**Database Management Systems (COP 5725)**

## (Fall 2019)

Instructor: Dr. Markus Schneider

TA: Kyuseo Park

Homework 5

|  |  |
| --- | --- |
| Name: | Yue Wang |
| UFID: | 11797072 |
| Email Address: | yue.wang1@ufl.edu |

Pledge (Must be signed according to UF Honor Code)

On my honor, I have neither given nor received unauthorized aid in doing this assignment.

Signature

For scoring use only:

|  |  |  |
| --- | --- | --- |
|  | Maximum | Received |
| Exercise 1 | 20 |  |
| Exercise 2 | 25 |  |
| Exercise 3 | 25 |  |
| Exercise 4 | 15 |  |
| Exercise 5 | 15 |  |
| Total | 100 |  |

Consider the following table which is used to store students and courses records.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| UFID | Course\_ID | Grade | Student\_Name | Department | Tuition Fee | Instructor |
| 4114123 | COP01, COP02, COP03 | A, A, B | John Smith | CISE | 250 | James, Andrew, Peter |
| 3124234 | BU01, BU02 | B, B | Roger Hicks | Business | 300 | Alan, Alan |

Please note that *Tuition Fee* depends on the department.

1. Normalize the table to the 1st Normal Form and explain your answer. [5 points]

A relation schema is in first normal form (1NF) if, and only if, the domains of all its attributes only contain atomic (or indivisible) values . Thus, we need to change the intersection of each row and each column contains one and only one atomic value. The result is as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| UFID | Course\_ID | Grade | Student\_Name | Department | Tuition Fee | Instructor |
| 4114123 | COP01 | A | John Smith | CISE | 250 | James |
| 4114123 | COP02 | A | John Smith | CISE | 250 | Andrew |
| 4114123 | COP03 | B | John Smith | CISE | 250 | Peter |
| 3124234 | BU01 | B | Roger Hicks | Business | 300 | Alan |
| 3124234 | BU02 | B | Roger Hicks | Business | 300 | Alan |

1. Explain the criteria for 2nd Normal Form and normalize the table you obtained from the previous part to meet them. Then explain which anomalies can occur with your answer. [5 points]

A relation schema R is in the second normal form (2NF) with respect to a set F of FDs if, and only if, it is in 1NF and every nonprime attribute A in R is fully functionally dependent on every candidate key of R. In other words, it is in 1NF and for every candidate key *K* of *R* and for every nonprime attribute *A* in *R* the FD *K* → *A* is left-reduced.

In the table of previous part, the primary keys are UFID and Course\_ID. But for the FDs:

UFID → Student\_Name, UFID → Department, UFID → Tuition Fee and Course\_ID → Instructor hold. So it’s a violation of the 2NF.

|  |  |  |  |
| --- | --- | --- | --- |
| students | | | |
| UFID | Student\_Name | Department | Tuition Fee |
| 4114123 | John Smith | CISE | 250 |
| 3124234 | Roger Hicks | Business | 300 |

|  |  |
| --- | --- |
| courses | |
| Course\_ID | Instructor |
| COP01 | James |
| COP02 | Andrew |
| COP03 | Peter |
| BU01 | Alan |
| BU02 | Alan |

|  |  |  |
| --- | --- | --- |
| take\_courses | | |
| UFID | Course\_ID | Grade |
| 4114123 | COP01 | A |
| 4114123 | COP02 | A |
| 4114123 | COP03 | B |
| 3124234 | BU01 | B |
| 3124234 | BU02 | B |

The 2NF still allows transitive dependencies.

The following anomalies can occur:

Insertion anomaly: We cannot insert a new department without no student in it.

Update anomaly: If there are one more studnets in a department, a change of the department requires a change for each student in it.

Delete anomaly: When we delete the student John Smith, the information of the department of CISE will be delete as well.

1. Explain the criteria for 3rd Normal Form and normalize the table you obtained for the previous question to meet them. [5 points]

A relation schema R is in the third normal form (3NF) with respect to a set F of FDs if, and only if, it is in 2NF and no nonprime attribute A in R is transitively dependent on any candidate key of R. In other word, for each FD *X* → *Y* in *F+* with *X* ⊆ *R* and *Y* ⊆ *R* at least one of the following conditions holds:

▪ X → Y is a trivial FD (i.e., Y ⊆ X holds), or  
▪ X is a superkey of R, or  
▪ Every element of Y – X is a prime attribute (i.e., contained in some candidate key) of R

Relation schema *student(UFID, Student\_Name, Department, Tuition Fee)* with the additional FD *{Department} → {Tuition Fee}*; both *Department* and *Tuition Fee* are nonprime attributes. Thus, we splitting of the schema *student* into the two 3NF schemas *student(**UFID, Student\_Name, Department)* and *department\_fee(Department, Tuition Fee)* as follows.

|  |  |  |
| --- | --- | --- |
| student | | |
| UFID | Student\_Name | Department |
| 4114123 | John Smith | CISE |
| 3124234 | Roger Hicks | Business |

|  |  |
| --- | --- |
| department\_fee | |
| Department | Tuition Fee |
| CISE | 250 |
| Business | 300 |

|  |  |
| --- | --- |
| courses | |
| Course\_ID | Instructor |
| COP01 | James |
| COP02 | Andrew |
| COP03 | Peter |
| BU01 | Alan |
| BU02 | Alan |

|  |  |  |
| --- | --- | --- |
| take\_courses | | |
| UFID | Course\_ID | Grade |
| 4114123 | COP01 | A |
| 4114123 | COP02 | A |
| 4114123 | COP03 | B |
| 3124234 | BU01 | B |
| 3124234 | BU02 | B |

1. Explain if the tables you obtained for the previous question is in BCNF and, if not, normalize it to BCNF. [5 points]

Yes. The tables I obtained is in BCNF. Because for each left-reduced FD X → Y in F+ with X ⊆ R and Y ⊆ R, X → Y is a trivial FD (i.e., Y ⊆ X holds), or X is a candidate key of R, which satisfies the two conditions for the BCNF.

Consider the relation schema *R* = (A, B, C, D, E) for the following questions.

1. Assume we have the following functional dependencies:
   * AB → C
   * C → D
   * B → E

Briefly explain if the relation R is in 2NF. If not, what modifications can be made to normalize it into 2NF? [5 points]

Answer: No. Check the attributes that are not in the right-hand sides of F: AB. AB+ =ABCDE, so candidate key is AB. In particular the FDs AB → C hold. But additionally the FDs: B → E hold, we know that attribute E is partially functionally dependent on AB. Therefore, R is not in 2NF.

To become 2NF, we can split R into two schema:

R1(A, B, C ,D) with FDs: AB → C, C → D

R2(B, E) with FD: B → E.

Both schemas satisfy the 2NF.

1. Is R in 2NF with the following functional dependencies? If not, normalize it. [5 points]
   * A → BC
   * AD → E
   * B → C

Answer: No. Check the attributes that are not in the right-hand sides of F: AD. AD+ =ABCDE, so candidate key is AD. In particular the FDs AD → E hold. But additionally the FDs: A → BC hold, so BC is partially functionally dependent on any candidate key of R, which violates the 2NF.

To become 2NF, we can split R into two schema:

R1(A, B, C) with FDs: A → BC, B → C

R2(A, D, E) with FD: AD → E

Both schemas satisfy the 2NF.

1. Are the relations from the answer of question 2 in 3NF? If not, normalize it. [5 points]

Answer: No. Relation schema R1(A, B, C) with the FDs: A → BC and B →C hold, but B and C are nonprime attributes. Thus it violates 3NF.

To become 3NF, we can split R1(A, B, C) into two schema:

R11(A, B) with FD: A → B

R12(B, C) with FD: B → C

R2(A, D, E) with FD: AD → E

1. Briefly explain if the relation R is in 2NF. [2 points].
   * A → BCDE
   * BC → ADE
   * D → E

Further, is R in 3NF? If not, what modifications can be made to normalize it into 3NF? [3 points]

Answer:

2NF? Yes. By using the Armstrong’s Axioms, we can get **A**+=ABCDE, B+=B, C+=C, D+=DE, E+=E, **BC**+=ABCDE, BD+=BDE, BE+=BE, CD+=CDE, CE+=CE, DE+=DE. So, the candidate keys are A and BC.

A is a key of single attribute, so it satisfies the 2NF.For BC, since we can compute that B+ = B and C+ = C, we cannot find a partial functional dependency. Therefore, R is in 2NF.

3NF? No. Because the FD: D → E hold, we can compute that D+ = DE, and D is not a candidate key. Thus, it violates 3NF.

To because 3NF, we can split R(A, B, C, D, E) into two schema:

R1(A, B, C, D) with FDs: A → BCD, BC → AD

R2(D, E) with FD: D → E

1. Assume we have the following functional dependencies:
   * AB → D
   * C → E
   * E → C
   * C → A
   * A → C

We decompose R into schemas R1(ABC) and R2(ABDE). Show whether it is dependency preserving by using one of the algorithms that covered in the lecture. [5 points]

Using algorithm 2:

For AB → D: Result = AB

Round 1: OldResult = AB

For R1(ABC), C = CalculateAttributeClosure(F, AB) ∩ R1 = ABC

Result = ABC

For R2(ABDE), C = CalculateAttributeClosure(F, AB) ∩ R2 = ABDE

Result = ABCDE ≠ OldResult

Round 2: OldResult = ABCDE

For R1(ABC), C = CalculateAttributeClosure(F, ABC) ∩ R1 = ABC

Result = ABCDE

For R2(ABDE), C = ClaculateAttributeClosure(F, ABDE) ∩ R2 = ABDE

Result = ABCDE = OldResult

D ∩ Result = D

For C → E: Result = C

Round 1: OldResult = C

For R1(ABC), C = CalculateAttributeClosure(F, C) ∩ R1 = AC

Result = AC

For R2(ABDE), C = CalculateAttributeClosure(F, A) ∩ R2 = AE

Result = ACE ≠ OldResult

Round 2: OldResult = ACE

For R1(ABC), C = CalculateAttributeClosure(F, AC) ∩ R1 = AC

Result = ACE

For R2(ABDE), C = CalculateAttributeClosure(F, AE) ∩ R2 = AE

Result = ACE = OldResult

E ∩ Result = E

For E → C: Result = E

Round 1: OldResult = E

For R1(ABC), C = CalculateAttributeClosure(F, ) ∩ R1 =

Result = E

For R2(ABDE), C = CalculateAttributeClosure(F, E) ∩ R2 = AE

Result = AE ≠ OldResult

Round 2: OldResult = AE

For R1(ABC), C = CalculateAttributeClosure(F, A) ∩ R1 = AC

Result = ACE

For R2(ABDE), C = ClaculateAttributeClosure(F, AE) ∩ R2 = AE

Result = ACE ≠ OldResult

Round 3: OldResult = ACE

For R1(ABC), C = CalculateAttributeClosure(F, AC) ∩ R1 = AC

Result = ACE

For R2(ABDE), C = ClaculateAttributeClosure(F, AE) ∩ R2 = AE

Result = ACE = OldResult

C ∩ Result = C

For C → A: Result = C

Round 1: OldResult = C

For R1(ABC), C = CalculateAttributeClosure(F, C) ∩ R1 = AC

Result = AC

For R2(ABDE), C = CalculateAttributeClosure(F, A) ∩ R2 = AE

Result = ACE ≠ OldResult

Round 2: OldResult = ACE

For R1(ABC), C = CalculateAttributeClosure(F, AC) ∩ R1 = AC

Result = ACE

For R2(ABDE), C = ClaculateAttributeClosure(F, AE) ∩ R2 = AE

Result = ACE = OldResult

A ∩ Result = A

For A → C: OldResult = A

Round 1: OldResult = A

For R1(ABC), C = CalculateAttributeClosure(F, A) ∩ R1 = AC

Result = AC

For R2(ABDE), C = CalculateAttributeClosure(F, A) ∩ R2 = AE

Result = ACE ≠ OldResult

Round 2: OldResult = ACE

For R1(ABC), C = CalculateAttributeClosure(F, AC) ∩ R1 = AC

Result = ACE

For R2(ABDE), C = ClaculateAttributeClosure(F, AE) ∩ R2 = AE

Result = ACE = OldResult

C ∩ Result = C

Therefore, it is dependency preserving.

1. For the relation schema R = (ABCDEF) and functional dependencies F =

{AB → C, AC → B, AD → E, B → D, BC → A, E → F}, determine whether the following decomposition is lossless. Also, determine if it is dependency preserving.

P = {R1(AB), R2(BC), R3(ABDE), R4(EF)} [10 points]

Apply the chase test:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F |
| a | b | c1 | d1 | e1 | f1 |
| a2 | b | c | d2 | e2 | f2 |
| a | b | c3 | d | e | f3 |
| a4 | b4 | c4 | d4 | e | f |

Apply AB → C:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F |
| a | b | c1 | d1 | e1 | f1 |
| a2 | b | c | d2 | e2 | f2 |
| a | b | c1 | d | e | f3 |
| a4 | b4 | c4 | d4 | e | f |

Apply AC → B:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F |
| a | b | c1 | d1 | e1 | f1 |
| a2 | b | c | d2 | e2 | f2 |
| a | b | c1 | d | e | f3 |
| a4 | b4 | c4 | d4 | e | f |

Apply AD → E:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F |
| a | b | c1 | d1 | e1 | f1 |
| a2 | b | c | d2 | e2 | f2 |
| a | b | c1 | d | e | f3 |
| a4 | b4 | c4 | d4 | e | f |

Apply B → D:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F |
| a | b | c1 | d | e1 | f1 |
| a2 | b | c | d | e2 | f2 |
| a | b | c1 | d | e | f3 |
| a4 | b4 | c4 | d4 | e | f |

Apply BC → A:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F |
| a | b | c1 | d | e1 | f1 |
| a2 | b | c | d | e2 | f2 |
| a | b | c1 | d | e | f3 |
| a4 | b4 | c4 | d4 | e | f |

Apply E → F:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F |
| a | b | c1 | d | e1 | f1 |
| a2 | b | c | d | e2 | f2 |
| a | b | c1 | d | e | f |
| a4 | b4 | c4 | d4 | e | f |

Apply AD → E:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F |
| a | b | c1 | d | e | f1 |
| a2 | b | c | d | e2 | f2 |
| a | b | c1 | d | e | f |
| a4 | b4 | c4 | d4 | e | f |

There is no row that is fully unsubscripted. Thus, the decomposition is lossy.

We can check the FD AB → C, and it is easy to find that this FD cannot be fit into any of the new relations. Thus, it is not dependency preserving.

1. Consider the relation schema R = (ABCDE).
   1. For the functional dependencies F = {AB → C, C → E, B → D, E → A}, is P = {R1 (BCD), R2 (ACE)} a lossless decomposition? Show all the steps. [5 points]

Using the chase test:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | E |
| a1 | b | c | d | e1 |
| a | b2 | c | d2 | e |

Apply AB → C:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | E |
| a1 | b | c | d | e1 |
| a | b2 | c | d2 | e |

Apply C → E:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | E |
| a1 | b | c | d | e |
| a | b2 | c | d2 | e |

Apply B → D:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | E |
| a1 | b | c | d | e |
| a | b2 | c | d2 | e |

Apply E → A:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | E |
| a | b | c | d | e |
| a | b2 | c | d2 | e |

The first row is fully unsubscripted, so it is a lossless decomposition.

* 1. For the functional dependencies F = {A → CD, B → CE, E → B}, give a lossless-join decomposition of R into BCNF. [5 points]

1. Decomposition by A → CD. R1 = (A, B, E), R2 = (A, C, D).

2. Decomposition of R1 by E → B. R11 = (A, E), R12 = (B, E).

Thus, (A, E), (B, E) and (A, C, D) form a decomposition into BCNF.

* 1. For the functional dependencies F = {A → CD, B → CE, E → B}, give a lossless-join decomposition of R into 3NF preserving functional dependencies. [5 points]

Step 1: Computation of a minimal cover: The given FDs are already the minimal cover

Step 2: Generation of relation schemas from the FDs

From A → CD, we obtain: R1 = (A, C, D), F1 = {A → CD}

From B → CE, we obtain: R2 = (B, C, E), F2 = {B → CE, E → B}

From E → B, we obtain: R3 = (B, E), F3 = {E → B}, but E, B are in R2.

Step 3: Check if a relation schema contains a candidate key

The candidate keys are AB and AE. There is no relation has a candidate key, so we add R4 = (A, B).

Step 4: We can have that R3 R2

Step 5: So we have the decomposition R1 = ACD, R2 = BCE, R3 = AB.

# Exercise 4 - Normalization [15 points]

Suppose we have a relation schema R(A, B, C, D, E, F, G) and a set of functional dependencies F = {BCD  A, BC  E, A  F, F  G, C  D, A  G, A  B}. Decompose R into 3NF by using the 3NF synthesis algorithm. Show all steps and argue precisely. Is this decomposition also in BCNF? If so, why? If not, why not? [15 points]

Step 1: Computation of a minimal cover: {BC →AE, A → BF, F → G, C → D}

Step 2: Generation of relation schemas from the FDs

From BC → AE, we obtain: R1 = ABCE, F1 = {BC → AE, A → B}

From A → BF, we obtain: R2 = ABF, F2 = {A → BF}

From F → G, we obtain R3 = FG, F3 = {F → G}

From C → D, we obtain R4 = CD, F4 = {C → D}

Step 3: Check if a relation schema contains a candidate key

The candidate keys are AC and BC, and the R1 contains AC and BC. Therefore, we don’t need to add extra schemas.

Step 4: So we have the decomposition R1 = ABCE, R2 = ABF, R3 = FG, R4 = CD.

The decomposition is not in BCNF, since for R1 = ABCE, the FD A → B holds, but A is not a superkey.

Consider the following tables:

CREATE TABLE PRODUCT (MAKER VARCHAR2(50), MODEL VARCHAR2(50), TYPE VARCHAR2(30));

CREATE TABLE DESKTOP

(MODEL VARCHAR2(50) NOT NULL, SPEED NUMBER(8),

RAM VARCHAR2(30), HD VARCHAR2(30), PRICE NUMBER(8));

CREATE TABLE LAPTOP

(MODEL VARCHAR2(50) NOT NULL, SPEED NUMBER(8),

RAM VARCHAR2(30), HD VARCHAR2(30),

SCREEN VARCHAR2(30), PRICE NUMBER(8));

CREATE PRINTER

(MODEL VARCHAR2(50) NOT NULL, COLOR VARCHAR2(30),

TYPE VARCHAR2(30), PRICE NUMBER(8));

1. Write a check condition to ensure that no manufacturer of desktops also makes laptops. [3 points]

create assertion no\_manufacturer

check (

not exists (

select x.maker

from (select MAKER

from product, desktop

where product.model = desktop.model) x,

(select MAKER

From product, laptop

Where product.model = laptop.model) y

where x.maker = y.maker));

1. Write a check condition to ensure that a manufacturer of a desktop also makes a laptop with at least the same processor speed. [4 points]

Create assertion speed\_constraint

check (

Not Exists (

Select x.maker

From (Select product.maker, desktop.speed

From product, desktop

Where product.model = desktop.model) x,

(select product.maker, laptop.speed

From product, laptop

Where product.model = laptop.model) y

Where x.maker = y.maker and x.speed > y.speed));

1. Create a trigger that checks that there is no lower priced desktop with the same speed when the price of a desktop is updated. [4 points]

create trigger no\_lower\_price

before update on desktop

for each row when (:new.price<(select d.price from desktop d

where d.speed=:new.speed))

begin

:new.price := :old.price;

end;

1. Create a trigger that checks if the model number exists in the *Product* table when a new printer is inserted. [4 points]

create trigger insert\_printer

before insert on printer

for each row when (exists (select \* from Product p

where :new.model=p.model))

begin

insert into printer values( :new.model, :new.color, :new.type, :new.price)

end;