Cinema Hall Ticket Booking System

Ashish Dabhi - 202303039
Meet Patel - 202303012
Bhavin Dabhi - 202303041

Functional Dependencies (FDs) and Normal Forms:

Users Table

- Functional Dependencies (FDs):
  - `user_email` → `name`, `phone_number`, `password`
- Normal Form (NF) Status:
  - The table is in **1NF** (atomic values).
  - It's in **2NF** (no partial dependencies, as `user_email` is the primary key).
  - It's in **3NF** (no transitive dependencies).

Admin Table

- FDs:
  - `admin_email` → `name`, `phone_number`, `password`, `cinema_name`, `cinema_pincode`
- NF Status:
  - The table is in **1NF**, **2NF**, and **3NF**.

City Table

- FDs:
  - `city_name` → None (just a single attribute)
- NF Status:
  - **1NF**, **2NF**, and **3NF** (trivial dependencies).

Cinema Table

- FDs:
  - `(cinema_name, cinema_pincode)` → `cinema_area`, `cinema_city`, `total_screens`

    - `cinema_city` → `city_name` (because cinema_city refers to the `City` table)
- NF Status:
    - **1NF** (all atomic values).
    - In **2NF** since all non-prime attributes depend on the whole composite key.
    - In **3NF** (no transitive dependencies).

Movie Table

- FDs:
    - `movie_title` → `genre`, `duration`, `release_date`, `price`
- NF Status:
    - **1NF**, **2NF**, and **3NF**.

Showtime Table

- FDs:
    - `showtime_id` → `cinema_name`, `cinema_pincode`, `movie_title`, `screen_id`, `show_date`, `start_time`, `end_time`
- NF Status:
    - **1NF**, **2NF**, and **3NF**.

Screen Table

- FDs:
    - `screen_id` → `cinema_name`, `cinema_pincode`, `capacity`
- NF Status:
    - **1NF**, **2NF**, and **3NF**.

Booking Table

- FDs:
    - `booking_id` → `user_email`, `showtime_id`, `booking_date`, `seat_number`
- NF Status:
    - **1NF**, **2NF**, and **3NF**.

Cancellation Table

- FDs:
    - `{cancellation_date, user_email} → showtime_id, booking_date, seat_number`

- NF Status:
  - **1NF**, **2NF**, and **3NF**.

Revenue Table

- FDs:
  - `(cinema_name, cinema_pincode, movie_title)` → `booked_seats`, `total_revenue`
- NF Status:
  - **1NF**, **2NF**, and **3NF**.

Payment Table

- FDs:
  - `transaction_id` → `amount`, `showtime_id`, `user_email`
- NF Status:
  - **1NF**, **2NF**, and **3NF**.

All tables in the Cinema Hall Ticket Booking System are in 3NF and BCNF.

Insert, Update, and Delete Anomalies:

Insert Anomalies

An **insert anomaly** occurs when certain attributes cannot be inserted into the database without the presence of other attributes.

In this design, the following potential insert anomalies are avoided:

- **Cinema Table**: We can't insert a cinema without specifying its name, area, city, and pincode, which is good because all this information is needed for a complete cinema record.
- **Movie Table**: No movie can be inserted without specifying essential details such as title, genre, duration, release date, and price, avoiding any incomplete movie entries.

Update Anomalies

An **update anomaly** occurs when a single update needs to be made in multiple rows, leading to potential inconsistencies.

This design avoids update anomalies:

- **Cinema and Showtime Tables**: If a cinema's pincode changes, we only need to update it in the `Cinema` table. Thanks to the foreign key relationships, all references in the `Showtime`, `Admin`, and `Screen` tables automatically update due to the `ON UPDATE CASCADE` clause.
- **Movie Table**: Any change in movie details, like price or genre, only requires updating in one place (the `Movie` table) and all showtimes using that movie automatically reflect the change via foreign key relationships.

Delete Anomalies

A **delete anomaly** occurs when deleting data causes unintended loss of other data.

This design avoids delete anomalies:

- **Movie and Showtime Tables**: If a movie is removed from the database, all associated showtimes will be automatically deleted due to the `ON DELETE CASCADE` constraint. This ensures that no orphan showtimes exist for a non-existent movie.
- **Cinema Table**: Deleting a cinema will automatically remove all related records in the `Admin`, `Showtime`, `Screen`, and `Revenue` tables, maintaining referential integrity and avoiding orphan records.