Design

This document contains a database design for a fitness center. There are four sections:

1) Conceptual database design
2) Logical database design
3) Normalization analysis
4) Query description

The paper includes the results and the discussion of rationale behind the design. For the sake of simplicity as well as brevity, the design documentation does not include draft elements. The interested reader can find the drafts in the ‘documentation’ directory.

Conceptual Database Design

Overview

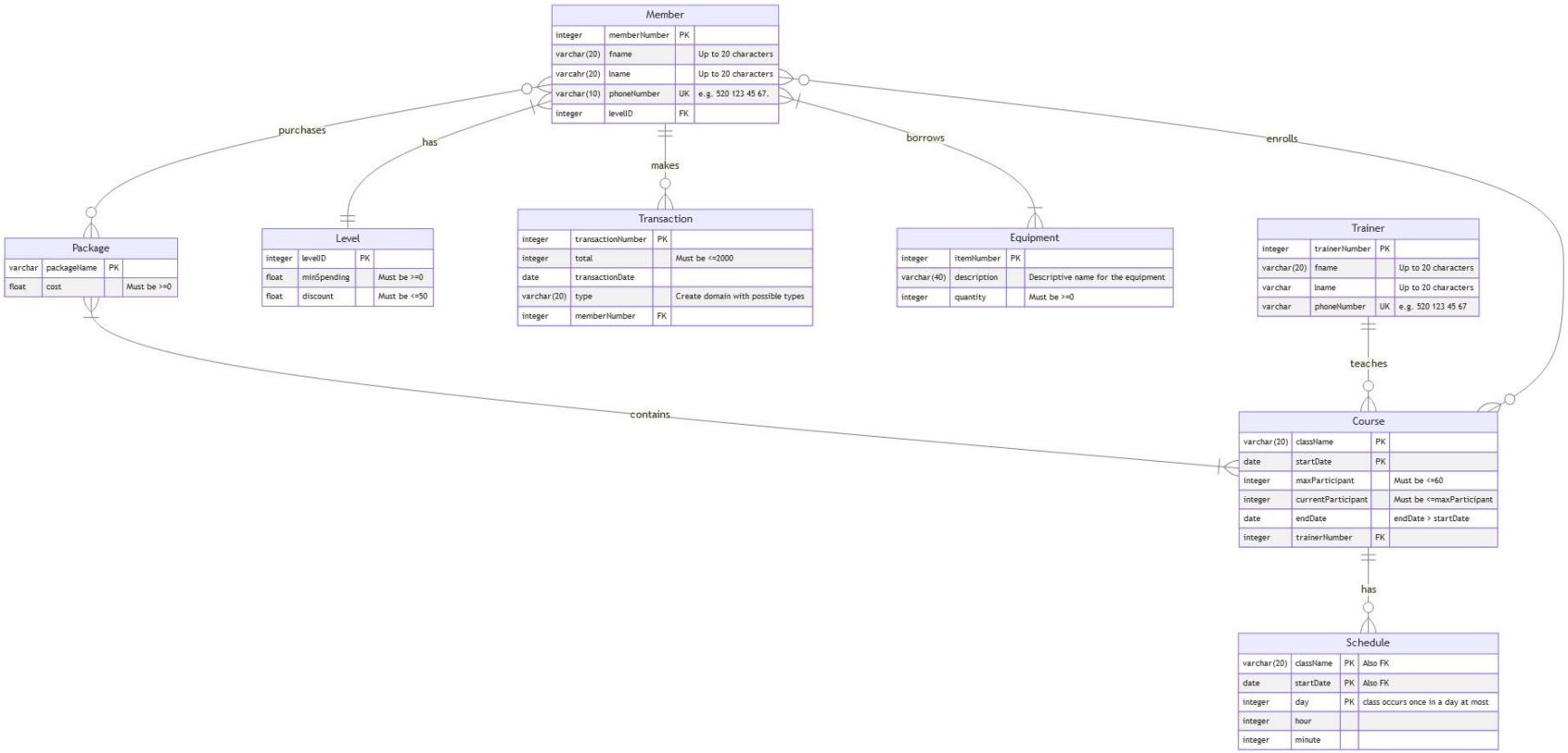
The objective of this step is to build a conceptual data model based on the data requirements of the gym. Software Mermaid is used to create the Entity-Relationship diagram. Due to software limitations, the attributes of relationships are presented in a separate page or can be viewed in OneNote using the link provided below. Some additional constraints that we could not show in the diagram are also included. In case, any of the diagram don't show up due to format limitations, please use the following links:

E-R Diagram Link Vector Image

E-R Diagram With Relationship Attributes in OneNote

It should be noted that the foreign keys are present in the final ER diagram.

GYM 460


Member
integer memberNumber PK
varchar(20) fname Up to 20 characters
varcahr(20) lname Up to 20 characters
varchar(10) phoneNumber UK e.g. 520 123 45 67.
integer levelID FK
purchases
has
makes
borrows
enrolls
Package
varchar packageName PK
float cost Must be >=0
Level
integer levelID PK
float minSpending Must be >=0
float discount Must be <=50
Transaction
integer transactionNumber PK
integer total Must be <=2000
date transactionDate
varchar(20) type Create domain with possible types
integer memberNumber FK
Equipment
integer itemNumber PK
varchar(40) description Descriptive name for the equipment
integer quantity Must be >=0
Trainer
integer trainerNumber PK
varchar(20) fname Up to 20 characters
varchar lname Up to 20 characters
varchar phoneNumber UK e.g. 520 123 45 67
teaches
contains
Course
varchar(20) className PK
date startDate PK
integer maxParticipant Must be <=60
integer currentParticipant Must be <=maxParticipant
date endDate endDate > startDate
integer trainerNumber FK
has
Schedule
varchar(20) className PK Also FK
date startDate PK Also FK
integer day PK class occurs once in a day at most
integer hour
integer minute

Relationship Attributes

purchases			
integer	transactionNumber	FK	
varchar	packageName	FK	

borrows			
integer	loanNumber	PK	
integer	quantity	<=2	Don't allow the member to borrow more than two items of the same type.
Date	borrowedDate		
Date	returnedDate		returnedDate >= borrowedDate
integer	itemNumber	FK	
integer	memberNumber	memberNumber	

Additional Constraints

1.Deletion of a member. All equipment borrowed by the members should be returned. If the total of transaction amounts for the use is <0, do not allow the deletion of the member. Remove the user from the enrolled course and update the number of active participants in those courses.

2.Deletion of a course. If the course does not end and has members, notify users (best to do programmatically.).

3.If a course package is modified, this should not impact members.

4.Only a user with admin credentials should be able to delete, update, and insert a course package, equipment, and courses. Only administrators can view information about all members, schedules, and trainers.

Logical Database Design

General

To show the process of how we derived the relations, we will use the draft ER diagram since the final ER diagram is the result of normal analysis done on the original ER-based relations.

Entity Types

For each entity type, create a relation that contains all simple attributes.
Package (packageName, cost)
Primary Key packageName

Trainer (trainerNumber, fname, lname, phoneNumber)
Primary Key trainerNumber

Course (classNmae, maxParticipant, currentParticipant, startDate, endDate)
Primary Key className, startDate

Member (memberNumber, fname, lname, phoneNumber)
Primary Key memberNumber

Transaction (transactionNumber, total, transactionDate, type)
Primary Key transactionNumber

Equipment (itemNumber, description, quantity)
Primary Key itemNumber

Level (levelId, minSpending, discount)
Primary Key levelId

One-to-many binary relationship types

For each 1:* binary relationship, the entity on the one side is considered a parent, and the entity on the many side is considered a child. We use the primary key attribute of the parent entity as a foreign key in a child entity.

Package (packageName, cost)
Primary Key packageName

Tainer (trainerNumber, fname, lname, phoneNumber)
Primary Key trainerNumber

Course (classNmae, maxParticipant, currentParticipant, startDate, endDate, trainerNumber, day, hour, minute)

Primary Key className, startDate; Foreign Key: trainerNumber;

Member (memberNumber, fname, lname, phoneNumber, levelId)
Primary Key memberNumber; Foreign Key: levelId

Transaction (transactionNumber, total, transactionDate, type, memberNumber)
Primary Key transactionNumber; Foreign Key: memberNumber

Equipment (itemNumber, description, quantity)
Primary Key itemNumber

Level (levelId, minSpending, discount)
Primary Key levelId

One-to-one binary relationship types.

N/A

One-to-one recursive relationships

N/A

Superclass/subclass relationship types.

N/A

Many-to-many binary relationship types.

For each many-to-many relationship, create a relation that includes the primary keys of each entity type.

Then, the new relations:

CoursePackage (packageName, className, startDate)
Primary Key: packageName, className, startDate; Foreign Key: packageName, className, startDate

Purchase (packageName, transactionNumber)
Primary Key: packageName, transactionNumber;
Foreign Key: packageName, transactionNumber;

EquipmentLoan (loanNumber, itemNumber, memberNumber, quantity, borrowedDate, returnedDate)
Primary key: loanNumber; Foreign Key: itemNumber, memberNumber

Enrollment (memberNumber, courseName, startDate)
Primary Key: memberNumber, courseName, startDate

Note: the primary key cannot be formed from itemNumber, memberItem and borrowedDate since the same item can be borrowed by the same user several times at the same day.

Then, the final set of relations before the normalization analysis are:

Package (packageName, cost)
Primary Key packageName

Tainer (trainerNumber, fname, lname, phoneNumber)
Primary Key trainerNumber

Course (className, startDate, maxParticipant, currentParticipant, endDate, trainerNumber)
Primary Key: className, startDate; Foreign Key: trainerNumber

Schedule (className, startDate, day, hour, minute, duration)
Primary Key: className, startDate, day; Foreign Key: className, startDate

Member (memberNumber, fname, lname, phoneNumber, levelId)
Primary Key memberNumber; Foreign Key: levelId

Transaction (transactionNumber, total, transactionDate, type, memberNumber)
Primary Key transactionNumber; Foreign Key: memberNumber

Equipment (itemNumber, description, quantity)
Primary Key itemNumber

MembershipLevel (levelId, minSpending, discount)
Primary Key levelId

CoursePackage (packageName, className, startDate)
Primary Key: packageName className, startDate; Foreign Key: packageName, className, startDate

Purchase (packageName, transactionNumber)
Primary Key: packageName, transactionNumber;
Foreign Key: packageName, transactionNumber;

EquipmentLoan (loanNumber, itemNumber, memberNumber, quantity, borrowedDate, returnedDate)
Primary key: loanNumber; Foreign Key: itemNumber, memberNumber

Enrollment (memberNumber, courseName, startDate)
Primary Key: memberNumber, courseName, startDate

The final results are in ‘Logical Relations Version 2.txt’.

Normalization Analysis

The original analysis of the relations can be read in 'Normalization Analysis.pdf'. This section performs the normalization analysis on final relations.

FDs

Package (packageName, cost)
packageName -> cost

Tainer (trainerNumber, fname, lname, phoneNumber)
trainerNumber -> fname, lname, phoneNumber

Course (className, startDate, maxParticipant, currentParticipant, endDate, trainerNumber)
className, startDate -> maxParticipant, currentParticipant, endDate, trainerNumber

Schedule (className, startDate day, hour, minute, duration)
className, startDate, day -> hour, minute, duration
className, startDate -> maxParticipant, currentParticipant, endDate, trainerNumber (Foreign Key)

Member (memberNumber, fname, lname, phoneNumber, levelId)
memberNumber -> fname, lname, phoneNumber, levelId

Transaction (transactionNumber, total, transactionDate, type, memberNumber)
transactionNumber -> total, transactionDate, type, memberNumber
memberNumber ⟶ fname, lname, phoneNumber, levelId (Foreign Key)

Equipment (itemNumber, description, quantity)
itemNumber -> description, quantity

MembershipLevel (levelId, minSpending, discount)
levelId -> minSpending, discount

CoursePackage (packageName, className, startDate)
packageName, className, startDate -> packageName, className, startDate
startDate, className -> packageName

Purchase (packageName, transactionNumber)
memberNumber, packageName, transactionNumber -> packageName, transactionNumber;
packageName ⟶ cost (Foreign Key)
transactionNumber ⟶ total, transactionDate, type, memberNumber (Foreign Key)

EquipmentLoan (loanNumber, itemNumber, memberNuber, quantity, borrowedDate, returnedDate)
loanNumber -> itemNumber, memberItem, quantity, borrowedDate, returnedDate
itemNumber -> description, quantity(different from this.quantity Foreign Key)

Enrollment (memberNumber, courseName, startDate)
memberNumber, courseName, startDate -> memberNumber, courseName, startDate
memberNumber ⟶ {fname, lname, phoneNumber, levelId} (Foreign Key)
className, startDate -> maxParticipant, currentParticipant, endDate, trainerNumber (Foreign Key)

Sample values

To effectively perform the process of normalization, let's populate the relations with some sample tuples.

Package

packageName	cost
Beginner	100
Intermediate	125

Trainer

trainerNumber	fname	lname	phoneNumber
1	Umid	Muzrapov	5202555555
2	Lola	Ahmedova	5207777777

Course

className	maxParticipant	currentParticipant	startDate	endDate	trainerNumber
Yoga 001	60	35	05/11/2023	07/11/2023	1
Strength 002	55	30	05/12/2023	07/11/2023	2

Schedule

className	startDate	day	hour	minute	duration
Yoga 001	05/11/2023	1	13	30	60
Yoga 001	05/11/2023	3	16	30	60
Strength 002	05/12/2023	2	13	20	50
Strength 002	05/12/2023	4	12	45	45

Member

memberNumber	fname	lname	phoneNumber	levelId
1	Pika	Chu	5201112233	1
2	Smith	John	5208888888	2

Transaction

transactionNumber	total	transactionDate	type	memberNumber
1	100	05/06/2023	Paid	1
2	125	05/03/202	Unpaid	2

Equipment

itemNumber	description	quantity
1	football	20
2	basketball	10

Level

LevelId	minSpending	discount
Gold Member	1000	20
Diamon Member	500	30

CoursePackage

packageName	className	startDate
Beginner	Yoga 001	05/11/2023
Intermediate	Strength 002	07/11/2023

Purchase

packageName	transactionNumber
Beginner	1
Intermediate	2

EquipmentLoan

loanNumber	itemNumber	memberItem	quantity	borrowedDate	returnedDate
11	1	1	2	06/05/2023	Null
12	2	2	1	05/20/2023	06/20/2023

Enrollment

memberNumber	courseName	startDate
1	Yoga 001	05/11/2023

2	Strength 002	05/12/2023

First Normal Form (INF)

In the first normal form, the intersection of each row and column contains one and only one value. None of the intersections contains several values. Hence, our relations are in the first normal form.

Second Normal Form.

A relation that is in 1NF and every non-primary-key attribute is fully functionally dependent on the primary key is in the second normal form.

We can skip relations with a single-attribute primary key because they are automatically in at least 2NG.

Course (className, startDate, maxParticipant, currentParticipant, endDate, trainerNumber)

The relation is in 2NF. Primary key is className, startDate which determines the other attributes. No partial or transitive dependencies.

Schedule (className, startDate, day, hour, minute, duration)

Schedule is in 2NF. Primary key of className, startDate, day determines the other attributes. There are no partial dependencies.

CoursePackage (packageName, className, startDate)

The relation is in 2NF since there are no partial dependencies.

Purchase (memberNumber, packageName, transactionNumber):

This table is in 2NF as all non-prime attributes are fully functionally dependent on the composite primary key {memberNumber, packageName, transactionNumber}.

EquipmentLoan (loanNumber, itemNumber, memberItem, quantity, borrowedDate, returnedDate):

This table is in 2NF as all non-prime attributes are fully functionally dependent on the candidate key (loanNumber)

Enrollment (memberNumber, courseName, startDate):

This table is in 2NF because all non-prime attributes are fully functionally dependent on the composite primary key {memberNumber, courseName, startDate}.

This concludes our analysis of relations again second normal form.

Third Normal Form

First, let us give us the definition of the third normal form. A relation is in the first normal form if it is in second normal form and no non-primary-key attribute is transitively dependent on the primary key.

Package (packageName, cost)

packageName -> cost

The relation is in the third normal form since cost does not depend on packageName transitively.

Tainer (trainerNumber, fname, lname, phoneNumber)

trainerNumber -> fname, lname, phoneNumber

No non-primary key depends on the primary key transitively. Hence, the relation is in the third normal form.

Course (className, startDate, maxParticipant, currentParticipant, endDate, trainerNumber)

className, startDate -> maxParticipant, currentParticipant, endDate, trainerNumber

trainerNumber -> fname, lname, phoneNumber (Foreign key)

No non-primary key depends on the primary key transitively. Hence, the relation is in the third normal form.

Schedule (className, startDate day, hour, minute, duration)

className, startDate, day -> hour, minute, duration

className, startDate -> maxParticipant, currentParticipant, endDate, trainerNumber (Foreign Key)

All non-primary keys do not depend on the composite primary key transitively.

Member (memberNumber, fname, lname, phoneNumber, levelId)

memberNumber -> fname, lname, phoneNumber, levelId

Transaction (transactionNumber, total, transactionDate, type, memberNumber)

transactionNumber -> total, transactionDate, type, memberNumber

memberNumber → fname, lname, phoneNumber, levelId (Foreign Key)

This relation is in 3NF. All non-prime attributes are directly dependent on the primary key.

Equipment (itemNumber, description, quantity)

itemNumber -> description, quantity

This relation is in 3NF. All non-prime attributes are directly dependent on the primary key.

MembershipLevel (levelId, minSpending, discount)

levelId -> minSpending, discount

All non-primary keys do not depend on the composite primary key transitively. Hence, the relation is in third normal form.

CoursePackage (packageName, className, startDate)

packageName, className, startDate -> packageName, className, startDate

startDate, className -> packageName

Purchase (packageName, transactionNumber)

memberNumber, packageName, transactionNumber -> packageName, transactionNumber;

packageName → cost (Foreign Key)

transactionNumber → total, transactionDate, type, memberNumber (Foreign Key)

All non-primary keys do not depend on the composite primary key transitively.

EquipmentLoan (loanNumber, itemNumber, memberNuber, quantity, borrowedDate, returnedDate)

loanNumber -> itemNumber, memberItem, quantity, borrowedDate, returnedDate

itemNumber -> description, quantity(different from this.quantity Foreign Key)

All non-primary keys do not depend on the composite primary key transitively. Hence, the relation is in third normal form.

Enrollment (memberNumber, courseName, startDate)

memberNumber, courseName, startDate -> memberNumber, courseName, startDate

memberNumber → {fname, lname, phoneNumber, levelId} (Foreign Key)

className, startDate -> maxParticipant, currentParticipant, endDate, trainerNumber (Foreign Key)

This relation is in 3NF. All non-prime attributes are directly dependent on the primary key.

Hence, all tables adhere to 3NF. It might be surprising that our relations did not violate any conditions up to 3NF. However, please consider that the document merely proves that carefully designed relations adhere up to 3NF. The analysis and evolution can be seen in other documents included in the repository.

Query Description

The useful query is “What are courses along with their schedules for a certain member ordered by day of the week?”. Often, members might forget their schedule, or they may just want to have a copy of their schedule along with course names. Hence, adding the feature that can answer that question is extremely handy.

e.g.

Input: Umid Muzrapov

Output:

Umid Muzrapov’s schedule

Yoga 100. Monday, 15:30. Duration: 60 minutes.

Strength 1000000. Monday 14:20. Duration: 120 minutes.