Databases

Lecture 5

Functional Dependencies. Normal Forms (II)

Sabina S. CS

Obs. The following simple properties for functional dependencies can be easily demonstrated:

1. If K is a key of $R[A_1, A_2, \ldots, A_n]$, then $K \to \beta$, $\forall \beta$ a subset of $\{A_1, A_2, \ldots, A_n\}$.

- such a dependency is always true, hence it will not be eliminated through decompositions

2. If $\beta \subseteq \alpha$, then $\alpha \to \beta$ - *trivial functional dependency* (*reflexivity*).

$$\Pi_\alpha(r_1) = \Pi_\alpha(r_2) \underset{\beta \subseteq \alpha}{\Rightarrow} \Pi_\beta(r_1) = \Pi_\beta(r_2) \Rightarrow \alpha \to \beta$$

3. If $\alpha \to \beta$, then $\gamma \to \beta$, $\forall \gamma$ with $\alpha \subset \gamma$.

$$\Pi_\gamma(r_1) = \Pi_\gamma(r_2) \underset{\alpha \subset \gamma, prop. 2}{\Rightarrow} \Pi_\alpha(r_1) = \Pi_\alpha(r_2) \underset{\alpha \to \beta}{\Rightarrow} \Pi_\beta(r_1) = \Pi_\beta(r_2) \Rightarrow \gamma \to \beta$$

Obs. The following simple properties for functional dependencies can be easily demonstrated:

4. If $\alpha \to \beta$ and $\beta \to \gamma$, then $\alpha \to \gamma$ - *transitivity*.

$$\mathit{\Pi}_{\alpha}(r_1) = \mathit{\Pi}_{\alpha}(r_2) \underset{\alpha \to \beta}{\Rightarrow} \mathit{\Pi}_{\beta}(r_1) = \mathit{\Pi}_{\beta}(r_2) \underset{\beta \to \gamma}{\Rightarrow} \mathit{\Pi}_{\gamma}(r_1) = \mathit{\Pi}_{\gamma}(r_2) \Rightarrow \alpha \to \gamma$$

5. If $\alpha \to \beta$ and γ a subset of $\{A_1, \dots, A_n\}$, then $\alpha\gamma \to \beta\gamma$, where $\alpha\gamma = \alpha \cup \gamma$.

$$\Pi_{\alpha\gamma}(r_1) = \Pi_{\alpha\gamma}(r_2) \Rightarrow \left| \begin{array}{l} \Pi_{\alpha}(r_1) = \Pi_{\alpha}(r_2) \Rightarrow \Pi_{\beta}(r_1) = \Pi_{\beta}(r_2) \\ \Pi_{\gamma}(r_1) = \Pi_{\gamma}(r_2) \end{array} \right| \Rightarrow \Pi_{\beta\gamma}(r_1) = \Pi_{\beta\gamma}(r_2)$$

Definition. An attribute A (simple or composite) is said to be *prime* if there is a key K and A ⊆ K (K can be a composite key; A can itself be a key). If an attribute isn't included in any key, it is said to be *non-prime*.

Definition. Let $R[A_1, A_2, \dots, A_n]$ be a relation, and let α, β be two subsets of attributes of R. Attribute β is *fully functionally dependent on* α if:

- β is functionally dependent on α (i.e., $\alpha \rightarrow \beta$) and
- β is not functionally dependent on any proper subset of α, i.e., $\forall \gamma \subset \alpha$, $\gamma \rightarrow \beta$ is not true.

Definition. A relation is in the second normal form (2NF) if:

1. it is in the first normal form and
2. every (simple or composite) non-prime attribute is fully functionally dependent on every key of the relation.

- obs. Let R be a 1NF relation that is not 2NF. Then R has a composite key (and a functional dependency $\alpha \rightarrow \beta$, where α (simple or composite) is a proper subset of a key and β is a non-prime attribute).
- decomposition
  - relation $R[A]$ (A - the set of attributes), K - a key
  - β non-prime, β functionally dependent on α, $\alpha \subset K$ (β is functionally dependent on a proper subset of attributes from a key)
  - the $\alpha \rightarrow \beta$ dependency can be eliminated if R is decomposed into the following 2 relations:

$$R'[\alpha \cup \beta] = \Pi_{\alpha \cup \beta}(R)$$
$$R''[A - \beta] = \Pi_{A - \beta}(R)$$

- we'll analyze the relation from Example 6

EXAM[<u>StudentName, Course</u>, Grade, FacultyMember]

- key: {StudentName, Course}
- the functional dependency *{Course}* → *{FacultyMember}* holds => attribute *FacultyMember* is not fully functionally dependent on a key, hence the EXAM relation is not in 2NF
- this dependency can be eliminated if EXAM is decomposed into the following 2 relations:

RESULTS[<u>StudentName, Course</u>, Grade]

COURSES[<u>Course</u>, FacultyMember]

Example 7. Consider the following relation, storing students' learning contracts:

CONTRACTS[LastName, FirstName, <u>CNP, CourseId</u>, CourseName]

- key: {CNP, CourseId}
- functional dependencies: *{CNP}* → *{LastName, FirstName}*, *{CourseId}* → *{CourseName}*
- to eliminate these dependencies, the relation is decomposed into the following three relations:

STUDENTS[<u>CNP</u>, LastName, FirstName]

COURSES[<u>CourseId</u>, CourseName]

LEARNING_CONTRACTS[<u>CNP, CourseId</u>]

- the notion of *transitive dependency* is required for the third normal form

Definition. An attribute Z is transitively dependent on an attribute X if $\exists Y$ such that $X \rightarrow Y, Y \rightarrow Z, Y \rightarrow X$ does not hold (and Z is not in X or Y).

Definition. A relation is in the third normal form (3NF) if it is in the second normal form and no non-prime attribute is transitively dependent on any key in the relation.

Another definition: A relation R is in the third normal form (3NF) if, for every non-trivial functional dependency $X \rightarrow A$ that holds over R:

- X is a superkey, or
- A is a prime attribute.

Example 8. The BSc examination results are stored in the relation:

BSC_EXAM [<u>StudentName</u>, Grade, Supervisor, Department]

- the relation stores the supervisor and the department in which she works
- since the relation contains data about students (i.e., one row per student), *StudentName* can be chosen as the key
- the following functional dependency holds: *{Supervisor}* → *{Department}* ⇒ the relation is not in 3NF
- to eliminate this dependency, the relation is decomposed into the following 2 relations:

RESULTS [<u>StudentName</u>, Grade, Supervisor]

SUPERVISORS [<u>Supervisor</u>, Department]

Example 9. The following relation stores addresses for a group of people:

ADDRESSES [<u>CNP</u>, LastName, FirstName, ZipCode, City, Street, No]

- key: {CNP}
- identified dependency: *{ZipCode}* → *{City}* (can you identify another dependency in this relation?)
- since this dependency holds, relation ADDRESSES is not in 3NF, therefore it must be decomposed:

ADDRESSES'[<u>CNP</u>, LastName, FirstName, ZipCode, Street, No]

ZIPCODES[<u>ZipCode</u>, City]

Example 10. The following relation stores the exam session schedule:

EX_SCHEDULE[Date, Hour, Faculty_member, Room, Group]

- the following restrictions are expressed via key definitions and functional dependencies:

  1. a group of students has at most one exam per day

  => *{Date, Group}* is a key

  2. on a certain date and time, a faculty member has at most one exam

  => *{Faculty_member, Date, Hour}* is a key

  3. on a certain date and time, there is at most one exam in a room

  => *{Room, Date, Hour}* is a key

  4. a faculty member doesn't change the room in a day

  => the following dependency holds: *{Faculty_member, Date}* $\rightarrow$ *{Room}*

- all attributes appear in at least one key, i.e., there are no non-prime attributes
- given the normal forms' definitions specified thus far, the relation is in 3NF
- objective: eliminate the *{Faculty_member, Date}* → *{Room}* functional dependency

Definition. A relation is in the Boyce-Codd (BCNF) normal form if every determinant (for a functional dependency) is a key (informal definition - simplifying assumption: determinants are not too big; only non-trivial functional dependencies are considered).

- to eliminate the functional dependency, the original relation must be decomposed into:

EX_SCHEDULE'[Date, Hour, Faculty_member, Group],

ROOM_ALLOCATION[Faculty_member, Date, Room]

- these relations don't contain other functional dependencies, i.e., they are in BCNF
- however, the key associated with the 3rd constraint, *{Room, Date, Hour}*, does not exist anymore
- if this constraint is to be kept, it needs to be checked in a different manner (e.g., through the program)

- R[A] - a relation
- F - a set of functional dependencies
- α – a subset of attributes

- problems

I. compute the closure of F: F^+

II. compute the closure of a set of attributes under a set of functional dependencies, e.g., the closure of α under F: α^+

- R[A] - a relation
- F - a set of functional dependencies
- problems

I. compute the closure of F: F^+

- the set F^+ contains all the functional dependencies implied by F
- F implies a functional dependency f if f holds on every relation that satisfies F
- the following 3 rules can be repeatedly applied to compute F^+ (Armstrong's Axioms):
  - α, β, γ - subsets of attributes of A

  1. reflexivity: if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$
  2. augmentation: if $\alpha \rightarrow \beta$, then $\alpha\gamma \rightarrow \beta\gamma$
  3. transitivity: if $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$

- these rules are complete (they compute the closure) and sound (no erroneous functional dependencies can be derived)

- R[A] - a relation
- F - a set of functional dependencies
- problems

I. compute the closure of F: F^+

- the following rules can be derived from Armstrong's Axioms:

4. union: if $\alpha \to \beta$ and $\alpha \to \gamma$, then $\alpha \to \beta\gamma$

$$\left.\begin{array}{l} \alpha \to \beta \underset{\text{augmentation}}{\Rightarrow} \alpha\alpha \to \alpha\beta \\[2em] \alpha \to \gamma \underset{\text{augmentation}}{\Rightarrow} \alpha\beta \to \beta\gamma \end{array}\right\} \underset{\text{transitivity}}{\Rightarrow} \quad \alpha \to \beta\gamma$$

5. decomposition: if $\alpha \to \beta\gamma$, then $\alpha \to \beta$ and $\alpha \to \gamma$

$$\left.\begin{array}{l} \alpha \to \beta\gamma \\ \beta\gamma \to \beta \text{ (reflexivity)} \end{array}\right\} \underset{\text{transitivity}}{\Rightarrow} \alpha \to \beta \text{ (} \alpha \to \gamma \text{ can similarly be shown to hold)}$$

- R[A] - a relation
- F - a set of functional dependencies
- problems

I. compute the closure of F: F^+

- the following rules can be derived from Armstrong's Axioms:

6. pseudotransitivity: if $\alpha \rightarrow \beta$ and $\beta\gamma \rightarrow \delta$, then $\alpha\gamma \rightarrow \delta$

$$\left.\begin{array}{l} \alpha \rightarrow \beta \Rightarrow \alpha\gamma \rightarrow \beta\gamma \\ \beta\gamma \rightarrow \delta \end{array}\right\} \underset{\text{transitivity}}{\Rightarrow} \quad \alpha\gamma \rightarrow \delta$$

- $\alpha, \beta, \gamma, \delta$ - subsets of attributes of A

- R[A] - a relation
- F - a set of functional dependencies
- α – a subset of attributes
- problems

II. compute the closure of a set of attributes under a set of functional dependencies

- determine the closure of α under F, denoted as α^+
- α^+ - the set of attributes that are functionally dependent on attributes in α (under F)

- R[A] - a relation
- F - a set of functional dependencies
- α – a subset of attributes
- problems

II. compute the closure of a set of attributes under a set of functional dependencies

- algorithm

```
closure := α;
repeat until there is no change:
  for every functional dependency β → γ in F
    if β ⊆ closure
      then closure := closure ∪ γ;
```

- see lecture problems
  - R - a relation, F - a set of functional dependencies, f - a functional dependency
    - show that f is in F^+
  - R - a relation, F - a set of functional dependencies, α - a subset of the set of attributes of R
    - compute α^+

References

- [Ta13] ȚÂMBULEA, L., Curs Baze de date, Facultatea de Matematică și Informatică, UBB, 2013-2014
- [Ra00] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems (2nd Edition), McGraw-Hill, 2000
- [Da03] DATE, C.J., An Introduction to Database Systems (8th Edition), Addison-Wesley, 2003
- [Ga08] GARCIA-MOLINA, H., ULLMAN, J., WIDOM, J., Database Systems: The Complete Book, Prentice Hall Press, 2008
- [Ha96] HANSEN, G., HANSEN, J., Database Management And Design (2nd Edition), Prentice Hall, 1996
- [Ra07] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems, McGraw-Hill, 2007, http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed.html
- [Ul11] ULLMAN, J., WIDOM, J., A First Course in Database Systems, http://infolab.stanford.edu/~ullman/fcdb.html