi ini dari himpunan karena masing-masingnya redundant jika salah satu tetap ada dalam himpunan. Dalam kasus itu terdapat lebih dari satu himpunan minimal.

Sebagai contoh, ambil himpunan:

BJ→K, K→W, BJ→W, W→K.

Disini anda akan menemukan masing-masing BJ→K dan BJ→W adalah redundant. Akan tetapi, ketika BJ→K dibuang dari himpunan tersebut maka BJ→W tidak lagi redundant. Sama halnya, jika BJ→W dibuang dari himpunan tersebut maka BJ→K tidak lagi redundant.

Dengan demikian terdapat dua himpunan yang tidak mepunyai dependensi redundan:

1. BJ→K, K→W dan W→K dan
2. BJ→W, K→W dan W→K.

SEBUAH CARA UNTUK MENEMUKAN KUNCI RELASI

Sekarang kita kembali ke properti penting dari relasi, yaitu kunci relasi. Dalam Chapter 4 kunci relasi dibutuhhkan untuk menentukan bentuk normal dari relasi. Dalam chapter tersebut kunci relasi ditentukan oleh sebuah pengujian dari relasi yang diberikan. Juga memungkinkan untuk menentukan kunci relasi secara langsung dari dependensi fungsional.

Terdapat sebuah algoritma yang dapat menentukan sebuah kunci relasi dari dependensi fungsional. Algoritma ini diilustrasikan dengan penerapan himpunan dependensi fungsional:

R=({A,B,C,D},{AB→C, C→D, CB→AD,D→A})

Algoritma untuk Penentuan Kunci Relasi

Algorithma untuk menurunkan kunci relasi dari dpendensi diilustrasikan dalam Gambar 5.8 bersama dengan contoh. Dibuat dari tiga langkah berikut:

Gambar 5.8 Menemukan kunci relasi

LANGKAH-LANGKAH UNTUK MENEMUKAN KUNCI RELASI	CONTOH
Daftarkan semua dependensi fungsional (termasuk dependesi yang diberikan maupun sembarang dependensi yang dapat diturunkan dari dependensi yang diberikan	R=({A,B,C,D},{AB→C, C→D, CB→AD,D→A})
Buat sebuah himpunan dasar dari semua determinan	
Hapus atribut dari himpunan dasar (sembarang atribut yang dapt ditemukan dari atribut lain dieliminasi dari himpunan dasar	