# Normalization

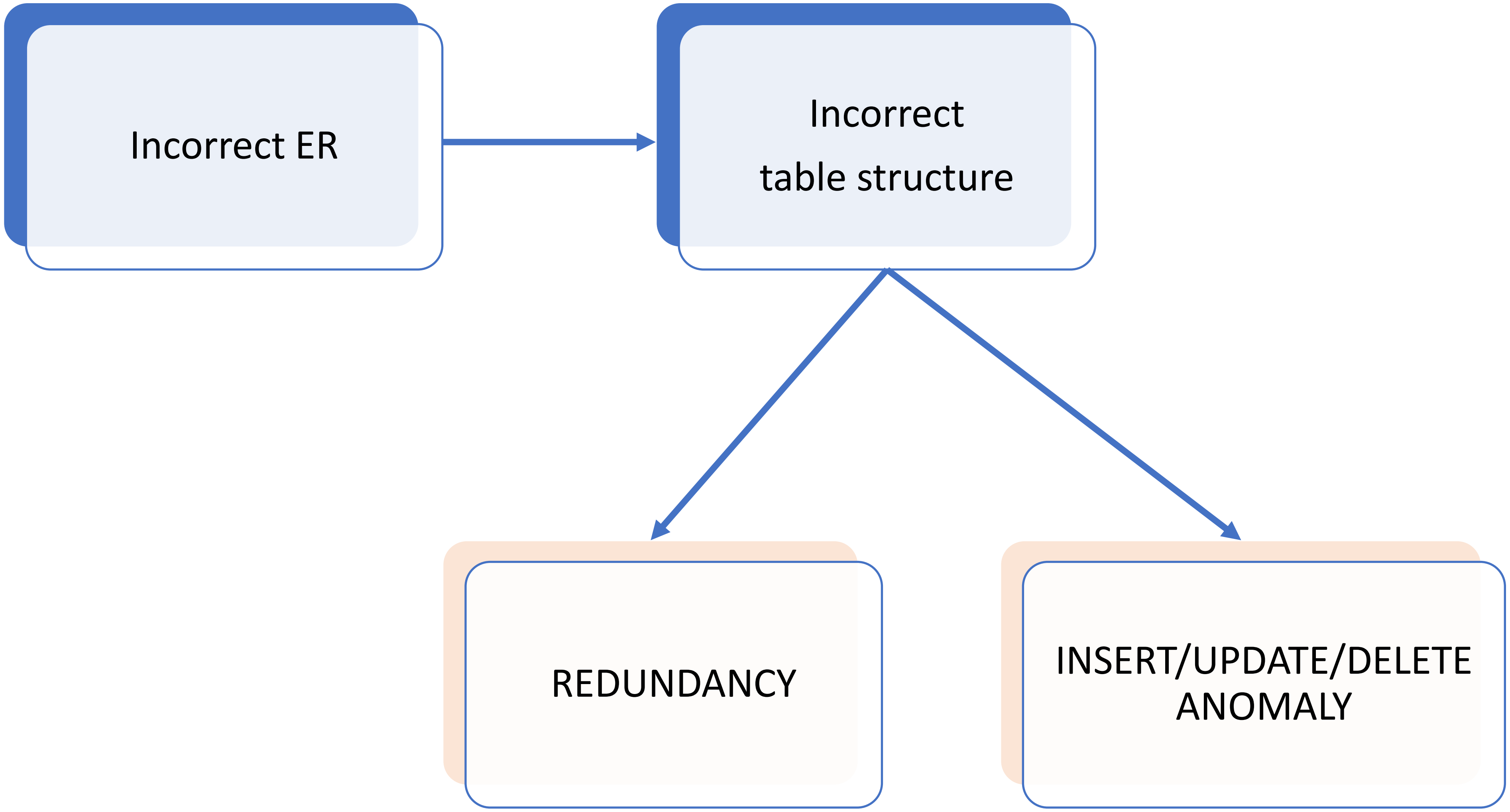
COURSE 5: Databases

# Normalization

when and why

# Normalization

- Informal:
  - Organize data in a relational database in order to avoid redundancy and data manipulation anomalies.
  - Decompose a relation (table) without loosing information.

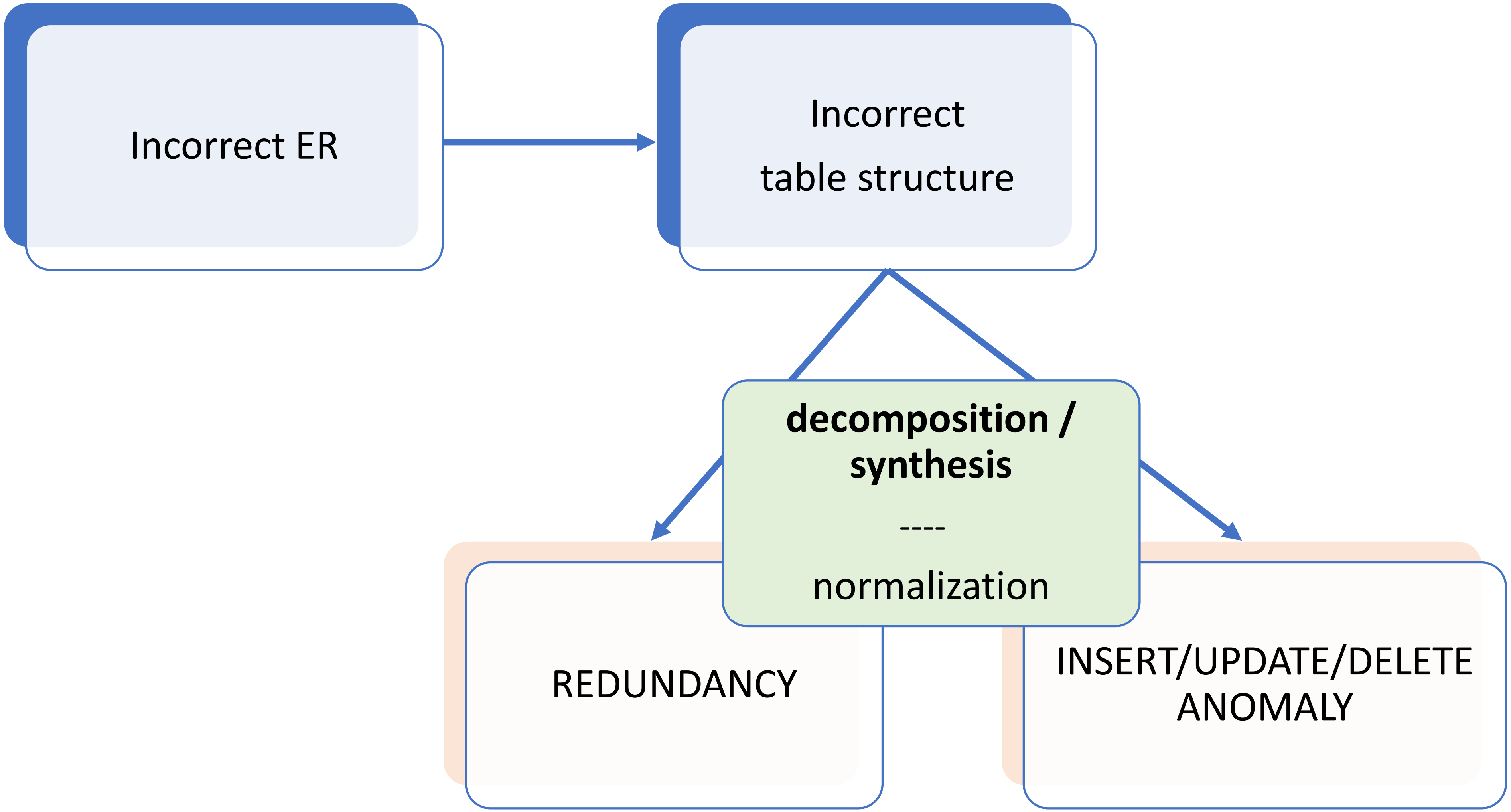


Incorrect ER

Incorrect table structure

REDUNDANCY

INSERT/UPDATE/DELETE ANOMALY



Incorrect ER

Incorrect table structure

**decomposition / synthesis**

----

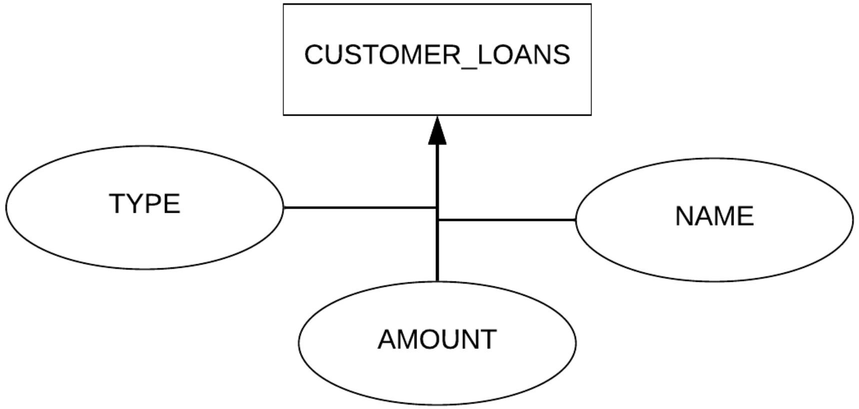
normalization

REDUNDANCY

INSERT/UPDATE/DELETE ANOMALY

# Normalization

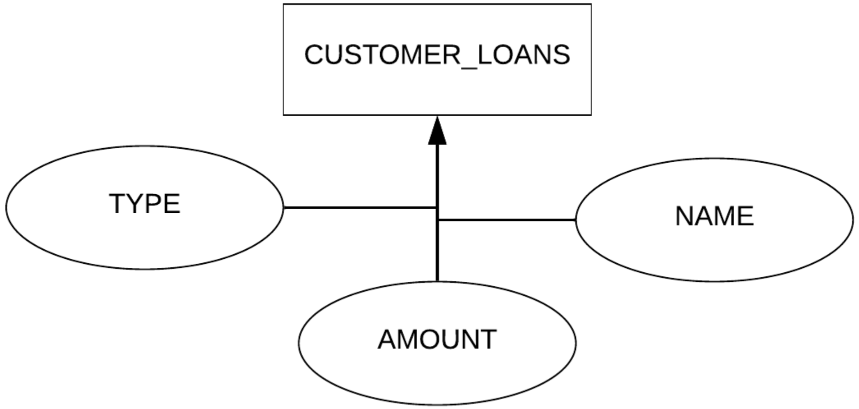
- Avoid redundancy



| CUSTOMER_ID | NAME | LOAN_ID | TYPE | AMOUNT |
|---|---|---|---|---|
| 1 | Smith | 101 | mortgage | 125000 |
| 1 | Smith | 102 | credit card | 25000 |
| 2 | Green | 103 | credit card | 12500 |
| 2 | Green | 127 | mortgage | 20000 |
| 3 | Avery | 389 | mortgage | 75000 |
| 3 | Avery | 486 | credit card | 5000 |
| 3 | Avery | 769 | mortgage | 45000 |

# Normalization

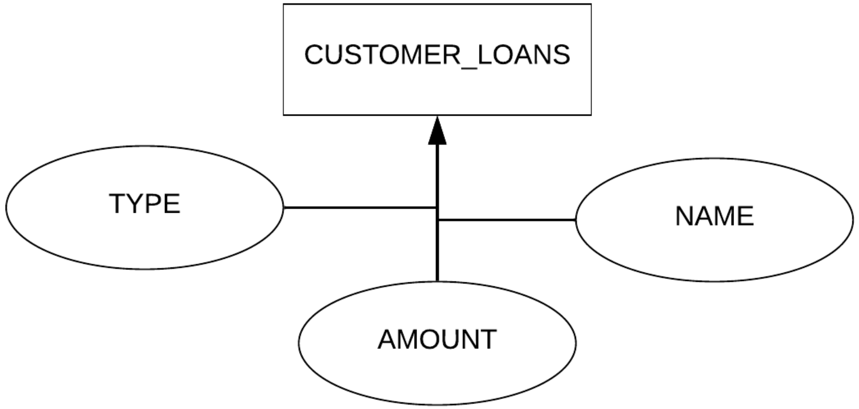
- INSERT anomaly: insert a new customer



| CUSTOMER_ID | NAME | LOAN_ID | TYPE | AMOUNT |
|---|---|---|---|---|
| 1 | Smith | 101 | mortgage | 125000 |
| 1 | Smith | 102 | credit card | 25000 |
| 2 | Green | 103 | credit card | 12500 |
| 2 | Green | 127 | mortgage | 20000 |
| 3 | Avery | 389 | mortgage | 75000 |
| 3 | Avery | 486 | credit card | 5000 |
| 4 | Stark | ??? | null | null |

# Normalization

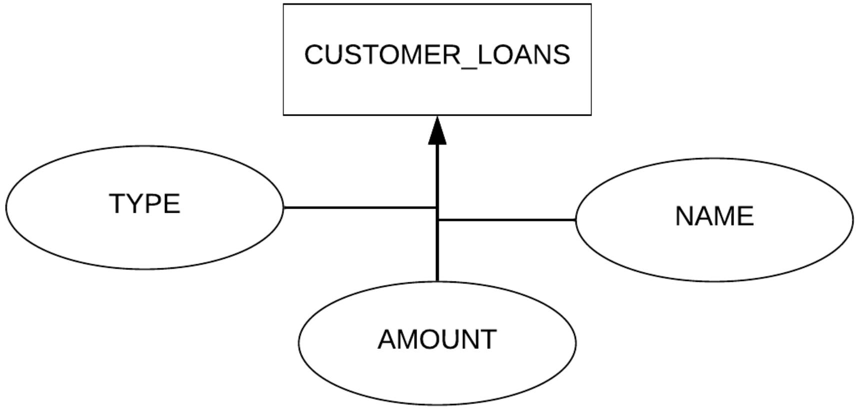
- UPDATE anomaly: update customer name



| CUSTOMER_ID | NAME | LOAN_ID | TYPE | AMOUNT |
|---|---|---|---|---|
| 1 | Smith | 101 | mortgage | 125000 |
| 1 | Smith | 102 | credit card | 25000 |
| 2 | Green | 103 | credit card | 12500 |
| 2 | Green | 127 | mortgage | 20000 |
| 3 | Avery | 389 | mortgage | 75000 |
| 3 | Avery | 486 | credit card | 5000 |
| 3 | Avery | 769 | mortgage | 45000 |

# Normalization

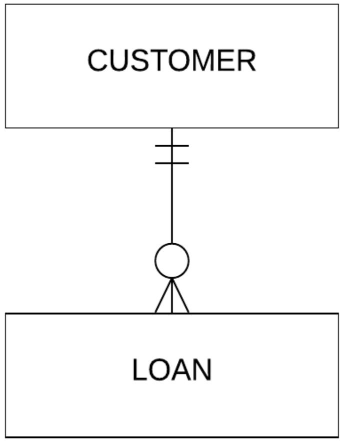
- DELETE anomaly: delete loan

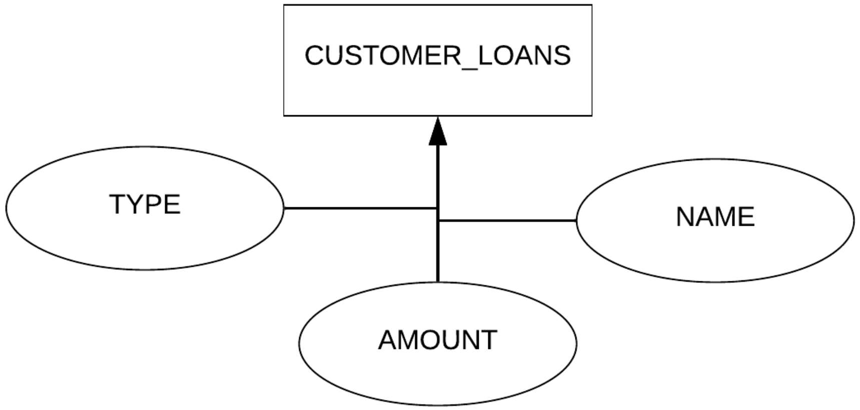


| CUSTOMER_ID | NAME | LOAN_ID | TYPE | AMOUNT |
|---|---|---|---|---|
| 1 | Smith | 101 | mortgage | 125000 |
| 1 | Smith | 102 | credit card | 25000 |
| 2 | Green | 103 | credit card | 12500 |
| 2 | Green | 127 | mortgage | 20000 |
| 3 | Avery | 389 | mortgage | 75000 |
| 3 | Avery | 486 | credit card | 5000 |
| 4 | Stark | 700 | mortgage | 45000 |

# Normalization

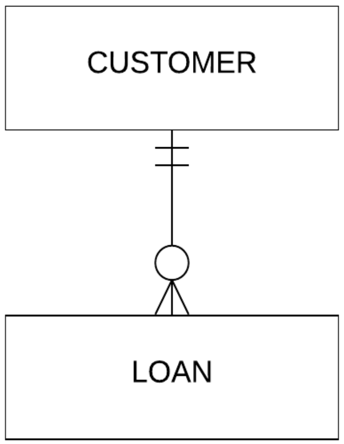
- Decompose relation





# Normalization

- Decompose relation



| CUSTOMER | | | |
|---|---|---|---|
| **CUSTOMER_ID** | **LAST_NAME** | **…** | **….** |
| 1 | Smith | … | …. |
| 2 | Green | … | …. |
| 3 | Avery | … | …. |

| LOAN | | | |
|---|---|---|---|
| **LOAN_ID** | **CUSTOMER_ID** | **AMOUNT** | **DATE** |
| 101 | 1 | 125000 | 18/04/21 |
| 102 | 1 | 25000 | 14/04/22 |
| 103 | 2 | 12500 | 03/05/21 |
| 127 | 2 | 20000 | … |
| 389 | 3 | 75000 | … |

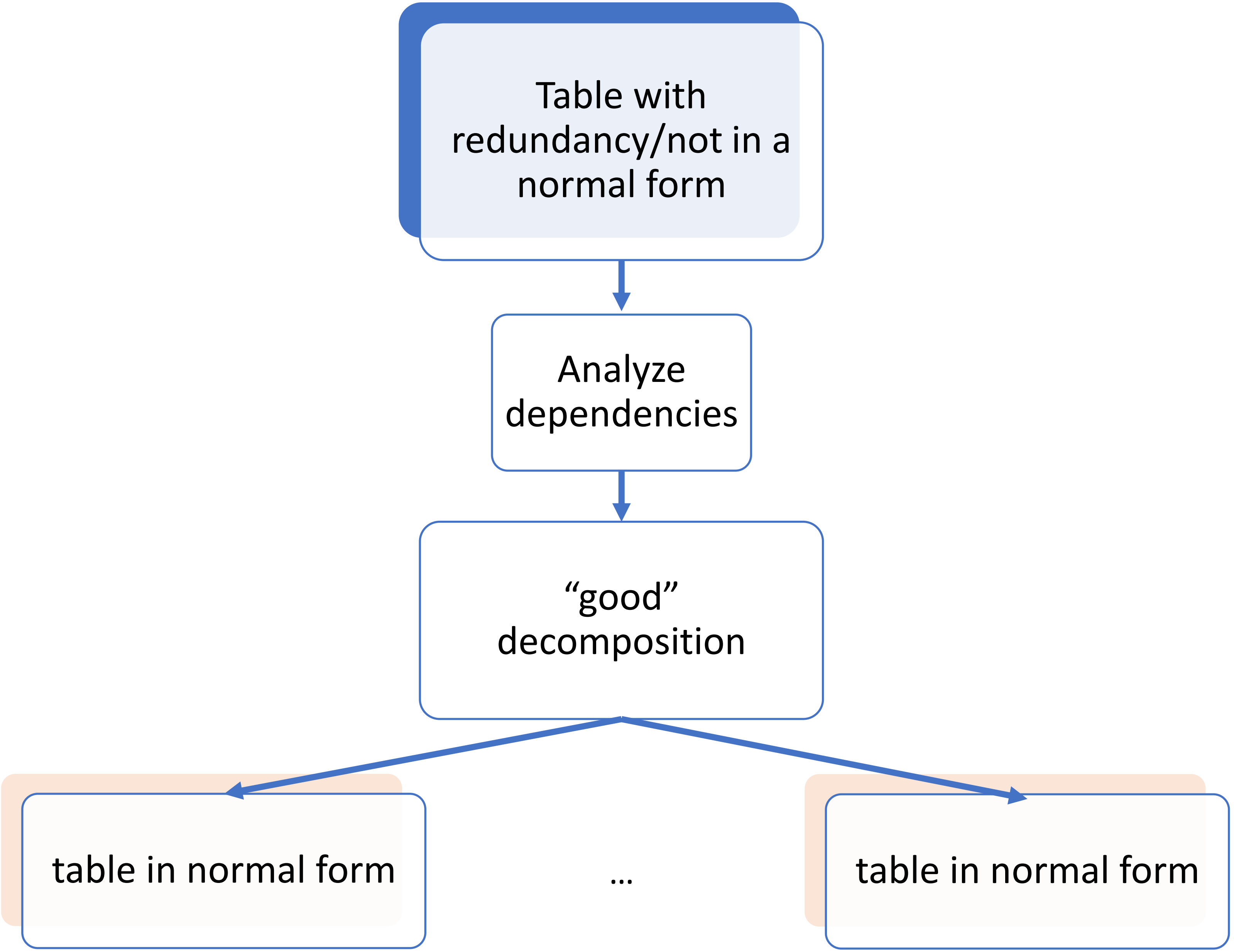


Table with redundancy/not in a normal form
Analyze dependencies
“good” decomposition
table in normal form
…
table in normal form

# Decomposition

# Decomposition Step 1: Projection

- $S_1 = \Pi_{(NAME,LOANID,TYPE,AMOUNT)} R$

| NAME | LOAN_ID | TYPE | AMOUNT |
|---|---|---|---|
| Smith | 101 | mortgage | 125000 |
| Smith | 102 | credit card | 25000 |
| Green | 103 | credit card | 12500 |
| Smith | 389 | mortgage | 75000 |

| CUSTOMER_ID | NAME | LOAN_ID | TYPE | AMOUNT |
|---|---|---|---|---|
| 1 | Smith | 101 | mortgage | 125000 |
| 1 | Smith | 102 | credit card | 25000 |
| 2 | Green | 103 | credit card | 12500 |
| 3 | Smith | 389 | mortgage | 75000 |

- $S_2 = \Pi_{(CUSTOMER\_ID,NAME)} R$

| CUSTOMER_ID | NAME |
|---|---|
| 1 | Smith |
| 2 | Green |
| 3 | Smith |

# Decomposition Step 2: Join

| CUSTOMER_ID | NAME |
|---|---|
| 1 | Smith |
| 2 | Green |
| 3 | Smith |

| NAME | LOAN_ID | TYPE | AMOUNT |
|---|---|---|---|
| Smith | 101 | mortgage | 125000 |
| Smith | 102 | credit card | 25000 |
| Green | 103 | credit card | 12500 |
| Smith | 389 | mortgage | 75000 |

- Lossy decomposition
  $S_1 \bowtie S_2 \supseteq R$

| CUSTOMER_ID | NAME | LOAN_ID | TYPE | AMOUNT |
|---|---|---|---|---|
| 1 | Smith | 101 | mortgage | 125000 |
| 1 | Smith | 102 | credit card | 25000 |
| 1 | Smith | 389 | mortgage | 75000 |
| 3 | Smith | 101 | mortgage | 125000 |
| 3 | Smith | 102 | credit card | 25000 |
| 3 | Smith | 389 | mortgage | 75000 |
| 2 | Green | 103 | credit card | 12500 |

# Decomposition Step 1: Projection

- $S_1 = \Pi_{(CUSTOMERID,LOANID,TYPE,AMOUNT)}$

| CUSTOMER_ID | LOAN_ID | TYPE | AMOUNT |
|---|---|---|---|
| 1 | 101 | mortgage | 125000 |
| 1 | 102 | credit card | 25000 |
| 2 | 103 | credit card | 12500 |
| 3 | 389 | mortgage | 75000 |

| CUSTOMER_ID | NAME | LOAN_ID | TYPE | AMOUNT |
|---|---|---|---|---|
| 1 | Smith | 101 | mortgage | 125000 |
| 1 | Smith | 102 | credit card | 25000 |
| 2 | Green | 103 | credit card | 12500 |
| 3 | Smith | 389 | mortgage | 75000 |

| CUSTOMER_ID | NAME |
|---|---|
| 1 | Smith |
| 2 | Green |
| 3 | Smith |

- $S_2 = \Pi_{(CUSTOMERID,NAME)} R$

# Decomposition Step 2: Join

| CUSTOMER_ID | NAME |
|---|---|
| 1 | Smith |
| 2 | Green |
| 3 | Smith |

| CUSTOMER_ID | LOAN_ID | TYPE | AMOUNT |
|---|---|---|---|
| 1 | 101 | mortgage | 125000 |
| 1 | 102 | credit card | 25000 |
| 2 | 103 | credit card | 12500 |
| 3 | 389 | mortgage | 75000 |

- Lossless decomposition
  $\boldsymbol{S_1 \bowtie S_2 = R}$

| CUSTOMER_ID | NAME | LOAN_ID | TYPE | AMOUNT |
|---|---|---|---|---|
| 1 | Smith | 101 | mortgage | 125000 |
| 1 | Smith | 102 | credit card | 25000 |
| 3 | Smith | 389 | mortgage | 75000 |
| 2 | Green | 103 | credit card | 12500 |

# Decomposition

- lossy decompositions and lossless decompositions.

- Lossy: R → decompose(R): S1, S2 → recompose(S1,S2) $\supseteq$ R

lossy =/= less data, (less is more!)

lossy = lost information

- Lossless R → decompose(R): S1, S2 → recompose(S1,S2) = R

# Decomposition

- Lossy

$$\prod_{R1} R \bowtie \prod_{R2} R \supseteq \mathrm{R}$$

- Lossless

$$\prod_{R1} R \bowtie \prod_{R2} R = \mathrm{R}$$

# Functional dependencies

# Functional dependencies

| CUSTOMER_ID | NAME |
|---|---|
| 1 | Smith |
| 2 | Green |
| 3 | Smith |

| X | Y | Z | T |
|---|---|---|---|
| X1 | Y1 | Z1 | T1 |
| X1 | Y2 | Z1 | T2 |
| X2 | Y2 | Z2 | T2 |
| X2 | Y3 | Z2 | T3 |
| X3 | Y3 | Z2 | T4 |

- CUSTOMER_ID → NAME
- X → Z
- Z ↛ T, (Y,Z) ↛ X
- X → X

U → V functional dependency:
every value of U uniquely
determines the value of V

U determinant, V dependent

# Functional dependencies

| CUSTOMER_ID | NAME |
|---|---|
| 1 | Smith |
| 2 | Green |
| 3 | Smith |

| X | Y | Z | T |
|---|---|---|---|
| X1 | Y1 | Z1 | T1 |
| X1 | Y2 | Z1 | T2 |
| X2 | Y2 | Z2 | T2 |
| X2 | Y3 | Z2 | T3 |
| X3 | Y3 | Z2 | T4 |

- CUSTOMER_ID → CUSTOMER_ID
- (X, Y) → (X, Y)
- Z → Z
- (Z,T) → T

U → V trivial dependency:

$V \subseteq U$

# Functional dependencies

| CUSTOMER_ID | NAME |
|---|---|
| 1 | Smith |
| 2 | Green |
| 3 | Smith |

| X | Y | Z | T |
|---|---|---|---|
| X1 | Y1 | Z1 | T1 |
| X1 | Y2 | Z1 | T2 |
| X2 | Y2 | Z2 | T2 |
| X2 | Y3 | Z2 | T3 |
| X3 | Y3 | Z2 | T4 |

- CUSTOMER_ID -> NAME
- (X, Y) → (T)

U → V non-trivial dependency:

$V \nsubseteq U$

# Functional dependencies

| CUSTOMER_ID | NAME |
|---|---|
| 1 | Smith |
| 2 | Green |
| 3 | Smith |

| X | Y | Z | T |
|---|---|---|---|
| X1 | Y1 | Z1 | T1 |
| X1 | Y2 | Z1 | T2 |
| X2 | Y2 | Z2 | T2 |
| X2 | Y3 | Z2 | T3 |
| X3 | Y3 | Z2 | T4 |

- CUSTOMER_ID -> NAME
- $(X, Y) \rightarrow T$
- $X \nrightarrow T$
- $Y \nrightarrow T$

$U \rightarrow V$ fully-functional dependency:

$$U' \subseteq U \Rightarrow U' \nrightarrow V$$

# Functional dependencies properties

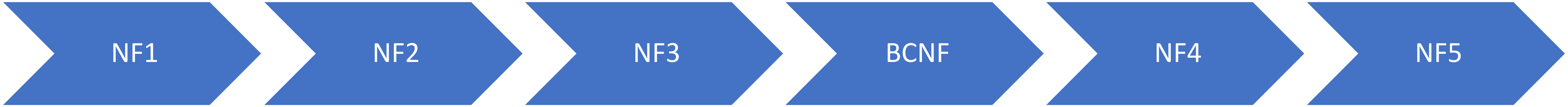
- Reflexive if $V \subseteq U$ then $U \rightarrow V$
- Transitivity $U \rightarrow V$ and $V \rightarrow W$, then $U \rightarrow W$
- Augmentation $X \rightarrow Y$ then $X \cup Z \rightarrow Y \cup Z$

  $X \rightarrow Y$ then $X \cup Z \rightarrow Y$

  $X \rightarrow Y$ and $X \subseteq Z$, then $Z \rightarrow Y$

  $X \rightarrow Y$ and $W \subseteq Z$, then $X \cup Z \rightarrow Y \cup W$

# Normal Forms



NF1 → NF2 → NF3 → BCNF → NF4 → NF5

# NF1

ATOMIC ATTRIBUTES

# NF1

- Atomic attributes
- No multi-valued attributes

- The domain of each attribute contains only atomic values and each attribute contains only a value of its domain.

- A relational  database is **at least in NF1**

# NF1

| EMP_ID | NAME | EMAIL |
|---|---|---|
| 1 | Williams | williams@gmail.com<br>williams@yahoo.com |
| 2 | Davis | davis@gmail.com<br>davis@academy.com |
| 3 | Miller | miller@gmail.com |
| 4 | Stewart | stewart@gmail.com<br>office@academy.com |

Optional: Artificial key

Composed key

| EMAIL_ID | EMP_ID | EMAIL |
|---|---|---|
| 1 | 1 | williams@gmail.com |
| 2 | 1 | williams@yahoo.com |
| 3 | 2 | davis@gmail.com |
| 4 | 2 | davis@academy.com |
| 5 | 3 | miller@gmail.com |
| 6 | 4 | stewart@gmail.com |
| 7 | 4 | office@academy.com |

# NF2

NO PARTIAL DEPENDENCIES

# NF2

- Tables in NF1
- No non-key attributes (not part of the key) that depend on a subset of the attributes forming the key.

- There are no partial dependencies.

# Functional dependencies

| X | Y | Z | T |
|---|---|---|---|
| X1 | Y1 | Z1 | T1 |
| X2 | Y1 | Z1 | T2 |
| X2 | Y2 | Z2 | T3 |
| X2 | Y3 | Z2 | T3 |
| X2 | Y3 | Z2 | T3 |

| X | Y | Z | T |
|---|---|---|---|
| X1 | **Y1** | Z1 | … |
| X2 | **Y1** | Z1 | … |
| X2 | **Y2** | Z2 | … |
| X2 | **Y3** | Z2 | … |
| X2 | **Y3** | Z2 | … |

- partial (X,Y) → Z
  - Y → Z

# Functional dependencies

| X | Y | Z | T |
|---|---|---|---|
| X1 | Y1 | … | T1 |
| X2 | Y1 | … | T2 |
| X2 | Y2 | … | T3 |
| X2 | Y3 | … | T3 |
| X2 | Y3 | … | T3 |

- total (X,Y) → T
  - X ↛ T
  - Y ↛ T

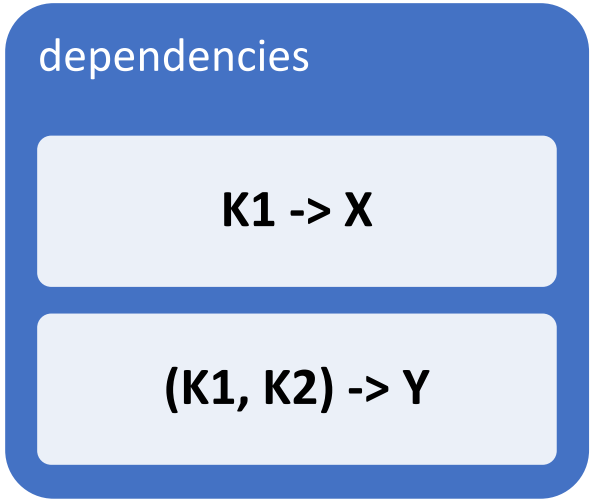
| X | Y | Z | T |
|---|---|---|---|
| … | **Y1** | … | **T1** |
| … | **Y1** | … | **T2** |
| … | **…** | … | … |
| … | **…** | … | … |
| … | **…** | … | … |

| X | Y | Z | T |
|---|---|---|---|
| … | **…** | … | **…** |
| **X2** | **…** | … | **T2** |
| **X2** | **…** | … | **T3** |
| … | **…** | … | … |
| … | **…** | … | … |

# Functional dependencies

| AIRPORT_ID | AIRPLANE_ID | DEPARTURE | AIRPLANE_MODEL | BOARDING_GATE |
|---|---|---|---|---|
| 1 | 101 | 30/03/20 17:00 | Boeing 777 | 42 |
| 1 | 102 | 02/05/20 09:30 | Airbus A320 | 50 |
| 2 | 201 | 06/08/20 10:45 | Boeing 757 | 35 |
| 2 | 202 | 10/10/20 06:20 | Airbus A320 | 10 |
| 1 | 101 | 06/04/20 16:35 | Boeing 777 | 23 |

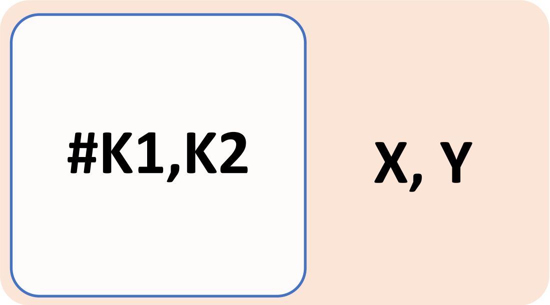
# NF2



dependencies

K1 -> X

(K1, K2) -> Y



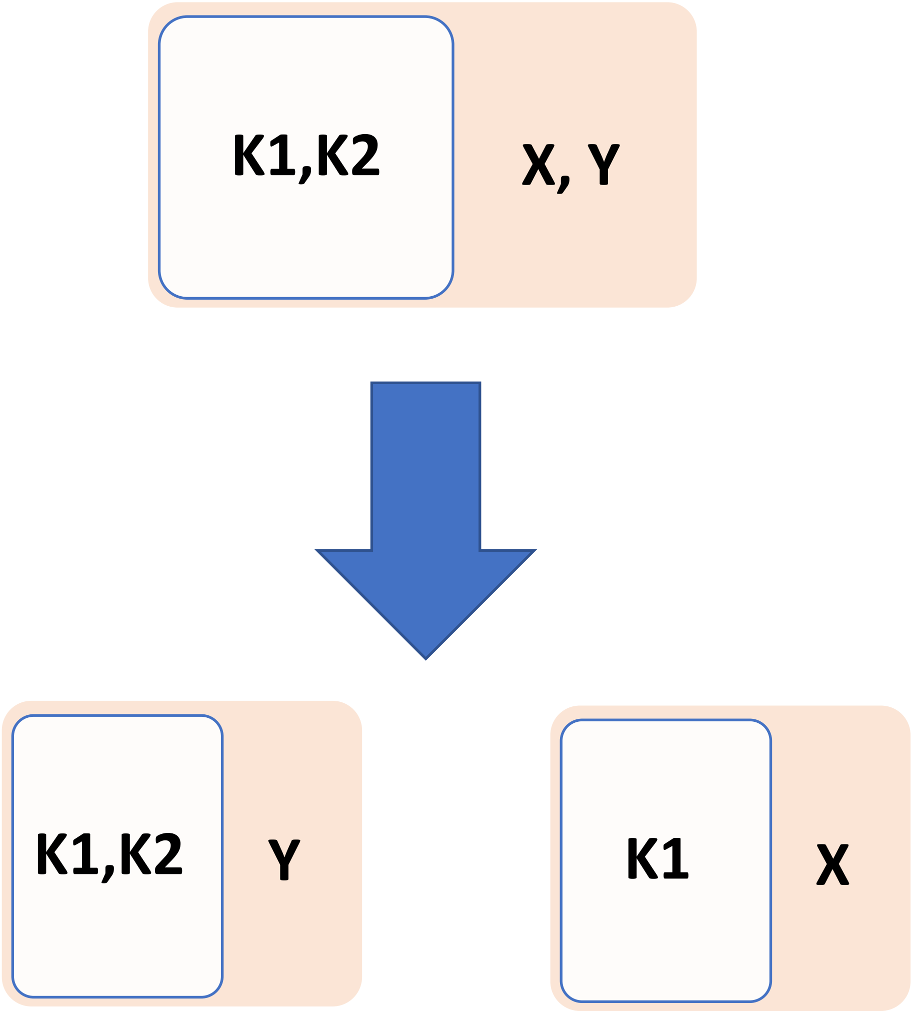
#K1,K2

X, Y

# NF2

dependencies

- K1 -> X
- (K1, K2) -> Y



K1,K2 X, Y

K1,K2 Y

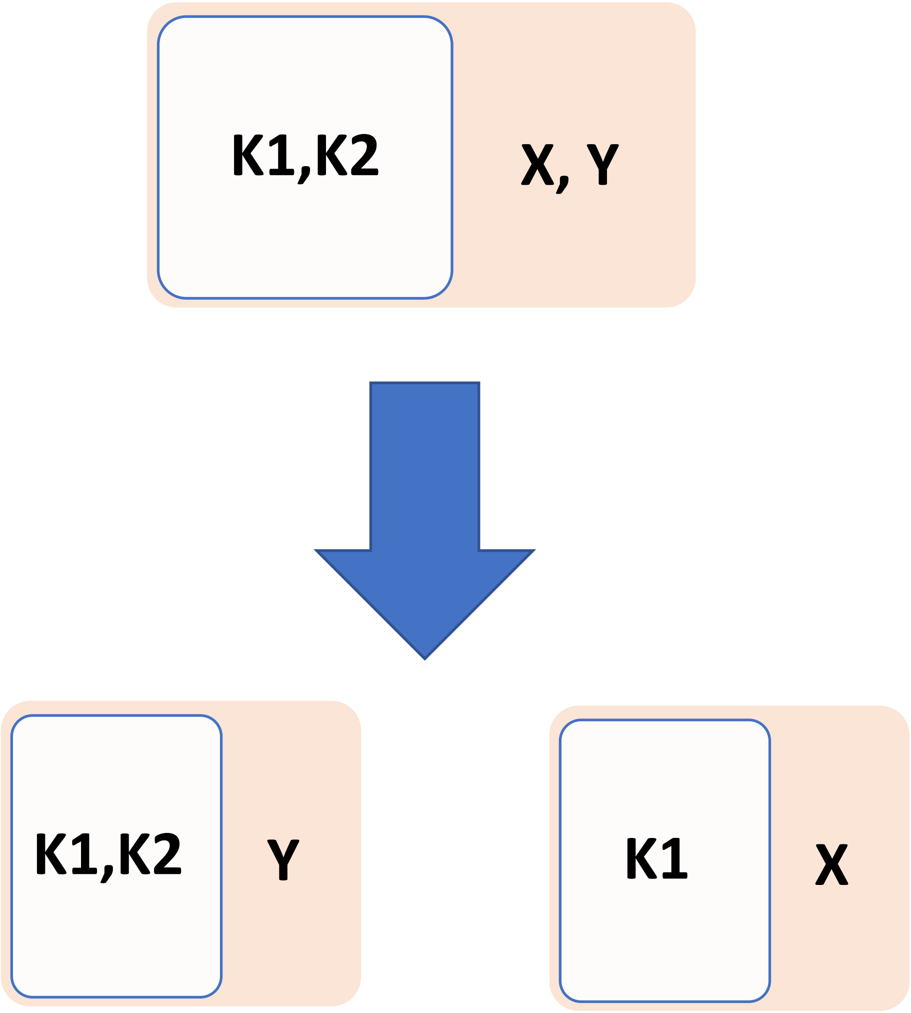
K1 X

# NF2

dependencies

K1 -> X

(K1, K2) -> Y



K1,K2

X, Y

K1,K2

Y

K1

X

K1 = AIRPLANE_ID
K2 = AIRPORT_ID, DEPARTURE
Y = BOARDING_GATE
X = AIRPLANE_MODEL

| AIRPORT_ID | AIRPLANE_ID | DEPARTURE | AIRPLANE_MODEL | BOARDING_GATE |
|---|---|---|---|---|
| 1 | 101 | 30/03/20 17:00 | Boeing 777 | 42 |
| 1 | 102 | 02/05/20 09:30 | Airbus A320 | 50 |
| 2 | 201 | 06/08/20 10:45 | Boeing 757 | 35 |
| 2 | 202 | 10/10/20 06:20 | Airbus A320 | 10 |
| 1 | 101 | 06/04/20 16:35 | Boeing 777 | 23 |

| AIRPORT_ID | AIRPLANE_ID | DEPARTURE | BOARDING_GATE |
|---|---|---|---|
| 1 | 101 | 30/03/20 17:00 | 42 |
| 1 | 102 | 02/05/20 09:30 | 50 |
| 2 | 201 | 06/08/20 10:45 | 35 |
| 2 | 202 | 10/10/20 06:20 | 10 |
| 1 | 101 | 06/04/20 16:35 | 23 |

| AIRPLANE_ID | AIRPLANE_MODEL |
|---|---|
| 101 | Boeing 777 |
| 102 | Airbus A320 |
| 201 | Boeing 757 |
| 202 | Airbus A320 |

# NF2

SCORES(PLAYER_ID, TOURNAMENT_ID, SCOR, PLAYER_NAME, RANK)

BILL(BILL_ID, CLIENT_ID, CLIENT_NAME, CLIENT_PHONES, PRODUCT_ID, PRODUCT_NAME, PROD_CAT_ID, PROD_CAT_NAME, QUANTITY)

RENT(CLIENT_ID, RENT_DATE, CAR_ID, CAR_MODEL, DISCOUNT, PRICE)

RENT_DATE → DISCOUNT

RESERVATION(CUSTOMER_ID, HOTEL_ID, CHECKIN_DATE, HOTEL_CITY, CUSTOMER_NAME)

# NF2

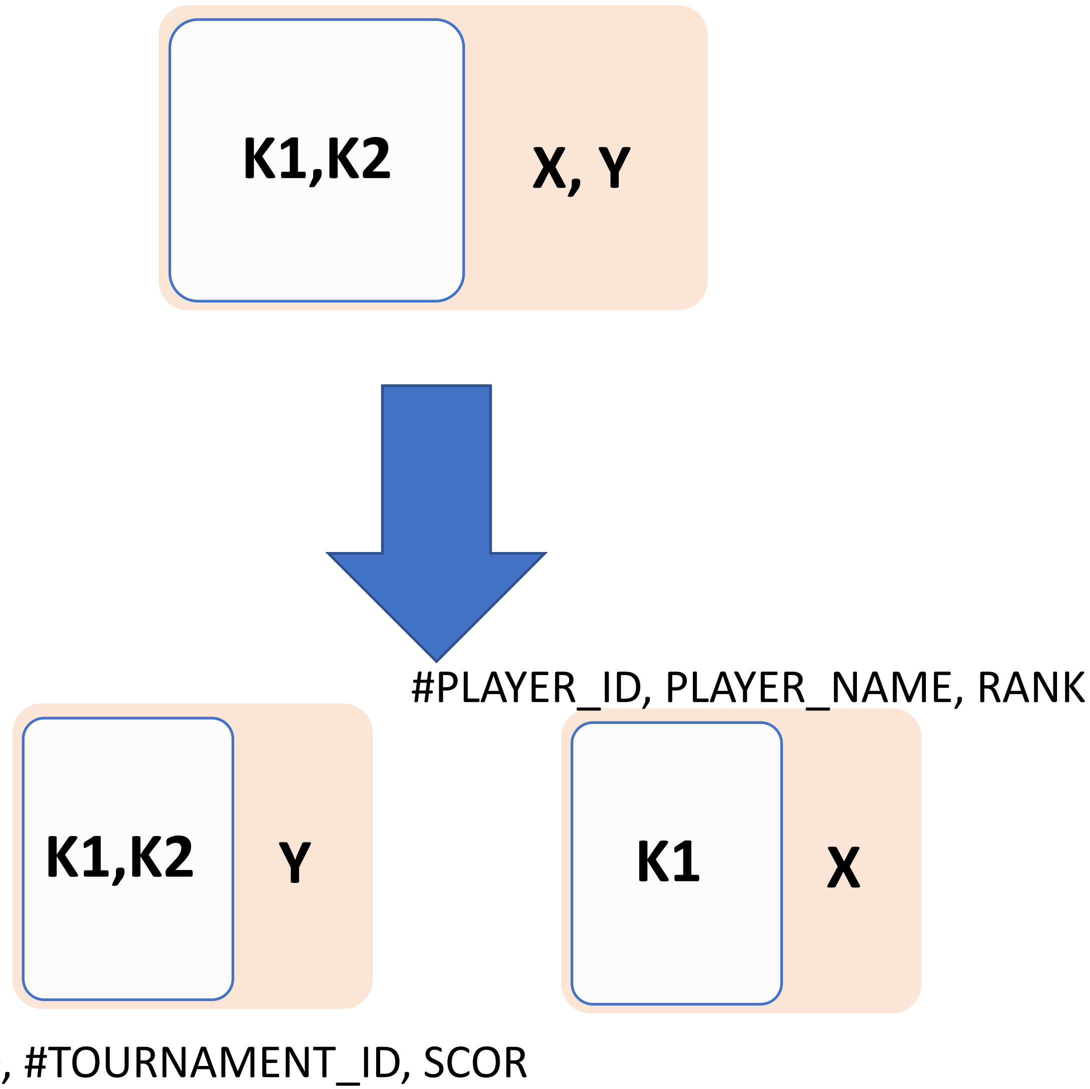
dependencies

**K1 -> X**

**(K1, K2) -> Y total**

SCORES(#PLAYER_ID, #TOURNAMENT_ID, SCOR, PLAYER_NAME, RANK)

K1 = PLAYER_ID
K2 = TOURNAMENT_ID
Y = SCOR
X = PLAYER_NAME, RANK


K1,K2
X, Y
#PLAYER_ID, PLAYER_NAME, RANK
K1,K2
Y
K1
X
#PLAYER_ID, #TOURNAMENT_ID, SCOR

# NF2

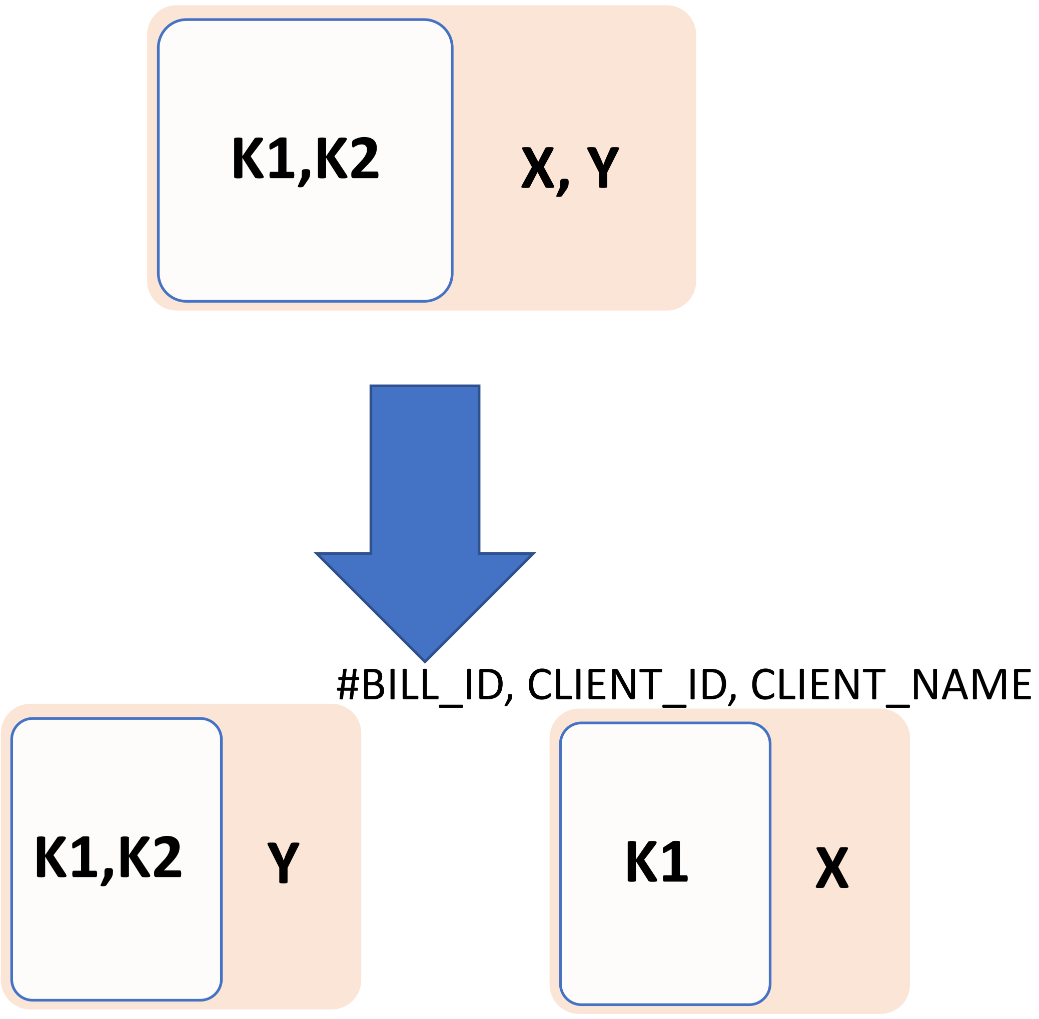
dependencies

K1 -> X

(K1, K2) -> Y total

BILL_PROD(#BILL_ID, CLIENT_ID, CLIENT_NAME, #PRODUCT_ID, PRODUCT_NAME, PROD_CAT_ID, PROD_CAT_NAME, QUANTITY)

K1 = BILL_ID
K2 = PRODUCT_ID
Y = PRODUCT_NAME,PROD_CAT_ID, PROD_CAT_NAME, QUANTITY
X = CLIENT_NAME, CLIENT_ID



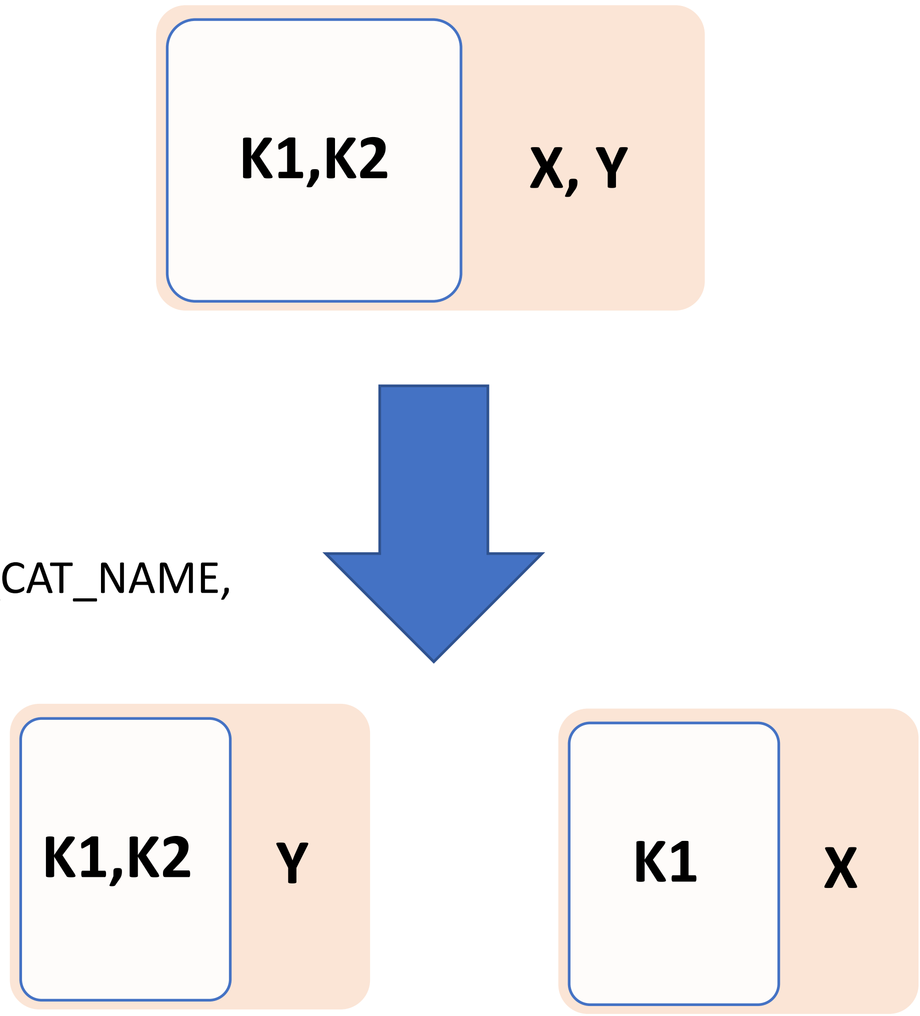
#BILL_ID, CLIENT_ID, CLIENT_NAME

#BILL_ID, #PRODUCT_ID, PRODUCT_NAME, PROD_CATE_ID, PROD_CAT_NAME, QUANTITY

# NF2

dependencies

- K1 -> X
- (K1, K2) -> Y total



#BILL_ID, #PRODUCT_ID, PRODUCT_NAME, PROD_CATE_ID, PROD_CAT_NAME, QUANTITY

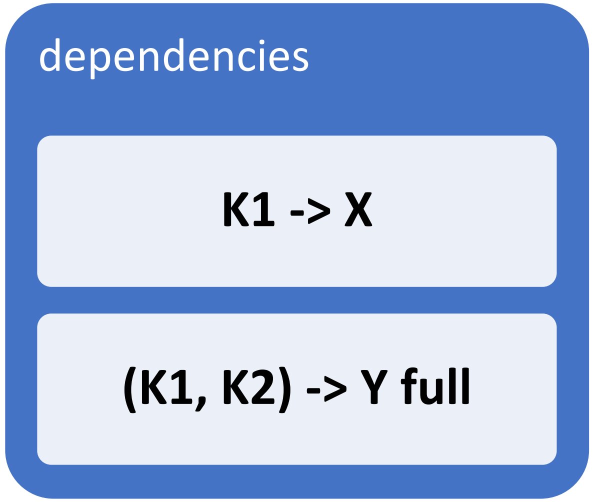
K1 = PRODUCT_ID
K2 = BILL_ID
Y = QTE
X = PRODUCT_NAME,PROD_CAT_ID,
PROD_CAT_NAME

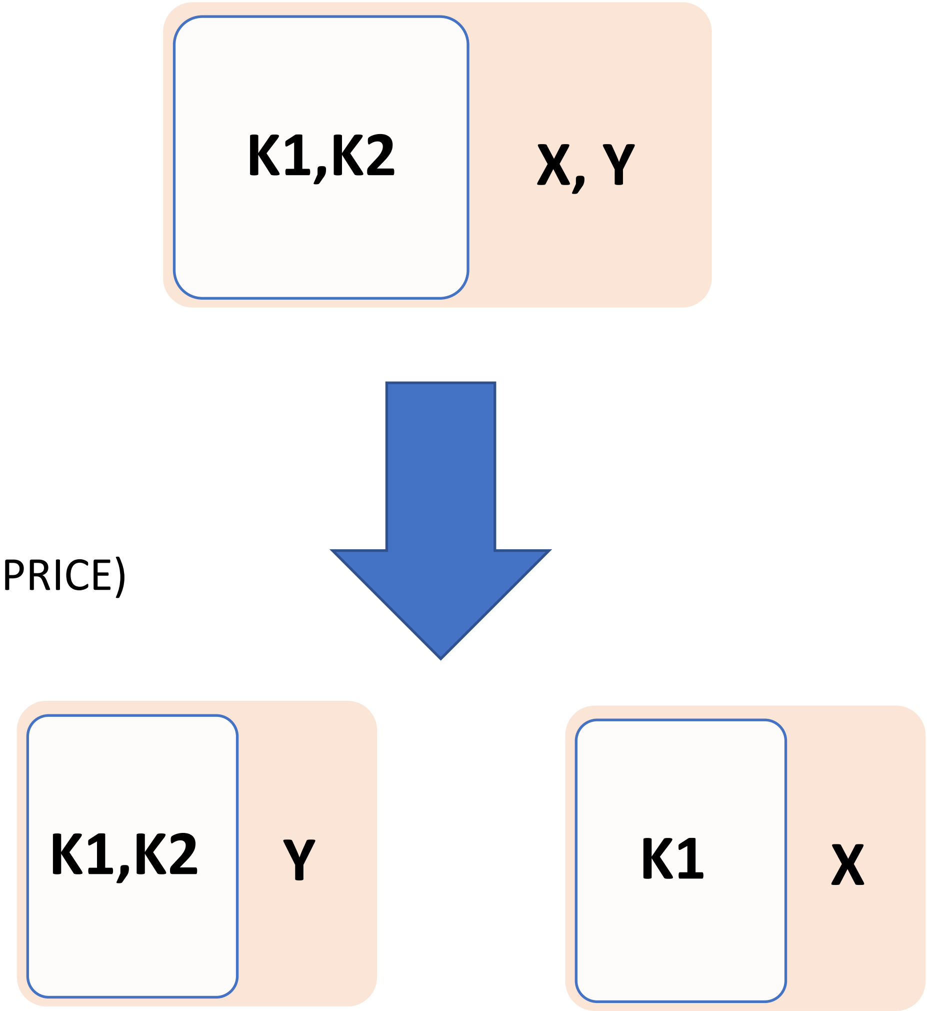
#BILL_ID, #PRODUCT_ID,QTE

#PRODUCT_ID,
PRODUCT_NAME,PROD_CATE_ID,
PROD_CAT_NAME

#BILL_ID, CLIENT_ID, CLIENT_NAME

# NF2





RENT(#CLIENT_ID, #RENT_DATE, CAR_ID, CAR_MODEL, DISCOUNT, PRICE)

RENT_DATE → DISCOUNT

CLIENT_ID, RENT_DATE → CAR_ID, CAR_MODEL, PRICE

CAR_MODEL -> PRICE

K1 = RENT_DATE

K2 = CLIENT_ID, RENT_DATE

Y = CAR_ID, CAR_MODEL, PRICE

X = DISCOUNT

#CLIENT_ID, #RENT_DATE, CAR_ID,
CAR_MODEL, PRICE

#RENT_DATE, DISCOUNT

# NF3

NO TRANSITIVE DEPENDENCIES

# NF3

- Tables in NF2
- Non-key attributes (not part of the key) depend on the entire key and only on the key.
- There are no transitive dependencies.

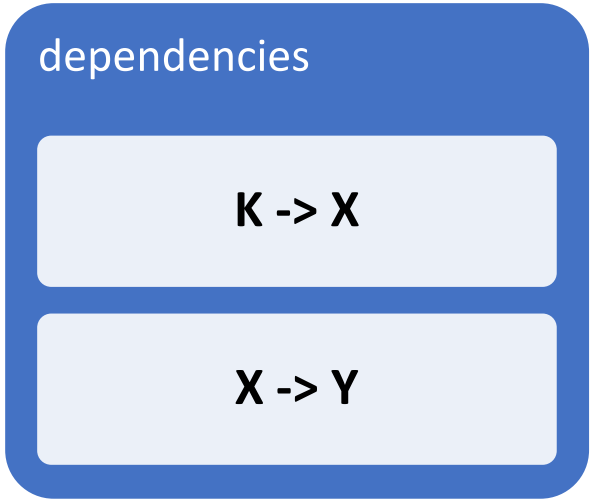
| AIRPORT_ID | AIRPLANE_ID | DEPARTURE | MODEL | CAPACITY | REVISION_DATE | BOARDING_GATE |
|---|---|---|---|---|---|---|
| 1 | 101 | 30/03/20 17:00 | Boeing 777 | 451 | 01/01/2021 | 42 |
| 1 | 102 | 02/05/20 09:30 | Airbus A320 | 150 | 01/03/2020 | 50 |
| 2 | 201 | 06/08/20 10:45 | Boeing 757 | 295 | 03/05/2020 | 35 |
| 2 | 202 | 10/10/20 06:20 | Airbus A320 | 150 | 04/06/2021 | 10 |
| 1 | 101 | 06/04/20 16:35 | Boeing 777 | 451 | 08/09/2020 | 23 |

| AIRPORT_ID | AIRPLANE_ID | DEPARTURE | BOARDING_GATE |
|---|---|---|---|
| 1 | 101 | 30/03/20 17:00 | 42 |
| 1 | 102 | 02/05/20 09:30 | 50 |
| 2 | 201 | 06/08/20 10:45 | 35 |
| 2 | 202 | 10/10/20 06:20 | 10 |
| 1 | 101 | 06/04/20 16:35 | 23 |

| AIRPLANE_ID | MODEL | CAPACITY | REVISION_DATE |
|---|---|---|---|
| 101 | Boeing 777 | 451 | 01/01/2021 |
| 102 | Airbus A320 | 150 | 01/03/2020 |
| 201 | Boeing 757 | 259 | 03/05/2020 |
| 202 | Airbus A320 | 150 | 04/06/2021 |

| AIRPLANE_ID | MODEL | CAPACITY | REVISION_DATE |
|---|---|---|---|
| 101 | Boeing 777 | 451 | 01/01/2021 |
| 102 | Airbus A320 | 150 | 01/03/2020 |
| 201 | Boeing 757 | 259 | 03/05/2020 |
| 202 | Airbus A320 | 150 | 04/06/2021 |

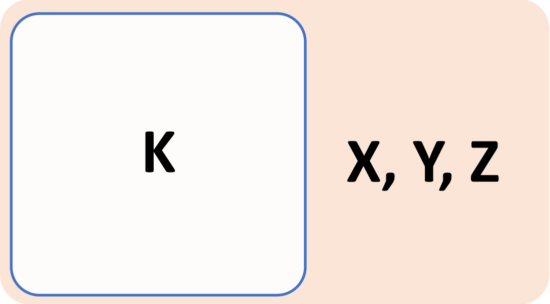
# NF3



dependencies

K -> X

X -> Y



K

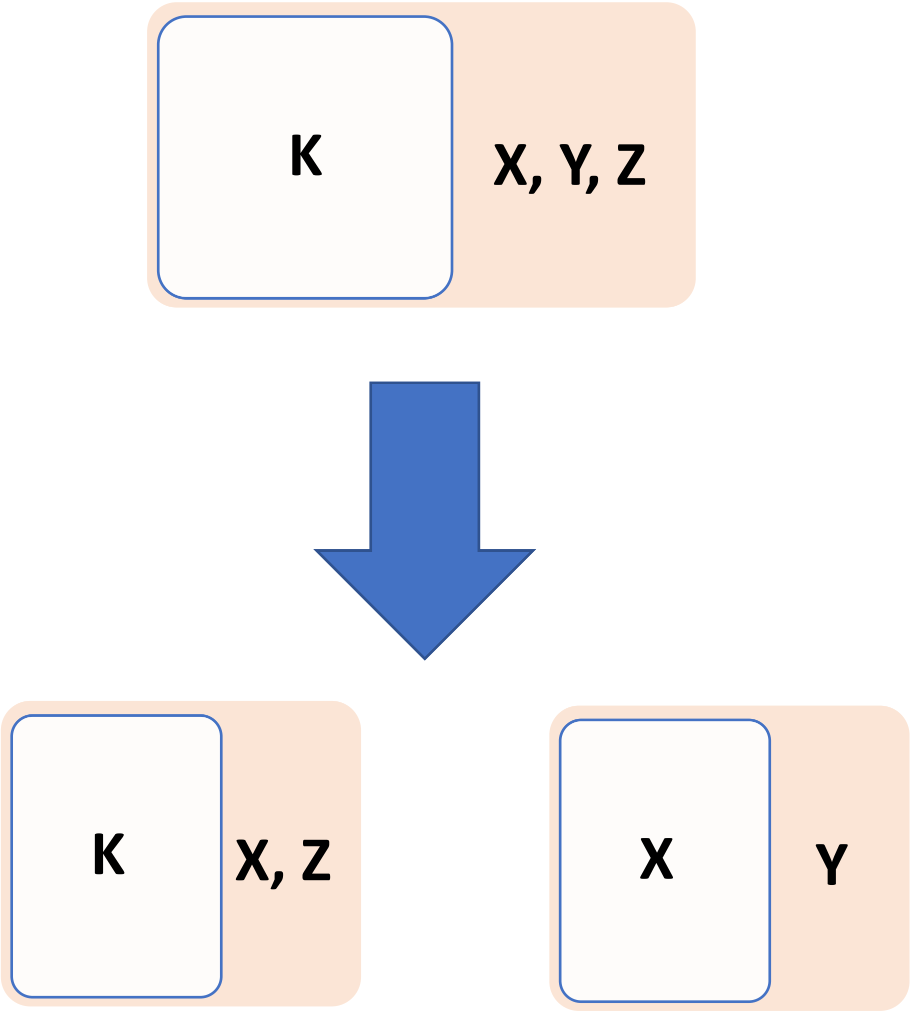
X, Y, Z

# NF3

dependencies

K -> X

X -> Y



K

X, Y, Z

K

X, Z

X

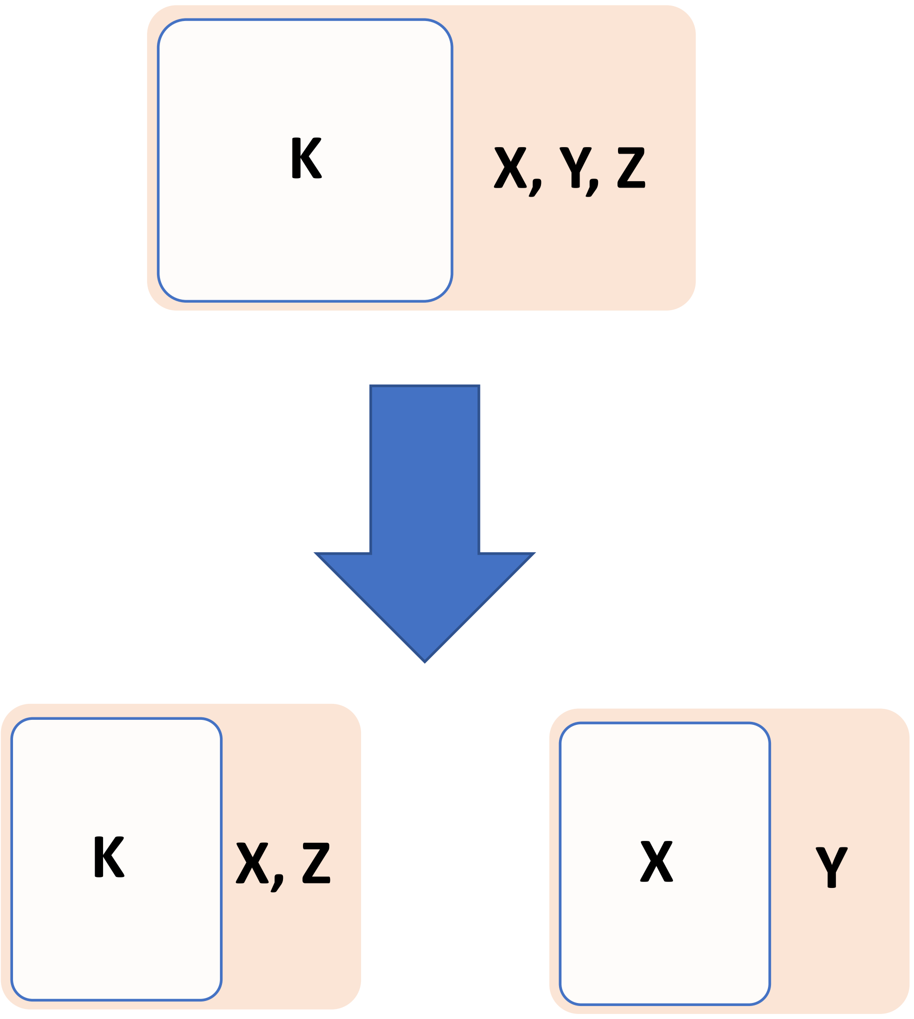
Y

# NF3

dependencies

K -> X

X -> Y



K = AIRPLANE_ID
X = AIRPLANE_MODEL
Y = CAPACITY
Z=  REVISION_DATE

| AIRPLANE_ID | MODEL | CAPACITY | REVISION_DATE |
|---|---|---|---|
| 101 | Boeing 777 | 451 | 01/01/2021 |
| 102 | Airbus A320 | 150 | 01/03/2020 |
| 201 | Boeing 757 | 259 | 03/05/2020 |
| 202 | Airbus A320 | 150 | 04/06/2021 |

| AIRPLANE_ID | REVISION_DATE |
|---|---|
| 101 | 01/01/2021 |
| 102 | 01/03/2020 |
| 201 | 03/05/2020 |
| 202 | 04/06/2021 |

| MODEL | CAPACITY |
|---|---|
| Boeing 777 | 451 |
| Airbus A320 | 150 |
| Boeing 757 | 259 |

# NF3

PAYMENTS(EMPLOYEE_ID, JOB_ID, SALARY, JOB_BONUS)

AWARD(YEAR, CATEGORY, WINNER, WINNER_NATIONALITY, CEREMONY_DATE, CEREMONY_LOCATION)

EXAMS(TEST_ID, STUDENT_ID, SUBJECT_ID, S_CREDITS, GRADE, CREDITS)

INSURANCE(CLIENT_ID, AUTO_ID, TYPE, PRICE, AUTO_NO, DATE, CLIENT_NAME)

# NF3

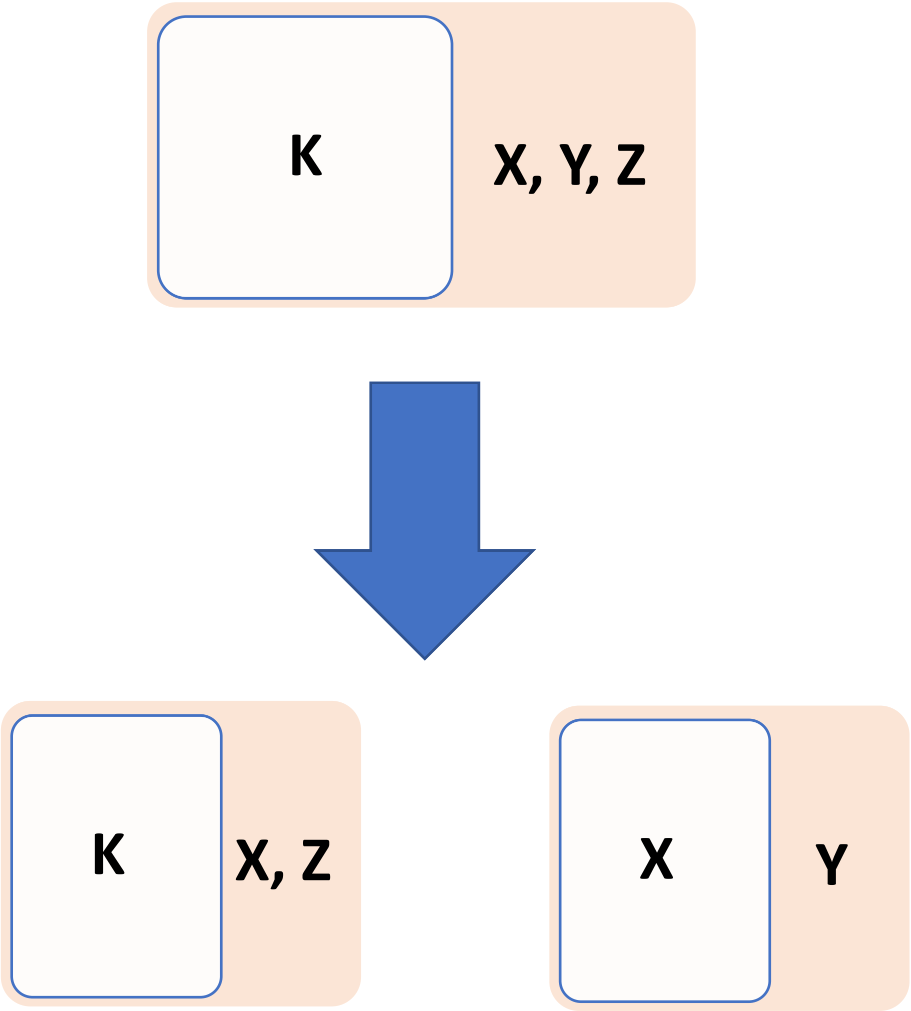
dependencies

K -> X

X -> Y

PAYMENTS(EMPLOYEE_ID, JOB_ID, SALARY, JOB_BONUS)
JOB_ID -> JOB_BONUS

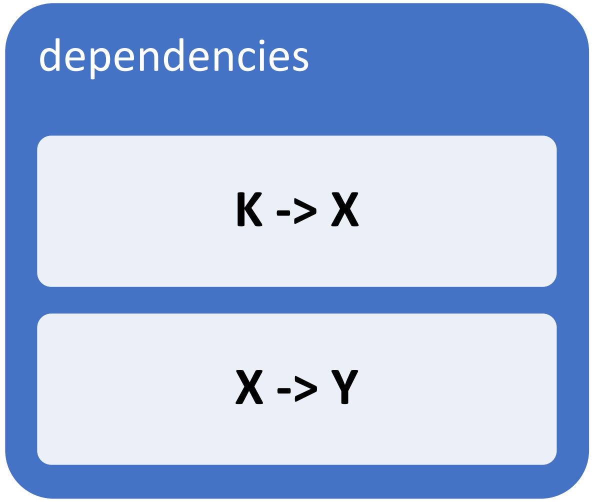
K = EMPLOYEE_ID
X = JOB_ID
Y = JOB_BONUS
Z= SALARY

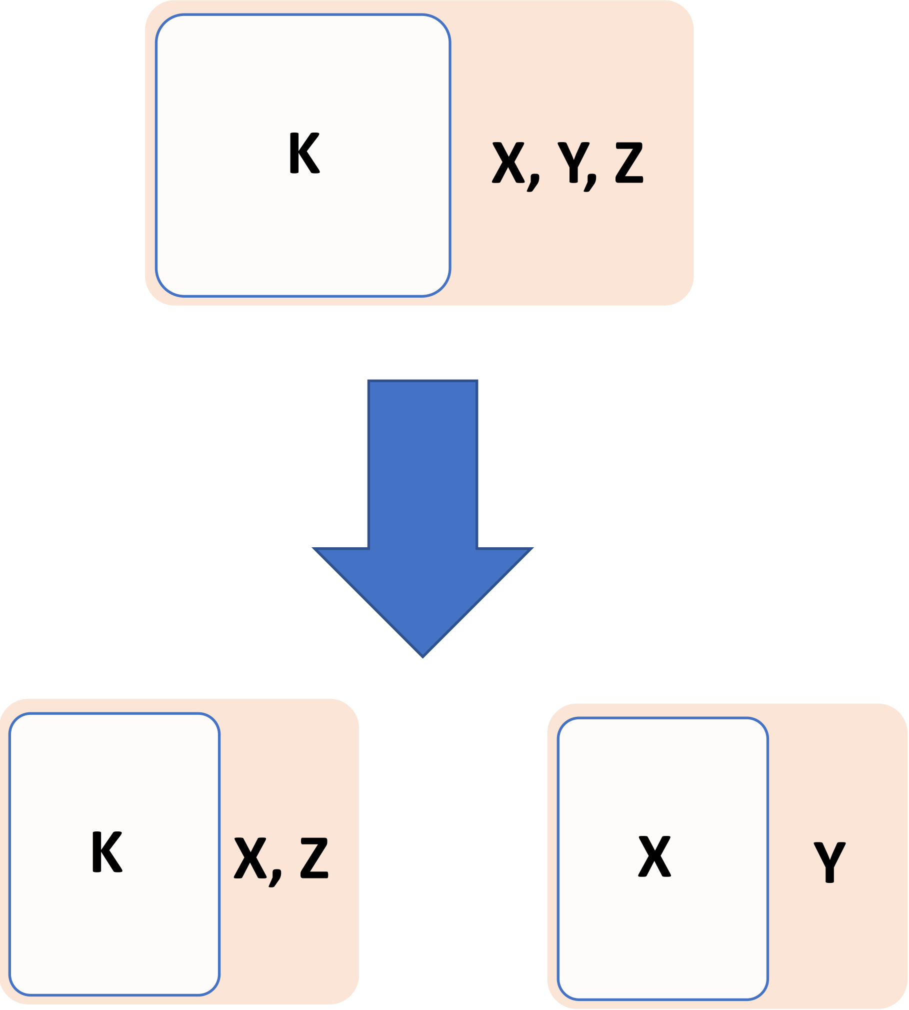


#EMPLOYEE_ID, JOB_ID, SALARY

JOB_ID, JOB_BONUS

# NF3





AWARD(#YEAR, #CATEGORY, WINNER, WINNER_NATIONALITY, CEREMONY_DATE, CEREMONY_LOCATION)

YEAR - > CEREMONY_DATE
YEAR -> CEREMONY_LOCATION
(YEAR, CATEGORY) -> WINNER ….

K =
X =
Y =
Z=

#

# NF3

dependencies

**K -> X**

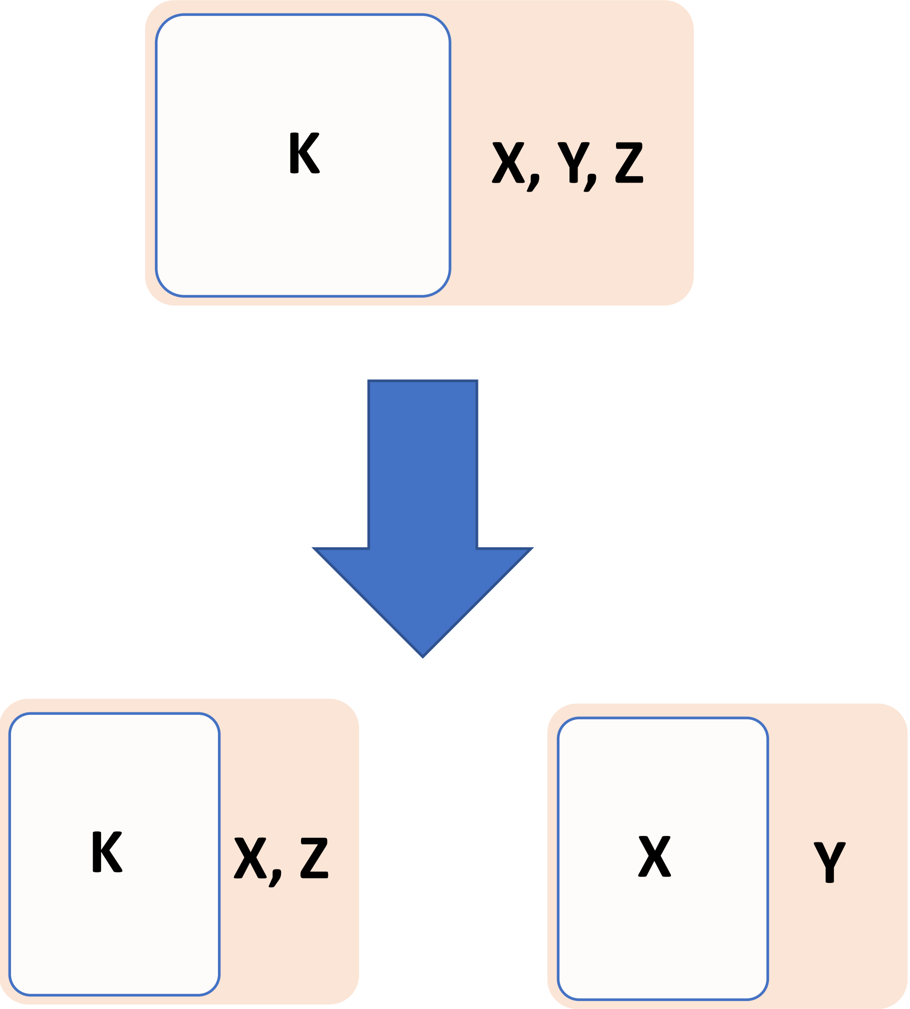
**X -> Y**

AWARD(#YEAR, #CATEGORY, WINNER, WINNER_NATIONALITY, CEREMONY_DATE, CEREMONY_LOCATION)

(YEAR, CATEGORY) -> WINNER -> WINNER_NATIONALITY

CEREMONY(#YEAR, CEREMONY_DATE, CEREMONY_LOCATION)
AWARD(#YEAR, #CATEGORY, WINNER, WINNER_NATIONALITY)

K = (YEAR, CATEGORY)
X = WINNER
Y = WINNER_NATIONALITY
Z=



(YEAR, CATEGORY, WINNER)

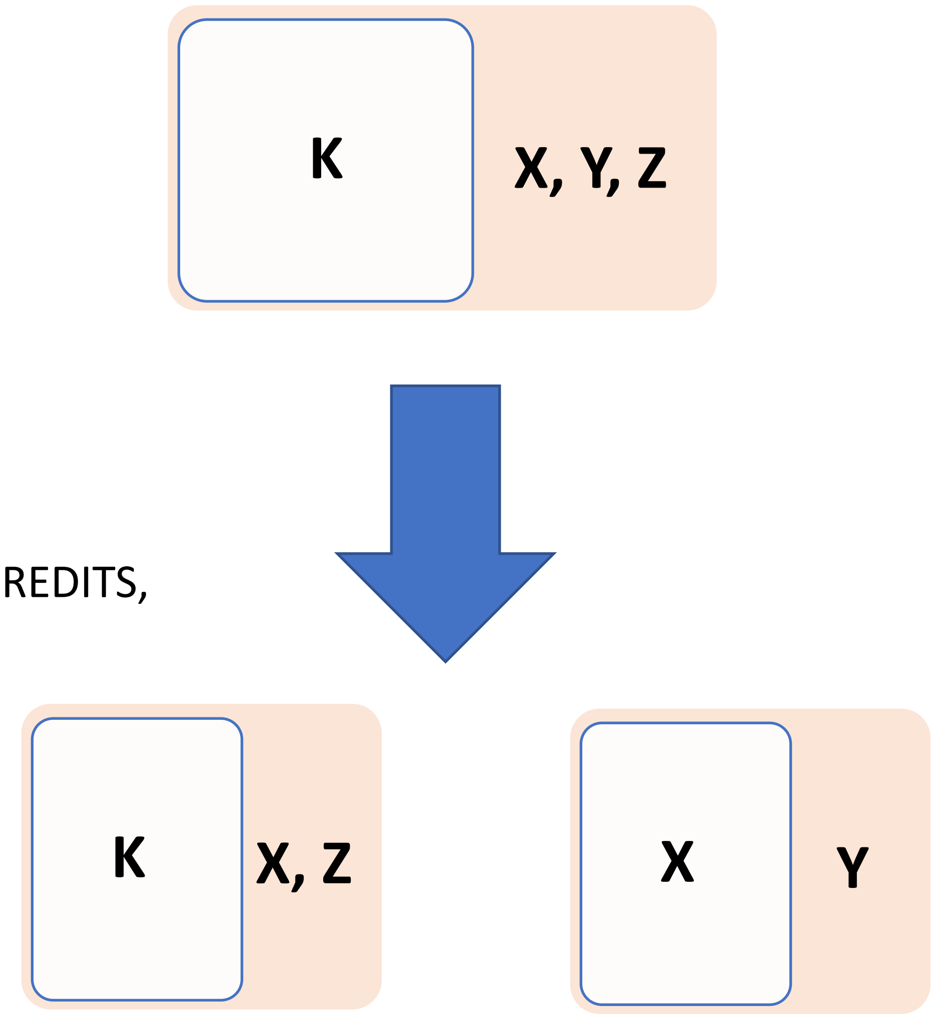
(WINNER,
WINNER_NATIONALITY )

# NF3

dependencies

K -> X

X -> Y



K

X, Y, Z

K

X, Z

X

Y

EXAMS( #TEST_ID, #STUDENT_ID, SUBJECT_ID, S_CREDITS, GRADE, CREDITS, DATE)

SUBJECT_ID -> S_CREDITS
TEST_ID -> SUBJECT_ID
#TEST_ID, #STUDENT_ID -> GRADE
#TEST_ID, #STUDENT_ID -> CREDITS
#TEST_ID, #STUDENT_ID -> SUBJECT_ID

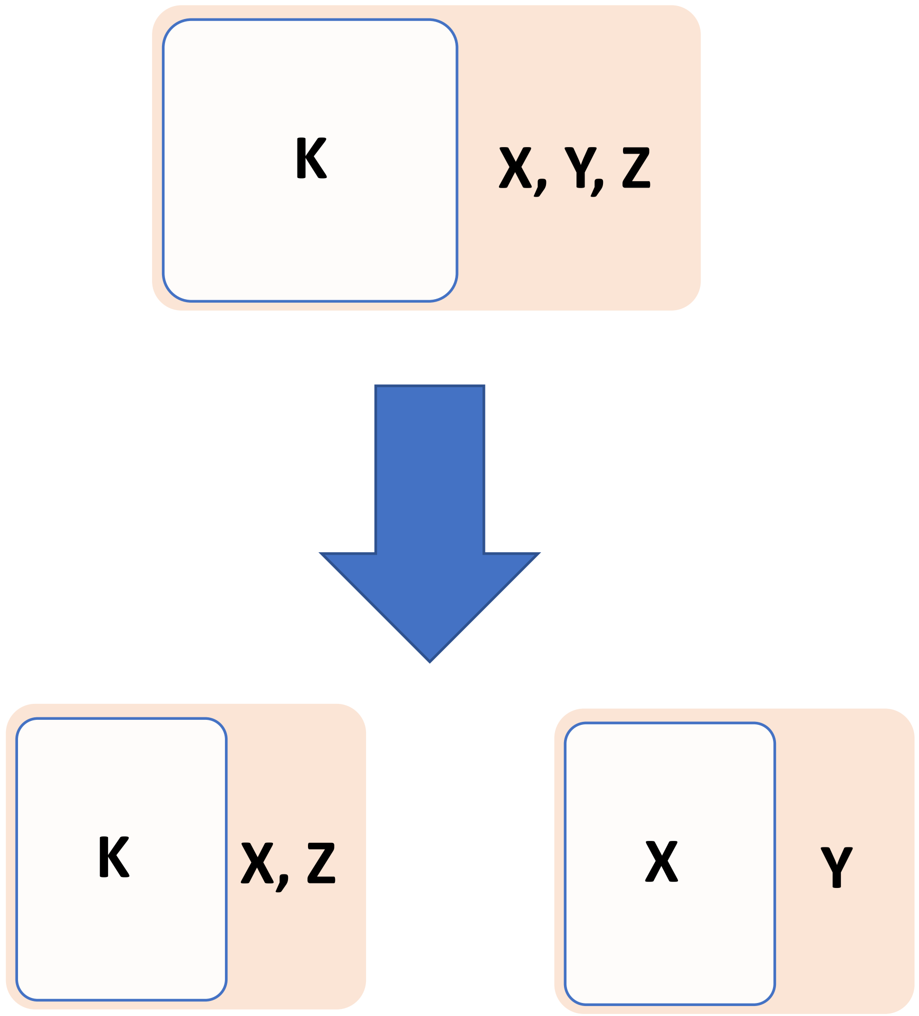
K =
X =
Y =
Z=

# NF3

dependencies

K -> X

X -> Y



(TEST_ID, SUBJECT_ID)

(SUBJECT_ID, S_CREDITS)

EXAMS( #TEST_ID, #STUDENT_ID, GRADE, CREDITS, DATE)
TEST(#TEST_ID, SUBJECT_ID, S_CREDITS)

SUBJECT_ID -> S_CREDITS TEST_ID -> SUBJECT_ID
#TEST_ID, #STUDENT_ID -> GRADE
#TEST_ID, #STUDENT_ID -> CREDITS
#TEST_ID, #STUDENT_ID -> SUBJECT_ID

K = TEST_ID
X = SUBJECT_ID
Y =
Z= S_CREDITS

# NF3

dependencies

**K -> X**

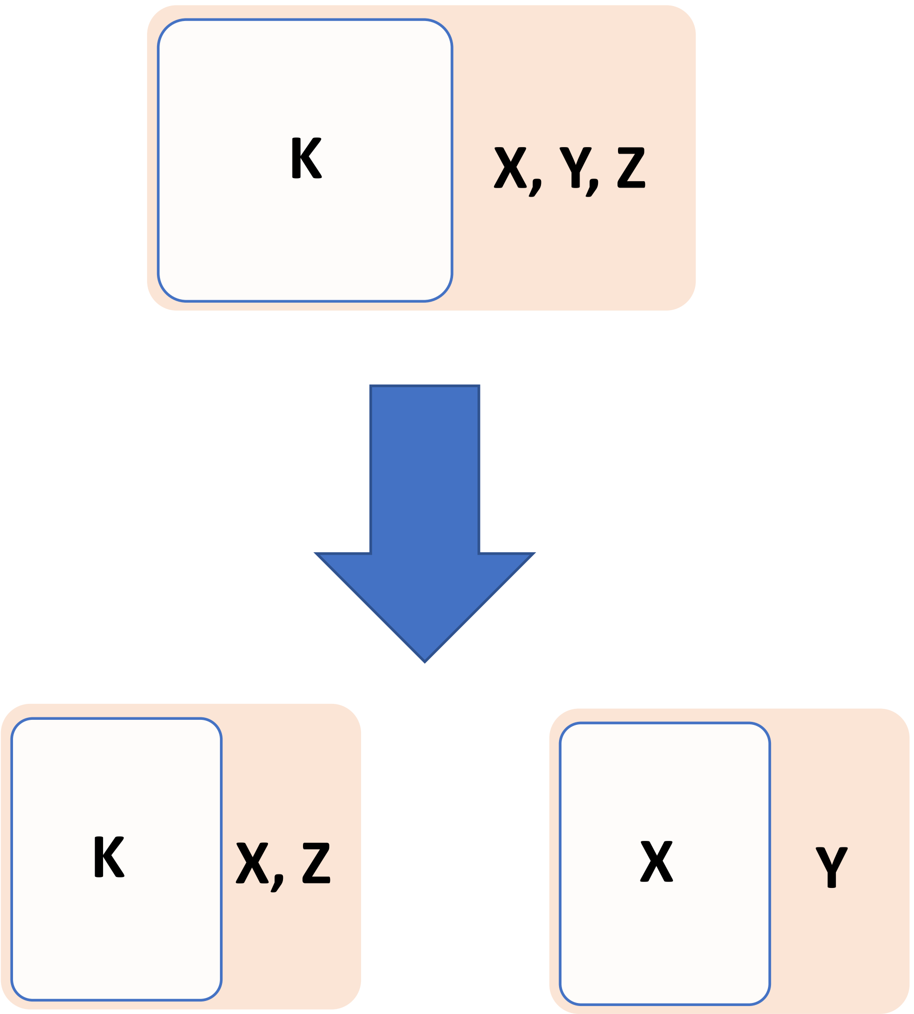
**X -> Y**

EXAMS( #TEST_ID, #STUDENT_ID, GRADE, CREDITS, DATE)

SUBJECT_ID -> S_CREDITS TEST_ID -> SUBJECT_ID

#TEST_ID, #STUDENT_ID -> GRADE
#TEST_ID, #STUDENT_ID -> CREDITS
#TEST_ID, #STUDENT_ID -> SUBJECT_ID

K = TEST_ID
X = SUBJECT_ID
Y =
Z= S_CREDITS



TEST(TEST_ID, SUBJECT_ID)

SUBJECT(SUBJECT_ID, S_CREDITS)

# BCNF

ALL DETERMINANT ARE CANDIDATE KEYS

# BCNF

- Tables in NF3
- For any functional dependency U → V, U is a candidate key.

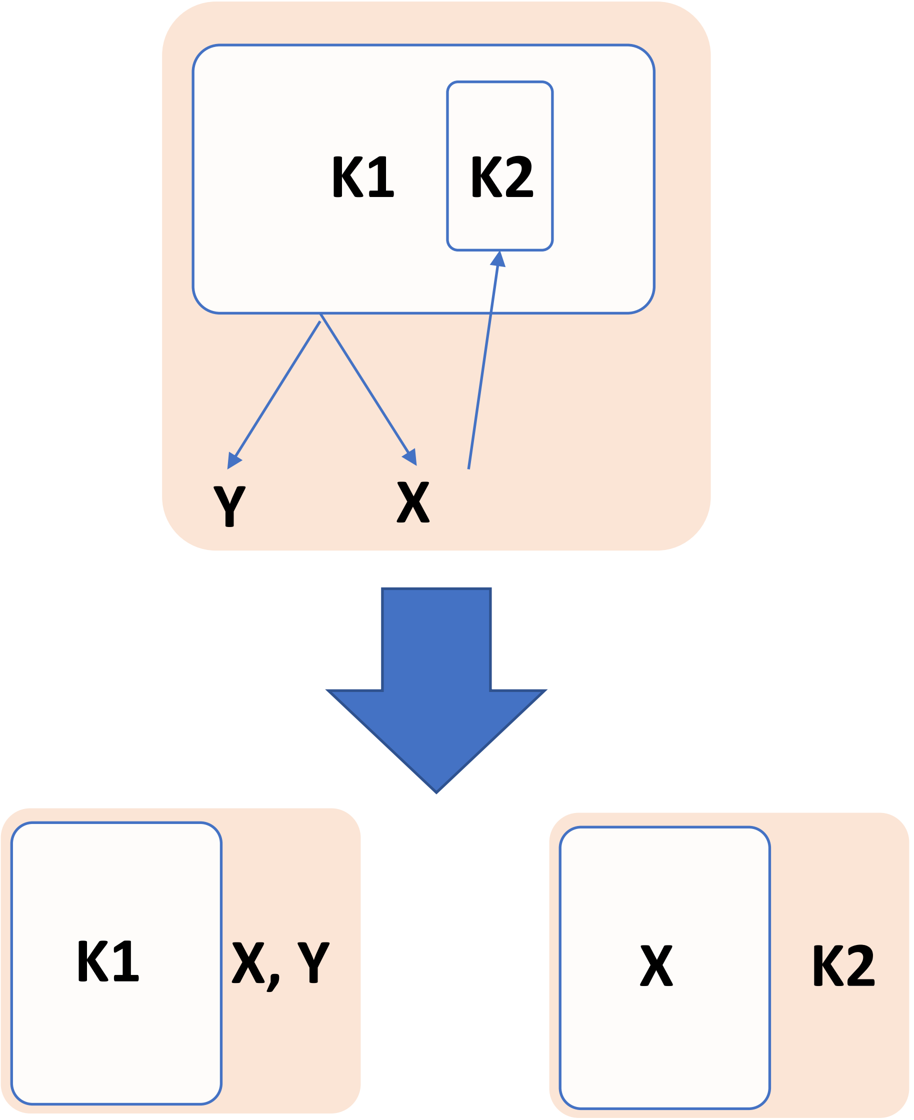
- All determinants are candidate keys.

# BCNF

dependencies

(K1,K2) -> X,Y

X -> K2



MANUFACTURING(CAR, STEP, STATION, TIME)
STATION -> STEP
(CAR, STEP) -> STATION
(CAR, STEP) -> TIME

K1 = CAR
K2 = STEP
X = STATION
Y= TIME

# BCNF

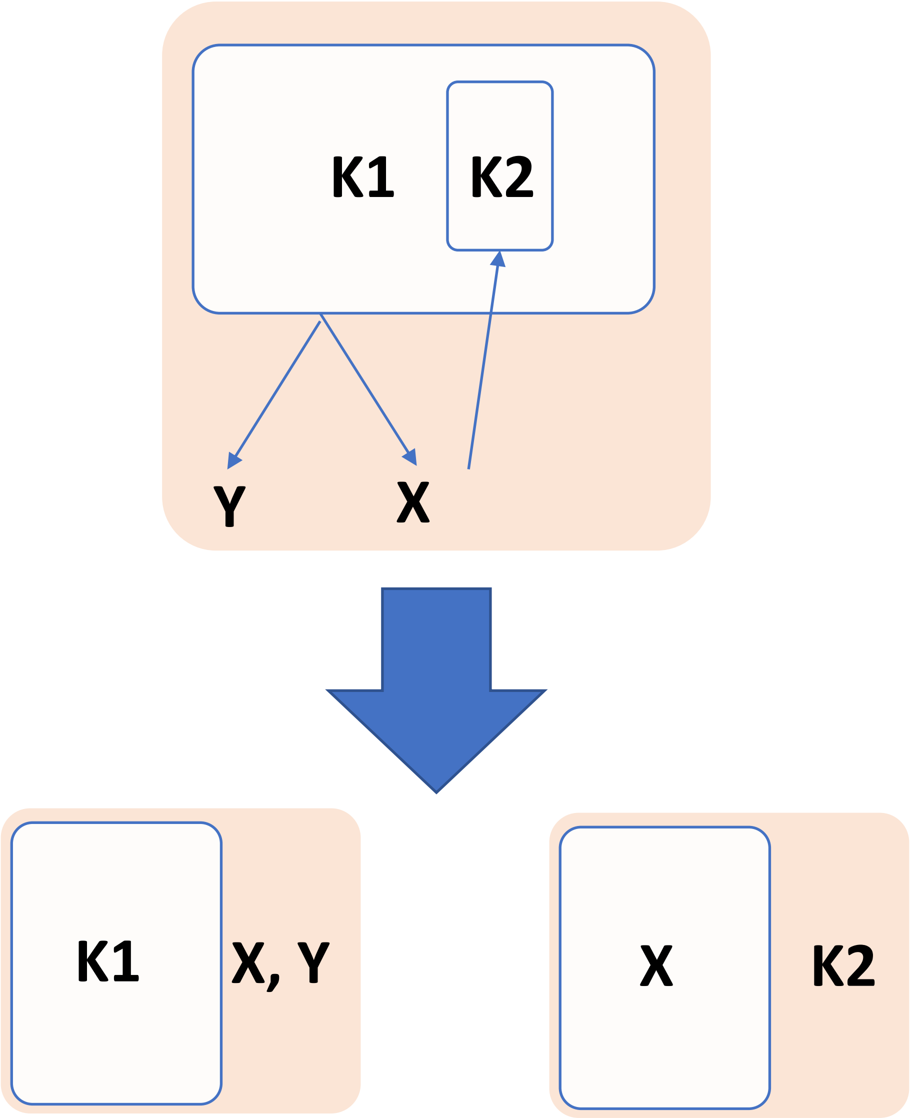
dependencies

**(K1,K2) -> X,Y**

**X -> K2**

MANUFACTURING(CAR, STEP, STATION, TIME)
STATION -> STEP
(CAR, STEP) -> STATION
(CAR, STEP) -> TIME

K1 = CAR
K2 = STEP
X = STATION
Y= TIME



(CAR, STATION, TIME)

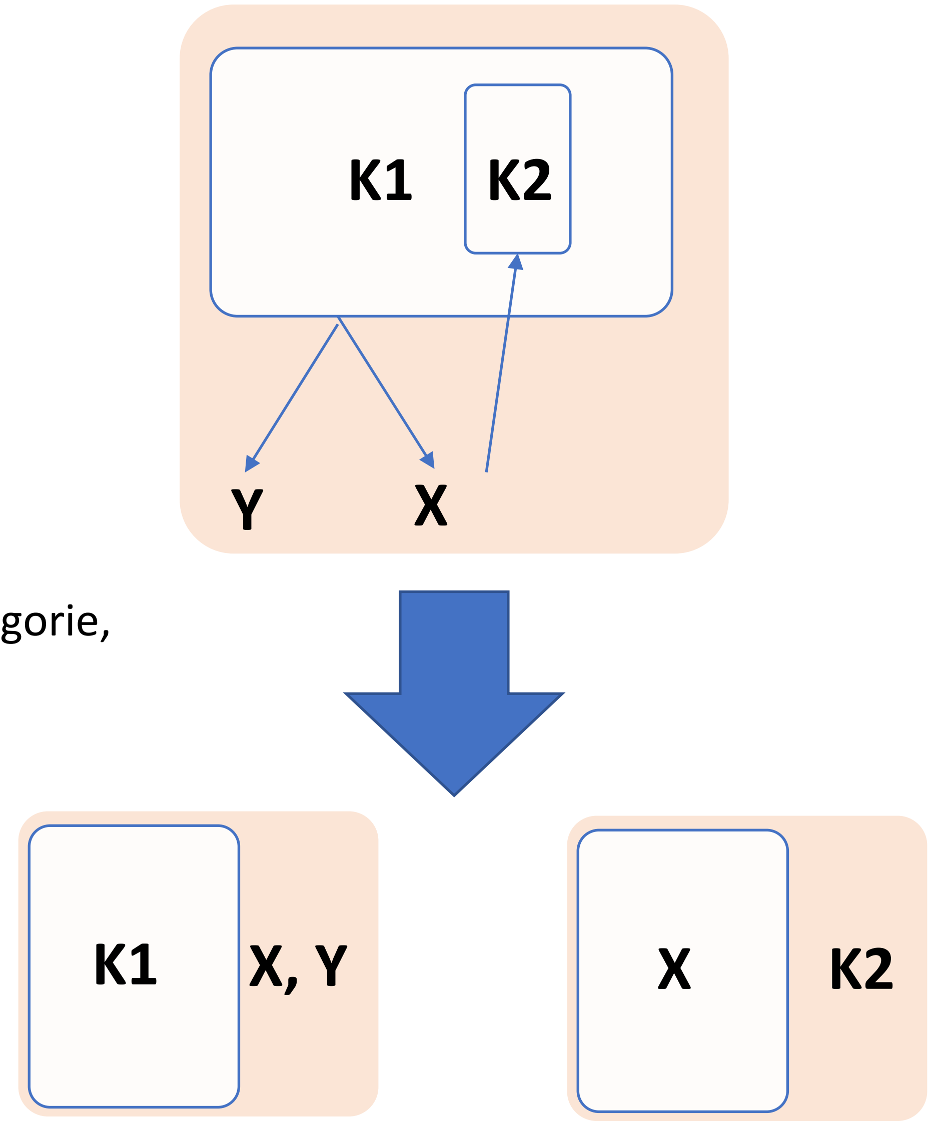
(STATION, STEP)

# BCNF

dependencies

**(K1,K2) -> X,Y**

**X -> K2**



(#USER, #PROD_CATEGORY, WIN_PROD)

--fiecare utilizator isi allege un produs preferat pentru o categorie,

--un utilizator alege un unic produs pentru o categorie

(USER, PROD_CATEGORY) -> WIN_PROD

WIN_PROD -> PROD_CATEGORY

K1 = USER
K2 = PROD_CATEGORY
X = WIN_PROD
Y=

(USER, WIN_PROD , !!!
PROD_CATEGORY)

(WIN_PROD,
PROD_CATEGORY)