Реляционные базы данных

1. Понятие о концептуальном (ER) моделировании баз данных

Концептуальное (ER) моделирование — это процесс создания абстрактного представления данных, отражающего структуру предметной области и игнорирующего аспекты реализации.

Основные элементы концептуальной модели:

Тип сущности

Тип сущности — абстрактный класс объектов реального мира с общими свойствами (Студент, Книга, Заказ). На ER-диаграммах обозначается прямоугольником.

Сущность

Сущность — конкретный экземпляр типа сущности (студент "Иванов И.И.").

Атрибуты

Атрибуты — свойства, характеризующие тип сущности (имя, дата рождения, номер). На диаграммах обозначаются овалами, соединенными с типом сущности.

Виды атрибутов:

- **Простые** — атомарные, неделимые (имя)
- **Составные** — состоящие из нескольких простых (адрес)
- **Однозначные** — имеющие одно значение (номер паспорта)
- **Многозначные** — имеющие несколько значений (телефоны)
- **Производные** — вычисляемые на основе других атрибутов (возраст)

Ключевые атрибуты

Ключевые атрибуты — атрибуты, уникально идентифицирующие сущность. Обозначаются подчеркиванием.

Тип связи

Тип связи — логическое отношение между типами сущностей. Обозначается ромбом, соединенным с участвующими типами сущностей.

Связь

Связь — конкретный экземпляр типа связи (студент "Иванов" учится в группе "ПИ-101").

Сильные и слабые типы сущностей:

- **Сильные** типы сущностей существуют независимо от других сущностей
- **Слабые** типы сущностей зависят от существования других (родительских) сущностей и не имеют собственных ключевых атрибутов

Связи между сущностями:

Арность связи

Количество типов сущностей, участвующих в связи:

- **Унарная (рекурсивная)** — связь между сущностями одного типа (сотрудник руководит сотрудниками)
- **Бинарная** — связь между двумя типами сущностей (студент учится в группе)
- **N-арная** — связь между тремя и более типами сущностей

Показатель кардинальности связи

- **1:1** — одной сущности первого типа соответствует не более одной сущности второго типа
- **1:N** — одной сущности первого типа соответствует произвольное количество сущностей второго типа
- **M:N** — между сущностями первого и второго типа существует произвольное соответствие

Степень участия в связи

- **Полная (обязательная)** — каждый экземпляр сущности обязательно участвует в связи
- **Частичная (необязательная)** — не каждый экземпляр сущности участвует в связи

Атрибуты связи

Атрибуты, характеризующие не сущности, а их взаимоотношения (дата найма, оценка за экзамен).

Реляционные базы данных

2. EER-моделирование и его необходимость

EER-моделирование (Enhanced Entity-Relationship) — это расширение стандартной ER-модели, добавляющее концепции и возможности объектно-ориентированного подхода.

Необходимость EER-моделирования:

- Более точное моделирование сложных предметных областей
- Представление иерархических структур данных

- Отражение отношений наследования
- Улучшение соответствия между концептуальной моделью и реальным миром

Подклассы и суперклассы:

- **Суперкласс** (суперипт) — обобщающий тип сущности
- **Подкласс** (подтип) — специализированный тип сущности

Варианты моделирования: специализация и генерализация

Специализация

Процесс перехода от суперкласса к подклассам. Выделяет подтипы на основе различий в атрибутах и связях:

- Начинается с суперкласса и определяет его подклассы
- Подклассы наследуют все атрибуты и связи суперкласса
- Подклассы могут иметь уникальные атрибуты и связи
- Позволяет выразить "IS-A" отношения (Сотрудник ЭТО Человек)

Генерализация

Процесс перехода от подклассов к суперклассу:

- Начинается с нескольких типов сущностей с общими свойствами
- Определяет общий суперкласс, объединяющий эти свойства
- Извлекает общие атрибуты и связи в суперкласс
- Нисходящий подход (снизу вверх)

Ограничения непересечения и полноты:

Ограничения непересечения:

- **Непересекающиеся подклассы** (disjoint) — сущность может принадлежать только к одному подклассу
- **Пересекающиеся подклассы** (overlapping) — сущность может принадлежать к нескольким подклассам

Ограничения полноты:

- **Полное покрытие** (total) — каждая сущность суперкласса должна принадлежать хотя бы одному подклассу
- **Частичное покрытие** (partial) — сущность суперкласса может не принадлежать ни одному подклассу

Категоризация:

Особый тип отношения, при котором сущность может принадлежать к подклассу нескольких разнотипных суперклассов.

Варианты отображения EER-модели в реляционную модель:

1. **Вертикальное отображение** (Single Table Inheritance):
    - Одна таблица для всей иерархии
    - Включает атрибуты всех подклассов
    - Содержит дискриминатор для определения типа
2. **Горизонтальное отображение** (Class Table Inheritance):
    - Отдельная таблица для каждого подкласса
    - Каждая таблица содержит как свои, так и унаследованные атрибуты
3. **Смешанное отображение** (Joined Table Inheritance):
    - Таблица для суперкласса с общими атрибутами
    - Отдельные таблицы для подклассов с уникальными атрибутами
    - Связаны через первичный ключ

Иерархии и решётки:

- **Иерархия** — структура, где каждый подкласс имеет только один суперкласс
- **Решётка** — структура, где подкласс может иметь несколько суперклассов (множественное наследование)

Реляционные базы данных

3. Кортежи (записи) реляционных таблиц. Атрибуты сущностей, домены атрибутов.

Кортежи (записи) реляционных таблиц

Кортеж (tuple) — строка таблицы в реляционной модели данных, представляющая один экземпляр сущности или один экземпляр отношения между сущностями. В теории реляционных баз данных каждый кортеж соответствует утверждению, что описываемый объект обладает определёнными значениями атрибутов.

Свойства кортежей:

- Все кортежи разные (отсутствуют дубликаты)
- Порядок кортежей не имеет значения
- Каждый кортеж состоит из значений атрибутов
- Каждый кортеж должен быть уникально идентифицируем

Атрибуты сущностей

Атрибут — именованное свойство сущности, характеризующее её. В реляционной таблице атрибуты представлены столбцами.

Виды атрибутов:

По структуре:

- **Простые** — атомарные, неделимые значения (имя, год рождения)
- **Составные** — состоят из нескольких компонентов (полный адрес = улица + дом + квартира)

По количеству значений:

- **Однозначные** — содержат одно значение для каждой сущности (номер паспорта)
- **Многозначные** — могут содержать несколько значений (телефоны, навыки)

По происхождению:

- **Хранимые** — значения хранятся в базе данных
- **Производные** — значения вычисляются из других атрибутов (возраст из даты рождения)

Ключевые атрибуты:

- **Первичный ключ** — атрибут или набор атрибутов, уникально идентифицирующий кортеж
- **Потенциальный ключ** — атрибут или группа атрибутов, которые могут служить первичным ключом
- **Альтернативный ключ** — потенциальный ключ, не выбранный в качестве первичного
- **Простой ключ** — ключ, состоящий из одного атрибута
- **Составной ключ** — ключ, состоящий из нескольких атрибутов
- **Внешний ключ** — атрибут или набор атрибутов таблицы, ссылающийся на первичный ключ другой таблицы

Домены атрибутов

Домен — множество допустимых значений атрибута. Определяет:

- Тип данных (числовой, строковый, дата)
- Диапазон значений (минимальное и максимальное значение)
- Ограничения (формат, шаблон, перечисления)
- Семантику (смысловое значение)

Домены обеспечивают:

- Целостность данных
- Согласованность значений атрибутов
- Возможность семантической проверки
- Основу для поддержки бизнес-правил

Примеры доменов:

- Возраст человека (целое число от 0 до 150)
- Номер телефона (строка определённого формата)

- Статус заказа (одно из перечисленных значений: "новый", "обработан", "доставлен")
- Дата рождения (дата, не превышающая текущую дату)

Реляционные базы данных

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

Понятие отношения (таблицы) как объекта реляционной алгебры

Отношение — основное понятие реляционной модели данных, представляющее собой подмножество декартова произведения доменов. В практическом смысле отношение — это таблица со следующими свойствами:

1. Каждая ячейка таблицы содержит ровно одно атомарное (неделимое) значение
2. Все значения в одном столбце имеют один и тот же домен
3. Каждый столбец имеет уникальное имя (имя атрибута)
4. Порядок столбцов не имеет значения
5. Каждая строка уникальна (нет дублирующих строк)
6. Порядок строк не имеет значения

В реляционной алгебре отношение обозначается как: $R(A_1, A_2, ..., A_n)$, где:

- R — имя отношения
- $A_1, A_2, ..., A_n$ — имена атрибутов

Формирование отношений-таблиц на основе концептуальной (ER) схемы

Для сущностей с простыми атрибутами

Каждому типу сущности соответствует отдельная таблица:

- Имя таблицы = имя типа сущности
- Столбцы таблицы = атрибуты сущности
- Первичный ключ таблицы = ключевой атрибут сущности

Пример:

```
Сущность СОТРУДНИК (Табельный_номер, ФИО, Дата_рождения, Должность)
↓
Таблица СОТРУДНИК
Табельный_номер (PK) | ФИО | Дата_рождения | Должность
```

Для сущностей с многозначными атрибутами

Создается дополнительная таблица для многозначного атрибута:

- Первичный ключ новой таблицы = комбинация первичного ключа исходной сущности и значения многозначного атрибута

Пример:

```
Сущность СОТРУДНИК (Табельный_номер, ФИО, Телефоны, Должность)
↓
Таблица СОТРУДНИК
Табельный_номер (PK) | ФИО | Должность

Таблица ТЕЛЕФОНЫ_СОТРУДНИКОВ
Табельный_номер (PK, FK) | Телефон (PK)
```

Реализация связей различной кардинальности

1. **Связь 1:1**:
   - Первичный ключ одной таблицы становится внешним ключом в другой
   - Можно также объединить таблицы, если это семантически оправдано

   Пример: Сотрудник имеет один рабочий ноутбук, ноутбук закреплен за одним сотрудником

   ```
   СОТРУДНИК
   Табельный_номер (PK) | ФИО | ...

   НОУТБУК
   Инвентарный_номер (PK) | Модель | Табельный_номер_сотрудника
   (FK, UNIQUE)
   ```

2. **Связь 1:N**:
   - Первичный ключ таблицы на стороне "1" становится внешним ключом в таблице на стороне "N"

   Пример: Отдел содержит множество сотрудников

   ```
   ОТДЕЛ
   Код_отдела (PK) | Название | ...

   СОТРУДНИК
   Табельный_номер (PK) | ФИО | ... | Код_отдела (FK)
   ```

3. **Связь M:N**:
   - Создается промежуточная таблица (таблица связей)
   - Первичный ключ таблицы связей = комбинация первичных ключей связываемых таблиц

Пример: Студенты изучают курсы, один курс может изучаться многими студентами

```
СТУДЕНТ
Номер_студента (PK) | ФИО | ...

КУРС
Код_курса (PK) | Название | ...

ИЗУЧЕНИЕ
Номер_студента (PK, FK) | Код_курса (PK, FK) | Дата_начала |
Оценка
```

Отсутствующие и неопределённые значения (NULL)

Значение NULL в реляционной модели используется для обозначения:

- Отсутствующего значения (значение существует, но неизвестно)
- Неприменимого значения (значение не существует для данного кортежа)

Особенности NULL:

- NULL ≠ NULL (два NULL-значения не считаются равными)
- Операции с NULL дают NULL или специфический результат
- При использовании NULL должны соблюдаться ограничения целостности

Реляционные базы данных

5. Элементы реляционной алгебры

Реляционная алгебра — формальная система операций над отношениями, являющаяся теоретической основой языков запросов реляционных СУБД, включая SQL.

Основные операции реляционной алгебры:

Унарные операции (над одним отношением):

1. **Проекция (π)** — создает новое отношение, содержащее подмножество столбцов исходного отношения.

   ```
   π[Имя, Фамилия](Сотрудники)
   ```

   Выбирает только указанные атрибуты. Дубликаты строк автоматически удаляются.

2. **Селекция (σ)** — создает новое отношение, содержащее подмножество строк исходного отношения, удовлетворяющих заданному условию.

   ```
   σ[Отдел='ИТ'](Сотрудники)
   ```

Выбирает только строки, удовлетворяющие условию.

Бинарные операции (над двумя отношениями):

3. **Объединение (∪)** — создает новое отношение, содержащее все строки, входящие хотя бы в одно из двух исходных отношений.

   ```
   Сотрудники_Москва ∪ Сотрудники_СПб
   ```

   Требует совместимости отношений по типу (одинаковая схема).

4. **Пересечение (∩)** — создает новое отношение, содержащее строки, входящие в оба исходных отношения.

   ```
   Программисты ∩ Аналитики
   ```

   Требует совместимости отношений по типу.

5. **Разность (–)** — создает новое отношение, содержащее строки первого отношения, не входящие во второе.

   ```
   Сотрудники – Уволенные
   ```

   Требует совместимости отношений по типу.

6. **Декартово произведение (×)** — создает новое отношение, каждая строка которого содержит все атрибуты первого отношения, за которыми следуют все атрибуты второго, для всех возможных комбинаций строк исходных отношений.

   ```
   Сотрудники × Отделы
   ```

   Результат содержит m×n строк, где m и n — количество строк в исходных отношениях.

7. **θ-соединение (⋈)** — создает новое отношение путем комбинирования строк двух отношений, удовлетворяющих заданному условию θ (где θ — операция сравнения).

   ```
   Сотрудники ⋈[Сотрудники.ИдОтдела = Отделы.Ид] Отделы
   ```

   Специальные случаи:

   - **Естественное соединение (⋈)** — соединение по всем одноименным атрибутам
   - **Эквисоединение** — соединение с условием равенства атрибутов
   - **Внешнее соединение** — сохраняет строки, не имеющие пары в другом отношении

Агрегативные функции и операция группировки:

Агрегативные функции выполняют вычисления над группами строк и возвращают одно значение для каждой группы:

- **COUNT** — количество строк
- **SUM** — сумма значений
- **AVG** — среднее значение
- **MIN** — минимальное значение
- **MAX** — максимальное значение

Операция группировки делит отношение на группы и применяет агрегативные функции к каждой группе:

```
GROUP BY[Отдел](SUM[Зарплата] AS СуммаЗарплат)(Сотрудники)
```

Примеры запросов в реляционной алгебре:

1. Найти имена и фамилии сотрудников ИТ-отдела:

   ```
   π[Имя, Фамилия](σ[Отдел='ИТ'](Сотрудники))
   ```

2. Найти сотрудников с их отделами:

   ```
   Сотрудники ⋈[Сотрудники.ИдОтдела = Отделы.Ид] Отделы
   ```

3. Найти отделы с количеством сотрудников:

   ```
   GROUP BY[ИдОтдела](COUNT[*] AS КоличествоСотрудников)
   (Сотрудники)
   ```

Реляционные базы данных

6. Элементы языка SQL. Оператор SELECT и его запись. Переименование полей и таблиц в запросах.

SQL (Structured Query Language) — язык структурированных запросов, применяемый для управления данными в реляционных базах данных.

Оператор SELECT и его запись

Оператор SELECT используется для извлечения данных из базы данных и имеет следующую базовую структуру:

```
SELECT [DISTINCT] <список_выбора>
FROM <источник_данных>
[WHERE <условие_фильтрации>]
[GROUP BY <критерий_группировки>]
[HAVING <условие_для_групп>]
[ORDER BY <порядок_сортировки>]
[LIMIT <ограничение>]
```

Компоненты оператора SELECT:

1. **SELECT** — определяет список извлекаемых столбцов:

   ```
   SELECT имя, фамилия, зарплата
   ```

   Можно использовать:

   - `*` для выбора всех столбцов:

     ```
     SELECT *
     ```

   - `DISTINCT` для исключения дубликатов:

     ```
     SELECT DISTINCT город
     ```

   - Выражения и функции:

     ```
     SELECT имя, фамилия, зарплата * 1.1 AS новая_зарплата
     ```

2. **FROM** — указывает источник данных (таблицы, представления):

   ```
   FROM сотрудники
   ```

   Можно использовать соединения таблиц:

   ```
   FROM сотрудники JOIN отделы ON сотрудники.id_отдела = отделы.id
   ```

3. **WHERE** — задает условия фильтрации строк:

   ```
   WHERE зарплата > 50000 AND отдел = 'ИТ'
   ```

   Операторы в условиях:

   - Сравнения: `=`, `<>`, `<`, `>`, `<=`, `>=`
   - Логические: `AND`, `OR`, `NOT`
   - Включения: `IN`, `NOT IN`

- Шаблоны: `LIKE`, `NOT LIKE`
- Диапазоны: `BETWEEN`, `NOT BETWEEN`
- Проверка NULL: `IS NULL`, `IS NOT NULL`

4. **GROUP BY** — группирует строки по указанным столбцам:

```
GROUP BY отдел
```

5. **HAVING** — фильтрует результаты группировки:

```
HAVING COUNT(*) > 5
```

6. **ORDER BY** — определяет порядок сортировки результатов:

```
ORDER BY фамилия ASC, имя DESC
```

7. **LIMIT** — ограничивает количество возвращаемых строк:

```
LIMIT 10
```

Переименование полей и таблиц в запросах

Переименование полей (назначение псевдонимов):

Используется ключевое слово `AS` (может быть опущено):

```
SELECT
    имя AS first_name,
    фамилия AS last_name,
    зарплата * 1.1 AS новая_зарплата
FROM сотрудники
```

Псевдонимы полей:

- Позволяют давать понятные имена выражениям
- Необходимы для именования результатов функций и выражений
- Могут использоваться для форматирования выходных данных
- Не могут использоваться в WHERE той же инструкции

Переименование таблиц:

Таблицам также можно назначать псевдонимы:

```sql
SELECT
    e.имя,
    e.фамилия,
    d.название AS название_отдела
FROM
    сотрудники AS e
JOIN
    отделы AS d ON e.id_отдела = d.id
```

Преимущества псевдонимов таблиц:

- Сокращение записи (особенно для длинных имен таблиц)
- Обязательны при соединении таблицы с самой собой
- Повышают читаемость при работе с несколькими таблицами
- Позволяют избежать конфликтов имен столбцов

Реализация в SQL реляционных операций

Селекция (выборка строк):

```sql
SELECT * FROM сотрудники WHERE отдел = 'ИТ'
```

Проекция (выборка столбцов):

```sql
SELECT имя, фамилия FROM сотрудники
```

Объединение:

```sql
SELECT * FROM сотрудники_москва
UNION
SELECT * FROM сотрудники_спб
```

Пересечение:

```sql
SELECT * FROM программисты
INTERSECT
SELECT * FROM аналитики
```

Разность:

```sql
SELECT * FROM сотрудники
EXCEPT
SELECT * FROM уволенные
```

Реляционные базы данных

7. Нормализация отношений-таблиц. Возможные аномалии обновления.

Аномалии обновления

Аномалии обновления — проблемы, возникающие при вставке, изменении или удалении данных в таблицах с плохой структурой. Различают три типа аномалий:

1. Аномалия вставки

Невозможность добавить новые данные из-за отсутствия значений для всех атрибутов или из-за дублирования ключей.

Пример: Необходимо добавить информацию о новом отделе, но для этого требуется указать хотя бы одного сотрудника, хотя отдел ещё не укомплектован.

2. Аномалия изменения

Необходимость изменения данных во многих строках при изменении одного значения, что может привести к несогласованности.

Пример: При изменении названия отдела требуется обновить это значение во всех строках, относящихся к сотрудникам этого отдела.

3. Аномалия удаления

Непреднамеренное удаление ценных данных при удалении других данных.

Пример: При удалении последнего сотрудника отдела теряется информация о самом отделе.

Функциональные зависимости между атрибутами

Функциональная зависимость ☑ = $t_2[X]$, то $t_1[Y] = t_2[Y]$.

Проще говоря, если мы знаем значение X, то всегда можем определить значение Y.

Типы функциональных зависимостей:

- **Детерминанты зависимостей** — атрибуты, от которых зависят другие атрибуты
- **Полные функциональные зависимости** — Y функционально зависит от X, и не существует подмножества X', такого что Y функционально зависит от X'
- **Частичные функциональные зависимости** — функциональные зависимости от части составного ключа
- **Транзитивные функциональные зависимости** — если X → Y и Y → Z, то X → Z называется транзитивной зависимостью

Первая, вторая, третья нормальные формы

Первая нормальная форма (1NF)

Отношение находится в первой нормальной форме, если:

- Все атрибуты атомарны (неделимы)
- Нет повторяющихся групп атрибутов

Приведение к 1NF:

1. Выделить повторяющиеся группы в отдельные таблицы
2. Разделить составные атрибуты на компоненты

Вторая нормальная форма (2NF)

Отношение находится во второй нормальной форме, если:

- Оно находится в 1NF
- Все неключевые атрибуты полностью функционально зависят от первичного ключа (нет частичных зависимостей)

Приведение к 2NF:

1. Определить первичный ключ
2. Идентифицировать атрибуты, зависящие только от части составного ключа
3. Выделить эти атрибуты в отдельную таблицу с соответствующей частью ключа

Третья нормальная форма (3NF)

Отношение находится в третьей нормальной форме, если:

- Оно находится в 2NF
- Все неключевые атрибуты нетранзитивно зависят от первичного ключа (нет транзитивных зависимостей)

Приведение к 3NF:

1. Определить транзитивные зависимости
2. Выделить атрибуты, образующие транзитивную зависимость, в отдельную таблицу

Порядок приведения таблиц к каждой из нормальных форм

Порядок нормализации:

1. **Приведение к 1NF**:
    - Идентифицировать и устранить повторяющиеся группы
    - Создать отдельную таблицу для каждого набора связанных данных
    - Определить первичный ключ для каждой таблицы
2. **Приведение к 2NF**:

- Если первичный ключ — составной, проверить все неключевые атрибуты на зависимость от части ключа
- Выделить атрибуты, зависящие от части ключа, в отдельные таблицы
- Установить соответствующую часть первичного ключа как первичный ключ в новых таблицах

3. **Приведение к 3NF**:

- Идентифицировать неключевые атрибуты, которые определяют другие неключевые атрибуты
- Выделить эти атрибуты в отдельные таблицы
- Использовать детерминант как первичный ключ новой таблицы
- Оставить копию детерминанта в исходной таблице как внешний ключ

Пример нормализации:

Исходная таблица:

```
ЗАКАЗЫ (НомерЗаказа, ДатаЗаказа, КодКлиента, ИмяКлиента,
АдресКлиента, КодТовара, НазваниеТовара, ЦенаТовара, Количество)
```

1NF (устранение неатомарных атрибутов):

```
ЗАКАЗЫ (НомерЗаказа, ДатаЗаказа, КодКлиента, ИмяКлиента, Улица, Дом,
Квартира, КодТовара, НазваниеТовара, ЦенаТовара, Количество)
```

2NF (устранение частичных зависимостей от составного ключа {НомерЗаказа, КодТовара}):

```
ЗАКАЗЫ (НомерЗаказа, ДатаЗаказа, КодКлиента, ИмяКлиента, Улица, Дом,
Квартира)
ЗАКАЗЫ_ТОВАРЫ (НомерЗаказа, КодТовара, Количество)
ТОВАРЫ (КодТовара, НазваниеТовара, ЦенаТовара)
```

3NF (устранение транзитивных зависимостей КодКлиента → ИмяКлиента, АдресКлиента):

```
ЗАКАЗЫ (НомерЗаказа, ДатаЗаказа, КодКлиента)
ЗАКАЗЫ_ТОВАРЫ (НомерЗаказа, КодТовара, Количество)
ТОВАРЫ (КодТовара, НазваниеТовара, ЦенаТовара)
КЛИЕНТЫ (КодКлиента, ИмяКлиента, Улица, Дом, Квартира)
```