FIB - Disseny de Bases de Dades

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

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

Updating anomaly

Suppliying

prov	item	quant	city	
1	a1	100	~~BCN~~	Athens
1	a2	150	~~BCN~~	Athens
2	a1	200	MAD	
2	a2	300	MAD	
3	a2	100	MAD	

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

Deleting anomaly

Suppliying

prov	item	quant	city
1	a1	100	BCN
1	a2	150	BCN
2	a1	200	MAD
2	a2	300	MAD
~~3~~	~~a2~~	~~100~~	~~MAD~~

Elementary data may be lost unintentionally!

Inserting anomaly

prov	item	quant	city
1	a1	100	BCN
1	a2	150	BCN
2	a1	200	MAD
2	a2	300	MAD
3	a2	100	MAD
~~4~~	~~NULL~~	~~NULL~~	~~Athens~~

Elementary data cannot be inserted independently!

Motivation

Objective:

- Formalize a set of simple ideas that guide a good database design

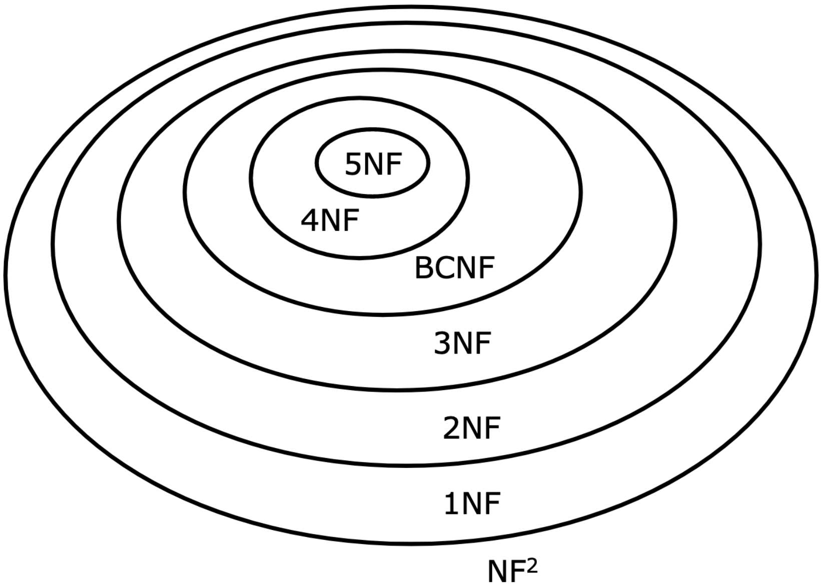
Foundations:

- Every relation must correspond to one semantic concept
  - Normalization theory allows us to recognize when this principle is not fulfilled

NF Structure

Dependencies:

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



Functional Dependencies

$$R(A_1, A_2, ..., A_n)$$

An `FD {X} -> {Y}` guarantees that given a value of `{X}`, this univocally determines the value of `{Y}`

`∀s,t ∈R, s[X] = t[X] ⇒ s[Y] = t[Y]`

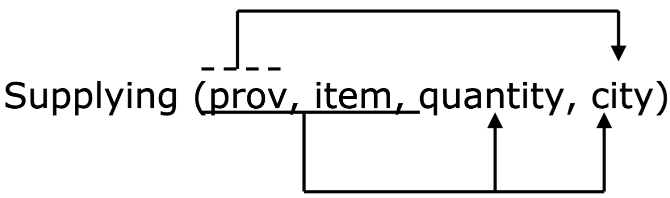
- `{X}` functionally **determines** `{Y}`
- `{Y}` functionally **depends on** `{X}`

Exemple:

> - 08025 -> Barcelona
> - Barcelona -> No ha de ser 08025
>
> Tots els que tinguin el mateix codi postal, obligatòriament han de tenir la mateixa ciutat.
>
> Clau primària sempre la compleix

Fully Functional Dependencies

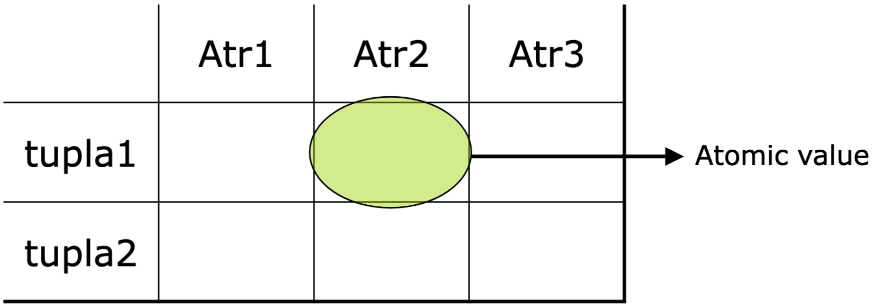
An `FD {X} -> {Y}` is fully (FFD) **iff** there is no proper subset of `{X}` which determines `{Y}`



Proveïdor, item -> quantitat és plena
Proveïdor, item -> ciutat NO és plena (Es pot fer de proveïdor -> ciutat)

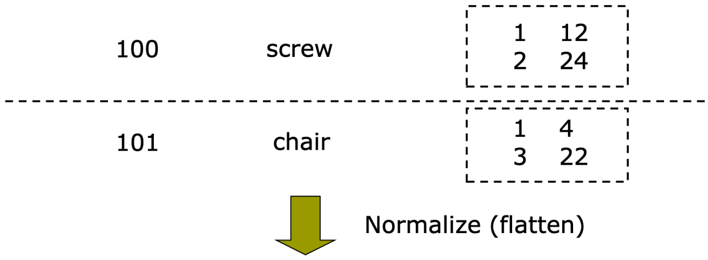
First Normal Form - 1NF - ATÒMIC

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)



Example

Pieces (#piece, description, proj_quantity)



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

> FLATTERN -> fer dues tuples

Només s'aconsegueix si les úniques dependències són de clau primària.

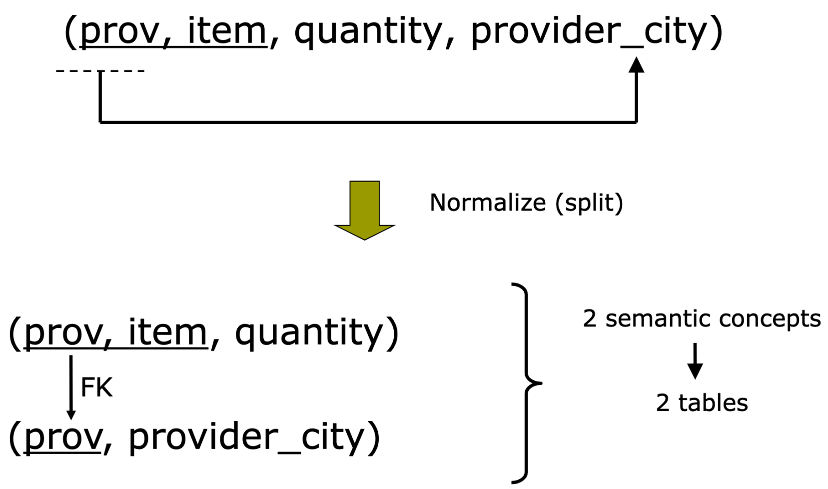
Second Normal Form – 2NF

A relation (SQL table) is in 2NF **iff**:

- It is in 1NF
- **AND**
- 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



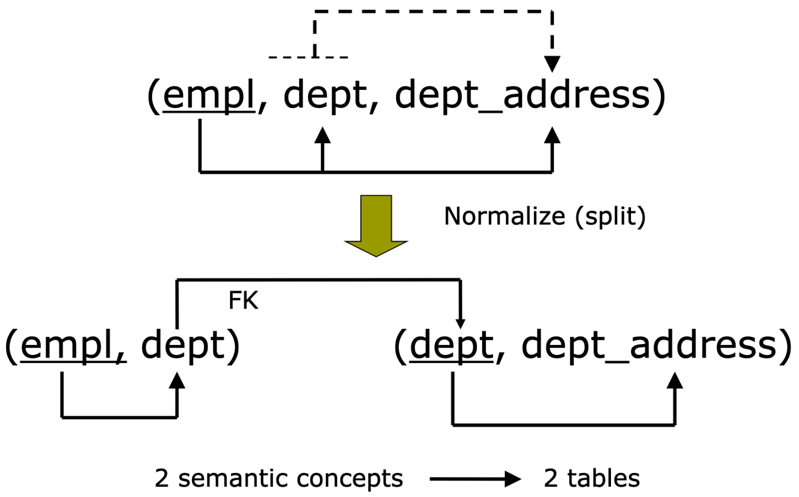
Third Normal Form - 3FN

A relation (SQL table) is in 3NF iff:

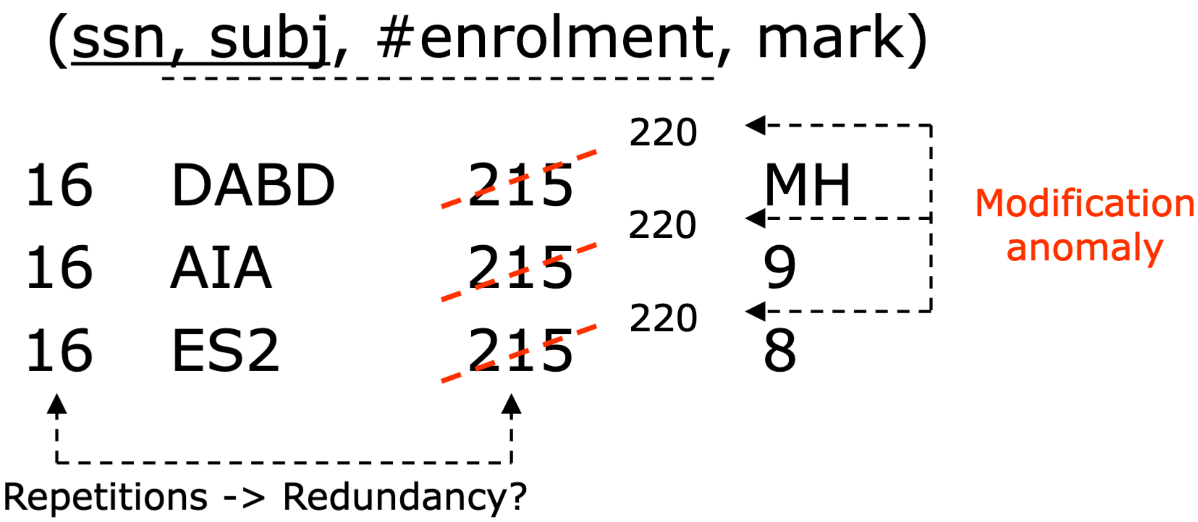
- It is in 2NF
- **AND**
- There is no non-key attribute functionally depending on another non-key attribute

Exception:

- propagates that of 2NF



Boyce-Codd Normal Form – BCNF



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

What happens if #enrolment changes from 215 to 220?

A relation (SQL table) is in BCNF **iff**:

- It is in 1NF
- **AND**
- 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)

(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)

Dos s'autodefineixen entre ells i ells sols NO són candidats clau.

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

BCNF

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

When to denormalize?

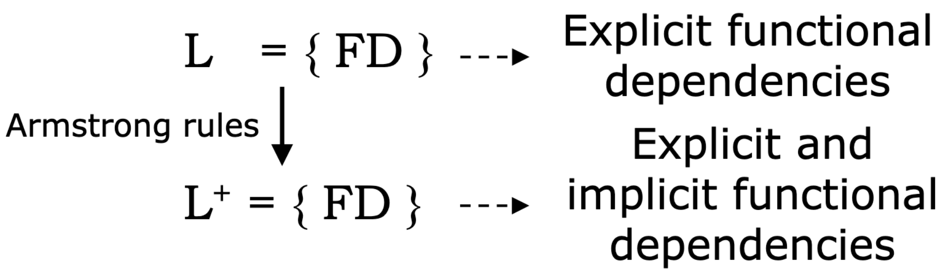
- When otherwise the join would be performed too often

- When changes are not expected or rare
- When coherence is guaranteed by other means

Armstrong rules

Name	Description
Reflexivity	`∀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



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

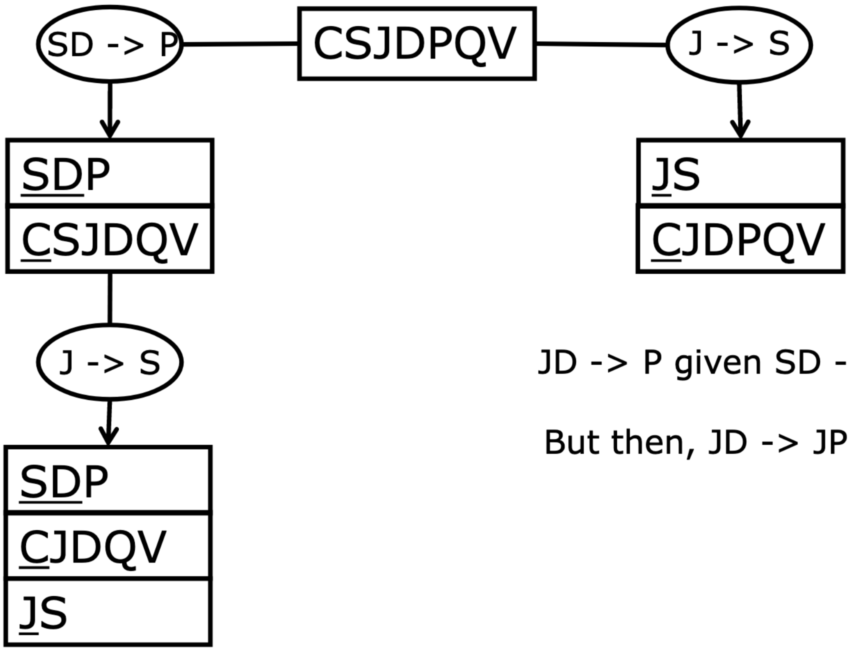
Analysis

- Algorithm:
  1. If relation R with attributes A is not in BCNF (i.e. $A_L \Rightarrow A_R$ exists, with A_L and A_R being subsets of A, violating BCNF)
     - Decompose R into two relations with attribute sets: $A - A_R$ and $A_L \cup A_R$, respectively
  2. If either $A - A_R$ or $A_L \cup A_R$ 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}



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