Closure of X

SAPIENZA
UNIVERSITÀ DI ROMA

What we want to achieve

- when decomposing a relation schema R on which a set of functional dependencies F is defined, **in addition to** obtaining schemas in 3NF it is necessary to
  - **preserving dependencies**
  - **to** be able to **reconstruct by join** all and only the original information.

- the functional dependencies that we want to preserve are **all** those that are satisfied **by every legal instance of R**, i.e., the functional dependencies in $\mathbf{F^+}$

- so, we are interested in calculating $\mathbf{F^+}$ and we know how to do it, but...

What we want to achieve

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- ...calculating F^+ requires exponential time in R

- luckily, for our purposes it is enough to have a method to decide whether a functional dependency $X \rightarrow Y$ belongs to F^+ (i.e., to the closure of a set of dependencies)

- this can be done by calculating X^+ and checking whether $Y \subseteq X^+$

in fact, because of the lemma : $X \rightarrow Y \in F^A$ if and only if $Y \subseteq X^+$

- and of the theorem, showing that $F^A = F^+$

Use of X^+

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- we will see that the calculation of $\boldsymbol{X}^{+}$ is useful in several cases:
  - to check if a set of attributes is the key of a schema
  - to check whether a decomposition preserves the functional dependencies of the original schems

How to compute X^+

Algorithm "Calculation of X^+"

Input a relation scheme R, a set F of functional dependencies on R, a subset X of R

X may be a **single attribute**

Output the closure of X with respect to F (**returned in the variable Z**)

begin
$Z = X$
$S = A \mid Y \to V \in F, A \in V \wedge Y \subseteq Z$
while $S \not\subset Z$ **do:**
begin
$Z = Z \cup S;$
$S = A \mid Y \to V \in F, A \in V \wedge Y \subseteq Z$
end
end

We insert into S the **individual** attributes that make up the right-hand parts of dependencies **in F** whose left-hand part **is contained in Z** (in practice decomposing the right-hand parts)

at first, Z is just X, so we insert attributes that are **functionally determined** by X; once these have entered Z, from these we add more (by **transitivity**).

we can "number" the values of Z

How to compute X^+

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Algorithm "Calculation of X^+"

Input a relation scheme R, a set F of functional dependencies on R, a subset X of R

X may be a **single attribute**

Output the closure of X with respect to F (**returned in the variable Z**)

begin

$Z = X$

$S = A \mid Y \rightarrow V \in F, A \in V \wedge Y \subseteq Z$

while $S \not\subset Z$ **do:**

begin

$Z = Z \cup S;$

$S = A \mid Y \rightarrow V \in F, A \in V \wedge Y \subseteq Z$

end

end

at the iteration i+1 we add to S the **single** attributes that make up the right-hand parts of dependencies **in F** whose left part **is contained in Z^{i-1}**, i.e. $S^i = A \mid Y \rightarrow V \in F, A \in V \wedge Y \subseteq Z^{i-1}$

at the end of each iteration we add something to Z, **but we NEVER delete** any attributes

the algorithm stops when the **new** set S^i we obtain is (**already**) contained in the set Z^i, i.e., when **we cannot** add **new** attributes to the transitive closure of X

We are implicitly using F^A

F= {AB→C, B→D, AD→E, CE→H}
R=ABCDEHL

we want to calculate the closure of AB

Z=AB
S={C, D}   AB→C in F , to insert D: AB→B (RIF) + B→D (in F) = AB→D (TRANS)

does S have anything extra?
Z={A,B,C,D}
S={C,D,E } to insert E: AB→B (RIF) + B→D (in F) + AB→AD (AUM) + AD→E (in F) = AB→E (TRANS)

does S have anything extra?
Z= ABCDE
S= CDEH to insert H: AB→C (in F) + AB→AD (AUM of B→D in F) + AD→E (in F) + AB→E (TRANS) + AB→CE (UNION) + CE→H (in F) = AB→H (TRANS)

does S have anything extra?
Z= ABCDEH
S=CDEH

does S have anything extra?
STOP

extending the variable Z from which we take the determinants in successive while cycles is equivalent to applying Armstrong's axioms

The algorithm is correct

Theorem: The Algorithm **"Calculation of X^+"** correctly computes the closure of a set of attributes X with respect to a set F of functional dependencies.

Dim. Let us denote by $Z^{(0)}$ the initial value of Z ($Z^{(0)}$=X) and by $Z^{(i)}$ and $S^{(i)}$ the values of Z and S **after** the i-th execution of the while loop; it is easy to see that $\mathbf{Z^{(i)} \subseteq Z^{(i+1)}}$, for each i

Remember:
In $Z^{(i)}$ there are the attributes added to Z ***up to*** *the i-th iteration*
At the end of each iteration we add something to Z, **but we NEVER delete** any attributes

Let j be such that S(j) ⊆ Z(j) (i.e., Z(j) is the value of Z when the algorithm **terminates**); we will prove that:

$$\boxed{\mathbf{A \in Z^{(j)} \text{ if and only if } A \in X^+}}$$

Theorem

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- **part 1 (only if)** we will show by **induction** on i that $Z^{(i)} \subseteq X^+$, for **each i (**and therefore, in particular, $Z^{(j)} \subseteq X^+$)

reflexivity

- basis of induction (i = 0): since $\mathbf{Z^{(0)}=X}$ and $\mathbf{X \subseteq X^+}$, we have that $\mathbf{Z^{(0)} \subseteq X^+}$

it was added during the i-th iteration, as it **was not** in $Z^{(i-1)}$

- induction step (i > 0): for **the inductive hypothesis** $\mathbf{Z^{(i-1)} \subseteq X^+}$; let **A** be an attribute in $\mathbf{Z^{(i)}-Z^{(i-1)}}$
- there **must** exist a dependency $\mathbf{Y \rightarrow V \in F}$ such that $\mathbf{Y \subseteq Z^{(i-1)}}$ and $\mathbf{A \in V}$; since $\mathbf{Y \subseteq Z^{(i-1)}}$, by the inductive hypothesis we have that $\mathbf{Y \subseteq X^+}$ and by the lemma $\mathbf{X \rightarrow Y \in F^A}$
- since $\mathbf{X \rightarrow Y \in F^A}$ and $\mathbf{Y \rightarrow V \in F}$, by **transitivity** we obtain $\mathbf{X \rightarrow V \in F^A}$ and by the Lemma , $\mathbf{V \subseteq X^+}$
- therefore, **for each** $\mathbf{A \rightarrow Z^{(i)} - Z^{(i-1)}}$ we obtain $\mathbf{A \in X^+}$ and $\mathbf{Z^{(i)} \subseteq X^+}$

the attributes in $Z^{(i-1)}$ *are there by inductive hypothesis, and we have shown that those inserted in Z* ***at the i-th iteration of*** *the loop also go there*

Theorem

- **part 2 (if)** let **A be an attribute in X^+** and let j be such that $S^{(j)} \subset Z^{(j)}$ (i.e., $Z^{(j)}$ is the value of Z when the algorithm terminates); w**e will show that A $\in Z^{(j)}$**
- as **A $\in X^+$** , we know that **$X \rightarrow A \in F^+$** (by the Theorem); therefore, $X \rightarrow A$ must be satisfied by each legal instance of R
- let us consider the following instance r of R:

	$Z^{(j)}$				$R- Z^{(j)}$			
r	1	1	…	1	1	1	…	1
	1	1	…	1	0	0	…	0

- we first show that r is a legal instance of R

Theorem

	$Z^{(j)}$				$R- Z^{(j)}$			
r	1	1	…	1	1	1	…	1
	1	1	…	1	0	0	…	0

- if we assume, by contradiction, that there exists a functional dependency $V \to W \in F$ which is not satisfied by r
- then it should be that $V \subseteq Z^{(j)}$ (the values of the two tuples **are equal ONLY** in that subset of R, and we **need them to be equal** to be able to say that a dependency IS NOT satisfied) and $W \cap (R- Z^{(j)}) \neq \emptyset$
- **but**, in that case, we would have $S^{(j)} \not\subset Z^{(j)}$ **(contradiction)**

Theorem

Sapienza Università di Roma
Dipartimento di Informatica

	$Z^{(j)}$				$R- Z^{(j)}$			
r	1	1	…	1	1	1	…	1
	1	1	…	1	0	0	…	0

- we assumed that j is the value for which $S^{(j)} \not\subset Z^{(j)}$ (at iteration j we haven't added ANYTHING new)
- If $V \subseteq Z^{(j)}$ and $W \cap (R- Z^{(j)}) \neq \emptyset$, there would exist some elements of W that are not yet in $Z^{(j)}$; by applying the algorithm at the iteration j+1 we could collect these NEW elements via $V \rightarrow W$ and then insert them in S and $Z^{(j+1)}$
- but we said that the algorithm stops only when it is no longer possible to insert new elements in Z, so we reached a contradiction

Theorem

	$Z^{(j)}$				$R- Z^{(j)}$			
r	1	1	…	1	1	1	…	1
	1	1	…	1	0	0	…	0

- again, by contradiction, let's assume that $A \in X^+$, $A \notin Z^{(j)}$
- as r is a legal instance of R it **must** satisfy $\mathbf{X \rightarrow A \in F^+}$
- so, if there are two tuples in r with the same values on X, then they must be equal also on A
- are there two tuples with the same values on X? yes, as $X \subseteq Z^{(0)} \subseteq Z^{(j)}$
- then the two tuples MUST also be equal on A, which, then, must be in $Z^{(j)}$ (contradiction)

Reminder: properties of the empty set

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- $\varnothing$ - the empty set
- $\{\varnothing\}$ - a **set that contains the empty set**
- the empty set is a subset of each set A:
  $\forall A : A \supseteq \varnothing$
- the union of any set A with the empty set is $\forall A : A \cup \varnothing = A$
- the intersection of any set A with the empty set is the empty set: $\forall A : A \cap \varnothing = \varnothing$
- the Cartesian product of any set A with the empty set is the empty set: $\forall A : A \times \varnothing = \varnothing$
- the only subset of the empty set is the empty set itself
- the number of elements of the empty set (i.e., its cardinality) is zero; the empty set is therefore finite: $|\varnothing| = 0$

Example

- given the relation scheme R = (A, B, C, D, E, H) and the following set of functional dependencies on R
- F = { AB CD, EH D, D H }
- calculate the closure of the sets A, D and AB

Example

R = (A, B, C, D, E, H) F = { AB → CD, EH → D, D → H }

begin

Z = **A**

S = { L|Y → V ∈ **F,** L∈V ^ Y⊆A} (A alone does not determine any other attribute)

while (S ⊄ Z)? no, we do **not** enter the first iteration of while

$A^+ = A$

Example

$R = (A, B, C, D, E, H)$ $F = \{ AB \rightarrow CD, EH \rightarrow D, D \rightarrow H \}$

begin

Z = ***D***

S = { L|Y → V ∈ **F,** L∈V ^ Y⊆D} = H (for the dependency D → H)

while (S ⊄ Z): ***H***⊄D **so** we enter the first iteration of the while

  begin (first iteration of while)

    Z=Z∪S = D∪H = ***DH***

    S = { L|Y → V ∈ **F,** L∈V ^ Y⊆DH} = H(for dependency D → H)

  end

while (S ⊄ Z): H⊂DH (we haven't added anything new)

we exit the while

D $^{+}$***= DH***

Example

R = (A, B, C, D, E, H) F = { AB → CD, EH → D, D → H }

begin

Z = **AB**

S = {L|Y → V ∈ **F,** L∈V ^ Y ⊆ AB} = CD (for thr dependency AB → CD)

while (S ⊄ Z): CD ⊄ AB so, we enter the first iteration of while

begin

Z = Z ∪ S = AB ∪ CD= **ABCD**

S = {L|Y → V ∈ **F,** L∈V ^ Y ⊆ ABCD} = {C, D (for dependency AB → CD), **H** (for dependency D → H)} = CDH

end

while: CDH ⊄ ABCD so, we enter the second iteration

begin

Z = Z ∪ S = ABCD ∪ **CDH** = **ABCDH**

S = {L|Y → V ∈ **F,** L∈V ^ Y ⊆ ABCDH} = {C, D (for dependency AB → CD), H (for dependency D → H)}=CDH

end

while: CDH ⊂ ABCDH so, we exit the loop

we get out of the while

end

AB^+= ABCDH

Exercise

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- given the relation scheme R = (A, B, C, D, E, H, I) and the following set of functional dependencies on R

  F = { A → E, AB → CD, EH → I, D → H }

  calculate the closure of AB

- the sequence of Z's and S's will be:

  Z= AB

  S= CDE

  while loop

  iter. 1 start with Z= ABCDE, compute S= CDEH

  iter. 2 start with Z= ABCDEH, compute S= CDEHI

  iter. 3 start with Z= ABCDEHI, compute S= CDEHI

AB^+=ABCDEHI AB functionally determines the whole schema ...