Модуль № 1: Проектирование и разработка информационных систем

Текст задания: На основании описания брифинга и документов, представленных заказчиком, необходимо спроектировать ER-диаграмму для информационной системы.

Обязательна 3 нормальная форма с обеспечением ссылочной целостности.

При разработке диаграммы обратите внимание на согласованную осмысленную схему именования, создайте необходимые первичные и внешние ключи.

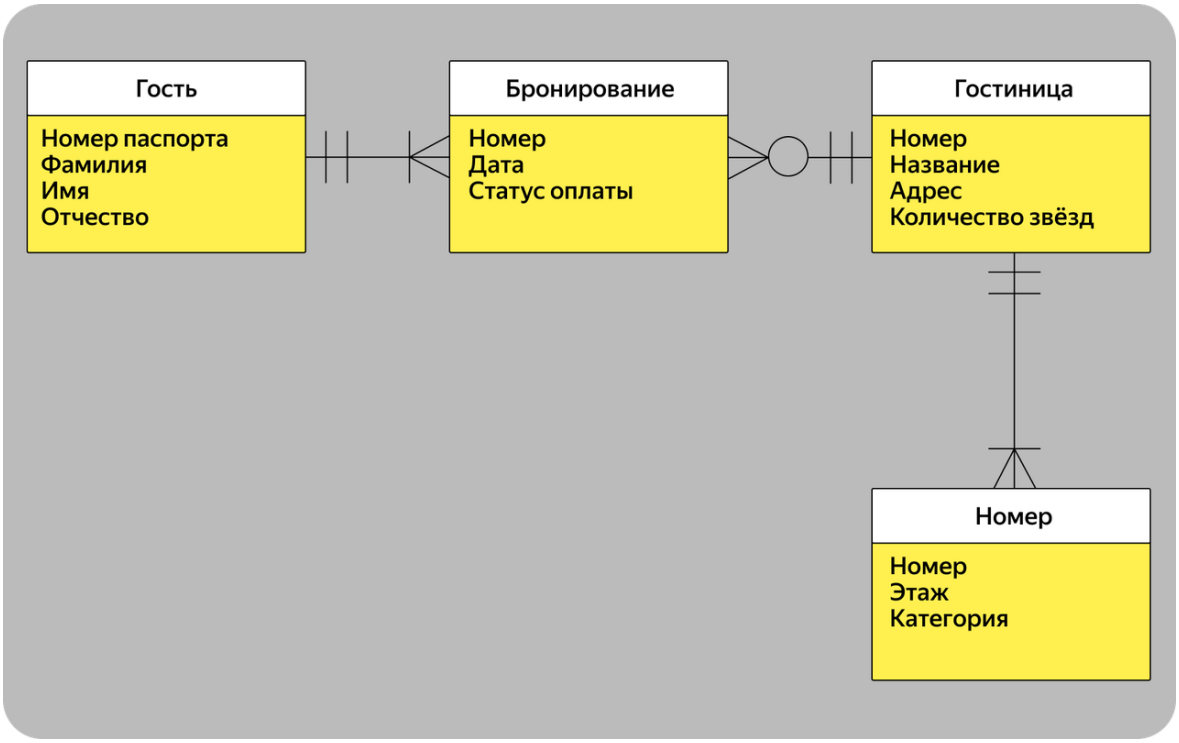
ER - диаграмма должна быть представлена в формате .pdf и содержать таблицы, связи между ними, атрибуты и ключи (типами данных на данном этапе можно пренебречь).

Необходимые приложения: Текст брифинга.pdf, Документы заказчика.zip

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

ER-диаграмма (схема «сущность-связь») - это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы.

Что такое ER-диаграмма



Системный аналитик начинает работу над новым проектом с **изучения его предметной области и терминов (сущностей),** которые в ней используют.

Например, нужно создать систему для бронирования билетов на самолёт. *Аэропорт, авиакомпания, дата, рейс, пассажир, пункты прибытия и назначения, багаж* - термины проекта. Их ещё называют понятиями или **сущностями**.

В системе сущность представлена в виде **экземпляров.** Например, *экземпляры сущности «Аэропорт» - аэропорты «Домодедово», «Пулково», «Воронеж».*

У сущностей есть **атрибуты** - характеристики, которые их описывают. Например, *атрибутами сущности «Аэропорт» будут код, адрес, номер телефона.* Атрибуты есть у каждого экземпляра сущности, но у них разные значения. У аэропортов «Домодедово» и «Воронеж» есть одинаковый атрибут «Адрес», но у каждого из них разное значение этого атрибута.

Собрав все сущности будущего проекта, системный аналитик выясняет, как они **связаны между собой**, и составляет ER-модель (сокр. от entity–relationship модель или **модель «сущность-связь»**).

В модели есть три типа связей:

«Один-к-одному» - один экземпляр сущности связан только с одним экземпляром другой сущности. Например, пассажир рейса и его место в самолете.

«Один-ко-многим» - один экземпляр сущности связан со множеством экземпляров другой сущности. Например, у одного пассажира может быть несколько единиц багажа, при этом каждая единица багажа может быть связана только с одним пассажиром.

«**Многие-ко-многим**» - множество экземпляров одной сущности связаны со множеством экземпляров другой сущности. Например, аэропорт обслуживает несколько авиакомпаний. При этом каждая авиакомпания может обслуживаться в нескольких аэропортах.

Системный аналитик создаёт ER-диаграмму модели данных. Это схема, которая показывает, с какими данными нужно будет работать для реализации проекта и как эти данные связаны между собой. Например, ER-диаграмма проиллюстрирует, что багаж связан с номером рейса, но не связан со временем окончания посадки пассажиров на него.

Нормализация данных

Данные в базе могут быть в любом виде: числа, проценты, текст.

Нормализация - это способ организации данных. В нормализованной базе нет повторяющихся данных, с ней проще работать и можно менять её структуру для разных задач.

В процессе нормализации данные преобразуют, чтобы они занимали меньше места, а поиск по элементам был быстрым и результативным. Для этого создают дополнительные таблицы и связывают друг с другом ключами - колонками, в которых нет повторяющихся элементов.

Рассмотрим на **примере** суть нормирования данных.

Александр Сушков каждый день ходит в магазин продуктов рядом с домом. Покупает продукты, например, хлеб и творог, и оплачивает их картой. Данные о покупках, за которые рассчитывались картой, ежедневно сохраняются на сервере магазина в таблицу Google Sheets.

Товар	Стоимость	Покупатель
Хлеб	75,00	Александр Сушков
Творог	119,99	Александр Сушков

К концу дня данные можно проанализировать: посчитать выручку магазина, какие категории товаров продавались лучше и сколько денег принесли. Когда количество записей в таблице превысит миллион - работать с данными станет сложнее. Таблица будет медленно открываться, в ней будет труднее найти ошибки и собрать статистику по определённым товарам или категориям покупателей.

Кроме Александра Сушкова, в магазине ежедневно делают покупки и другие люди, которые тоже живут поблизости - Иван Иванов и Егор Кузнецов. Фамилии и имена покупателей повторяются в каждой записи их покупок. А это в среднем 10–20 символов. Чтобы не дублировать эти данные, можно создать отдельную таблицу.

Покупатель	Ключ
Александр Сушков	1
Егор Кузнецов	2
Иван Иванов	3

Товар	Стоимость	Покупатель
Хлеб	75,00	1
Творог	119,99	1
Яблоки	85,00	3
Рис	59,99	2

В отдельной таблице каждому покупателю можно присвоить номер — ключ. Тогда в основной таблице вместо имён будут ключи, которые свяжут обе таблицы. Это сократит количество символов в основной таблице, уберёт повторяющиеся сущности и упростит поиск

Имена и фамилии сотни постоянных покупателей - это в среднем 1 500 символов. Если в год каждый из них посещает магазин 300 раз, то получается 450 000 символов. Только замена имён и фамилий на цифры сократит количество символов в записях в 3–4 раза.

С товарами можно поступить так же: ввести категории, например, хлебобулочные изделия или молочная продукция, и создать для них отдельную таблицу. Так работает нормализация, цель которой - оптимизировать работу с базами данных.

Зачем нормализовать данные в БД

Работа с нормализованными данными отдалённо напоминает поход за продуктами с книгой рецептов. Чтобы купить продукты для салата оливье, нужно открыть первую таблицу - книгу рецептов, перейти в подтаблицу - раздел «Салаты», найти нужный рецепт и положить в корзину продукты из него. Примерно так работают связи между таблицами данных.

Без нормализации данных книга рецептов была бы одной большой таблицей с ингредиентами блюд. Чтобы купить продукты для оливье, пришлось бы сначала собрать в корзину все ингредиенты из книги. Потом найти те, что нужны для салата, а остальные выложить из корзины.

Что даёт нормализация данных и как она упрощает работу с базами:

1. Уменьшает объём базы данных и экономит место.

За счёт отдельных таблиц для категорий и повторяющихся элементов можно уменьшить размер записей в базе данных, а значит, и её вес.

*2. Упрощает поиск и делает работу с базой удобне*е.

Нормализованную базу данных, которая состоит из связанных таблиц, можно оптимизировать для задач без дополнительных действий. Например, для поиска по заданной категории не придётся искать и перебирать уникальные элементы в базе. Для этого можно обратиться к отдельной таблице с категориями и быстрее найти нужные данные.

3. Уменьшает вероятность ошибок и аномалий.

Нормальные формы данных в таблицах взаимосвязаны. Например, если нужно изменить или удалить данные в одной таблице, то остальные связанные с ней данные автоматически обновятся. Не придётся перебирать все записи в поисках полей, которые нужно изменить или удалить, а значит, не будет ошибок, когда в базу внесут изменения.

Правила нормализации баз данных

По правилам нормализации есть семь нормальных форм баз данных:

первая,
вторая,
третья,
нормальная форма Бойса-Кодда,
четвёртая,
пятая,
шестая.

Приводить данные к нормальным формам **можно только последовательно**. То есть в базе данных второй нормальной формы данные по умолчанию уже должны быть нормализованы по правилам первой нормальной формы и так далее. В итоге база данных в шестой нормальной форме - идеально нормализованная.

В некоторых случаях попытка нормализовать данные до «идеального» состояния может привести к созданию множества таблиц, ключей и связей. Это усложнит работу с базой и снизит производительность СУБД. Поэтому **обычно данные нормализуют до третьей нормальной формы**.

Разберём на примере.

Первая нормальная форма

В базе данных не должно быть дубликатов и составных данных.

Покупатель
Александр Александрович Сушков
Егор Сергеевич Кузнецов
Иван Иванович Иванов
Сергей Александрович Петров

Имя	Отчество	Фамилия
Александр	Александрович	Сушков
Егор	Сергеевич	Кузнецов
Иван	Иванович	Иванов
Сергей	Александрович	Петров

Слева данные о фамилии, имени и отчестве покупателей записаны в одно поле. Справа эти данные приведены к первой нормальной форме — каждый элемент записан в отдельное поле

Элементы составных данных лучше разнести по разным полям, иначе в процессе работы с данными могут появиться ошибки и аномалии.

Например, отдел маркетинга решил поздравить всех Александров с именинами и сделать рассылку с промокодом. Если таблица соответствует первой нормальной форме, можно найти нужные данные без дополнительных действий. Когда имя, отчество и фамилия записаны в одно поле, при поиске и сортировке в выборку попадут, например, Александровичи, Александровны и Александровы.

Другой пример - адреса. Их тоже лучше приводить к первой нормальной форме. То есть город, район, улицу, номер дом и номер квартиры записывать в отдельные поля.

Если какие-то данные дублируются, как в случае с именами и фамилиями постоянных покупателей, их нужно перенести в другую таблицу.

Вторая нормальная форма

Если упростить: **у каждой записи в базе данных должен быть первичный ключ**.

Первичный ключ - это элемент записи, который не повторяется в других записях.

Допустим, 10 декабря покупатель Егор Кузнецов купил цельнозерновой хлеб за 75 рублей в сетевом магазине продуктов города Москвы. Запись о его покупке появилась в базе данных. Нельзя исключать, что другой Егор Кузнецов в этот день купит такой же товар в другом магазине сети. Запись о покупке тоже появится в базе.

Номер чека	Имя	Фамилия	Товар	Стоимость
1283	Егор	Кузнецов	Цельнозерновой хлеб	75,00
4569	Егор	Кузнецов	Цельнозерновой хлеб	75,00

Чтобы записи не перепутались, можно добавить к ним идентификатор покупки, например номер чека. Идентификатор покупки — это первичный ключ

Третья нормальная форма

В записи не должно быть столбцов с неключевыми значениями, которые зависят от других неключевых значений.

Данные о руководителях отделов нужно вынести в другую таблицу. Тогда в основной таблице не будет транзитивных связей, и она будет соответствовать третьей нормальной форме.

Данные не во всех случаях преобразовывают до третьей формы. Иногда для этого придётся создать много небольших таблиц, что усложнит базу и задачу разработчикам

Личный № сотрудника	Должность	Отдел	Руководитель отдела
120	Программист	Отдел разработки	Иванов И.И.
121	Инженер данных	Отдел аналитики	Кузнецов Е.А.

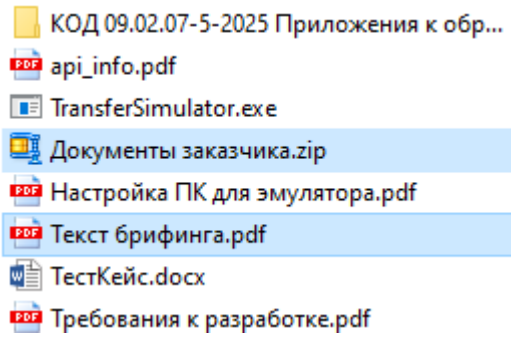
Личный номер сотрудника — это первичный ключ. Данные во втором и третьем столбце напрямую зависят от первичного ключа. Но между личным номером сотрудника и руководителем отдела только косвенная или транзитивная связь. Её в базе данных третьей нормальной формы быть не должно

Проектирование ER-диаграммы – **повторение, работа по образцу**
https://nationalteam.worldskills.ru/skills/proektirovanie-er-diagrammy/

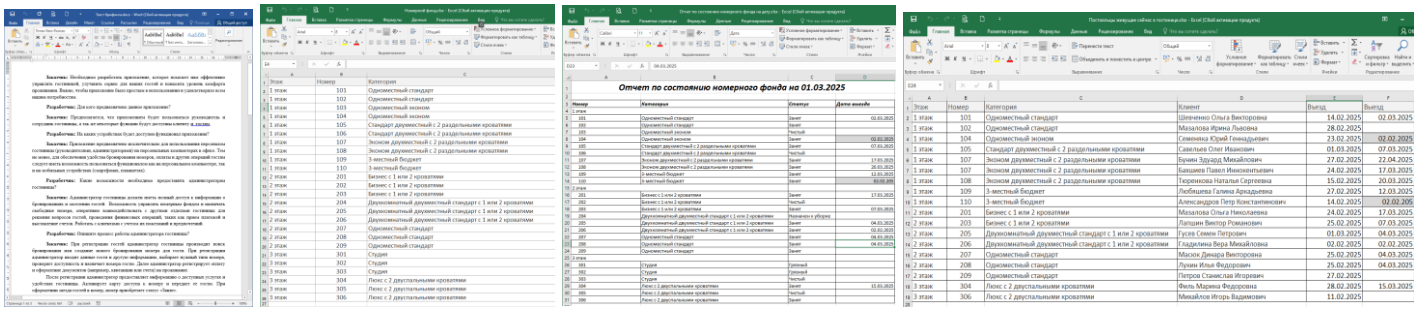
Выполнение задания

На основании описания брифинга и документов, представленных заказчиком, необходимо спроектировать ER-диаграмму для информационной системы.

Необходимые приложения: Текст брифинга.pdf, Документы заказчика.zip



Для определения структуры диаграммы нам предоставлены 4 документа



Выделим основные сущности и атрибуты

Три основные роли: Руководитель, администратор и клиент

Сформируем **таблицы Пользователи и Роли**

Таблица Роли

Поле	*Тип данных*	
ID	числовой	Первичный ключ
Роль	текстовый	

Таблица Пользователи

Поле	*Тип данных*	
ID	числовой	Первичный ключ
Фамилия	текстовый	
Имя	текстовый	
Отчество	текстовый	
Роль	числовой	

Сформируем таблицы для управления номерным фондом

Таблица Категория номера

Поле	*Тип данных*	
ID	числовой	Первичный ключ
Наименование категории	текстовый	

Таблица Номерной фонд

Поле	*Тип данных*	
Номер	числовой	Первичный ключ
Категория	числовой	
Этаж	числовой	
Статус	числовой	

Таблица Этажи

Поле	*Тип данных*	
ID	числовой	Первичный ключ
Наименование этажа	текстовый	

Таблица Статус номера

Поле	*Тип данных*	
ID	числовой	Первичный ключ
Наименование статуса	текстовый	

Сформируем таблицу сведений о постояльцах

Таблица Постояльцы

Поле	*Тип данных*	
ID	числовой	Первичный ключ
Номер	числовой	
Клиент	числовой	
Дата въезда	дата	
Дата выезда	дата	

Таблица Заказы

Поле	*Тип данных*	
ID	числовой	Первичный ключ
Номер	числовой	
Клиент	числовой	
Количество дней	числовой	
Стоимость	вычисляемое поле