CIS 5500: Database and Information Systems
Homework 2: Relational DB Design

February 26, 2026

Mustafa Rashid

Spring 2026

---

1. **Question 1 (30 points)**

   A. **(20 points)**

   hw2_q1.png

   B. **(5 points)**

   Ans: A TravelSegment could start or end at a Station that is not served by its associated TransportLine. This would be valid in the ER diagram because there is no constraint requiring the start and end stations of a TravelSegment to be among the stations served by that TransportLine.

   To enforce this constraint, we must require that the start and end stations of a TravelSegment appear in the Serves relationship with the same TransportLine. In other words, a TravelSegment may only start and end at stations that are served by its associated TransportLine.

   C. **(5 points)**

   Ans: Add an attribute to the relationship `Serves` called `no_of_platforms`. Add an attribute to the Station entity set called `maximum_no_of_platforms`.

   We then add a constraint that, for each Station, the sum of `no_of_platforms` across all TransportLines assigned to that Station does not exceed `maximum_no_of_platforms`.

2. **Question 2 (20 points)**

   A. **12 points**

Ans:

```
CREATE TABLE Artists(
ArtistID INT PRIMARY KEY,
Name VARCHAR(50) NOT NULL,
Nationality VARCHAR(50) NOT NULL,
BirthYear INT(4) NOT NULL
);

CREATE TABLE Customers(
CustomerID INT PRIMARY KEY,
Name VARCHAR(50) NOT NULL,
Email VARCHAR(100) NOT NULL
);

CREATE TABLE Artworks(
ArtworkID INT PRIMARY KEY,
OwnerID INT FOREIGN KEY REFERENCES Customers(CustomerID),
CreatorID INT FOREIGN KEY REFERENCES Artists(ArtistID),
Title VARCHAR(50) NOT NULL,
AYear INT(4),
Medium VARCHAR(50)
);

CREATE TABLE Exhibitions(
ExhibitionID INT PRIMARY KEY,
ExhibitionName VARCHAR(100) NOT NULL
);

CREATE TABLE DisplayedIn(
ArtworkID INT,
ExhibitionID INT,
PRIMARY KEY (ArtworkID, ExhibitionID),
FOREIGN KEY (ArtworkID) REFERENCES Artworks(ArtworkID),
FOREIGN KEY (ExhibitionID) REFERENCES
   Exhibitions(ExhibitionID),
StartDate DATE,
EndDate DATE
);
```

B. **4 points**

Ans: We can create an assertion as follows

```
        CREATE ASSERTION exhibition_cardinality
CHECK (
    NOT EXISTS (
        SELECT ExhibitionID
        FROM DisplayedIn
        GROUP BY ExhibitionID
        HAVING COUNT(*) NOT BETWEEN 5 AND 20
    )
);
```

C. **4 points**

Ans: We change the cardinality on the Artists entity set from 0..∗ to 1..∗. We also add the following assertion

```
CREATE ASSERTION artist_has_artwork
CHECK (
  NOT EXISTS (
    SELECT *
    FROM Artists A
    WHERE NOT EXISTS (
      SELECT *
      FROM Artworks W
      WHERE W.CreatorID = A.ArtistID
    )
  )
);
```

3. **Question 3 (40 points)**

A. **4 points**

Ans: No they cannot.
If they did then the functional dependencies PassengerID → PassengerEmail and PassengerEmail → PassengerID would not hold.

B. **4 points**

Ans: Yes. The functional dependency TrainNo → ConductorID only ensures each train has one conductor, but it does not prevent multiple trains from having the same conductor.

C. **6 points**

Ans: The candidate keys are {TrainNo, PassengerID} and {TrainNo, PassengerEmail}. TrainNo determines DepartureDate and ConductorID, PassengerID determines PassengerEmail, and together TrainNo and PassengerID determine SeatNo and BaggageQty, giving all attributes. Since PassengerEmail ↔ PassengerID, replacing PassengerID with PassengerEmail also forms a key.

D. **5 points**

Ans: The dependency TrainNo → DepartureDate, ConductorID violates 3NF because TrainNo is not a superkey and neither DepartureDate nor ConductorID is a member of any candidate key. All other dependencies satisfy 3NF because either the left-hand side is a superkey or the right-hand side attribute is a member of some candidate key.

E. **2 points**

F. **2 points**

Ans: Yes, the decomposition has a lossless join. $R_1 \cap R_2 = \{TrainNo, PassengerID\}$, which is a candidate key of R and therefore a superkey of R_2. By the lossless-join condition, a decomposition is lossless if the common attributes form a superkey of at least one of the relations, so the join is lossless.

G. **2 points**

Ans: No, this decomposition does **not** have a lossless join. $R_1 \cap R_2 = \{PassengerID\}$, which is neither a superkey of R_1 nor of R_2 (PassengerID alone does not determine DepartureDate or ConductorID, nor SeatNo or BaggageQty), so the lossless-join condition fails.

H. **12 points**

Ans: We first compute a minimal cover F_c by splitting all RHS, removing redundant attributes from LHS, and removing redundant FDs:

$$F_c = \{\, TrainNo \to DepartureDate,\ TrainNo \to ConductorID,\ PassengerID \to Passenger$$

$$TrainNo, PassengerID \to SeatNo,\ TrainNo, PassengerID \to BaggageQty \,\}$$

Grouping by LHS and creating one relation per group gives:

Relation	**Attributes**	**Preserved FDs**
R_1	(TrainNo, DepartureDate, ConductorID)	TrainNo → DepartureDate, Cond
R_2	(PassengerID, PassengerEmail)	PassengerID ↔ PassengerEmail
R_3	(TrainNo, PassengerID, SeatNo, BaggageQty)	TrainNo, PassengerID → SeatNo,

Since R_3 already contains the candidate key {TrainNo, PassengerID} of R, no additional relation is needed.

I. **5 points**

Ans: All three relations are in BCNF.

- R_1: The only non-trivial FD is TrainNo → DepartureDate, ConductorID, and TrainNo is the sole candidate key, so it is a superkey. BCNF holds.
- R_2: Every non-trivial FD (PassengerID → PassengerEmail and PassengerEmail → PassengerID) has a candidate key on the LHS. BCNF holds.
- R_3: The only non-trivial FD is {TrainNo, PassengerID} → SeatNo, BaggageQty, and {TrainNo, PassengerID} is the candidate key. BCNF holds.