Normalization (BCNF)

Knowledge Objectives

1. Remember the goal of the relational normalization and how to reach it
2. Remember the inclusion dependencies between different normal forms
3. Explain through an example why sometimes it may be better to denormalize a relational schema

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 INSERT, UPDATE and DELETE anomalies that may appear in a relation
4. Explain 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

Application Objectives

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

Updating 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

Updating 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

Updating 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!

Deleting 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

Deleting 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~~

Deleting 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~~

Elementary data may be lost unintentionally!

Inserting 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

Inserting 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

Inserting anomaly

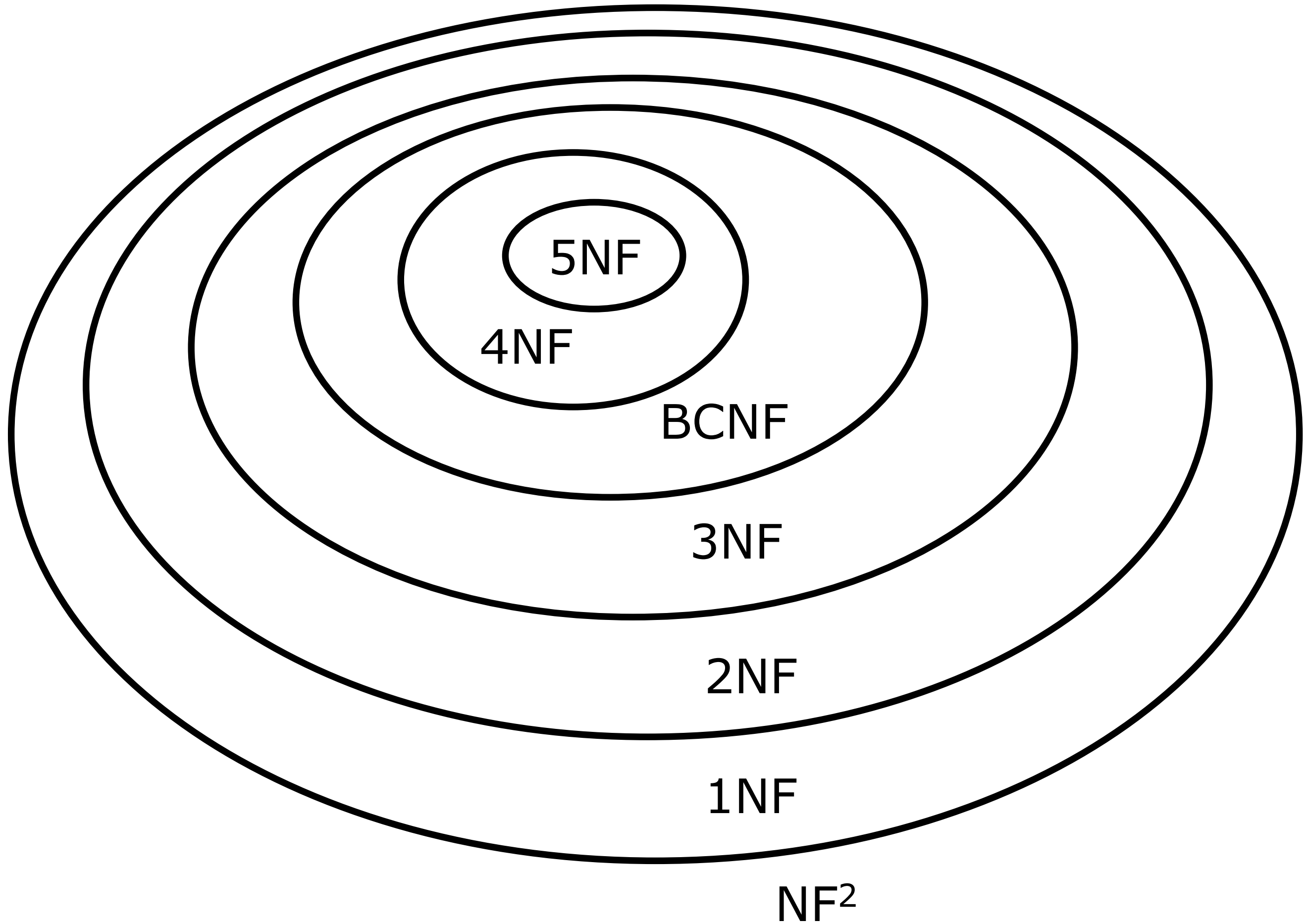
Supplying			
prov	Item	Quant	city
1	a1	100	BCN
1	a2	150	BCN
2	a1	200	MDR
2	a2	300	MDR
3	a2	100	MDR
~~4~~	~~NULL~~	~~NULL~~	~~Athens~~

Elementary data cannot be inserted independently!

Motivation

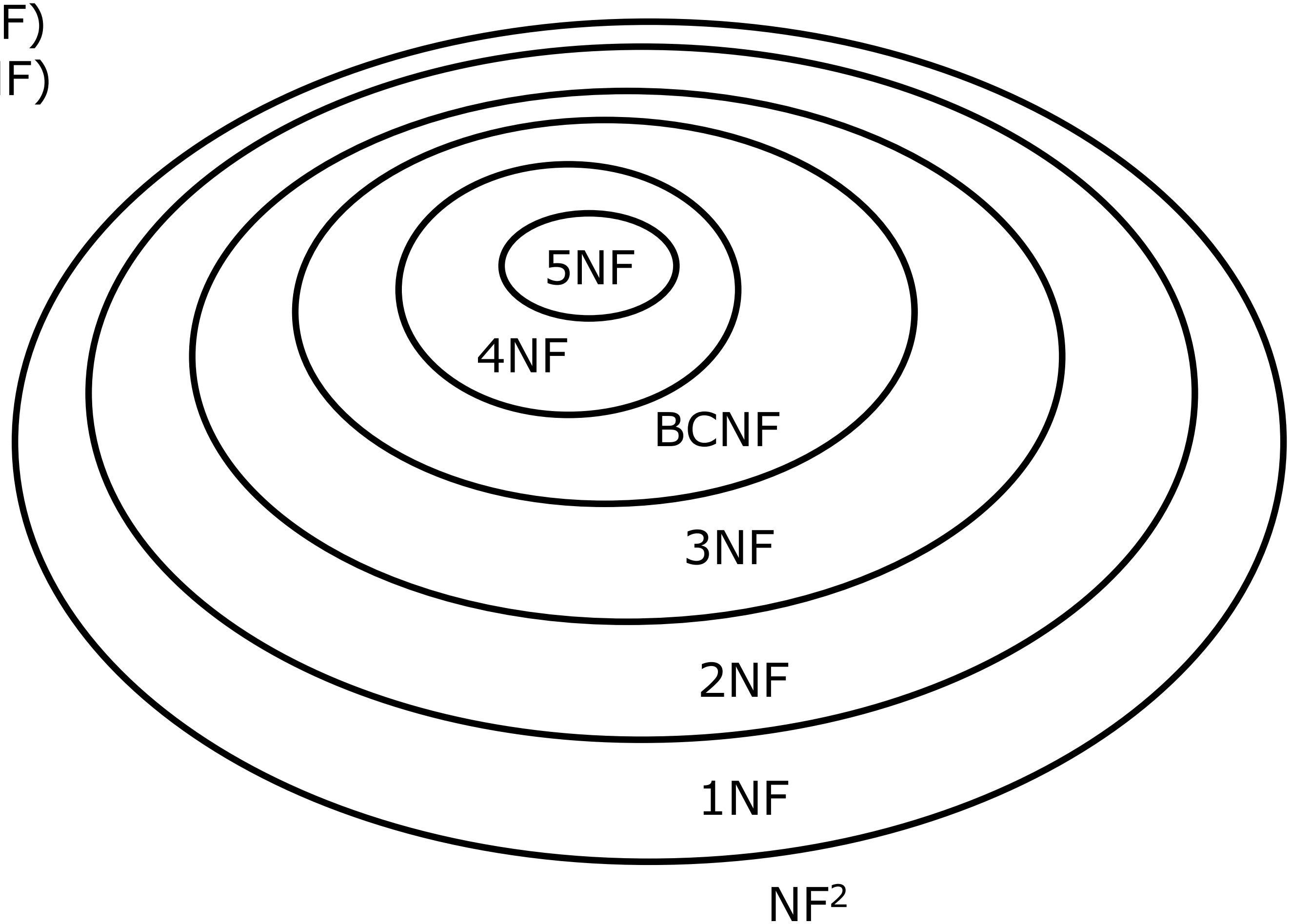
- Objective:
  - Formalize a set of simple ideas that guide a good database design
- Foundations:
  - Every relation must correspond to one semantic concept
    - Normalization theory allows us to recognize when this principle is not fulfilled

NF Structure



NF Structure

Dependencies:
- Functional (1NF, 2NF, 3NF, BCNF)
- Multivalued (4NF)
- Project-Join (5NF)



Functional Dependencies

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

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

Supplying (prov, item, quantity, city)

Fully Functional Dependencies

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

Supplying (prov, item, quantity, city)

Fully Functional Dependencies

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

Supplying (prov, item, quantity, city)

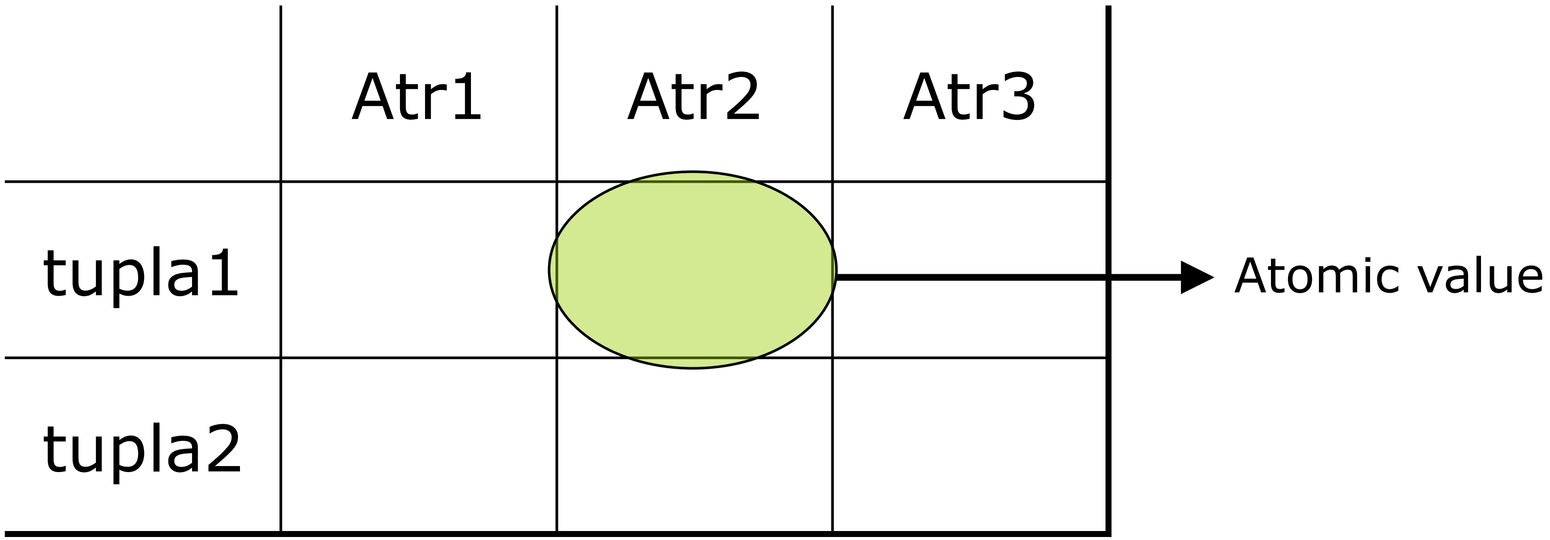
First Normal Form - 1NF (I)

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

	Atr1	Atr2	Atr3
tupla1			
tupla2			

First Normal Form - 1NF (I)

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



First Normal Form - 1NF (II)

Pieces (#piece, description, proj_quantity)

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

First Normal Form - 1NF (II)

Pieces (#piece, description, proj_quantity)

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

Normalize (flatten)

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

First Normal Form - 1NF (II)

Pieces (#piece, description, proj_quantity)

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

Normalize (flatten)

PK?

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

First Normal Form - 1NF (II)

Pieces (#piece, description, proj_quantity)

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

Normalize (flatten)

PK?

100	screw	1	12
100	screw	2	24
101	chair	1	4
101	chair	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)

(<u>prov, item</u>, quantity, provider_city)

Second Normal Form – 2NF (II)

(prov, item, quantity, provider_city)

Second Normal Form – 2NF (II)

(prov, item, quantity, 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 another non-key attribute
- Exception: propagates that of 2NF

Third Normal Form - 3FN (II)

(<u>empl</u>, dept, dept_address)

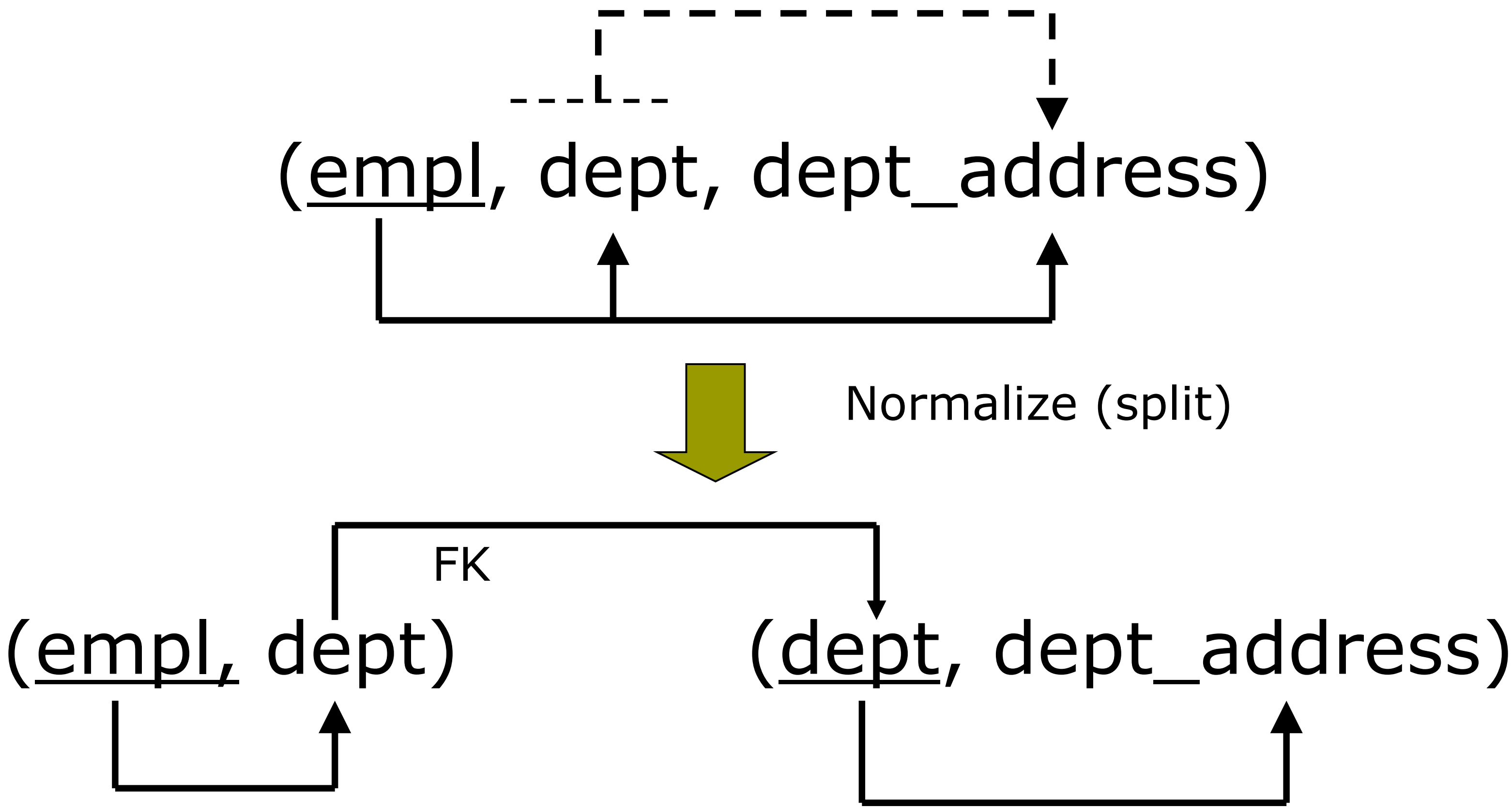
Third Normal Form - 3FN (II)

(empl, dept, dept_address)

Third Normal Form - 3FN (II)

(<u>empl</u>, dept, dept_address)

Third Normal Form - 3FN (II)



2 semantic concepts → 2 tables

Boyce-Codd Normal Form – BCNF (I)

(ssn, subj, #enrolment, mark)

16	DABD	215	MH
16	AIA	215	9
16	ES2	215	8

Boyce-Codd Normal Form – BCNF (I)

(<u>ssn, subj</u>, #enrolment, mark)

16	DABD	215	MH
16	AIA	215	9
16	ES2	215	8

- 1NF?

Boyce-Codd Normal Form – BCNF (I)

(ssn, subj, #enrolment, mark)

16	DABD	215	MH
16	AIA	215	9
16	ES2	215	8

- 1NF?
- 2NF?

Boyce-Codd Normal Form – BCNF (I)

(ssn, subj, #enrolment, mark)

16	DABD	215	MH
16	AIA	215	9
16	ES2	215	8

- 1NF?
- 2NF?
- 3NF?

Boyce-Codd Normal Form – BCNF (I)

(ssn, subj, #enrolment, mark)

16	DABD	~~215~~ 220	MH
16	AIA	215	9
16	ES2	215	8

- 1NF?
- 2NF?
- 3NF?

What happens if #enrolment changes from 215 to 220?

Boyce-Codd Normal Form – BCNF (I)

(<u>ssn, subj</u>, #enrolment, mark)

16	DABD	~~215~~ 220	MH
16	AIA	~~215~~ 220	9
16	ES2	~~215~~ 220	8

Modification anomaly

- 1NF?
- 2NF?
- 3NF?

What happens if #enrolment changes from 215 to 220?

Boyce-Codd Normal Form – BCNF (I)

(ssn, subj, #enrolment, mark)

16	DABD	~~215~~ 220	MH
16	AIA	~~215~~ 220	9
16	ES2	~~215~~ 220	8

Modification anomaly

Repetitions -> Redundancy?

- 1NF?
- 2NF?
- 3NF?

What happens if #enrolment changes from 215 to 220?

Boyce-Codd Normal Form – BCNF (II)

- A relation (SQL table) is in BCNF 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	
#enrolment, subj	
ssn	
#enrolment	

Boyce-Codd Normal Form – BCNF (III)

(ssn, subj, #enrolment, mark)

Determinant	Is it candidate key?
ssn, subj	Yes
#enrolment, subj	
ssn	
#enrolment	

Boyce-Codd Normal Form – BCNF (III)

(ssn, subj, #enrolment, mark)

Determinant	Is it candidate key?
ssn, subj	Yes
#enrolment, subj	Yes
ssn	
#enrolment	

Boyce-Codd Normal Form – BCNF (III)

(ssn, subj, #enrolment, mark)

Determinant	Is it candidate key?
ssn, subj	Yes
#enrolment, subj	Yes
ssn	No
#enrolment	

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

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

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

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

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

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

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

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

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

Conclusions up to BCNF (strong 3NF)

- Any schema can always be normalized up to BCNF
- Normalization is not unique
- The normalized schema (in 3NF) is equivalent to that at the beginning (maybe not true in BCNF)
- The normalized schema is better than that at the beginning because:
  - Eliminates redundancies and anomalies
  - Separates semantically different concepts

Denormalizing

People (<u>id</u>, name, address, telephone, city, province)

Denormalizing

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

Denormalizing

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

BCNF

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

Denormalizing

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

BCNF

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

- When to denormalize?
  - When otherwise the join would be performed too often
  - When changes are not expected or rare
  - When coherence is guaranteed by other means

Armstrong rules

- Reflexivity

For all x, x -> x

Armstrong rules

- Reflexivity

  For all x, x -> x

- Augmentation

  If x->y **then** xz -> y

Armstrong rules

- Reflexivity

  For all x, x -> x

- Augmentation

  If x->y **then** xz -> y

- Projectability or Decomposition

  If x ->yz **then** x->y **and** x->z

Armstrong rules

- Reflexivity

  For all x, x -> x

- Augmentation

  If x->y **then** xz -> y

- Projectability or Decomposition

  If x ->yz **then** x->y **and** x->z

- Addition

  If x->y **and** x->w **then** x->yw

Armstrong rules

- Reflexivity

  For all x, x -> x

- Augmentation

  If x->y **then** xz -> y

- Projectability or Decomposition

  If x ->yz **then** x->y **and** x->z

- Addition

  If x->y **and** x->w **then** x->yw

- Transitivity

  If x->y **and** y->z **then** x->z

Armstrong rules

- Reflexivity

  For all x, x -> x
- Augmentation

  If x->y **then** xz -> y
- Projectability or Decomposition

  If x ->yz **then** x->y **and** x->z
- Addition

  If x->y **and** x->w **then** x->yw
- Transitivity

  If x->y **and** y->z **then** x->z
- Pseudo-transitivity

  If x->y **and** yz->w **then** xz->w

Closure of dependencies

$L \;= \{ FD \}$ ---▶ Explicit functional dependencies

Armstrong rules ↓

$L^+ = \{ FD \}$ ---▶ Explicit and implicit functional dependencies

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

Analysis

- Algorithm:
  1. If relation R with attributes A is not in BCNF (i.e. A_L->A_R exists, with A_L and A_R being subsets of A, violating BCNF)
     1. Decompose R into two relations with attribute sets: A-A_R and A_LUA_R, respectively
  2. If either A-A_R or A_LUA_R is not in BCNF, go back to 1

- Decomposition may be not unique
- Some dependencies may be lost

Example of analysis

R (C, S, J, D, P, Q, V)

{DF} = {SD->P, J->S, JP->C, C->SJDPQV}

$\boxed{\text{CSJDPQV}}$

Example of analysis

R (C, S, J, D, P, Q, V)

{DF} = {SD->P, J->S, JP->C, C->SJDPQV}

CSJDPQV

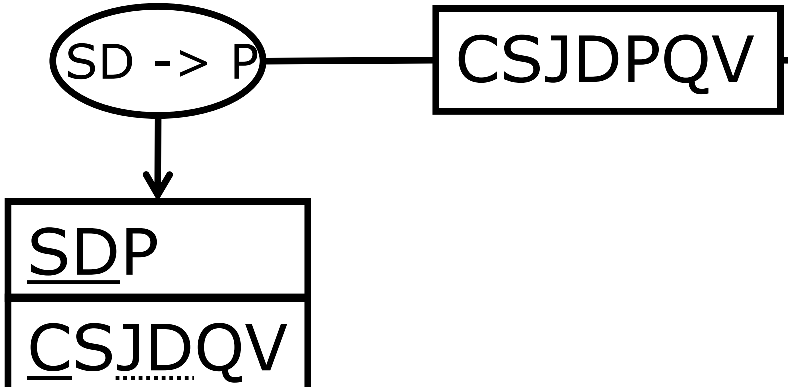
JD -> P given SD -> P and J -> S

But then, JD -> JP and JD is key

Example of analysis

R (C, S, J, D, P, Q, V)

{DF} = {SD->P, J->S, JP->C, C->SJDPQV}



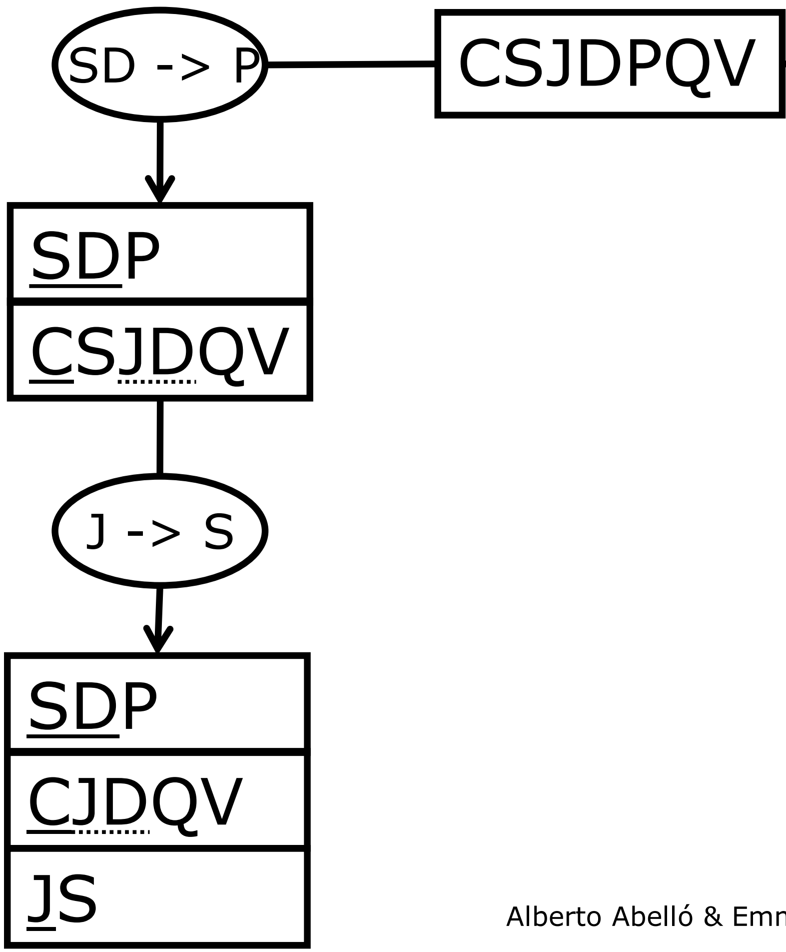
JD -> P given SD -> P and J -> S

But then, JD -> JP and JD is key

Example of analysis

R (C, S, J, D, P, Q, V)

{DF} = {SD->P, J->S, JP->C, C->SJDPQV}



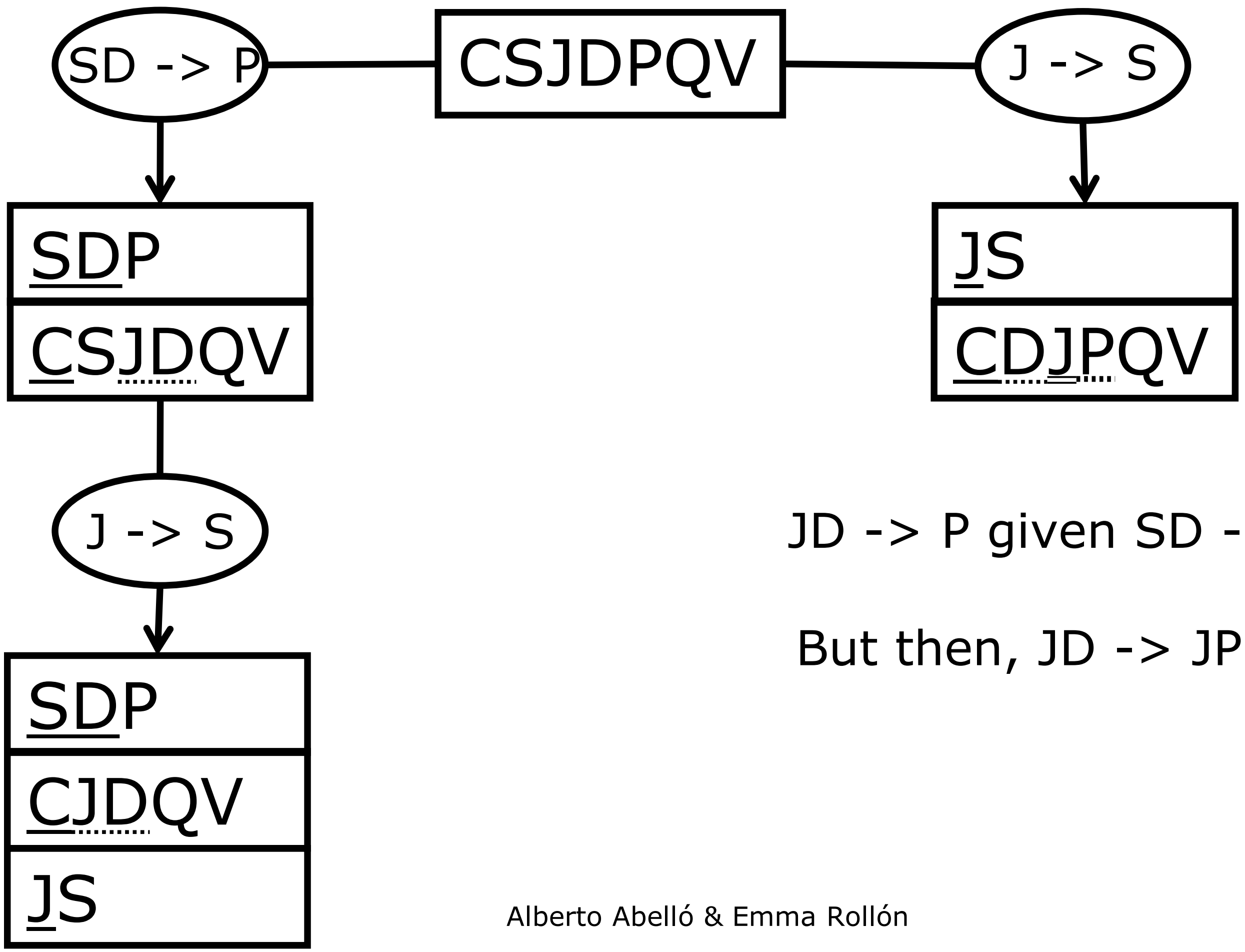
JD -> P given SD -> P and J -> S

But then, JD -> JP and JD is key

Example of analysis

R (C, S, J, D, P, Q, V)

{DF} = {SD->P, J->S, JP->C, C->SJDPQV}



JD -> P given SD -> P and J -> S

But then, JD -> JP and JD is key

Summary

- Functional Dependencies
- Anomalies
  - Update
  - Delete
  - Insert
- Normal Forms:
  - 1NF (Codd '70)
  - 2NF (Codd '70)
  - 3NF (Codd '70)
  - BCNF (Boyce-Codd '74)
- Design methods
  - Armstrong rules
  - Closure
  - Analysis

Bibliography

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