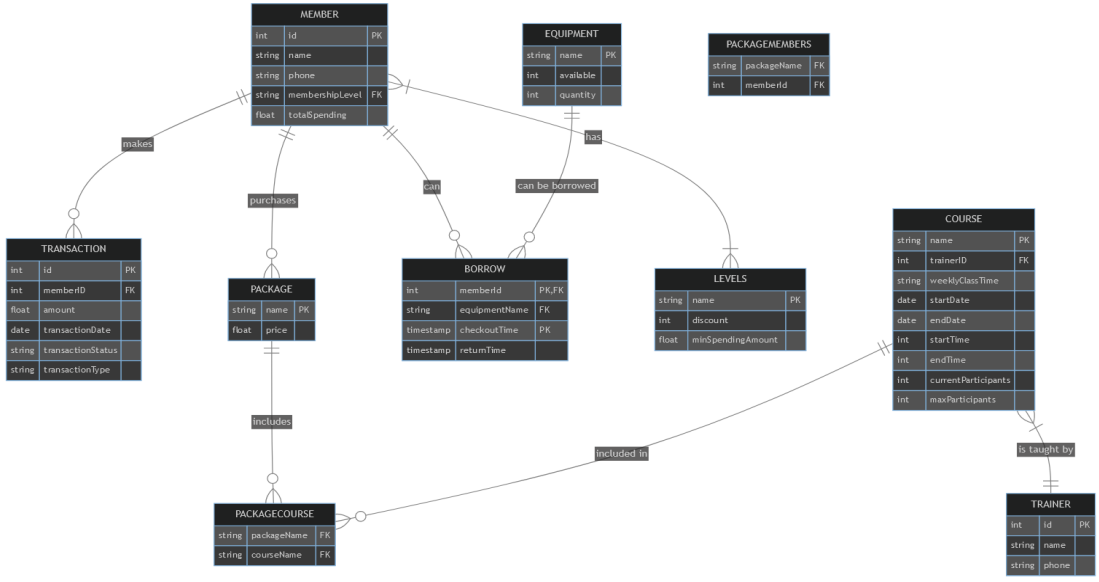
GitHub public repo:
https://github.com/alielbekov/theFitnessApp

Team Members:

Ali Elbekov,
Aman Bhatia,
Eduardo Esau Ibarra,
Otabek Abduraimov


MEMBER
int id PK
string name
string phone
string membershipLevel FK
float totalSpending
EQUIPMENT
string name PK
int available
int quantity
PACKAGEMEMBERS
string packageName FK
int memberId FK
makes
purchases
can
has
can be borrowed
TRANSACTION
int id PK
int memberID FK
float amount
date transactionDate
string transactionStatus
string transactionType
PACKAGE
string name PK
float price
BORROW
int memberId PK,FK
string equipmentName FK
timestamp checkoutTime PK
timestamp returnTime
LEVELS
string name PK
int discount
float minSpendingAmount
COURSE
string name PK
int trainerID FK
string weeklyClassTime
date startDate
date endDate
int startTime
int endTime
int currentParticipants
int maxParticipants
includes
included in
is taught by
PACKAGECOURSE
string packageName FK
string courseName FK
TRAINER
int id PK
string name
string phone

Constraints and Additional Notes:

1. Phone Numbers: Follow a specific format and are validated using regular expressions.
2. Membership Level: The 'MembershipLevel' in Member entity references the 'Name' in Levels entity, ensuring consistency in membership categorization.
3. Time Attributes: For Course timings, a 24-hour format is used.
4. Composite Primary Keys: In entities like PackageCourse and PackageMembers, composite primary keys are used to uniquely identify each record.

Design Rationale:

1. The design ensures normalization to avoid data redundancy.
2. Relationships are carefully crafted to reflect real-world associations between different aspects of a fitness app, like courses, trainers, members, and equipment.
3. The use of foreign keys maintains referential integrity across different entities.
4. Composite keys in junction tables (like PackageCourse) allow for flexible and efficient many-to-many relationships.

Relations:

Trainer:

id	INT	PK
name	STR	
phone	STR	

Levels:

name	STR	PK
discount	INT	
minSpendingAmount	FLT	

Member:

id	INT	PK
name	STR	
phone	STR	
membershipLevel	STR	FK
totalSpending	FLT	

Course:
- name STR PK
- trainerID INT FK
- weeklyClassTime STR
- startDate DATE
- endDate DATE
- startTime INT
- endTime INT
- currentParticipants INT
- maxParticipants INT

Package:
- name STR PK
- price FLT

PackageCourse:
- packageName STR PK, FK
- courseName STR PK, FK

PackageMembers:
- packageName STR PK, FK
- memberId INT PK, FK

Equipment:
- name STR PK
- available INT
- quantity INT

Borrow:
- memberId INT PK, FK
- equipmentName STR PK, FK
- checkoutTime TIMEST PK
- returnTime TIMEST

Transaction:

id	INT	PK
memberID	INT	FK
amount	FLT	
transactionDate	DATE	
transactionStatus	STR	
transactionType	STR	

Functional Dependencies:
Trainer:
{id} -> {name, phone}
Levels:
{name} -> {discount, minSpendingAmount}
Member:
{id} -> {name, phone, membershipLevel, totalSpending}
Course:
{name} ->
{trainerID, weeklyClassTime, startDate, endDate, startTime, endTime, currentParticipants, maxParticipants}
Package:
{name} -> {price}
PackageCourse:
(no FD's)
PackageMembers:
(no FD's)

Equipment:
{name} -> {available, quantity}
Borrow:
{memberId, equipmentName, checkoutTime} -> {returnTime}
Transaction:
{id} -> {memberID, amount, transactionDate, transactionStatus, transactionType}

Discussion on normal forms:

1NF: The tables in this project is in 1NF because we do not have any set valued attributes. All of the attributes in the tables are either integers, floats, strings, dates, and timestamps. None of these types are sets.

2NF: Every non-prime attribute is fully functionally dependent on the primary keys of their tables. Since primary keys are candidate keys, all tables must be in 2NF.

3NF/BCNF: In all FD's above, the determinate of the FD is always the full primary key of the table. This means that the determinate of all FD's identified are superkeys. This means that all tables are in BCNF and thus 3NF.

Query description:

The self-designed query finds the contact details of the members that have borrowed equipment that is currently out of stock. This information can be used to see where all the inventory is and also send periodic reminders to return the equipment on time. We can also send them a reminder to return the equipment if they are done using it in case we run out of a certain equipment.