Functional Dependency, Normal Forms for Relational DB, Relational DB Design

Tuesday, 5 May 2020 8:35 PM

A "good" database schema should not lead to update anomalies.

Functional dependencies

A function f from S_1 to S_2 has the property

$$if\ x, y\ \in S_1 \text{ and } x = y, \text{then}\ \ f(x) = f(y)\ .$$

A generalization of keys to avoid design flaws violating the above rule.

Let X and Y be sets of attributes in R.

X (*functionally*) determines Y , $X \rightarrow Y$, iff $t_1[X] = t_2[X]$ implies $t_1[Y\ ] = t_2[Y\ ]$.

i.e., f (t(X)) = t [Y]

We also say $X \rightarrow Y$ is a *functional* dependency, and that Y is *functionally* dependent on X.

X is called the *left side*, Y the *right side* of the dependency.

Armstrong's axioms (1974)

Notation: If X and Y are sets of attributes, we write XY for their union.

e.g. X = {A, B}, Y = {B, C}, XY = {A, B, C}

F1 (Reflexivity) If $X \supseteq Y$ then $X \rightarrow Y$.

F2 (Augmentation) $\{X \rightarrow Y\} \models XZ \rightarrow YZ$.

F3 (Transitivity) $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$.

F4 (Additivity) $\{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$.

F5 (Projectivity) $\{X \rightarrow YZ\} \models X \rightarrow Y$.

F6 (Pseudotransitivity) $\{X \rightarrow Y, YZ \rightarrow W\} \models XZ \rightarrow W$.

To compute closure

F = { A → B, BC → D, A → C }, compute $\{A\}^+$

1st scan of F:
$X^+ := \{A\}$
$X^+ := \{A, B\}$
$X^+ := \{A, B, C\}$

2nd scan of F:
$X^+ := \{A, B, C, D\}$

3rd scan of F: no change, therefore the algorithm terminates.

$\{A\}^+ := \{A, B, C, D\}$

To compute a Candidate key

Given a relational schema R and a set F of functional dependencies on R.

A key X of R must have the property that $X^+ = R$.

Algorithm to compute a candidate key

Step 1: Assign X a superkey in F.

Step 2: Iteratively remove attributes from X while retaining the property $X^+ = R$ till no reduction on X.

The remaining X is a key.

$R = \{A, B, C, D\}$ and $F = \{ A \rightarrow B, BC \rightarrow D, A \rightarrow C \}$

$X = \{A, B, C\}$ if the left hand side of F is a super key.

A cannot be removed because $\{BC\}^+ = \{B, C, D\} \neq R$

B can be removed because $\{AC\}^+ = \{A, B, C, D\} = R$
$\longrightarrow X = \{ A, C\}$

C can be further removed because $\{A\}^+ = \{A, B, C, D\}$
$\longrightarrow X = \{A\}$

Normal Forms for Relational Databases
• criteria for a good database design (i.e., to resolve update anomalies)
• formalized by functional (or other) dependencies

Non-first normal form

Fac_Dept	Prof	Course Preferences	
		Course	Course_Dept
Comp Sci	Smith	353	Comp Sci
		379	Comp Sci
		221	Decision Sci
	Clark	353	Comp Sci
		351	Comp Sci
		379	Comp Sci
		456	Mathematics
Chemistry	Turner	353	Comp Sci
		456	Mathematics
		272	Chemsitry
Mathematics	Jameison	353	Comp Sci
		379	Comp Sci
		221	Decision Sci
		456	Mathematics
		469	Mathematics

First Normal Form (1NF)
This simply means that attribute values are atomic, and is part of the definition of the relational model.
Atomic: multivalued attributes, composite attributes, and their combinations are disallowed.

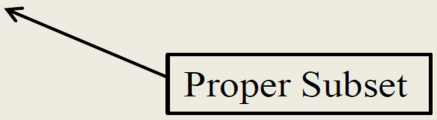
CRS_PREF			
Prof	Course	Fac_Dept	Crs_Dept
Smith	353	Comp Sci	Comp Sci
Smith	379	Comp Sci	Comp Sci
Smith	221	Comp Sci	Decision Sci
Clark	353	Comp Sci	Comp Sci
Clark	351	Comp Sci	Comp Sci
Clark	379	Comp Sci	Comp Sci
Clark	456	Comp Sci	Mathematics
Turner	353	Chemistry	Comp Sci
Turner	456	Chemistry	Mathematics
Turner	272	Chemistry	Chemistry
Jamieson	353	Mathematics	Comp Sci
Jamieson	379	Mathematics	Comp Sci
Jamieson	221	Mathematics	Decision Sci
Jamieson	456	Mathematics	Mathematics
Jamieson	469	Mathematics	Mathematics

Second Normal Form (2NF)

Second Normal Form (2NF)

A *prime* attribute is one that is part of a candidate key. Other attributes are *non-prime*.

Definition: In an FD $X \rightarrow Y$, Y is *fully functionally dependent* on X if there is no $Z \subset X$ such that $Z \rightarrow Y$. Otherwise Y is *partially* dependent on X.



Definition (*Second Normal Form*): A relation scheme is in second normal form (2NF) if all non-prime attributes are fully functionally dependent on the candidate keys.

A database scheme is in 2NF if all its relations are in 2NF.

Possible 2NF decomposition of the relation above is:

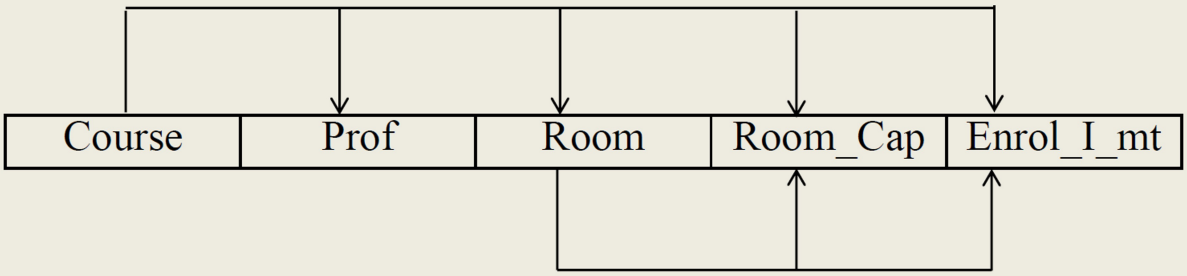
COURSE_PREF	
Prof	Course
Smith	353
Smith	379
Smith	221
Clark	353
Clark	351
Clark	379
Clark	456
Turner	353
Turner	456
Turner	272
Jamieson	353
Jamieson	379
Jamieson	221
Jamieson	456
Jamieson	469

COURSE	
Course	Dept
353	Comp Sci
379	Comp Sci
221	Decision Sci
351	Comp Sci
456	Mathematics
272	Chemistry
469	Mathematics

FACULTY	
Prof	Dept
Smith	Comp Sci
Clark	Comp Sci
Turner	Chemistry
Jamieson	Mathematics

2NF Drawback example:

TEACHES				
Course	Prof	Room	Room_Cap	Enrol_I_mt
353	Smith	A532	45	40
351	Smith	C320	100	60
355	Clark	H940	400	300
456	Turner	B278	50	45
459	Jamieson	D110	50	45



Third Normal Form (3NF) (cont)

Definition (Third Normal Form): A relation scheme is in *third normal form (3NF)* if for all non-trivial FD's of the form X→A that hold, either X is a superkey or A is a prime attribute.

Note: a FD X → Y is trivial iff Y is a subset of X.

Alternative definition: A relation scheme is in third normal form if every non-prime attribute is fully functionally dependent on the keys and not transitively dependent on any key.

A database scheme is in 3NF if all its relations are in 3NF.

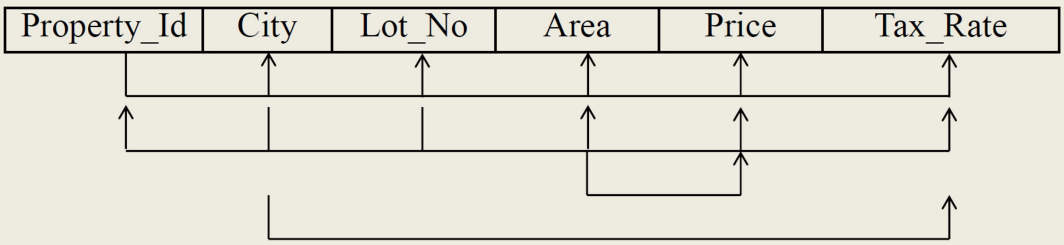
TEACHES can be decomposed into 3NF:

ROOM_DETAILS		
Room	Room_Cap	Enrol_I_mt
A532	45	40
C320	100	60
B278	50	45
D110	50	45
H940	400	300

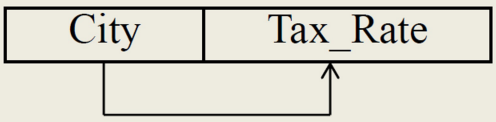
COURSE_DETAILS		
Course	Prof	Room
353	Smith	A532
351	Smith	C320
456	Turner	B278
459	Jamieson	D110
355	Clark	H940

Another example:

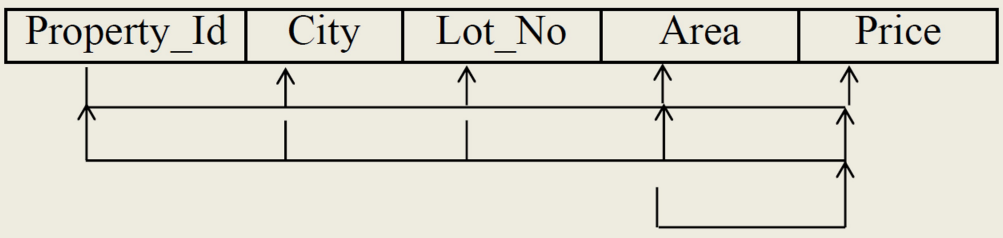
LOTS



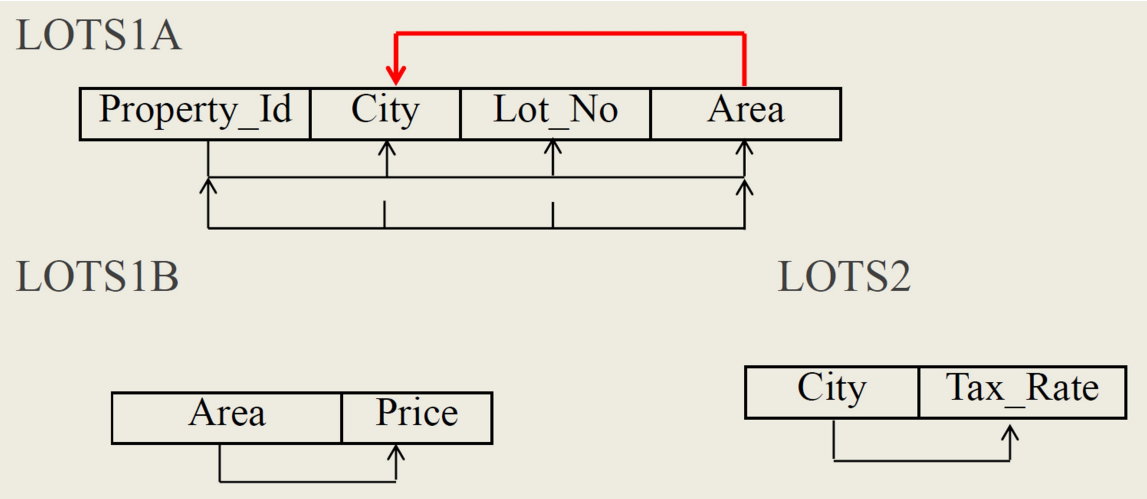
LOTS1



LOTS2



We could fix this too:



Boyce-Codd Normal Form (BCNF)

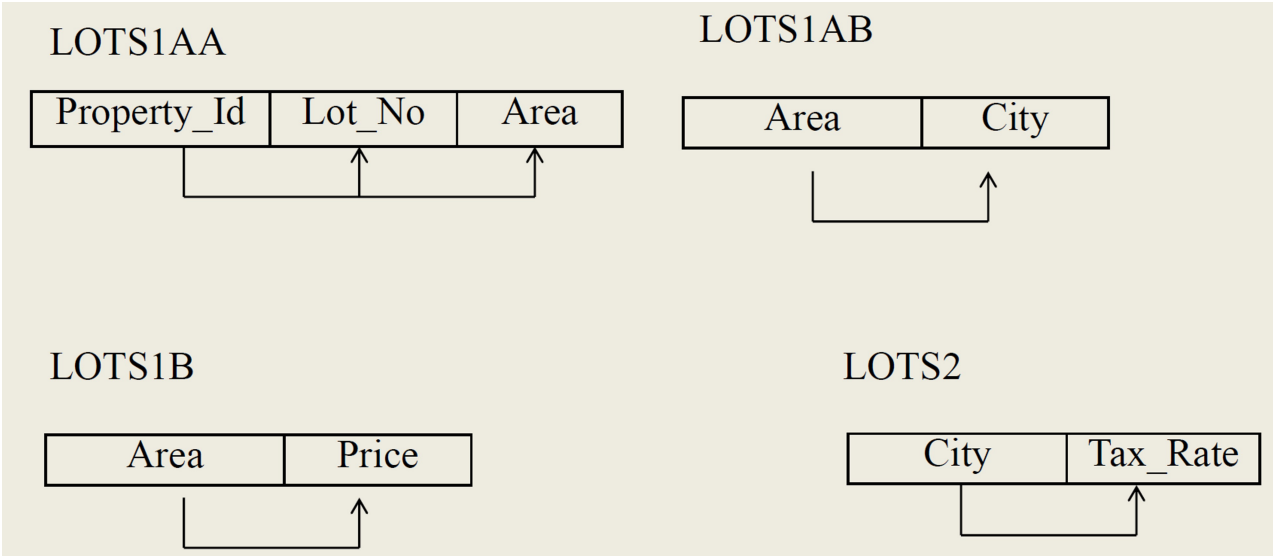
Definition (*Boyce-Codd Normal Form*):

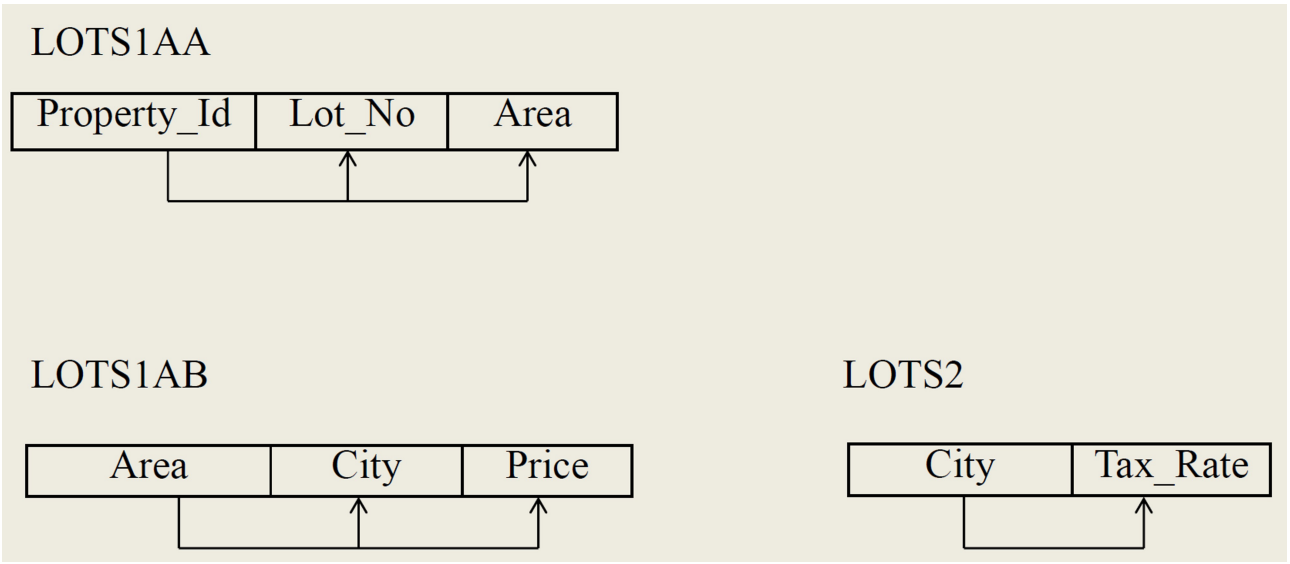
A relation scheme is in *Boyce-Codd* Normal Form (BCNF) if whenever

$X \rightarrow A$ holds and $X \rightarrow A$ is non-trivial, X is a superkey.

A database scheme is in BCNF if all its relations are in BCNF.

We can make our example into BCNF:





Relational Database Design

Anomalies can be removed from relation designs by decomposing them until they are in a normal form.

Several problems should be investigated regarding a decomposition.

A decomposition of a relation scheme, R, is a set of relation schemes $\{R_1, \ldots, R_n\}$ such that $R_i \subseteq R$ for each i, and $\bigcup_{i=1}^{n} R_i = R$

Note that in a decomposition $\{R_1, \ldots, R_n\}$, the intersect of each pair of R_i and R_j does not have to be empty.

Example: R = {A, B, C, D, E}, R_1 = { A, B}, R_2={A, C}, R_3= {C, D, E}

A naive decomposition: each relation has only attribute.

A good decomposition should have the following two properties.

Dependency Preserving

Examples

F = { A → BC, D → EG, M → A }, R = (A, B, C, D, E, G, M, A)

1) Given R_1= (A, B, C, M) and R_2 = (C, D, E, G),

F_1 = { A → BC, M → A}, F_2 = {D → EG}

F = F_1 U F_2. thus, dependency preserving

2) Suppose that F' = F U {M → D}. R_1 and R_2 remain the same.

Thus, F_1 and F_2 remain the same.

We need to verify if M→D is inferred by F_1 U F_2.

Since $M^+ |_{F1 U F2}$ = {M, A, B, C}, M→D is not inferred by F_1 U F_2.

Thus, R_1 and R_2 are not dependency preserving regarding F'.

3) F'' = {A → BC, D → EG, M → A , M→C, C→ D, M→ D}

F_1 = {A → BC, M→ A, M→ C}, F_2 = {D→ EG, C→ D}

It can be verified that M→ D is inferred by F_1 and F_2.

Thus, $F''^+ = (F_1 \cup F_2)^+$

Hence, R_1 and R_2 are dependency preserving regarding F''.

Lossless Join Decomposition

A second necessary property for decomposition:

A decomposition $\{R_1, \ldots, R_n\}$ of R is a *lossless join* decomposition with respect to a set F of FD's if for every relation instance r that satisfies F:

$$r = \pi R_1(r) \bowtie \cdots \bowtie \pi R_n(r).$$

If $r \subset \pi R_1(r) \bowtie \cdots \bowtie \pi R_n(r)$, the decomposition is *lossy*.

Example

STUDENT_ADVISOR

Name	Department	Advisor
Jones	Comp Sci	Smith
Ng	Chemistry	Turner
Martin	Physics	Bosky
Dulles	Decision Sci	Hall
Duke	Mathematics	James
James	Comp Sci	Clark
Evan	Comp Sci	Smith
Baxter	English	Bronte

STUDENT_DEPARTMENT

Name	Department
Jones	Comp Sci
Ng	Chemistry
Martin	Physics
Duke	Mathematics
Dulles	Decision Sci
James	Comp Sci
Evan	Comp Sci
Baxter	English

DEPARTMENT_ADVISOR

Department	Advisor
Comp Sci	Smith
Chemistry	Turner
Physics	Bosky
Decision Sci	Hall
Mathematics	James
Comp Sci	Clark
English	Bronte

If we join these decomposed relations we get:

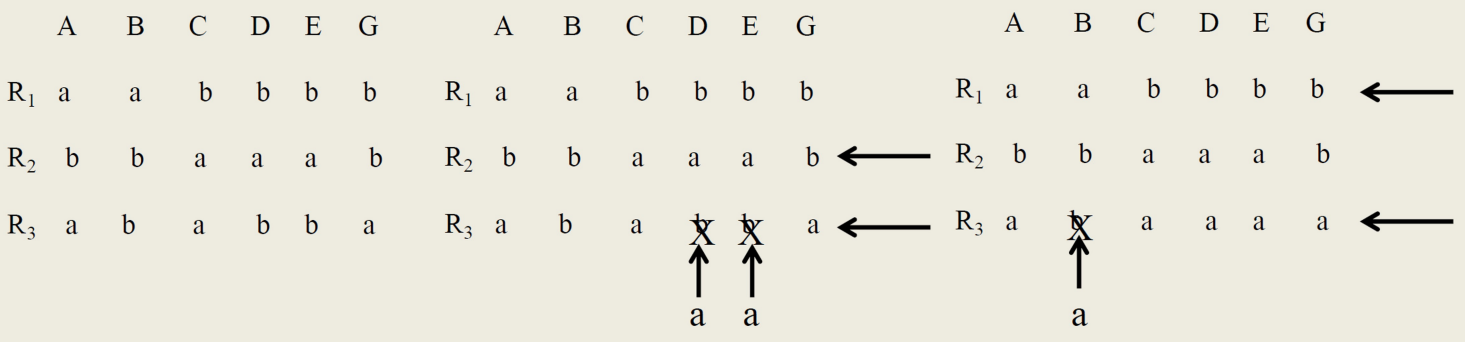
Name	Department	Advisor
Jones	Comp Sci	Smith
Jones	Comp Sci	Clark*
Ng	Chemistry	Turner
Martin	Physics	Bosky
Dulles	Decision Sci	Hall
Duke	Mathematics	James
James	Comp Sci	Smith*
James	Comp Sci	Clark
Evan	Comp Sci	Smith
Evan	Comp Sci	Clark*
Baxter	English	Bronte

Testing for the lossless join property

Example 4: $R = (A,B,C,D,E,G)$,

$F = \{AB \rightarrow G,\ C \rightarrow DE,\ A \rightarrow B\}$.

Let $R_1 = (A,B)$, $R_2 = (C, D, E)$ and $R_3 = (A,C,G)$.



Lossless decomposition into BCNF

F = {A→B,A→C, A→D, C→E, E→D, C→G},

R1 = (C, D, E, G), R2 = (A, B, C, D)

R11 = (C, E, G), R12 = (E, D) due E→D

R21 = (A, B, C), R22 = (C, D) because of C → D

Example: (From Desai 6.31)

Find a BCNF decomposition of the relation scheme below:

SHIPPING(*Ship* , *Capacity* , *Date* , *Cargo* , *Value*)

F consists of:

Ship→ *Capacity*

{*Ship* , *Date*}→ *Cargo*

{*Cargo* , *Capacity*}→ *Value*

From *Ship*→ *Capacity*, we decompose *SHIPPING* into

R_1(*Ship* , *Date* , *Cargo* , *Value*)

Key: {*Ship*,*Date*}

A nontrivial FD in F^+ violates BCNF:

{*Ship* , *Cargo*} → Value

Ship → Capacity
{*Ship* , *Date*}→ *Cargo*
{*Cargo* , *Capacity*}→ *Value*

and

R_2(*Ship* , *Capacity*)

Key: {*Ship*}

Only one nontrivial FD in F^+: *Ship* → *Capacity*

R_1 is not in BCNF so we must decompose it further into

$R_{11}(Ship, Date, Cargo)$

Key: $\{Ship, Date\}$

Ship → Capacity
$\{Ship, Date\} \rightarrow Cargo$
$\{Cargo, Capacity\} \rightarrow Value$

Only one nontrivial FD in F^+ with single attribute on the right side: $\{Ship, Date\} \rightarrow$Cargo

and

$R_{12}(Ship, Cargo, Value)$

Key: $\{Ship, Cargo\}$

Only one nontrivial FD in F^+ with single attribute on the right side: $\{Ship, Cargo\} \rightarrow Value$

This is in BCNF and the decomposition is lossless but not dependency preserving (the FD $\{Capacity, Cargo\} \rightarrow Value$) has been lost.

Or we could have chosen $\{Cargo, Capacity\} \rightarrow Value$, which would give us:

$R_1(Ship, Capacity, Date, Cargo)$

Key: $\{Ship, Date\}$

A nontrivial FD in F^+ violates BCNF:

$Ship \rightarrow Capacity$

Ship → Capacity
$\{Ship, Date\} \rightarrow Cargo$
$\{Cargo, Capacity\} \rightarrow Value$

and

$R_2(Cargo, Capacity, Value)$

Key: $\{Cargo, Capacity\}$

Only one nontrivial FD in F^+ with single attribute on the right side: $\{Cargo, Capacity\} \rightarrow Value$

and then from $Ship \rightarrow Capacity$,

Ship → Capacity
$\{Ship, Date\} \rightarrow Cargo$
$\{Cargo, Capacity\} \rightarrow Value$

$R_{11}(Ship, Date, Cargo)$

Key: $\{Ship, Date\}$

Only one nontrivial FD in F^+ with single attribute on the right side: $\{Ship, Date\} \rightarrow Cargo$

And

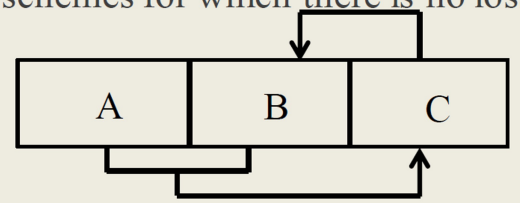
$R_{12}(Ship, Capacity)$

Key: $\{Ship\}$

Only one nontrivial FD in F^+: $Ship \rightarrow Capacity$

This is in BCNF and the decomposition is both lossless and dependency preserving.

However, there are relation schemes for which there is no lossless, dependency preserving decomposition into BCNF.



Lossless and dependency-preserving decomposition into 3NF

A lossless and dependency-preserving decomposition into 3NF is always possible.

More definitions regarding FD's are needed.

A set F of FD's is minimal if

1. Every FD $X \rightarrow Y$ in F is simple: Y consists of a single attribute,
2. Every FD $X \rightarrow A$ in F is *left-reduced*: there is no proper subset $Y \subset X$ such that $X \rightarrow A$ can be replaced with $Y \rightarrow A$.

   that is, there is no $Y \subset X$ such that → Iff $F \models Y \rightarrow A$

   $$((F - \{X \rightarrow A\}) \cup \{Y \rightarrow A\})^+ = F^+$$

3. No FD in F can be removed; that is, there is no FD $X \rightarrow A$ in F such that → Iff $X \rightarrow A$ is inferred From $F - \{X \rightarrow A\}$

   $$(F - \{X \rightarrow A\})^+ = F^+.$$

Example:

R = (A, B, C, D, E, G)

F = {A → BCD, B → CDE, AC → E}

Step 1: F' = {A → B, A→C, A → D, B→ C, B→ D, B→ E, AC → E}

Step 2: AC → E

C^+ = {C}; thus C → E is not inferred by F'.

Hence, AC → E cannot be replaced by A → E.

A^+ = {A, B, C, D, E}; thus, A→ E is inferred by F'.

Hence, AC→ E can be replaced by A → E.

F'' = {A→ B, A→C, A→ D, A→ E, B→ C, B→D, B→ E}

Step 3: $A+|_{F''-\{A \to B\}}$= {A, C, D, E}; thus A→B is not inferred by F'' –{A→B}.

That is, A→B is not redundant.

$A+|_{F''-\{A \to C\}}$= {A, B, C, D, E}; thus, A→ C is redundant.

Thus, we can remove A→C from F'' to obtain F'''.

Iteratively, we can A→D and A→E but not the others.

Thus, F_{min}={A→B, B→C, B→D, B→E}.

3NF decomposition algorithm

Algorithm 3NF decomposition

1. Find a minimum cover F' of F.
2. For each left side X that appears in F', do:

   create a relation schema $X \cup A_1 \cup A_2 \ldots \cup A_m$ where $X \to \{A_1\}, \ldots, X \to \{A_m\}$ are all the

   dependencies in F' with X as left side.
3. if none of the relation schemas contains a key of R,

   create one more relation schema that contains attributes that form a key for R.

Example:

R = (A, B, C, D, E, G)

F_{min}={A→B, B→C, B→D, B→E}.

Candidate key: (A, G)

R_1 = (A, B), R_2 = (B, C, D, E)

R_3 = (A, G)

Example: (From Desai 6.31)

Beginning again with the *SHIPPING* relation. The functional dependencies already form a canonical cover.

- From $Ship \rightarrow Capacity$, derive $R_1(\underline{Ship}, Capacity)$,
- From $\{Ship, Date\} \rightarrow Cargo$, derive

  $R_2(\underline{Ship, Date}, Cargo)$,
- From $\{Capacity, Cargo\} \rightarrow Value$, derive

  $R_3(\underline{Capacity, Cargo}, Value)$.
- There are no attributes not yet included and the original key $\{Ship, Date\}$ is included in R_2.

Another Example: Apply the algorithm to the LOTS example given earlier.

A minimal cover is

$$\{ Property\_Id \rightarrow Lot\_No,$$
$$Property\_Id \rightarrow Area,\ \{City, Lot\_No\} \rightarrow Property\_Id,$$
$$Area \rightarrow Price,\ Area \rightarrow City,\ City \rightarrow Tax\_Rate \}.$$

This gives the decomposition:

R_1 ($\underline{Property\_Id}$, *Lot_No*, *Area*)

R_2 ($\underline{City, Lot\_No}$, *Property_Id*)

R_3 ($\underline{Area}$, *Price*, *City*)

R_4 ($\underline{City}$, *Tax_Rate*)