

ГЛАВА 5. ПРОЕКТИРОВАНИЕ ДАННЫХ

Данная глава посвящена проектированию структуры базы данных информационной системы, включающей объектную модель, логическую схему данных, определение связей, ограничений целостности, а также разработку вспомогательных механизмов — триггеров и хранимых процедур.

Целью проектирования схемы данных является формирование устойчивого, непротиворечивого и удобного для дальнейшей разработки фундамента, обеспечивающего корректное хранение, обработку и доступ к данным.

Объектная модель базы данных отражает ключевые хранимые сущности информационной системы учёта и проведения тренировок. В модели представлены укрупнённые объекты, каждый из которых соответствует логической группе данных в системе.

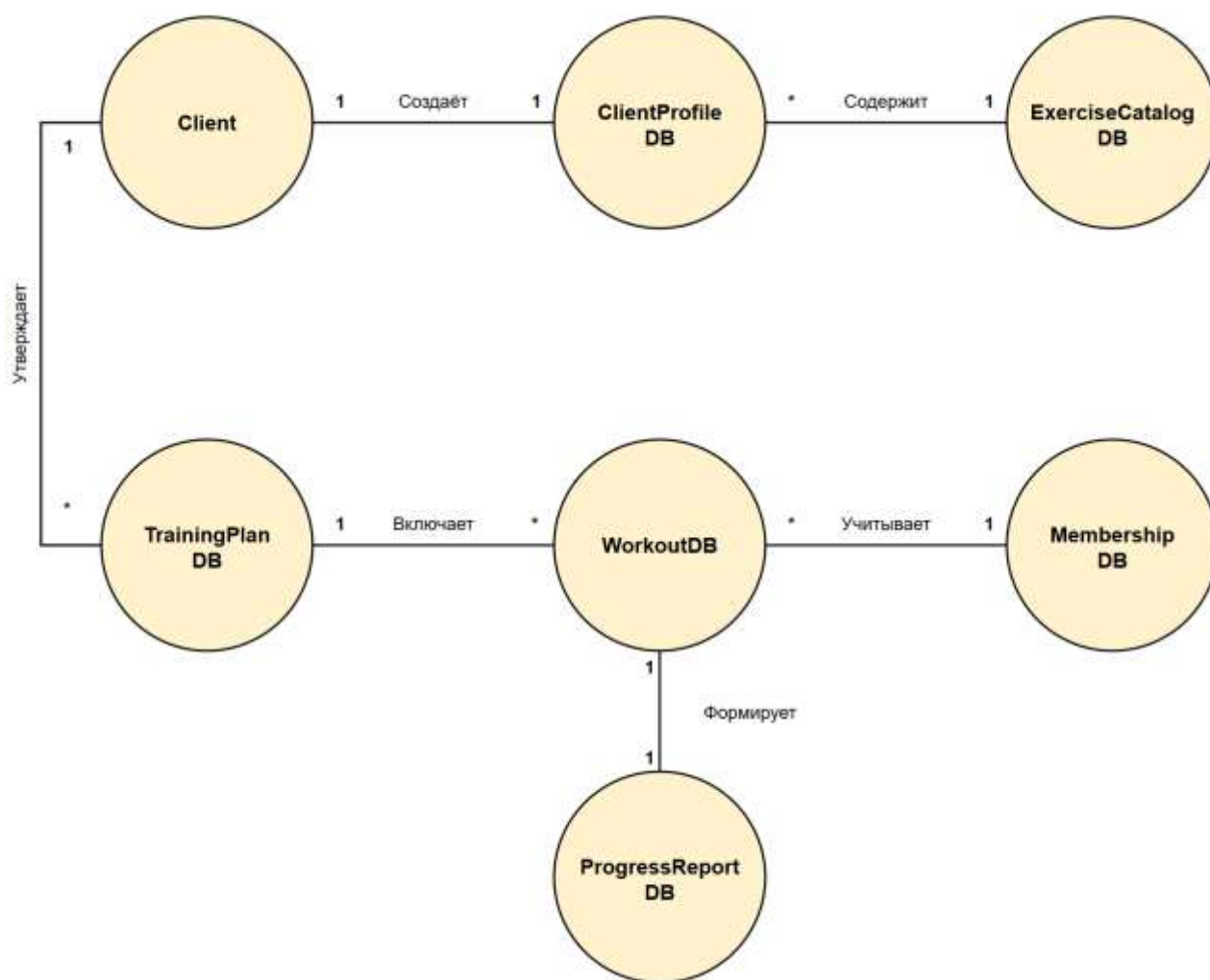


Рисунок 5.1 – Объектная модель базы данных

Сущность Client связана с профилем клиента (ClientProfileDB), так как клиент инициирует создание своего профиля для получения плана тренировок. Сущность ExerciseCatalogDB содержит множество упражнений, которые используются при формировании тренировочной программы. На основании выбранных упражнений формируется объект TrainingPlanDB, представляющий собой набор упражнений для конкретного клиента. Результаты выполненных тренировок сохраняются в объекте WorkoutDB, который связан с формированием отчёта о прогрессе (ProgressReportDB). Завершением процесса является учёт абонемента и посещений клиента, что также отражено в модели соответствующими связями между объектами.

Представленная объектная модель служит основой для последующего построения логической и физической моделей данных, включая ER-диаграмму и схему реляционных таблиц.

На основании объектной модели была разработана схема данных основной базы информационной системы фитнес-клуба SportFit. Схема данных представлена на рисунке 5.2.

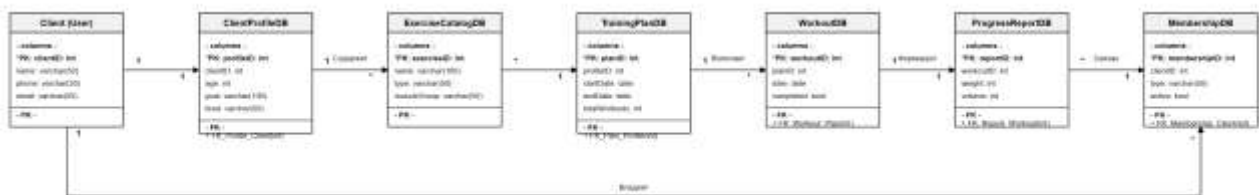


Рисунок 5.2 – Схема данных

В схеме данных участвуют следующие таблицы:

- Client (User) – содержит сведения о клиентах клуба: идентификатор клиента clientID, имя name, контактный телефон phone, адрес электронной почты email. Первичный ключ таблицы – PK_Client(int) по полю clientID.
- ClientProfileDB – хранит профили клиентов. Для каждого профиля сохраняются поля: profileID, clientID, возраст age, цель тренировок goal, уровень подготовки level, медицинские ограничения restrictions. Поле clientID является внешним ключом FK_Profile_Client(int), связывающим профиль с таблицей Client по схеме связи один клиент – один профиль (1:1).
- ExerciseCatalogDB – таблица каталога упражнений. Содержит идентификатор упражнения exerciseID, название name, тип type (кардио, силовое, функциональное), целевую группу мышц muscleGroup и рекомендуемую нагрузку load. Первичный ключ – PK_ExerciseCatalog(int) по полю exerciseID.

- TrainingProgramDB – хранит тренировочные программы, сформированные для конкретных профилей. Таблица включает поля: programID, profileID, название программы name, тип type, длительность duration. Поле profileID является внешним ключом FK_Program_Profile(int) и задаёт связь один профиль – несколько программ (1:*). Связь с таблицей ExerciseCatalogDB реализуется логически через множество идентификаторов упражнений, входящих в программу (exerciseID ∈ programExercises).

- TrainingPlanDB – содержит индивидуальные планы тренировок. Поля: planID, programID, дата начала startDate, дата окончания endDate, общее количество тренировок totalWorkouts, флаг согласования approved. Поле programID – внешний ключ FK_Plan_Program(int), задающий связь одна программа – несколько планов (1:*).

- WorkoutDB – таблица выполненных тренировок. Для каждой тренировки хранятся поля: workoutID, planID, дата проведения date, длительность duration, флаг выполнения completed. Поле planID – внешний ключ FK_Workout_Plan(int), задающий связь один план – несколько тренировок (1:*).

- ProgressReportDB – таблица отчётов о прогрессе. Поля: reportID, workoutID, период period, вес weight, объём volume, выносливость endurance. Поле workoutID – внешний ключ FK_Report_Workout(int), задающий связь одна тренировка – один отчёт (1:1).

- MembershipDB – таблица абонементов. Поля: membershipID, clientID, тип type, дата начала startDate, дата окончания endDate, флаг активности active. Поле clientID – внешний ключ FK_Membership_Client(int), задающий связь один клиент – несколько абонементов (1:*).

- VisitDB – таблица посещений. Поля: visitID, membershipID, дата date, время time, флаг подтверждения confirmed. Поле membershipID – внешний ключ FK_Visit_Membership(int), задающий связь один абонемент – несколько посещений (1:*).

Наличие внешних ключей и индексов (например, IX_Workout_Plan(int) для таблицы WorkoutDB) обеспечивает целостность данных и ускоряет выполнение запросов по основным бизнес-связям между сущностями. Такая схема данных отражает последовательность бизнес-процесса: клиент → профиль → программа → план → тренировки → отчёт о прогрессе; а также абонемент → посещения.

В процессе формирования схемы данных был разработан триггер, предотвращающий удаление строк из таблицы WorkoutDB. Это необходимо для того, чтобы уже выполненные тренировки (особенно те, по которым сформированы отчёты о прогрессе) нельзя было случайно удалить и нарушить историю тренировок клиента.

```
CREATE TRIGGER trg_PreventDeleteWorkout ON WorkoutDB
FOR DELETE
AS
BEGIN
    PRINT N'Удаление записей из таблицы WorkoutDB запрещено. ' +
        N'История тренировок должна сохраняться.';
    ROLLBACK TRAN;
END;
GO
```

Триггер срабатывает при попытке удалить одну или несколько строк из таблицы WorkoutDB, выводит служебное сообщение и откатывает транзакцию, тем самым обеспечивая логическую целостность данных.

Для удобства получения агрегированной информации была создана хранимая процедура, которая по идентификатору клиента возвращает основные сведения о его последнем плане: цель, уровень подготовки, дату начала, количество тренировок и ФИО тренера.

```
CREATE PROC prClientPlanInfo @clientID INT
AS
SELECT
    cp.goal AS Goal,
    cp.level AS Level,
    tp.startDate AS StartDate,
    tp.totalWorkouts AS TotalWorkouts,
    t.name AS TrainerName
FROM Client c
JOIN ClientProfileDB cp ON c.clientID = cp.clientID
JOIN TrainingProgramDB tpr ON cp.profileID = tpr.profileID
JOIN TrainingPlanDB tp ON tpr.programID = tp.programID
JOIN Trainer t ON tpr.trainerID = t.trainerID
```

```
WHERE c.clientID = @clientID  
ORDER BY tp.startDate DESC;  
GO
```

Данная процедура инкапсулирует типовой запрос, используемый тренерами при подготовке отчётов и просмотре текущего плана клиента. Использование хранимой процедуры позволяет централизовать бизнес-логику доступа к данным и упростить работу прикладного кода.