UNIVERSITAS INDONESIA

FAKULTAS ILMU KOMPUTER

12

Database Design Theory & Normalization
(part 1)

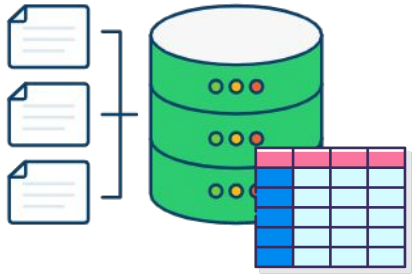
CSF2600700 - BASIS DATA

Acknowledgements

This slide is a modification to a slide titled
Database Design Theory & Normalization (part 1) used in “Basis Data” course in academic years 2019/2020 in the Faculty of Computer Science, Universitas Indonesia.

Tujuan Pembelajaran

Mengevaluasi model database relasional yang telah dibuat berdasarkan teori desain database relasional sehingga dapat menghasilkan skema database yang “baik” dalam level conceptual maupun physical.



Outline

1. Panduan Informal dalam Merancang Basis Data Relasional
2. Functional Dependency
3. Normalisasi Berdasarkan Primary Key
4. General Normal Form
2. Functional Dependency

Panduan Informal

Panduan secara umum berkaitan dengan:

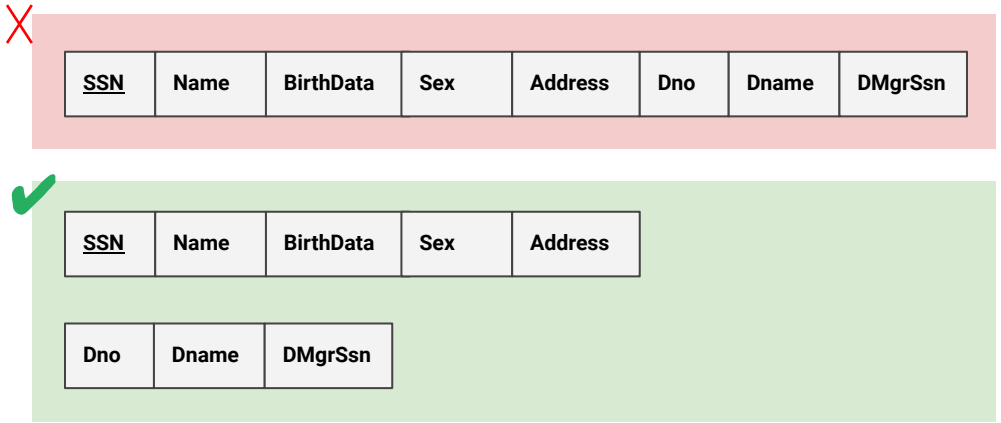
- ➔ **Semantik** atribut-atribut relasi
- ➔ Data yang **berulang** dan kaitannya dengan update anomaly
- ➔ Nilai `NULL`
- ➔ **Spurious tuples**

Panduan 1: Terkait Semantik Atribut-Atribut Relasi

Panduan 1

Setiap tuple pada relasi seharusnya merepresentasikan **satu entity atau relationship instance**. Skema perlu dirancang agar mudah dijelaskan relasi demi relasi. Semantik dari atribut sebaiknya mudah diinterpretasikan

Atribut-atribut dari entity yang berbeda-beda seharusnya tidak bergabung dalam satu relasi. Hanya foreign key yang digunakan untuk mengacu ke relasi lain Atribut-atribut entity dan relationship sebaiknya dipisahkan



Panduan 2: Terkait Data Berulang dan Update Anomaly

Panduan 2

Informasi yang disimpan berulang (redundant) menyebabkan **pemborosan ruang penyimpanan** dan berpotensi menyebabkan **update anomaly**.

Panduan 2: Terkait Data Berulang dan Update Anomaly (Cntd.)

- Relation EMP_DEPT berasal dari natural join antara 2 buah relation: EMPLOYEE dan DEPARTMENT.
- Menyimpan relasi hasil join ini dapat menyebabkan **update anomaly**

Redundancy

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

Panduan 2: Terkait Data Berulang dan Update Anomaly (Cntd.)

Update anomaly meliputi:

Insertion anomaly **Deletion anomaly** **Modification anomaly**

Panduan 2

Rancanglah skema yang **bebas dari anomali**. Jika terpaksa ada anomali, perlu dicatat dan ditangani oleh aplikasi

Panduan 2: Terkait Data Berulang dan Update Anomaly (Cntd.)

Insertion anomaly

Redundancy

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

Contoh: Insert new tuple of **Department**, (misal: departemen 'Production' dengan Dnumber: 2, dan Dmgr_ssn: '123456789')

NULL	NULL	NULL	NULL	2	Production	123456789

Harus ikut ditambahkan atau `NULL`
Tetapi jika Ssn: `NULL`, Entity Integrity Constraint dilanggar karena Ssn adalah primary key

Panduan 2: Terkait Data Berulang dan Update Anomaly (Cntd.)

Deletion anomaly

Redundancy

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

Jika tuple ini **dihapus**, maka departemen “Headquarters” akan **ikut terhapus**

Panduan 2: Terkait Data Berulang dan Update Anomaly (Cntd.)

Modification Anomaly

Redundancy

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

Mengubah nilai atribut departemen “Research” pada tuple di baris ini **harus diikuti dengan pengubahan pada tuple pada baris lain** yang menyimpan informasi departemen “Research” (baris 1,2, dan 5)

Panduan 3: Terkait Nilai NULL

Nilai NULL pada suatu atribut:

- ➔ **Pemborosan storage** di physical level
- ➔ Terjadi masalah ketika dijalankan dilakukan JOIN atau fungsi aggregate (misalnya SUM, COUNT)
- ➔ Tidak dapat dibedakan dengan jelas penyebab nilai NULL (does not apply, unknown, or known but absent ?)

Panduan 3

Relations harus dirancang agar nilai **null** yang ada pada baris-barisnya **sesedikit mungkin**

Panduan 4: Terkait Spurious Tuples

Dalam merancang, kadang-kadang kita melakukan dekomposisi relasi maupun penggabungan beberapa relasi. Ada 2 hal yang perlu diperhatikan

- ➔ Apakah dekomposisi (pemecahan) sebuah relasi menjadi dua/lebih relasi baru akan **menyebabkan adanya data yang hilang?**
- ➔ Apakah penggabungan dua/lebih relasi menjadi sebuah relation baru akan **menghasilkan kelebihan data (yang sebenarnya tidak ada)?**

Panduan 4: Terkait Spurious Tuples (Cntd.)

Redundancy Redundancy

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation
123456789	1	32.5	Smith, John B.	ProductX	Bellaire
123456789	2	7.5	Smith, John B.	ProductY	Sugarland
666884444	3	40.0	Narayan, Ramesh K.	ProductZ	Houston
453453453	1	20.0	English, Joyce A.	ProductX	Bellaire
453453453	2	20.0	English, Joyce A.	ProductY	Sugarland

EMP_LOCS

Ename	Plocation
Smith, John B.	Bellaire
Smith, John B.	Sugarland
Narayan, Ramesh K.	Houston
English, Joyce A.	Bellaire
English, Joyce A.	Sugarland

EMP_PROJ1

Ssn	Pnumber	Hours	Pname	Plocation
123456789	1	32.5	ProductX	Bellaire
123456789	2	7.5	ProductY	Sugarland
666884444	3	40.0	ProductZ	Houston
453453453	1	20.0	ProductX	Bellaire
453453453	2	20.0	ProductY	Sugarland

Panduan 4: Terkait Spurious Tuples (Cntd.)

Contoh: Hasil Natural Join antara EMP_LOCS dengan EMP_PROJ1

	Ssn	Pnumber	Hours	Pname	Plocation	Ename
	123456789	1	32.5	ProductX	Bellaire	Smith, John B.
*	123456789	1	32.5	ProductX	Bellaire	English, Joyce A.
	123456789	2	7.5	ProductY	Sugarland	Smith, John B.
*	123456789	2	7.5	ProductY	Sugarland	English, Joyce A.
*	123456789	2	7.5	ProductY	Sugarland	Wong, Franklin T.
	666884444	3	40.0	ProductZ	Houston	Narayan, Ramesh K.
*	666884444	3	40.0	ProductZ	Houston	Wong, Franklin T.
*	453453453	1	20.0	ProductX	Bellaire	Smith, John B.
	453453453	1	20.0	ProductX	Bellaire	English, Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Smith, John B.
	453453453	2	20.0	ProductY	Sugarland	English, Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Wong, Franklin T.
*	333445555	2	10.0	ProductY	Sugarland	Smith, John B.
*	333445555	2	10.0	ProductY	Sugarland	English, Joyce A.
	333445555	2	10.0	ProductY	Sugarland	Wong, Franklin T.
*	333445555	3	10.0	ProductZ	Houston	Narayan, Ramesh K.
	333445555	3	10.0	ProductZ	Houston	Wong, Franklin T.
	333445555	10	10.0	Computerization	Stafford	Wong, Franklin T.
*	333445555	20	10.0	Reorganization	Houston	Narayan, Ramesh K.

Panduan 4: Terkait Spurious Tuples (Cntd.)

Panduan 4

Relasi harus dirancang agar memenuhi lossless join condition

- ➔ Tidak ada spurious tuples yang dihasilkan oleh natural join antar relasi
- ➔ Hindari relasi yang memiliki matching attribute selain kombinasi PK dan FK.

EMP_LOCS

Ename	Plocation
Smith, John B.	Bellaire
Smith, John B.	Sugarland
Narayan, Ramesh K.	Houston
English, Joyce A.	Bellaire
English, Joyce A.	Sugarland

EMP_PROJ1

Ssn	Pnumber	Hours	Pname	Plocation
123456789	1	32.5	ProductX	Bellaire
123456789	2	7.5	ProductY	Sugarland
666884444	3	40.0	ProductZ	Houston
453453453	1	20.0	ProductX	Bellaire
453453453	2	20.0	ProductY	Sugarland

Panduan Informal - A Summary

Akibat yang ditimbulkan karena kesalahan mendefinisikan suatu relasi:

Redundancy → Update Anomalies

Munculnya nilai `NULL` → Pemborosan storage / kesulitan dalam melakukan JOIN/aggregate function

Kesalahan proses dekomposisi → Spurious tuples

Outline

1. Panduan Informal dalam Merancang Basis Data Relasional

2. Functional Dependency

3. Normalisasi Berdasarkan Primary Key

4. General Normal Form

2. Functional Dependency

Definisi

Def: Let A,B be sets of attributes
We write A $\rightarrow$ B or say A functionally determines B if, for any tuples t_1 and t_2 : t_1 [A] = t_2 [A] implies t_1 [B] = t_2 [B] and we call A $\rightarrow$ B a functional dependency

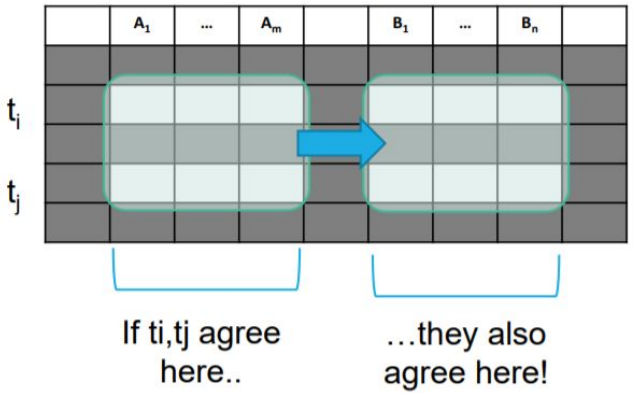
A$\rightarrow$B means that "**whenever two tuples agree on A then they agree on B**."

A Picture Of FDs

Def (again): Given attribute sets A={A_1 ,…,A_m} and B = {B_1 ,…B_n } in R,

The functional dependency A→ B on R holds if for any t_i ,t_j in R:

if t_i [A_1] = t_j [A_1] AND t_i [A_2]=t_j [A_2] AND … AND t_i [A_m] = t_j [A_m] then
t_i [B_1] = t_j [B_1] AND t_i [B_2]=t_j [B_2] AND … AND t_i [B_n] = t_j [B_n]



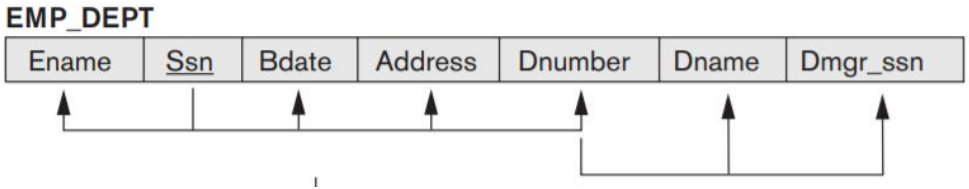
Contoh Functional Dependency

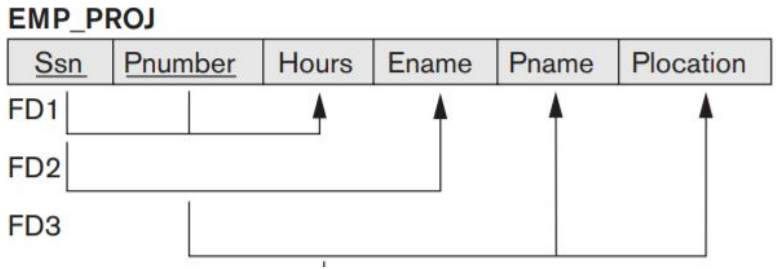
TEACH

Teacher	Course	Text
Smith	Data Structures	Bartram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

Figure 14.7
A relation state of TEACH with a *possible* functional dependency TEXT → COURSE. However, TEACHER → COURSE, TEXT → TEACHER and COURSE → TEXT are ruled out.

Contoh Functional Dependency (Cntd.)





Pada relasi **EMP_PROJ** terdapat 3 FD:

- ➔ {SSN, PNUMBER} → HOURS
- ➔ SSN → ENAME
- ➔ PNUMBER → {PNAME, PLOCATION}

Bagaimana menemukan FD lainnya?

Contoh:

Products

Name	Color	Category	Dep	Price
Gizmo	Green	Gadget	Toys	49
Widget	Black	Gadget	Toys	59
Gicmo	Green	Whatsit	Garden	99

Provided FDs:

1. **{Name} → {Color}**
2. **{Category} → {Department}**
3. **{Color, Category} → {Price}**

Given the provided FDs, we can see that {Name, Category} → {Price} must also hold on any instance…

Which / how many other FDs do?

Bagaimana menemukan FD lainnya? (Cntd.)

Sama saja bertanya: Given a set of FDs, F = {f_1 ,...f_n }, does an FD x hold?

Inference problem: How do we decide?

Answer:

Three simple rules called Armstrong's Rules.
1. Reflexive 2. Augmentation 3. Transitive

Inference Rule untuk FD

Amstrong's Inference Rule

1. Reflexive (IR1): If $\{X \supseteq Y\}$ then $X \rightarrow Y$.
2. Augmentation (IR2): If $\{X \rightarrow Y\} \models XZ \rightarrow YZ$
3. Transitive (IR3): If $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$

Derived Inference Rule

4. Decomposition: If $\{X \rightarrow YZ\} \models X \rightarrow Y$
5. Additive (Union): If $\{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$
6. Pseudotransitive: If $\{X \rightarrow Y, WY \rightarrow Z\} \models WX \rightarrow Z$

Inference Rule untuk FD (Cntd.)

Provided FDs:

1. {Name} → {Color}
2. {Category} → {Department}
3. {Color, Category} → {Price}

Amstrong's Inference Rule

1. Reflexive (IR1): If {X ⊇ Y} then X → Y.
2. Augmentation (IR2): If {X → Y} |= XZ → YZ
3. Transitive (IR3): If {X→ Y, Y→ Z} |= X → Z

Derived Inference Rule

4. Decomposition: If {X→ YZ} |= X → Y
5. Additive (Union): If {X→ Y, X→ Z} |= X → YZ
6. Pseudotransitive: If {X→ Y, WY→ Z} |= WX → Z

Inferred FDs:

Inferred FD	Rule used
4. {Name, Category} -> {Name}	Reflexive
5. {Name, Category} -> {Color}	Transitive
6. {Name, Category} -> {Category}	Reflexive
7. {Name, Category -> {Color, Category}	Union
8. {Name, Category} -> {Price}	Transitive

Inference Rule untuk FD (Cntd.)

Manakah FD yang didapatkan jika diberikan FD sbb.

F = {SSN → {EName, BDate, Address, DNumber}, {SSN, ProjNo} → Hours }

- SSN → Ename
- SSN → Bdate
- SSN → Address
- SSN → Dnumber
- SSN → SSN
- {SSN, ProjNo} → ProjNo ……dst

Amstrong's Inference Rule

1. Reflexive (IR1): If {X ⊇ Y} then X → Y.
2. Augmentation (IR2): If {X → Y} |= XZ → YZ
3. Transitive (IR3): If {X→ Y, Y→ Z} |= X → Z

Derived Inference Rule

4. Decomposition: If {X→ YZ} |= X → Y
5. Additive (Union): If {X→ Y, X→ Z} |= X → YZ
6. Pseudotransitive: If {X→ Y, WY→ Z} |= WX → Z

Closure (Penutup)

Closure dari himpunan functional dependency F adalah
himpunan F^+ yang beranggotakan F dan semua FD yang dapat diturunkan dari F

Closure dari himpunan attribute X yang berkaitan dengan F adalah
himpunan X+ yang beranggotakan semua attribute yang ditentukan secara fungsional oleh X

X^+ dapat dicari dengan menerapkan secara berulang IR1, IR2, IR3 dengan menggunakan FD yang ada pada F

Closure Algorithm

Start with Closure=A.

Until closure doesn't change do:

if $A_1, A_2, \ldots A_n \longrightarrow$ B is in C, **and**

$A_1, A_2, \ldots A_n$ Are all in the closure, **and**
B is not in Closure

then

add B to closure.

Closure Algorithm (Cntd.)

Start with Closure=A.

Until closure doesn't change do:

if $A_1, A_2, \ldots A_n \longrightarrow B$ is in C, **and**

$A_1, A_2, \ldots A_n$ Are all in the closure, **and**

B is not in Closure

then

add B to closure.

{name, category}$^+$ =
{name, category}

F =

{{name} → {color},
{category} → {dept},
{color, category} → {price}}

Closure Algorithm (Cntd.)

Start with Closure=A.

Until closure doesn't change do:

if $A_1, A_2, \ldots A_n \longrightarrow B$ is in C, **and**

$A_1, A_2, \ldots A_n$ Are all in the closure, **and**

B is not in Closure

then

add B to closure.

{name, category}$^+$ =
{name, category}

{name, category}$^+$ =
{name, category, color}

F =

{{name} → {color},
{category} → {dept},
{color, category} → {price}}

Closure Algorithm (Cntd.)

Start with Closure=A.

Until closure doesn't change do:

if $A_1, A_2, \ldots A_n \longrightarrow B$ is in C, **and**

$A_1, A_2, \ldots A_n$ Are all in the closure, **and**

B is not in Closure

then

add B to closure.

F =

{{name} → {color},
{category} → {dept},
{color, category} → {price}}

{name, category}$^+$ =
{name, category}

{name, category}$^+$ =
{name, category, color}

{name, category}$^+$ =
{name, category, color, dept}

Closure Algorithm (Cntd.)

Start with Closure=A.

Until closure doesn't change do:

if $A_1, A_2, \ldots A_n \longrightarrow B$ is in C, **and**

$A_1, A_2, \ldots A_n$ Are all in the closure, **and**

B is not in Closure

then

add B to closure.

F =

{{name} → {color},
{category} → {dept},
{color, category} → {price}}

{name, category}$^+$ =
{name, category}

{name, category}$^+$ =
{name, category, color}

{name, category}$^+$ =
{name, category, color, dept}

{name, category}$^+$ =
{**name, category, color, dept, price**}

Contoh Closure

Diberikan
F = {
SSN → ENAME,
PNUMBER → {PNAME,
PLOCATION}, {SSN, PNUMBER} →
HOURS
}

- {SSN}+ = {SSN, ENAME}
- {PNUMBER}+ = {PNUMBER, PNAME, PLOCATION }
- {SSN, PNUMBER}+ = {SSN, PNUMBER, ENAME, PNAME, PLOCATION, HOURS}

Cara lain menentukan Closure

Menentukan $\{SSN, PNUMBER\}^+$

Apakah $\{SSN, PNUMBER\} \supseteq SSN$?
Ya – Tambahkan ENAME

Apakah $\{SSN, PNUMBER, ENAME\} \supseteq PNUMBER$?
Ya – Tambahkan PNAME, PLOCATION

Apakah $\{SSN, PNUMBER, ENAME, PNAME, PLOCATION\} \supseteq \{SSN, PNUMBER\}$?
Ya – Tambahkan HOURS

Sehingga **$\{SSN, PNUMBER\}^+$ = {SSN, PNUMBER, ENAME, PNAME, PLOCATION, HOURS}**

Contoh Closure

Diketahui G =
{ SSN → {ENAME, BDATE, ADDRESS, DNUMBER}, DNUMBER → {DNAME, DMGRSSN} }

➔ Carilah $\{DNUMBER\}^+$

...

➔ Carilah $\{SSN\}^+$

...

Contoh Closure

R(A,B,C,D,E,F)

$\{A,B\} \rightarrow \{C\}$
$\{A,D\} \rightarrow \{E\}$
$\{B\} \rightarrow \{D\}$
$\{A,F\} \rightarrow \{B\}$

Compute $\{A,B\}^+ = \{A, B,$ $\}$

Compute $\{A,F\}^+ = \{A, F,$ $\}$

Menginterpretasikan Himpunan Closure

Apa yang ekuivalen dengan $\{SSN\}^+ = \{SSN, ENAME\}$?

SSN → ENAME, SSN → SSN, SSN → SSN, ENAME

Bagaimana dengan $\{PNUMBER\}^+ = \{PNUMBER, PNAME, PLOCATION\}$?

PNUMBER → PNAME
PNUMBER → PLOCATION
PNUMBER → PNUMBER
PNUMBER → {PNAME, PLOCATION}
PNUMBER → {PNUMBER, PLOCATION}
PNUMBER → {PNAME, PNUMBER}
PNUMBER → {PNUMBER, PNAME, PLOCATION}

Bagaimana dengan $\{SSN, PNUMBER\}^+ =$
{SSN, PNUMBER, ENAME, PNAME, PLOCATION, HOURS} ?

Kenapa kita membutuhkan closure?

With closure **we can find all FD's easily**

To check if X → A

1. **Compute X^+**
2. **Check if A ∈ X^+**

Note here that X is a set of attributes, but A is a single attribute.
Why does considering FDs of this form suffice?

Recall the Split (Decomposition) / combine (Union/Additive) rule:

4. Decomposition:	If {X→ YZ} |= X → Y
5. Additive (Union):	If {X→ Y, X→ Z} |= X → YZ

Kenapa kita membutuhkan closure? (Cntd.)

For each set of attributes X

1. Compute X^+
2. If X^+ = set of all attributes then **X is a superkey**
3. If X is minimal, then it is a **key**

Temukan key-nya

Product(name, price, category, color)

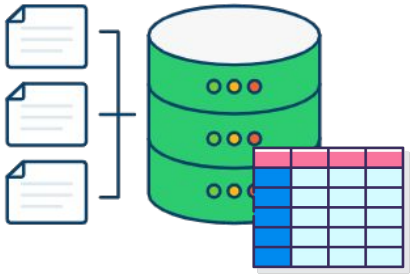
{name, category} → price
{category} → color

What is a key?

Summary

- Panduan informal untuk Relational DB
- Functional Dependency
- Closure

Next? **DB Normalization**



Q&A