Finding the keys of a schema

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.

Keys of a schema

Sapienza
Università di Roma
Dipartimento di Informatica

- we leverage the closure of a set of attributes to determine the keys of a schema R

Example

Sapienza
Università di Roma
Dipartimento di Informatica

- given the following schema:

  R = ABCDEH

and the following set of functional dependencies:

F = { AB $\rightarrow$ CD, C $\rightarrow$ E, AB $\rightarrow$ E, ABC $\rightarrow$ D }

calculate the closure of ABH

by applying the algorithm, **we initialize Z=ABH** and at the **first iteration of the while loop** we add the attributes C, D and E to Z, thanks to the functional dependencies **in which the left part is contained in ABH**, that is AB $\rightarrow$ CD and AB $\rightarrow$ E

actually, at this point we might stop, we already added **all** the attributes of the schema, i.e., we have verified that $(ABH)^+=$ {A,B,C,D,E,H}

Key?

- $ABH^+ = ABCDEH = R$
- **is it a <u>key</u>?**

Definition

Given a relation scheme *R* and a set *F of* functional dependencies

a subset K of a relation scheme R is **a key** of R if:

1. $K \rightarrow R \in F^+$
2. there **is no proper subset** K' **of** K such that $K' \rightarrow R \in F^+$

<u>ATTENTION</u>: **we must always** verify that <u>no</u> subset of K determines functionally R:

- we still use the algorithm on the subsets of K!

Key?

- R = ABCDEH, F = { AB → CD, C → E, AB → E, ABC → D }
- we calculate the closure of **the subsets of ABH**
- **remark 1**: it is better to start from those **with higher cardinality**... **if** their **closure does not contain R**, it is not necessary to check their subsets
- **remark 2**: the attributes that **never appear on the right of the functional dependencies of F**, are not functionally determined by any other attribute, so they must be in all the keys of the schema
- **remark 3: a brute force approach** (try **ALL** subsets anyway**) is not wrong but can be inefficient**
- **remark 4:** in the exercises, every shortcut in the calculations must be **justified**

Key?

- R = ABCDEH, F = { AB → CD, C → E, AB → E, ABC → D }

- H is not determined by other attributes, it must be part of **any** key
- the subsets of cardinality 2 to be checked are **AH and BH**

Key?

$R = ABCDEH$ $F = \{ AB \rightarrow CD, C \rightarrow E, AB \rightarrow E, ABC \rightarrow D \}$

begin

$Z = \mathbf{AH}$

$S = A | Y \rightarrow V \in \mathbf{F}, A \in V \wedge Y \subseteq Z = AH$

while ($S \not\subset Z$) *is false, so the while is not executed*

end $\mathbf{AH^+ = AH \neq R}$

begin

$Z = \mathbf{BH}$

$S = A | Y \rightarrow V \in \mathbf{F}, A \in V \wedge Y \subseteq Z = BH$

while ($S \not\subset Z$) *is false, so the while is not executed*

end $\mathbf{BH^+ = BH}$

Key

we have verified that:

1. $ABH \to R \in F^+$
2. there **is no proper subset** K **of** ABH such that $K \to R \in F^+$

- **ABH is <u>a</u> key of R**

How many keys?

given a schema R and a set of functional dependencies on R, there **can be multiple keys of R**

What are the keys?

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- R = ABCDEH F = { AB → CD, C → E, AB → E, ABC → D }

- we said that H must be included in every key
- we have already verified that H, AH and BH are not keys
- let's check CH, DH and EH

What are the keys?

- R = ABCDEH, F = { AB → CD, C → E, AB → E, ABC → D }

begin

Z = ***CH***

$S = A|Y \rightarrow V \in \mathbf{F}, A \in V \wedge Y \subseteq Z$ = E

while (*S ⊄ Z*): *yes,* we enter the first iteration of while

  begin

    $Z=Z \cup S = CH \cup E$ = ***CEH***

    $S = A|Y \rightarrow V \in \mathbf{F}, A \in V \wedge Y \subseteq Z$ = **E**

  end

while (*S ⊄ Z*): no, we exit the while

CH⁺ = CEH

What are the keys?

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- R =ABCDEH, F = { AB → CD, C → E, AB → E, ABC → D }

begin

Z = ***DH***

$S = A|Y \rightarrow V \in \mathbf{F},\ A \in V \wedge Y \subseteq Z = \varnothing$

while (*S* ⊄ *Z*): no, so we do not enter the while loop

end *DH⁺ = DH*

---

begin

Z = ***EH***

$S = A|Y \rightarrow V \in \mathbf{F},\ A \in V \wedge Y \subseteq Z = \varnothing$

while (*S* ⊄ *Z* ?): no, so we do not enter the while loop

end *EH⁺ = EH*

What are the keys?

- R = ABCDEH, F = { AB → CD, C → E, AB → E, ABC → D }

- we should try other subsets of R with **three attributes including H**
- **but adding A <u>without</u> B or vice versa** does not lead us to have a closure that includes the whole pattern, as the attribute D functionally depends on subsets of R in which **both A and B appear**
- **also, neither A nor B functionally depend on other attributes**, otherwise **we could have** included them in Z in the algorithm, and then we could have included D
- **so, A and B must also be in each key**
- **it follows that ABH is the only key**

Example

given the following schema:

R = ABCDEGH

and the following set of functional dependencies:

$F = \{ AB \rightarrow D, G \rightarrow A, G \rightarrow B, H \rightarrow E, H \rightarrow G, D \rightarrow H \}$

determine the **4 keys** of R

- let's start by attributes that **functionally determine** others, and those that **are NOT** functionally **determined** by others

Example

R = ABCDEGH

F = { AB → D, G → A, G → B, H → E, H → G, D → H }

- AB, G, D and H are good candidates, as hey determine other attributes
- E does **not determine any attribute**, but **is determined** by H
- C does not **determine any attribute, but is not determined by any attributes** either, so **it will appear in all the keys**

We calculate the closures

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

$R = ABCDEGH$, $F = \{ AB \rightarrow D, G \rightarrow A, G \rightarrow B, H \rightarrow E, H \rightarrow G, D \rightarrow H \}$

$ABC^+ = ABCDEGH = R$

$AC^+ = AC$ $BC^+ = BC$

so, ABC is a key

$GC^+ = ABCDEGH = R$ so, GC is a key to R (no need to check the subsets)

$DC^+ = ABCDEGH = R$ so, DC is a key of R (no need to check the subsets)

$HC^+ = ABCDEGH = R$ so, HC is a key of R (no need to check the subsets)

note: ABC has more attributes than the other keys, but it satisfies the minimality condition, as we have verified that <u>its subsets are not key in turn</u>

In conclusion

- to decide "where to start from" to look for keys, we begin with the determinants and individual attributes
- alternatively, we can also begin with the sets identified by the functional dependencies: given a functional dependency $V \rightarrow W \in F$, we calculate the closure of the set of attributes $X = R-(W-V)$
  - internal difference excludes reflexive dependencies
  - external difference excludes dependent attributes
  - these attributes can be taken into account if they appear in the determinant of another dependency
- if the closure of X contains R, we have identified a superkey but we will need to check its minimality

Example

Sapienza Università di Roma
Dipartimento di Informatica

- R = ABCDE
- $F = \{AB \rightarrow C, AC \rightarrow B, D \rightarrow E\}$
- based on the dependencies, we are going to calculate the closures of ABDE, ACDE, ABCD
  - $(ABDE)^+$ = ABDEC = R
  - $(ACDE)^+$ = ACDEB = R
  - $(ABCD)^+$ = ABCDE = R
- are they keys of R?
  - $(AB)^+$ = ABC are missing D and E, but by adding D we also determine E
  - $(ABD)^+$ = ABCDE = R so ABDE and ABCD are not keys
  - $(AC)^+$ = ABC missing D and E, but by adding D we also determine E
  - $(ACD)^+$ = ABCDE =R so ACDE is not key
  - $(D)^+$ = DE
  - $(E)^+$ = E
- the two keys are ABD and ACD

Uniqueness test

- given a relation scheme R and a set of functional dependencies F, we compute the intersection of the sets $X=R-(W-V)$ with $V \rightarrow W \in F$
- if the intersection of these sets determines R, then the intersection is the only key to R
- In our example
  - $(ABDE \cap ACDE \cap ABCD) = AD$ and $AD^+=AD$
  - so, there is more than one key
- if the intersection of these sets does **NOT** determine all of R, then there are multiple keys, and **ALL** of them must be identified for checking 3NF

Example

given the following relationship diagram

$R = ABCDEGH$

and the following set of functional dependencies

$F = \{ AB \rightarrow CD, EH \rightarrow D, D \rightarrow H \}$

determine a key to R, and, knowing that the key is unique, verify that R is not in 3NF

Example

$R = ABCDEGH$, $F = \{ AB \rightarrow CD, EH \rightarrow D, D \rightarrow H \}$

analyzing the dependencies, **we start by considering the sets of attributes that appear to the left of** the dependencies and compute their closures using the algorithm:

$(AB)^+_F = \{A,B,C,D,H\}$ so, AB is not a superkey

$(EH)^+_F = \{E,H,D\}$ so, EH is not a superkey

$(D)^+_F = \{D,H\}$ so D is not a superkey

Example

$R = ABCDEGH$, $F = \{ AB \to CD, EH \to D, D \to H \}$

- now, we observe that **G never appears in the dependencies, so it must surely be part of the key**
- moreover, **the attribute E never appears to the right of** the dependencies, so even this attribute **cannot be determined by the key, unless it is part of it**
- by observing that the closure of AB lacks exactly these two attributes, we compute the closure of the set ABEG:

$(ABEG)^+_F = \{A,B,E,G,C,D,H\}$ so, we have found a superkey

Example

Sapienza Università di Roma
Dipartimento di Informatica

- R = ABCDEGH, F = { AB → CD, EH → D, D → H }
  - we now check the minimality of the superkey
  - in theory, we should compute the closures of all possible subsets of ABEG
  - in practice, we can make some considerations:
  - for example, it is useless to check the closure of subsets that do not contain G
  - so, we start by calculating the closures of ABG, BEG, AEG:

- $(ABG)^{+}_{F}$ = {A,B,G,C,D,H} so, ABG is not a superkey
- $(BEG)^{+}_{F}$ = {B,E,G} so, BEG is not a superkey
- $(AEG)^{+}_{F}$ = {A,E,G} so, AEG is not a super key

Example

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- R = ABCDEGH, F = { AB $\rightarrow$ CD, EH $\rightarrow$ D, D $\rightarrow$ H }

- as there are no three-attribute subsets of ABEG that determine R, it is useless to compute the closures of subsets with fewer attributes

- we have verified both conditions that allow us to state that **ABEG is the key to the given scheme**

Example

- R = ABCDEGH, F = { AB → CD, EH → D, D → H }

- having found the key **ABEG**, it is enough to check the dependence AB → CD to say that R is not in 3NF

- AB → CD is a **partial** dependency (the determinant is properly contained in a key); equivalently, by decomposing it into AB → C and AB → D, the attribute on the right is not prime and AB is not a superkey

Next steps?

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- if a schema is not in 3NF:
  - it must be decomposed
- a "good" decomposition:
  - sub-schemas in 3NF
  - dependencies in F^+ are preserved
  - by joining the instances of the decomposition schemas we obtain legal instances of the source schema (there is no loss of information)