BAB V

KETERGANTUNGAN FUNGSIONAL & NORMALISASI UNTUK BASIS DATA RELASIONAL

TUJUAN:

Mengupayakan untuk memperoleh skema relasi yang ”baik’ yaitu untuk mengukur secara formal mengapa satu set pengelompokan attribute menjadi sejumlah skema relasi adalah lebih baik dari yang lain.

5.1 PETUNJUK-PETUNJUK INFORMAL DALAM DESAIN SKEMA-SKEMA RELASI

Empat ”ukuran informal” mengenai kualitas desain skema relasi :

- Semantik dari attributes
- Reduksi nilai-nilai yang redundan (rangkap) dalam tuples
- Reduksi nilai-nilai null dalam tuples (record)
- Tidak mempunyai tuples yang aneh (spurious tuples)

5.1.1 Semantik dari Attribute

- Berkaitan dengan asumsi mengenai arti (semantic) tertentu yang diasosiakan terhadap sejumlah attribute yang membentuk suatu skema relasional.
- Semantik (arti) menjelaskan bagaimana mengiterprestasikan nilai-nilai attribute yang disimpan dalam suatu tuples dari suatu relasi (bagaimana nilai-nilai attribute dala suatu tuples berkaitan dengan yang lain).

❖ PETUNJUK 1:

- Desain suatu skema relasional sedemikian rupa sehingga semantik yang dikandungnya mudah untuk untuk dijelaskan.
- Jangan mengkombinasikan attribute-attribute dari sejumlah entity type dan relationship types menjadi satu relasi tunggal.
- Secara intuitif, jika suatu skema berkorespondensi dengan satu entity type atau satu relatioship type saja, maka arti yang dikandungnya akan cenderung menjadi lebih jelas.

➔ Skema-skema relasi dalam basis data COMPANY yang diberikan dalam contoh-contoh kuliah sebelumnya, merupakan skema relasi yang memenuhi petunjuk-1 di atas.

Dibawah ini diberikan contoh penurunan skema relasional yang menyalahi petunjuk-1 di atas :

a) EMP_DEPT (EName, SSN , BDate, DNumber, DName, DMgrSSN)
   ---- mengkombinasikan relasi-2 EMPLOYEE dan DEPARTMENT
b) EMP_PROJ (SSN, PNumber, Hours, EName, PName, PLocation)
   ---- mengkombinasikan relasi-2 WORKS_ON dan PROJECT

5.1.2 Informasi yang Redundan dan Update Anomalies

- Salah satu tujuan dari desain skema adalah untuk meminimumkan pemakaian storage yang dipakai oleh base relations (files).
- Pengelompokan sejumlah attribute menjadi skema-2 relasi yang baik mempunyai dampak yang berarti dalam mengurangi pemakaian storage.

EMP_DEPT (EName, SSN, Bdate. Dnumber, Dname, DMgrSSN)

Yang merupakan hasil NATURAL JOIN dari sebagian attribute EMPLOYEE dan DEPARTMENT, dan relasi :

EMP_PROJ (SSN, PNumber, Hours, EName, PName, PLocation)

Yang merupakan modifikasi dari relasi WORKS_ON dengan tambahan attribute dari PROJECT dan EMPLOYEE, akan membutuhkan pemakaian storage yang lebih besar, karena adanya pengulangan (repeating group) dari :

(DName, DNumber, DMgrSSN)

→ dalam relasi EMP_DEPT (untuk setiap employee dalam satu department Yang sama)

(PName, PLocation)

→ dalam relasi EMP_PROJ (untuk setiap employee yang bekerja dalam satu project yang sama)

Persoalan lain yang lebih serius dari kedua relasi di atas bilamana dijadikan sebagai base relations adalah timbulnya "Update Anomalies", yang meliputi : (3)

- Insertion anomalies
- Deletion anomalies
- Modification anomalies

** Insertion Anomalies.

Terdiri dari dua :

- Persoalan kecenderungan terjadinya inkonsistensi data.
  Sebagai contoh, dalam relasi EMP_DEPT, setiap kali suatu tuple baru ditambahkan, maka attribute-2 dari Department dimana seorang employee bekerja HARUS dituliskan secara tepat. Jika tidak maka akan terjadi nilai-2 yang inkonsisten untuk sejumlah nomor Department yang sama.
- Persoalan kesulitan penyisipan tuple yang baru
  Sebagai contoh, masih untuk relasi EMP_DEPT, jika suatu Department telah ada (didefinisikan) tapi belum ada employee di dalamnya, maka satu-satunya cara adalah dengan mengisi nilai-nilai NULL pada sejumlah attribute untuk employee. Tetapi cara ini menyalahi entity integrity constraint, dimana key dari suatu relasi tidak boleh bernilai NULL.

** Deletion Anomalies.

Anomali ini berkaitan erat dengan persoalan kedua dalam insertion anomalies; dimana untuk kasus relasi EMP_DEPT, jika suatu tuple employee yang merupakan satu-satunya employee untuk suatu Department dihapus, maka informasi mengenai Department akan terhapus dari basis data.

** Modification Anomalies.

Dalam relasi EMP_DEPT, jika nilai dari salah satu attribute employee untuk suatu Department tertentu diubah (misalnya nama department diubah), maka semua tuple employee yang bekerja pada Department tersebut juga harus diubah. Jika ada tuple yang tertingal tidak diubah, maka akn terdapat dua nama yang berbeda untuk satu department yang sama (yang seharusnya tidak boleh terjadi)

❖ PETUNJUK-2:

- Desain suatu skema relasi dasar (base relation schema) sedemikian rupa sehingga ketiga jenis anomali (insertion, deletion dan modification) tidak akan terjadi.

5.1.3 Nilai-nilai NULL dalam tuples

Dalam hasil desain suatu skema relasi, mungkin saja terdapat pengelompokan sejumlah attribute menjadi suatu relasi dengan jumlah attribute yang "besar". Jika terdapat sejumlah sub-set attributes yang tidak berlaku untuk semua tuple dalam relasi, maka akan terdapat sejumlah nilai-2 NULL dalam sejumlah tuples tersebut yang mengakibatkan:

- Pemborosan storage
- Timbulnya persoalan semantik dari attribute
- Kesulitan dalam merealisasikan operasi Join (kosong dengan kosong)
- Kesulitan dalam merealisasikan fungsi – fungsi agregate (seperti SUM, COUNT, dan AVERAGE)

Selain kesulitan-2 di atas, nilai – nilai NULL dapat memberikan interpretasi jamak (multiple, intepretations) terhadap tuple yang dalamnya terdapat attribute – attribute dengan nilai null :

- Attribute – 2 tersebut tidak terpakai untuk tuple
- Nilai – nilai attribute untuk tuple tidak diketahui
- Nilai – nilainya diketahui, tapi belum tercatat atau tersedia

❖ PETUNJUK 3:

- Sedapat mungkin, hindari penempatan attribute – attribute dalam suatu base relation yang memungkinkan timbulnya nilainya null. Jika nilai – nilai null tidak dapat dihindari, yakinkan bahwa hal tersebut hanya berlaku untuk kasus – kasus khusus dan jangan diberlakukan terhadap sebagaian besar dari tuple dalam suatu relasi

→ Sebagai contoh, jika hanya terdapat 10% dari keseluruhan employee yang mempunyai kantor pribadi, maka merupakan suatu cara perancangan yang beralasan apabila satu attribute OFFICE_NUMBER dimasukkan dalam relasi EMPLOYEE; tetapi akan lebih baik apabila dibuatkan satu relasi baru yang terdiri dari dua attribute (ESSN, OFFICE_NUMBER) yang dipakai untuk menyimpan data employee yang mempunyai kantor pribadi.

5.1.4 Tuples yang Tidak Dikehendaki (Spurious Tuples)

Untuk ini, seandainya relasi:

EMP_PROJ (SSN, PNumber, Hours, EName, PLocation)

Didekomposisi menjadi dua relasi:

EMP_LOCS (ENama, PLocation)

EMP_PROJ1 (SSN, PNUMBER, Hours, PName, PLocation)

Maka, jika kedua relasi hasil dekomposisi di atas diupayakan untuk dilakukan NATURAL JOIN (lewat attribute ”Plocation”), akan diperoleh sejumlah tuple yang melebihi (tidak ada) dalam EMP_PROJ.

Keadaan ini dapat terjadi karena terdapat sejumlah tuple hasil JOIN utuk “PLOCATION” yang sama, pasangan tuple “SSN” dan “EName” yang dihasilkan bukan merupakan pasangan yang valid.

Sejumlah tuple yang ada dalam hasil JOIN tetapi tidak ada dalam relasi EMP_PROJ disebut Spurious tuples, yaitu tuples yang tidak dikehendaki dan tidak valid.

❖ PETUNJUK 4:

- Dalam mendesain skema-skema relasi harus diupayakan sehingga skema-skema yang dihasilkan dapat dilakukan JOIN dengan kondisi keamanan (EQUI JOIN atau NATURAL JOIN) pada attribute-attribute yang berupa primary key atau foreign key, dengan cara yang menjamin bahwa squrious tuples tidak akan dihasilkan.

5.2 KETERGANTUNGAN FUNGSIONAL (FUNCTIONAL DEPENENCIES)

Konsep Ketergantungan Fungsional merupakan salah satu konsep yang angat penting dalam desain skema relasional, karena ini dapat secara formal mendefinisikan bentuk-bentuk relasi yang normal (normalisasi data).

5.2.1 Definisi Ketergantungan Fungsional

Ketergantungan fungsional merupakan satu constraint antara dua set attribute suatu basis data.

Jika suatu skema basis data relasional dengan n buah attribute dinyatakan dala bentuk unversal:

$$R = \{A_1, A_2, ..., A_{n-1}, A_n, \}$$

Maka ketergantungan fungsional (disingkat FD) antara dua set attribute X dan Y (keduanya subset dari R), dinotasikan X → Y, menyatakan suatu constraint pada sejumlah tuples yang yang meungkinkan dapat membentuk ”relation instance” r dari R; yaitu:

> *Untuk sembarangan pasangan tuples t_1 dan t_2 dalam r sedemikian rupa sehingga berlaku $t_1[X] = t_2[X]$, maka juga harus berlaku $t_1[Y] = t_2[Y]$. (menyatakan konsep key (FK, PK))*

Dari constraint di atas, dapat dikatakan bahwa nilai-nilai komponen tuple dari X dapat secara unik (atau secara fungsional) menentukan nilai-nilai dari komponen Y. Sebaliknya, dapat juga dikatakan bahwa Y secara fungsional tergantung pada X.

Jadi, X secara fungsional menentukan Y dalam suatu skema relasi R dan hanya jika, bilamana dua tuples dari r(R) mempunyai nilai X yang sama, maka kedua tuples ini juga harus mempunyai nilai Y sama

→ Perlu diingat bahwa:

- Jika suatu constraint pada R berlaku bahwa tidak boleh ada lebih dari satu tuple untuk suatu nilai X dalam sembarang relation instance r(R) (yaitu X merupakan candidate key dari R) mengisyaratkan bahwa X → Y untuk sembarang subset attribute Y dari R.

- Jika berlaku X → Y dalam R, hal ini tidak menyatakan bahwa apakah berlaku atau tidak Y → X dalam R.

→ Penggunaan utama dari konsep ketergantungan fungsional adalah untuk memberikan penjelasan lebih jauh suatu skema relasi R dengan menyatakan constraint pada sejumlah attributenya yang harus berlaku pada setiap saat.

Sebagai contoh, perhatikan skema relasi EMP_PROJ:

EMP_PROJ (SSN, Pnumber, Hours, EName, Pname, Plocation)

Yang dari semantik attributenya berlaku ketergantungan fungsional berikut:

(a) SSN → EName
(b) Pnumber → {Pname, Plocation}
(c) {SSN, Pnumber} → Hours

→ Ketergantungan fungsional merupakan sifat dari skema (intension) relasi R, bukan merupakan keadaan relasi tertentu(extension). Dengan demikian, suatu FD tidak dapat diturunkan secara otomatis dari suatu relasi, tetapi harus didefinisikan secara eksplisit oleh mereka yang mengerti semantik attribute dari relasi R.

5.2.2 Aturan-Aturan Penurunan (Inference Rules) untuk FD.

Suatu FD X→Y diturunkan dari satu set dependencies F dalam R jika X→Y berlaku dalam setiap keadaan relasi r ; yaitu bilamana r memenuhi semua dependencies dalam. F, maka X→Y juga berlaku dalam r.

Satu set dari semua functional dependencies yang dapat diturunkan dari F disebut: "Closure F+ dari F".

Untuk memperoleh cara yang sistematik dalam menurunkan dependencies, diperlakukan satu set "INFERENCE RULES" yang dapat digunakan untuk menurunkan dependencies yang baru dari satu set dependencies yang diberikan.

- Notasi F ⊨ X→Y digunakan untuk menyatakan bahwa functional dependency X→Y diturunkan dari satu set FDF.
- Untuk tujuan mempersingkatkan penulisan variable-variabel attribute, digunakan notasi:

  FD{X, Y}→ Z disingkat XY→Z

  FD{X, Y, Z}→{U,V} disingkat XYZ→UV

Terdapat 6 aturan(rumus) untuk functional dependencies:

1. Rumus Reflexive:
   Jika $X \supseteq Y$, maka $X \rightarrow Y$
2. Rumus Augmentation:
   $\{X \rightarrow Y\} \models XZ \rightarrow YZ$
3. Rumus Transitif;
   $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$
4. Rumus Decomposition(Projection):
   $\{X \rightarrow YZ\} \models X \rightarrow Y$
5. Rumus Union(Additive):
   $\{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$
6. Rumus Pseudotransitive:
   $\{X \rightarrow Y, WX \rightarrow Z\} \models WX \rightarrow Z$

Rumus 1 s/d 3 dikenal sebagai 'Armstrong's Inference Rules', dimana set dependence F dapat diturunkan hanya dengan menggunakan rumus-rumus 1 s/d 3 di atas (telah dibuktikan oleh Armstrong pada 1974)

➔Pembuktian keenam rumus di atas dibahas di kelas!

Algoritma mencari X+

(X + : closure of X under F)

- Biasanya, perancang basis data pertama mendifinisikan functional dependencies F yang dapat ditentukan dari semantik attribute dalam R
- Functional dependencies tambahan yang juga berlaku dalam R dapat diturunkan dengan menggunakan Armstrong's rule pada F

  ↳ secara sistematik dapat diperoleh dengan cara, pertama menentukan setiap set attribute X yang muncul disisi sebelah kiri dari FD dalam F yang kemudian dengan menggunakan Armstrong's rule cari semua attribute yang tergantung pada X.

Algoritmanya:

X+ := X;

REPEAT

Oldx+ := x+ ;

FOR each FD Y→Z dalam F DO

IF Y ≤ X+ THEN X+ :=X+ UZ;

UNTIL (oldx+ = x+);

Attribute-attribute yang bisa ditentukan disuatu attribute

CONTOH:

Perhatikan skema EMP_PROJ yang mempunyai satu set FD F berikut :

F = { SSN → EName,
Pnumber→ {Pname, Plocation},
{SSN, PNumber}→ Hours
}

Dengan menggunakan algoritma untuk menghitung X+ dengan berdasarkan pada F, maka diperoleh:

- {SSN}+ = {SSN, EName}
- {Pnumber}+ = {Pnumber, Pname, Plocation}
- {SSN, PNumber}+ = {SSN, PNumber, EName, PName, PLocation, Hours}

EMP_PROJ = { SSN, EName, PNumber, PName, PLocation, Hours}

1. SSN+ = {SSN}
   Ssn → Ename➔ SSN ≤ SSN+ ➔ SSN+ = {SSN}U{Ename}={SSN, Ename}
2. Pnumber+ = {Pnumber}
   Pnumber→{Pname, Plocation}➔ Pnumber ≤ Pnumber+ ➔ Pnumber+ = {Pnumber, Pname, Plocation}
3. {SSN, Pnumber}+ = → {SSN, PNumber}
   a. SSN→ Ename ➔ x+ = {SSN, Ename, Pnumber}
   b. Pnumber→ {Pname, Plocation}➔ x+ = {SSN, Ename, Pnumber, Pname, Plocation}
   c. {SSN, Pnumber}→ Hours ➔ x+ = {SSN, Ename, Pnumber, Pname, Plocation}

5.2.3. Set Functional Dependencies Yang Ekivalen

DEFINISI:

Satu set FD E dilingkup (covered) oleh satu set FD F (F melingkupi E), jika setiap FD dalam E juga ada dalam F+; yaitu E dilingkup oleh F jika setiap dependency dalam E dapat diturunkan dari F.

➔ Dua set functional dependencies E dan F dikatakan ekivalen (E=F) jika E+ = F+. ekivalen berarti bahwa setiap FD dalam E dapat diturunkan dari F, dan setiap FD dalam F dapat diturunkan dari E.

Jadi E=F jika kedua kondisi, yaitu E melingkupi F dan F melingkupi E terpenuhi.

➔ Untuk menentukan apakah F melingkupi E dapat dilakukan dengan ;

1. Menghitung x+ dengan berdasarkan pada F untuk setiap FD X→Y dalam E, maka dikatakan F melingkupi E.
2. Periksa apakah attribute – attribute dalam Y ada dalam X^+. Jika ”Ya” untuk setiap FD dalam D maka, dikatakan F melingkupi E.

5.2.4. Set Functional Depedencies Yang Minimal

Satu set functional dependencies F dikatakan minimal apabila memenuhi kondisi-kondisi:

a) Setiap dependency dalam F mempunyai satu atribute tunggal pada sisi kanannya (canonical form).
b) Sembarang dependency dalam F tidak dapat dihapus dan tetap mempertahankan bahwa satu set FD yang dihasilkan adalah ekivalen dengan F.
c) Sembarang dependency X→A tidak dapat diganti dengan satu dependency Y→A (di mana $Y \subset X$), dan tetap menghasilkan FD yang ekivalen dengan F.

Set FD yang minimal di atas dapat dipandang sebagai satu set dependencies dalam bentuk standar (atau canonical) tanpa redundansi.

|→ kondisi b) dan c) menjamin bahwa tidak ada redundansi dalam dependencies.

|→ kondisi a) menjamin bahwa setiap dependency ada dalam bentuk canonical dengan satu atribut tunggal pada sisi kanannya.

5.3 NORMALISASI DATA YANG DIDASARKAN PADA PRIMARY KEYS

Salah satu tujuan dari proses normalisasi data adalah untuk menjamin bahwa hasil dekomposisi suatu skema yang lebih kecil terhindar dari “update anomalies”.

Dalam perancangan basis data, normalisasi berperan sebagai:

- Kerangka kerja formal untuk menganalisa skema relasi yang didasarkan pada primary keys dan functional dependencies antar atributnya.
- Satu urutan test yang dapat dilakukan pada masing-masing skema relasi, sehingga basis data relasional dapat dinormalisasi ke suatu tingkat tertentu.

  Bilamana suatu test gagal, maka relasi yang menyalahi test tersebut harus didekomposisi menjadi sejumlah relasi yang masing-masing memenuhi kaidah normalisasi.

WARNING !!!!!!!!!!

- Bentuk-bentuk normal, jika hanya dilihat secara terisolasi dari faktor-faktor lain tidak menjamin diperoleh desain basis data yang baik.
- Secara umum, merupakan hal yang tidak cukup untuk memeriksa secara terpisah bahwa setiap relasi ada dalam bentuk normal tertentu, misalnya 3NF atau BCNF.

  Untuk ini, proses normalisasi melalui dekomposisi harus disertai dengan pemeriksaan bahwa sifat-sifat tambahan berikut juga berlaku dalam skema relasional:

  1. Sifat lossless join atau non-additive join

     →untuk menjamin bahwa persoalan spurious tuple tidak terjadi.
  2. Sifat dependency preservation

     →untuk menjamin bahwa semua functional dependencies tersaji dalam sejumlah relasi-relasi yang dihasilkan.

  → sifat (1) dan (2) di atas akan dibahas dalam bab berikutnya!

Bentuk-bentuk normal yang dibahas:

1 NF
2 NF
3 NF
BCNF
} didefinisikan hanya dengan memperhatikan functional dependencies dan key constraints

4 NF → + constraint multivalued dependency
5 NF → + constraint join dependency

DEFINISI-DEFINISI DASAR:

- Superkey dari suatu skema relasi R = {A1,A2,…,An} adalah satu set atribut S ⊂ R dengan sifat bahwa tidak ada dua tuple t_1 dan t_2 dalam sembarang keadaan relasi r dari R sehingga $t_1[s] = t_2[s]$.
- Suatu key K disebut superkey bilamana berlaku sifat tambahan bahwa penghapusan sembarang atribut dari K akan menyebabkan K tidak menjadi superkey lagi.

  ➔Jadi perbedaan antara key dan superkey adalah bahwa key harus "minimal", yaitu jika key K={A1,A2,…,Ak}, maka K-A1 bukan merupakan key untuk $1 \leq i \leq K$.
- Jika dalam suatu skema relasi terdapat lebih dari satu key, maka masing-masing disebut "candidate key". Salah satu candidate key harus dipilih menjadi primary key; dan yang lain disebut secondary key. Setiap relasi harus mempunyai primary key!
- Suatu atribut dalam skema relasi R disebut "prime atribut" dari R bilamana ia anggota dari sembarang key dari R.

  Sebuah atribut disebut non-prime jika ia bukan prima atribut; yaitu bukan anggota dari candidate key yang ada.

  →contoh: SSN dan PNUMBER, keduanya adalah prime atribut dalam relasi **works-on**

5.3.1. FIRST NORMAL FORM (1NF)

Secara histories, 1NF didefinisikan untuk tidak membolehkan adanya multivalued atribut atau composite atribut, atau kombinasi keduanya.

Dalam 1NF disebutkan bahwa domains dari atibut harus hanya terdiri dari atomic value (atomic = simple, indivisible); dan nilai dari sembarang atribut dalam suatu tuple harus berupa single value dari domain atribut tersebut.

Contoh:

EMPROJ (SSN,EName,{PROJ(PNumber,Hours)})

⇩ 1NF

EMP_PROJ1 (SSN,EName)

EMP_PROJ2(SSN,PNumber,Hours)

5.3.2 SECOND NORMAL FORM (2NF)

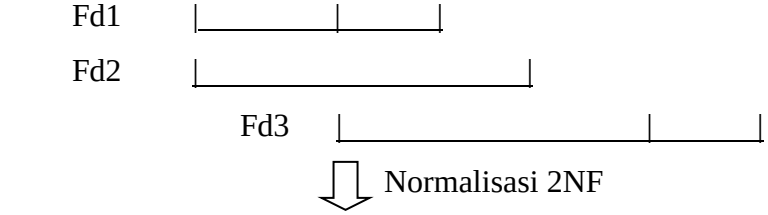
Didasarkan pada konsep "full functional dependency". FD X→Y disebut full functional dependency, jika penghapusan sembarang atribut A dari X menyebabkan sifat dependency tidak berlaku lagi; yaitu untuk sembarang atribut $A \in X$, maka $(X - \{A\}) \neq Y$.

Sebaliknya, FD X→Y disebut partial dependency jika beberapa atribut $A \in X$ dapat dihapus dari X dan tetap mempertahankan dependency yang ada; yaitu untuk $A \in X$, $(X-\{A\}) \rightarrow Y$.

Contoh:
{SSN,PNumber}→Hours (full dependency, karena: SSN-/->Hours; PNumber-/->Hours)
{SSN,PNumber}→EName (partial dependency, karena: SSN→EName)

DEFINISI:
Suatu skema relasi R berada dalam 2NF, jika setiap non-prime atributnya secara fungsional bergantung penuh pada key dari R.

Contoh:
EMPROJ(SSN,PNumber,Hours,EName,PName,PLocation)



EP1 (SSN,PNumber,Hours)
Fd1

EP2 (SSN,EName)
Fd2

EP3 (PNumber,PName,PLocation)
Fd3

KESIMPULAN:
Jika suatu skema relasi tidak berada dalam 2NF, ia dapat dinormalisasi lanjut menjadi sejumlah relasi 2NF dengan cara mengasosiasikan non-prime atribut yang ada hanya

dengan sebagian dari primary key di mana atribut-atribut tersebut secara fungsional bergantung penuh.

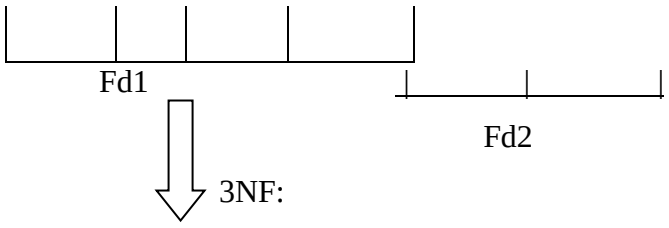
5.3.3. THIRD NORMAL FORM (3NF)

Didasarkan pada konsep "transitive dependency" suatu FD X→Y dalam suatu relasi R merupakan transitive dependency jika terdapat satu set atribut Z yang tidak merupakan subset dari sembarang key dari R, dan berlaku X→Z dan Z→Y.

Suatu skema relasi R dikatakan berada dalam 3NF, jika ia berada dalam 2NF dan tidak terdapat atribut dari R yang non-prime dan bergantung secara transitive pada primay key.

Contoh:

EMP_DEPT(EName,SSN,BDate,Address,DNumber,DName,DMgrSSN)



ED1(EName,SSN,BDate,Address,DNumber)

ED2 (DNumber,DName,DMgrSSN)

⇒ Operasi Natural Join terhadap ED1 dan ED2 akan menghasilkan relasi EMP_DEPT semula (tanpa menghasilkan spurious tuples).

DEFINISI UMUM 3NF:

Suatu skema relasi R ada dalam 3NF bilamana berlaku suatu FD X→A dalam R, dan memenuhi kondisi: (a) X adalah superkey dari R

atau

(b) A adalah prime atribut dari R

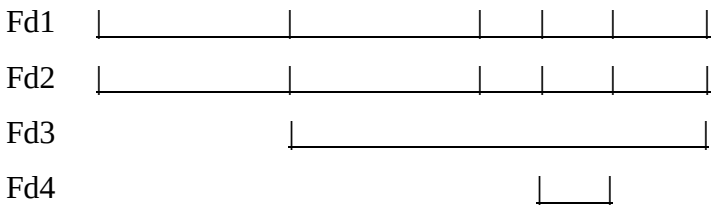
⇒ Implikasinya:

- Suatu skema relasi R menyalahi definisi umum 3NF, jika berlaku FD X→A dalam R yang menyalahi baik kondisi (a) dan (b).
- Suatu skema relasi R berada dalam 3NF, jika setiap non-prime atribut dari R:
    - Secara fungsional bergantung penuh pada setiap key dari R, dan
    - Secara non-transitive bergantung pada setiap key dari R.

⇒ Definisi umum dari 3NF dapat diaplikasikan secara langsung untuk memeriksa apakah suatu skema relasi berada dalam 3NF (tanpa perlu melalui pemeriksaan 2NF terlebih dahulu).

Contoh lain penurunan relasi 2NF dan 3NF:

LOTS(Property-ID#,County_Name,Lot#,Area,Price,Tax_Rate)



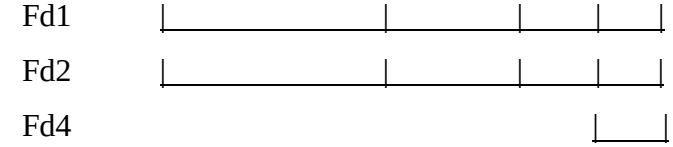
- menyajikan basis data “parcel of land for sale” dalam berbagai Negara bagian di USA.

- Ada 2 candidate keys:
    - Property_ID#
    - {County_Name,Lot#}

Melalui normalisasi 2NF:

- fd3 menyalahi aturan normalisasi 2NF:

LOTS1 (Property_ID#,County_Name,Lot#,Area,Price)



LOTS2 (<u>County_Name</u>,Tax_Rate)

Fd3 |_____|

- fd4 menyalahi aturan normalisasi 3NF:

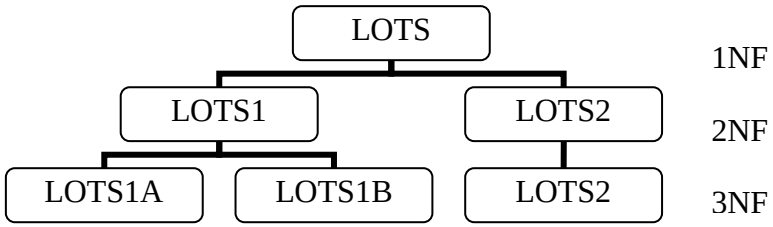
LOTS1A (<u>Property_ID#</u>,County_Name,Lot#,Area)

Fd1 |_____|_____|____|

Fd2 |_____|_____|____|

LOTS1B (<u>Area</u>,Price)

Fd4 |____|



<u>=➔Tanpa melalui hasil normalisasi 2NF:</u>

– Dari keempat FD yang ada (fd1-fd4) dalam skema relasi LOTS, terlihat bahwa fd3 dan fd4 menyalahi 3NF; sehingga dapat langsung didekomposisi menjadi LOTS1A, LOTS1B, dan LOTS2.

5.3.4. <u>Boyce-Codd Normal Form (Bcnf)</u>

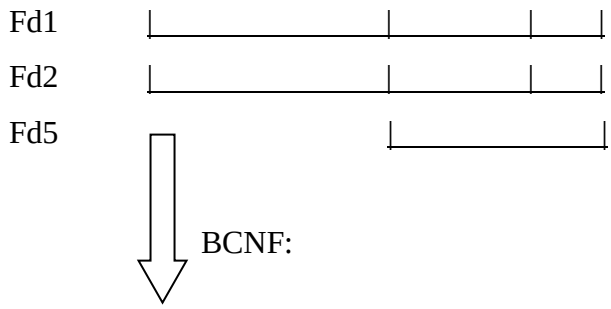
- BCNF merupakan bentuk normal yang <u>lebih ketat</u> (stricter) dibandingkan 3NFl; yang berarti bahwa setiap relasi dalam BCNF juga berada dalam 3NF, tetapi tidak sebaliknya.

- Suatu skema relasi berada dalam BCNF bilmana berlaku suatu FD X→A dalam R, maka X merupakan superkey dari R.

➔ Jadi, perbedaan bahwa BCNF dan 3NF adalah kondisi (b) dari 3NF yang membolehkan A untuk tidak prime jika X bukan superkay tidak boleh ada dalam BCNF.

Contoh:
Jika dalam relasi LOTS1A berlaku FD tambahan (fd5):

LOTS1A (Property_ID#,County_Name,Lot#,Area)



LOTS1AX (Property_ID#,Area,Lot#)
LOTS1AY (Area,County_Name)

Fd2 hilang dari hasil dekomposisi

KESIMPULAN AKHIR:

- Dalam praktek, kebanyakan skema relasi yang berada dalam 3NF juga berada dalam BCNF. Hanya jika ada dependency X→A dalam skema relasi R dengan X bukan superkey dan A adalah prime atribut, akan menjadikan R berada dalam 3NF tetapi tidak berarti berada dalam BCNF.

  Kasus umum:

  R(A,B,C) berada dalam 3NF, tetapi tidak dalam BCNF.

- Jadi, dalam proses normalisasi: upayakan untuk membentuk BCNF, dan jika tidak memungkinkan baru biarkan berada dalam 3NF.