Контрольная работа 3

Логическое проектирование реляционной БД. Часть 1.

Уважаемый, студент!
Вам необходимо:

1. Провести преобразование ER–диаграммы в схему базы данных
2. Составить реляционные отношения

На этапе логического проектирования – ER-диаграмма формальным способом преобразуется в схему реляционной базы данных (РБД). На основании схемы РБД и описания сущностей ПрО составляются отношения (таблицы) базы данных. Потом выполняется нормализация отношений (контрольная работа 4). Это необходимо сделать для того, чтобы исключить нарушения логической целостности данных и повысить таким образом надёжность и достоверность данных.

В результате всех этих операций создаётся концептуальная схема БД – основной документ для базы данных.

Пример выполнения заданий (продолжение той же БД, что и в КР2)

1. Преобразование ER–диаграммы в схему базы данных

База данных создаётся на основании схемы базы данных. Для преобразования ER–диаграммы в схему БД приведём уточнённую ER–диаграмму, содержащую атрибуты сущностей (рис. 1).

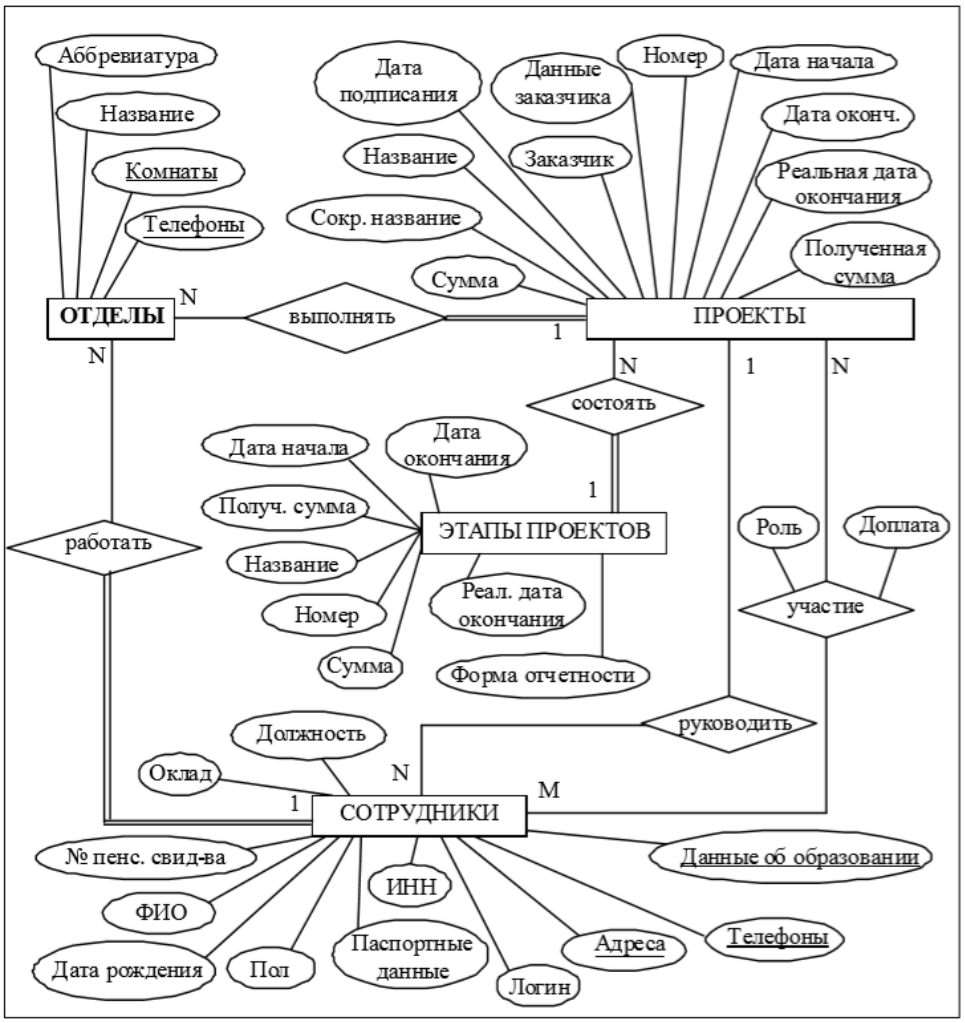

Рис. 1. Уточнённая ER–диаграмма проектной организации

Примечание. Многозначные атрибуты на рисунке выделены подчеркиванием.

Преобразование ER–диаграммы в схему БД выполняется путем сопоставления каждой сущности и каждой связи, имеющей атрибуты, отношения (таблицы) БД. Связь типа 1:n (один-ко-многим) между отношениями реализуется через внешний ключ. Ключ вводится для того отношения, к которому осуществляется множественная связь. Внешнему ключу должен соответствовать первичный или уникальный ключ основного (родительского) отношения.

Связь *участвовать* между *ПРОЕКТАМИ* и *СОТРУДНИКАМИ* принадлежит к типу n:m (многие-ко-многим). Этот тип связи реализуется через вспомогательное отношение *Участие*, которое содержит комбинации первичных ключей соответствующих исходных отношений.

Для схемы БД будем использовать обозначения, представленные на рис. 2.

Рис. 2. Обозначения, используемые на схеме базы данных

Полученная схема реляционной базы данных (РБД) приведена на рис. 3.

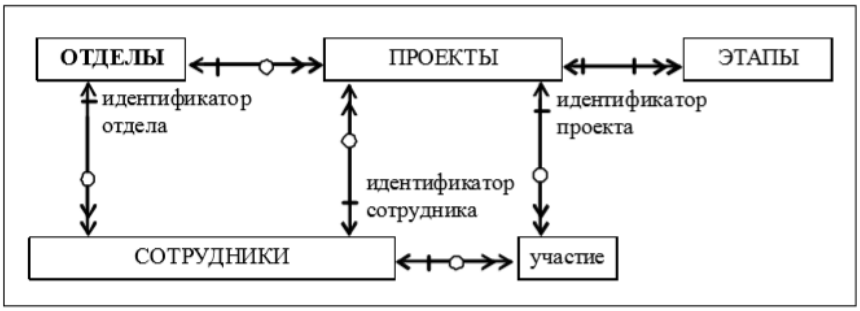

Рис. 3. Схема РБД, полученная из ER–диаграммы проектной организации

Бинарная связь между отношениями не может быть обязательной для обоих отношений. Такой тип связи означает, что, например, прежде чем добавить новый проект в отношение *ПРОЕКТЫ*, нужно добавить новую строку в отношение *ЭТАПЫ*, и наоборот. Поэтому для такой связи необходимо снять с одной стороны условие обязательности. Так как все эти связи будут реализованы с помощью внешнего ключа, снимем условие обязательности связей для отношений, содержащих первичные ключи.

Схема на рисунке 3 содержит три цикла:

- «сотрудники–проекты–участие–сотрудники»;
- «отделы–сотрудники–проекты–отделы»;
- «отделы–сотрудники–участие–проекты–отделы».

Цикл допустим только в том случае, если связи, входящие в него, независимы друг от друга. Например, для нашей ПрО справедливо такое правило: сотрудник любого отдела может быть участником (исполнителем или консультантом) проекта любого отдела. Эти связи независимы, поэтому цикл «отделы–сотрудники–участие–проекты–отделы» не будет приводить к нарушению логической целостности данных.

С другой стороны, только сотрудник отдела, отвечающего за выполнение проекта, может быть руководителем проекта. Но система не помешает нам назначить руководителем проекта сотрудника любого отдела. При добавлении проекта с внешним ключом *Руководитель* система проверит только, что такой человек есть в таблице *СОТРУДНИКИ*. А значение внешних ключей *Отдел* в таблицах *СОТРУДНИКИ* и *ПРОЕКТЫ* сравнивать не будет.

Таким образом, остальные циклы могут приводить к возможности нарушения логической целостности данных. Существует несколько подходов для разрешения ситуаций, в которых связи, входящие в цикл, зависят друг от друга.

Рассмотрим эту ситуацию в общем случае. Сначала слегка упростим схему: реализуем связь «руководить» через таблицу *УЧАСТИЕ* – это позволит не отвлекаться на малозначительные детали.

Будем считать, что в выполнении проекта могут участвовать только сотрудники, работающие в том же отделе, к которому относится проект (рис. 4,а). При циклической схеме СУБД не сможет гарантировать логическую целостность данных без использования дополнительных средств.

Один из способов разрешения таких ситуаций – разорвать цикл, исключив одну из связей (рис. 4,б) или введя промежуточное отношение (рис. 4,в). В нашем случае можно было бы разорвать связь «сотрудники–проекты», если бы каждый сотрудник участвовал во всех проектах своего отдела. Промежуточное отношение можно было бы использовать, если бы существовала общая связь между сущностями, входящими в цикл. Например, если бы каждый сотрудник заключал договор с отделом на выполнение работ в рамках проекта, то отношение *ДОГОВОРЫ* отражало бы связь между отделом, сотрудником и проектом.

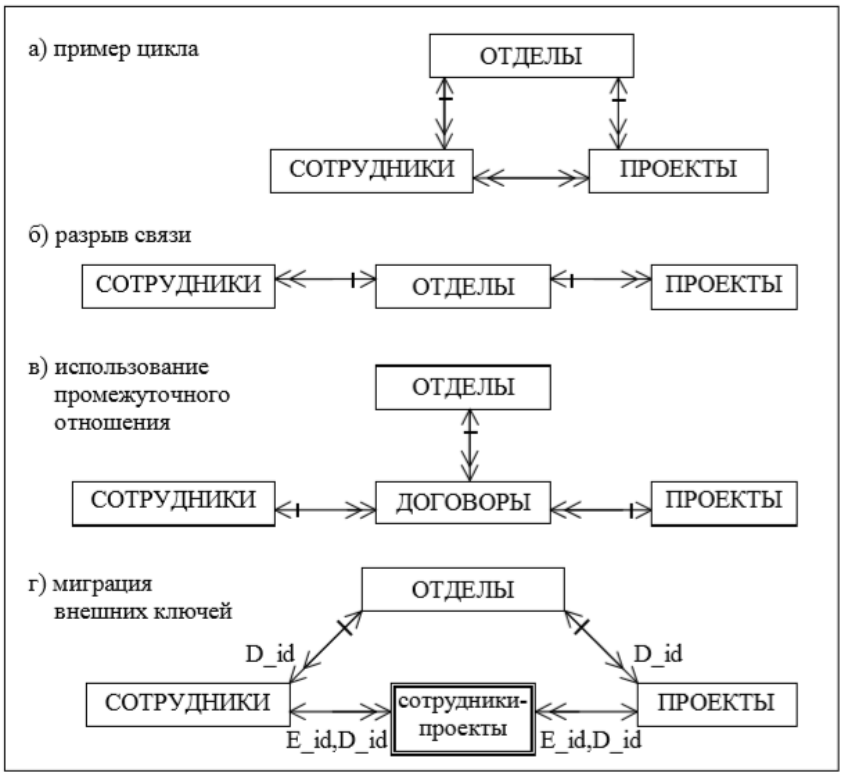

Рис.4. Некоторые способы разрешения циклов в схеме базы данных

Другой способ разрешения цикла заключается в том, что в промежуточное отношение *СОТРУДНИКИ – ПРОЕКТЫ*, которое реализует связь многие-комногим, добавляются (мигрируют) внешние ключи *Код отдела (D_id)* из отношений *СОТРУДНИКИ* и *ПРОЕКТЫ* (рис. 4,г). Эти ключи проверяются на равенство друг другу с помощью соответствующего ограничения целостности (check). Использование этого способа

возможно в том случае, когда соответствующие связи (*отдел–проект* и *отдел–сотрудник*) имеют тип один-ко-многим и являются обязательными.

В тех ситуациях, когда все эти способы непригодны, логическая целостность контролируется программно или вручную. Если принято решение переложить обязанности по контролю за логической целостностью данных на пользователя, то эти обязанности должны быть отражены в документации (в руководстве пользователя).

Примем для нашей ПрО, что руководитель проекта может одновременно выполнять и другие обязанности в этом проекте, чтобы цикл «сотрудники– проекты–участие–сотрудники» не приводил к возможности нарушения логической целостности данных. Зато цикл «отделы–сотрудники (руководители)–проекты–отделы» включает зависимые связи: руководитель проекта назначается из того отдела, который отвечает за выполнение проекта в целом. Здесь можно было бы применить разрыв связи «отделы–проекты» и определять, к какому отделу относится проект через руководителя (по отделу руководителя проекта). Но такой подход в данном случае имеет существенный недостаток. Заменив руководителя проекта сотрудником другого отдела, можно одновременно изменить отдел, отвечающий за выполнение проекта, т.е. объединить в одно действие два независимых изменения, а это недопустимо.

Исходя из вышесказанного, мы не будем разрывать связь, а примем решение реализовать эту проверку программно. Приложение должно будет при назначении руководителя проекта выдавать список сотрудников того отдела, который отвечает за выполнение данного проекта. Руководителя можно будет выбрать только из этого списка, а не вводить вручную.

2. Составление реляционных отношений

Каждое реляционное отношение соответствует одной сущности (объекту ПрО) и в него вносятся все атрибуты этой сущности. Для каждого отношения определяются первичный ключ и внешние ключи (в соответствии со схемой БД). В том случае, если базовое отношение не имеет потенциальных ключей, вводится *суррогатный первичный ключ*, который не несёт смысловой нагрузки и служит только для идентификации записей.

Отношения приведены в табл. 1-5. Для каждого отношения указаны атрибуты с их внутренним названием, типом и длиной. Типы данных обозначаются так: N – числовой, C – символьный тип фиксированной длины, V – символьный тип переменной длины, D – дата (этот тип имеет стандартную длину, зависящую от СУБД, поэтому она не указывается).

Потенциальными ключами отношения *ОТДЕЛЫ* являются атрибуты *Аббревиатура* и *Название отдела*. Первый занимает меньше места, поэтому мы выбираем его в качестве первичного ключа.

Таблица 1. Схема отношения *ОТДЕЛЫ* (Departs)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Аббревиатура отдела	D_ID	C(10)	**первичный ключ**
Название отдела	D_NAME	V(100)	обязательное поле
Комнаты	D_ROOMS	V(20)	обязательное многозначное поле
Телефоны	D_PHONE	V(40)	обязательное многозначное поле

Потенциальными ключами отношения *СОТРУДНИКИ* являются поля *Паспортные данные*, *ИНН* и *Номер страхового пенсионного свидетельства*. Все они занимают достаточно много места, а паспортные данные кроме того могут меняться. Введём суррогатный первичный ключ *Номер сотрудника*.

Таблица 2. Схема отношения *СОТРУДНИКИ* (Employees)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер	E_ID	N(4)	**суррогатный первичный ключ**
Фамилия, имя, отчество	E_NAME	V(50)	обязательное поле
Дата рождения	E_BORN	D	обязательное поле
Пол	E_SEX	C(1)	обязательное поле, 'м' или 'ж'
Паспортные данные	E_PASP	V(50)	обязательное поле
ИНН	E_INN	C(12)	обязательное поле
Номер пенсионного страхового свидетельства	E_PENS	C(14)	обязательное уникальное поле
Отдел	E_DEPART	C(10)	обязательное уникальное поле
Должность	E_POST	V(30)	обязательное поле
Оклад	E_SAL	N(8,2)	обязательное поле, > 12130 руб.
Данные об образовании	E_EDU	V(200)	обязательное многозначное поле
Адреса	E_ADDR	V(100)	многозначное поле
Телефоны	E_PHONE	V(30)	многозначное поле
Логин	E_LOGIN	V(30)	

Примечание. Суррогатный первичный ключ также может вводиться в тех случаях, когда потенциальный ключ имеет большой размер (например, длинная символьная строка) или является составным (не менее трёх атрибутов).

В отношении *ПРОЕКТЫ* три потенциальных ключа: *Номер проекта*, *Название проекта* и *Сокращённое название*. Меньше места занимает первый из них, но он малоинформативен. Зато сокращённое название, используемое в качестве внешнего ключа в других таблицах, позволит специалисту идентифицировать проект без необходимости соединения с отношением *ПРОЕКТЫ*.

Таблица 3. Схема отношения *ПРОЕКТЫ* (Projects)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер проекта	P_ID	N(6)	обязательное уникальное поле
Название проекта	P_TITLE	V(100)	обязательное поле
Сокращённое название	P_ABBR	C(10)	**первичный ключ**
Отдел	P_DEPART	C(10)	внешний ключ (к Departs)
Заказчик	P_COMPANY	V(40)	обязательное поле
Данные заказчика	P_LINKS	V(200)	обязательное поле
Руководитель	P_CHIEF	N(4)	внешний ключ (к Employees)
Дата начала проекта	P_BEGIN	D	обязательное поле
Дата окончания проекта	P_END	D	обязательное поле, больше даты начала проекта
Реальная дата окончания	P_FINISH	D	
Стоимость проекта	P_COST	N(10)	обязательное поле
Полученная сумма	P_SUM	N(10)	обязательное поле, значение по умолчанию – 0

Потенциальным ключом отношения *ЭТАПЫ* является комбинация внешнего ключа и номера этапа, а потенциальным ключом вспомогательного отношения *УЧАСТИЕ* является комбинация первых трёх полей этого отношения. Можно вообще не вводить первичный ключ для данных отношений, т.к. на них никто не ссылается. Но уникальность этих комбинации является в данном случае ограничением целостности данных, поэтому мы возьмём эти комбинации в качестве первичных ключей соответствующих отношений.

Таблица 4. Схема отношения *ЭТАПЫ ПРОЕКТА* (Stages)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*	
Проект	S_PRO	C(10)	внешний ключ (к Projects)	**составной первичный ключ**
Номер этапа	S_NUM	N(2)		
Название этапа	S_TITLE	V(200)	обязательное поле	
Дата начала этапа	S_BEGIN	D	обязательное поле	
Дата окончания этапа	S_END	D	обязательное поле, > даты начала	
Реальная дата окончания	S_FINISH	D	больше даты начала этапа	
Стоимость этапа	S_COST	N(10)	обязательное поле	
Полученная сумма по этапу	S_SUM	N(10)	обязательное поле, значение по умолчанию – 0	
Форма отчётности	S_FORM	V(100)	обязательное поле	

Таблица 5. Схема отношения УЧАСТИЕ (Job)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания**
Проект	J_PRO	C(10)	внешний ключ (к Projects)
Сотрудник	J_EMP	N(4)	внешний ключ (к Employees)
Роль	J_ROLE	V(20)	обязательное поле
Доплата	J_BONUS	N(2)	

* – в отношении *УЧАСТИЕ* первичный ключ состоит из первых 3-х полей этого отношения.

Требования к оформлению:

Обязательно наличие титульного листа, на котором студент указывает выбранную тему, номер группы, Ф.И.О.

Параметры страницы: верхнее – 20 мм; нижнее – 20 мм; левое – 30 мм; правое – 10 мм.

Шрифт Times New Roman, размер шрифта- 14 (текст в таблицах-12), межстрочный интервал – 1,5 строки, абзацный отступ-1,25. Подчеркивание слов и выделение их курсивом не допускается.