

# **Тема 5. Проектирование баз данных. Нормализация.**

# Цель занятия:

**Ознакомиться с концепциями работы с несколькими таблицами в базах данных, с видами связей между таблицами, понятиями первичных и внешних ключей, а также операциями объединения таблиц.**

# **Учебные вопросы:**

- 1. Введение в проектирование баз данных.**
- 2. Этапы проектирования базы данных.**
- 3. ER-диаграммы.**
- 4. Оптимизация структуры базы данных.**
- 5. Нормализация данных.**
- 6. Нормальные формы.**
- 7. Процесс нормализации.**

# Введение в проектирование БД.

## Задачи проектирование баз данных:

- **Эффективное хранение данных.** Проектирование помогает структурировать данные таким образом, чтобы минимизировать их дублирование и избыточность. Это позволяет сократить объемы хранимых данных и избежать ненужных повторений, что снижает затраты на хранение.
- **Обеспечение целостности данных.** В процессе проектирования определяются правила и ограничения для поддержания корректности данных (например, внешние ключи, уникальные индексы). Это гарантирует, что в базе данных не будут присутствовать некорректные или противоречивые данные.
- **Обеспечение производительности.** Правильно спроектированная база данных позволяет ускорить выполнение запросов за счет оптимальной организации таблиц, индексов, а также грамотного распределения данных между ними. Это особенно важно при работе с большими объемами данных и сложными запросами.

- **Масштабируемость и поддержка роста данных.**  
Проектирование учитывает возможное увеличение объема данных и нагрузки на базу данных, что позволяет предусмотреть механизмы масштабирования и обеспечить стабильную работу системы при росте данных.
- **Удобство в администрировании и поддержке.**  
Хорошо спроектированная база данных упрощает её дальнейшее сопровождение, добавление новых функций и поддержание в актуальном состоянии.

Т.о., проектирование базы данных — это критический этап, который позволяет создать надежную, эффективную и устойчивую систему для управления данными.

## 2. Этапы проектирования базы данных.

Этапы проектирования базы данных:

1. Анализ требований
2. Проектирование концептуальной модели
3. Проектирование логической модели
4. Физическое проектирование

## 1. Анализ требований:

**Цель:** понять бизнес-процессы, для которых создается база данных, и определить, какие данные необходимо хранить и обрабатывать.

**Действия:**

- Общение с заказчиками и пользователями для сбора информации о том, какие задачи должна решать система.
- Определение объема данных, частоты их обновления и основных операций (чтение, запись, обновление).
- Формулирование целей системы и ключевых требований, например, объем данных, скорость обработки запросов, безопасность и масштабируемость.

**Результат:** четкое понимание, как база данных будет поддерживать бизнес-процессы, какие данные будут обрабатываться и какие требования к ней предъявляются.

## 2. Проектирование концептуальной модели:

**Цель:** создание наглядной модели данных, которая отражает ключевые сущности и их связи без учета специфики СУБД.

**Действия:**

- Определение сущностей (например, "Клиент", "Заказ", "Продукт"), атрибутов (свойства сущностей, такие как имя клиента, дата заказа) и связей между сущностями.
- Создание ER-диаграммы для отображения всех сущностей, их атрибутов и связей между ними.
- Определение типов связей (один к одному, один ко многим, многие ко многим).

**Результат:** ER-диаграмма, которая является концептуальной моделью базы данных, предоставляющей общее представление о структуре данных.



### 3. Проектирование логической модели:

**Цель:** преобразование концептуальной модели в логическую модель, которая уже учитывает требования нормализации и особенности реляционных баз данных.

#### **Действия:**

- Преобразование сущностей из ER-диаграммы в таблицы, где каждая сущность становится таблицей, а атрибуты — полями таблицы.
- Применение нормализации для устранения избыточности данных (приведение к 1NF, 2NF, 3NF, при необходимости — BCNF).
- Определение первичных и внешних ключей для таблиц.

**Результат:** логическая модель базы данных, включающая структуры таблиц, ключи и связи между таблицами.

## 4. Физическое проектирование:

**Цель:** реализация логической модели на уровне конкретной СУБД с учетом производительности, безопасности и будущих изменений.

### Действия:

- Выбор конкретной СУБД (например, MySQL, PostgreSQL, SQL Server), учитывая требования системы (масштабируемость, стоимость, доступные функции).
- Проектирование таблиц, включая выбор типов данных для каждого поля, с учетом их объема и типа информации (например, INT для чисел, VARCHAR для текста).
- Создание индексов для ускорения поиска и выполнения сложных запросов.
- Определение стратегии управления транзакциями и обеспечения безопасности данных.

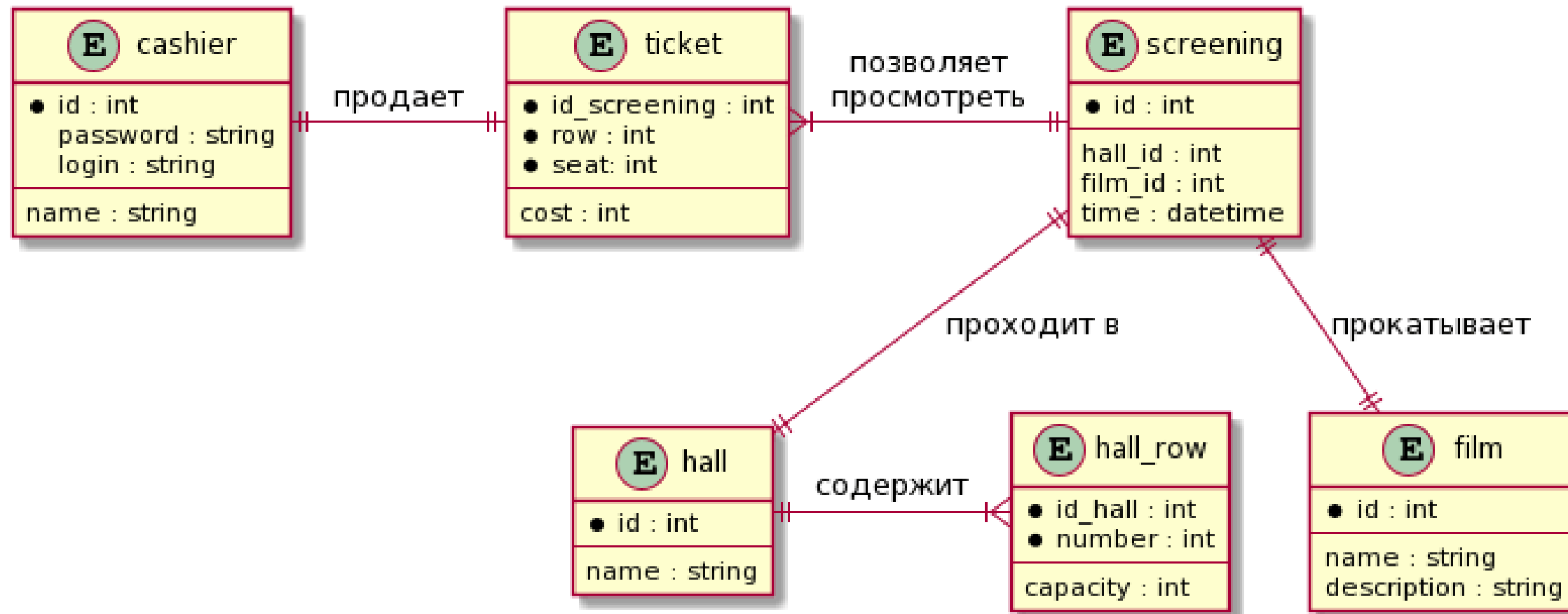
**Результат:** готовая к реализации физическая структура базы данных, которая оптимизирована для выбранной СУБД и учитывает требования по производительности и надежности.

# 3. ER-диаграммы.

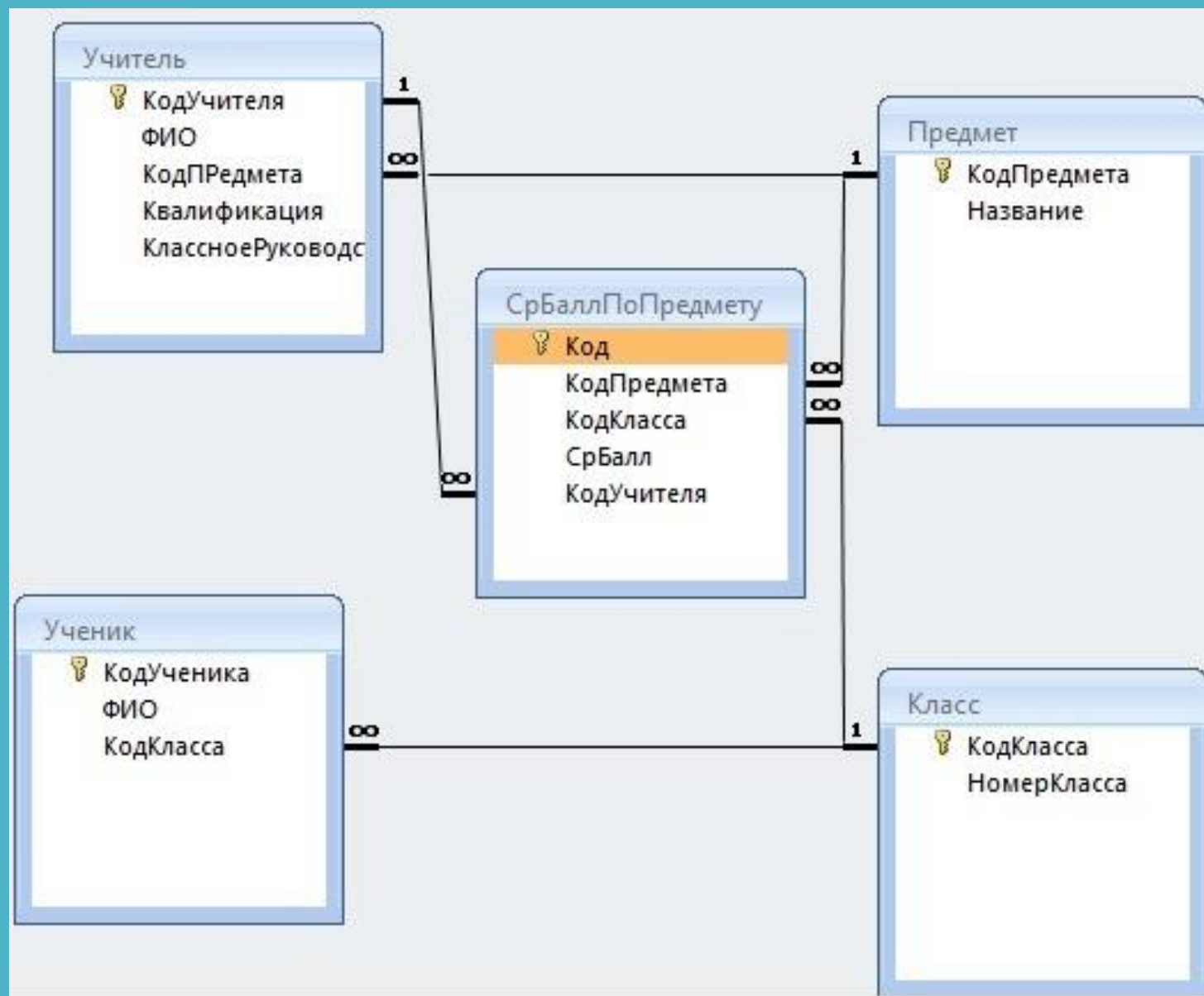
Понятие ER-диаграммы:

ER-диаграмма (Entity-Relationship диаграмма) — это графическая модель, которая используется для представления данных и взаимосвязей между ними. Она помогает проектировщикам баз данных наглядно представить структуру данных, их взаимосвязь и основные атрибуты сущностей.

# Пример. ER-диаграмма. Нотация Мартина.



# Диаграмма классов UML.



## Основные элементы ER-диаграммы:

**Сущности (таблицы):** Объекты, для которых необходимо хранить данные. Сущности обычно представлены прямоугольниками. Примеры сущностей: "Студент", "Курс", "Преподаватель".

**Атрибуты (поля):** Свойства или характеристики сущности, которые содержат данные. Атрибуты находятся внутри прямоугольника класса. Примеры атрибутов для сущности "Студент": "Имя", "Дата рождения", "Номер студенческого билета".

**Связи (отношения):** Отражают взаимодействие или ассоциации между сущностями. Связи изображаются линиями, которые соединяют сущности. Например, связь между сущностями "Клиент" и "Заказ" может означать, что один клиент может разместить несколько заказов.

## Типы связей:

- **Один к одному.** Каждая сущность А может быть связана только с одной сущностью В, и наоборот. Пример: каждый человек имеет один паспорт, и каждый паспорт принадлежит только одному человеку.
- **Один ко многим.** Одна сущность А может быть связана с несколькими сущностями В, но каждая сущность В может быть связана только с одной сущностью А. Пример: один преподаватель ведет несколько курсов, но каждый курс преподается только одним преподавателем.
- **Многие ко многим.** Несколько сущностей А могут быть связаны с несколькими сущностями В. Пример: студенты могут посещать несколько курсов, и каждый курс может быть посещён несколькими студентами.

## Популярные инструменты для построения ER-диаграмм:

- **MySQL Workbench:** Интегрированная среда разработки для MySQL, которая поддерживает создание ER-диаграмм и автоматическое преобразование их в схемы базы данных. Поддерживает реверс-инжиниринг для анализа существующих баз данных.
- **Microsoft Visio:** Мощный инструмент для создания диаграмм, включая ER-диаграммы. Предлагает шаблоны для моделирования баз данных с возможностью настройки сущностей, атрибутов и связей.
- **Lucidchart:** Онлайн-инструмент для создания различных диаграмм, включая ER-диаграммы. Поддерживает совместную работу в реальном времени, шаблоны и удобный интерфейс для моделирования баз данных.
- **Draw.io (diagrams.net):** Бесплатный онлайн-инструмент для создания диаграмм, включая ER-диаграммы. Поддерживает интеграцию с Google Drive, OneDrive и другими облачными сервисами для хранения проектов.
- **dbdiagram.io:** Простое и удобное онлайн-решение для создания ER-диаграмм с поддержкой синтаксиса для описания таблиц и связей. Предлагает экспорт моделей в различные форматы, такие как SQL и PDF.
- **и другие.**



# 4. Оптимизация структуры БД.

## Зачем нужна оптимизация?

- **Устранение избыточности.** Оптимизация помогает минимизировать повторяющиеся данные в базе, что уменьшает объем хранимой информации и снижает риск ошибок при обновлении данных.
- **Улучшение производительности.** Оптимизация ускоряет выполнение запросов, снижает нагрузку на сервер и повышает общую эффективность работы базы данных. Это особенно важно при работе с большими объемами данных или высокой частоте операций.

## Методы оптимизации:

- **Выбор правильных типов данных.** Подбор оптимальных типов данных для полей таблиц позволяет сократить объем хранимой информации и ускорить обработку запросов. Например, использование типа INT вместо BIGINT для числовых полей, когда это возможно, или выбор VARCHAR с оптимальной длиной для текстовых данных. **Пример:** если поле содержит данные о возрасте, вместо использования VARCHAR лучше выбрать TINYINT, что позволит сэкономить место и ускорить обработку данных.
- **Применение индексов.** Индексы значительно ускоряют поиск и выборку данных по определённым столбцам. Однако индексы также могут замедлить операции вставки и обновления данных, поэтому их следует применять разумно. **Пример:** индексирование поля "email" в таблице "Пользователи" позволит быстрее находить пользователей по их электронной почте.

- **Разбиение таблиц** (партиционирование).  
Партиционирование разделяет большие таблицы на меньшие сегменты (разделы) по определённым критериям, что ускоряет доступ к данным и упрощает управление ими.  
**Пример:** если таблица "Заказы" содержит данные за несколько лет, её можно разделить на части по году, что ускорит выборку данных за конкретный период.
- **Применение внешних ключей** для поддержания ссылочной целостности. Внешние ключи обеспечивают связь между таблицами и помогают поддерживать целостность данных, предотвращая создание «осиротевших» записей (например, заказов, связанных с несуществующими клиентами).  
**Пример:** связь между таблицами "Заказы" и "Клиенты", где внешний ключ в "Заказы" указывает на первичный ключ в "Клиенты", обеспечивает, что каждая запись о заказе всегда связана с существующим клиентом.

# 5. Нормализация данных.

**Нормализация** — это процесс организации структуры базы данных, направленный на минимизацию избыточности данных и предотвращение аномалий при обновлении, удалении или добавлении данных.

Нормализация разбивает большие, дублирующиеся таблицы на более маленькие связанные таблицы, что делает базу данных более логичной и управляемой.

**Роль нормализации:** Нормализация играет ключевую роль в проектировании базы данных, обеспечивая, что структура данных будет соответствовать принципам правильного хранения и организации. Она помогает предотвратить избыточное хранение данных и ошибки при обновлении информации, делая базы данных гибкими и согласованными.

## Цели нормализации:

- **Устранение избыточности данных.** Избыточные данные занимают лишнее место и могут привести к несогласованным записям. Нормализация помогает избежать дублирования, распределяя данные по нескольким связанным таблицам. Например, вместо хранения информации о клиенте в каждой записи о заказе, создается отдельная таблица клиентов, на которую ссылаются заказы.
- **Обеспечение целостности данных.** Нормализация позволяет поддерживать целостность данных, то есть гарантирует, что данные всегда будут корректными и непротиворечивыми. Это достигается за счет четкой структуры и разделения данных по смысловым категориям. Например, если номер клиента изменяется, достаточно обновить информацию в одной таблице, а не во всех связанных таблицах.
- **Улучшение производительности запросов.** Хотя нормализация изначально направлена на структурную целостность, в некоторых случаях она также может способствовать улучшению производительности запросов, особенно при работе с обновлениями и удалением данных. Меньшие таблицы с четкими связями позволяют быстрее выполнять выборки данных и избегать лишних операций.

Пример избыточных и несогласованных данных:

Customers			
id	email	name	city
1	<a href="mailto:ivanoff@bk.ru">ivanoff@bk.ru</a>	Иванов	Алматы
2	<a href="mailto:sidorov@gmail.com">sidorov@gmail.com</a>	Сидоров	Алма-Ата
3	<a href="mailto:dtepanof@ya.ru">dtepanof@ya.ru</a>	Степанова	Алмата
4	<a href="mailto:egorova@list.ru">egorova@list.ru</a>	Егорова	Almaty
5	<a href="mailto:petrovv@gmail.com">petrovv@gmail.com</a>	Петров	Астана
6	<a href="mailto:zuev@bk.com">zuev@bk.com</a>	Зуев	Актау

Пример без избыточных и несогласованных данных:

Customers				Cities	
id	email	name	city_id	id	title
1	<a href="mailto:ivanoff@bk.ru">ivanoff@bk.ru</a>	Иванов	1	1	Алматы
2	<a href="mailto:sidorov@gmail.com">sidorov@gmail.com</a>	Сидоров	1	2	Астана
3	<a href="mailto:dtepanof@ya.ru">dtepanof@ya.ru</a>	Степанова	1	3	Актау
4	<a href="mailto:egorova@list.ru">egorova@list.ru</a>	Егорова	1		
5	<a href="mailto:petrovv@gmail.com">petrovv@gmail.com</a>	Петров	2		
6	<a href="mailto:zuev@bk.com">zuev@bk.com</a>	Зуев	3		

# 6. Нормальные формы.

В теории нормализации данных существует несколько нормальных форм, которые используются для улучшения структуры базы данных. Основные нормальные формы включают:

1. Первая нормальная форма (1NF) — Гарантирует, что все данные в таблице атомарны (неделимы).
2. Вторая нормальная форма (2NF) — Устраняет частичные зависимости от первичного ключа, обеспечивая, что каждый неключевой атрибут полностью зависит от всего первичного ключа.
3. Третья нормальная форма (3NF) — Устраняет транзитивные зависимости, обеспечивая, что все неключевые атрибуты зависят только от первичного ключа и не зависят друг от друга.
4. Бойс-Кодд нормальная форма (BCNF) — Дополняет 3NF, устраняя случаи, когда определение функциональных зависимостей нарушает целостность данных, хотя она уже находится в 3NF.
5. Четвертая нормальная форма (4NF) — Устраняет многозначные зависимости, обеспечивая, что каждая независимая многозначная зависимость должна находиться в отдельной таблице.
6. Пятая нормальная форма (5NF) — Устраняет случаи, когда таблица может быть разделена на несколько таблиц, не теряя информацию о связях между данными, и предотвращает проблемы с проекцией данных.
7. Шестая нормальная форма (6NF) — Используется для обработки временных зависимостей и изменения данных, которые могут быть разделены по времени.



На практике чаще всего рассматривают первые три нормальные формы. Почему?

- **Практическая применимость.** 1NF, 2NF и 3NF покрывают большинство случаев в проектировании баз данных и решают основные проблемы избыточности и целостности данных, что делает их достаточными для большинства бизнес-приложений.
- **Сложность и избыточность.** BCNF и 4NF и последующие нормальные формы предназначены для более специализированных случаев и могут вводить дополнительную сложность, которая не всегда оправдана в реальных приложениях. Например, 4NF и 5NF часто применяются в теоретических исследованиях или очень сложных системах, где многозначные зависимости и проектирование данных играют ключевую роль.

## Первая нормальная форма (1NF).

**Определение:** Таблица находится в первой нормальной форме, если все её поля содержат только **атомарные** (неделимые) значения и каждая ячейка содержит только одно значение.

### **Требования:**

- Каждое поле таблицы должно содержать только **одно значение** (например, не должно быть списков или наборов значений в одной ячейке).
- Каждая строка таблицы должна быть **уникальной**, что достигается использованием первичного ключа.

Пример. Таблица, содержащая информацию о студентах и их курсах:

Students_Courses		
id	name	courses
1	Иванов	Python, Java
2	Сидоров	Python, CSharp

Students_Courses		
id	name	courses
1	Иванов	Python
2	Иванов	Java
3	Сидоров	Python
4	Сидоров	CSharp

## Вторая нормальная форма (2NF).

**Определение:** Таблица находится во второй нормальной форме, если она уже находится в 1NF и все неключевые атрибуты полностью функционально зависят от всего первичного ключа.

### **Требования:**

- Таблица должна быть в 1NF.
- Все атрибуты, не входящие в первичный ключ, должны **зависеть** от всего первичного ключа, а не от его части.

Пример. Таблица с данными о студентах и их факультетах:

Students_faculties			
Student_ID	StudentName	faculty	Dean_of_th_Faculty
1	Иванов	программирование	ДТН Егорова
2	Сидоров	кибербезопасность.	КТН Петров
3	Степанова	робототехника	КТН Зуев

Поле **Dean\_of\_th\_Faculty** (декан факультета) зависит только от **Faculty** (Факультет), а не от первичного ключа. Это означает, что знание **Faculty** позволяет определить **Dean\_of\_th\_Faculty**, независимо от значения StudentID.

Преобразование в 2NF. Для приведения таблицы в 2NF нужно устранить частичные зависимости и разделить данные на несколько таблиц:

Students_faculties		
Student_ID	StudentName	faculty
1	Иванов	программирование
2	Сидоров	кибербезопасность.
3	Степанова	робототехника

faculties	
faculty	Dean_of_th_Faculty
программирова	ДТН Егорова
кибербезопасно	КТН Петров
робототехника	КТН Зуев

Пример. Таблица с данными о заказах:

Orders			
OrderID	ProductID	ProductName	Quantity
1	101	Widget	10
1	102	Gadget	5
2	101	Widget	20

В данной таблице первичный ключ составной, состоящий из двух полей: OrderID и ProductID.

Поле ProductName зависит только от ProductID, а не от всего первичного ключа (OrderID и ProductID).

Это означает, что знание ProductID позволяет определить ProductName, независимо от значения OrderID.

Из-за этой частичной зависимости данные о продукте (например, ProductName) повторяются для каждого заказа, в котором этот продукт появляется.

Если название продукта изменится, его нужно будет обновить в каждой строке таблицы, что может привести к ошибкам и несогласованности.

Чтобы привести таблицу в 2NF, нужно устранить частичные зависимости и разделить данные на несколько таблиц:

Orders		
OrderID	ProductID	Quantity
1	101	10
1	102	5
2	101	20

Products	
ProductID	ProductName
101	Widget
102	Gadget



## Третья нормальная форма (3NF).

Определение: Таблица находится в третьей нормальной форме, если она уже находится во 2NF и все неключевые атрибуты не зависят транзитивно от первичного ключа.

### Требования:

- Таблица должна быть в 2NF.
- Никакой неключевой атрибут не должен зависеть от другого неключевого атрибута.

Предположим, у нас есть таблица с информацией о сотрудниках и их проектах:

Projects				
EmployeeID	EmployeeName	ProjectID	ProjectName	ProjectManager
1	John Doe	101	Alpha	Sarah Lee
1	John Doe	102	Beta	Mike Brown
2	Jane Smith	101	Alpha	Sarah Lee
3	Alice Johnson	103	Gamma	Emma Davis

Нарушение 3NF: Третья нормальная форма требует, чтобы все неключевые атрибуты зависели только от первичного ключа и не имели транзитивных зависимостей.

В этом примере, ProjectManager транзитивно зависит от EmployeeID, через ProjectID и ProjectName.

Чтобы привести таблицу в 3NF, нужно устранить транзитивные зависимости и разделить данные на несколько таблиц:

EmployeeProjects	
EmployeeID	ProjectID
1	101
1	102
2	101
3	103

Projects		
ProjectID	ProjectName	ProjectManager
101	Alpha	Sarah Lee
102	Beta	Mike Brown
103	Gamma	Emma Davis

# Пошаговое преобразование базы данных от 1NF до 3NF на конкретном примере базы данных магазина:

Orders					
OrderID	CustomerName	Product	Quantity	Price	CustomerAddress
1	Иван Иванов	Товар1	2	200	Адрес1
1	Иван Иванов	Товар2	1	100	Адрес1
2	Петр Петров	Товар1	1	200	Адрес2

Приведение к 2NF. 2NF требует устранения частичных зависимостей. Мы разделяем таблицу на две:

Orders		
OrderID	CustomerName	CustomerAddress
1	Иван Иванов	Адрес1
2	Петр Петров	Адрес2

OrderDetails			
OrderID	Product	Quantity	Price
1	Товар1	2	200
1	Товар2	1	100
2	Товар1	1	200

3NF требует устранения транзитивных зависимостей.  
Добавляем таблицы для клиентов и продуктов:

Customers		
CustomerID	CustomerName	CustomerAddress
1	Иван Иванов	Адрес1
2	Петр Петров	Адрес2

Products		
ProductID	ProductName	Price
1	Товар1	200
2	Товар2	100

Orders	
OrderID	CustomerID
1	1
2	2

OrderDetails		
OrderID	ProductID	Quantity
1	1	2
1	2	1
2	1	1

# Пример проектирования БД для онлайн-магазина:

## 1. Анализ требований

На этапе анализа требований проводится сбор данных о том, как будет использоваться база данных, какие процессы она должна поддерживать, и какие данные должны храниться.

В нашем случае цель — разработать базу данных для интернет-магазина, которая должна:

- Хранить информацию о продуктах (название, категория, цена, описание).
- Хранить данные о клиентах (имя, email, телефон).
- Хранить заказы клиентов (заказ, продукты в заказе, дата заказа).
- Обеспечивать целостность данных и эффективные запросы для выполнения основных операций (просмотр товаров, создание заказов и т. д.).



## 2. Проектирование концептуальной модели

### 2.1 Определение сущностей и атрибутов

Сущности — это ключевые объекты в системе. В нашем случае выделим три основные сущности:

- Product (Продукт) — продукт, продаваемый в интернет-магазине.
- Customer (Клиент) — человек, совершающий покупку.
- Order (Заказ) — заказ, который делает клиент.

**Для каждой сущности определяются основные атрибуты:**

- Product: ProductID, ProductName, Category, Price, Description.
- Customer: CustomerID, Name, Email, Phone.
- Order: OrderID, OrderDate, CustomerID (ссылка на сущность Customer), ProductID (ссылка на сущность Product), Quantity.

## 2.2 Определение связей

Связь между Customer и Order: один клиент может сделать много заказов. Связь «Один ко многим».

Связь между Order и Product: один заказ может содержать много продуктов, и один продукт может входить в разные заказы. Связь «Многие ко многим».

## 2.3 Построение ER-диаграммы (Entity-Relationship диаграмма)

На ER-диаграмме сущности изображаются в виде прямоугольников, атрибуты внутри прямоугольников, а связи — в виде линий.

# **Домашнее задание:**

1. Повторить материал лекции.

# Контрольные вопросы:

- Что такое нормализация и какова её основная цель?
- Какие этапы включает в себя проектирование базы данных?
- Что такое ER-диаграмма и какие элементы она включает?
- Какие существуют нормальные формы и чем они отличаются друг от друга?
- Почему важно определять первичные и внешние ключи в таблицах?
- Как нормализация помогает устранить избыточность данных?
- Что такое транзитивные зависимости и как их устраняют в процессе нормализации?
- Какую роль играют индексы в оптимизации базы данных?
- Как ER-диаграммы используются в процессе проектирования базы данных?
- Какие типы связей между таблицами вы знаете и как они отражаются в ER-диаграммах?

# Список литературы:

1. В. Ю. Кара-ушанов SQL — язык реляционных баз данных
2. А. Б. ГРАДУСОВ. Введение в технологию баз данных
3. А.Мотеев. Уроки MySQL

# **Материалы лекций:**

<https://github.com/ShViktor72/Education>

# **Обратная связь:**

[colledge20education23@gmail.com](mailto:colledge20education23@gmail.com)