Relational Database Design Theory (II)

CS348 Spring 2023

Instructor: Sujaya Maiyya

Sections: **002 & 004 only**

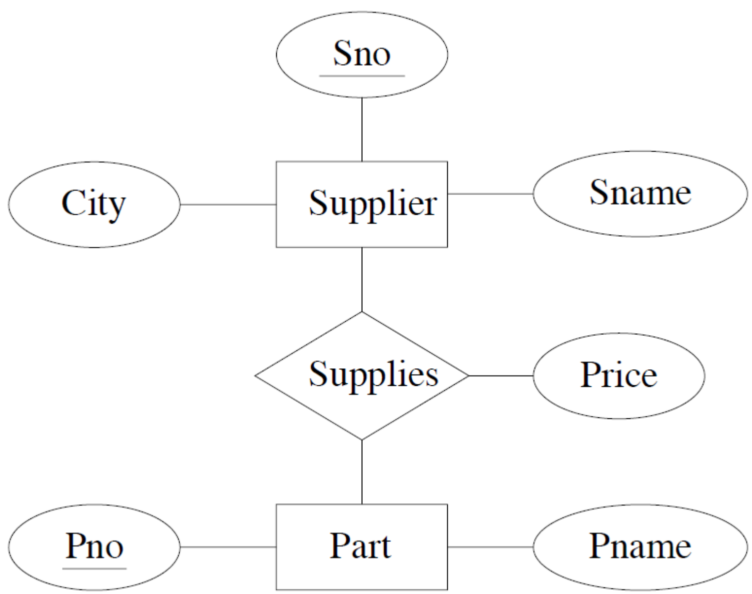
Outline For Today

1. Application Constraints and Decompositions
2. Functional Dependencies
3. Boyce-Codd Normal Form (BCNF) & BCNF Decomposition Alg.
4. Dependency Preservation and 3rd Normal Form

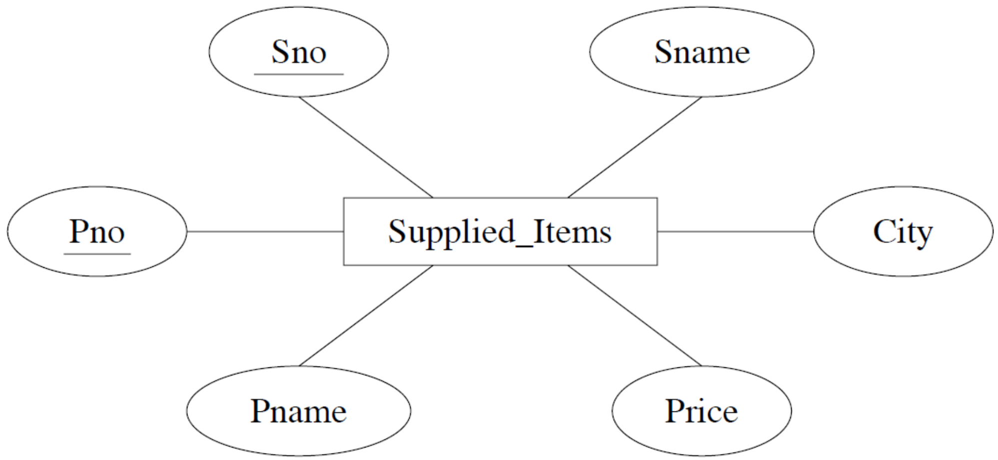
This lecture

A Parts/Suppliers database example

- Each type of part has a name and an identifying number and may be supplied by zero or more suppliers.
- Each supplier has an identifying number, a name, and a contact location for ordering parts.
- Each supplier may offer the part at a different price.



Single table?



Supplied_Items

Sno	Sname	City	Pno	Pname	Price
S1	Magna	Ajax	P1	Bolt	0.50
S1	Magna	Ajax	P2	Nut	0.25
S1	Magna	Ajax	P3	Screw	0.30
S2	Budd	Hull	P3	Screw	0.40

Decomposed tables?

- An instance

Suppliers

Sno	Sname	City
S1	Magna	Ajax
S2	Budd	Hull

Parts

Pno	Pname
P1	Bolt
P2	Nut
P3	Screw

Supplies

Sno	Pno	Price
S1	P1	0.50
S1	P2	0.25
S1	P3	0.30
S2	P3	0.40

Schema decomposition

- Let R be a relation schema (= set of attributes).
- The collection $\{R_1, \dots, R_n\}$ of relations is a decomposition of R if $R = R_1 \cup \cdots \cup R_n$

Supplied_Items (R)

Sno	Sname	City	Pno	Pname	Price
S1	Magna	Ajax	P1	Bolt	0.50
S1	Magna	Ajax	P2	Nut	0.25
S1	Magna	Ajax	P3	Screw	0.30
S2	Budd	Hull	P3	Screw	0.40

Suppliers (R1)

Sno	Sname	City
S1	Magna	Ajax
S2	Budd	Hull

Parts (R2)

Pno	Pname
P1	Bolt
P2	Nut
P3	Screw

Supplies (R3)

Sno	Pno	Price
S1	P1	0.50
S1	P2	0.25
S1	P3	0.30
S2	P3	0.40

- What is a good decomposition?

Is this a good decomposition?

- Example 1

Marks

Student	Assignment	Group	Mark
Ann	A1	G1	80
Ann	A2	G3	60
Bob	A1	G2	60

SGM

Student	Group	Mark
Ann	G1	80
Ann	G3	60
Bob	G2	60

AM

Assignment	Mark
A1	80
A2	60
A1	60

Natural Join

Student	Assignment	Group	Mark
Ann	A1	G1	80
Ann	A2	G3	60
Ann	A1	G3	60
Bob	A2	G2	60
Bob	A1	G2	60

But computing the natural join of SGM and AM, we get extra data (spurious tuples).

We would therefore lose information if we were to replace Marks by SGM and AM

"Good" Schema Decomposition

- Lossless-join decompositions
  - We should be able to construct the instance of the original table from the instances of the tables in the decomposition

A decomposition $\{R_1, R_2\}$ of R is lossless iff the common attributes of R_1 and R_2 form a superkey for either schema,

$$R_1 \cap R_2 \rightarrow R_1 \text{ or } R_1 \cap R_2 \rightarrow R_2$$

**If X is a superkey of R , then $X \rightarrow R$ (all the attributes) [last lecture]*

Is this a lossless join decomposition?

- Example 1
  - $R = \{Student, Assignment, Group, Mark\}$

Student	Assignment	Group	Mark
Ann	A1	G1	80
Ann	A2	G3	60
Bob	A1	G2	60

$\mathcal{F}$ includes:
$Student, Assignment \rightarrow Group, Mark$

  - $R_1 = \{Student, Group, Mark\}, R_2 = \{Assignment, Mark\}$

R1

Student	Group	Mark
Ann	G1	80
Ann	G3	60
Bob	G2	60

R2

Assignment	Mark
A1	80
A2	60
A1	60

$R_1 \cap R_2 = \{Mark\}$ is not a superkey of either R_1 or R_2

→ This decomposition is lossy

Which one is a better decomposition?

- Example 2: a table for a company database
  - $R = \{Proj, Dept, Div\}$

$\mathcal{F}$ includes:

FD1: $Proj \rightarrow Dept$ FD2: $Dept \rightarrow Div$ FD3: $Proj \rightarrow Div$

- Consider 2 decompositions

$$D_1 = \begin{Bmatrix} R_1\{Proj, Dept\}, \\ R_2\{Dept, Div\} \end{Bmatrix} \qquad D_2 = \begin{Bmatrix} R_1\{Proj, Dept\}, \\ R_2\{Proj, Div\} \end{Bmatrix}$$

- Both are lossless. (Why?) $R_1 \cap R_2 \rightarrow R_1$ or R_2

- However, testing FDs is easier on one of them. (Which?)

Testing FDs

- Example 2: a table for a company database
  - $R = \{Proj, Dept, Div\}$

$\mathcal{F}$ includes:

FD1: $Proj \rightarrow Dept$ FD2: $Dept \rightarrow Div$ FD3: $Proj \rightarrow Div$

- Consider 2 decompositions

$$\boxed{D_1 = \begin{Bmatrix} R_1\{Proj, Dept\}, \\ R_2\{Dept, Div\} \end{Bmatrix}}$$

$$D_2 = \begin{Bmatrix} R_1\{Proj, Dept\}, \\ R_2\{Proj, Div\} \end{Bmatrix}$$

- FD1 (in R1)
- FD2 (in R2)
- FD3 (join R1 and R2?)
- → No need, if FD1 and FD2 hold, then FD3 hold

Testing FDs

- Example 2: a table for a company database
  - $R = \{Proj, Dept, Div\}$

$\mathcal{F}$ includes:

FD1: $Proj \rightarrow Dept$ FD2: $Dept \rightarrow Div$ FD3: $Proj \rightarrow Div$

- Consider 2 decompositions

$$D_1 = \begin{Bmatrix} R_1\{Proj, Dept\}, \\ R_2\{Dept, Div\} \end{Bmatrix}$$

$$\boxed{D_2 = \begin{Bmatrix} R_1\{Proj, Dept\}, \\ R_2\{Proj, Div\} \end{Bmatrix}}$$

- FD1 (in R1)
- FD2 (in R2)
- FD3 (join R1 and R2?)
- → No need, if FD1 and FD2 hold, then FD3 hold

interrelational

- FD1 (in R1)
- FD3 (in R2)
- FD2 (join R1 and R2?)

→ Yes. FD1 and FD3 are not sufficient to guarantee FD2

Testing FDs

- Example 2: a table for a company database
  - $R = \{Proj, Dept, Div\}$

$\mathcal{F}$ includes:

FD1: $Proj \rightarrow Dept$ FD2: $Dept \rightarrow Div$ FD3: $Proj \rightarrow Div$

- Consider 2 decompositions

$$D_1 = \begin{Bmatrix} R_1\{Proj, Dept\}, \\ R_2\{Dept, Div\} \end{Bmatrix} \qquad D_2 = \begin{Bmatrix} R_1\{Proj, Dept\}, \\ R_2\{Proj, Div\} \end{Bmatrix}$$

- FD1 (in R1)
- FD2 (in R2)
- FD3 (join R1 and R2?)
- → No need, if FD1 and FD2 hold, then FD3 hold

(i) Equivalent to $\mathcal{F}$
(ii) Not interrelational

FD1 (in R1)
FD3 (in R2)
- FD2 (join R1 and R2?)

→ Yes. FD1 and FD3 are not sufficient to guarantee FD2

interrelational

"Good" Schema Decomposition

- Lossless-join decompositions
- Dependency-preserving decompositions

Given a schema R and a set of FDs $\mathcal{F}$,
decomposition of R is dependency preserving
if there is an equivalent set of FDs $\mathcal{F}'$,
none of which is interrelational in the decomposition.

- Next, how to obtain such decompositions?
  - BCNF → guaranteed to be a lossless join decomposition!

Boyce-Codd Normal Form (BCNF)

- A relation R is in BCNF iff whenever $(X \to Y) \in \mathcal{F}^+$ and $XY \subseteq R$, then either
  - $(X \to Y)$ is trivial (i.e., $Y \subseteq X$), or
  - X is a super key of R (i.e., $X \to R$)
    - That is, all non-trivial FDs follow from "key → other attributes"

- Example: $R = \{Sno, Sname, City, Pno, Pname, Price\}$

$\mathcal{F}$ includes:

$\boxed{\text{FD1: } Sno \to Sname, City}$ FD2: $Pno \to Pname$ FD3: $Sno, Pno \to Price$

- The schema is not in BCNF because, for example, *Sno* determines *Sname,City*, is non-trivial but is not a superkey of R

BCNF decomposition algorithm

- Find a BCNF violation
  - That is, a non-trivial FD $X \rightarrow Y$ in $\mathcal{F}^+$ of R where X is not a super key of R
    - Example: R = {*Sno*,*Sname*,*City*,*Pno*,*Pname*,*Price*}

$\mathcal{F}$ includes:

FD1: $\boxed{Sno \rightarrow Sname, City}$ FD2: $Pno \rightarrow Pname$ FD3: $Sno, Pno \rightarrow Price$

- Decompose R into R_1 and R_2, where
  - R_1 has attributes $X \cup Y$;
  - R_2 has attributes $X \cup Z$, where Z contains all attributes of R that are in neither X nor Y

R = {*Sno*,*Sname*,*City*,*Pno*,*Pname*,*Price*}

BCNF violation: $Sno \rightarrow Sname, City$

R2{*Sno*,*Pno*,*Pname*,*Price*} R1{*Sno*,*Sname*,*City*}

- Repeat (till all are in BCNF)

BCNF decomposition example

- $R = \{Sno, Sname, City, Pno, Pname, Price\}$

$\mathcal{F}$ includes:

FD1: $Sno \to Sname, City$ FD2: $Pno \to Pname$ FD3: $Sno, Pno \to Price$

$\{Sno, Sname, City, Pno, Pname, Price\}$

BCNF violation: $Sno \to Sname, City$

R2$\{Sno, Pno, Pname, Price\}$

$Pno \to Pname$ $Sno, Pno \to Price$

BCNF violation: $Pno \to Pname$

R2b$\{Sno, Pno, Price\}$

BCNF: $Sno, Pno \to Price$

R2a$\{Pno, Pname\}$

BCNF: $Pno \to Pname$

R1$\{Sno, Sname, City\}$

BCNF: $Sno \to Sname, City$

$\{SNo\}^{+} = \{Sno, Sname, City\}$
→ a superkey of R1

BCNF helps remove redundancy

Sno	Sname	City	Pno	Pname	Price
S1	Magna	K-W	P1	A	$25
S1	Magna	K-W	P2	B	$34
S1	Magna	K-W	P3	A	$20
S2	Box	London	…	…	…

BCNF violation: $Sno \rightarrow Sname, City$

Sno	Pno	Pname	Price
S1	P1	A	$25
S1	P2	B	$34
S1	P3	A	$20
S2	…	…	…

Sno	Sname	City
S1	Magna	K-W
S2	Box	London
..	…	…

Another example

$\mathcal{F}$ includes:

uid → *uname, twittered*
twitterid → *uid*
uid, gid → *fromDate*

UserJoinsGroup (*uid, uname, twitterid, gid, fromDate*)

Another example

$\mathcal{F}$ includes:

uid → *uname*, *twitterid*
twitterid → *uid*
uid, *gid* → *fromDate*

UserJoinsGroup (*uid*, *uname*, *twitterid*, *gid*, *fromDate*)

BCNF violation: *uid* → *uname*, *twitterid*

{uid}$^+$={uid, uname, twitterid}

User (*uid*, *uname*, *twitterid*)

uid → *uname*, *twitterid*
twitterid → *uid*

BCNF

{uid}$^+$={uid, uname, twitterid}
{twitterid}$^+$={uid, uname, twitterid}

Member (*uid*, *gid*, *fromDate*)

uid, *gid* → *fromDate*

BCNF

{uid,gid}$^+$={uid,gid ,fromeDate}

Alt. solution

$\mathcal{F}$ includes:

uid → *uname*, *twitterid*
twitterid → *uid*
uid, *gid* → *fromDate*

UserJoinsGroup (*uid*, *uname*, *twitterid*, *gid*, *fromDate*)

BCNF violation: *twitterid* → *uid*

UserId (*twitterid*, *uid*)

twitterid → *uid*

BCNF

No FDs in $\mathcal{F}$ violate BCNF here!
(as uid is missing in this relation)

UserJoinsGroup (*twitterid*, *uname*, *gid*, *fromDate*)

But we need to check all the FDs in $\mathcal{F}^+$!!

twitterid → *uname*
twitterid, *gid* → *fromDate*

BCNF violation: *twitterid* → *uname*

UserName (*twitterid*, *uname*)

BCNF

Member (*twitterid*, *gid*, *fromDate*)

BCNF

"Good" Schema Decomposition

- Lossless-join decompositions
- Dependency-preserving decompositions
- BCNF → guaranteed to be a lossless join decomposition!
    - Depend on the on the sequence of FDs for decomposition
    - Not necessarily dependency preserving

Example: consider R={A, B, C}

$\mathcal{F}$ includes: FD1: $AB \to C$ FD2: C $\to B$

BCNF violation: C $\to B$

{A, C} {C, B}

$AB \to C$ is interrelational and cannot be tested directly

“Good” Schema Decomposition

- Lossless-join decompositions
- Dependency-preserving decompositions
- BCNF → guaranteed to be a lossless join decomposition!
    - Depend on the on the sequence of FDs for decomposition
    - Not necessarily dependency preserving
- 3NF → both lossless join and dependency preserving

Third normal form (3NF)

- A relation R is in 3NF iff
  whenever $(X \rightarrow Y) \in \mathcal{F}^+$ and $XY \subseteq R$, then either
  - $(X \rightarrow Y)$ is trivial (i.e., $Y \subseteq X$), or
  - X is a super key of R (i.e., $X \rightarrow R$) or
  - **Each attribute in $\mathbf{Y} - \boldsymbol{X}$ is contained in a candidate key of $\boldsymbol{R}$**

  - Example: consider R={A, B, C}
    - Satisfies 3NF, but not BCNF

$\mathcal{F}$ includes: FD1: $AB \rightarrow C$ FD2: $C \rightarrow B$

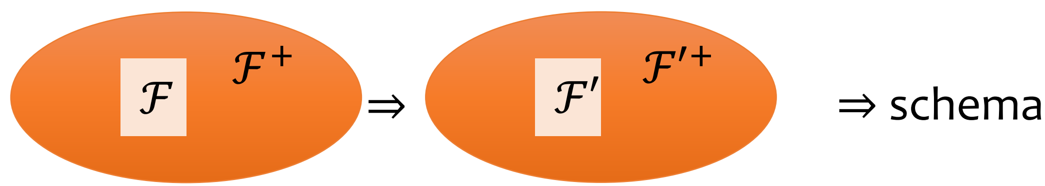
{B}-{C} = {B} is part of the key {AB}

- 3NF is looser than BCNF → Allows more redundancy

How to find a 3NF relation schemas?

- Lossless-join, dependency-preserving decomposition into 3NF relation schemas always exists.

  - Step 1: Finding the minimal cover of the FD set $\mathcal{F}$



  Given a set of FDs $\mathcal{F}$, we say $\mathcal{F}'$ is equivalent to $\mathcal{F}$ if their closures are the same: $\mathcal{F}^+ = \mathcal{F}'^+$.

  - Step 2: Decompose based on the minimal cover (i.e., $\mathcal{F}'$ is minimal).

Minimal cover

- A set of FDs $\mathcal{F}$ is minimal if

1. every right-hand side of a FD in $\mathcal{F}$ is a single attribute

- Example: $R = \{Sno, Sname, City, Pno, Pname, Price, PType\}$

$\mathcal{F}$: FD1: $Sno \rightarrow Sname, City$
FD2: $Pno \rightarrow Pname$
FD3: $Sno, Pno \rightarrow Price$
FD4: $Sno, Pname \rightarrow Price$
FD5: $Pno, Pname \rightarrow Ptype$

Fail condition 1

Minimal cover

No redundant attributes in X

- A set of FDs $\mathcal{F}$ is minimal if

1. every right-hand side of a FD in $\mathcal{F}$ is a single attribute
2. there does not exist $X \rightarrow A$, and Z a proper subset of X, such that the set $(\mathcal{F} - \{X \rightarrow A\}) \cup \{Z \rightarrow A\}$ is equivalent to F,
   English: no extraneous (redundant) attributes in the left-hand side of an FD in F

- Example: $R = \{Sno, Sname, City, Pno, Pname, Price, PType\}$

$\mathcal{F}$: FD1: $Sno \rightarrow Sname, City$
FD2: $Pno \rightarrow Pname$
FD3: $Sno, Pno \rightarrow Price$
FD4: $Sno, Pname \rightarrow Price$
FD5: $Pno, Pname \rightarrow Ptype$

Fail condition 2: replace by
FD5': $Pno \rightarrow Ptype$
$(\mathcal{F} - \{FD5\} + \{FD5'\})$ is equiv. to $\mathcal{F}$

$computeX^+(\{Pno\}, \{FD1, FD2, FD3, FD4, FD5\}) = \{\ldots, Ptype, \ldots\}$

[visit Lecture 9 for how to compute closure]

Minimal cover

- A set of FDs $\mathcal{F}$ is minimal if

1. Every right-hand side of a FD in $\mathcal{F}$ is a single attribute
2. There does not exist $X \rightarrow A$ and Z a proper subset of X, such th
$(\mathcal{F} - \{X \rightarrow A\}) \cup \{Z \rightarrow A\}$ is equivalent to F,
English: no extraneous (redundant) attributes in the left-hand side of a FD in F
3. There does not exist $X \rightarrow A$ in $\mathcal{F}$,such that $\mathcal{F} - \{X \rightarrow A\}$ equivalent to $\mathcal{F}$

No redundant FD in $\mathcal{F}$

Example: $R = \{Sno, Sname, City, Pno, Pname, Price, PType\}$

$\mathcal{F}$: FD1: $Sno \rightarrow Sname, City$
FD2: $Pno \rightarrow Pname$
FD3: $Sno, Pno \rightarrow Price$
FD4: $Sno, Pname \rightarrow Price$
FD5: $Pno, Pname \rightarrow Ptype$

Fail condition 3: FD2+FD4 can give FD3
$(\mathcal{F} - \{\text{FD3}\})$ is equiv. to $\mathcal{F}$

$computeX^{+}(\{\text{Sno, Pno}\}, \{\text{FD1,FD2,FD4,FD5}\})$
$= \{\ldots, \text{Price}, \ldots\}$

Finding minimal cover

- A minimal cover for $\mathcal{F}$ can be computed in 3 steps.
  1. Replace $X \to YZ$ with the pair $X \to Y$ and $X \to Z$
  2. Remove A from the left-hand side of $X \to B$ in $\mathcal{F}$ if B ∈ $computeX^+(X - \{A\}, \mathcal{F})$
  3. Remove $X \to A$ from $\mathcal{F}$ if $A \in computeX^+(X, \mathcal{F} - \{X \to A\})$
  - Note that each step must be repeated until it no longer succeeds in updating $\mathcal{F}$.

- Example: $R = \{$Sno,Sname,City,Pno,Pname,Price, $PType$ $\}$

$\mathcal{F}$: FD1: $Sno \to Sname, City$
FD2: $Pno \to Pname$
FD3: $Sno, Pno \to Price$
FD4: $Sno, Pname \to Price$
FD5: $Pno, Pname \to Ptype$

$Sno \to Sname,$
$Sno \to City$

Remove FD3

$Pno \to Ptype$

Computing 3NF decomposition

Efficient algorithm for computing a 3NF decomposition of R with FDs $\mathcal{F}$:

1. Initialize the decomposition with empty set
2. Find a minimal cover for $\mathcal{F}$, let it be $\mathcal{F}^*$
3. For every $(\mathrm{X} \rightarrow \mathrm{Y}) \in \mathcal{F}^*$, add a relation {XY} to the decomposition
4. If no relation contains a candidate key for R, then compute a candidate key K for R, and add relation {K} to the decomposition.

Example for 3NF decomposition

- $R = \{Sno, Sname, City, Pno, Pname, Price\}$

$\mathcal{F}$: FD1: $Sno \rightarrow Sname, City$
FD2: $Pno \rightarrow Pname$
FD3: $Sno, Pno \rightarrow Price$
FD4: $Sno, Pname \rightarrow Price$

- Minimal cover $\mathcal{F}^*$

Exercise

$\mathcal{F}^*$: FD1a: $Sno \rightarrow Sname$
FD1b: $Sno \rightarrow City$
FD2: $Pno \rightarrow Pname$
FD4: $Sno, Pname \rightarrow Price$

R1a(Sno, Sname)
R1b(Sno, City)
R2(Pno, Pname)
R4(Sno,Pname,Price)

R5(Sno,Pno)

Exercise

- Add relation for candidate key
- Optimization for this example: combine relations R1a and R1b

Summary

- Functional dependencies: provide clues towards elimination of (some) redundancies in a schema.
  - Closure of FDs (rules, e.g. Armstrong's axioms)
  - Compute attribute closure

- Schema decomposition
  - Lossless join decompositions
  - Dependency preserving decompositions
  - Normal forms based on FDs
    - BCNF → lossless join decompositions
    - 3rd NF → lossless join and dependency-preserving decompositions with more redundancy