Database Management System – 33
Database design – Armstrong's Inference rules

Ajay James
Asst. Prof in CSE
Government Engineering College Thrissur

Outline

- Introduction
- Functional Dependencies (recap)
- Armstrong's inference rules
- Closure

- ***Reference: Principles of Database Systems by Elmasri and Navathe, 7th edition, Chapter 15***

Introduction

- Relational design by analysis (top-down)
- Relational design by synthesis (bottom-up)

Functional Dependencies

- A set of attributes X *functionally determines* a set of attributes Y if the value of X determines a unique value for Y.
- X → Y holds if whenever two tuples have the same value for X, they *must have* the same value for Y
  - For any two tuples t1 and t2 in any relation instance r(R): If t1[X]=t2[X], *then* t1[Y]=t2[Y]
  - X →Y in R specifies a *constraint* on all relation instances r(R)

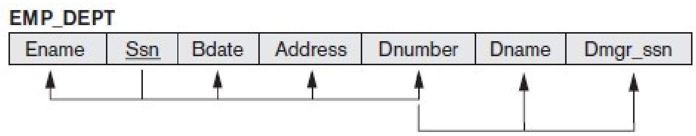
What is Inference for FDs?

- An FD *X → Y is* ***inferred from*** or ***implied by*** a set of dependencies F specified on R
  – if X → Y holds in every legal relation state r of R;
  – that is, whenever r satisfies all the dependencies in F, X → Y also holds in r.
- Given a set of FDs F, we can **infer** additional FDs that hold whenever the FDs in F hold

Example

- Dept_no → Mgr_ssn
- Mgr_ssn → Mgr_phone
- **Inferred from or implied by**
  – **Dept_no → Mgr_phone**

Example



- F = { SSN -> {Ename, Bdate , Address, Dnumber},

  Dnumber -> {Dname, Dmgr_ssn}}
- We can **infer**

  SSN -> Dname,

  SSN -> Dmgr_ssn

  Dnumber -> Dname etc...

Closure of FD

- **Closure** of a set F of FDs is the set $\mathbf{F^+}$ of all FDs that can be inferred from F
- **Inference rules**
  – To determine **a systematic way** to infer dependencies

Notations

- F $\models$ X → Y
  - to denote that the functional dependency X → Y is inferred from the set of functional dependencies F.
- FD {X,Y} → Z is abbreviated to XY → Z
- FD {X, Y, Z} → {U, V} is abbreviated to XYZ → UV

Armstrong's axioms

- Axioms - actually inference rules
- **IR1 (reflexive rule):**
  - **If X ⊇ Y, then X →Y**
  - If Y *subset-of* X, then X →Y
  - set of attributes always determines itself or any of its subsets
  - a functional dependency X → Y is **trivial if** X ⊇ Y; otherwise, it is **nontrivial**

Example for IR1

- Student (<u>Admission_no</u>, Name, Dob, Address)
  – (<u>Admission_no</u>, Name) → Name
  – **Name** is a *subset of* {**Admission_no, Name}**
- The reflexive rule can also be stated as X → X;
- Any set of attributes functionally determines itself
  – **Name → Name**
  – **Admission_no → Admission_no**

IR2 (augmentation rule)

- IR2 (augmentation rule):
  – **{X → Y}** |= **XZ → YZ**
- adding the **same set of attributes** to both the LHS and RHS of a dependency results in another valid dependency
- can also be stated as **X → Y** |= **XZ → Y**
- Augmenting the **left-hand-side attributes** of an FD produces another valid FD.

Example for IR2

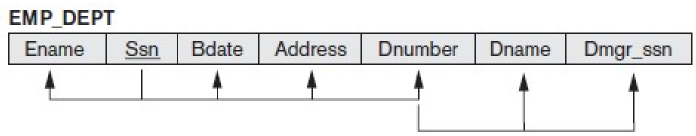
- Student (<u>Admission_no</u>, Name, Dob, Address)
  - Admission_no → Name
- As per IR2
  - {Admission_no, **Dob}** → {Name,**Dob**}

  Or
  - {Admission_no, **Dob**} → Name

IR3 (transitive rule)

- IR3 (transitive rule):
  – {X → Y, Y → Z} |=X → Z

Example for IR3



- F = { SSN -> {Ename, Bdate , Address, Dnumber},

  Dnumber -> {Dname, Dmgr_ssn}}
- We can **infer**

  SSN -> Dname,

  SSN -> Dmgr_ssn

IR1 to IR3 is **Sound**

- Given a set of functional dependencies F specified on R
- Any dependency that is **inferred from F** by using IR1 through IR3 **will hold** in every relation state

IR1 to IR3 is **Complete**

- Applying IR1 through IR3 repeatedly on F
- Results in the complete set of all possible dependencies that can be inferred from F
- **F^+** - **closure of F,** can be determined from F by using only inference rules IR1 through IR3

IR4 (decomposition, or projective rule)

- IR4 (decomposition, or projective, rule):
  - **$\{X \rightarrow YZ\} \models X \rightarrow Y$**
- $X \rightarrow \{A_1, A_2, \ldots , A_n\}$
  - $\{X \rightarrow A_1, X \rightarrow A_2, \ldots , X \rightarrow A_n\}$
- Admission_no → {Name, Dob, Address}
  - Admission_no → Name
  - Admission_no → Dob
  - Admission_no → Address

IR5 (union, or additive, rule)

- IR5 (union, or additive, rule):
  - **{X → Y, X → Z} |=X → YZ**
- {X → A_1, X → A_2, … , X → A_n}
  - X → {A_1, A_2, … , A_n}
- {Admission_no → Name, Admission_no → Dob, Admission_no → Address}
  - Admission_no → {Name, Dob, Address}

IR6 (pseudotransitive rule)

- IR6 (pseudotransitive rule):
  - **{X → Y, WY → Z} |=WX → Z**
- allows us to replace a set of attributes Y on the LHS of a dependency with another set X that functionally determines Y
- Student (Admission_no, Name, Dob, Address)
  - Admission_no → **Dob**, {Name,**Dob**} → Address
  - {Name, **Admission_no**} → Address

Note

- X → A and X → B implies
  - X → AB by the union rule stated above,
- X → A and Y → B implies
  - XY → AB
- XY → A **does not necessarily imply**
  - either X → A or Y → A.

Summary

- IR1 Reflexive
- IR2 Augmentation
- IR3 Transitive
- IR4 Decomposation
- IR5 Union
- IR6 Pseudotransitive

Reference

- Elmasri R. and S. Navathe, Database Systems: Models, Languages, Design and Application Programming, Pearson Education 6th edition and 7th edition

Thank you