Дайте определение 1НФ, 2НФ, 3НФ, НФБК, 4НФ

**Первая нормальная форма (1НФ)**

Таблица находится в первой нормальной форме, если:

1. Все атрибуты содержат атомарные (неделимые) значения.
2. В таблице нет повторяющихся строк.
3. Каждый столбец содержит значения только одного типа данных.

**Вторая нормальная форма (2НФ)**

Таблица находится во второй нормальной форме, если:

1. Она уже находится в 1НФ.
2. Все неключевые атрибуты полностью зависят от всего составного ключа (если ключ составной). Это означает, что не должно быть частичных зависимостей, когда неключевой атрибут зависит только от части составного ключа.

**Третья нормальная форма (3НФ)**

Таблица находится в третьей нормальной форме, если:

1. Она уже находится во 2НФ.
2. Никакой неключевой атрибут не зависит транзитивно от ключа. Это означает, что каждый неключевой атрибут должен зависеть только от первичного ключа и не должен зависеть от других неключевых атрибутов.

**Нормальная форма Бойса-Кодда (НФБК)**

Таблица находится в нормальной форме Бойса-Кодда, если:

1. Она уже находится в 3НФ.
2. Для любой функциональной зависимости X -> Y, X является суперклюзом, то есть множество X является или содержит ключ таблицы. НФБК более строгая форма 3НФ и решает некоторые аномалии, которые 3НФ может не устранить.

**Четвертая нормальная форма (4НФ)**

Таблица находится в четвертой нормальной форме, если:

1. Она уже находится в НФБК.
2. В таблице нет многозначных зависимостей. Многозначная зависимость возникает, когда один атрибут в таблице определяет множество значений другого атрибута независимо от других атрибутов.

Назовите основные правила перехода от концептуальной к логической реляционной схеме.

Переход от концептуальной модели данных к логической реляционной схеме включает преобразование высокоуровневых концептуальных элементов в реляционные структуры, которые могут быть реализованы в СУБД. Основные шаги этого процесса включают:

**1. Идентификация сущностей и атрибутов**

* **Сущности** концептуальной модели становятся таблицами в реляционной схеме.
* **Атрибуты** сущностей становятся столбцами этих таблиц.
* Определяются **первичные ключи** (Primary Keys) для каждой таблицы.

**2. Обработка связей (отношений)**

* **Связи один-к-одному (1:1)** могут быть реализованы как объединение двух сущностей в одну таблицу или путем добавления внешнего ключа (Foreign Key) из одной таблицы в другую.
* **Связи один-ко-многим (1:M)** реализуются путем добавления внешнего ключа в таблицу, представляющую "много".
* **Связи многие-ко-многим (M:N)** реализуются созданием промежуточной таблицы (связующей таблицы), которая содержит внешние ключи, ссылающиеся на первичные ключи обоих связанных таблиц.

**3. Нормализация данных**

* Применяются правила нормализации (1НФ, 2НФ, 3НФ и, при необходимости, НФБК и 4НФ) для устранения избыточности и предотвращения аномалий обновления.
* Таблицы разделяются или объединяются, чтобы обеспечить соблюдение нормальных форм.

**4. Определение доменов данных**

* Определяются **типы данных** для каждого столбца в соответствии с атрибутами концептуальной модели (например, INT, VARCHAR, DATE и т.д.).
* Задаются **ограничения целостности** (например, NOT NULL, UNIQUE).

**5. Уточнение ключей и ограничений**

* Уточняются **первичные ключи** для каждой таблицы.
* Определяются **внешние ключи** для установления и поддержания связей между таблицами.
* Добавляются другие **ограничения целостности** (например, CHECK, DEFAULT).

**6. Управление производительностью**

* Рассматривается **денормализация** для повышения производительности, если это необходимо, при сохранении целостности данных.
* Планируются **индексы** для ускорения поиска и соединений.

Как в реляционной модели реализуются супертипы и подтипы?

В реляционной модели супертипы и подтипы могут быть реализованы с помощью методов, таких как:

1. **Единая таблица** (Single Table Inheritance)
2. **Отдельные таблицы для каждого подтипа** (Class Table Inheritance)
3. **Смешанный подход** (Hybrid Approach)

**Единая таблица (Single Table Inheritance)**

* Все атрибуты супертипа и подтипов хранятся в одной таблице.
* Используется дополнительный столбец для идентификации типа записи.
* Столбцы, не применимые к определенному подтипу, могут содержать значения NULL.

**Отдельные таблицы для каждого подтипа (Class Table Inheritance)**

* Создаются отдельные таблицы для каждого подтипа.
* Каждая таблица содержит атрибуты подтипа и ссылку на супертип.
* Позволяет избежать NULL-значений и более гибко управлять атрибутами подтипов.

**Смешанный подход (Hybrid Approach)**

* Комбинирует преимущества двух предыдущих методов.
* Некоторые атрибуты хранятся в общей таблице супертипа, а специфичные атрибуты хранятся в отдельных таблицах для каждого подтипа.

**Супертипы (Supertypes)**

Супертип — это общая сущность, которая объединяет общие характеристики и атрибуты, общие для нескольких подтипов. Супертип представляет собой обобщение, из которого могут быть унаследованы общие атрибуты и связи.

Например, в базе данных, представляющей сотрудников организации, можно определить супертип "Сотрудник", который включает атрибуты, общие для всех типов сотрудников, такие как "ID", "Имя", "Фамилия", "Дата приема на работу".

**Подтипы (Subtypes)**

Подтип — это специализированная сущность, которая наследует атрибуты супертипа и добавляет свои уникальные атрибуты и связи. Подтипы могут иметь свои уникальные характеристики, которые отличают их от других подтипов того же супертипа.

Продолжая предыдущий пример, супертип "Сотрудник" может иметь подтипы "Административный персонал" и "Инженерный персонал". У подтипа "Административный персонал" могут быть дополнительные атрибуты, такие как "Отдел", а у подтипа "Инженерный персонал" — "Специализация".

**Как в реляционной модели реализуются рекурсивные связи?**

Рекурсивные связи в реляционных базах данных позволяют таблице ссылаться на саму себя. Это полезно для представления иерархий, таких как структура организации или категории товаров.

**Как в реляционной модели реализуются взаимно исключающие связи?**

В реляционной модели взаимно исключающие связи реализуются с помощью ограничения целостности, которое гарантирует, что определенные условия выполняются, чтобы предотвратить наличие конфликтующих данных. Чаще всего это достигается с помощью столбцов-суррогатов и проверки целостности данных.

Пример с двумя типами сотрудников

Предположим, у нас есть таблица сотрудников, и каждый сотрудник может быть либо полным, либо частичным, но не оба сразу.

Структура таблицы

Создаем таблицу Employee:

* EmployeeID — уникальный идентификатор сотрудника.
* Name — имя сотрудника.

Создаем таблицы для полных и частичных сотрудников:

* FullTimeEmployee с информацией для сотрудников на полной ставке.
* PartTimeEmployee с информацией для сотрудников на частичной ставке.

Ограничение целостности:

* Для реализации взаимно исключающих связей можно использовать триггеры или бизнес-логику в приложении, чтобы гарантировать, что EmployeeID существует только в одной из таблиц FullTimeEmployee или PartTimeEmployee, но не в обеих сразу.

**Какие шаблоны проектирования относятся к шаблонам 1:М?**

Если степень бинарной связи равна 1:М и класс принадлежности М-связной сущности обязательный, то достаточно использовать два отношения: по одному на каждую сущность, при условии, что ключ сущности служит в качестве первичного ключа для соответствующего отношения. Ключ же односвязной сущности должен быть добавлен как атрибут в отношение, отводимое Мсвязной сущности.

Если степень бинарной связи равна 1:М и класс принадлежности Мсвязной сущности необязателен, то необходимо использовать три отношения: по одному на сущность и одно для связи. Связь должна иметь среди своих атрибутов ключ сущности от каждой сущности.

**Какие шаблоны проектирования относятся к темпоральным шаблонам?**

Темпоральные шаблоны проектирования баз данных используются для отслеживания изменений данных во времени. Эти шаблоны полезны для обеспечения исторической точности данных, анализа изменений и поддержки временных запросов.

Шаблон "Период времени" (Period Table)

Шаблон "Темпоральная таблица" (Temporal Table)

Шаблон "История изменений" (Change History)

Шаблон "Состояние-Временная метка" (State-Timestamp)

Шаблон "Би-Темпоральная таблица" (Bi-Temporal Table)

Темпоральные шаблоны проектирования помогают отслеживать изменения данных и обеспечивают историческую точность. В зависимости от потребностей системы можно выбрать один или несколько таких шаблонов для реализации темпоральных данных.

**Назовите шаблоны проектирования иерархических структур.**

В проектировании баз данных и программного обеспечения иерархические структуры часто используются для представления данных с родительско-дочерними отношениями. Существует несколько шаблонов проектирования, которые помогают эффективно управлять иерархическими данными.

Adjacency List (Список смежности)

Этот шаблон представляет иерархические данные, указывая на родителя каждой записи.

Path Enumeration (Перечисление путей)

Этот шаблон сохраняет полный путь от корня до каждого узла.

Nested Sets (Вложенные множества)

Этот шаблон использует левый и правый индексы для обозначения вложенности.

Closure Table (Таблица замыканий)

Этот шаблон использует дополнительную таблицу для хранения всех пар предок-потомок.

Materialized Path (Материализованный путь)

Этот шаблон сохраняет путь в виде строки, представляющей путь от корня к узлу.

Каждый из этих шаблонов имеет свои преимущества и недостатки в зависимости от требований к данным иерархии и предполагаемых операций над этими данными. Выбор подходящего шаблона зависит от конкретных потребностей в производительности и сложности операций, связанных с иерархическими данными.

**Как реализуется семантическая целостность?**

Семантическая целостность в базе данных означает, что данные соответствуют правилам и ограничениям, определенным бизнес-логикой и требованиями приложения. Для обеспечения семантической целостности используются различные методы и инструменты.

Обеспечение семантической целостности данных является критически важным аспектом проектирования баз данных. Для этого используются разнообразные методы и инструменты, такие как ограничения, триггеры, хранимые процедуры, вьюхи, правила на уровне приложения и индексы. Выбор конкретного метода или их комбинации зависит от конкретных требований и структуры данных приложения.

**Приведите примеры нарушения семантическая целостность на уровне отношений.**

Нарушение семантической целостности на уровне отношений в базе данных происходит, когда данные не соответствуют бизнес-правилам или логике, которую они должны отражать. Вот несколько примеров таких нарушений:

1. Несоответствие внешнего ключа

Пример

В таблице "Сотрудники" (Employees) есть внешние ключи, которые ссылаются на таблицу "Отделы" (Departments). Если значение внешнего ключа в таблице "Сотрудники" указывает на несуществующий отдел, это нарушение семантической целостности.

Таблица "Отделы" (Departments):

DepartmentID (Primary Key)

DepartmentName

Таблица "Сотрудники" (Employees):

EmployeeID (Primary Key)

Name

DepartmentID (Foreign Key, ссылается на DepartmentID в таблице Departments)

Пример данных:

Таблица "Отделы": DepartmentID = 1, DepartmentName = "HR"

Таблица "Сотрудники": EmployeeID = 1, Name = "John", DepartmentID = 2

Здесь DepartmentID = 2 в таблице "Сотрудники" ссылается на несуществующий отдел.

2. Нарушение уникальности

Пример

В таблице "Сотрудники" (Employees) есть уникальное ограничение на столбец Email. Если в таблице есть две записи с одинаковым email, это нарушение семантической целостности.

Таблица "Сотрудники" (Employees):

EmployeeID (Primary Key)

Name

Email (UNIQUE)

Пример данных:

EmployeeID = 1, Name = "John", Email = "john@example.com"

EmployeeID = 2, Name = "Jane", Email = "john@example.com"

Здесь Email не уникален.

3. Нарушение ограничений на значения

Пример

В таблице "Сотрудники" (Employees) есть ограничение, что возраст сотрудников должен быть не менее 18 лет. Если в таблице есть запись с возрастом меньше 18, это нарушение семантической целостности.

Таблица "Сотрудники" (Employees):

EmployeeID (Primary Key)

Name

Age (CHECK Age >= 18)

Пример данных:

EmployeeID = 1, Name = "John", Age = 17

Здесь Age не соответствует заданному ограничению.

4. Несоответствие данных между связанными таблицами

Пример

В таблице "Заказы" (Orders) есть столбец CustomerID, который ссылается на таблицу "Клиенты" (Customers). Если в таблице "Клиенты" нет записи с CustomerID, указанным в таблице "Заказы", это нарушение семантической целостности.

Таблица "Клиенты" (Customers):

CustomerID (Primary Key)

CustomerName

Таблица "Заказы" (Orders):

OrderID (Primary Key)

OrderDate

CustomerID (Foreign Key, ссылается на CustomerID в таблице Customers)

Пример данных:

Таблица "Клиенты": CustomerID = 1, CustomerName = "Alice"

Таблица "Заказы": OrderID = 1, OrderDate = "2024-05-22", CustomerID = 2

Здесь CustomerID = 2 в таблице "Заказы" ссылается на несуществующего клиента.

5. Логическая несогласованность данных

Пример

В таблице "Сотрудники" (Employees) есть столбец HireDate (дата найма) и TerminationDate (дата увольнения). Если TerminationDate раньше, чем HireDate, это нарушение семантической целостности.

Таблица "Сотрудники" (Employees):

EmployeeID (Primary Key)

Name

HireDate

TerminationDate

Пример данных:

EmployeeID = 1, Name = "John", HireDate = "2024-01-01", TerminationDate = "2023-12-31"

Здесь TerminationDate раньше HireDate, что нелогично.

Заключение

Эти примеры показывают, как нарушение семантической целостности может проявляться на уровне отношений в базе данных. Чтобы предотвратить такие нарушения, необходимо правильно использовать ограничения, внешние ключи, триггеры и хранимые процедуры, а также тщательно проектировать структуру базы данных и бизнес-логику.

**Какие связи недопустимы для реляционной логической модели? По-другому: от каких связей избавляются при переходе от концептуальной модели к реляционной логической?**

При переходе от концептуальной модели к реляционной логической модели базы данных необходимо преобразовать связи "многие ко многим", иерархические связи, многозначные атрибуты и композитные сущности в структуры, поддерживаемые реляционной моделью. Это включает использование промежуточных таблиц для связей "многие ко многим", шаблонов проектирования для иерархий, разделение многозначных атрибутов на отдельные таблицы и декомпозицию композитных сущностей на более простые структуры.

Предварительные отношения для бинарных связей с типом соответствия 1:1

Правило 1: Если степень бинарной связи равна 1:1 и класс принадлежности обеих сущностей обязательный, то требуется только одно отношение. Первичным ключом этого отношения может быть ключ любой из этих двух сущностей. В этом случае гарантируется однократное появление каждого значения ключа в любом экземпляре отношения.

Правило 2: Если степень бинарной связи равна 1:1 и класс одной из сущностей необязательный, то необходимо построение двух отношений, под каждую сущность необходимо выделение одного отношения. Ключ сущности, для которого класс принадлежности является необязательным, добавляется в качестве атрибута в отношение, выделенное для сущности с обязательным классом принадлежности.

Правило 3: Если степень бинарной связи равна 1:1 и класс принадлежности ни одной из сущностей не является необязательным, то используется три отношения — по одному для каждой сущности — ключи которых служат в качестве первичных в соответствующих отношениях и одного для связи. Отношение, выделенное для связи, будет иметь по одному ключу сущности от каждой сущности.

Предварительные отношения для бинарных связей с типом соответствия М:М

Правило 6: Если степень бинарной связи равна М:М, то для хранения данных необходимо три отношения: по одному на сущность и одно для связи. Ключи сущности входят в связь. Если одна из сущностей вырождена, то - два отношения (т.е. достаточно будет двух таблиц).

Предварительные отношения для многосторонних связей

Правило 7: В случае трехсторонней связи необходимо использовать четыре отношения: по одному на сущность и одно для связи. Отношение, порождаемое связью, имеет в себе среди атрибутов ключи сущности от каждой сущности.