Preserving F

SAPIENZA
UNIVERSITÀ DI ROMA

The content of the slides is authored by Prof. Maria de Marsico, Sapienza University of Rome
Part of the material is taken from the book: Atzeni, Ceri, Paraboschi, Torlone Basi di dati - Modelli e linguaggi di interrogazione , McGraw-Hill, 2002.

What it means to preserving dependencies

- a schema is usually decomposed:
- **when it is not in 3NF**
- for efficiency reasons:
  - the smaller the size of the tuples, the greater the number we can load into memory in the same read operation
  - if the tuple information is not used by the same type of operations in the database, it is better to decompose the schema
  - Example: the Student schema could be decomposed by separating the personal information (CF, FirstName, LastName, DateBirth, PlaceBirth, etc.) from the academic information (Matriculation, CourseDegree, YearCourse, etc.).

What it means to preserving dependencies

Sapienza Università di Roma
Dipartimento di Informatica

- we have seen that when a schema is decomposed, **it is not enough for the** subschemas to be in 3NF

- let's review the two examples (one more "abstract", the other more "concrete")

What we want to achieve - 3NF is not enough

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- a schema that is not in 3NF can be **decomposed** in **multiple ways** into a set of schemas in 3NF
- for example, the schema $R = ABC$ with the set of functional dependencies $F\{A \rightarrow B, B \rightarrow C\}$ is not in 3NF, due to the presence in F^+ of the transitive dependency $B \rightarrow C$ (the key is A)
- R can be decomposed into:
  - $R1=AB$ with $A \rightarrow B$ and
  - $R2=BC$ with $B \rightarrow C$
  - **or**
  - $R1=AB$ with $A \rightarrow B$ and
  - $R2=AC$ with $A \rightarrow C$
- both schemas **are** in 3NF, however the second solution **is not satisfactory**

What we want to achieve - 3NF is not enough

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- if we consider two **legal** instances of the obtained schemas:

R1

A	B
a1	b1
a2	b1

R2

A	C
a1	c1
a2	c2

- the instance of the original *R* schema that I can reconstruct from this (the only way is to reconstruct it by doing a natural join!) is:

R

A	B	C
a1	b1	c1
a2	b1	c2

- **BUT** it is not a legal instance of R, since it **does not satisfy** the functional dependency $B \to C$

we want to preserve ALL dependencies in F^+

Example

Sapienza Università di Roma
Dipartimento di Informatica

- let us consider the schema *R = {Matriculation, Town, Province}* with the set of functional dependencies:
- *F={Matriculation → Town, Town → Province}*
- the schema is not in 3NF due to the presence in F^+ *of* the transitive dependency Town → *Province (*the key is *Matriculation and Province* transitively depends on *Matriculation)*
- *R* can be decomposed into:
- *R1(Matriculation, Town)* with Matriculation → *Town*
- *R2(Town, Province)* with Town → *Province*
  - **or**
- *R1(Matriculation, Town)* with Matriculation → *Town*
- *R2(Matriculation, Province)* with Matriculation → *Province*
- both schemas **are** in 3NF, but the second solution **is not satisfactory**

What we want to achieve - 3NF is not enough

- consider the **legal** instances of the obtained schemas:

R1

Matriculation	Town
501	Tivoli
502	Tivoli

R2

Matriculation	Province
501	Rome
502	Rieti

- the instance of the original *R* schema that we can reconstruct from this with a natural join is:

R

Matriculation	Town	Province
501	Tivoli	Rome
502	Tivoli	Rieti

- **but** it is not a legal instance of R, since it **does not satisfy** the functional dependency Town$\rightarrow$*Province*
- **clearly there was an error in the data, but we could not detect it**

Definitions

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- **Definition:** let *R* be a relation scheme; a <u>*decomposition*</u> of *R* is a *family* $R_1, R_2, \ldots, R_k$ of subsets of *R* **covering** *R, that is*

$$R = \bigcup_{i=1}^{k} R_i$$

(the subsets may have **nonempty** intersection)

- **in other words**: decomposing it means defining **subschemas, each of them** containing **a subset of** the attributes of R and their union is R
- **so,** R is a set of attributes and a decomposition of R is a family of sets of attributes

Example

Sapienza Università di Roma
Dipartimento di Informatica

- given the schema R=(TaxC, Name, Surname, Matriculation, DateB, PlaceB, Degree, Year) we can have the decompositions:

  - R1=(TaxC, Matriculation, Name, Surname, DateB, PlaceB)
  - R2=(Matriculation, TaxC, Name, Surname, Degree, Year)
  - or
  - R1=(TaxC, Matriculation, Name,Surname, DateB, PlaceB)
  - R2=(Matriculation, Degree, Year)

- in both cases, the families of subsets of R (subschemas) cover the schema, but then we need to check if both decompositions are "good", i.e., if the subschemas are in 3NF, F is preserved, and we can reconstruct any legal instance of R by natural join of the instances of the subschemas

Attention

Sapienza Università di Roma
Dipartimento di Informatica

- "**decomposing**" an **instance of** a relation **with a certain schema**, **based on the decomposition of the schema itself**, means **projecting each** tuple of the original instance onto the attributes of the individual subschemes ...

- ...**eliminating the duplicates** that might be generated by the fact that two tuples are distinct **but have a common portion** that falls into the same subschema

Example

R

TaxC	Name	Last name	Matriculation	Datan	PlaceB	Degree	Year
DDD	Davide	Bigi	1111	12-12-90	Bari	Letters	3
FFF	Gianni	Neri	1212	15-09-91	Milan	Law	2
AAA	Antonio	Rossi	1313	09-08-92	Naples	Mathematics	1

R1

TaxC	Name	Surname	Matriculation	DateB	PlaceB
DDD	Davide	Bigi	1111	12-12-90	Bari
FFF	Gianni	Neri	1212	15-09-91	Milan
AAA	Antonio	Rossi	1313	09-08-92	Naples

subschema of R

projection of the instance onto the subschema

R2

TaxC	Name	Surname	Matriculation	Degree	Year
DDD	Davide	Bigi	1111	Letters	3
FFF	Gianni	Neri	1212	Law	2
AAA	Antonio	Rossi	1313	Mathematics	1

subschema of R

projection of the instance onto the subschema

Definition

- we continue with the definition of **equivalence between two sets of functional dependencies.**

- **Definition:** let *F* and *G* be two sets of functional dependencies; *F* and *G* are ***equivalent***, written ***F≡G*** if

$$F^{+} = G^{+}$$

- **F and G do NOT contain the same dependencies, but their <u>closures</u> do**

Checking equivalence

- **checking the equivalence of** two sets of functional dependencies *F* and *G* requires verifying the **equality of F^+ and G^+**, i.e., that

$$F^+ \subseteq G^+ \text{ and that } F^+ \supseteq G^+$$

- as mentioned earlier, computing the closure of a set of functional dependencies takes **exponential time**
- however, the following **lemma** and an algorithm allow us to check the equivalence of the two sets of functional dependencies **in polynomial time**

Lemma on closures

- **Lemma:** let F and G be two sets of functional dependencies; if **$F \subseteq G^+$ then $F^+ \subseteq G^+$**
- **Demonstration:** let **$f \in F^+ - F$** (f is a dependency in **F^+** that **does not appear** in F)
- **every functional dependency in F** can be **derived from G** by Armstrong's axioms (for **the hypothesis** it is in **G^+,** and we know that **$F^+ = F^A$**)
- always for the Theorem that proves that **$F^+ = F^A$**, f $\in$ **F^+** can be derived **from F by applying the Armstrong's axioms**
- **so, any $f \in F^+$ can be derived from G by Armstrong's axioms**

$$G \xrightarrow{A} F \xrightarrow{A} F^+$$

Definition

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- **Definition:** let **R** be a **relation scheme**, **F a set of functional dependencies on R** and $\rho = R_1, R_2, \ldots, R_k$ a **decomposition** of R:
- we say that ρ **preserves F** if

$$F \equiv G = \cup_{i=1}^{k} \pi_{Ri}(F),$$

where $\pi_{Ri}(F) = \{ X \rightarrow Y \in F^+ \mid XY \subseteq R_i \}$

- **note:**
- obviously, $G = \cup_{i=1}^{k} \pi_{Ri}(F)$ is a set of functional dependencies
- each $\pi_{Ri}(F)$ is a set of functional dependencies obtained by projecting F onto the subschema R_i
- projecting a set of dependencies F onto a subschema R_i means **taking all** the dependencies **derivable from F** by the Armstrong's axioms (so, those **in F^+**) that have **all** the attributes (determinants and dependents) **in R_i**

Check if F is preserved

SAPIENZA
Università di Roma
Dipartimento di Informatica

- suppose we **already have** a **decomposition** and want to **check if it** preserves the functional dependencies in F

- checking **whether** a decomposition preserves a set of functional dependencies F requires verifying **the equivalence** of the two sets of functional dependencies F and G = $\cup_{i=1}^{k} \pi_{R_i}(F)$

- **note**: **because the definition of G**, it will always be that $\mathbf{F^+ \supseteq G}$
- each **projection of F** that is **included in G** is also in $\mathbf{F^+}$**,** and, by the lemma, that implies that $\mathbf{G^+ \subseteq F^+}$
- so, given the Lemma, it is enough to verify that $\mathbf{F \subseteq G^+}$

Check if F is preserved

Algorithm - F in *G*⁺

Input: two sets *F* and *G* of functional dependencies on *R*;

Output: the success variable (***true*** if $F \subseteq G^+$, ***false*** otherwise)

begin

success = true

for each $X \rightarrow Y \in F$

begin

compute X^+_G

if $Y \not\subseteq X^+_G$ **then** *success = false*

end

end

- **if** $Y \subseteq X^+_G$ then $X \rightarrow Y \in G^A$ (for the lemma) and $X \rightarrow Y \in G^+$ (for the Theorem)
- it is enough to verify that **even a single dependency** does not belong to the closure of G to be able to state that **the equivalence does not exist**

Problem

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- **how do we compute X^+_G?**
- to use the Algorithm already seen for calculating the closure of X, we would **first need** to **calculate *G***
- … but, **by the definition of *G***, this **requires** the computation of **F⁺**, which takes **exponential** time
- we present an algorithm to compute X^+_G **from *F***

Calculating X^+_G from F

Algorithm - X^+_G from F

Input: a schema R, a set F of functional dependencies on R, a decomposition $\rho = R_1, R_2, \ldots, R_k$, a subset X of R

Output: the closure of X with respect to $G = \cup_{i=1}^{k} \pi_{R_i}(F)$, (**in variable *Z***)

begin

$Z = X$

$S = \varnothing$

for $i = 1$ **to** k

  $\mathbf{S = S \cup (Z \cap R_i)^+_F \cap R_i}$

while $S \not\subset Z$

  begin

    $\mathbf{Z = Z \cup S}$

    for $i = 1$ **to** k

      $\mathbf{S = S \cup (Z \cap R_i)^+_F \cap R_i}$

  end

end

Observation

we note the relationship between the definition of the projections of F:

$\pi_{Ri}(F) = \{ X \rightarrow Y \in F^+ \mid XY \subseteq R_i \}$

and the update step of S:

$S = S \cup (Z \cap R_i)^+_F \cap R_i$

we are collecting the attributes functionally determined **by the part of Z** that intersects $\boldsymbol{R_i}$, (for the lemma, $A \in \boldsymbol{X^+}$ if and only if $\boldsymbol{X \rightarrow A \in F^+}$, ***and in this case*** $\boldsymbol{X = Z \subseteq R_i}$) according to the dependencies in $\boldsymbol{F^+}$; and from these we select those contained in $\boldsymbol{R_i}$

starting from a set X, we are "reconstructing" its **closure** from the projections of F that **make up G**, and from there iterating from the set $\boldsymbol{G^+}$ (through transitivity)

by intersectiing with $\boldsymbol{R_i}$, we ensure that the dependency is valid in the individual subschema

Observation

- we also note the difference with the update step of S in the two algorithms:

$S = A \mid Y \to V \in F,\ A \in V \wedge Y \subseteq Z$
$S = S \cup (Z \cap R_i)^+_F \cap R_i$

  - here the union is needed as **we are considering** the different subschemas **in turn**
  - we also take dependencies in the closure $\boldsymbol{F^+}$ **($\boldsymbol{Z \cap R_i \to A \in F^+}$ and… the lemma)** as **the dependencies in the set G** are included in the projections of F onto the subschemas, which are however, dependencies in $\boldsymbol{F^+}$
- in conclusion we will find the attributes that functionally depend on X even if they belong to subschemas in which <u>X is not included!</u> as they depend on attributes that are in the same subschema as X and depend on X, but they are also in other subschemas!

Careful!

- the algorithm (by definition) **always terminates**
- the fact that the algorithm terminates **does not indicate** that a dependency $X \rightarrow Y$ **is preserved!**
- to **check if** $X \rightarrow Y$ is preserved, we need to use the previous algorithm

Theorem

- **Theorem:** let *R* be a relation schema, *F* a set of functional dependencies on *R*, $\rho = R_1, R_2, \ldots, R_k$ a decomposition of *R* and *X* a subset of *R;* the Algorithm correctly computes X^+_G, where $G = \pi_{Ri}(F)$