Hanil Ahmed
23k-6005
BS AI-5A.

DATABASE SYSTEMS

Date: _____

ASSIGNMENT NO 2

QUESTION 1:

A. Primary Entities and Attributes

- Film: FilmID (PK), Title, Budget, Release Date, DirectorName
- Actor: ActorID (PK), Name, Phone
- Production Company: CompanyID (PK), Name

Associative Entities (for relationships, not primary):

Casting: FilmID (FK, part of composite PK), ActorID (FK, part of composite PK), CharacterName

Funding: FilmID (FK, part of composite PK), CompanyID (FK, part of composite PK), Amount(

B. Main Relationships and Cardinalities:

1. Film ↔ Actor → M:N
2. Film ↔ Production Company (funding) → M:N
3. Film ↔ Director → 1:1 (DirectorName stored inside FILM, no separate entity)

C. Associative Entity for the casting M:N Relationship.

Name: Casting

Attributes:

- FilmID (part of composite PK, FK referencing Film.FilmID)
- ActorID (part of composite PK; FK referencing Actor.ActorID)
- CharacterName (required, NotNULL)

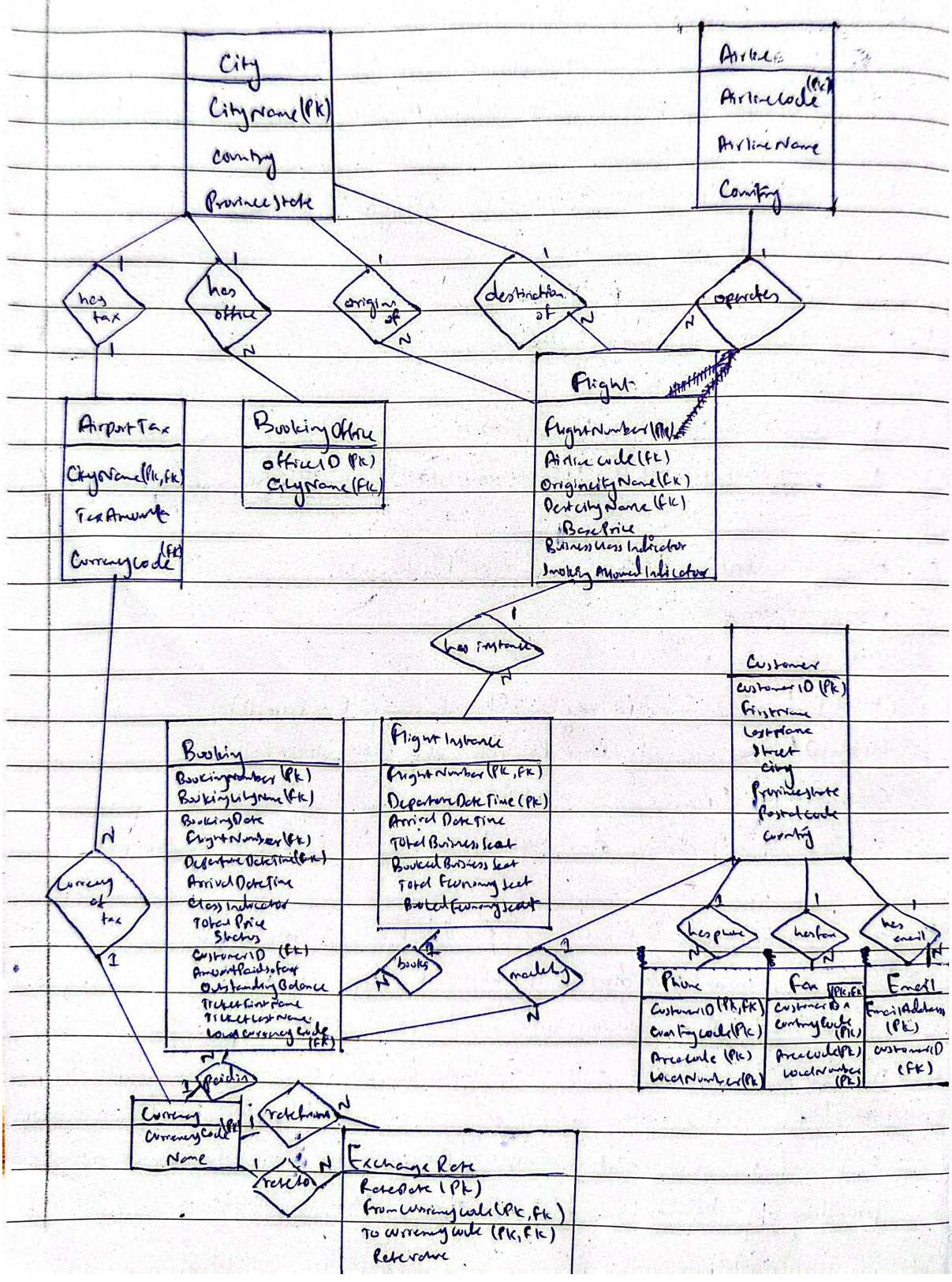
(The composite PK ensures no duplicate castings for the same film-actor pair.

D. In this simplified ~~scenario~~ Scenario, DirectorName should not be made into a seperate entity because the business rules specify that directors are not tracked seperately as reusable entities (eg no unique DirectorID, no additional attributes like phone or bio), and no need to handle relationships like a director working on multiple films). Only the name is required per film, stored as simple text data. Creating a seperate Director table would introduce unnecessary complexity, such as an extra table, potential FK constraints, and joins for queries, without providing any benefits like data reuse or normalization for multi-film directors. This

Date: _____

denormalized approach (embedding DirectorName directly in Film) is appropriate here to keep the schema efficient and aligned with the limited requirements. It needs evolve (tracking director details across films), it would be normalized later.

QUESTION 2:



Date: _____

QUESTION 3:

a. Update Anomalies Example:

- Insertion Anomaly: If we want to add a new school (eg "Blue school" with schoolID S500 in London) that has no temporary teachers assigned yet, we cannot insert the record because the table requires values for NIN, contractNo, hours, and tName, which are related to teachers and assignments, we did have to use Nulls or dummy data, violating data integrity.
- Deletion Anomaly: If we delete the record for teacher 2205 (the only row for school S114), we lose all information about "Green school" in Birmingham, including its name and its city, even though the school still exists.
- Update Anomaly: If "Park School" (schoolID S301) changes its name to "Park Academy", we must update the schoolName in multiple rows (the two rows where S301 appears), missing one row would lead to inconsistent data.

b. Normalization Process to 3NF:

Assumptions:

- NIN is unique to each teacher and serves as their primary identifier.
- schoolID is unique to each school
- ContractNo identifies a specific contract or assignment- period, a teacher can work at multiple school, under the same or different contracts, but not multiple times at the same school under the same contract (based on date)
- No other dependencies (eg schoolCity does not determine schoolName uniquely, as multiple schools could be in the same city)
- The table is already in 1NF (no multivalued attributes or repeating groups, beyond the flat structure)

Functional Dependencies:

- NIN → tName (teacher ID determines teacher name)
- schoolID → schoolName, schoolCity (schoolID determines the school name and city)
- {NIN, contractNo, schoolID} → hours (the combination determines hours worked; partial keys do not fully determine it).

The Unnormalized table has redundancies due to these dependencies.

Date: _____

TO 2NF: Eliminate Partial dependencies on the composite key {NIN, contractNO, schoolID}

Create TeacherTable: NIN (PK), tName

Create School Table: schoolID (PK), schoolName, schoolCity

Create Assignment Table: NIN (FK to Teacher), contractNO, schoolID (FK to school), hours (PK: {NIN, contractNO, schoolID})

TO 3NF: No transitive dependencies remain (e.g. no non-key attributes depend on other non-keys). The tables are now in 3NF.

Normalized Tables (with sample Data):

Teacher:

- NIN: 2103, tName: BrownA
- NIN: 2198, tName, Khan R
- NIN: 2205, tName, Lewis H

School:

- schoolID: S301, schoolName: Parkes school, schoolCity: Manchester
- schoolID: S114, schoolName: Green school, schoolCity: Birmingham.

Assignment

- NIN: 2103, contractNO: T9001, schoolID: S301, hours: 10
- NIN: 2198, contractNO: T9001, schoolID: S301, hours: 18
- NIN: 2205, contractNO: T9001, schoolID: S114, hours: 20
- NIN: 2103, contractNO: T9001, schoolID: S114, hours: 12.

QUESTION 4:

a. The primary key (PK) for the FILM_CASTING table is the composite key {filmID, ActorID} as per FD 4, which states that this combination determines all other attributes, and no single attribute alone determines the whole record.

b. The table violates 2NF (not in 2NF but could potentially satisfy lower forms). 2NF requires that no non prime attributes depends on a proper subset of the candidate key (partial dependency). Here the PK is {filmID, ActorID}, but there are partial dependencies eg filmID → filmTitle DirectorName (filmTitle and DirectorName depends only on filmID, not the full PK). This is partial dependency violation.

Date: _____

C. Decompose to 2NF by removing partial dependencies (FilmID → FilmTitle, DirectorName and ActorID → ActorName, AgentID).

- Film : FilmID (PK), FilmTitle, DirectorName
- Actor : ActorID (PK), ActorName, AgentID, AgentPhone.
- Casting : FilmID (FK to FILM), ActorID (FK to Actor) (PK: {FilmID, ActorID})

D. From 2NF, decompose ACTOR further to 3NF by removing the transitive dependency (ActorID → AgentID → AgentPhone).

- Film : FilmID (PK), FilmTitle, DirectorName
- Agent : AgentID (PK), AgentPhone
- Actor : ActorID (PK), ActorName, AgentID (FK to Agent)
- Casting : FilmID (FK to Film), ActorID (FK to Actor) (PK : {FilmID, ActorID})

E. Update Anomaly: If an agent's phone number changes, it would require updating AgentPhone in every row where that agent (via the actor) appears. Missing any row would lead to inconsistent data.

QUESTION 5:

1. Functional Dependencies and Assumptions:

Assumptions:

- StudentID is unique and identifies a single student.
- CourseID is unique and identifies a single course.
- A student can enroll in multiple courses, and the enrollment details (like Date) are specific to each student-course combination.
- CourseName, Credits, and InstructorName are attributes of the course and do not change per enrollment.
- EnrollmentDate : is the date when the student enrolled in that specific course.
- There are no additional attributes or relationships (eg no separate InstructorID, InstructorName is directly tied to the course).
- The form represents a repeating group for courses, implying an unnormalized structure.

Date: _____

Functional Dependencies:

- StudentID → lastName, firstName, Address,
- CourseID → CourseName, Credits, InstructorName
- {StudentID, CourseID} → EnrollmentDate.

2. Normalization Process to 3NF:

The form represents an unnormalized relation with a repeating group for courses:

UNF: ENROLLMENT_FORM (StudentID, lastName, firstName, Address, (CourseID, CourseName, Credits, InstructorName, EnrollmentDate)*) where * denotes repeating group.

1NF: Remove repeating groups by creating a flat table with one row per enrollment:

ENROLLMENT (StudentID, lastName, firstName, Address, CourseID, CourseName, Credits, InstructorName, EnrollmentDate).

Candidate key: {StudentID, CourseID}

2NF: Eliminate partial dependencies (non-prime attributes depending on part of the key)

- lastName, firstName, Address depend only on StudentID (partial dependency)
- CourseName, Credits, InstructorName depend only on CourseID (partial dependency)

Decompose into:

- Student (StudentID, lastName, firstName, Address)
- Course (courseID, courseName, Credits, InstructorName)
- Enrollment (StudentID, CourseID, EnrollmentDate)

3NF: Check for transitive dependencies (non-prime attributes depending on another non prime).

Non Exist:

- In Student, all depend on PK.
- In Course, all depend on PK
- In Enrollment, EnrollmentDate depends on the full PK.

The relations are now in 3NF.

3. Keys in 3NF relations:

- STUDENT: Primary key: StudentID, No alternate keys, No foreign key.
- COURSE: Primary key: CourseID, No alternate keys, No foreign key
- ENROLLMENT: Primary key: {StudentID, CourseID}, No alternate keys, Foreign keys: StudentID (references STUDENT.StudentID), CourseID (references COURSE.CourseID).