Databases

Lecture 7

Functional Dependencies. Normal Forms (IV)

Relational Algebra

Sabina S. CS

Functional Dependencies. Normal Forms (IV)

- a simple functional dependency $\alpha \rightarrow \beta$ means, by definition, that every value u of α is associated with a unique value v for β

Definition. Let $R[A]$ be a relation with the set of attributes $A = \alpha \cup \beta \cup \gamma$. The multi-valued dependency $\alpha \rightrightarrows \beta$ (read α *multi-determines* β) is said to hold over R iff each value u of α is associated with a set of values v for β: $\beta(u) = \{v_1, v_2, \dots, v_n\}$, and this association holds regardless of the values of γ.

->

- obs. $\sigma_{\alpha=u}(R)$ produces a relation that contains the tuples of R where $\alpha = u$

- let $R[A]$ be a relation, $\alpha \rightrightarrows \beta$ a multi-valued dependency, and $A = \alpha \cup \beta \cup \gamma$, with γ a non-empty set
- the association among the values in $\beta(u)$ for β and the value u of α holds regardless of the values of γ (the context)
- i.e., these associations (between u and an element in $\beta(u)$) exist for any value w in γ:
  - $\forall w \in \Pi_{\gamma}\left(\sigma_{\alpha=u}(R)\right),\ \exists r_1, r_2, \ldots, r_n$ such that $\Pi_{\alpha}(r_i) = u,\ \Pi_{\beta}(r_i) = v_i,\ \Pi_{\gamma}(r_i) = w$

- if $\alpha \rightrightarrows \beta$ and the following rows exist:

α	β	γ
u_1	v_1	w_1
u_1	v_2	w_2

then the following rows must exist as well:

α	β	γ
u_1	v_1	w_2
u_1	v_2	w_1

Property. Let $R[A]$ be a relation, $A = \alpha \cup \beta \cup \gamma$. If $\alpha \rightrightarrows \beta$, then $\alpha \rightrightarrows \gamma$.

Justification.

- Let u be a value of α in R.
- Let $\beta(u) = \Pi_{\beta}\left(\sigma_{\alpha=u}(R)\right)$, $\gamma(u) = \Pi_{\gamma}\left(\sigma_{\alpha=u}(R)\right)$ (the β and γ values in the tuples where $\alpha = u$).

Since $\alpha \rightrightarrows \beta \Rightarrow$

$\forall w \in \gamma(u), \forall v \in \beta(u), \exists r = (u, v, w)$, or

$\forall v \in \beta(u), \forall w \in \gamma(u), \exists r = (u, v, w)$,

therefore $\alpha \rightrightarrows \gamma$.

- for relation DFM' (in the previous example):

{Department} ⇉ {FacultyMember}, {Department} ⇉ {MeetingDate}

Definition. A relation R is in 4NF iff, for every multi-valued dependency $\alpha \rightrightarrows \beta$ that holds over R, one of the statements below is true:

- $\beta \subseteq \alpha$ or $\alpha \cup \beta = R$, or
- α is a superkey.

- trivial multi-valued dependency $\alpha \rightrightarrows \beta$ in relation R: $\beta \subseteq \alpha$ or $\alpha \cup \beta = R$

- if $R[\alpha, \beta, \gamma]$ and $\alpha \rightrightarrows \beta$ (non-trivial, α not a superkey), R is decomposed into the following relations:

$R_1[\alpha, \beta] = \Pi_{\alpha \cup \beta}(R)$

$R_2[\alpha, \gamma] = \Pi_{\alpha \cup \gamma}(R)$

- relation DFM' is decomposed into:

  DF [Department, FacultyMember]

  DM [Department, MeetingDate]

Example 12. Consider relation FaPrCo[FacultyMember, Program, Course], storing the programs and courses for different faculty members

- its key is {FacultyMember, Program, Course}
- this relation has no nontrivial functional dependencies or multi-valued dependencies, it's in 4NF
- consider the following data in the relation:

Fa	Pr	Co
F1	P1	C2
F1	P2	C1
F2	P1	C1
F1	P1	C1

- the relation cannot be decomposed into 2 relations (via projection), because new data would be introduced through the join
- this claim can be justified by considering the three possible projections on two attributes:

FaPr	Fa	Pr
	F1	P1
	F1	P2
	F2	P1

FaCo	Fa	Co
	F1	C2
	F1	C1
	F2	C1

PrCo	Pr	Co
	P1	C2
	P2	C1
	P1	C1

- when evaluating $\mathrm{FaPr} * \mathrm{PrCo}$, the following data is obtained:

$\mathbf{R' = FaPr * PrCo}$	Fa	Pr	Co
	F1	P1	C2
	F1	P1	C1
	F1	P2	C1
	F2	P1	C2
	F2	P1	C1

- this result set contains an extra tuple, which didn't exist in the original relation
- the same is true for the other join combinations: $\mathrm{FaPr} * \mathrm{FaCo}$ and $\mathrm{PrCo} * \mathrm{FaCo}$

- when evaluating $\mathrm{R}' * \mathrm{FaCo}$ (i.e., $\mathrm{FaPr} * \mathrm{PrCo} * \mathrm{FaCo}$), the original relation FaPrCo is obtained
- conclusion: FaPrCo cannot be decomposed into 2 projections, but it can be decomposed into 3 projections, i.e., FaPrCo is *3-decomposable*:

$$\mathrm{FaPrCo} = \mathrm{FaPr} * \mathrm{PrCo} * \mathrm{FaCo}, \text{ or } \mathrm{FaPrCo} = *(\mathrm{FaPr}, \mathrm{PrCo}, \mathrm{FaCo})$$

- this conclusion (FaPrCo is 3-decomposable) is true for the data in the relation
- 3-decomposability can be specified as a constraint:

* *if (F1, P1) ∈ FaPr and (F1, C1) ∈ FaCo and (P1, C1) ∈ PrCo then (F1, P1, C1) ∈ FaPrCo*

- this restriction can be expressed on FaPrCo (all legal instances must satisfy the constraint):

* *if (F1, P1, C2) ∈ $FaPrCo$ and (F1, P2, C1) ∈ $FaPrCo$ and (F2, P1, C1) ∈ $FaPrCo$ then (F1, P1, C1) ∈ $FaPrCo$*

- consider the following relation instance:

Fa	Pr	Co
F1	P1	C2
F1	P2	C1

- if the previous restriction holds, then, if (F2, P1, C1) is added to the relation, (F1, P1, C1) must be also added:

Fa	Pr	Co
F1	P1	C2
F1	P2	C1
F2	**P1**	**C1**
F1	**P1**	**C1**

* what if (F1, P1, C1) is removed from the instance?

Definition. Let $R[A]$ be a relation and $R_i[\alpha_i]$, i=1,2, ...,m, the projections of R on α_i. R satisfies the join dependency $*\{\alpha_1, \alpha_2, ..., \alpha_m\}$ iff $R = R_1 * R_2 * \cdots * R_m$.

- FaPrCo has a join dependency ($\mathrm{FaPrCo} = \mathrm{FaPr} * \mathrm{PrCo} * \mathrm{FaCo}$)

Definition. Relation R is in 5NF iff every non-trivial JD is implied by the candidate keys in R.

- JD $*\{\alpha_1, \alpha_2, ..., \alpha_m\}$ on R is trivial iff at least one α_i is the set of all attributes of R.
- JD $*\{\alpha_1, \alpha_2, ..., \alpha_m\}$ on R is implied by the candidate keys of R iff each α_i is a superkey in R.

=> FaPrCo not in 5NF

- decomposition: projections on FaPr, PrCo, FaCo

Relational Algebra

Sabina S. CS

- query languages in the relational model
  - relational algebra and calculus - formal query languages with a significant influence on SQL
    - relational algebra
      - queries are specified in an operational manner
    - relational calculus
      - queries describe the desired answer, without specifying how it will be computed (declarative)
  - not expected to be Turing complete
  - not intended for complex calculations
  - provide efficient access to large datasets
  - allow optimizations

- relational algebra
  - used by DBMSs to represent query execution plans
  - a relational algebra query:
    - is built using a collection of operators
    - describes a step-by-step procedure for computing the result set
    - is evaluated on the input relations' instances
    - produces an instance of the output relation
  - every operation returns a relation, so operators can be composed; the algebra is closed
  - the result of an algebra expression is a relation, and a relation is a set of tuples
- relational algebra on bags (multisets) - duplicates are not eliminated

Conditions

- conditions that can be used in several algebraic operators
- similar to the SELECT filter conditions

1. *attribute_name relational_operator value*

- *value* - attribute name, expression

2. *attribute_name IS [NOT] IN single_column_relation*

- a relation with one column can be considered a set
- the condition tests whether a value belongs to a set

3. *relation {IS [NOT] IN | = | <>} relation*

- the relations in the condition must be union-compatible

Conditions

4. *(condition)*
NOT condition
condition$_1$ AND condition$_2$
condition$_1$ OR condition$_2$,

where *condition*, *condition$_1$*, *condition$_2$* are conditions of type 1-4.

Operators in the Algebra

- equivalent SELECT statements can be specified for the relational algebra expressions
- *selection*
  - notation: $\sigma_C(R)$
  - resulting relation:
    - schema: R's schema
    - tuples: records in R that satisfy condition C
  - equivalent SELECT statement

```
SELECT *
FROM R
WHERE C
```

- *projection*
  - notation: $\pi_{\alpha}(R)$
  - resulting relation:
    - schema: attributes in α
    - tuples: every record in R is projected on α
  - α can be extended to a set of expressions, specifying the columns of the relation being computed
  - equivalent SELECT statement

```
SELECT DISTINCT α
FROM R

SELECT α
FROM R          -- algebra on bags
```

- *cross-product*
  - notation: $R_1 \times R_2$
  - resulting relation:
    - schema: the attributes of R_1 followed by the attributes of R_2
    - tuples: every tuple r_1 in R_1 is concatenated with every tuple r_2 in R_2
  - equivalent SELECT statement

```
SELECT *
FROM R1 CROSS JOIN R2
```

- *union, set-difference, intersection*
  - notation: $R_1 \cup R_2$, $R_1 - R_2$, $R_1 \cap R_2$
  - R_1 and R_2 must be union-compatible:
    - same number of columns
    - corresponding columns, taken in order from left to right, have the same domains
  - equivalent SELECT statements

```
SELECT *        SELECT *        SELECT *
FROM R1         FROM R1         FROM R1
UNION           EXCEPT          INTERSECT
SELECT *        SELECT *        SELECT *
FROM R2         FROM R2         FROM R2
```

```
-- algebra on bags: SELECT statements that don't eliminate
duplicates (e.g., UNION ALL)
```

- join operators
  - *condition join* (or *theta join*)
    - notation: $R_1 \bigotimes_{\Theta} R_2$
    - result: the records in the cross-product of R_1 and R_2 that satisfy a certain condition
  - definition $\Rightarrow R_1 \bigotimes_{\Theta} R_2 = \sigma_{\Theta}(R_1 \times R_2)$
  - equivalent SELECT statement

```
SELECT *
FROM R1 INNER JOIN R2 ON Θ
```

- join operators
  - *natural join*
    - notation: $R_1 * R_2$
    - resulting relation:
      - schema: the union of the attributes of the two relations (attributes with the same name in R_1 and R_2 appear once in the result)
      - tuples: obtained from tuples <r_1, r_2>, where r_1 in R_1, r_2 in R_2, and r_1 and r_2 agree on the common attributes of R_1 and R_2
    - let $R_1[\alpha]$, $R_2[\beta]$, $\alpha \cap \beta = \{A_1, A_2, \dots, A_m\}$; then:

$$R_1 * R_2 = \pi_{\alpha \cup \beta}(R_1 \otimes_{R_1.A_1=R_2.A_1\ AND\ \dots\ AND\ R_1.A_m=R_2.A_m} R_2)$$

    - equivalent SELECT statement

```
SELECT *
FROM R1 NATURAL JOIN R2
```

References

- [Ta13] ȚÂMBULEA, L., Curs Baze de date, Facultatea de Matematică și Informatică, UBB, 2013-2014
- [Ra02] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems (3rd Edition), McGraw-Hill, 2002
- [Da03] DATE, C.J., An Introduction to Database Systems (8th Edition), Addison-Wesley, 2003
- [Ga09] GARCIA-MOLINA, H., ULLMAN, J., WIDOM, J., Database Systems: The Complete Book (2nd Edition), Pearson Education, 2009
- [Ha96] HANSEN, G., HANSEN, J., Database Management And Design (2nd Edition), Prentice Hall, 1996
- [Ra02S] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems, Slides for the 3rd Edition, http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed.html
- for the 7th Edition, http://codex.cs.yale.edu/avi/db-book/
- [Ul11] ULLMAN, J., WIDOM, J., A First Course in Database Systems, http://infolab.stanford.edu/~ullman/fcdb.html