Universitas Siber Asia

MODUL PEMBELAJARAN
PROGRAM STUDI
PJJ INFORMATIKA S1

Mata Kuliah	**:**	**Sistem Basis Data**
Bobot Sks	**:**	**3 sks**
Dosen Pengembang	**:**	**Cian Ramadhona Hassolthine, S.Kom., M.Kom**
Tutor	**:**	**Anita Ratnasari, S.Kom., M.Kom**
Capaian Pembelajaran Mata Kuliah	**:**	**Mahasiswa dapat memahami functional depedency dan primary key, foreign key dan contraint dari tabel.**
Kompetentsi Akhir Di Setiap Tahap (Sub-Cpmk)		**Mahasiswa dapat memahami *konsep functional depedency,* dan primary key, foreign key dan contraint dari tabel dan implementasinya pada tabel**
Minggu Perkuliahan Online Ke-		**Sesi 4**

FUNCTIONAL DEPEDENCY

1. Definisi *functional depedency*

Ketergantungan fungsional merupakan satu *constraint* antara dua *set attribute* suatu basis data. Jika suatu skema basis data relasional dengan n buah *attribute* dinyatakan dalam bentuk universal:

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

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 memungkinkan dapat membentuk *"relation instance"* r dari R; yaitu:

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

Dari *contraint* 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 fungsoional 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.

2. Contoh *functional depedency*

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, Pmumber} → 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.

3. 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 variabel-variabel *attribute*, digunakan notasi:
  - FD{X, Y}→ Z disingkat XY→Z
  - FD{X, Y, Z}→{U,V} disingkat XYZ→UV

4. Terdapat 6 Aturan untuk *Functional Dependencies:*

1. Rumus *Reflexive*:
   Jika X ≥ Y , maka X→Y
2. Rumus *Augmentation*:
   {X→Y} ⊨ XZ → YZ
3. Rumus Transitif;
   {X→Y, Y→Z} ⊨ X→Z
4. Rumus *Decomposition(Projection):*
   {X→YZ} ⊨ X→Y
5. Rumus *Union(Additive):*
   {X→Y, X→Z} ⊨ X→YZ
6. Rumus *Pseudotransitive:*
   {X→Y, WX→Z} ⊨ WX→Z

Rumus 1 s/d 3 dikenal sebagai *'Armstrong's Inference Rules'*, dimana set dependence F dapat diturunkan hanya dengan menggunakan rumus-rumus.
1s/d3 di atas (telah dibutuhkan oleh Armstrong pada 1974)

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. *Set Functional Dependencies* yang Ekivalen

DEFINISI:

Satu set FD E di lingkup *(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.

6. *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 YCX), 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.

7. Istilah -istilah Dasar

- *Superkey* dari suatu skema relasi R={A1,A2,…,An} adalah satu *set* atribut S C R dengan sifat bahwa **tidak ada** dua *tuple* t1 dan t2 dalam sembarang keadaan relasi r dari R sehingga t1[s]=t2[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 ≤ i ≤ 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: SSV dan DNUMBER, keduanya adalah prime atribut dalam relasi
- **works-on**

8. *Primary Key*

Primary Key merupakan sebuah aturan dimana fungsinya adalah untuk membedakan anatara baris satu dengan baris lainnya yang ada pada tabel dan **bersifat unik**.

Berikut adalah **contoh *primary key*** pada salah satu tabel.

Ada ketentuan yang harus diperhatikan ketika *field* yang menjadi *primary key* yaitu :

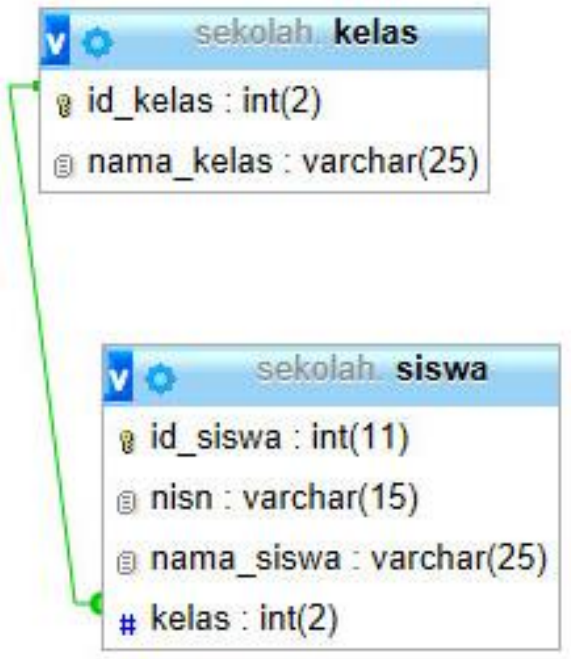
- Data tidak boleh sama atau ganda (unik)
- Data tidak boleh bernilai null

  Contoh sederhana penerapan ***primary key*** adalah seperti contoh diatas adalah **id**.

9. *Foreign Key*

Dari namanya bisa mengira bahwa ***foreign* (tamu) *key***, merupakan suatu atribut untuk **melengkapi hubungan yang menunjukan ke induknya**, itu artinya *field* pada tabel merupakan **kunci** tamu dari tabel lain. Dan biasanya penggunaan *foreign key* akan sangat dibutuhkan ketikan kita menemukan banyak tabel dan ingin menghubungkan satu tabel dengan tabel lainnya.

Contohnya seperti pada gambar dibawah ini **:**



10. *Candidate Key*

Suatu atribut ataupun ***super key*** yang mengidentifikasi secara unik untuk kejadian spesifik dari entitas.

Berikut ini adalah contoh *candidate key* :

NIP	Nama	NoStruk	Jumlah
P001	Maman	125	2.000,00
P002	Ujang	133	3.000,00
P003	Mumun	121	4.000,00
P004	Julaeha	142	1.000,00
P005	Kosim	123	5.000,00

11. REFERENSI/DAFTAR PUSTAKA

1. Juman, Kundang K. *Teknik Pencarian Fakta Dalam Perancangan Sistem Informasi*, 2012
2. Elmasri & Navathe. *Fundamental of Database Systems, 5thEdition, Chapter 4*, 2007
3. Bertalya. *MODEL EER (Enhanced Entity Relationship).* 2008
4. Korth, H & Mc Graw Hill. *Database System Concept, 4th edition*. New York