Chapter 8: Relational Database Design

Database System Concepts, 6th Ed.

Features of Good Relational Design

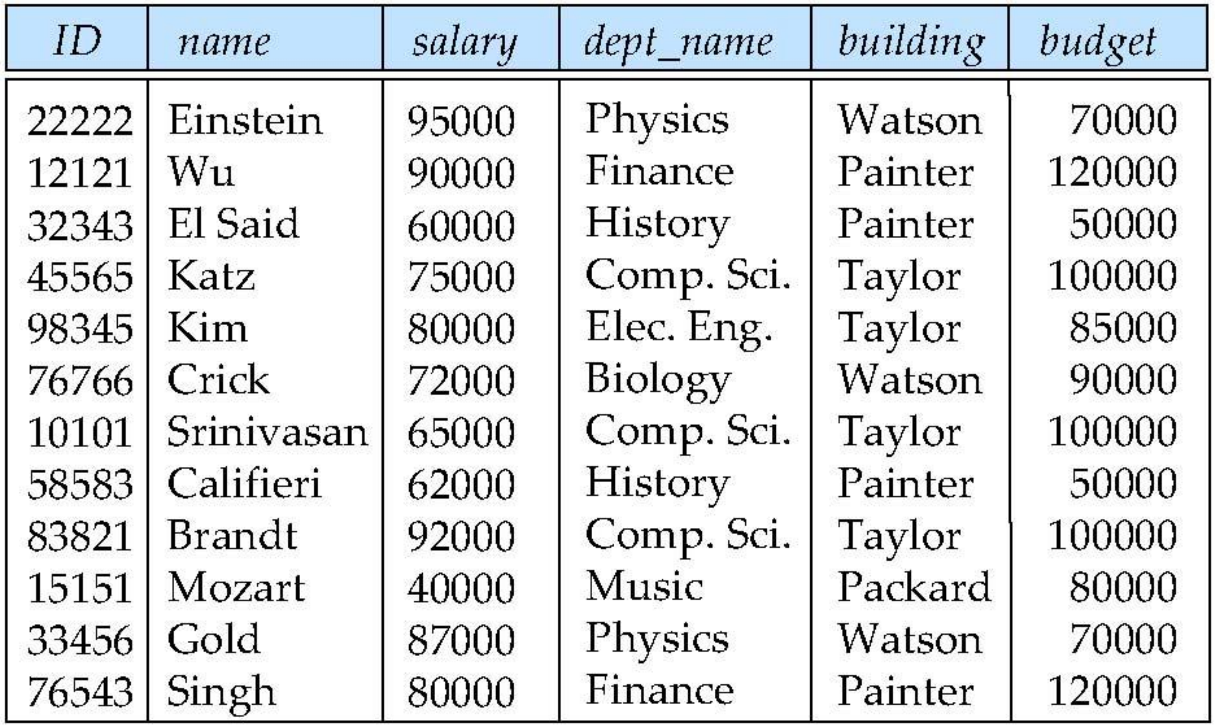
- In general, the goal of relational database design is to generate a set of relation schemas that allows us to store information without unnecessary redundancy, yet also allows us to retrieve information easily. This is accomplished by designing schemas that are in an appropriate normal form.
- It is possible to generate a set of relation schemas directly from the E-R design. Obviously, the goodness (or badness) of the resulting set of schemas depends on how good the E-R design was in the first place.

Schema For The University Database

- classroom(building, room number, capacity)
- department(dept_name, building, budget)
- course(course_id, title, dept_name, credits)
- instructor (ID, name, dept_name, salary)
- section(course_id, sec_id, semester, year, building, room_number, time_slot_id)
- teaches(ID, course_id, sec_id, semester, year)
- student(ID, name, dept_name, tot_cred)
- takes(ID, course_id, sec_id, semester, year, grade)
- advisor(s_ID, i_ID)
- time slot(time_slot_id, day, start_time, end_time)
- prereq(course_id, prereq_id)

Design Alternative: Larger Schemas

- Suppose we combine *instructor* and *department* into *inst_dept*
  - *(No connection to relationship set inst_dept)*
- Result is possible repetition of information



ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

A Combined Schema Without Repetition

- Consider combining relations
  - *sec_class(sec_id, building, room_number)* and
  - *section(course_id, sec_id, semester, year)*

- into one relation
  - *section(course_id, sec_id, semester, year, building, room_number)*

- No repetition in this case

What About Smaller Schemas?

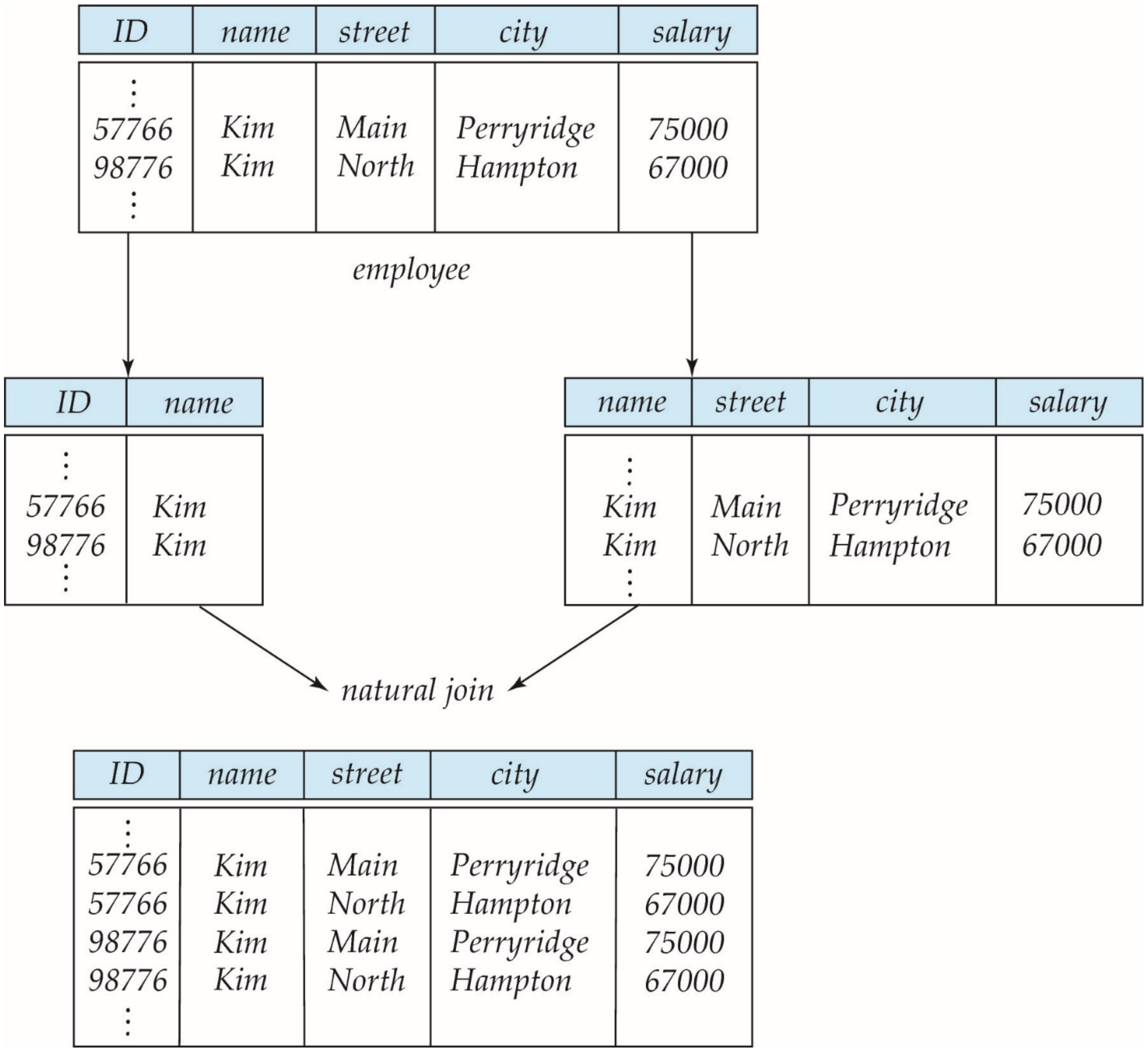
- Suppose we had started with *inst_dept.* How would we know to split up (**decompose**) it into *instructor* and *department*?
- Write a rule "if there were a schema (*dept_name, building, budget*), then *dept_name* would be a candidate key"
- Denote as a **functional dependency**:

  dept_name → *building, budget*
- In *inst_dept*, because *dept_name* is not a candidate key, the building and budget of a department may have to be repeated.
  - This indicates the need to decompose *inst_dept*
- Not all decompositions are good. Suppose we decompose *employee(ID, name, street, city, salary)* into

  employee1 (*ID*, *name*)

  employee2 (*name*, *street, city, salary*)
- The next slide shows how we lose information -- we cannot reconstruct the original *employee* relation -- and so, this is a **lossy decomposition**.

A Lossy Decomposition



ID	name	street	city	salary
⋮				
57766	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000
⋮				

ID	name
⋮	
57766	Kim
98776	Kim
⋮	

name	street	city	salary
⋮			
Kim	Main	Perryridge	75000
Kim	North	Hampton	67000
⋮			

ID	name	street	city	salary
⋮				
57766	Kim	Main	Perryridge	75000
57766	Kim	North	Hampton	67000
98776	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000
⋮				

Example of Lossless-Join Decomposition

- **Lossless join decomposition**
- Decomposition of $R = (A, B, C)$
  $R_1 = (A, B)$ $R_2 = (B, C)$

A	B	C
α	1	A
β	2	B

r

A	B
α	*1*
β	*2*

$\prod_{A,B}(r)$

B	C
1	A
2	B

$\prod_{B,C}(r)$

$\prod_{A}(r) \bowtie \prod_{B}(r)$

A	B	C
α	1	A
β	2	B

Data Normalization

- Primarily a tool to validate and improve a logical design so that it satisfies certain constraints that avoid unnecessary duplication of data
- The process of decomposing relations with anomalies to produce smaller, well-structured relations

- **Results of Normalization**
- Removes the following modification anomalies (integrity errors) with the database
  - Insertion
  - Deletion
  - Update

ANOMALIES

- Insertion
  - inserting one fact in the database requires knowledge of other facts unrelated to the fact being inserted
- Deletion
  - Deleting one fact from the database causes loss of other unrelated data from the database
- Update
  - Updating the values of one fact requires multiple changes to the database

ANOMALIES EXAMPLES
TABLE: COURSE

COURSE#	SECTION#	C_NAME
CIS564	072	Database Design
CIS564	073	Database Design
CIS570	072	Oracle Forms
CIS564	074	Database Design

ANOMALIES EXAMPLES

Insertion:

Suppose our university has approved a new course called CIS563: SQL & PL/SQL.

COURSE#	SECTION#	C_NAME
CIS564	072	Database Design
CIS564	073	Database Design
CIS570	072	Oracle Forms
CIS564	074	Database Design

ANOMALIES EXAMPLES

Deletion:

Suppose not enough students enrolled for the course CIS570 which had only one section 072. So, the school decided to drop this section and delete the section# 072 for CIS570 from the table COURSE. But then, what other relevant info also got deleted in the process?

COURSE#	SECTION#	C_NAME
CIS564	072	Database Design
CIS564	073	Database Design
CIS570	072	Oracle Forms
CIS564	074	Database Design

ANOMALIES EXAMPLES

Update:

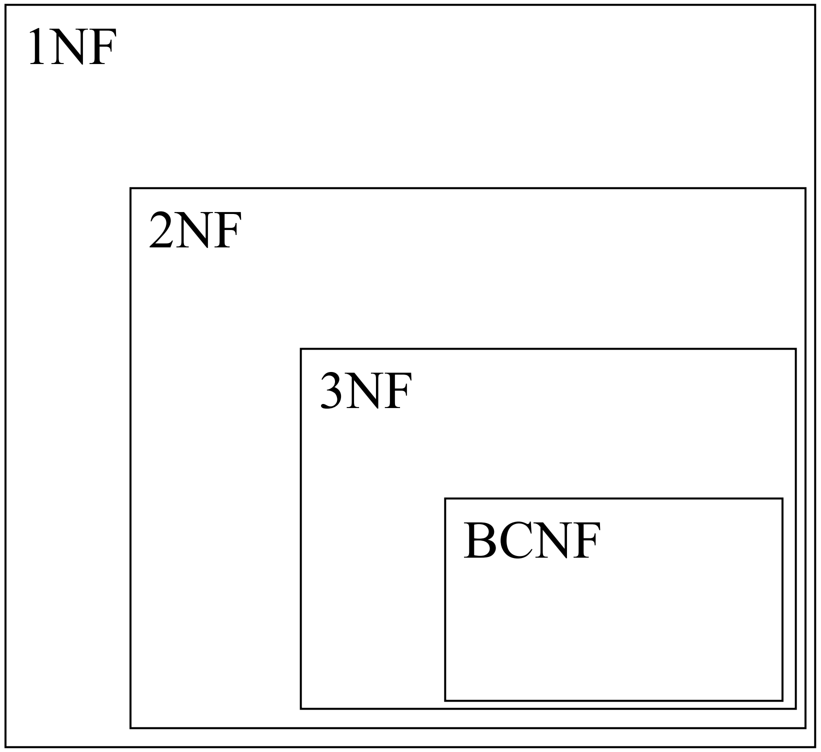
Suppose the course name (C_Name) for CIS 564 got changed to Database Management. How many times do you have to make this change in the COURSE table in its current form?

<u>COURSE#</u>	<u>SECTION#</u>	C_NAME
CIS564	072	Database Design
CIS564	073	Database Design
CIS570	072	Oracle Forms
CIS564	074	Database Design

ANOMALIES

- So, a table (relation) is a stable ('good') table only if it is free from any of these anomalies at any point in time.
- You have to ensure that each and every table in a database is always free from these modification anomalies. And, how do you ensure that?
- 'Normalization' theory helps.

Normalization



a relation in BCNF, is also in 3NF

a relation in 3NF is also in 2NF

a relation in 2NF is also in 1NF

Normalization

We consider a relation in BCNF to be fully normalized.

The benefit of higher normal forms is that update semantics for the affected data are simplified. This means that applications required to maintain the database are simpler.

A design that has a lower normal form than another design has more redundancy. Uncontrolled redundancy can lead to data integrity problems.

First we introduce the concept of ***functional dependency***

Functional Dependencies

Functional Dependencies

We say an attribute, B, has a *functional dependency* on another attribute, A, if for any two records, which have the same value for A, then the values for B in these two records must be the same. We illustrate this as:

A □ B

Example: Suppose we keep track of employee email addresses, and we only track one email address for each employee. Suppose each employee is identified by their unique employee number. We say there is a functional dependency of email address on employee number:

employee number □ email address

Functional Dependencies

EmpNum	EmpEmail	EmpFname	EmpLname
123	jdoe@abc.com	John	Doe
456	psmith@abc.com	Peter	Smith
555	alee1@abc.com	Alan	Lee
633	pdoe@abc.com	Peter	Doe
787	alee2@abc.com	Alan	Lee

If EmpNum is the PK then the FDs:

EmpNum □ EmpEmail
EmpNum □ EmpFname
EmpNum □ EmpLname

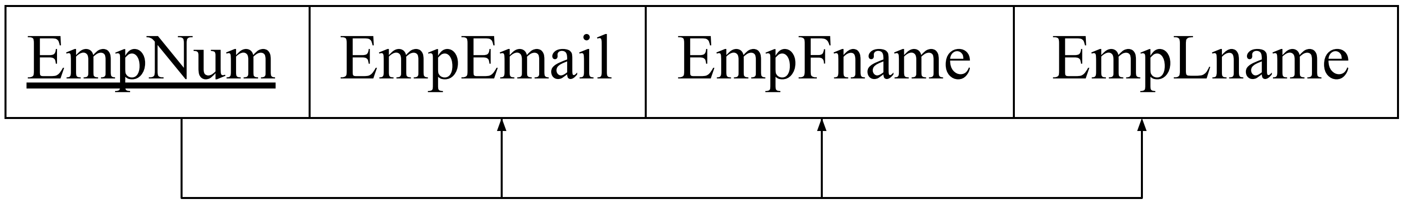
must exist.

Functional Dependencies

EmpNum □ EmpEmail
EmpNum □ EmpFname
EmpNum □ EmpLname

3 different ways you might see FDs depicted

$$\begin{tikzcd} & \text{EmpEmail} \\ \text{EmpNum} \arrow[ur] \arrow[r] \arrow[dr] & \text{EmpFname} \\ & \text{EmpLname} \end{tikzcd}$$



Determinant

Functional Dependency

EmpNum □ EmpEmail

Attribute on the LHS is known as the ***determinant***

- EmpNum is a determinant of EmpEmail

Transitive dependency

Consider attributes A, B, and C, and where

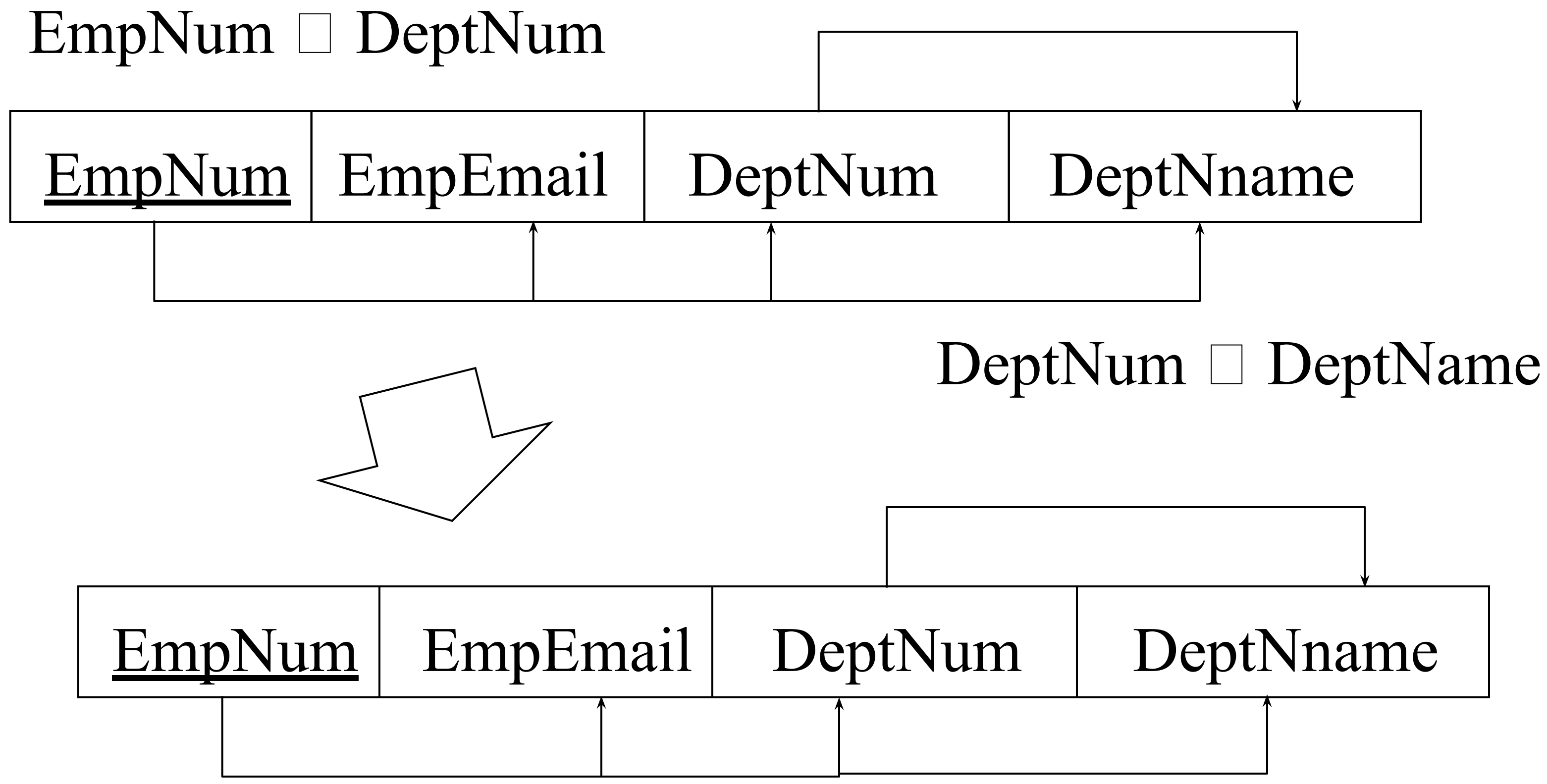
A □ B and B □ C.

Functional dependencies are transitive, which means that we also have the functional dependency

A □ C

We say that C is transitively dependent on A through B.

Transitive dependency

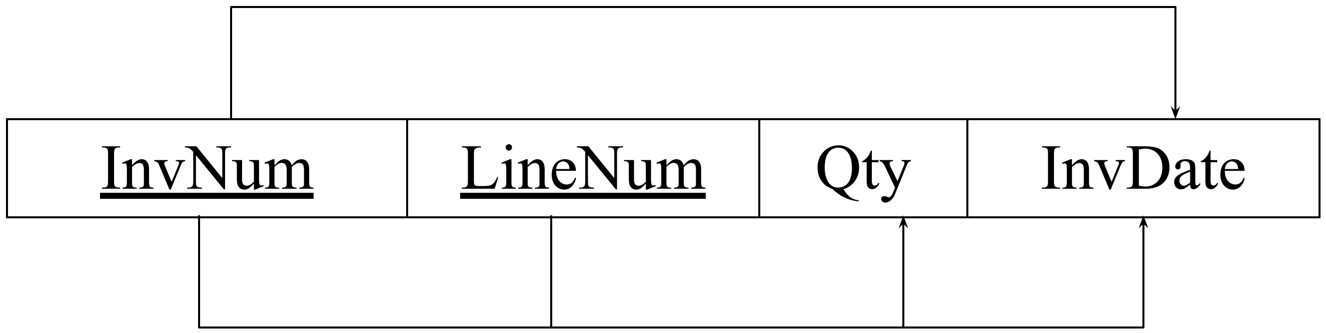


DeptName is *transitively dependent* on EmpNum via DeptNum

EmpNum □ DeptName

Partial dependency

A **partial dependency** exists when an attribute B is functionally dependent on an attribute A, and A is a component of a multipart candidate key.



Candidate keys: {InvNum, LineNum} InvDate is *partially dependent* on {InvNum, LineNum} as InvNum is a determinant of InvDate and InvNum is part of a candidate key

Closure of a Set of Functional Dependencies

- Given a set F of functional dependencies, there are certain other functional dependencies that are logically implied by F.
  - For example: If $A \rightarrow B$ and $B \rightarrow C$, then we can infer that $A \rightarrow C$
- The set of **all** functional dependencies logically implied by F is the **closure** of F. We denote the *closure* of F by **F^+**. $\underline{F^+ \text{ is a superset of } F.}$
- Candidate Key Example:
  - *FD1*: $A \rightarrow B,\ B \rightarrow C,\ C \rightarrow D$
  - *FD2*: $A \rightarrow B,\ B \rightarrow C,\ C \rightarrow D,\ D \rightarrow A$
  - *FD3*: $A \rightarrow B,\ BC \rightarrow D,\ E \rightarrow C,\ D \rightarrow A$
- Prime Attribute VS Non-Prime Attribute

Closure of a Set of Functional Dependencies

- Given a set F set of functional dependencies, there are certain other functional dependencies that are logically implied by F.
  - For e.g.: If $A \rightarrow B$ and $B \rightarrow C$, then we can infer that $A \rightarrow$ C
- The set of **all** functional dependencies logically implied by F is the **closure** of F.
- We denote the *closure* of F by $\boldsymbol{F^+}$.

Closure of a Set of Functional Dependencies

- We can find $F^{+,}$ the closure of F, by repeatedly applying **Armstrong's Axioms:**
  - if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ (**reflexivity**)
  - if $\alpha \rightarrow \beta$, then $\gamma\, \alpha \rightarrow \gamma\, \beta$ (**augmentation**)
  - if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$ (**transitivity**)
- These rules are
  - **sound** (generate only functional dependencies that actually hold), and
  - **complete** (generate all functional dependencies that hold).

Example

- $R = (A, B, C, G, H, I)$
  $F = \{ A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- some members of F^+
  - $A \rightarrow H$
    - by transitivity from $A \rightarrow B$ *and* $B \rightarrow H$
  - $AG \rightarrow I$
    - by augmenting $A \rightarrow C$ with G, to get $AG \rightarrow CG$
      and then transitivity with $CG \rightarrow I$
  - $CG \rightarrow HI$
    - by augmenting $CG \rightarrow I$ to infer $CG \rightarrow$ CGI,
      and augmenting of $CG \rightarrow H$ to infer $CGI \rightarrow HI$,
      and then transitivity

Closure of Functional Dependencies (Cont.)

- Additional rules:
  - If $\alpha \rightarrow \beta$ holds *and* $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta\gamma$ holds (**union**)
  - If $\alpha \rightarrow \beta\gamma$ holds, then $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds (**decomposition**)
  - If $\alpha \rightarrow \beta$ holds *and* $\gamma\beta \rightarrow \delta$ holds, then $\alpha\gamma \rightarrow \delta$ holds (**pseudotransitivity**)

  The above rules can be inferred from Armstrong's axioms.

Example of Attribute Set Closure

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- $(AG)^+$
  1. $result = AG$
  2. $result = ABCG \quad (A \rightarrow C$ and $A \rightarrow B)$
  3. $result = ABCGH \quad (CG \rightarrow H$ and $CG \subseteq AGBC)$
  4. $result = ABCGHI \quad (CG \rightarrow I$ and $CG \subseteq AGBCH)$
- Is AG a candidate key?
  1. Is AG a super key?
     1. Does $AG \rightarrow R$? == Is $(AG)^+ \supseteq R$
  2. Is any subset of AG a superkey?
     1. Does $A \rightarrow R$? == Is $(A)^+ \supseteq R$
     2. Does $G \rightarrow R$? == Is $(G)^+ \supseteq R$

Closure of a Set of Functional Dependencies

R(ABCDEF)
FD{ $C \rightarrow F$, $E \rightarrow A$, E$C \rightarrow D$, $A \rightarrow B$}

CK={EC}
PA={E,C}
NPA={A,B,D,F}

First Normal Form

First Normal Form

We say a relation is in **1NF** if all values stored in the relation are single-valued and atomic.

1NF places restrictions on the structure of relations. Values must be simple.

First Normal Form

The following in **not** in 1NF

EmpNu	EmpPhon	EmpDegrees
m 123	e233-9876	
333	233-1231	BA, BSc,
679	233-1231	PhD BSc,

MSc

EmpDegrees is a multi-valued field:

employee 679 has two degrees: *BSc* and *MSc*

employee 333 has three degrees: *BA, BSc, PhD*

First Normal Form

- To obtain 1NF relations we must, replace the above with single value multiple row

EmpNum	EmpPhone	EmpDegree
123	233-9876	
333	233-1231	BA
333	233-1231	BSc
333	233-1231	PhD
679	233-1231	BSc
679	233-1231	MSc

First Normal Form

- To obtain 1NF relations we must, replace the above with single value multiple Column

EmpNum	EmpPhone	EmpDegree 1	EmpDegree 2	EmpDegree 3
123	**233-9876**			
333	**233-1231**	BA	BSc	PhD
679	**233-1231**	BSc	MSc	

First Normal Form

To obtain 1NF relations we must, replace the above with two relations

Employee

EmpNum	**EmpPhone**
123	233-9876
333	233-1231
679	233-1231

EmployeeDegree

EmpNum	**EmpDegree**
333	BA
333	BSc
333	PhD
679	BSc
679	MSc

An outer join between Employee and EmployeeDegree will produce the information we saw before

Second Normal Form

Second Normal Form

A relation is in **2NF** if it is in 1NF, and **every non-key attribute is fully dependent on each candidate key.** (That is, we don't have any partial functional dependency.)

- 2NF (and 3NF) both involve the concepts of key and non-key attributes.

- A *key attribute* is any attribute that is part of a key; any attribute that is not a key attribute, is a *non-key attribute*.

SECOND NORMAL FORM

Table purchase detail

Customer_id	Store_id	Location
1	1	Patna
1	3	Noida
2	1	Patna
3	2	Delhi
4	3	Noida

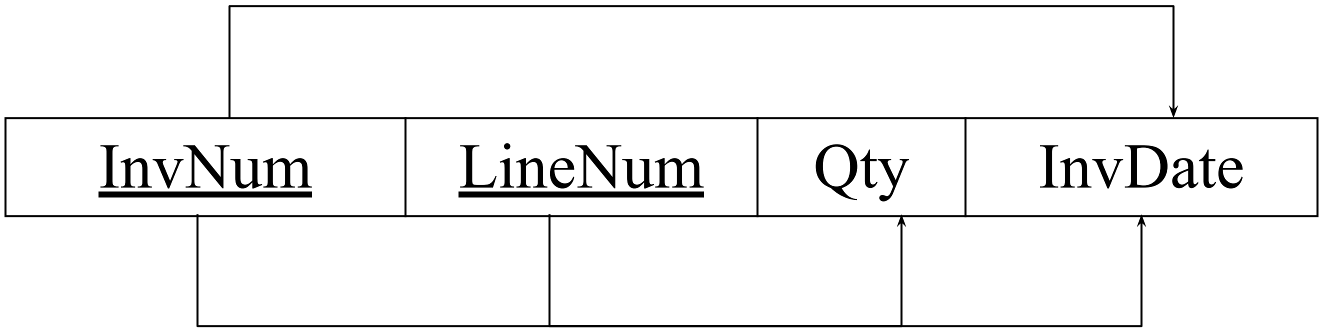
- This table has a composite primary key i.e. customer id, store id. The non key attribute is location. In this case location depends on store id, which is part of the primary key.

After decomposing it into second normal form it looks like:

Table Purchase	
Customer_id	Store_id
1	1
1	3
2	1
3	2
4	3

Table Store	
Store_id	Location
1	Patna
2	Delhi
3	Noida

Second Normal Form



We correct the situation by decomposing the original relation into two 2NF relations.

InvNum	LineNum	Qty

InvNum	InvDate

Second Normal Form

R(ABCDEF)

$$\mathrm{FD}\{\ \underset{\mathrm{PD}}{C \rightarrow F},\ \underset{\mathrm{PD}}{E \rightarrow A},\ EC \rightarrow D,\ A \rightarrow B\}$$

- A relation is in **2NF** if it is in 1NF, and every non-prime attribute is fully dependent on each candidate key (If L.H.S. is proper subset of candidate key and R.H.S. is non prime attribute then there is a partial dependency).

CK={EC}
PA={E,C}
NPA={A,B,D,F}

R(ABCDEF)
After Decomposition:
Step1: R1(ABCDE), R2(CF)
Step2: R11(BCDE), R12(AE), R2(CF)

Third Normal Form (3NF)

A table design is said to be in 3NF if both the following conditions hold:

- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should be removed.

An attribute that is not part of any candidate key is known as non-prime attribute.

In other words 3NF can be explained like this: A table is in 3NF if it is in 2NF and for each functional dependency X-> Y at least one of the following conditions hold:

- ✓ X is a super key of table
- ✓ Y is a prime attribute of table

An attribute that is a part of one of the candidate keys is known as prime attribute.

THIRD NORMAL FORM

Table Book Details			
Bood_id	Genre_id	Genre type	Price
1	1	Fiction	100
2	2	Sports	110
3	1	Fiction	120
4	3	Travel	130
5	2	sports	140

- In the table, book_id determines genre_id and genre_id determines genre type. Therefore book_idd determines genre type via genre_id and we have transitive functional dependency.

After decomposing it into third normal form it looks like:

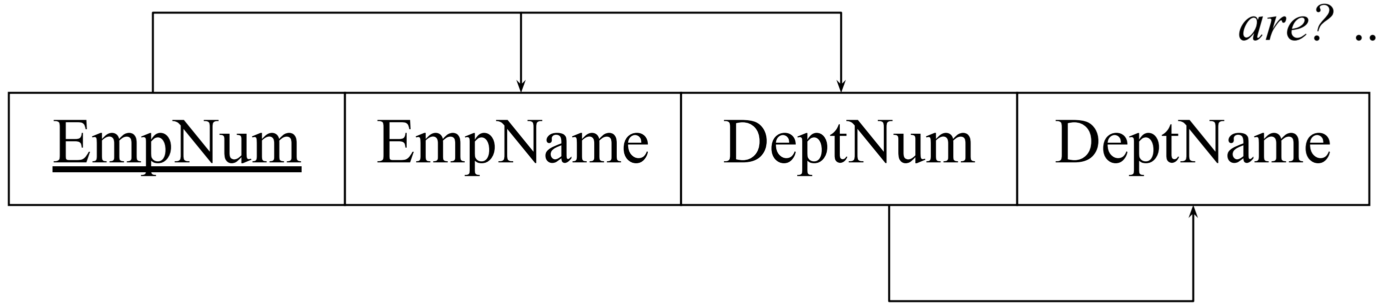
TABLE BOOK		
Book_id	Genre_id	Price
1	1	100
2	2	110
3	1	120
4	3	130
5	2	140

TABLE GENRE	
Genre_id	Genre type
1	Fiction
2	Sports
3	Travel

Third Normal Form

Consider this **Employee** relation

Candidate keys are? ...



EmpName, DeptNum, and DeptName are non-key attributes.

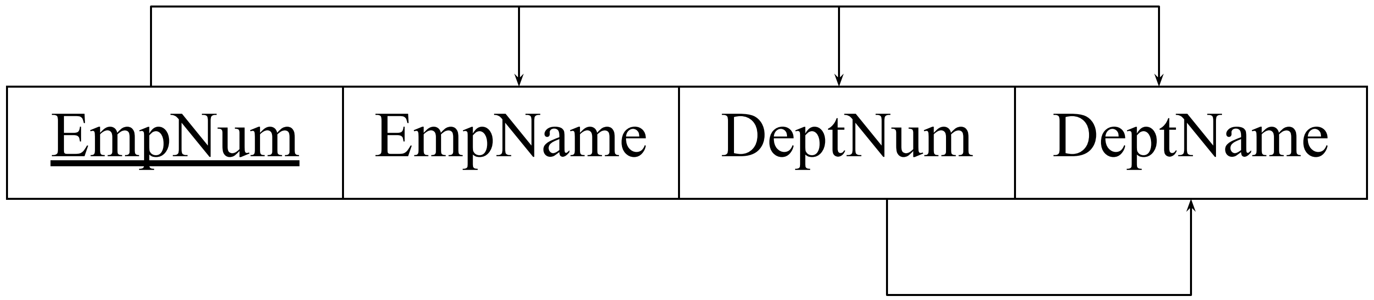
DeptNum determines DeptName, a non-key attribute, and DeptNum is not a candidate key.

Is the relation in 3NF? … no

Is the relation in BCNF? … no

Is the relation in 2NF? … yes

Third Normal Form



We correct the situation by decomposing the original relation into two 3NF relations. Note the decomposition is *lossless*.

EmpNum	EmpName	DeptNum

DeptNum	DeptName

Verify these two relations are in 3NF.

Third Normal Form

R(ABCD)

FD$\{ AB \rightarrow C,\ C \underset{\text{TD}}{\rightarrow} D \}$

- A relation is in 3**NF** if it is in 2NF, and no non-prime attribute is dependent on another non-prime attribute. (If L.H.S. is a candidate key or R.H.S. is prime attribute then it is already in 3NF otherwise not in 3NF).

CK={AB}
PA={A,B}
NPA={C,D}

R(ABCD)
After Decomposition:
Step1: R1(ABC), R2(CD)

Boyce-Codd Normal Form (BCNF)

- It is an advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency X->Y, X should be the super key of the table.

Boyce-Codd Normal Form

Student	Course	Teacher
Aman	DBMS	AYUSH
Aditya	DBMS	RAJ
Abhinav	E-COMM	RAHUL
Aman	E-COMM	RAHUL
abhinav	DBMS	RAJ

- KEY: {Student, Course}
- Functional dependency

  {student, course} -> Teacher

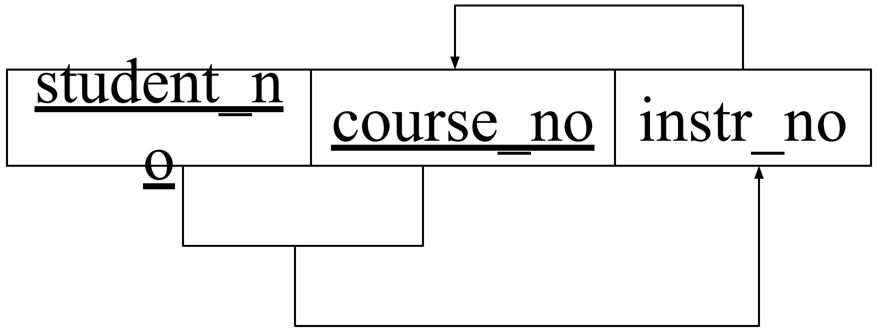
  Teacher-> Course
- Problem: teacher is not superkey but determines course.

After decomposing it into Boyce-Codd normal form it looks like:

Student	Course
Aman	DBMS
Aditya	DBMS
Abhinav	E-COMM
Aman	E-COMM
Abhinav	DBMS

Course	Teacher
DBMS	AYUSH
DBMS	RAJ
E-COMM	RAHUL

In 3NF, but not in BCNF:

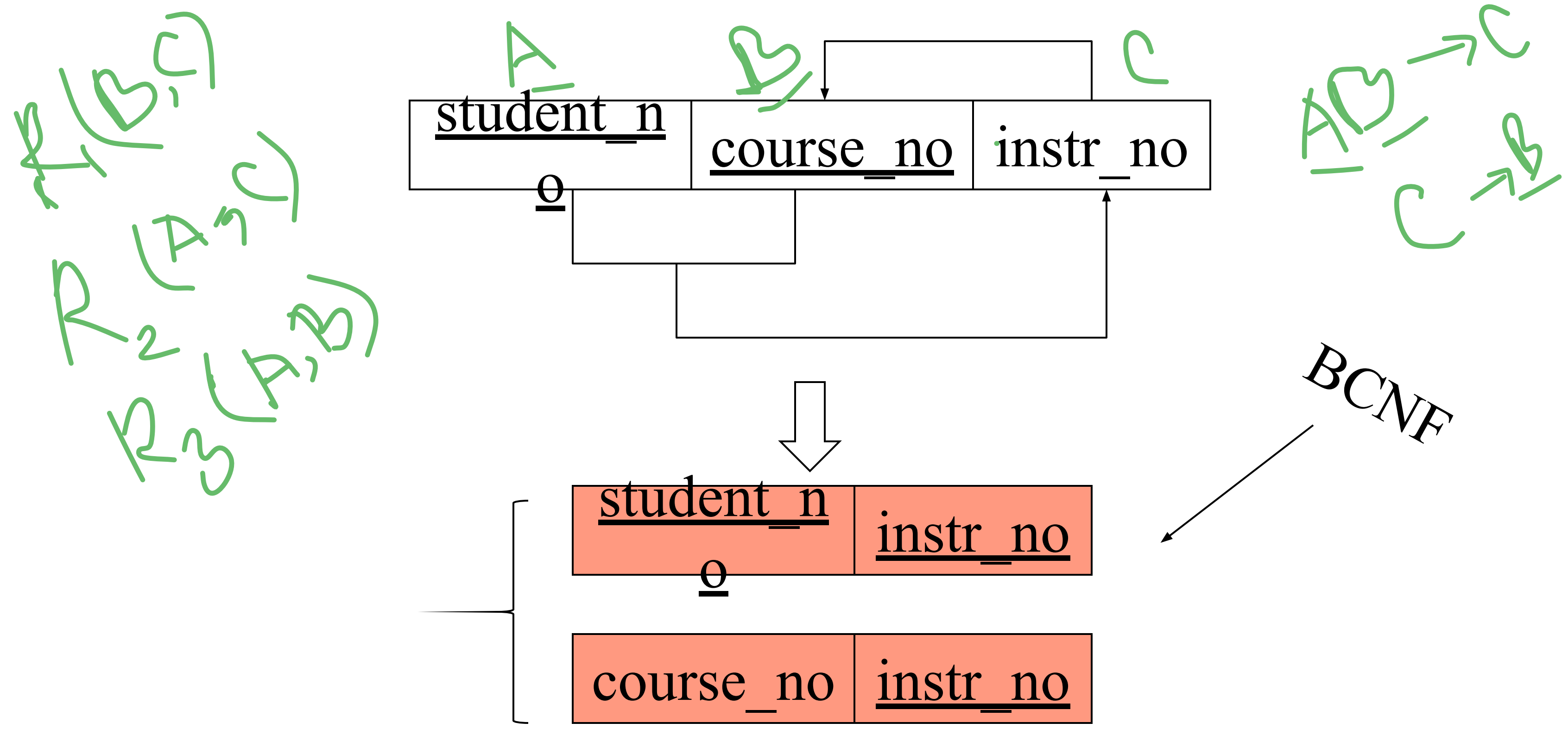


Instructor teaches one course only.

Student takes a course and has one instructor.

{student_no, course_no} → instr_no
instr_no → course_no

since we have instr_no → course-no, but instr_no is not a Candidate key.



{student_no, instr_no} → student_no
{student_no, instr_no} → instr_no
instr_no → course_no

Third Normal Form

R(ABCEF)

$FD\{\,AB \rightarrow C,\ C \underset{NC}{\rightarrow} E,\ E \underset{NC}{\rightarrow} F,\ F \underset{NC}{\rightarrow} A\}$

- A relation is in BC**N**F if it is in 3NF, and non-prime attribute must be dependent on candidate key. (If L.H.S. is a candidate key then it is already in BCNF otherwise not in BCNF).

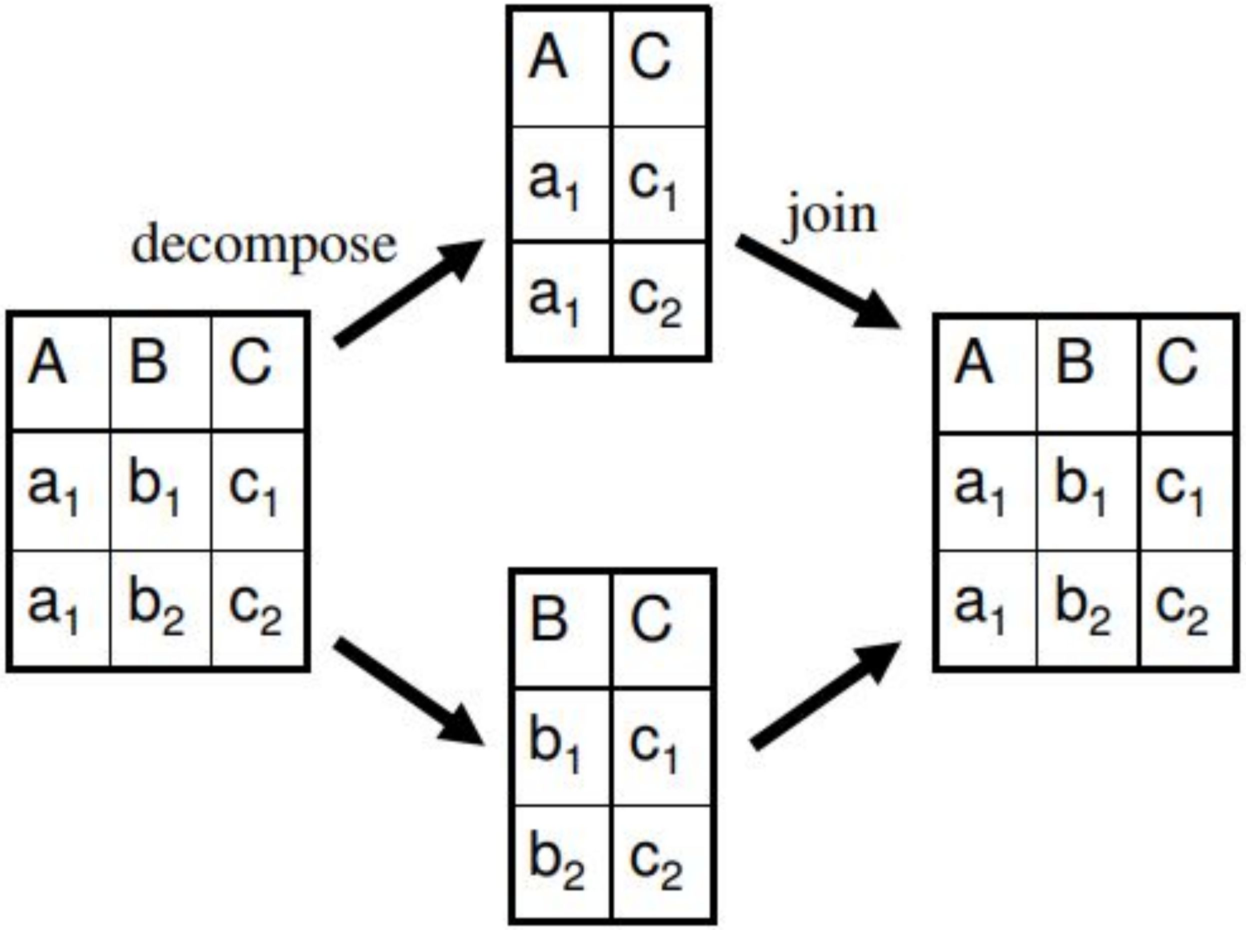
CK={AB, FB, EB,CB}
PA={A,B,C,E,F}
NPA={Ø}

R(ABCEF)
After Decomposition:
R1(CE), R2(EF), R3(FA), R4(ABC)

Goals of Normalization

- Let *R* be a relation scheme with a set *F* of functional dependencies.
- Decide whether a relation scheme *R* is in "good" form.
- In the case that a relation scheme *R* is not in "good" form, decompose it into a set of relation scheme $\{R_1, R_2, ..., R_n\}$ such that
  - each relation scheme is in good form
  - the decomposition is **a lossless-join decomposition**
  - Preferably, the decomposition should be dependency preserving.

Example (Lossless-Join)



A	B	C
a_1	b_1	c_1
a_1	b_2	c_2

A	C
a_1	c_1
a_1	c_2

B	C
b_1	c_1
b_2	c_2

A	B	C
a_1	b_1	c_1
a_1	b_2	c_2

Testing for Lossless-Join Decomposition

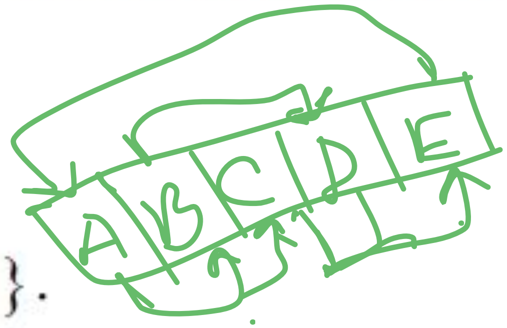
- Rule: A decomposition of R into $(R1, R2)$ is *lossless*, iff:

$$R1 \cap R2 \rightarrow R1 \quad \text{or}$$

$$R1 \cap R2 \rightarrow R2$$

in $F+$.

Exercise: Lossless-join Decomposition



R = {A,B,C,D,E}.

F = {A→BC, CD →E, B → D, E→A }.

Is the following decomposition a lossless join?

1. R1 = {A,B,C}, $\rightarrow A$ R2 = {A,D,E}

   Since R1 ∩ R2 = A, and A is a key for R1, the decomposition is lossless join.

2. R1 = {A,B,C}, C $\rightarrow A$ R2 = {C,D,E} $\rightarrow CD \rightarrow CK$

   Since R1 ∩ R2 = C, and C is not a key for R1 or R2, the decomposition is not lossless join.

Exercise 3

R =(A, B, C, D).

F = $\{C \rightarrow D, C \rightarrow A, B \rightarrow C\}$.

Question 1: Identify all candidate keys for R.

Question 2: Identify the best normal form that R satisfies.

Question 3: Decompose R into a set of BCNF relations.

Question 4: Decompose R into a set of 3NF relations.

Exercise 3 Solution

R =(A, B, C, D).

F = $\{C \rightarrow D, C \rightarrow A, B \rightarrow C\}$.

Question 1: Identify all candidate keys for R.

$$
\begin{aligned}
B^+ &= B & (B \rightarrow B)\\
&= BC & (B \rightarrow C)\\
&= BCD & (C \rightarrow D)\\
&= ABCD & (C \rightarrow A)
\end{aligned}
$$

so the candidate key is B.

B is the ONLY candidate key, because nothing determines B:

There is no rule that can produce B, except B $\rightarrow$B.

Exercise 3 Solution

R =(A, B, C, D).

F = {C→D, C→A, B→C}.

Question 2: Identify the best normal form that R satisfies.

R is not 3NF, because:

C→D causes a violation,

- C→D is non-trivial ({D} $\not\subset$ {C}).
- C is not a superkey.
- D is not part of any candidate key.

C→A causes a violation

- Similar to above

B→C causes no violation

Since R is not 3NF, it is not BCNF either.

Exercise 3 Solution

R =(A, B, C, D).

F = {$C \rightarrow D$, $C \rightarrow A$, $B \rightarrow C$}.

Question 3: Decompose R into a set of BCNF relations

(1)$C \rightarrow D$ and $C \rightarrow A$ both cause violations of BCNF.

Take $C \rightarrow D$: decompose R to R_1= {A, B, C} , R_2={C, D}.

(2)Now check for violations in R_1 and $R_{2.}$ (Actually, using F^+)

R_1 still violates BCNF because of $C \rightarrow A$.

Decompose R_1 to R_{11} = {B, C} R_{12} = {C, A}.

Final decomposition: R_2 = {C, D}, R_{11} = {B, C}, R_{12} = {C, A}.

No more violations: Done!

Exercise 3 Solution

$R = (A, B, C, D)$.

$F = \{C \rightarrow D, C \rightarrow A, B \rightarrow C\}$.

Question 4: Decompose R into a set of 3NF relations.

The canonical cover is $F_c = \{C \rightarrow DA, B \rightarrow C\}$.

For each functional dependency in F_c we create a table:

$R_1 = \{C, D, A\}$, $R_2 = \{B, C\}$.

The table R_2 contains the candidate key for R – we done.

Exercise 4

R = (A, B, C, D)

F = $\{AB \rightarrow C, AB \rightarrow D, C \rightarrow A, D \rightarrow B\}$

1. Is R in 3NF, why? If it is not, decompose it into 3NF

2. Is R in BCNF, why? If it is not, decompose it into BCNF

Exercise 4 Solution

R = (A, B, C, D)

F = {AB→C, AB→D, C→A, D→B}

1. Is R in 3NF, why? If it is not, decompose it into 3NF

Yes.

Find all the Candidate Keys:

AB, BC, CD, AD

Check all FDs in F for 3NF condition

2. Is R in BCNF, why? If it is not, decompose it into BCNF

No. Because for C→A, C is not a superkey. Similar for D→B

R1 = $\underbrace{\{C, D\}}_{ck}$, $\underbrace{R2 = \{A, C\}, R3 = \{B, D\}}_{ck}$

End of Chapter