Database Management Systems

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Sai University

Lecture 11, 22 September 2023

Relational database design

- Set of attributes that one needs to keep track of
- Split into multiple tables to avoid duplication
  - Redundant storage
  - Maintaining consistency — updates and insertion/deletion

One source of "truth"

Decomposition and information

- Decompose `(customer_name,regd_phone,regd_email)` as `(customer_name,regd_phone)` and `(customer_name,regd_email)`
- Name is not unique — loss of information
- Recombining decomposed relation should not add tuples
- Lossless decomposition
  - Decompose R as R_1 and R_2
  - Want $R = R_1 \bowtie R_2$
- Decomposition is lossless if at least one of the following functional dependencies hold
  - $\boxed{R_1 \cap R_2} \rightarrow R_1$
  - $R_1 \cap R_2 \rightarrow R_2$

A	P1	E1
A	P2	E2

$$\begin{matrix} A & P1 \\ A & P2 \end{matrix} \bowtie \begin{matrix} A & E1 \\ A & E2 \end{matrix}$$

$R \subseteq R_1 \bowtie R_2$

Functional dependencies

- $A_1, A_2, \ldots, A_k \rightarrow B_1, B_2, \ldots B_m$
  - LHS atributes uniquely fix RHS attributes
  - Must hold for every instance — semantic property of attributes
- Need not correspond to superkeys
  - `dept_name` $\rightarrow$ `building`
  - `dept_name` $\rightarrow$ `budget`
- Use to identify sources of redundancy, guide decomposition

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

Computing the closure of a set of attributes

- Given $A_1, A_2, \ldots, A_k$ and B, does $A_1, A_2, \ldots, A_k \rightarrow B$?
- Iterative algorithm — check if B is in closure A^+

Initialize A^+ to $\{A_1, A_2, \ldots, A_k\}$
repeat
  for each $\beta \rightarrow \gamma$ in F
    if $\beta \subseteq A^+$, add γ to A^+
  end
until no change in A^+

$F = \{\alpha_1 \rightarrow \beta_1,$
$\alpha_2 \rightarrow \beta_2,$
$\vdots$

$A_3 \rightarrow C$ $C \rightarrow B$

α, β are set of attributes

Is $B \in A^+$?

$\dfrac{R_1 \cap R_2}{\alpha} \rightarrow \dfrac{R_1}{\beta}$

$B_1, B_2 \ldots, B_n$

$\alpha \rightarrow B_1$
$\alpha \rightarrow B_1$
$\vdots$

$$\underline{A_1 A_2 \ldots A_k \to B_1 B_2 \ldots B_m} \in F$$

$$\boxed{\begin{array}{l} (a_1, a_2, \ldots a_k) \rightsquigarrow (b_1, b_2, \ldots, b_m) \\ (a_1, a_2 \ldots a_k) \rightsquigarrow (b_1', b_2 \ldots b_m) \end{array}} \quad \text{Would violate}$$

$$\Downarrow$$

$$\left.\begin{array}{l} A_1 A_2 \ldots A_k \to B_1 \\ A_1, A_2 \ldots A_k \to B_2 \\ \vdots \\ A_1, A_2 \ldots A_k \to B_m \end{array}\right] \Rightarrow A_1 A_2 \ldots A_k \to B_1 \ldots B_m$$

$$\begin{array}{l} \alpha \to B_1 \\ \alpha \to B_2 \\ \vdots \\ \alpha \to B_m \\ \hline \hline \alpha \to \beta = B_1 \ldots B_m \end{array}$$

Boyce-Codd Normal Form (BCNF)

- Relational schema R, set of functional dependencies F
- R is in BCNF if, for every $\alpha \to \beta \in F^+$, one of the following holds
  - $\alpha \to \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
  - α is a superkey for R

Boyce-Codd Normal Form (BCNF)

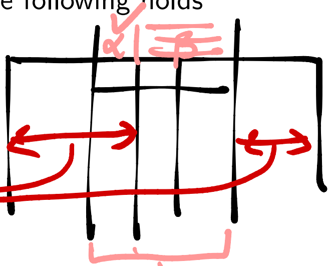
- Relational schema R, set of functional dependencies F
- R is in BCNF if, for every $\alpha \to \beta \in F^+$, one of the following holds
  - $\alpha \to \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
  - α is a superkey for R
- $\alpha \to \beta \in F^+$ is a BCNF violation for R if neither of the following holds
  - $\alpha \to \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
  - α is a superkey for R

Boyce-Codd Normal Form (BCNF)

- Relational schema R, set of functional dependencies F
- R is in BCNF if, for every $\alpha \to \beta \in F^+$, one of the following holds
  - $\alpha \to \beta$ is trivial (i.e., $\beta \subseteq \alpha$
  - α is a superkey for R

$A_1, A_2 \to A_2, B_3$

- $\alpha \to \beta \in F^+$ is a BCNF violation for R if neither of the following holds
  - $\alpha \to \beta$ is trivial (i.e., $\beta \subseteq \alpha$
  - α is a superkey for R



- To fix this, decompose R as
  - R_1: $\alpha \cup \beta$
  - R_2: $R \setminus (\beta \setminus \alpha)$

$R \setminus (\beta \setminus \alpha)$

$\alpha \cup \beta$

$R_1 \cap R_2 = \alpha$

α key

Dependency preservation

- Advisor(student_id, faculty_id, dept_name)
- Each faculty member is in only one department
- Students can be across multiple departments
- Each student has at most one advisor in each department
- BCNF decomposition is (student_id, faculty_id), (faculty_id, dept_name)
- Functional dependencies
  - faculty_id → dept_name
  - student_id, dept_name → faculty_id
- Need join to check second dependency

BCNF violation

$\alpha \to \beta$

$R - (\beta - \alpha)$

$\alpha \cup \beta$

Third normal form (3NF)

- R is in 3NF if, for every $\alpha \to \beta \in F^+$, one of the following holds

BCNF

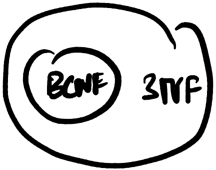
  - $\alpha \to \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
  - α is a superkey for R
  - Each attribute A in $\beta \setminus \alpha$ is contained in some candidate key for R
- BCNF is a stricter condition than 3NF
- Priorities
  - Lossless decomposition
  - BCNF
  - Dependency preservation

Adv(stud, fac, dept)

fac → dept

stud, dept → fac

key



Tradeoff redundancy for local dependency checks

3NF → Redundancy

S1	F1	D1
S2	F1	D1

2 copies

S1	F1
S2	F1

F1	D1

Computing the closure of a set of attributes — correctness

- Iterative algorithm — check if B is in closure A^+

Initialize A^+ to $\{A_1, A_2, \ldots, A_k\}$
repeat
  for each $\beta \rightarrow \gamma$ in F
    if $\beta \subseteq A^+$, add γ to A^+
  end
until no change in A^+

Given F

All attributes fixed by A_1 — A_k

Correctness ?

Soundness – No wrong attribute is added to A^+

Completeness – Do not miss any attribute in A^+

Computing the closure of a set of attributes — correctness

- Iterative algorithm — check if B is in closure A^+

Initialize A^+ to $\{A_1, A_2, \ldots, A_k\}$
repeat
  for each $\beta \rightarrow \gamma$ in F
    if $\underline{\beta \subseteq A^+}$, add γ to A^+
  end
until no change in A^+

Soundness

By induction

After 0 iterations

$A^+ = \{A_1 \ldots, A_k\}$ — trivially fixed

$A_1, A_2, \ldots, A_k \rightarrow \beta$ (by induction hypothesis)

$+$

$\beta \rightarrow \gamma \in F$

$\Rightarrow$ Transitivity $A_1 \ldots A_k \rightarrow \gamma$

Step i

$\underbrace{A_1 \ldots A_k, C_1, \ldots, C_l}_{\beta} \longrightarrow \gamma$

Computing the closure of a set of attributes — correctness

- Iterative algorithm — check if B is in closure A^+

Initialize A^+ to $\{A_1, A_2, \ldots, A_k\}$
repeat
  for each $\beta \to \gamma$ in F
    if $\beta \subseteq A^+$, add γ to A^+
  end
until no change in A^+

Completeness

Every B fixed by $A_1 .. A_k$ is in A^+

If $B \notin A^+$ then it can vary

Is there a relation/table that satisfies all of F but does not have $A_1 .. A_k \to B$

Construct such a table

Computing the closure of a set of attributes — correctness

- Iterative algorithm — check if B is in closure A^+

  Initialize A^+ to $\{A_1, A_2, \ldots, A_k\}$
  repeat
      for each $\beta \to \gamma$ in F
          if $\beta \subseteq A^+$, add γ to A^+
      end
  until no change in A^+

A^+							Other Attributes		
A_1	A_2	..	A_k	C_1	–	C_m	B_1	- - -	-
a_1	a_2	-- .		c_1		c_m	0		0
a_1	a_2	-- .		c_1	- -	c_m	1		1

Demonstrates that $A_1, \ldots A_k \not\to B$

But does it meet all f.d. in F?

Suppose $\beta \to \gamma \in F$ is violated in this table

$\Rightarrow \beta \to D$ is violated

Here β is same, D is different in 2 rows

$D \notin A^+ \Rightarrow A^+$ was not calculated well!

$\beta \in A^+$

$D \in$ RHS

Madhavan Mukund Database Management Systems DBMS, Lecture 11, 22 Sep 2023 9 / 14

Canonical basis, extraneous attributes

When populating a database, need to check constraints

Constraints are functional dependence F, need to check all constraints in F^+ (closure of F)

F: $A \to B$, $B \to C$ $\Rightarrow$ F^+: $A \to C$

A	C

B	C

check this $\longrightarrow$ need not check

enforce $A \to C$

Canonical basis, extraneous attributes

Find a "minimal" set of rules to check st all of F^+ is covered

Extraneous attributes

$A, B \rightarrow C, D$

stronger

Suppose $A \rightarrow C, D$

B is extraneous in LHS

Can B be removed?

Can I derive

$A \rightarrow C, D$

from F ?

Canonical basis, extraneous attributes

$A,B \to C,D$

$\downarrow$

$A,B \to C$

alone is weaker

$A,B \to D$

Preserves F^+

Check if

$F \setminus (A,B \to C,D)$

$+ \; A,B \to C$

$\Rightarrow A,B \to C,D$?

Canonical basis, extraneous attributes

Remove extraneous attribute → minimal set of dependencies that have same closure F^+ as original F

$F = \{ AB \to CD,\ A \to E,\ E \to C \}$

$\{AB \to D,\ A \to E,\ E \to C\}$.

vs $AB \to CD$

$AB \to CD$, eliminate C?

$AB \to D$

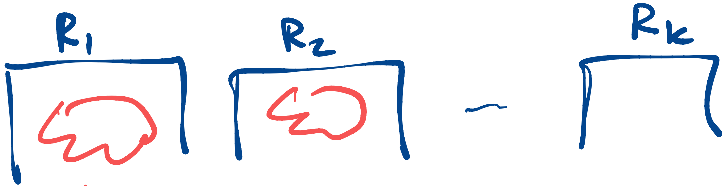
$A \to E,\ E \to C \Rightarrow A \to C$

Does $(AB)^+$ include C?

Dependency preservation, formally

BCNF may violate this

F^+



check rules within R_1 $\quad$ R_2

$$F^+ = \left[(F^+ \cap R_1) \cup (F^+ \cap R_2) \cup \cdots \cup (F^+ \cap R_k) \right]^+$$

Dependency preservation, formally

(stud, fac, dept)

stud, dept → fac

(stud, fac)   (fac, dept)

R_i — Attributes $X_1, X_2 \ldots X_k$

$F^+ \cap R_i$?

Take each $Y \subseteq \{X_1, \ldots, X_k\}$

Compute Y^+ wrt F^+

Retain all rules that stay in R_i

Dependency preservation, formally

R_7

X_1	X_2	X_3	X_4	X_5

$(\{X_1, X_3\})^+$

Every $\beta \to \gamma$ st $\beta \in (\ )^+$
add γ to $(\ )^+$

$X_3 \to Z_8$ $\quad Z_8 \to X_4$

$(\{X_1, X_3\})^+$ contains $X_4, X_5, Z, \ldots$

X_1
X_2

$$\left.\begin{array}{l} X_1 X_3 \to X_4 \\ X_1 X_3 \to X_5 \end{array}\right\} = F^+ \cap R_7$$

Dependency preservation, formally

$R_1 \; \ldots \; R_k$

$F_1' \qquad F_k'$

$$F_i' = F^+ \cap R_i$$

$$(F_1' \cup F_2' \cup \ldots F_k')^+ = F^+$$

If so, F^+ is locally checkable in this decomp.

Decomp preserves dependencies

Beyond functional dependencies

Student — Emergency contacts

- ≥ 2 phone numbers
- ≥ 2 email address

	Stud ID	Emerg Phone	Emer Email
	ID_1	P_1	E_1
	ID	P_2	E_2
$\Rightarrow$	ID_1	P_1	E_2
	ID_1	P_2	E_1