Department IDI

Databases - IDTG2002

Fitness Tracker Database Design

Sanitized cover

Table of Contents

1 Background

Our fitness tracking application is designed to help users monitor and improve their health and fitness by providing comprehensive tracking of workouts, health metrics, goals, and progress. The app leverages a robust database system to store and manage user data efficiently. Built using Python and Streamlit, the application implements CRUD (Create, Read, Update, Delete) operations to ensure seamless interaction with the database.

1.1 Outlining Our Domains

The database design includes several tables, each serving a specific purpose. The `Users` table stores essential user information such as name, weight, date of birth, and sex. The `Health` table tracks various health metrics, including heart rate, VO2 max, heart rate variation, and sleep time, with each entry linked to a specific user. The `Goals` table allows users to set and monitor their fitness goals, detailing the goal name, amount, metric, and completion status.

Workouts are recorded in the `Workout` table, capturing details like start and end times, maximum heart rate, and workout type. Specific workout types, such as running and weightlifting, have dedicated tables (`Run` and `Weightlift`) to store relevant data like intervals, distances, pace, exercises, sets, reps, and weights. The `Exercise` table catalogs various exercises, including their names and muscle groups targeted.

This structured approach ensures that all aspects of a user's fitness journey are meticulously tracked, providing valuable insights and facilitating personalized fitness plans. The integration of Streamlit enhances the user experience by offering an intuitive and interactive interface for data visualization and analysis.

2 Database design

We used a top-down approach when designing the database. The tables we identified during our analysis were connected based on their respective relationships and modeled in the following EER-diagram:

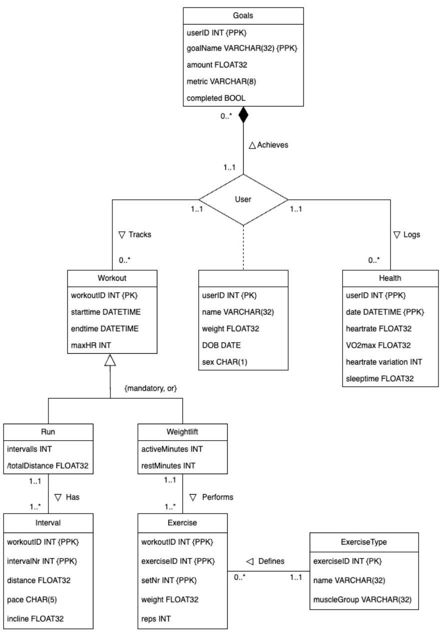

Figure 1: EER-diagram for the fitness tracking app

2.1 Data Dictionary

The following tables represent the data dictionary derived from our EER diagram. Each table corresponds to a database entity and defines its attributes, data types, keys, and relevant constraints. These structures provide a clear foundation for implementation, ensuring consistency, integrity, and compliance with normalization principles.

The data dictionary complements the EER diagram and served as a critical reference throughout the normalization process described in Section 2.2.

Entity: Goals

Attribute	Data Type	Description	Key
userID	INT	Reference to User	PPK
goalName	VARCHAR(32)	Name of the goal	PPK
amount	FLOAT	Goal target amount	
metric	VARCHAR(8)	Metric for measuring the goal	
completed	BOOL	Whether the goal is completed	

Figure 2: Data dictionary for Goals

Entity: User

Attribute	Data Type	Description	Key
userID	INT	Unique identifier for the user	PK
fName	VARCHAR(32)	First name of the user	
lName	VARCHAR(32)	Last name of the user	
weight	DECIMAL(4,1)	Weight of the user	
DOB	DATE	Date of birth	
sex	CHAR(1)	Sex ('M' or 'F')	

Figure 3: Data dictionary for User

Entity: Health

Attribute	Data Type	Description	Key
userID	INT	Reference to User	PPK
date	DATE	Date of the health record	PPK
heartrate	FLOAT	Heart rate on the given date	
VO2max	FLOAT	VO2 max on the given date	
HRvariation	INT	Heart rate variation	
sleeptime	FLOAT	Sleep time in hours	

Figure 4: Data dictionary for Health

Entity: Workout

Attribute	Data Type	Description	Key
workoutID	INT	Unique identifier for the workout	PK
userID	INT	Reference to User	FK
startTime	DATETIME	Start time of the workout	
endTime	DATETIME	End time of the workout	
maxHR	INT	Maximum heart rate during workout	
workoutType	VARCHAR(20)	Type of workout ('Run' or 'Weightlift')	

Figure 5: Data dictionary for Workout

Entity: Run

Attribute	Data Type	Description	Key
workoutID	INT	Reference to Workout	PPK
intervalNr	INT	Interval number within the workout	PPK
distance	DECIMAL(6,2)	Distance in the interval	
pace	CHAR(5)	Pace during the interval	
incline	DECIMAL(3,1)	Incline during the interval	

Figure 6: Data dictionary for Run

Entity: Weightlift

Attribute	Data Type	Description	Key
workoutID	INT	Reference to Workout	PPK
exerciseID	INT	Reference to Exercise	PPK
setNr	INT	Set number within the workout	PPK
reps	INT	Number of repetitions in the set	
weight	DECIMAL(4,1)	Weight used for the set	

Figure 7: Data dictionary for Weightlift

Entity: Exercise

Attribute	Data Type	Description	Key
exerciseID	INT	Unique identifier for the exercise	PK
name	VARCHAR(32)	Name of the exercise	
muscleGroup	VARCHAR(32)	Targeted muscle group	

Figure 8: Data dictionary for Exercise

2.2 Normalization and Functional Dependencies

To verify that the EER diagram is correctly constructed and free from anomalies (such as deletion or modification anomalies), we applied normalization. This process involved identifying the functional dependencies and ensuring the schema satisfies Third Normal Form (3NF).

2.2.1 Functional Dependencies

The functional dependencies are as follows:

User: `userID` → `fName, lName, weight, DOB, sex`
Health: (`userID`, `date`) → `heartrate, VO2max, HRvariation, sleeptime`
Workout: `workoutID` → `userID, startTime, endTime, maxHR, workoutType`
Run: (`workoutID`, `intervalNr`) → `distance, pace, incline`
Weightlift: (`workoutID`, `exerciseID`, `setNr`) → `reps, weight`
Goal: `goalID` → `userID, goalType, amount, metric, completed`;
also, (`userID`, `goalType`) → `goalID, amount, metric, completed`

2.2.2 First Normal Form (1NF)

To satisfy 1NF, we ensured that all attributes contain atomic values and that each table has a primary key. For the `Run` table, repeating interval data was separated into a dedicated `Intervals` table. The `Exercise` table was updated to include atomic fields such as `name` and `muscleGroup`.

2.2.3 Second Normal Form (2NF)

To achieve 2NF, all partial dependencies were removed so that every non-key attribute is fully dependent on the entire primary key. For example, the `Exercise` table was split into `Exercise` and `ExerciseType`, isolating attributes related to exercise classification.

2.2.4 Third Normal Form (3NF)

To satisfy 3NF, transitive dependencies were eliminated so that non-key attributes depend only on the primary key. In the `Goal` table, we introduced a separate `goalName` to avoid redundancy and maintain consistency across goal entries.

2.3 Conclusion

By applying normalization up to 3NF, we improved data integrity and reduced redundancy. The resulting schema is robust against common anomalies and aligns well with the conceptual EER model.