CSCI 4125/5125 Course Project
Data Models and Database Systems
Spring 2022
Course Project
Phase 6: Normalization

Due: Thursday, 5/12 @ 11:59pm
Reading: Silberschatz Chapter 7

Submission Guidelines:
1. This assignment is worth 100 points for all students.

2. It is your responsibility to make sure all files are readable and submitted on time.

Submission:

- Parts A & B. Submit a single file for all answers. You may draw out solutions by hand, but they must be legible and clearly organized into a single file.

- Part C. Extra Credit. Submit a separate file(s) for your submission to the extra credit portion.

Part A. Bed Relation (40 points). Since you originally designed the hospital database, fields have been added to the original Bed relation. You start to notice some update anomalies caused by data redundancy. Your manager wants you to evaluate the relation and fix any problems.

Original Bed Relation

BED (BedNumber, RoomNumber, Unit, PatientNumber, NurseID)

New Bed Relation

BED (BedNumber, RoomNumber, Unit, PatientNumber, NurseID, Floor, View, Windows, Manager, AdministrationOffice)

F = {

BedNumber, Unit → NurseID

BedNumber → RoomNumber, PatientNumber, NurseID

RoomNumber → Floor, View, Windows

Unit → Manager, AdministrationOffice

}

1. [10 points] Compute the canonical cover for F. What is the candidate key(s)?

Step1: Put all the functional dependencies in minimal form (only one attribute on the RHS).

F_c = {

BedNumber, Unit → NurseID

BedNumber → RoomNumber

BedNumber → PatientNumber

BedNumber → NurseID

RoomNumber → Floor

RoomNumber → View

RoomNumber → Windows

Unit → Manager

Unit → AdministrationOffice

}

}

Step2. Remove all extraneous LHS attributes.

BedNumber can define NurseID without the Unit, so the Unit is removed.

F_c = {

BedNumber, (✗)Unit → NurseID

BedNumber → RoomNumber

BedNumber → PatientNumber

BedNumber → NurseID

RoomNumber → Floor

RoomNumber → View

RoomNumber → Windows

Unit → Manager

Unit → AdministrationOffice

}

Step3. Remove all extraneous RHS attributes.

One of the BedNumber → NurseID is extraneous, it is removed.

F_c = {

BedNumber → NurseID ✗

BedNumber → RoomNumber

BedNumber → PatientNumber

BedNumber → NurseID

RoomNumber → Floor

RoomNumber → View

RoomNumber → Windows

Unit → Manager

Unit → AdministrationOffice

}

Step4. Combine dependencies with the same LHS (union rule)

F_c = {

BedNumber → RoomNumber, PatientNumber, NurseID

RoomNumber → Floor, View, Windows

Unit → Manager, AdministrationOffice

}

Candidate keys: ***BedNumber, Unit*** and ***BedNumber, RoomNumber, Unit***

Primary key: ***BedNumber, Unit***

2. [15 points] Remove any partial key dependencies to create a set of linked relational schemas in Second Normal Form. Primary keys require a solid underline. Foreign keys require a dotted underline and an arrow to the attribute(s) they reference.

T1 (BedNumber, Unit, Floor, View, Windows)

T3 (Unit, Manager, AdministrationOffice)

T2 (BedNumber, RoomNumber, PatientNumber, NurseID)

3. [15 points] Remove any transitive dependencies to create a set of linked relational schemas in Third Normal Form. Primary keys require a solid underline. Foreign keys require a dotted underline and an arrow to the attribute(s) they reference.

T1 (BedNumber, Unit)

T3 (Unit, Manager, AdministrationOffice)

T2 (BedNumber, RoomNumber, PatientNumber, NurseID)

T4 (RoomNumber, Floor, View, Windows)

Part B. PhysicianPatient Relation (60 points). The hospital wants to store a new field called illness. The illness can be determined by the Patient Number and the Physician ID, and the illness determines the Physician ID (this is represented in F below). One of your colleagues (who never took CSCI 4125/5125) doesn't know how to store this new information so they throw all the physician data, patient data, and the new illness field into a new relation called PhysicianPatient (show below). Your manager does not think this looks right but trusts your opinion. It is your job to evaluate this new relation and fix any problems.

Original Relations

PHYSICIAN (PhysID, PhysName, Specialty)

PATIENT (PatientNumber, PatientName, Age)

New Relation

PhysicianPatient (PhysID, PhysName, Specialty, PatientNumber, PatientName, Age, Illness)

F = {

PhysID → PhysName, Specialty

PatientNumber → PatientName, Age

PatientNumber, PhysID → Illness

Illness → PhysID

}

4. [10 points] Compute the canonical cover for F. What is the primary key?

Step1: Put all the functional dependencies in minimal form (only one attribute on the RHS).

Step2. Remove all extraneous LHS attributes.

Step3. Remove all extraneous RHS attributes

All 3 steps give us the following canonical coverage:

F_c = {

PhysID → PhysName

PhysID → Specialty

PatientNumber → PatientName

PatientNumber → Age

PatientNumber, PhysID → Illness

Illness → PhysID

}

Step4. Combine dependencies with the same LHS (union rule)

F_c = {

PhysID → PhysName, Specialty

PatientNumber → PatientName, Age

PatientNumber, PhysID → Illness

Illness → PhysID

}

Candidate keys: ***PhysID, PatientNumber*** and ***PatientNumber, Illness***

Primary key: ***PhysID, PatientNumber***

5. [15 points] Remove any partial key dependencies to create a set of linked relational schemas in Second Normal Form. Primary keys require a solid underline. Foreign keys require a dotted underline and an arrow to the attribute(s) they reference.

T1 (PhysID, PatientNumber, Illness)

T2 (PhysID, PhysName, Specialty)

T3 (PatientNumber, PatientName, Age)

6. [15 points] Remove any transitive dependencies to create a set of linked relational schemas in Third Normal Form. Primary keys require a solid underline. Foreign keys require a dotted underline and an arrow to the attribute(s) they reference.

Both sides should be non-prime for the transitive dependencies. We do not have any functional dependencies where the both left-hand side and right-hand side are non-prime. Illness → PhysID is not a transitive dependency because PhysID is a prime attribute.

7. [20 points] a. Is your schema in BCNF? If not, decompose it into BCNF.

b. State whether your decomposition lossless or not. If it is lossy (i.e., not lossless), does a lossless decomposition exist?

c. State whether your decomposition is dependency preserving or not. If not, what functional dependencies were lost?

PhysicianPatient (PhysID, PhysName, Specialty, PatientNumber, PatientName, Age, Illness)

F_c = {

PhysID → PhysName, Specialty

PatientNumber → PatientName, Age

PatientNumber, PhysID → Illness

Illness → PhysID

}

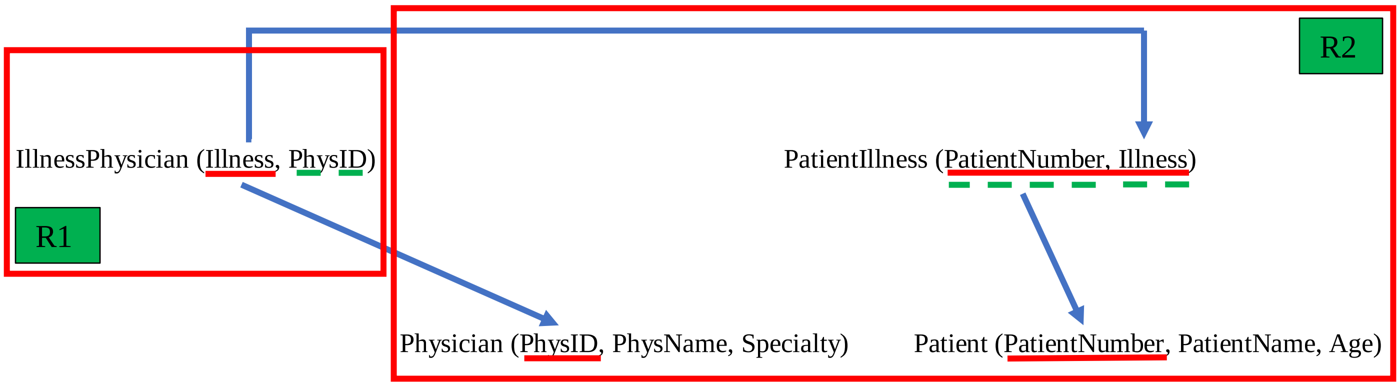
Not in BCNF because Illness is not a superkey by itself.

FDs that violate BCNF (FDs which are not a superkeys)

Illness → PhysID

Decomposing the relation into BCNF:

Decompose on Illness →PhysID



Lossless test:

1. Union. ✓
2. Intersection Illness ✓
3. Intersection is a key (Illness) in R1 and R2 ✓

Decomposition is lossless.

Dependency preserving:

Projection of F on **R1**:

1. Illness → PhysID

Projection of F on **R2**:

1. PhysID → PhysName, Specialty
2. PatientNumber → PatientName, Age

We lost **PatientNumber, PhysID → Illness** functional dependency and it cannot be derived because nothing else determines Illness.

Part C. Extra Credit (20 points). Recall that functional dependencies are constraints we enforce in our database. If any constraints are lost during decomposition, we can enforce them using something we learned earlier this semester (think back to how we can enforce multi-table constraints). Implement a solution to enforce any constraints lost in Part B.

Hint: Think back to the example in lecture where we only want to allow our pizza orders to have one topping type. In our decomposition, we lost the constraint that says OrderNumber and ToppingType determines Topping. If a record is inserted on T2(<u>OrderNumber, Topping</u>) the count for each ToppingType must be <= 1 for that order.

Note: I also recommend creating some example data to test your solution. Since this is extra credit, it will be up to you to do this.

```
DROP TABLE PatientIllness;
DROP TABLE IllnessPhysician;

CREATE TABLE PatientIllness (
        PatientNumber CHAR(3),
        Illness VARCHAR2(20),
        CONSTRAINT PatientIllness_PK PRIMARY KEY (PatientNumber, Illness),
        CONSTRAINT Illness_FK FOREIGN KEY (Illness) REFERENCES IllnessPhysician(Illness)
);

CREATE TABLE IllnessPhysician (
        Illness VARCHAR(20),
        PhysID CHAR(3),
        CONSTRAINT IllnessPhysician_PK PRIMARY KEY (Illness)
);

CREATE OR REPLACE TRIGGER PatientPhysicianOneToMany
BEFORE INSERT OR UPDATE ON PatientIllness
FOR EACH ROW
DECLARE
        Phys CHAR(3);
        newPhys CHAR(3);
BEGIN
        SELECT DISTINCT PhysID INTO Phys
        FROM IllnessPhysician IP, PatientIllness PI
        WHERE IP.Illness = PI.Illness
```

```
	AND PatientNumber = :new.PatientNumber;

	SELECT PhysID INTO newPhys
	FROM IllnessPhysician
	WHERE Illness = :new.Illness;

	DBMS_OUTPUT.PUT_LINE('Phys: ' || Phys);
	DBMS_OUTPUT.PUT_LINE('New Phys: ' || newPhys);

	IF Phys != newPhys THEN
		RAISE_APPLICATION_ERROR(-20200, 'Patient can't have more than one physician');
	END IF;
EXCEPTION
	WHEN NO_DATA_FOUND THEN
		DBMS_OUTPUT.PUT_LINE('No physician for this patient yet');
END;

INSERT INTO IllnessPhysician VALUES ('Cold', 'D01');
INSERT INTO IllnessPhysician VALUES ('Flu', 'D01');
INSERT INTO IllnessPhysician VALUES ('Allergies', 'D01');
INSERT INTO IllnessPhysician VALUES ('BrokenBone', 'D02');
INSERT INTO IllnessPhysician VALUES ('Joint Pain', 'D02');

INSERT INTO PatientIllness VALUES ('P01', 'Cold');
INSERT INTO PatientIllness VALUES ('P01', 'Flu');
INSERT INTO PatientIllness VALUES ('P01', 'Allergies');
INSERT INTO PatientIllness VALUES ('P02', 'BrokenBone');

commit;

INSERT INTO PatientIllness VALUES ('P03', 'Cold');
INSERT INTO PatientIllness VALUES ('P03', 'BrokenBone');
INSERT INTO PatientIllness VALUES ('P03', 'Flu');

commit;
```