Principles of Database Systems

Relational Database Design

Relational Database Design

- **Features of Good Relational Designs**
- **Atomic Domains and First Normal Form**
- **Decomposition Using Functional Dependencies**
- **Functional-Dependency Theory**
- **Algorithms for Decomposition**
- Decomposition Using Multivalued Dependencies
- More Normal Forms
- Database-Design Process
- Modeling Temporal Data

Features of Good Relational Designs

Think…

- Which is better?

instructor(*ID*, *name*, *dept name*, *salary*)
department(*dept name*, *building*, *budget*)

VS

inst dept (*ID*, *name*, *salary*, *dept name*, *building*, *budget*)

Features of Good Relational Designs

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Features of Good Relational Designs

- Are all decompositions(分解) of schemas helpful?

employee (ID, name, street, city, salary)

VS

employee1 (ID, name)
employee2 (name, street, city, salary)

Features of Good Relational Designs

ID	*name*	*street*	*city*	*salary*
⋮				
57766	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000
⋮				

employee

ID	*name*
⋮	
57766	Kim
98776	Kim
⋮	

name	*street*	*city*	*salary*
⋮			
Kim	Main	Perryridge	75000
Kim	North	Hampton	67000
⋮			

natural join

ID	*name*	*street*	*city*	*salary*
⋮				
57766	Kim	Main	Perryridge	75000
57766	Kim	North	Hampton	67000
98776	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000
⋮				

lossless decompositions
(无损分解)

lossy decompositions
(有损分解)

Atomic Domains

Atomic Domains

- The **E-R model** allows entity sets and relationship sets to have attributes that have some degree of substructure(子结构).

- However, when we **create tables** from E-R designs that contain these types of attributes, we eliminate (消除) this substructure.

- A domain is **atomic**(原子的) if elements of the domain are considered to be indivisible(不可分的) units.

Decomposition(分解) Using Functional Dependencies(函数依赖)

Notation

- Greek letters(希腊字母) for sets of **attributes** (for example, α)
- Use a lowercase Roman letter followed by an uppercase Roman letter in parentheses to refer to a **relation schema** (for example, r (R)).
  - use just R when the relation name does not matter to us
- When a set of attributes is a superkey(超码), we denote it by K.

Keys and Functional Dependencies

- Given an instance of $r(R)$, we say that the instance **satisfies** the **functional dependency** $\boldsymbol{\alpha \rightarrow \beta}$ if for all pairs of tuples $t1$ and $t2$ in the instance such that $t1[\alpha] = t2[\alpha]$, it is also the case that $t1[\beta] = t2[\beta]$.
- We say that the functional dependency $\alpha \rightarrow \beta$ **holds** on schema $r(R)$ if, in every legal instance of $r(R)$ it satisfies the functional dependency.

Student(<u>Sno</u>,Sname,Ssex,Sage,Sdetp)

Keys and Functional Dependencies

- We shall use functional dependencies in two ways:
  - To test instances of relations to see whether they satisfy a given set *F* of functional dependencies.
    - 判定关系的实例是否满足给定函数依赖集F
  - To specify constraints on the set of legal relations.
    - 说明合法关系集上的约束

R(Sname,birthday,phone) Sname→birthday

- If we wish to constrain ourselves to relations on schema *r*(*R*) that satisfy a set *F* of functional dependencies, we say that *F* **holds** on *r* (*R*).

Keys and Functional Dependencies

- Example
  - $A \rightarrow C$ is satisfied
  - $C \rightarrow A$ is not satisfied

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_3	c_2	d_3
a_3	b_3	c_2	d_4

Keys and Functional Dependencies

- Let $r(R)$ be a relation schema. A subset K of R is a **superkey** of $r(R)$ if, in any legal instance of $r(R)$, for all pairs $t1$ and $t2$ of tuples in the instance of r if $t1 \neq t2$, then $t1[K] \neq t2[K]$.
- A functional dependency allows us to express constraints that uniquely identify the values of certain attributes.
- Consider a relation schema $r(R)$, and let $\alpha \subseteq R$ and $\beta \subseteq R$.

Keys and Functional Dependencies

- K is a superkey for relation schema R if and only if $K \rightarrow R$
- K is a candidate key for R if and only if
  - $K \rightarrow R$, and
  - for no $\alpha \subset K$, $\alpha \rightarrow R$

- In other words, K is a superkey if, for every legal instance of r (R), for every pair of tuples t1 and t2 from the instance, whenever t1[K] = t2[K], it is also the case that t1[R] = t2[R]

Keys and Functional Dependencies

- Some functional dependencies are said to be **trivial**(平凡的) because they are satisfied by all relations.
  - $A \rightarrow A$
  - $AB \rightarrow A$
- A functional dependency of the form $\alpha \rightarrow \beta$ is **trivial** if $\beta \subseteq \alpha$.
- We will use the notation F^+ to denote the **closure**(闭包) of the set F, that is, the set of all functional dependencies that can be inferred given the set F.

Functional-Dependency Theory

Closure of a Set of Functional Dependencies

- **Armstrong's axioms**(公理)
  - **Reflexivity rule**(自反律). If α is a set of attributes and $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ holds.
  - **Augmentation rule**(增补律). If $\alpha \rightarrow \beta$ holds and γ is a set of attributes, then $\gamma\alpha \rightarrow \gamma\beta$ holds.
  - **Transitivity rule**(传递律). If $\alpha \rightarrow \beta$ holds and $\beta \rightarrow \gamma$ holds, then $\alpha \rightarrow \gamma$ holds.

Closure of a Set of Functional Dependencies

- additional rules
  - **Union rule(合并律)**. If $\alpha \to \beta$ holds and $\alpha \to \gamma$ holds, then $\alpha \to \beta\gamma$ holds.
  - **Decomposition rule(分解律)**. If $\alpha \to \beta\gamma$ holds, then $\alpha \to \beta$ holds and $\alpha \to \gamma$ holds.
  - **Pseudotransitivity rule(伪传递律)**. If $\alpha \to \beta$ holds and $\gamma\beta \to \delta$ holds, then $\alpha\gamma \to \delta$ holds.

Closure of a Set of Functional Dependencies

- Given a relational schema $r(R)$, a functional dependency f on R is **logically implied**(逻辑蕴含) by a set of functional dependencies F on r if every instance of $r(R)$ that satisfies F also satisfies f.
- $r\ (A,\ B,\ C,\ G,\ H,\ I)$
  - $A \rightarrow B$
  - $A \rightarrow C$
  - $CG \rightarrow H$
  - $CG \rightarrow I$
  - $B \rightarrow H$

$$\boxed{A \rightarrow H}$$

Closure of a Set of Functional Dependencies

- Let F be a set of functional dependencies. The **closure**(闭包) of F, denoted by F^+, is the set of all functional dependencies logically implied by F.

Normal Forms

Atomic Domains and First Normal Form

- A domain is **atomic**(原子的) if elements of the domain are considered to be indivisible(不可分的) units.

- We say that a relation schema *R* is **in first normal form** (1NF) if the domains of all attributes of *R* are atomic.

$$R \in 1NF$$

Boyce–Codd Normal Form

- **Boyce–Codd normal form** (**BCNF**), eliminates all redundancy that can be discovered based on functional dependencies
  - BCNF消除所有基于函数依赖能够发现的冗余
- A relation schema R is in BCNF with respect to a set F of functional dependencies if, for all functional dependencies in F^+ of the form $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds:
  - $\alpha \rightarrow \beta$ is a trivial functional dependency (that is, $\beta \subseteq \alpha$).
  - α is a superkey for schema R.

Boyce–Codd Normal Form

- A database design is in BCNF if each member of the set of relation schemas that constitutes the design is in BCNF.

- *EXAMPLE*
  - *inst_dept* (*ID*, *name*, *salary*, *dept_name*, *building*, *budget*)
  - Is it in BCNF?
    - ——NO
    - *dept_name* → *budget*

  - *instructor(ID, name, dept_name, salary)*
  - *department(dept_name, building, budget)*

Boyce–Codd Normal Form

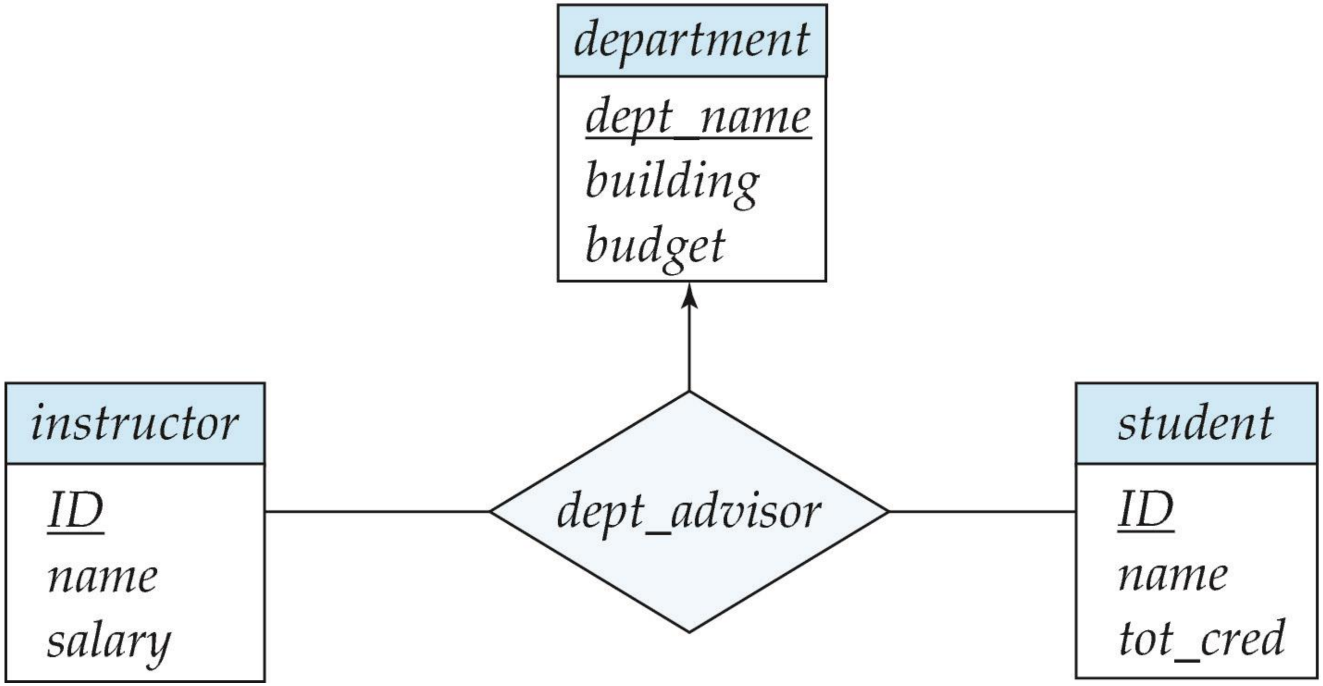
- We now state a general rule for decomposing(分解) that are not in BCNF.
  - Let R be a schema that is not in BCNF.
  - Then there is at least one nontrivial functional dependency $\alpha \rightarrow \beta$ such that α is not a superkey for R.
  - We replace R in our design with two schemas:
    - $(\alpha \cup \beta)$
    - $(R - (\beta - \alpha))$

Boyce–Codd Normal Form

- In the case of *inst_dept* above
  - α = *dept_name*
  - β= {*building*, *budget*}
- then *inst_dept* is replaced by
  - $(\alpha \cup \beta) = (dept\_name,\ building,\ budget)$
  - $(R - (\beta - \alpha)) = (ID,\ name,\ dept\_name,\ salary)$

BCNF and Dependency Preservation

- In some cases, decomposition(分解) into BCNF can prevent efficient testing of certain functional dependencies.



BCNF and Dependency Preservation

- *dept_advisor* (*s_ID*, *i_ID*, *dept_name*)
  - "an instructor can act as advisor for only a single department."
  - "a student may have more than one advisor, but at most one corresponding to a given department".

$$i\_ID \rightarrow dept\_name$$

$$s\_ID,\ dept\_name \rightarrow i\ ID$$

- We see that *dept_advisor* is not in BCNF because *i ID* is not a superkey.

BCNF and Dependency Preservation

- We see that *dept_advisor* is not in BCNF because *i_ID* is not a superkey. Following our rule for BCNF decomposition, we get:
  - (*s_ID*, *i_ID*)
  - (*i_ID*, *dept_name*)

BCNF and Dependency Preservation

- However, in our BCNF design, there is no schema that includes all the attributes appearing in the functional dependency ***s_ID, dept_name→i_ID***.

- We say our design is **not dependency preserving**(不是保持依赖的).

Third Normal Form

- A relation schema R is in **third normal form** with respect to a set F of functional dependencies if, for all functional dependencies in F^+ of the form $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds:
  - $\alpha \rightarrow \beta$ is a trivial functional dependency.
  - α is a superkey for R.
  - **Each attribute A in $\beta - \alpha$ is contained in a candidate key for R.**

Third Normal Form

- The definition of 3NF allows certain functional dependencies that are not allowed in BCNF.

- A dependency $\alpha \rightarrow \beta$ that satisfies only the third alternative of the 3NF definition is not allowed in BCNF, but is allowed in 3NF.

Third Normal Form

- Now, let us again consider the *dept_advisor* relationship set, which has the following functional dependencies:
  - $i\_ID \rightarrow dept\ name$
  - $s\_ID,\ dept\_name \rightarrow i\ ID$
- $\alpha = i\ ID$, $\beta = dept\_name$, and $\beta - \alpha =\ dept\_name$.
  - ***dept_advisor* is in 3NF.**

Closure of Attribute Sets

- We say that an attribute B is **functionally determined**(函数确定) by α if $\alpha \rightarrow B$.
- To test whether a set α is a superkey, we must devise an algorithm for computing the set of attributes functionally determined by α.
- One way of doing this is to compute F^+, take all functional dependencies with α as the left-hand side, and take the union of the right-hand sides of all such dependencies.

Closure of Attribute Sets

- Let α be a set of attributes.
- We call the set of all attributes functionally determined by α under a set F of functional dependencies the **closure**(闭包) of α under F;
- We denote it by α^+.

$r\ (A, B, C, G, H, I)$

$A \rightarrow B$

$A \rightarrow C$

$CG \rightarrow H$

$CG \rightarrow I$

$B \rightarrow H$

Closure of Attribute Sets

- There are several uses of the attribute closure algorithm:
  - To test if α is a superkey, we compute α^+, and check if α^+ contains all attributes in R.
  - We can check if a functional dependency $\alpha \rightarrow \beta$ holds (or, in other words, is in F^+), by checking if $\beta \subseteq \alpha^+$.
  - It gives us an alternative way to compute F^+: For each $\gamma \subseteq R$, we find the closure γ^+, and for each $S \subseteq \gamma^+$, we output a functional dependency $\gamma \rightarrow S$.

Lossless Decomposition

- We say that the decomposition is a **lossless decomposition**(无损分解) if there is no loss of information by replacing $r(R)$ with two relation schemas $r1(R1)$ and $r2(R2)$.

```
select *
from (select R1 from r)
        natural join
        (select R2 from r)
```

$$\Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) = r$$

Algorithms for Decomposition

BCNF Decomposition

- Testing for BCNF
  - To check if a nontrivial(非平凡) dependency $\alpha \rightarrow \beta$ causes a violation of BCNF, compute α^+ (the attribute closure of α), and verify that it includes all attributes of R; that is, it is a superkey of R.
  - To check if a relation schema R is in BCNF, it suffices to check only the dependencies in the given set F for violation of BCNF, rather than check all dependencies in F^+.

- R（A, B, C, D）

  D→A, C→A, C→D, B→C

  候选码是? 是否满足BCNF?

Thanks