Database Management Systems

(COP 5725)

Fall 2021

Instructor: Dr. Markus Schneider

TA: Kyuseo Park

Homework 5

Name:	Qinxuan Shi
UFID:	8351-8162
Email Address:	qinxuan.shi@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.

Qinxuan Shi

Signature

For scoring use only:

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

1 Exercise 1

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

Since 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.

However, attribute EventNumber and Winning have non-atomic values, which are lists. So the table have to be divided into two tables, A(CustomerID, EventNumber, Winning), B(CustomerID, customerGrade, DiscountRate). According to the original table, CustomerID and EventNumberthe determine attribute Winning functionally; and only use CustomerID can also determines attributes customerGrade and DiscountRate. As a result, A(CustomerID, EventNumber, Winning), B(CustomerID, customerGrade, DiscountRate) these two tables are both in 1NF.

(2). Explain the criteria for 2nd Normal Form and determine if the table you obtained from the previous part is in 2nd NF. Then explain which anomalies can occur with your answer. Explain the anomalies specifically. If it is not in 2nd NF, describe the reason specifically and normalize the table to 2nd NF. [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.

Since the 2NF holds automatically for relation schemas with only single-attribute keys, so B(CustomerID, customerGrade, DiscountRate) must in 2NF. We only need to consider A(CustomerID, EventNumber, Winning). It's easy to find out that the FD (CustomerID, EventNumber→ Winning) holds, and (EventNumber→CustomerID) holds. There is no other FDs and both these FDs are not violeting 2NF. As a result, both A and B are in 2NF.

There should be anomalies occur in B. Since there are nonprime attributes functionally dependent on other nonprime attributes, (customerGrade→DiscountRate) and (DiscountRate→customerGrade).

(3). Explain the criteria for 3rd Normal Form. Determine if the table you obtained from the previous part is in 3rd NF. If it is not in 3rd NF, describe the reason specifically and normalize the table to 3rd NF. [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, 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

X is a superkey of R

Every element of Y – X is a prime attribute of R

To A(CustomerID, EventNumber, Winning), (CustomerID, EventNumber) is the primary key, and we could find the FDs F = {(CustomerID, EventNumber→Winning), (EventNumber→CustomerID)}. It's easy to find that F^+ will only add trivial FDs to F, they all hold for 3NF, so we only need to consider these two FDs.

To the first FD (CustomerID, EventNumber)→Winning, since CustomerID, EventNumber is the primary key, so it holds.

To the second FD EventNumber→CustomerID, it is not trivial, and also not superkey. CustomerID - EventNumber = CustomerID ∈prime attributes.

To B(CustomerID, customerGrade, DiscountRate), CustomerID is primary key, and the FDs F = {(CustomerId→customerGrade, DiscountRate), customerGrade→DiscountRate, DiscountRate→customerGrade}. There is no other FDs except trivial FDs.

Follow the same steps as A, customerGrade→DiscountRate, DiscountRate→customerGrade violates 3NF. So B is not in 3NF.

Normalize B to 3NF. (CustomerId as C, customerGrade as G, DiscountRate as D)

Step1: minimal cover of F = {C→G, G→D, D→G}

f1 = C→G,

f2 = G→D,

f3 = D→G

Step2: Generation of relation schemas from the FDs

R1 = {C, G}, F1 = f1

R2 = {G, D}, F2 = f2, f3

R3 = {D, G}, F3 = f2, f3

Step3: the candidate key C contained in R1.

Step4: R3⊑R2

Step5: {(R1, F1), (R2, F2)} =

{ ({CustomerId, customerGrade}, {CustomerId→customerGrade}), ({customerGrade, DiscountRate}, {DiscountRate→custome
customerGrade→DiscountRate}) }
As a result, A is in 3NF. B is normalized into C(CustomerId, customerGrade) and D(customerGrade, DiscountRate).

(4). Explain if the tables you obtained for the previous question are in BCNF and if not, normalize them to BCNF. [5 points]

To A(CustomerID, EventNumber, Winning), {(CustomerID, EventNumber→Winning), (EventNumber→CustomerID)} is in BCNF.

To C(CustomerId, customerGrade), {CustomerId→customerGrade} is in BCNF.

To D(customerGrade, DiscountRate), {DiscountRate→customerGrade, customerGrade→DiscountRate} is also in BCNF.

As a result, A(CustomerID, EventNumber, Winning), C(CustomerId, customerGrade) and D(customerGrade, DiscountRate), all these tables are in BCNF.

2 Exercise 2

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

1. Assume we have the following functional dependencies: AB → C, D → A, C → E. Briefly explain if the relation R is in 2NF? If not, what modifications can be made to normalize it into 2NF. [5 points]

The relation R is not in 2NF, since according to the FDs, the candidate key is BD. However, BD→A is not left-reduced since D→A, which violates 2NF.

Decompose the relation schema R into R1(B, D, C, E) and R2(D, A). Both two schemas are in 2NF.

2. Is R in 2NF with the following functional dependencies? If not, normalize it. [5 points]

AB → D, C → E, E → C, C → A, A → C

According to the FDs, the candidate keys are BC and BE. Non-prime attributes are A, D. We can know that BC→D is fully dependent since (C → A, AB → D) and there is no subset of BC, which is B or C that can determines D. It is the same as BE→D. However, BC→A violates 2NF, because C→A; and to BE→A, E→A also violates 2NF.

Decompose the relation schema R into R1($\underline{\text{A, B}}$, D) and R2($\underline{\text{C}}$, A, E). Both two schemas are in 2NF.

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

To R1($\underline{\text{A, B}}$, D), we have FDs F = {AB → D}, AB is candidate key, so R1 is in 3NF.

To R2($\underline{\text{C}}$, A, E), we have FDs F = {C → E, E → C, C → A, A → C}, C is candidate key. However, according to the transitivity rules, E → A and A → E holds, which violates 3NF.

Normalize R2($\underline{\text{C}}$, A, E).

Step1: minimal cover of F = {C → E, E → C, C → A, A → C}

f1 = C→E,

f2 = E→C,

f3 = C→A,

f4 = A→C

Step2: Generation of relation schemas from the FDs

R1 = {C, E}, F1 = f1, f2

R2 = {E, C}, F2 = f2, f1

R3 = {C, A}, F3 = f3, f4

R4 = {A, C}, F4 = f4, f3

Step3: the candidate key C contained in R1, R2, R3, R4.

Step4: R2$\sqsubseteq$R1, R4$\sqsubseteq$R3

Step5: {(R1, F1), (R2, F2)} =

{ ({C, E}, {C → E, E → C}), ({C, A}, {C → A, A → C}) }

As a result, the relation R is decomposed into R1($\underline{\text{A, B}}$, D), R2($\underline{\text{C, E}}$) and R3($\underline{\text{C, A}}$).

4. Are the relations from the answer to question 3 in BCNF? If not, identify FDs that violate BCNF and normalize them. [5 points].

To R1($\underline{\text{A, B}}$, D), F={AB → D}, AB is super key, so R1 is in BCNF.

To R2($\underline{\text{C, E}}$), F={C → E, E → C}, both C^+ and E^+ are equal to R2, so R2 is in BCNF.

To R3($\underline{\text{C, A}}$), F={C → A, A → C}, both C^+ and A^+ are equal to R3, so R3 is in BCNF.

As a result, all schemas R1($\underline{\text{A, B}}$, D), R2($\underline{\text{C, E}}$) and R3($\underline{\text{C, A}}$) are in BCNF.

5. Assume we have the following functional dependencies in F: {A → B, B → C, A → C, C → A}. We decompose R(A,B,C) into schemas R1(A, B) and R2(A, C). Show whether it is dependency preserving by using one of the algorithms that covered in the lecture. [5 points]

Step 1: Compute the closure of F

power set of R={∅, A, B, C, AB, AC, BC, ABC}

A^+: ABC

$\Longrightarrow A \to A, A \to B, A \to C, A \to AB, A \to AC, A \to BC, A \to ABC$ (7FDs)

B^+: ABC

$\Longrightarrow B \to A, B \to B, B \to C, B \to AB, B \to AC, B \to BC, B \to ABC$ (7FDs)

C^+: ABC

$\Longrightarrow C \to A, C \to B, C \to C, C \to AB, C \to AC, C \to BC, C \to ABC$ (7FDs)

AB^+: ABC

$\Longrightarrow AB \to A, AB \to B, AB \to C, AB \to AB, AB \to AC, AB \to BC, AB \to ABC$ (7FDs)

AC^+: ABC

$\Longrightarrow AC \to A, AC \to B, AC \to C, AC \to AB, AC \to AC, AC \to BC, AC \to ABC$ (7FDs)

BC^+: ABC

$\Longrightarrow BC \to A, BC \to B, BC \to C, BC \to AB, BC \to AC, BC \to BC, BC \to ABC$ (7FDs)

ABC^+: ABC

$\Longrightarrow ABC \to A, ABC \to B, ABC \to C, ABC \to AB, ABC \to AC, ABC \to BC, ABC \to ABC$ (7FDs)

Step 2: Compute the restrictions of F+ to the Ri

$F_{R1} = \{A \to A, B \to B, A \to B, B \to A, A \to AB, B \to AB, AB \to A, AB \to B, AB \to AB\}$ and $F_{R2} = \{A \to A, C \to C, A \to C, C \to A, A \to AC, C \to AC, AC \to A, AC \to C, AC \to AC\}$

Step 3: Form the union of all restrictions

$F_r = \{A \to A, B \to B, A \to B, B \to A, A \to AB, B \to AB, AB \to A, AB \to B, AB \to AB, C \to C, A \to C, C \to A, A \to AC, C \to AC, AC \to A, AC \to C, AC \to AC\}$

Step 4: Compute the closure of Fr

It's easy to know that F_r^+ is just like the F^+

Step 5: Check if the two closures are equal

Two closures are equal, so the decompose is dependency preserving.

3 Exercise 3

1. For the relation schema R = (A,B,C,D,E,F,G) and functional dependencies F = {AB → C, C → A, BC → D, ACD → B, D → EG, BE → C, CG → BD, CE → G}, determine whether the following decomposition is lossless. Also, determine if it is dependency preserving.

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

A	B	C	D	E	F	G
a	b	c1	d1	e1	f1	g1
a2	b	c	d2	e2	f2	g2
a	b	c3	d3	e	f3	g3
a4	b4	c4	d	e	f	g4

AB→C

A	B	C	D	E	F	G
a	b	c1	d1	e1	f1	g1
a2	b	c	d2	e2	f2	g2
a	b	c1	d3	e	f3	g3
a4	b4	c4	d	e	f	g4

C→A

A	B	C	D	E	F	G
a	b	c1	d1	e1	f1	g1
a2	b	c	d2	e2	f2	g2
a	b	c1	d3	e	f3	g3
a4	b4	c4	d	e	f	g4

BC→D

A	B	C	D	E	F	G
a	b	c1	d1	e1	f1	g1
a2	b	c	d2	e2	f2	g2
a	b	c1	d1	e	f3	g3
a4	b4	c4	d	e	f	g4

ACD→B

A	B	C	D	E	F	G
a	b	c1	d1	e1	f1	g1
a2	b	c	d2	e2	f2	g2
a	b	c1	d1	e	f3	g3
a4	b4	c4	d	e	f	g4

D→EG

A	B	C	D	E	F	G
a	b	c1	d1	e	f1	g3
a2	b	c	d2	e2	f2	g2
a	b	c1	d1	e	f3	g3
a4	b4	c4	d	e	f	g4

BE→C

A	B	C	D	E	F	G
a	b	c1	d1	e	f1	g3
a2	b	c	d2	e2	f2	g2
a	b	c1	d1	e	f3	g3
a4	b4	c4	d	e	f	g4

CG→BD

A	B	C	D	E	F	G
a	b	c1	d1	e	f1	g3
a2	b	c	d2	e2	f2	g2
a	b	c1	d1	e	f3	g3
a4	b4	c4	d	e	f	g4

CE→G

A	B	C	D	E	F	G
a	b	c1	d1	e	f1	g3
a2	b	c	d2	e2	f2	g2
a	b	c1	d1	e	f3	g3
a4	b4	c4	d	e	f	g4

the decomposition is lossy.

Dependency Preserving:

F = {AB → C, C → A, BC → D, ACD → B, D → EG, BE → C, CG → BD, CE → G}

P = {R1(AB), R2(BC), R3(ABE), R4(DEF)}

AB→C: result = AB

oldresult = AB, C := closure(F, AB∩AB)∩AB = AB, result = AB; C := closure(F, AB∩BC)∩BC = B, result = AB; C := closure(F, AB∩ABE)∩ABE = ABE, result = ABE; C := closure(F, ABE∩DEF)∩DEF = E, result = ABE;

oldresult = ABE, C := closure(F, ABE∩AB)∩AB = AB, result = ABE; C := closure(F, ABE∩BC)∩BC = B, result = ABE; C := closure(F, ABE∩ABE)∩ABE = ABE, result = ABE; C := closure(F, ABE∩DEF)∩DEF = E, result = ABE;

oldresult = result, C∩ABE is ∅, so it is not dependency preserving.

2. Consider the relation schema R = (A,B,C,D,E).

a. For functional dependencies F = {AB → C, BC → D, AD → E}, is P = {R1(ABC), R2(BCD), R3(ADE)} a lossless decomposition? Show all the steps.

A	B	C	D	E
a	b	c	d1	e1
a2	b	c	d	e2
a	b3	c3	d	e

AB→C

A	B	C	D	E
a	b	c	d1	e1
a2	b	c	d	e2
a	b3	c3	d	e

BC→D

A	B	C	D	E
a	b	c	d	e1
a2	b	c	d	e2
a	b3	c3	d	e

AD→E

A	B	C	D	E
a	b	c	d	e
a2	b	c	d	e2
a	b3	c3	d	e

The first row becomes to t, so P is a lossless decomposition.

b. For functional dependencies F = {A → CD, B → CE, E → B}, give a lossless-join decomposition of R into BCNF.

Since all FDs do not have a left-hand side which is super key, R is not in BCNF.

A → CD is the first FD that violates BCNF, we can get $A^+ = ACD$, so we decompose R into R1(A, C, D) and R2(A, B, E). Add R1 and R2 to D.

To compute F1, Fs=∅, R1={A, C, D}

A⊑R1, $A^+ = ACD$, Fs = {A→CD};

C⊑R1, $C^+ = C$, Fs = {A→CD};

D⊑R1, $D^+ = D$, Fs = {A→CD};

F1 = {A→CD}

we can know that R1 is in BCNF.

To compute F2, Fs=∅, R2={A, B, E}

A⊑R2, $A^+ = A$, Fs = {∅};

B⊑R2, $B^+ = BE$, Fs = {B→E};

E⊑R2, $E^+ = BE$, Fs = {B→E, E→B};

F2 = {B→E, E→B}

However, R2 is still not in BCNF. We find that B→E is the first FD that violates BCNF, $B^+ = BE$, R2 is decomposed into R21(B, E) and R22(A, B).

To compute F21, Fs=∅, R21={B, E}, F2 = {B→E, E→B}

B⊑R21, $B^+ = BE$, Fs = {B→E};

E⊑R21, $E^+ = BE$, Fs = {B→E, E→B};

F21 = {B→E, E→B}

we can also know that R21 is in BCNF.

To compute F22, Fs=∅, R22={A, B}

A⊑R22, $A^+ = A$, Fs = {∅};

B⊑R22, $B^+ = B$, Fs = {∅};

F22 = {∅}

As a result, R is decomposed into R1(A, C, D), R21(B, E) and R22(A, B), F1 = {A→CD}, F21 = {B→E, E→B} and F22 = {∅}. All these schemas are in BCNF.

c. For functional dependencies F = {A → CD, B → CE, E → B}, give a lossless-join decomposition of R into 3NF preserving functional dependencies.

Minimal Cover of F_c:= {A → CD, B → CE, E → B}

Step1: minimal cover of F = F_c

f1 = A → CD,

f2 = B → CE,

f3 = E → B,

Step2: Generation of relation schemas from the FDs

R1 = {A, C, D}, F1 = f1

R2 = {B, C, E}, F2 = f2, f3

R3 = {B, E}, F3 = f3

Step3: the candidate keys AB and AE are not contained in R1, R2, R3. So add R4(A, B) to the schemas.

Step4: R3⊑R2.

Step5: {(R1, F1), (R2, F2), (R3)} =

{ ({A, C, D}, {A → CD}), ({B, C, E}, {B → CE, E → B}), ({A, B}, ∅) }

4 Exercise 4

Suppose we have a relation schema R(A, B, C, D, E, F, G) and a set of functional dependencies F = {B → D, DG → C, BD → E, AG → B, ADG → B, ADG → C}. 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]

Minimal Cover of F_c:= {B → D, DG → C, B → E, AG → B}

Step1: minimal cover of F = F_c

f1 = B → D,

f2 = DG → C,

f3 = B → E,

f4 = AG → B

Step2: Generation of relation schemas from the FDs

R1 = {B, D}, F1 = f1

R2 = {C, D, G}, F2 = f2

R3 = {B, E}, F3 = f3

R4 = {A, G, B}, F4 = f4

Step3: the candidate key AGF is not contained in R1, R2, R3, R4. So add R5(A, G, F) to the schemas.

Step4: there's no redundant relation schemas.

Step5: {(R1, F1), (R2, F2), (R3, F3), (R4, F4), (R5)} =

{ ({B, D}, {B → D}), ({C, D, G}, {DG → C}), ({B, E}, {B → E}), ({A, G, B}, {AG → B}), ({A, G, F}, ∅) }

this decomposition is also in BCNF. Since to R1, $B^+ = BD = R1$, which means B is super key. It is the same to R2, R3, R4 and R5, the left-hand FDs are all super key.

5 Exercise 5

1. Add constraints to the table TAKES that checks STUDENT_NO and COURSE_NO refer to columns STUDENT.STNO and COURSE.CRNO, respectively. The constraint should guarantee that once a student or course are deleted from STUDENT or COURSE tables, the corresponding records in TAKES table are also deleted. [3 points]

ALTER TABLE TAKES ADD CONSTRAINT TCS FOREIGN KEY (STUDENT_NO) REFERENCES STUDENT(STNO) on delete cascade;

ALTER TABLE TAKES ADD CONSTRAINT TCC FOREIGN KEY (COURSE_NO) REFERENCES COURSE(CRNO) on delete cascade;

2. Create primary constraints for STNO and CRNO on the STUDENT and COURSE tables. Create a constraint that check STUDENT.EMAIL is unique. Create constraints that check TAKES.GRADE is not greater than 4 and COURSE.CREDIT is less than 9. [4 points]

ALTER TABLE STUDENT ADD PRIMARY KEY (STNO);

ALTER TABLE COURSE ADD PRIMARY KEY (CRNO);

ALTER TABLE STUDENT ADD UNIQUE (EMAIL);

ALTER TABLE TAKES ADD CHECK (GRADE ≤ 4);

ALTER TABLE COURSE ADD CHECK (CREDIT < 9);

3. Create a trigger that displays the average grades of students for course (identified by COURSE_NO), before inserting a record in TAKES table with the same COURSE_NO. [4 points]

```
CREATE trigger display
before insert on TAKES
for each row
begin
select avg(GRADE) from TAKES where COURSE_NO = :new.COURSE_NO
end;
```

4. Create a trigger that inserts STUDENT_NO, COURSE_NO and the current timestamp in the STUDENT_LOG table after inserting or updating a record in the table TAKES. The ID for the new record in the STUDENT_LOG table should be the current highest ID + 1. [4 points]

```
CREATE trigger newRecord
after insert or update on TAKES
for each row
begin
insert into STUDENT_LOG values((select max(ID)+1 from STUDENT_LOG), :new.STUDENT_NO, :new.COURSE_NO, systimestamp)
end;
```