Normalization of a Relational Schema

(part 5)

Update and Consistency

- Objective of a logical schema : describe a DB which will actually be used
  - loaded, accessed, updated
- Updates (insertions, deletions, modifications) must maintain the consistency of the database
  - referential integrity
  - all integrity constraints
  - in particular the dependencies between attributes
- According to the diagram it is more or less easy
  - The more redundancies there are in the database, the more difficult it is to update with consistency.

Example of update anomalies

Delivery (N°provid ,adrF, N°prod, priceP, qt)

3	Lausanne	52	65	10
22	Bienne	10	15	5
22	Bienne	25	10	12
3	Lausanne	25	10	5
3	Vevey	10	15	20

- If a provider changes addresses and only one tuple is updated $\Rightarrow$ inconsistency
- If a new tuple is inserted with a different address $\Rightarrow$ inconsistency
- If a supplier does not have a delivery in progress, his address is lost ...

What is an 'incorrect' relational database?

A relation is incorrect if:

- it involves repetitions at the level of its population
- it causes problems during the update (insertions, modifications and deletions)
- The conditions for a relationship to be correct can be formally defined:

=> normalization rules

Example (cont.)

Delivery (N°provid, adrF, N°prod, priceP, qt)

3	Lausanne	52	65	10
22	Bienne	10	15	5
22	Bienne	25	10	12
3	Lausanne	25	10	5
3	Lausanne	10	15	20

- The supplier's address depends on the supplier and not on the product.
- The price of the product depends only on the product

$\Rightarrow$ **REDUNDANCIES**

$\Rightarrow$ Update anomaly

- This relation is incorrect. It must be normalized.

Normalization of a logical schema

- Process of transforming a schema (S1) to obtain another schema (S2) :
  - which is equivalent (same content)
  - therefore the updates are simple
- Simple update:
  - 1 elementary change in the real world results in an (1) update of a (1) tuple
- Examples of elementary changes:
  - A provider changes address
  - A product changes price

  Delivery (N°provider, adrF, N°prod, priceP, qt)
  - In this relation the updates are complex

Normalization of a relation

- The process of decomposing a complex update relation into several simple update relations
- Process done to the relational schema
- Example: the relation
  Delivery (N°provider, adrF, N°prod, priceP, qt)
  decomposes into:

  Delivery (N°provider, N°prod, qt)

  Provider (N°product, adrF)

  Product (N°prod, priceP)

Formalization of the problem

The address of a supplier depends on the supplier, ...

$\Rightarrow$ FUNCTIONAL DEPENDENCY (FD)

- Notation :
  - A, B, C … attributes
  - X, Y, Z … sets of attributes
- Let R be a relation R (X, Y, Z)
- An FD exists : X→Y if and only if in R a given value of X always corresponds to the same value of Y.

Formalization

- R :

X	Y	Z
x1	y1	z1
.....		
x1	y1	z2
.....		

An FD exists : X→Y
if and only if in R a given value of X always corresponds to the same value of Y.

- X → Y : X determines Y Y depends on X
- X : source of the FD Y : target of the FD
- The source can be a set of attributes :
  (firstName, lastName) → address

Normal Forms : 1NF

- A relation is in 1NF if each value of each attribute of each tuple is a simple value (all the attributes are atomic and mono-valued).
- Example :

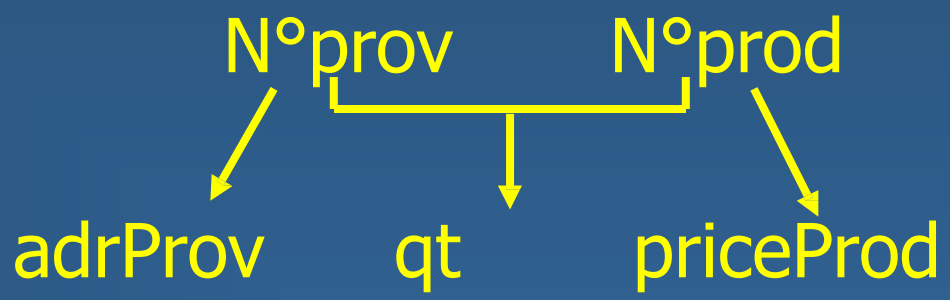
Delivery (N°provider, adrProv, N°prod, priceProd, quantity) is in 1NF

- In relational, for the most part, we only deal in 1FN relations

2nd Normal Form : 2NF

- Eliminates attributes that do not describe the "object" represented by the relation

Delivery (N°prov, adrProv, N°prod, priceProd, qt)

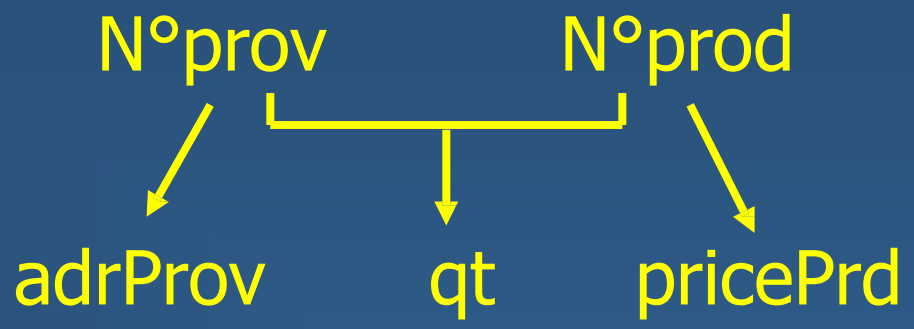


Delivery is a mixed description:

- → of the delivery (N°provider, N°product, quantity)
- → of the provider (N°provider, address-provider)
- → of the product (N°product, price)

2nd Normal Form : 2NF

Delivery (N°prov, adrProv, N°prod, pricePrd, qt)



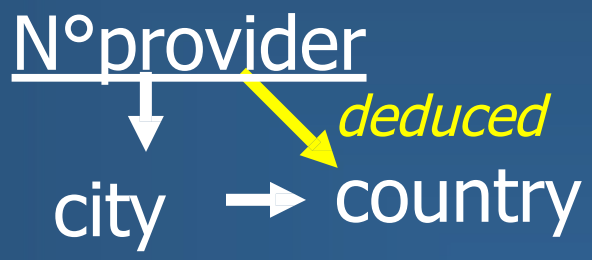
- Definition : a relation is in the 2NF if

→ it is in the 1NF, and
→ each attribute which does not compose the identifier depends on the identifier only

- FDs leave components of Delivery's identifier => Delivery **is not** in the 2NF

3NF : 3rd Normal Form

- The 3rd normal form eliminates sub-relations included in a relation
- Example : Provider (N°provider, city, country)



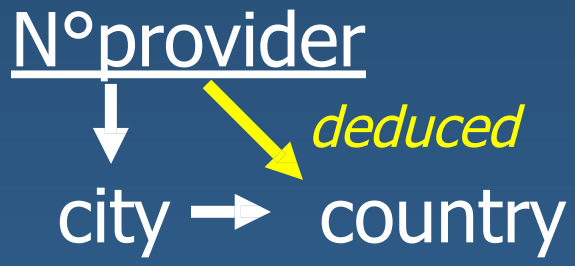
- must be decomposed into:

F (N°provider, city)

G (city, country)

3rd Normal Form : definition

- Provider (N°provider, city, pays)



- Definition : A relation is in 3NF if
  - it is in the 2NF, and
  - each attribute which is not a component of an identifier must depend directly on an identifier
- Depth of the FD tree >1 => Provider is not in the 3rd normal form

Importance of the 3rd Normal Form

- Any relation can be broken down into many relations conforming to the 3rd normal form without any loss
  - without loss of FD, **and**
  - without any information loss
- This is not true for higher forms
- So always make a schema conform to at least the 3rd normal form

Conclusion

- Design of a relational DB :

  -> Entity-Relationship Schema (conceptual level)

  -> Relational Schema (logical level)

  Objective : create a set of relations such that

  - Each relation describes an "object" with only the attributes directly related to it
  - Information redundancies (which generate problems during updates) are eliminated