**Реляционные базы данных**

**1. Понятие о концептуальном (ER) моделировании баз данных. Основные элементы концептуальной модели.**

**Концептуальное ER-моделирование (Entity-Relationship)** – процесс создания структурной схемы базы данных, отражающей объекты предметной области, их свойства и взаимоотношения. Эта модель является мостом между бизнес-требованиями и логической структурой базы данных.

**Основные элементы концептуальной модели:**

**Тип сущности (Entity Type)** – класс однородных объектов предметной области. Например, “Сотрудник”, “Отдел”, “Проект”. Сущности изображаются прямоугольниками на ER-диаграмме.

**Сущность (Entity)** – конкретный экземпляр типа сущности. Например, “Иванов И.И.” как представитель типа “Сотрудник”.

**Атрибуты** – свойства, характеризующие сущность. Например, для “Сотрудника” атрибутами могут быть “Имя”, “Дата рождения”, “Зарплата”. Атрибуты могут быть простыми, составными, однозначными, многозначными и производными.

**Тип связи (Relationship Type)** – логическая ассоциация между типами сущностей. Например, “Работает в” связывает “Сотрудника” и “Отдел”. Связи отображаются ромбами или линиями на диаграммах.

**Сильные и слабые типы сущностей:** слабый тип сущности зависит от существования сильного (идентифицирующего) типа. Например, “Иждивенец” может быть слабой сущностью по отношению к “Сотруднику”.

**Параметры связей:** - **Арность связи** – количество участвующих типов сущностей (бинарная, тернарная и т.д.) - **Кардинальность** – количественное соотношение связываемых сущностей (1:1, 1:N, M:N) - **Степень участия** – обязательность участия в связи (полная или частичная)

**2. EER-моделирование и его необходимость. Подклассы и суперклассы.**

**Enhanced Entity-Relationship (EER)** моделирование расширяет классическую ER-модель, добавляя концепции, заимствованные из объектно-ориентированного программирования. Эти расширения необходимы для моделирования сложных предметных областей с многоуровневыми иерархиями и взаимосвязями.

**Основные концепции EER-моделирования:**

**Подклассы и суперклассы** – иерархия типов сущностей, где суперкласс обобщает общие атрибуты и связи, а подклассы представляют специализированные виды сущностей. Например, “Транспортное средство” может быть суперклассом для подклассов “Автомобиль”, “Мотоцикл”, “Грузовик”.

**Варианты моделирования иерархий:** - **Специализация** – процесс определения подклассов суперкласса (сверху вниз) - **Генерализация** – процесс выявления суперкласса на основе общих характеристик подклассов (снизу вверх)

**Ограничения непересечения и полноты:** - **Непересечение (Disjoint)** – сущность может принадлежать только одному подклассу - **Пересечение (Overlapping)** – сущность может принадлежать нескольким подклассам - **Полнота (Total)** – каждая сущность суперкласса должна принадлежать хотя бы одному подклассу - **Частичность (Partial)** – сущность суперкласса может не принадлежать ни одному подклассу

**Категоризация** – особый случай, когда подкласс может объединять сущности из разных суперклассов. Например, “Финансовый продукт” может включать как “Банковские счета”, так и “Инвестиционные портфели”.

**Варианты отображения EER-модели в реляционную модель:** - Создание отдельной таблицы для каждого класса с внешними ключами - Создание общей таблицы с полями для всех подклассов и дискриминатором - Создание отдельных таблиц только для подклассов с дублированием атрибутов суперкласса

**3. Кортежи (записи) реляционных таблиц. Атрибуты сущностей, домены атрибутов. Виды атрибутов. Ключевые атрибуты.**

**Кортежи (записи)** реляционных таблиц представляют собой отдельные строки, содержащие данные об одном экземпляре сущности. Каждый кортеж должен быть уникальным и атомарным (неделимым в контексте таблицы).

**Атрибуты сущностей** – характеристики или свойства сущностей, представленные столбцами таблицы. Каждый атрибут имеет имя и определенный тип данных.

**Домены атрибутов** – множество допустимых значений для атрибута. Определяют тип данных, размер, ограничения (NOT NULL, CHECK) и другие свойства. Использование доменов обеспечивает целостность данных и согласованность.

**Виды атрибутов:**

**Простые** – атомарные, неделимые значения (имя, номер телефона)

**Составные** – состоят из нескольких компонентов (адрес, включающий улицу, город, индекс)

**Однозначные** – имеют одно значение для каждой сущности (дата рождения)

**Многозначные** – могут иметь несколько значений (телефоны, навыки)

**Производные** – вычисляются на основе других атрибутов (возраст из даты рождения)

**Ключевые атрибуты** играют особую роль в организации данных:

**Первичный ключ (Primary Key)** – атрибут или группа атрибутов, уникально идентифицирующие каждый кортеж. Не может содержать NULL и должен быть уникальным.

**Потенциальный ключ** – атрибут или комбинация атрибутов, которые могут служить первичным ключом (обладают свойствами уникальности и минимальности).

**Альтернативный, простой, составной ключи** – другие виды ключей с соответствующими свойствами.

**Внешний ключ (Foreign Key)** – атрибут или группа атрибутов, ссылающиеся на первичный ключ другой таблицы. Обеспечивает ссылочную целостность и связи между таблицами.

**4. Понятие отношения (таблицы) как объекта реляционной алгебры. Формирование отношений-таблиц на основе концептуальной схемы.**

**Отношение (таблица)** в реляционной алгебре – математическое представление набора данных как множества кортежей, имеющих одинаковую структуру. Формально отношение R определяется как подмножество декартова произведения доменов D₁ × D₂ × … × Dₙ.

**Основные свойства отношений:** - Отсутствие дубликатов строк (кортежей) - Атомарность значений атрибутов - Отсутствие упорядоченности строк и столбцов - Каждый атрибут имеет уникальное имя

**Формирование отношений-таблиц на основе концептуальной схемы** требует трансформации ER-модели в реляционную модель:

**Для сущностей с простыми атрибутами:** - Каждый тип сущности преобразуется в отдельную таблицу - Атрибуты становятся столбцами таблицы - Ключевой атрибут становится первичным ключом

**Для сущностей с составными атрибутами:** - Составные атрибуты могут быть разложены на компоненты - Каждый компонент становится отдельным столбцом - Либо создается отдельная таблица для составного атрибута с внешним ключом

**Для сущностей с многозначными атрибутами:** - Создается отдельная таблица для многозначного атрибута - Таблица содержит первичный ключ исходной сущности как внешний ключ

**Реализация связей различной кардинальности:**

**Связь 1:1** – добавление внешнего ключа в одну из таблиц или в обе с уникальным ограничением

**Связь 1:N** – добавление внешнего ключа в таблицу на стороне “многие”

**Связь M:N** – создание промежуточной таблицы с внешними ключами к обеим связываемым таблицам

**Отсутствующие и неопределённые значения (NULL)** указывают на отсутствие информации или неприменимость атрибута. NULL требует особой обработки в запросах и может влиять на целостность данных.

**5. Элементы реляционной алгебры. Операции: объединение, пересечение, разность, проекция, селекция, декартово произведение, θ-соединение.**

**Реляционная алгебра** – формальная система операций над отношениями (таблицами), которая служит теоретической основой языков запросов, включая SQL. Операции реляционной алгебры позволяют манипулировать таблицами для получения новых отношений.

**Основные операции реляционной алгебры:**

**Объединение (Union)** – формирует новое отношение, содержащее все уникальные кортежи из двух отношений. Записывается как R ∪ S. Требует совместимости по типу отношений.

**Пересечение (Intersection)** – создает отношение, содержащее только кортежи, присутствующие в обоих исходных отношениях. Записывается как R ∩ S.

**Разность (Difference)** – формирует отношение, содержащее кортежи из первого отношения, которых нет во втором. Записывается как R − S.

**Проекция (Projection)** – создает новое отношение путем выбора указанных атрибутов и удаления дубликатов. Записывается как π₁₂..ₙ(R).

**Селекция (Selection)** – выбирает кортежи, удовлетворяющие заданному условию. Записывается как σₚ(R), где p – предикат (условие отбора).

**Декартово произведение (Cartesian Product)** – комбинирует каждый кортеж первого отношения с каждым кортежем второго. Записывается как R × S.

**θ-соединение (Theta Join)** – комбинация декартова произведения и селекции по условию. Записывается как R ⋈θ S, где θ – условие соединения.

**Естественное соединение (Natural Join)** – эквисоединение по всем одноименным атрибутам с исключением дублирующихся столбцов. Записывается как R ⋈ S.

**Агрегативные функции** в реляционной алгебре включают COUNT, SUM, AVG, MIN, MAX.

**Операция группировки** позволяет применять агрегативные функции к группам кортежей, сформированным по одному или нескольким атрибутам.

**6. Элементы языка SQL. Оператор SELECT и его запись.**

**Язык SQL (Structured Query Language)** – универсальный язык для работы с реляционными базами данных, включающий операторы определения данных (DDL), манипулирования данными (DML), управления доступом (DCL) и транзакциями (TCL).

**Оператор SELECT** – основной инструмент для извлечения данных из таблиц. Общая структура запроса:

SELECT [DISTINCT] список\_столбцов  
FROM список\_таблиц  
[WHERE условие\_фильтрации]  
[GROUP BY столбцы\_группировки]  
[HAVING условие\_для\_групп]  
[ORDER BY столбцы\_сортировки [ASC|DESC]]  
[LIMIT количество [OFFSET смещение]];

**Переименование полей и таблиц** осуществляется с помощью псевдонимов (aliases): SELECT e.employee\_id AS "ID", e.last\_name AS "Фамилия" FROM employees e WHERE e.department\_id = 10;

**Реализация реляционных операций в SQL:**

**Селекция:** SELECT \* FROM employees WHERE salary > 5000;

**Проекция:** SELECT employee\_id, last\_name FROM employees;

**Объединение:** SELECT id FROM table1 UNION SELECT id FROM table2;

**Пересечение:** SELECT id FROM table1 INTERSECT SELECT id FROM table2;

**Разность:** SELECT id FROM table1 EXCEPT SELECT id FROM table2;

**Агрегативные функции:** SELECT COUNT(\*), AVG(salary), MAX(hire\_date) FROM employees;

**Запросы с группировкой:** SELECT department\_id, COUNT(\*), AVG(salary) FROM employees GROUP BY department\_id HAVING AVG(salary) > 5000;

**Условия IN, EXISTS:** SELECT employee\_id FROM employees WHERE department\_id IN (SELECT id FROM departments WHERE location\_id = 1700);

SELECT d.name FROM departments d WHERE EXISTS (SELECT 1 FROM employees e WHERE e.department\_id = d.id);

**7. Нормализация отношений-таблиц. Возможные аномалии обновления. Функциональные зависимости между атрибутами.**

**Нормализация** – процесс организации данных в реляционной базе для минимизации избыточности и устранения аномалий обновления. Она заключается в последовательном приведении таблиц к нормальным формам.

**Возможные аномалии обновления** в ненормализованных таблицах:

**Аномалия вставки** – невозможность добавить данные без наличия полной информации

**Аномалия удаления** – непреднамеренная потеря данных при удалении записи

**Аномалия обновления** – необходимость изменения множества записей для обновления одного факта

**Функциональные зависимости между атрибутами** – ключевое понятие нормализации:

**Функциональная зависимость (ФЗ)** – отношение между атрибутами, при котором значение одного атрибута однозначно определяет значение другого. Записывается как X → Y.

**Детерминант** – атрибут или группа атрибутов, от которых функционально зависят другие атрибуты.

**Полная функциональная зависимость** – зависимость Y от X, при которой Y не зависит от любого подмножества X (важно для 2НФ).

**Транзитивная зависимость** – ситуация, когда A → B и B → C, что означает A → C через B (важно для 3НФ).

**Нормальные формы и порядок приведения:**

**Первая нормальная форма (1НФ)** – атомарность значений, отсутствие повторяющихся групп.

Для приведения: разбить составные атрибуты на простые, создать отдельные таблицы для многозначных атрибутов.

**Вторая нормальная форма (2НФ)** – таблица в 1НФ, где каждый неключевой атрибут полностью зависит от первичного ключа.

Для приведения: выделить атрибуты, зависящие от части составного ключа, в отдельные таблицы.

**Третья нормальная форма (3НФ)** – таблица в 2НФ, где отсутствуют транзитивные зависимости неключевых атрибутов от первичного ключа.

Для приведения: выделить транзитивно зависимые атрибуты в отдельные таблицы.

На практике большинство баз данных нормализуют до 3НФ, что обеспечивает хороший баланс между устранением аномалий и производительностью.