CSCI 403 Database Management

18 – Normalization

This Lecture

- Normalization
- Boyce-Codd Normal Form

Example Relation

Figure 1: One possible relation storing Mines course information:

Instructor	Course_id	Section	Title	Office	Email
Painter-Wakefield, Christopher	CSCI403	A	DATABASE MANAGEMENT	BB 280I	cpainter@mines.edu
Painter-Wakefield, Christopher	CSCI262	A	DATA STRUCTURES	BB 280I	cpainter@mines.edu
Painter-Wakefield, Christopher	CSCI262	B	DATA STRUCTURES	BB 280I	cpainter@mines.edu
Mehta, Dinesh	CSCI406	A	ALGORITHMS	BB 280J	dmehta@mines.edu
Mehta, Dinesh	CSCI 561	A	THEORY OF COMPUTATION	BB 280J	dmehta@mines.edu
Hellman, Keith	CSCI 101	A	INTRO TO COMPUTER SCIENCE	BB 310F	khellman@mines.edu
Hellman, Keith	CSCI 101	B	INTRO TO COMPUTER SCIENCE	BB 310F	khellman@mines.edu
Hellman, Keith	CSCI 101	C	INTRO TO COMPUTER SCIENCE	BB 310F	khellman@mines.edu
Hellman, Keith	CSCI 274	A	INTRO TO LINUX OS	BB 310F	khellman@mines.edu

Functional Dependencies Review

- Our primary tool for eliminating redundancy and modification anomalies
- A kind of constraint between two sets of attributes in a relation schema
- Definition:

  Given a relation schema R and sets of attributes X and Y, then we say a functional dependency X → Y exists if, whenever tuples t_1 and t_2 are two tuples from any relation r(R) such that $t_1[X] = t_2[X]$, it is also true that $t_1[Y] = t_2[Y]$.
- The lingo:

  We say X functionally determines Y, or Y is functionally dependent on X.

Functional Dependencies Review 2

- In other words:

  If it is always true that whenever two tuples agree on attributes X, they also agree on Y, then X → Y.
- Example:

  If we assert that an instructor is always associated with one office and email, then

  $\underbrace{\{\text{ instructor }\}}_{X} \rightarrow \underbrace{\{\text{ office, email }\}}_{Y}$

  is a functional dependency (FD) on the example table in figure 1.

Normal Forms

- Developed to define "good" design for a database
- Several forms: First normal form (1NF), Second (2NF), etc.
- Each normal form describe certain properties of a database
  - E.g., 1NF eliminates multivalued and compound attributes
  - Mostly later normal forms subsume earlier normal forms
- 1NF – 3NF are not terribly interesting stepping stones to the forms we care about:
  - Boyce-Codd Normal Form (BCNF)
  - Fourth Normal Form (4NF)
- There exist even stronger normal forms (5NF etc.), but BCNF and 4NF suffice for most purposes.

Boyce-Codd Normal Form

Definition:

A relation *R* is in Boyce-Codd Normal Form (BCNF) if for every nontrivial functional dependency $X \rightarrow A$ on *R*, *X* is a superkey of *R*.

BCNF Example

Consider our example relation schema in Figure 1:

One of the (non-trivial) functional dependencies we identified was

instructor → office

Clearly, instructor is not a superkey of the relation.

Therefore, we say that the FD instructor → office *violates* BCNF, and the relation schema is not in BCNF.

Moving to BCNF

Our goal is a database in which every relation is in BCNF.

Fortunately, there is a straightforward algorithm for getting there.

- Find a relation schema not in BCNF
- Decompose it into two relation schemas, eliminating one of the BCNF violations
- Repeat

(Details on next page)

Decomposition Algorithm

while some relation schema is not in BCNF:

- choose some relation schema *R* not in BCNF
- choose some FD $X \rightarrow Y$ on *R* that violates BCNF
- (optional) expand *Y* so that $Y = X^+$ (closure of X)
- let *Z* be all attributes of *R* not included in *X* or *Y*
- replace *R* with two new relations:
  - *R1*, containing attributes {*X*, *Y*}
  - *R2*, containing attributes {*X*, *Z*}

Decomposition Notes

- The final step above is accomplished simply by projection onto the attributes in *R1* and *R2*. (Recall that this may result in fewer tuples.)
- After decomposing, you will need to establish which FDs now apply to *R1* and *R2*, as well as determine their superkeys, in order to determine if they are now in BCNF.
- The optional step of expanding *Y* is recommended, as it tends to result in fewer, larger relation schemas, and may reduce the problem faster - e.g., consider decomposing on instructor → office.

Decomposition Walkthrough

Let's use the relation schema in Figure 1 as an example.

For this schema, we listed the following FD's:

- instructor → office — violates BCNF
- instructor → email — violates BCNF
- {course_id, section} → instructor — does not violate BCNF
- course_id → title — violates BCNF

What superkeys do we have?

Answer: any superset of our only key, which is {course_id, section}.

Which FD's violate BCNF?

Walkthrough 2

- Let's pick our first violating FD to work with first: instructor → office
- Next, expand the RHS as much as possible (we want the closure of instructor):
  instructor → {instructor, office, email}
- Now we decompose into two new tables, shown on the next slide:
  - R1 = $\pi_{\text{instructor,office,email}}$ (R)
  - R2 = $\pi_{\text{instructor,course\_id,section,title}}$ (R)
- We can now discard the table from figure 1

CS@Mines

Tables After One Step

R1:

Instructor	Office	Email
Painter-Wakefield, Christopher	BB 280I	cpainter@mines.edu
Mehta, Dinesh	BB 280J	dmehta@mines.edu
Hellman, Keith	BB 310F	khellman@mines.edu

R2:

Instructor	Course_id	Section	Title
Painter-Wakefield, Christopher	CSCI403	A	DATABASE MANAGEMENT
Painter-Wakefield, Christopher	CSCI262	A	DATA STRUCTURES
Painter-Wakefield, Christopher	CSCI262	B	DATA STRUCTURES
Mehta, Dinesh	CSCI406	A	ALGORITHMS
Mehta, Dinesh	CSCI 561	A	THEORY OF COMPUTATION
Hellman, Keith	CSCI 101	A	INTRO TO COMPUTER SCIENCE
Hellman, Keith	CSCI 101	B	INTRO TO COMPUTER SCIENCE
Hellman, Keith	CSCI 101	C	INTRO TO COMPUTER SCIENCE
Hellman, Keith	CSCI 274	A	INTRO TO LINUX OS

CS@Mines

Walkthrough 3

- Table *R1* is now in BCNF (yay!)
  - Note this was not guaranteed by the algorithm – we could have had other violating FDs
- Table *R2* has a violating FD, though: course_id → title

CS@Mines

Walkthrough 4

Decomposition of *R2* via course_id → title:

course_id$^+$ = {course_id, title}

Decompose into *R3* and *R4*:

- R3 = $\pi_{\text{course\_id,title}}$(R2)
- R4 = $\pi_{\text{instructor,course\_id,section}}$(R2)

R3:

Course_id	Title
CSCI403	DATABASE MANAGEMENT
CSCI262	DATA STRUCTURES
CSCI406	ALGORITHMS
CSCI 561	THEORY OF COMPUTATION
CSCI 101	INTRO TO COMPUTER SCIENCE
CSCI 274	INTRO TO LINUX OS

R4:

Instructor	Course_id	Section
Painter-Wakefield, Christopher	CSCI403	A
Painter-Wakefield, Christopher	CSCI262	A
Painter-Wakefield, Christopher	CSCI262	B
Mehta, Dinesh	CSCI406	A
Mehta, Dinesh	CSCI 561	A
Hellman, Keith	CSCI 101	A
Hellman, Keith	CSCI 101	B
Hellman, Keith	CSCI 101	C
Hellman, Keith	CSCI 274	A

CS@Mines

Walkthrough Wrap-up

- Done!
  - Three tables remain: R1, R3, R4
  - All non-essential redundancy has been removed
  - Each table now represents a fundamental entity:
    - R1 = instructor info
    - R3 = course info
    - R4 = section info
- As a final note: this algorithm is not deterministic – you can different decompositions following different choices of FD to work with.

CS@Mines

Next Time

- Multi-valued dependencies and 4NF

CS@Mines