BCNF Normalization - examples

Functional dependencies

Example 1

- Database schema
  - *inst_dept(ID, name, salary, dept_name, building, budget)*

- Functional dependencies (domain knowledge)
  - *ID → name,*
  - *ID → salary,*
  - *ID → dept_name*
  - *dept_name → building*
  - *dept_name → budget*

- Derived dependencies
  - *ID → building*
  - *ID → budget*

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

Functional dependencies

Example 1

- Database schema
  - *inst_dept(ID, name, salary, dept_name, building, budget)*

- All functional dependencies (domain knowledge)
  - *ID* → *name,*
  - *ID* → *salary,*
  - *ID* → *dept_name*
  - *dept_name* → *building*
  - *dept_name* → *budget*
  - *ID* → *building*
  - *ID* → *budget*

- We see that
  - *ID* is a superkey
  - *dept_name* is NOT a superkey

- Can be “rewritten” (is covered by)
  - *ID* → *name, salary, dept_name, building, budget*
  - *dept_name* → *building, budget*

Functional dependencies

Example 1

- Database schema
  - *inst_dept(ID, name, salary, dept_name, building, budget)*

- functional dependencies to take into consideration
  - *ID → name, salary, dept_name, building, budget*
  - *dept_name → building, budget*

- We see that
  - *ID* is a superkey
  - *dept_name* is NOT a superkey
- thus
  - *dept_name → building, budget*
  - *violates BCNF*

BCNF Decomposition Algorithm

Example 1

Database schema

inst_dept(ID, name, salary, dept_name, building, budget)

- *dept_name* → *building, budget*
- *violates BCNF in inst_dept*

→ *result* := {R_1 } = {*inst_dept* };
done := false;
compute F^+;
while (not *done)* **do**
  if (there is a schema R_i in *result* that is not in BCNF)
    then begin
      let $\alpha \rightarrow \beta$ be a nontrivial functional dependency that
        holds on R_i such that $\alpha \rightarrow R_i$ is not in F^+,
        and $\alpha \cap \beta = \emptyset$;
      result := (*result* – R_i) ∪ (R_i – β) ∪ (α, β);
      end
    else *done* := **true;**

BCNF Decomposition Algorithm

Example 1

Database schema

inst_dept(ID, name, salary, dept_name, building, budget)

- *dept_name* $\rightarrow$ *building, budget*
- *violates BCNF in inst_dept*

let $\alpha \rightarrow \beta$ be a nontrivial functional dependency that
holds on R_i such that $\alpha \rightarrow R_i$ is not in F^+,
and $\alpha \cap \beta = \varnothing$;
result := (*result* – R_i) $\cup$ (R_i – β) $\cup$ (α, β);

for "$\alpha \rightarrow \beta$" = "*dept_name* $\rightarrow$ *building, budget*"

Means **we need to decompose into**

R1(dept_name, building, budget)

inst_dept1(ID, name, salary, dept_name)

BCNF Decomposition Algorithm

Example 1

Database schema

inst_dept(ID, name, salary, dept_name, building, budget)

- *dept_name* $\rightarrow$ *building, budget*
- *violates BCNF in inst_dept*

let $\alpha \rightarrow \beta$ be a nontrivial functional dependency that holds on R_i such that $\alpha \rightarrow R_i$ is not in F^+, and $\alpha \cap \beta = \varnothing$;

result := (*result* – R_i) $\cup$ (R_i – β) $\cup$ (α, β);

for "$\alpha \rightarrow \beta$" = "*dept_name* $\rightarrow$ *building, budget*"

A normalised schema with **better relation schema names**:

dept(dept_name, building, budget)

inst(ID, name, salary, dept_name)

Resulting database after BCNF-normalisation

Example 1

inst_dept

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

inst

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

dept

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

Functional dependencies

Example 2

- Database schema
  - *class (course_id, title, dept_name, credits, sec_id, semester, year, building, room_number, capacity, time slot id)*

- functional dependencies to take into consideration
  - *course_id → title, dept_name, credits*
  - *building, room_number → capacity*
  - *course_id, sec_id, semester, year → building, room_number, time slot_id*

- We see that
  - *course_id, sec_id, semester, year* is a superkey
  - *course_id* is NOT a superkey
- thus
  - *course_id → title, dept_name, credits*
  - *violates BCNF*

BCNF Decomposition

Example 2

Database schema

class (course_id, title, dept_name, credits, sec_id, semester, year, building, room_number, capacity, time slot id)

- *course_id → title, dept_name, credits*
- *violates BCNF in class*

→ *result* := {R_1 } = {*class* };
done := false;
compute F^+;
while (not *done)* **do**
  if (there is a schema R_i in *result* that is not in BCNF)
    then begin
      let $\alpha \rightarrow \beta$ be a nontrivial functional dependency that
        holds on R_i such that $\alpha \rightarrow R_i$ is not in F^+,
        and $\alpha \cap \beta = \varnothing$;
      result := (*result* – R_i) $\cup$ (R_i – β) $\cup$ (α, β);
    end
    else *done* := **true;**

Algorithm from DSC fig 8.11

BCNF Decomposition

Example 2

Database schema

class (course_id, title, dept_name, credits, sec_id, semester, year, building, room_number, capacity, time slot id)

- *course_id → title, dept_name, credits*
- *violates BCNF in class*

let $\alpha \rightarrow \beta$ be a nontrivial functional dependency that holds on R_i such that $\alpha \rightarrow R_i$ is not in F^+, and $\alpha \cap \beta = \varnothing$;

result := (*result* – R_i) $\cup$ (R_i – β) $\cup$ (α, β);

for "$\alpha \rightarrow \beta$" = "*course_id → title, dept_name, credits*"

Means we need to decompose *class* into

course (course_id, title, dept_name, credits)

class1 (course_id, sec_id, semester, year, building, room_number, capacity, time_slot_id)

BCNF Decomposition

Example 2

Database schema

course (course_id, title, dept_name, credits)

class1 (course_id, sec_id, semester, year, building, room_number, capacity, time_slot_id)

result := {R_1, R_2} = {*course, class1* };

done := false;

compute F^+;

while (not *done)* **do**

if (there is a schema R_i in *result* that is not in BCNF)

then begin

let $\alpha \rightarrow \beta$ be a nontrivial functional dependency that holds on R_i such that $\alpha \rightarrow R_i$ is not in F^+, and $\alpha \cap \beta = \varnothing$;

result := (*result* – R_i) $\cup$ (R_i – β) $\cup$ (α, β);

end

else *done* := **true;**

- *building, room_number → capacity*
- *violates BCNF in class1*

Algorithm from DSC fig 8.11

BCNF Decomposition

Example 2

Database schema

course (course_id, title, dept_name, credits)

class1 (course_id, sec_id, semester, year, building, room_number, capacity, time_slot_id)

- *building, room_number → capacity*
- *violates BCNF in class1*

let $\alpha \rightarrow \beta$ be a nontrivial functional dependency that holds on R_i such that $\alpha \rightarrow R_i$ is not in F^+, and $\alpha \cap \beta = \varnothing$;

result := (*result* – R_i) $\cup$ (R_i – β) $\cup$ (α, β);

for "$\alpha \rightarrow \beta$" = "*building, room_number → capacity*"

Means we need to decompose *class1* into

classroom (building, room_number, capacity)

section (course_id, sec_id, semester, year, building, room_number, time_slot_id)

BCNF Decomposition

Example 2

Database schema

course (course_id, title, dept_name, credits)
classroom (building, room_number, capacity)
section (course_id, sec_id, semester, year, building, room_number, time_slot_id)

result := {R_1 , R_2 , R_3 } = {*course, classroom, section*};
done := false;
compute F^+;
while (not *done)* **do**
  if (there is a schema R_i in *result* that is not in BCNF)
    then begin
      let $\alpha \rightarrow \beta$ be a nontrivial functional dependency that
        holds on R_i such that $\alpha \rightarrow R_i$ is not in F^+,
        and $\alpha \cap \beta = \varnothing$;
      result := (*result* – R_i) $\cup$ (R_i – β) $\cup$ (α, β);
    end
  else *done* := **true;**

- *No dependency violates BCNF*

Algorithm from DSC fig 8.11

- ***So we are done***