Department of Computer Science

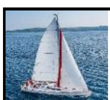
CS 44800: Introduction To Relational Database Systems

Prof. Chris Clifton

2 September 2021

Relational Database Design

Indiana Center for Database Systems ™



Features of Good Relational Designs

- Suppose we combine *instructor* and *department* into *in_dep,* which represents the natural join on the relations *instructor* and *department*

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

- There is repetition of information
- Need to use null values (if we add a new department with no instructors)

Decomposition

- The only way to avoid the repetition-of-information problem in the i*n_dep* schema is to decompose it into two schemas – instructor and *department* schemas.
- Not all decompositions are good. Suppose we decompose

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

  into

  employee1 (*ID*, *name*)

  employee2 (*name*, *street, city, salary*)

  The problem arises when we have two employees with the same name
- The next slide shows how we lose information -- we cannot reconstruct the original *employee* relation -- and so, this is a **lossy decomposition**.

A Lossy Decomposition

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 Decomposition

- Let R be a relation schema and let R_1 and R_2 form a decomposition of R
  - That is R = R_1 U R_2
- We say that the decomposition is a **lossless decomposition** if there is no loss of information by replacing R with the two relation schemas R_1 U R_2
- Formally,

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

- And, conversely a decomposition is lossy if

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

Example of Lossless Decomposition

- Decomposition of $R = (A, B, C)$
  $R_1 = (A, B)$ $R_2 = (B, C)$

A	B	C
α	1	A
β	2	B

r

A	B
α	1
β	2

$\Pi_{A,B}(r)$

B	C
1	A
2	B

$\Pi_{B,C}(r)$

$\Pi_A(r) \bowtie \Pi_B(r)$

A	B	C
α	1	A
β	2	B

Normalization Theory

- Decide whether a particular relation *R* is in "good" form.
- In the case that a relation *R* is not in "good" form, decompose it into set of relations $\{R_1, R_2, ..., R_n\}$ such that
  - Each relation is in good form
  - The decomposition is a lossless decomposition
- Our theory is based on:
  - Functional dependencies
  - Multivalued dependencies

Functional Dependencies

- There are usually a variety of constraints (rules) on the data in the real world.
- For example, some of the constraints that are expected to hold in a university database are:
  - Students and instructors are uniquely identified by their ID.
  - Each student and instructor has only one name.
  - Each instructor and student is (primarily) associated with only one department.
  - Each department has only one value for its budget, and only one associated building.

Department of Computer Science

Functional Dependencies

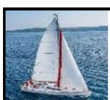
$X \rightarrow A$ = assertion about a relation R that whenever two tuples agree on all the attributes of X, then they must also agree on attribute A

Why do we care?

Knowing functional dependencies provides a formal mechanism to divide up relations *(normalization)*

Saves space

Prevents storing data that violates dependencies



Functional Dependencies Definition

- Let R be a relation schema

$$\alpha \subseteq R \text{ and } \beta \subseteq R$$

- The **functional dependency**

$$\alpha \rightarrow \beta$$

holds on R if and only if for any legal relations r(R), whenever any two tuples t_1 and t_2 of r agree on the attributes α, they also agree on the attributes β. That is,

$$t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]$$

- Example: Consider $r(A,B)$ with the following instance of r.

1	4
1	5
3	7

- On this instance, $B \rightarrow A$ hold; $A \rightarrow B$ does **NOT** hold,

Keys of Relations

K is a *key* for relation *R* if:

1. *K* → all attributes of *R*. **(Uniqueness)**
2. For no proper subset of *K* is (1) true. **(Minimality)**

- If *K* at least satisfies (1), then *K* is a *superkey*.

Conventions

- Pick one key; underline key attributes in the relation schema.
- *X*, etc., represent sets of attributes; *A* etc., represent single attributes.
- No set formers in FD's, *e.g.*, *ABC* instead of {*A*, *B*, *C*}.

Keys and Functional Dependencies

- Functional dependencies allow us to express constraints that cannot be expressed using superkeys. Consider the schema:

  in_dep (*ID*, *name*, *salary*, *dept_name*, *building*, *budget*).

  We expect these functional dependencies to hold:

  dept_name → *building*

  ID → *building*

  but would not expect the following to hold:

  dept_name → *salary*

PURDUE UNIVERSITY
Department of Computer Science

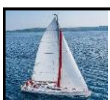
Example

$\underbrace{\text{Lastname} \quad \text{Firstname}}_{\text{Key (2 attributes)}} \qquad \underbrace{\underbrace{\text{Student ID}}_{\text{Key}} \quad \text{Major}}_{\text{Superkey}}$

Note: There are <u>alternate</u> keys

- Keys are {`Lastname, Firstname`} and {`StudentID`}

30



Use of Functional Dependencies

- We use functional dependencies to:
  - To test relations to see if they are legal under a given set of functional dependencies.
    - If a relation *r* is legal under a set *F* of functional dependencies, we say that *r* **satisfies** *F.*
  - To specify constraints on the set of legal relations
    - We say that *F* **holds on** *R* if all legal relations on *R* satisfy the set of functional dependencies *F.*
- Note: A specific instance of a relation schema may satisfy a functional dependency even if the functional dependency does not hold on all legal instances.
  - For example, a specific instance of *instructor* may, by chance, satisfy
    $name \rightarrow ID.$

Trivial Functional Dependencies

- *A* functional dependency is **trivial** if it is satisfied by all instances of a relation
- Example*:*
  - *ID, name* $\rightarrow$ *ID*
  - *name* $\rightarrow$ *name*
- In general, $\alpha \rightarrow \beta$ is trivial if $\beta \subseteq \alpha$

PURDUE UNIVERSITY
Department of Computer Science

Who Determines Keys/FD's?

- We could assert a key *K*.
  - Then the only FD's asserted are that $K \rightarrow A$ for every attribute *A*.
  - No surprise: *K* is then the only key for those FD's, according to the formal definition of "key."
- Or, we could assert some FD's and *deduce* one or more keys by the formal definition.
- Rule of thumb: FD's either come from keyness, many-1 relationship, or from physics.
  - *E.g.*, "no two courses can meet in the same room at the same time" yields `room time` $\rightarrow$ `course`.

PURDUE UNIVERSITY
Department of Computer Science

Functional Dependencies (FD's) and Many-One Relationships

- Consider $R(A1,\ldots, An)$ and X is a key
  then $X \rightarrow Y$ for any attributes Y in $A1,\ldots, An$
  even if they overlap with X. Why?
- Suppose R is used to represent a many $\rightarrow$ one relationship:
  $E1$ entity set $\rightarrow$ $E2$ entity set
  where X key for $E1$, Y key for $E2$,
  Then, $X \rightarrow Y$ holds,
  And $Y \rightarrow X$ does not hold unless the relationship is one-one.
- What about many-many relationships?

34

PURDUE UNIVERSITY
Department of Computer Science

Inferring FD's

And this is important because …

- When we talk about improving relational designs, we often need to ask "does this FD hold in this relation?"

Given FD's $X1 \rightarrow A1$, $X2 \rightarrow A2,\ldots, Xn \rightarrow An$, does FD $Y \rightarrow B$ necessarily hold in the same relation?

- Start by assuming two tuples agree in Y. Use given FD's to infer other attributes on which they must agree. If B is among them, then yes, else no.

35