Question 1 (Knowledge Questions) [20 points]

1. What is a "functional dependency"? Provide its formal definition. [4 points]

Let *R* be the relation schema of a relation R, and let A, B $\subseteq$ *R*. B is functionally dependent on A, written A $\rightarrow$ B if, and only if, to each value in A exactly one value in B belongs:

A $\rightarrow$ B $\Leftrightarrow$ $\forall$ t1, t2 $\in$ R: t1[A] = t2[A] $\Rightarrow$ t1[B] = t2[B] for all possible relations R over *R*.

2. Describe in your own words what the closure F^+ of a set F of functional dependencies is. What are the drawbacks of calculating F^+? [4 points]

F^+ is the set of all FDs that can be logically implied by the repeated application of the Armstrong axioms from the FDs in *F*.

Drawbacks of computing the closure F^+ are: (1) In general, F^+ contains a large number of FDs so that the handling of F^+ becomes difficult, (2) A large redundant set of FDs has to be checked in consistency tests of database modifications.

3. What are the advantages of integrity constraints? What are static integrity constraints? What are dynamic integrity constraints? [4 points]

The main advantages of integrity constraints are: (1) Consistency conditions are specified only once. (2) Consistency conditions are checked automatically by the DBMS. (3) Application programs do not need to care about a check of the integrity constraints.

Static integrity constraints relate to restrictions of the possible database states. Dynamic integrity constraints refer to restrictions of the possible database state transitions.

4. List the terms for the two main requirements of normalization. [4 points]

Losslessness of join decomposition, dependency preservation

5. How is the third normal form (3NF) formally defined? What are the differences, advantages, and disadvantages of 3NF and BCNF? [4 points]

A relation schema *R* with associated FDs *F* is in 3NF if, and only if, it is in 2NF and for each FD A $\rightarrow$ B $\in$ F at least one of the following conditions holds:
(1) B $\subseteq$ A, i.e., the FD A $\rightarrow$ B is trivial.
(2) A is a superkey of R.
(3) B is (part of) some candidate key of R.

3NF ensures losslessness of join decomposition and dependency preservation. The

BCNF is stricter than the 3NF in the sense that the third condition of 3NF is removed. The BCNF has the advantage that a particular kind of inconsistency in the 3NF is eliminated by the BCNF. BCNF has the disadvantage that dependency preservation is not guaranteed but only the losslessness of join decomposition.

Question 2 (Functional Dependencies and Normal Forms) [32 points]

Consider the relation schema $R(A, B, C, D, E)$ with the functional dependencies $F = \{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$. Answer the following questions.

1. What is the attribute closure of CE? Please apply the Armstrong axioms (do *not* use the attribute closure algorithm) and indicate the rules that you have used in each step. What can we say about the result of CE^+? [6 points]

(1) $E \rightarrow E$ reflexivity rule

(2) $E \rightarrow A$ given in F

(3) $E \rightarrow AE$ union rule with (1) and (2)

(4) $A \rightarrow BC$ given in F

(5) $E \rightarrow BC$ transitivity rule with (2) and (4)

(6) $E \rightarrow ABCE$ union rule with (3) and (5)

(7) $B \rightarrow D$ given in F

(8) $E \rightarrow B$ decomposition rule with (5)

(9) $E \rightarrow D$ transitivity rule with (8) and (7)

(10) $E \rightarrow ABCDE$ union rule with (6) and (9)

(11) $CE \rightarrow ABCDE$ augmentation rule

This means that the closure of CE is ABCDE, which is equal to R. CE is a superkey of R since it turns out in (10) that E is a candidate key.

2. List the candidate keys of R, and precisely describe your determination method and argumentation. [6 points]

Left	Both	Right
	ABCDE	

A^+=ABCDE

B^+=BD

C^+=C

D^+=D

E^+=ABCDE

A and E are candidate keys. Any attribute set with two attributes that contains A or E is therefore a superkey and can therefore be ignored in the following.

BC^+=ABCDE

BD^+=BD

CD^+=ABCDE

BC and CD are candidate keys. Forming attribute sets with three attributes that contain BD leads to superkeys.

The candidate keys are A, BC, CD, and E.

3. Suppose that we decompose the above schema R into $R1(A, B, C)$ and $R2(A, D, E)$. Is this decomposition a lossless join decomposition? Please give the reason for your answer. [5 points]

Yes, this is a lossless join decomposition. A sufficient condition of a lossless-join decomposition is that either R1 ∩ R2 → R1 or R1 ∩ R2 → R2 holds. In our case R1 ∩ R2 = A holds. Since A is a candidate key (from the above answer), we obtain a lossless-join decomposition.

In the following, let $R(A, B, C, D)$ be a relation schema with the FDs $F = \{B \rightarrow C, B \rightarrow D\}$.

4. Identify the candidate keys for R. [3 points]

Applying the reflexivity rule and the union rule leads to B→BCD. Applying the augmentation rule leads to AB→ABCD. Thus, AB is a candidate key. A and B alone are not candidate keys. C and D only appear on right sides of FDs and cannot be part of any candidate key. This means AB is the only candidate key.

5. Is the relation schema *R* in 3NF? If yes, justify your answer. If not, point out the violations, decompose *R* into 3NF, and show the detailed steps. [8 points]

No. $B \rightarrow C$ and $B \rightarrow D$ both violate 3NF because B is not a superkey, and C and D are not part of a candidate key.

Application of the 3NF synthesis algorithm:

Step 1: Determine a canonical cover F_c for F

We cannot left reduce and right reduce the two FDs $B \rightarrow C$ and $B \rightarrow D$ any further since both sides of each FD consist of a single attribute. But we can apply the union rule and obtain $F_c = \{B \rightarrow CD\}$.

Step 2: Assemble relation schemas from the FDs of F_c

We obtain the relation schema BCD.

Step 3: Check if a relation schema contains a candidate key. If not, add a new relation schema with a candidate key.

We have to add AB as another relation schema since BCD does not contain AB.

Step 4: Relation schemas contained in other relation schemas?

No

Therefore, the 3NF decomposition consists of R1(A, B} with $F1 = \varnothing$ and R2 (B, C, D} with $F2 = \{B \rightarrow CD\}$.

6. Is the above FDs in BCNF? If yes, justify your answer. If not, point out the violations. [4 points]

Yes. Both R1 and R2 are in BCNF. B is a superkey of R2.

Question 3 (Functional Dependencies and Normal Forms) [28 points]

1. Which of the following attribute sets is a candidate key for $R(A, B, C, D, E, F, G)$ with functional dependencies $\{AB \rightarrow C, CD \rightarrow E, EF \rightarrow G, FG \rightarrow E, DE \rightarrow C, BC \rightarrow A\}$? Please circle your answers and provide an argumentation for your choices. [3 points]

BDF *ACDF* *ABDFG* *BDFG*

$BDF^+ = BDF$, $ACDF^+ = ACDEFG$, $ABDFG^+ = ABCDEFG$, $BDFG^+ = ABCDEFG$

BDFG is a candidate key.

2. Let us decompose $R(A, B, C, D, E)$ into relations with the following three sets of attributes: $\{A, B, C\}$, $\{B, C, D\}$, and $\{A, C, E\}$. For the set $F = \{A \rightarrow D, D \rightarrow E, B \rightarrow CD\}$ of FDs, use the chase test to tell whether the decomposition of R is lossless. If it is not lossless, give an example of an instance of R that returns more tuples than R has when projected onto the decomposed relations and rejoined. [10 points]

A	B	C	D	E
a	b	c	d1	e1
a1	b	c	d	e2
a	b1	c	d2	e

A->D

A	B	C	D	E
a	b	c	d1	e1
a1	b	c	d	e2
a	b1	c	d1	e

D->E

A	B	C	D	E
a	b	c	d1	e
a1	b	c	d	e2
a	b1	c	d1	e

B->CD

A	B	C	D	E
a	b	c	d	e
a1	b	c	d	e2
a	b1	c	d1	e

Since there is an unsubscripted row, the decomposition for R is lossless for this set of FDs.

3. Compute a canonical cover F_c for a relation schema $R(A, B, C, D, E)$ with FDs $F = \{C \rightarrow B, CB \rightarrow AC, CAE \rightarrow FB, D \rightarrow E, CA \rightarrow B\}$. [10 points]

a) Left Reduction:

(1) CB→AC is replaced by C→AC. B on the left side is extraneous since we already know that C→B, i.e., B ⊆ AttrClosure(F, C).

(2) CAE→FB is replaced by CE→FB. A on the left side is extraneous since from C→AC we know that C→A, i.e., A ⊆ AttrClosure(F, C).

(3) CA→B is replaced by C→B. A on the left side is extraneous since from C→AC we know that C→A, i.e., A ⊆ AttrClosure(F, C).

Hence, we get: C→B, C→AC, CE→FB, D→E.

b) Right Reduction:

(1) C→AC is replaced by C→A. C on the right side is extraneous since we know that C→C, i.e., C ⊆ AttrClosure(F − {C →CA} ∪ {C→A}, C).

(2) CE→FB is replaced by CE→F. B on the right side is extraneous since we know that C→B, i.e., B ⊆ AttrClosure(F − {CE→FB} ∪ {CE→F}, C).

Hence, we get: C→B, C→A, CE→F, D→E.

c) FDs with empty sets on their right sides: Nothing to be done.

d) After applying the union rule we obtain F_c = {C→AB, CE→F, D→E}.

4. Are the FDs of F of the previous question equivalent to $F2 = \{C \rightarrow ABC, CB \rightarrow A, A \rightarrow B, CE \rightarrow F, D \rightarrow DE\}$? Please give reasons for your answer. [5 points]

No.

From Question 3, we know that the canonical cover F_c = {C→AB, CE→F, D→E}.

1) One can show that F ={C→B, CB→AC, CAE→FB, D→E, CA→B} cannot derive all FDs of $F2$ = {C→ABC, CB→A, A→B, CE→F, D→DE}. We check:

We can get C→ABC from the union of C→AB (got from F_c) and C→C.

We can get CB→A CB→AC in F.

We can get CE→F can get from the canonical cover F_c.

We can get D→DE from the union of D→E (got from F_c) and D→D.

But we cannot get A→B from any FD of F or F_c.

At this point we can already conclude that F and F2 are not equivalent.

2) One can show that $F2$ ={C→ABC, CB→A, A→B, CE→F, D→DE} can derive F = {C→B, CB→AC, CAE→FB, D→E, CA→B}. We do this by using F_c instead of F. As long as F2 can derive F_c, F2 can derive F.

We get C→AB from C→ABC by decomposition.

CE→F is in F2.

We get D→E from D→DE by decomposition.

Because F cannot derive all FDs of F2, these two sets are not equivalent.

Question 4 (Data Integrity) [20 Points]

Consider the following database schema of student application (primary keys are underlined):

Student(<u>sID: integer</u>, name: varchar(20), address: varchar(100), GPA: float, sizeHS: integer)
Campus(<u>cID: integer</u>, location: varchar(200), enrollment: integer, rank: integer)
Apply(<u>sID: integer</u>, <u>cID: integer</u>, date: varchar(20), major: varchar(20), decision: char)

1. Write SQL statements to create tables for these schemas and clearly specify primary key and foreign key using referential integrity. For any deletion in Table Student and Campus, table Apply will delete also. State clearly how the foreign key constraints will perform on the deletion operation of table Apply. [5 points]

```sql
CREATE TABLE Student
(sID integer,
 name varchar(20),
 address varchar(100),
 GPA float,
 sizeHS integer,
Primary key (sID));

CREATE TABLE Campus
(cID integer,
location varchar(200),
enrollment integer,
rank integer,
Primary key(cID));

CREATE TABLE Apply
(sID integer,
cID integer,
date varchar(20),
major varchar(20),
decision char,
Foreign key (sID) references Student(sID) on delete cascade,
Foreign key (cID) references Campus(cID) on delete cascade);
```

2. Write assertions for each of the following conditions: Students with GPA < 3.0 can only apply to campuses with rank > 4. [5 points]

```sql
CREATE ASSERTION RestrictApps CHECK(
    NOT EXISTS (SELECT * FROM Student, Apply, Campus
                WHERE Student.sID = Apply.sID
                      AND Apply.cID = Campus.cID
                      AND Student.GPA < 3.0 AND Campus.rank <= 4))
```

3. Suppose that "decision" in Table Apply can only have the three values 'Y', 'N', and 'U'. Add integrity constraints to Apply to check about that. [5 points]

```sql
alter table Apply

add constraint limit check(decision in ('Y', 'N', 'U'));
```

4. Write triggers for the following situation: If an application tuple is inserted for a student with GPA > 3.9 and sizeHS > 1500 to a campus whose ID is 5566, set decision to "Y". [5 points]

```
 CREATE TRIGGER AutoAccept
  AFTER INSERT ON Apply
  REFERENCING NEW ROW AS NewApp
  FOR EACH ROW
  WHEN (NewApp.cID = 5566 AND
       3.9 < (SELECT GPA FROM Student WHERE sID = NewApp.sID) AND
       1500 < (SELECT sizeHS FROM Student WHERE sID = NewApp.sID))
begin
  UPDATE Apply
    SET decision = 'Y'
    WHERE sID = NewApp.sID
    AND cID = NewApp.cID
    AND date = NewApp.date
End;
```

Or

```
CREATE TRIGGER AutoAccept
  Before INSERT ON Apply
  REFERENCING NEW ROW AS NewApp
  FOR EACH ROW
  WHEN (NewApp.cID = 5566 AND
       3.9 < (SELECT GPA FROM Student WHERE sID = NewApp.sID) AND
       1500 < (SELECT sizeHS FROM Student WHERE sID = NewApp.sID))
Begin
NewApp.decision = ‘Y’
End;
```