

Normalization

Knowledge objectives

1. Distinguish cartesian product, relation and function terms in set theory terminology
2. Explain the goal of the relational normalization and how to reach it
3. Give the definition of NF^2 , 1NF, 2NF, 3NF, BCNF and 4NF
4. Explain the inclusion dependencies between different normal forms

Understanding objectives

1. Explain whether a functional dependency is true or not, given the extension of the relation and the semantics of the attributes
2. Explain whether a functional dependency is full or not, given the extension of the relation and the semantics of the attributes
3. Explain through an example the insertion, update and deletion anomalies that may appear in a relation
4. Justify in which normal form a relation is, given its candidate keys, an explanation of its contents and possibly an extension
5. Normalize a relation up to BCNF, given its functional dependencies and using the analysis algorithm
6. Explain through an example why sometimes it may be better to denormalize a relational schema

Application objectives

- Find all functional dependencies in a relation, given its schema and an explanation of its contents

Anomalies

Updating

Deleting

Inserting



Update anomaly

Supplying			
prov	item	quant	city
1	a1	100	BCN
1	a2	150	BCN
2	a1	200	MDR
2	a2	300	MDR
3	a2	100	MDR

Athens

Athens

Several tuples need to be updated because of only one change!

Deletion anomaly

Supplying			
prov	item	quant	city
1	a1	100	BCN
1	a2	150	BCN
2	a1	200	MDR
2	a2	300	MDR
3	a2	100	MDR

Data may be lost unintentionally!

Insertion anomaly

Supplying			
prov	Item	Quant	city
1	a1	100	BCN
1	a2	150	BCN
2	a1	200	MDR
2	a2	300	MDR
3	a2	100	MDR
4	NULL	NULL	Athens

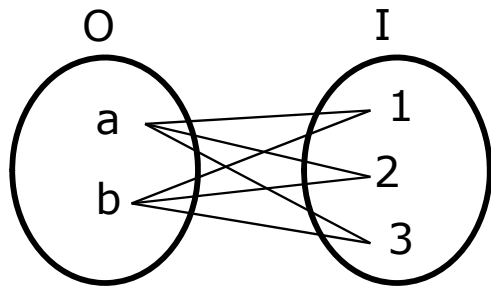
Data cannot be inserted independently!

Motivation

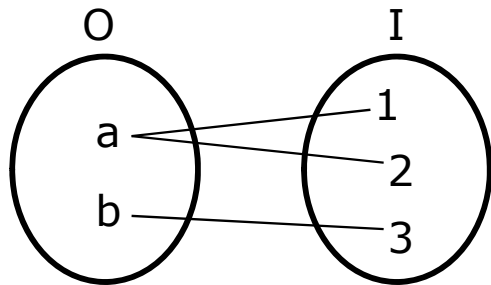
- Objective:
 - Formalize a set of simple ideas that guide a good database design
- Foundations:
 - Every relation (SQL table) must correspond to one semantic concept
 - Normalization theory allows us to recognize when this principle is not fulfilled

Functional dependencies

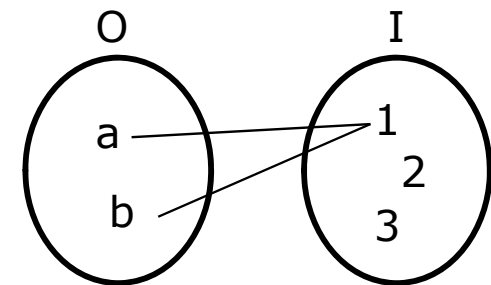
Set theory (I)



Cartesian product

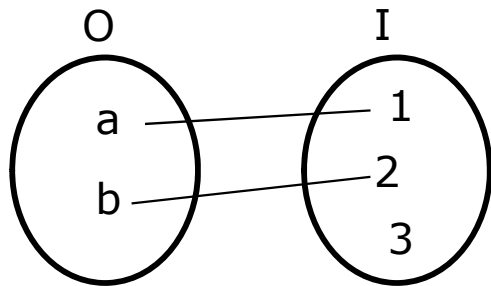


Relation
(subset of the cartesian product)



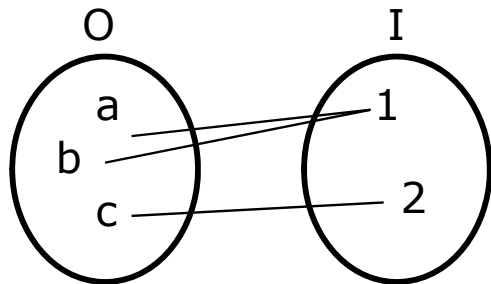
Function
(every origin at most one image)

Set theory (II)

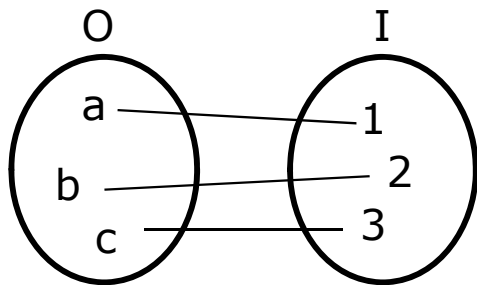


Injective function
(every image at most one origin)

PK are injective functions

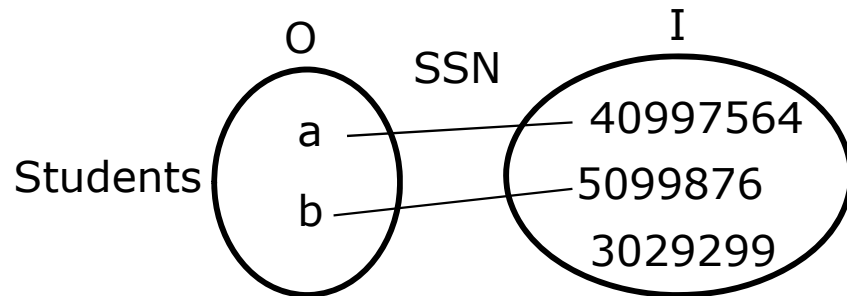


Exhaustive function
(every image at least one origin)

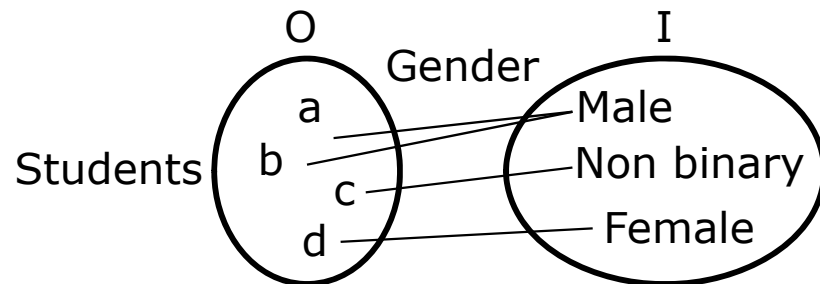


Bijective function
(every image exactly one origin)

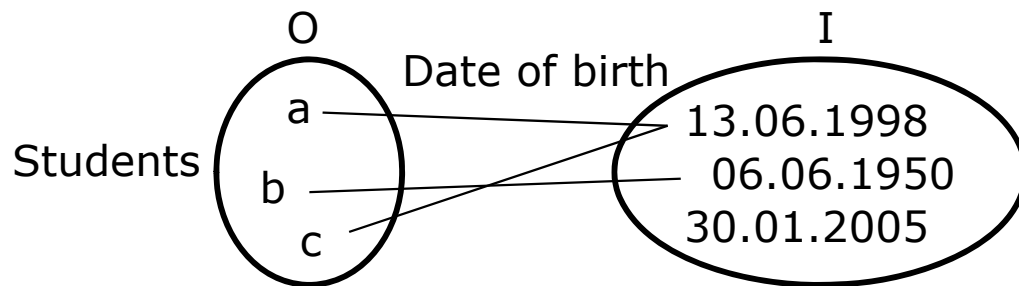
Set theory and databases



Injective function



Exhaustive function



It is not:
injective,
exhaustive,
bijective

Functional Dependencies

$$R (A_1, A_2, \dots, A_n)$$

- An FD $\{X\} \rightarrow \{Y\}$ (e.g., $\{SSN\} \rightarrow \{birthDate\}$, $\{provider\} \rightarrow \{city\}$) guarantees that given a value of $\{X\}$, this univocally determines the value of $\{Y\}$

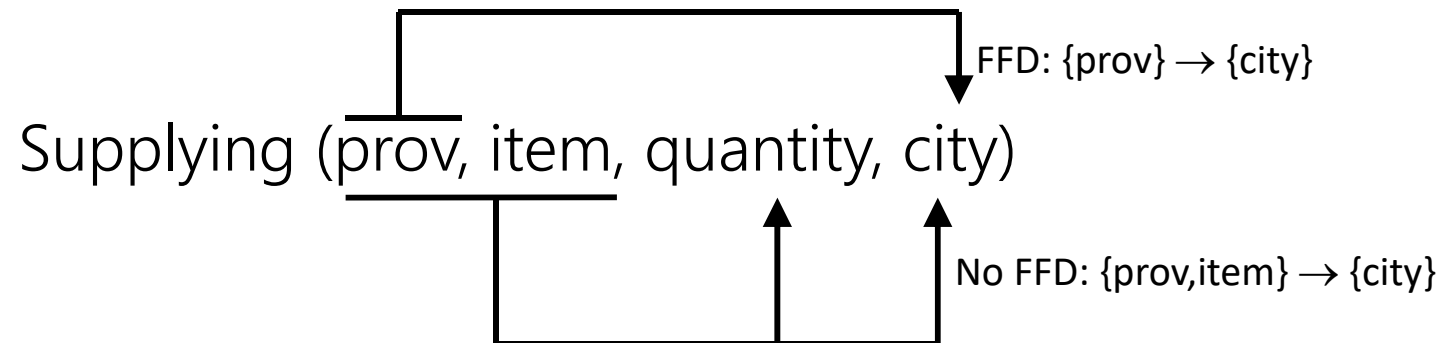
$$\forall s, t \in R, s[X] = t[X] \Rightarrow s[Y] = t[Y]$$

$\{X\}$ functionally determines $\{Y\}$

$\{Y\}$ functionally depends on $\{X\}$

Fully Functional Dependencies

- An FD $\{X\} \rightarrow \{Y\}$ is fully (FFD) iff there is no proper subset of $\{X\}$ which determines $\{Y\}$



Normal forms

NF²

1NF

2NF

3NF

BCNF

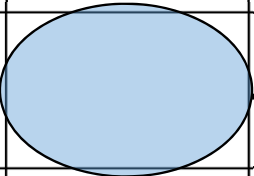


Overall ideas

- The main purpose is to eliminate redundancies and update anomalies
- A normal form restricts the set of dependencies in a relation
- Schemas in normal form are “good” schemas

First Normal Form - 1NF (I)

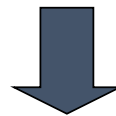
- A relation (SQL table) is in 1NF iff no attribute is itself a table; that is, every attribute value is atomic (non-breakable, non-aggregate and non-group)

	Attr1	Attr2	Attr3
tuple1			
tuple2			

First Normal Form - 1NF (II)

Pieces (#piece, description, proj_quantity)

100	{name: "screw", size: 0.5}	<table><tr><td>1</td><td>12</td></tr><tr><td>2</td><td>24</td></tr></table>	1	12	2	24
1	12					
2	24					
101	{name: "chair", size: 75}	<table><tr><td>1</td><td>4</td></tr><tr><td>3</td><td>22</td></tr></table>	1	4	3	22
1	4					
3	22					



Normalize (flatten)

Pieces (#piece, name, size, proj, quantity)

PK?


100	screw	0.5	1	12
100	screw	0.5	2	24
101	chair	75	1	4
101	chair	75	3	22

Second Normal Form – 2NF (I)

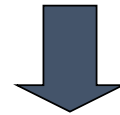
- A relation (SQL table) is in 2NF iff:
 - It is in 1NF
 - &
 - Every non-key attribute depends FFD on each of the candidate keys
- Exception: an attribute may functionally depend on a part of a candidate key if this attribute is part of another candidate key

Second Normal Form – 2NF (II)

(prov, item, quantity, provider_city)



A horizontal line is drawn under 'prov'. A vertical line descends from 'prov', a horizontal line extends to the right, and another vertical line ascends to 'provider_city', with an arrowhead pointing at 'provider_city'.



Normalize (split)

(prov, item, quantity)

⋮
↓ FK

(prov, provider_city)



2 semantic concepts

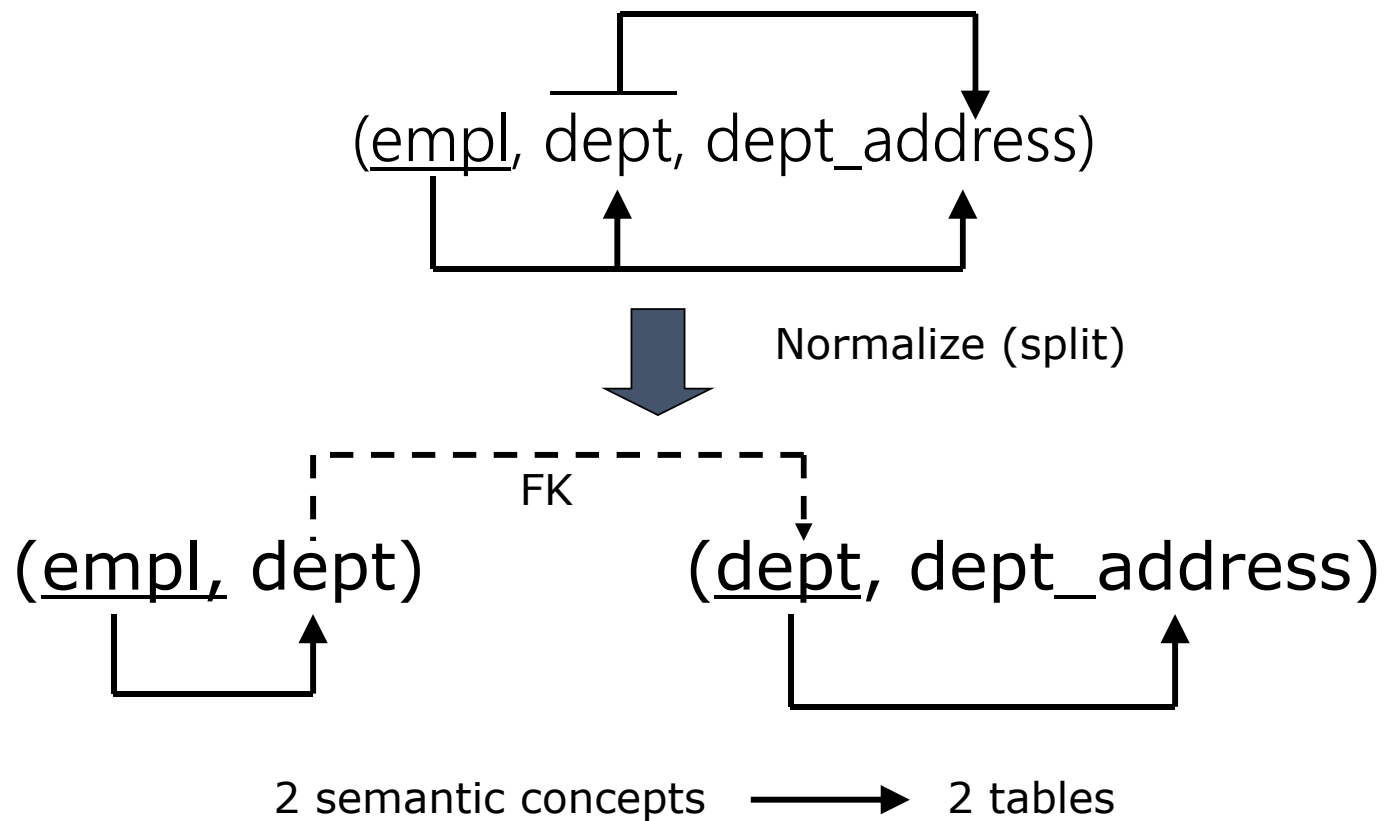


2 tables

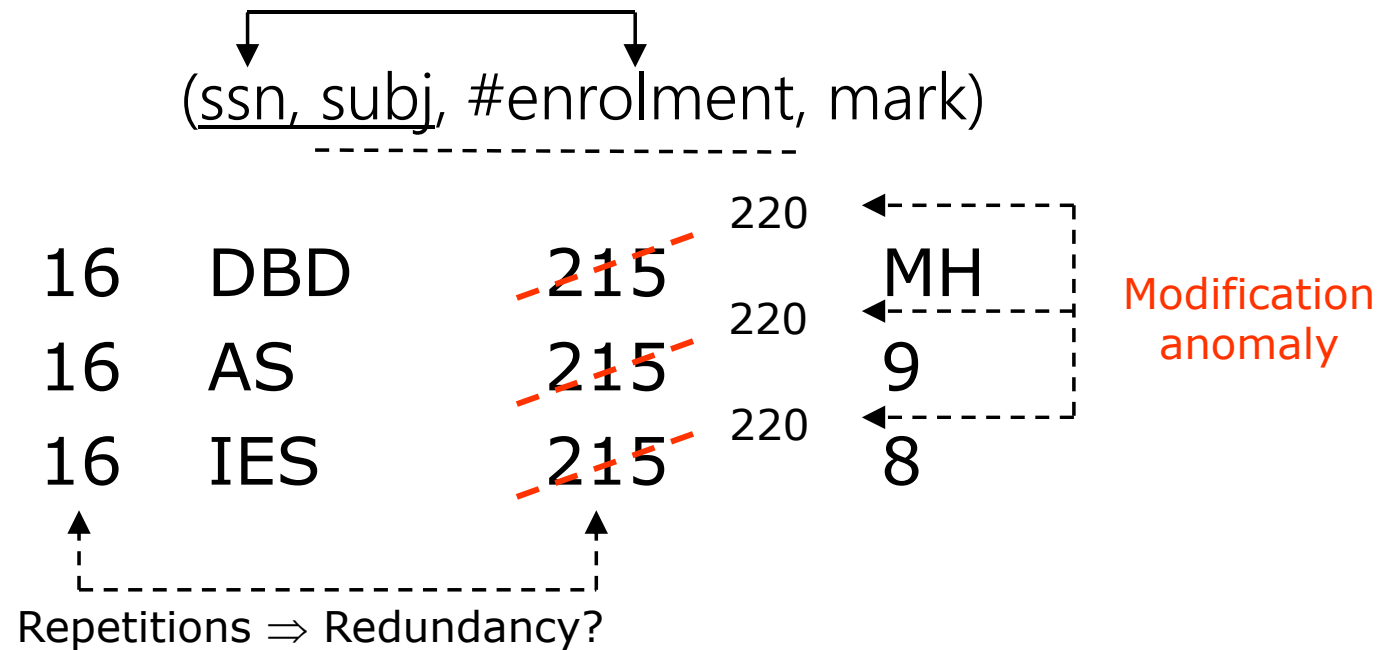
Third Normal Form - 3FN (I)

- A relation (SQL table) is in 3NF iff:
 - It is in 2NF
 - &
 - There is no non-key attribute functionally depending on something that is not a candidate key
- Exception: propagates that of 2NF

Third Normal Form - 3FN (II)



Boyce-Codd Normal Form – BCNF (I)



✓ 1NF?

✓ 2NF?

✓ 3NF?

Still, if #enrolment changes from 215 to 220, we have to modify three rows

Boyce-Codd Normal Form – BCNF (II)

- A relation (SQL table) is in BCNF (a.k.a. Strong 3NF) iff:
 - It is in 1NF
 - &
 - Each and every determinant (arrow tail) is a candidate key (either primary or alternative). That is, every determinant determines by itself all attributes in the relation (either directly or not)

Boyce-Codd Normal Form – BCNF (III)

(ssn, subj, #enrolment, mark)

Determinant

Is it candidate key?

ssn, subj

Yes

#enrolment, subj

Yes

ssn

No

#enrolment

No

Normalize (split)

(ssn, subj, mark)
(ssn, #enrolment)

(#enrolment, subj, mark)
(ssn, #enrolment)

(ssn, subj, mark)
(#enrolment, ssn)

(#enrolment, subj, mark)
(#enrolment, ssn)

Automatic normalization

Analysis

Synthesis

Armstrong rules

- Reflexivity

For all x , $x \rightarrow x$

- Augmentation

If $x \rightarrow y$ then $xz \rightarrow y$

- Projectability or Decomposition

If $x \rightarrow yz$ then $x \rightarrow y$ and $x \rightarrow z$

- Addition

If $x \rightarrow y$ and $x \rightarrow w$ then $x \rightarrow yw$

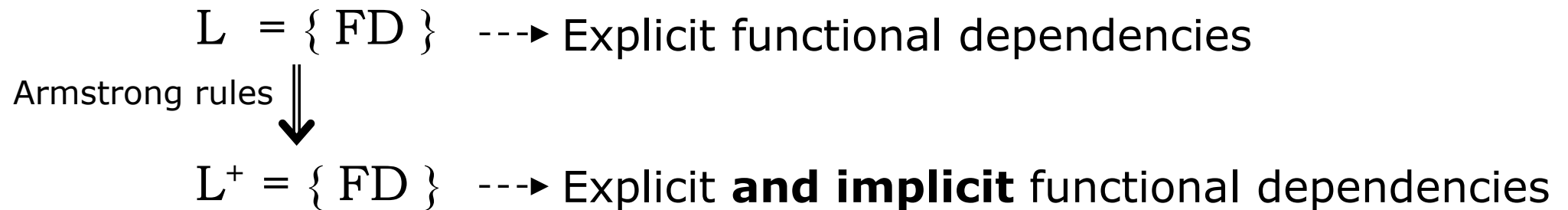
- Transitivity

If $x \rightarrow y$ and $y \rightarrow z$ then $x \rightarrow z$

- Pseudo-transitivity

If $x \rightarrow y$ and $yz \rightarrow w$ then $xz \rightarrow w$

Closure of dependencies



- What can be inferred from the closure?
 - Whether a functional dependency is true or not
 - Whether two relational schemas are equivalent or not
 - The whole set of determinants

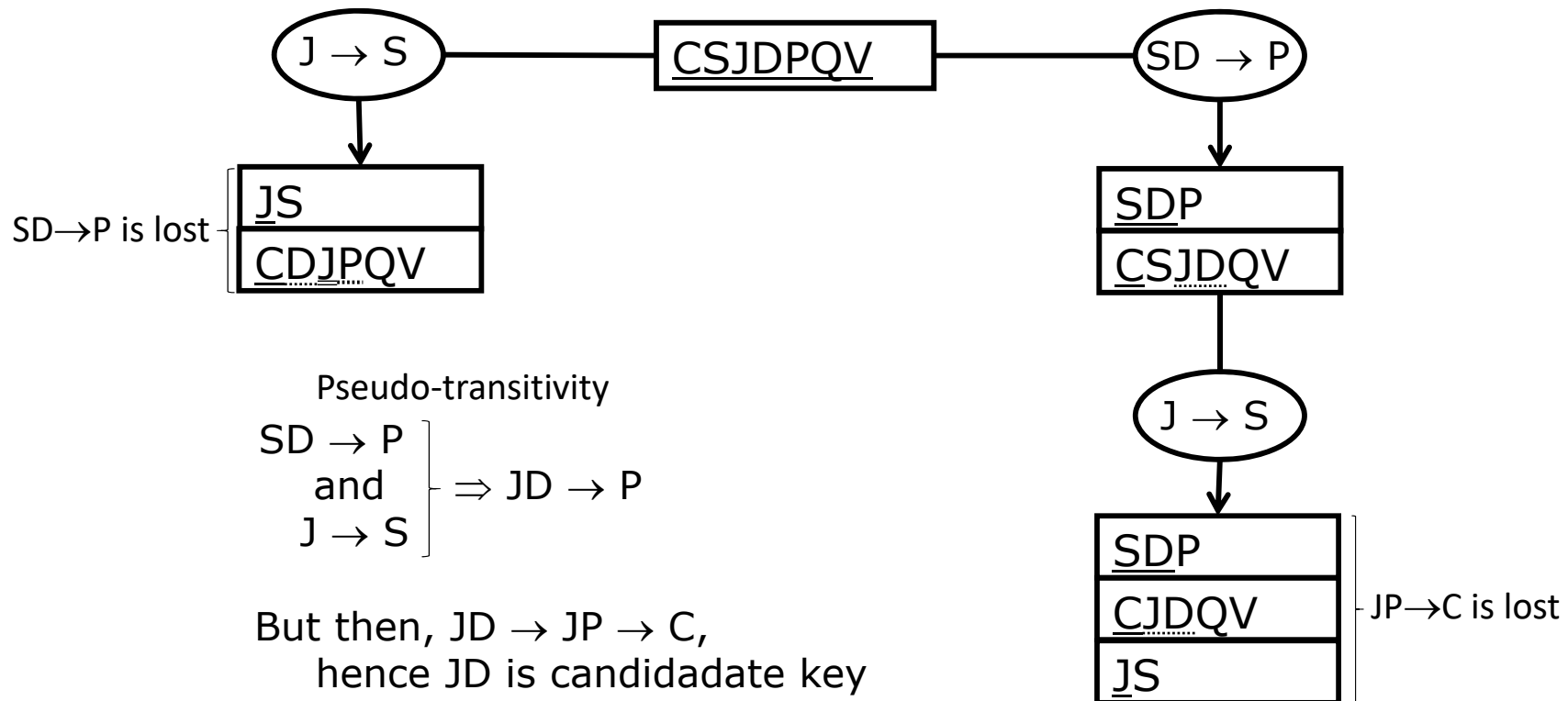
Analysis

- Input:
 - The “universal relation” containing all attributes in the database
 - The closure of functional dependencies (L^+)
 - This contains all the existing determinants in the database
- Algorithm:
 1. If relation R with attributes A is not in BCNF (i.e., exists a determinant that is not candidate key)
 1. Take $A_L \rightarrow A_R$ belonging to L^+ , with A_L and A_R being subsets of A, violating BCNF for R
 2. Split R into two relations with attribute sets: $A_L \cup A_R$ and $A - A_R$, respectively
 2. If either $A_L \cup A_R$ or $A - A_R$ is not in BCNF, go back to 1
- Considerations:
 - I. Some dependencies may be lost
$$\left. \begin{array}{l} R(\text{Professor, Hour, Course}) \\ L = \{C \rightarrow P, PH \rightarrow C\} \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} R1(\underline{C}, P) \\ R2(\underline{C}, H) \end{array} \right.$$
 - II. Decomposition may not be unique

Example of analysis

$R(\underline{C}, S, J, D, P, Q, V)$

$L = \{J \rightarrow S, SD \rightarrow P, JP \rightarrow C, C \rightarrow SJDPQV\}$



Examining normalization

Conclusions up to BCNF (strong 3NF)

- Any schema can always be normalized up to BCNF
 - Normalization is not unique
- Normalization can be fully automated (given a set of functional dependencies)
 - The normalized schema (in 3NF) is equivalent to that at the beginning (may not be true in BCNF)
- A normalized schema is better than a denormalized one because
 - a) Eliminates redundancies and, hence, anomalies
 - b) Separates semantically different concepts
 - It may be necessary to join them back to answer some queries!!!

Denormalizing

People (id, name, address, telephone, city, province)



BCNF



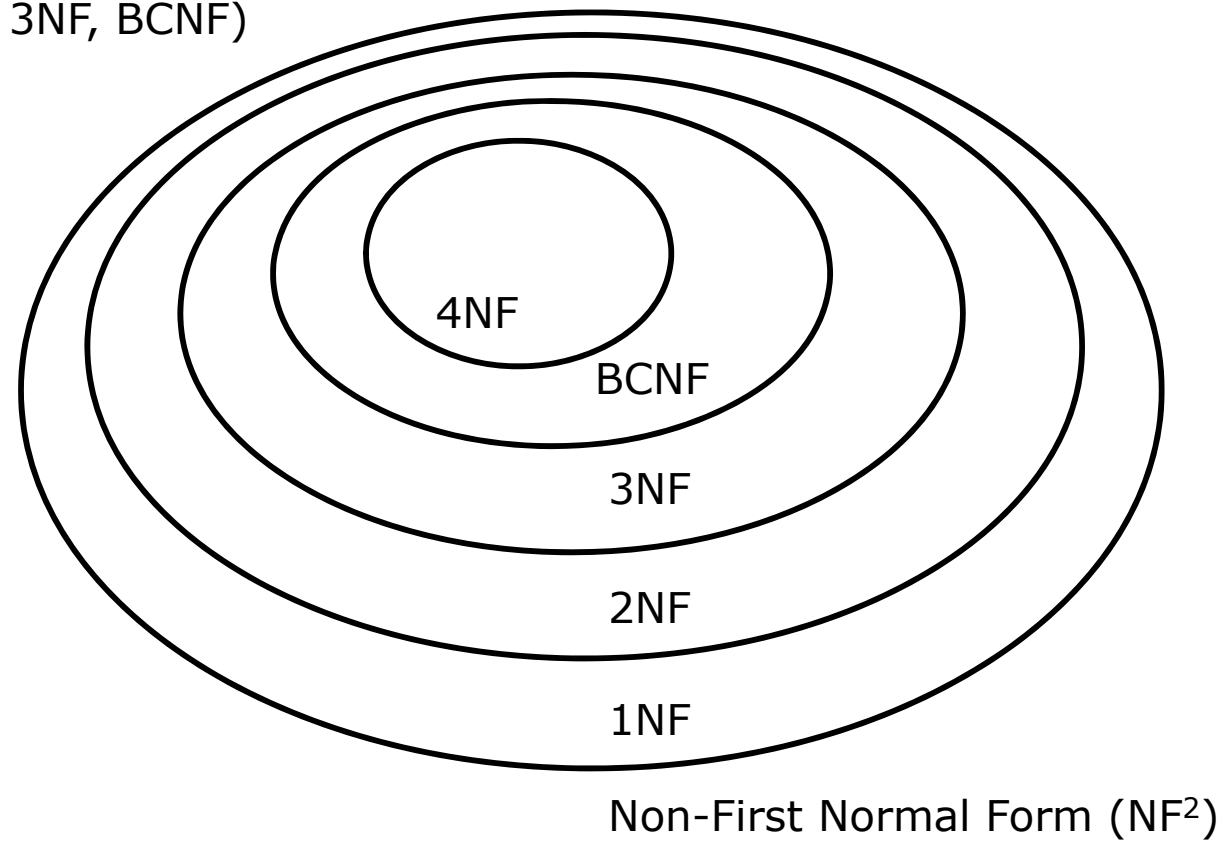
People(id, name, address, telephone, city)
Cities(city, province)

- We can denormalize ...
 - ... when otherwise the join would be performed too often
 - ... when changes are not expected or rare
 - ... when consistency is guaranteed by other means

Normal Forms Structure

Dependencies:

- Functional (1NF, 2NF, 3NF, BCNF)
- Multivalued (4NF)



4NF

Multivalued Dependencies



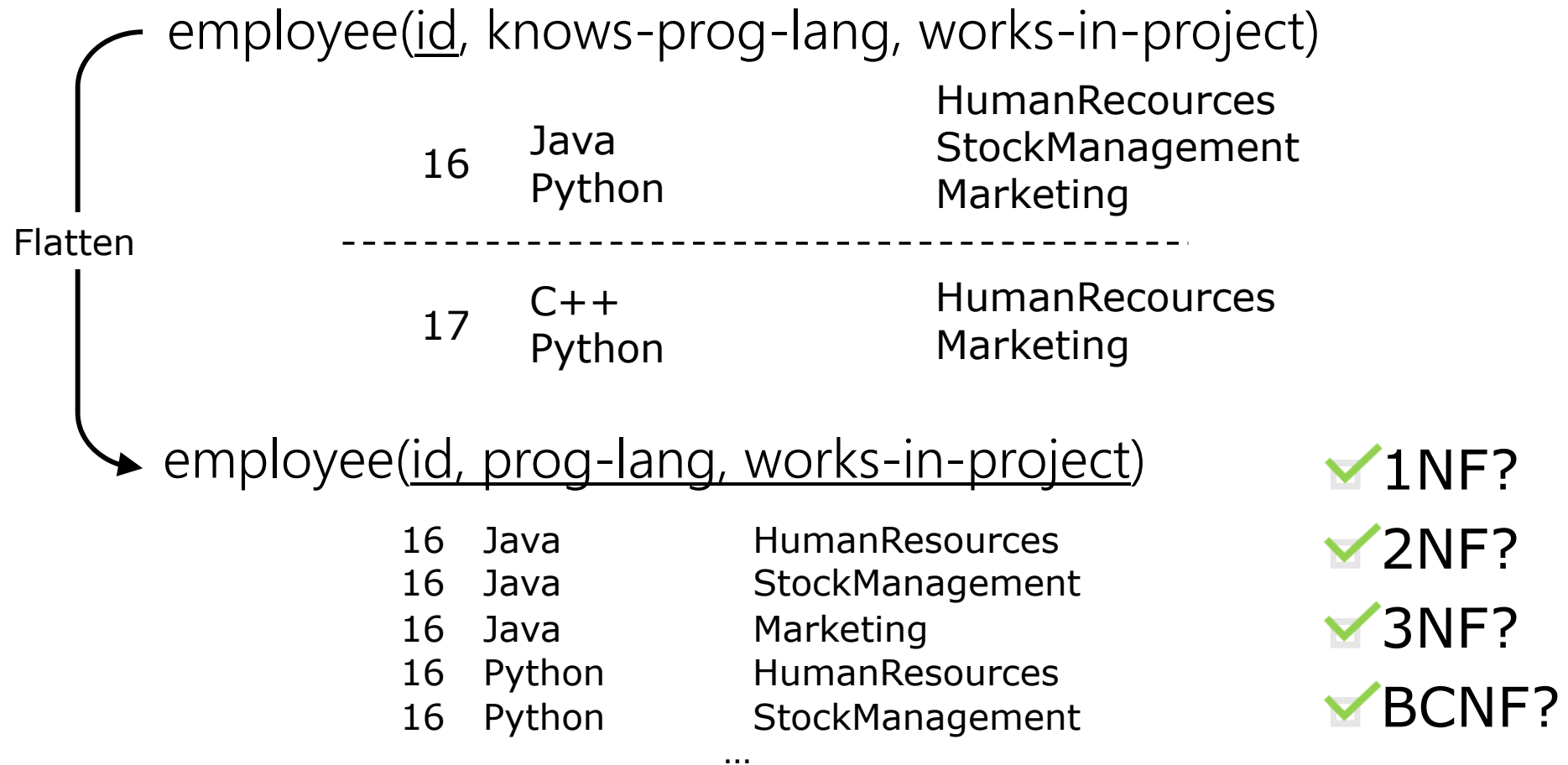
Multivalued Dependencies

$$R (A_1, A_2, \dots, A_n)$$

- An MVD $\{X\} \twoheadrightarrow \{Y\}$ (e.g., $\{\text{SSN}\} \twoheadrightarrow \{\text{spoken-language}\}$, $\{\text{team}\} \twoheadrightarrow \{\text{players}\}$) guarantees that given a value of $\{X\}$, this determines a set of values of $\{Y\}$

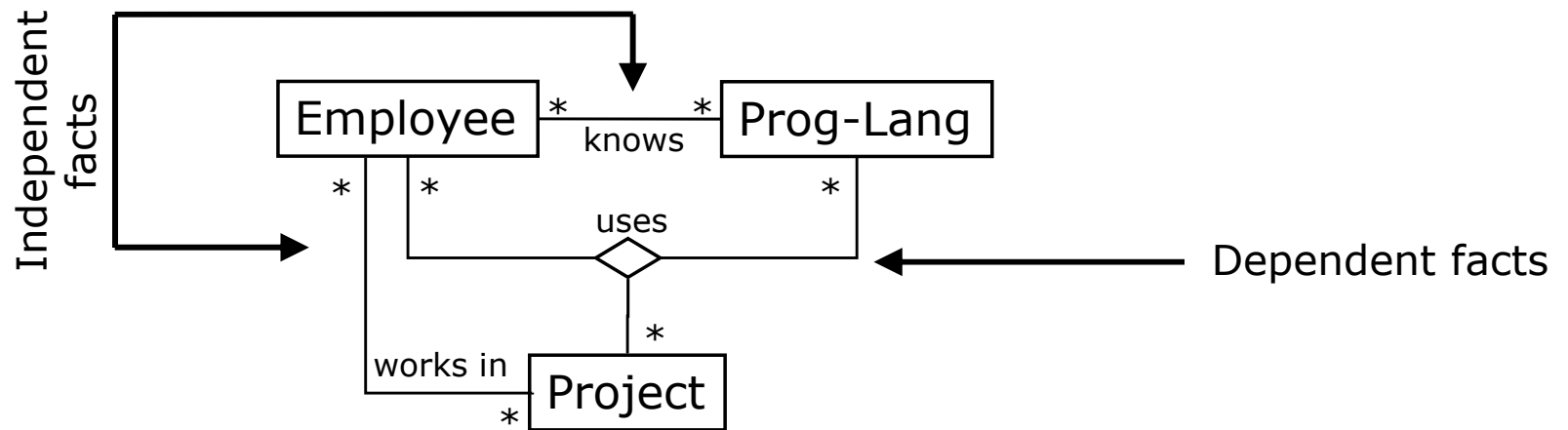
$$R = R[X \cup Y] \bowtie R[\{A_1, \dots, A_n\} - Y]$$

Fourth Normal Form - 4NF (I)

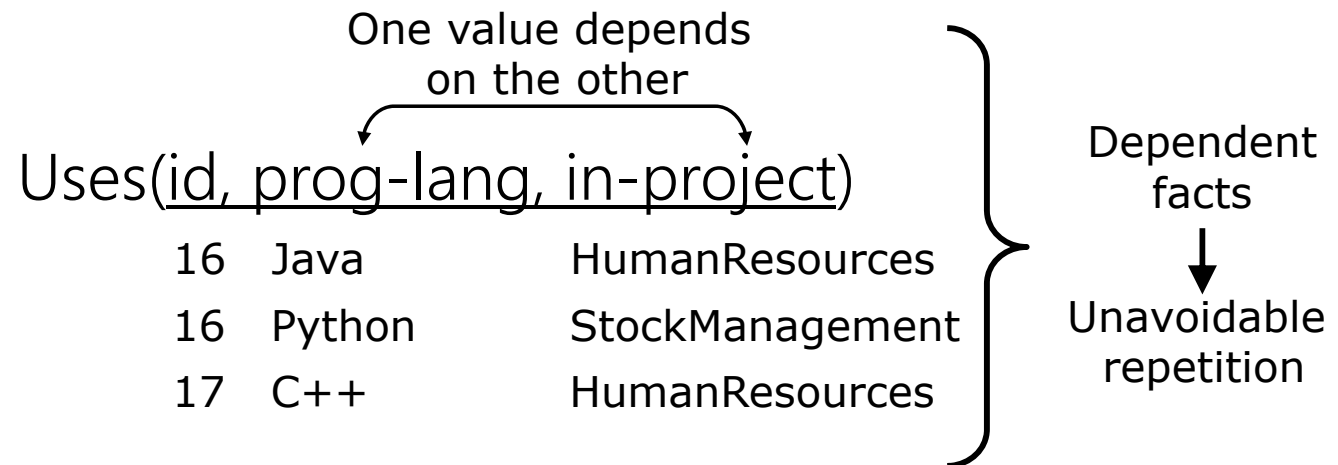
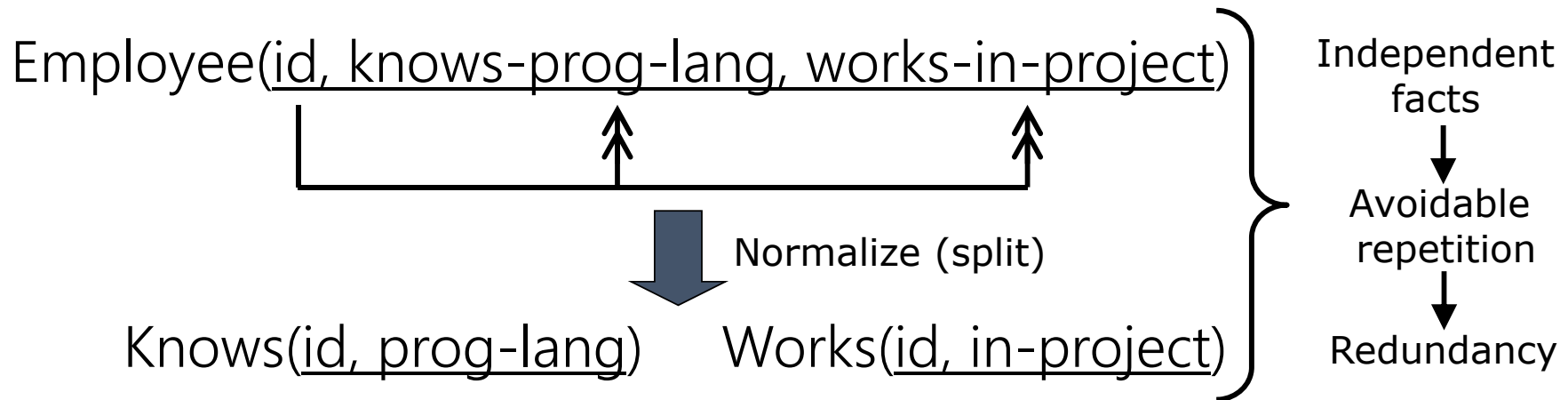


Fourth Normal Form - 4NF (II)

- A relation (SQL table) is in 4NF iff:
 - It is in BCNF
 - &
 - There are not independent multivalued dependencies



Fourth Normal Form - 4NF (III)



Closing



Summary

- Functional Dependencies
- Multivalued Dependencies
- Anomalies
 - Update
 - Delete
 - Insert
- Normal Forms:
 - 1NF (Codd '70)
 - 2NF (Codd '70)
 - 3NF (Codd '70)
 - BCNF (Boyce-Codd '74)
 - 4FN (Fagin '77)
- Automatic normalization using analysis algorithm
- Denormalization

Bibliography

- S. Abiteboul, R. Hull and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995
- Jaume Sistac et al. *Disseny de bases de dades*. Col·lecció Manuals, number 43. Editorial UOC, 2002
- R. Ramakrishnan and J. Gehrke. *Database Management Systems*, 3rd edition. McGraw-Hill, 2003
- T. Teorey et al. *Database modeling and design*, 4th edition. Morgan Kaufmann Publishers, 2006