Section: 09
Course: CPS510 - Database Systems 1
Team Number: 17
Assignment Number: A8
TA Name: Reza Barati
Team Members: Anmol Panchal, Deep Patel, Aryan Patel

Firstly, let's acknowledge the baseline conditions required for a table to be in BCNF.

1. No repeating groups, and each column having atomic values ← (1NF)
2. All non-keyed attributes as functional dependencies on the primary key + 1NF ← (2NF)
3. All transitive dependencies must be removed + 2NF ← (3NF)
4. All non-trivial, left irreducible FD has a candidate key as determinant ← (BCNF)

Bernstein's Algorithm:
The steps offered by Bernstein are used to decompose our database to BCNF tables.
Step 1.) Start with any relation, denoted R, with its set of functional dependencies, denoted F.
Step 2.) Remove all partial dependencies by decomposing the relation R.
Step 3.) Repeat for all tables, until they satisfy the BCNF constraints.

Film Table:
Candidate Key: film_id
Functional Dependency: film_id → title, runtime, release_year, director_id
director_id → title, runtime, release_year

A transitive dependency exists with director_id → title, runtime, release_year because film_id → director_id. To mitigate this, the functional dependencies have been split to enable a transitive dependency table to handle BCNF constraints with director_id → title, runtime, release_year.

This table is in BCNF because the LHS of each FD exhibits a candidate key.

Actor Table:
Candidate Key: actor_id
Functional Dependencies: actor_id → first_name, last_name, birthdate, nationality

This table is already in BCNF because the candidate key and functional dependencies are present, where the candidate key director_id is also a superkey on the LHS. We know this because all attributes are within the set of functional dependencies, thus actor_id is a candidate key.

Also in the last assignment, it was brought to 3NF by decomposing the filmography attribute.

Director Table:
Candidate Key: director_id
Functional Dependencies: director_id → first_name, last_name, birthdate, nationality

This table is already in BCNF because the candidate key and functional dependencies are present, where the candidate key director_id is also a superkey on the LHS. We know this because all attributes are within the set of functional dependencies, thus director_id is a candidate key.

Also in the last assignment, it was brought to 3NF by decomposing the filmography attribute.

Producer Table:
Candidate Key: producer_id
Functional Dependencies: producer_id → first_name, last_name, birthdate, nationality

This table is already in BCNF because the candidate key and functional dependencies are present, where the candidate key producer_id is also a superkey on the LHS. We know this because all attributes are within the set of functional dependencies, thus producer_id is a candidate key.

Also in the last assignment, it was brought to 3NF by decomposing the filmography attribute.

Review Table:
Candidate Key: review_id
Functional Dependencies: review_id → user_id, film_id, theDescription, rating, theDate

This table is already in BCNF because the candidate key and functional dependencies are present, where the candidate key review_id is also a superkey on the LHS. We know this because all attributes are within the set of functional dependencies, thus review_id is a candidate key.

Studio Table:
Candidate Key: studio_id
Functional Dependencies: studio_id → name, owner, location, credits
location → studio_id

This table is already in BCNF because the candidate key and functional dependencies are present, where the candidate key studio_id is also a superkey on the LHS. We know this because all attributes are within the set of functional dependencies, thus studio_id is a candidate key.

Awards Table:
Candidate Key: award_id
Func. Dependencies: award_id → theName, presenter, year_of_win, receiver_id, winner_type

This table is already in BCNF because the candidate key and functional dependencies are present, where the candidate key award_id is also a superkey on the LHS. We know this because all attributes are within the set of functional dependencies, thus award_id is a candidate key.

Receiver Table:
Candidate Key: receiver_id
Functional dependencies: receiver_id → theName, theDate

This table is already in BCNF because the candidate key and functional dependencies are present, where the candidate key receiver_id is also a superkey on the LHS. We know this because all attributes are within the set of functional dependencies, thus receiver_id is a candidate key.

TheUser Table:
Candidate Key: user_id
Functional dependencies: user_id → first_name, last_name, username, email, password

This table is already in BCNF because the candidate key and functional dependencies are present, where the candidate key user_id is also a superkey on the LHS. We know this because all attributes are within the set of functional dependencies, thus user_id is a candidate key.

Junction Tables:
The junction tables help fulfill the M:N relationships between some tables. In regards to its contribution in BCNF, we are reducing redundancy by decomposition (done in the previous assignment and improved upon in this assignment), and also taking the tables to BCNF using the Bernstein Algorithm outlined previously.

Conclusion:
This concludes the normalization process for our MovieDB. The next step is to implement the CRUD (create, read, update, delete) functionalities in a GUI.

The proposed method of implementation is using a Next.js project to create a frontend with buttons, text fields, and graphics, and use an API through Node.js called ‘oracledb’ which will allow us to comminute with the Oracle 11g database we have been building on sqldeveloper.