EZ Training: Normalized Relational Model

1. Mapping EER Model to Relational Schema

Starting from the EER model, we mapped each entity, relationship, and attribute to form a relational schema while addressing attributes that were multivalued, composite, or weak. Below is the process in detail:

Entities and Attributes

- **Individual**:
    - `Individual` (**Person_ID**, First_name, MI, Last_name, Age, Phone, Sex)
- **Employee** (subclassed into roles like `Group Trainer` and `Nutritionist`):
    - `Employee` (**Employee_ID**, Person_ID(FK), Role_ID(FK), Pay_rate, Date_of_joining, Employment_type)
    - `Role` (**Role_ID**, Role_name)
    - `Employee_Role` (**Employee_ID**(FK), **Role_ID**(FK)) – Handles employees with multiple roles
- **Shift** (a composite attribute of `Employee`):
    - `Shift` (**Employee_ID**(FK), Day, Time)
- **Member**:
    - `Member` (**Member_ID**, Person_ID(FK), Membership_status, Height, Weight, BMI, VR_Preferences, Body_measures)
    - `Medical_Condition` (**Condition_ID**, Member_ID(FK), Condition_name) – Addresses multivalued `Medical_Condition`
- **Group Trainer** (subclass of `Employee`):
    - `Group_Trainer` (**Employee_ID**(FK), Certification, Specialization, Rating)
- **Nutritionist** (subclass of `Employee`):
    - `Nutritionist` (**Employee_ID**(FK), Certifications, Experience_level, Active_clients)
- **Workout Log** (initially a weak entity capturing workout details):
    - `Workout_Log` (**Log_ID**, Member_ID(FK), Workout_Session_ID(FK), Calories_burnt, Heart_rate, Oxygen_level)
    - `Workout_Log_Exercise` (**Log_ID**(FK), **Exercise_ID**(FK), Equipment_ID(FK), Sets, Reps, Weight) – Captures each exercise in the log with specific details
- **Nutrition Plan** (1:1 relationship with `Member`):
    - `Nutrition_Plan` (**Name**, **Member_ID**(FK), Goal, Duration, Calories, Protein, Carbs, Fat, Employee_ID(FK))

- **Workout Session**:
  - `Workout_Session` (**Session_ID**, Member_ID(FK), Workout_Log_ID(FK), Program_type, VR_Environment)
- **Exercise**:
  - `Exercise` (**Exercise_ID**, Name, Schedule)
- **Payment**:
  - `Payment` (**Payment_ID**, Member_ID(FK), Payment_date, Mode_of_payment, Amount)
- **VR Equipment**:
  - `VR_Equipment` (**Equipment_ID**, Type, Name, Status, Maintenance_schedule)

Relationships

To handle many-to-many (N:M) relationships:

- **Workout Session Conducts Exercise**:
  - `Workout_Session_Exercise` (**Session_ID**(FK), **Exercise_ID**(FK))
- **Workout Session Uses Equipment**:
  - `Workout_Session_Uses_Equipment` (**Session_ID**(FK), **Equipment_ID**(FK))
- **Member Consults Nutritionist**:
  - `Member_Consults_Nutritionist` (**Member_ID**(FK), **Employee_ID**(FK))

2. Normalizing Relations to BCNF

1NF (First Normal Form)

- **Step**: Removed multivalued and composite attributes.
  - **Example**: Split `Role` into `Employee_Role` for multiple roles.
  - **Example**: Split `Medical_Condition` into a separate table to store multiple conditions per member.
- **Outcome**: All attributes in each table now contain atomic values.

2NF (Second Normal Form)

- **Step**: Eliminated partial dependencies in tables with composite primary keys.
  - **Example**: In `Nutrition_Plan` and `Exercise`, ensured that each non-key attribute depends on the entire composite primary key.
- **Outcome**: No partial dependencies, satisfying 2NF.

3NF (Third Normal Form)

- **Step**: Removed transitive dependencies.
  - **Example**: `Role` was separated as its own table with `Role_ID` used as a foreign key in `Employee` to eliminate dependencies.
  - **Example**: `Workout_Log_Exercise` table was introduced to avoid transitive dependencies in `Workout_Log`.
- **Outcome**: Each non-key attribute depends directly on the primary key, satisfying 3NF.

BCNF (Boyce-Codd Normal Form)

- **Step**: Ensured every determinant in non-trivial functional dependencies is a superkey.
  - **Example**: Confirmed `Workout_Log_Exercise` uses a composite key (`Log_ID`, `Exercise_ID`) to capture unique entries and avoid overlapping candidate keys.
  - **Example**: Checked each table to ensure no dependencies were based on non-superkey determinants.
- **Outcome**: Schema fully complies with BCNF, ensuring data integrity and minimal redundancy.

3. Evaluating for Potential Denormalization or Derived Attributes

After reaching BCNF, we evaluated opportunities for denormalization and derived attributes to improve performance where justified.

Denormalization for Performance

1. **Workout Log and Workout Log Exercise**
   - **Reason for Denormalization**: Queries frequently require combined workout log and exercise data (e.g., exercises, sets, reps, and equipment used in each log). Merging these tables would reduce joins and optimize query performance.
   - **Adjustment**: Merged `Workout_Log` and `Workout_Log_Exercise` into a single `Workout_Log` table with exercise-specific details included:
     - `Workout_Log`(**Log_ID**, Member_ID(FK), Workout_Session_ID(FK), Calories_burnt, Heart_rate, Oxygen_level, Exercise_ID(FK), Equipment_ID(FK), Sets, Reps, Weight)
   - **Trade-Off**: While this denormalization increases redundancy, it is manageable given the performance benefits. Proper application logic will mitigate update anomalies.

Derived Attributes for Frequent Calculations

1. **BMI in Member**
   - **Reason for Storing**: BMI can be derived from `Height` and `Weight`, but it is frequently accessed. Storing it reduces computational overhead for repeated

access.
   - **Adjustment**: Kept `BMI` as a derived attribute in `Member`. Update logic ensures it's recalculated when `Height` or `Weight` changes.
2. **Calories Burned in Workout_Log**
   - **Reason for Storing**: `Calories_Burnt` can be derived from `Sets`, `Reps`, `Weight`, and `Exercise_ID`. Storing it avoids recalculating it for each log retrieval.
   - **Adjustment**: Stored `Calories_Burnt` in `Workout_Log` and update it as necessary to maintain accuracy.

Final Denormalized Relational Model

After normalization to BCNF and selective denormalization, here's the final relational schema:

Individual:
`Individual` (**Person_ID**, First_name, MI, Last_name, Age, Phone, Sex)

Employee:
`Employee` (**Employee_ID**, Person_ID(FK), Role_ID(FK), Pay_rate, Date_of_joining, Employment_type)

Role:
`Role` (**Role_ID**, Role_name)

Employee Role:
`Employee_Role` (**Employee_ID**(FK), **Role_ID**(FK))

Shift:
`Shift` (**Employee_ID**(FK), Day, Time)

Member:
`Member` (**Member_ID**, Person_ID(FK), Membership_status, Height, Weight, BMI, VR_Preferences, Body_measures)

Medical Condition:
`Medical_Condition` (**Condition_ID**, Member_ID(FK), Condition_name)

Group Trainer:
`Group_Trainer` (**Employee_ID**(FK), Certification, Specialization, Rating)

Nutritionist:
`Nutritionist` (**Employee_ID**(FK), Certifications, Experience_level, Active_clients)

Workout Log (Denormalized):
`Workout_Log` (**Log_ID**, Member_ID(FK), Workout_Session_ID(FK), Calories_burnt, Heart_rate,

Oxygen_level, Exercise_ID(FK), Equipment_ID(FK), Sets, Reps, Weight)

Nutrition Plan:
`Nutrition_Plan` (**Name**, **Member_ID**(FK), Goal, Duration, Calories, Protein, Carbs, Fat, Employee_ID(FK))

Workout Session:
`Workout_Session` (**Session_ID**, Member_ID(FK), Workout_Log_ID(FK), Program_type, VR_Environment)

Exercise:
`Exercise` (**Exercise_ID**, Name, Schedule)

Payment:
`Payment` (**Payment_ID**, Member_ID(FK), Payment_date, Mode_of_payment, Amount)

VR Equipment:
`VR_Equipment` (**Equipment_ID**, Type, Name, Status, Maintenance_schedule)

Workout Session Exercise (Conducts):
`Workout_Session_Exercise` (**Session_ID**(FK), **Exercise_ID**(FK))

Workout Session Uses Equipment (Uses):
`Workout_Session_Uses_Equipment` (**Session_ID**(FK), **Equipment_ID**(FK))

Member Consults Nutritionist (Consults):
`Member_Consults_Nutritionist` (**Member_ID**(FK), **Employee_ID**(FK))

Summary Notes for Presentation

1. **EER to Relational Mapping**: We mapped each entity, weak entity, and relationship from the EER model to relational tables.
2. **Normalization to BCNF**: Through 1NF, 2NF, 3NF, and BCNF steps, we removed multivalued, composite attributes and addressed partial, transitive, and non-superkey dependencies.
3. **Denormalization for Performance**: Merged `Workout_Log` and `Workout_Log_Exercise` for frequently accessed combined data, improving performance.
4. **Derived Attributes**: Stored `BMI` in `Member` and `Calories_Burnt` in `Workout_Log` to avoid costly recalculations on each access.

This final model is optimized for both performance and data integrity, providing an efficient schema for frequent querying and ensuring data consistency.