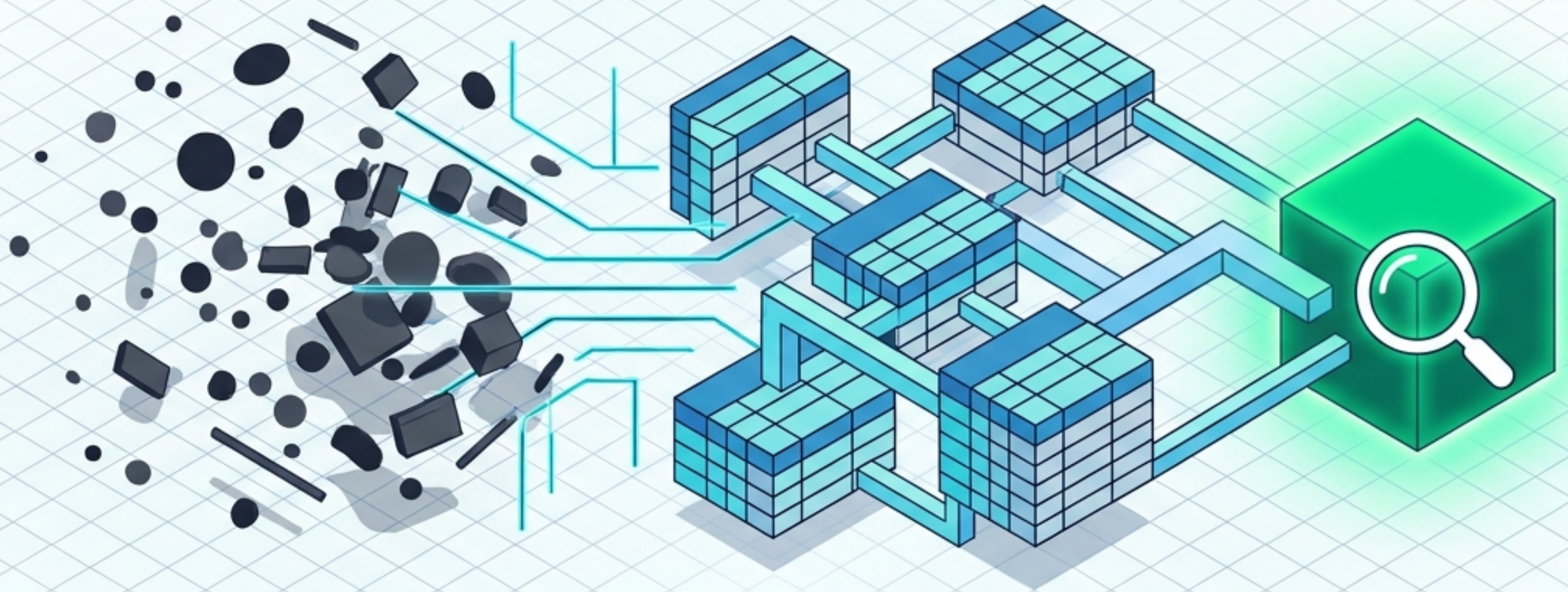


# Архитектура и Анализ Данных

От бизнес-требований до SQL-запросов: ER-моделирование, нормализация и извлечение данных.

## Проектирование (ERD)

Меню ресторана

Сущности становятся таблицами

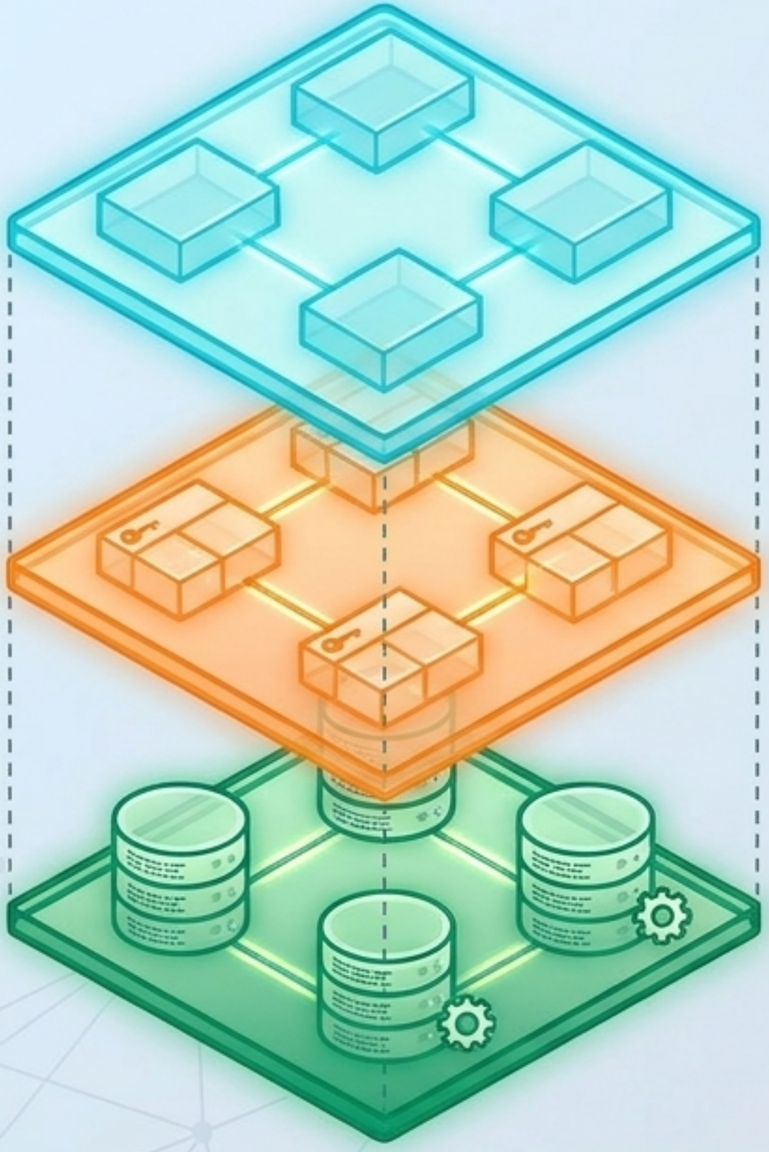
## Структура (Нормализация)

Правила бизнеса

Степень нормализации диктует количество JOIN-ов

## Извлечение (SQL)

Разговор с данными



| **Концептуальная (Бизнес)** | |
|---|---|
| Только сущности и связи (имеет, относится к). Никаких ключей. | Для: Заказчик, PO |

| **Логическая (Аналитик)** | |
|---|---|
| Полные атрибуты, PK/FK ключи, обобщённые типы (VARCHAR). | Для: Аналитик, Архитектор |

| **Физическая (DBA)** | |
|---|---|
| Точные СУБД-типы, индексы, партиции, правила ON DELETE. | Для: Разработчик, DBA |

**Золотое правило:** Не прыгайте в SQL, не выяснив бизнес-термины на концептуальном уровне.

## Сущности (Entities)

**Сильная сущность**
(Independent, has its own PK)

PK
FK

**Слабая сущность**
(Dependent, PK includes parent's FK)

PK
FK

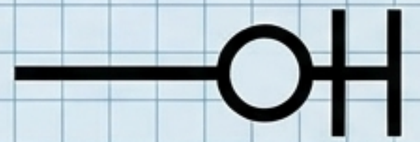
## Чтение связи


Слева направо: Один проект содержит от 1 до N задач.

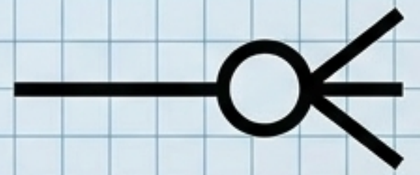
Справа налево: Каждая задача относится ровно к 1 проекту.

## Нотация «Лапка вороны» (Crow's Foot)

Ровно один (1)



Ноль или один (0..1)



Ноль или более (0..*)

Один или более (1..*)

## 1. Конфликт (Бизнес-уровень)


Реляционная база отвергает хранение списков в одной колонке.

## 2. Атомизация

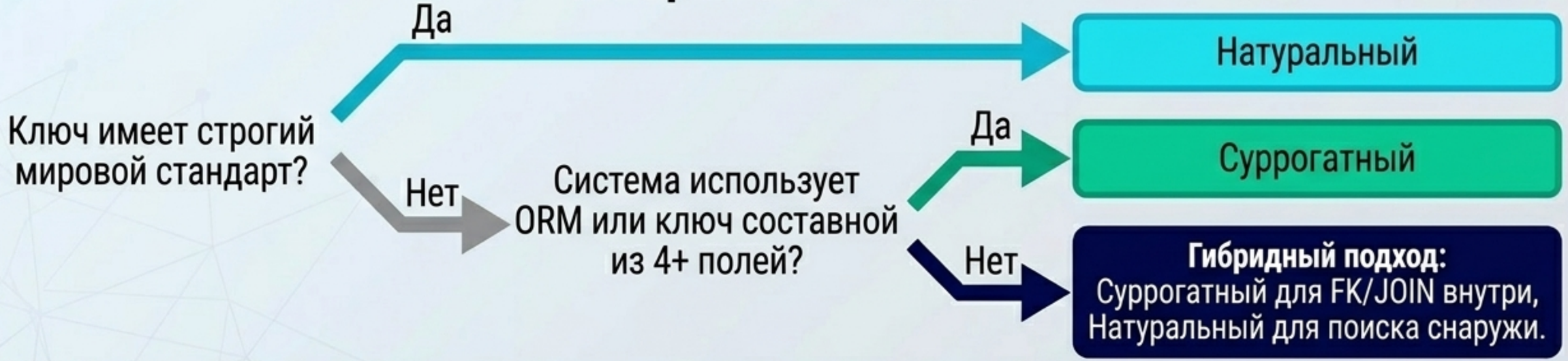

## 3. Результат (Физический уровень)


**Стратегия ключа для «Зачисления»:**

- Составной PK (student_id + course_id): Строгая гарантия уникальности без дубликатов.
- Суррогатный PK (id UUID): Если есть доп. атрибуты (дата, оценка) или используется ORM.

| Натуральный ключ (Natural Key) | Суррогатный ключ (Surrogate Key) |
| --- | --- |
| • **Примеры:** ISBN, VIN, Email, ISO-код<br>• **Плюсы:** Понятен человеку, уже существует в бизнесе.<br>• **Минусы:** Медленный JOIN (VARCHAR), может измениться. | • **Примеры:** id SERIAL, UUID<br>• **Плюсы:** Гарантированно уникален, стабилен, быстрый JOIN (INT).<br>• **Минусы:** Нет бизнес-смысла. |

## Алгоритм аналитика



## Аномалия Вставки

| Преподаватели-Курсы | | |
|---|---|---|
| teacher_id | course_id | semester |
| T101 | NUL | Fall2023 |

**Проблем:**
- Преподаватель нанят, но курс ещё не назначен.

**Бизнесс Імракт:**
- Бухгалтерия не может начислить оклад — «человека нет в системе» (Нельзя вставить NULL в PK).

## Аномалия Обновления

| Заказы Клиентов | | |
|---|---|---|
| order_id | client_name | phone |
| 1 | Клиента | +7 (495) 123-45-67 |
| 12 | Парянов | +7 (495) 123-45-67 |
| 13 | Дарья | +7 (495) 123-45-67 |
| 14 | Сярьер | +7 (495) 123-45-67 |
| 5 | Ледкливновна | +7 (916) 555-00-11 |

**Проблем:**
- Клиент сменил номер, оператор обновил только один заказ.

**Бизнесс Імраст:**
- Курьер звонит по старому номеру из старого заказа. Клиент недоволен.

## Аномалия Удаления

| Поставки-Поставщики | | | |
|---|---|---|---|
| order_id | product | supplier_id | supplier_name |
| 11 | Product 1 | 001 | Согатркра |
| 12 | Product 2 | 002 | США |
| 13 | Доскоп-потина | 013 | Bioyavile |

**Проблем:**
- Удаляем последнюю поставку снятого с производства товара.

**Бизнесс Імракт:**
- Вместе с поставкой из базы безвозвратно удалился сам поставщик.

Заказ1, КлиентА, Москва, +79991234567, Товар1, Товар2, 1500, 2500
Заказ2, КлиентБ, Казань, +79167654321, Товар3, 5000
Заказ3, КлиентА, Москва, +79991234567, Товар2, 2500

## 1НФ: Атомарность

Убираем списки в одной ячейке.
Каждое значение = отдельная строка.

## 2НФ: Частичные зависимости

Выносим атрибуты, зависящие только от части составного ключа (Справочник Товаров отдельно от Позиций Заказа).

**Заказы**

| Номер_Заказа | Клиент_ID | Клиент | Дата |
|---|---|---|---|
| 1 | КлиентА | 1 | 17.09.2023 |
| 12 | КлиентБ | 3 | 22.04.2023 |
| 13 | КлиентА | 4 | 12.09.2023 |

**Справочник Товаров**

| Товар_ID | Название | Цена |
|---|---|---|
| 1 | Товар1 | 1500 |
| 12 | Товар2 | 5000 |

## 3НФ: Транзитивные зависимости

Атрибут не должен зависеть от другого неключевого атрибута (Выносим Руководителя отдельно от сотрудника).

**Заказы**

| Номер_Заказа | Клиент_ID | Дата |
|---|---|---|
| 1 | | |

**Справочник Товаров**

| Товар_ID | Название | Цена |
|---|---|---|
| 1 | Товар1 | 1500 |

**Сотрудники**

| Сотрудник_ID | ФИО | Руководитель_ID |
|---|---|---|
| 1 | Лакуді... | 1 |

**3НФ — оптимальный стандарт для 90% систем.**

# 30%


# 70% Техники Денормализации

## Pre-joined columns

Дублирование редко меняемых данных (category_name) для избавления от JOIN.
Риск: Рассинхрон.

## Aggregates

Хранение вычисленных счётчиков (total_reviews) обновляемых триггерами.

## Materialized Views

Идеальный компромисс. Данные обновляются пачкой по расписанию.
Риск: Stale data.

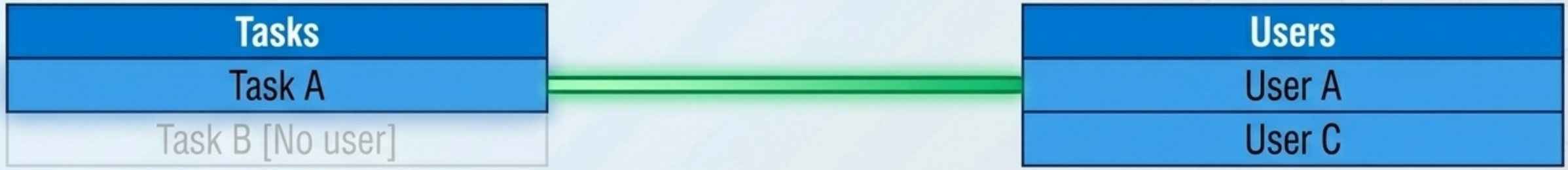
## JSONB Колонки

Хранение десятков динамических атрибутов товара без создания медленного EAV-антипаттерна.

# Логический порядок выполнения SQL

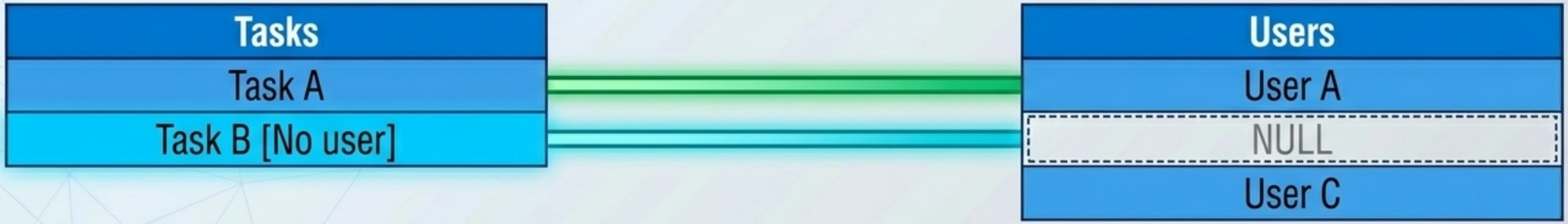
# Сравнение типов JOIN: INNER и LEFT

## INNER JOIN (Только совпадения)




Отбрасывает строки без совпадений. Задача без исполнителя исчезнет из отчёта.

## LEFT JOIN (Сохранить левую таблицу)




NULL

Сохраняет всю левую таблицу. Если совпадения нет — заполняет правые колонки пустотой (NULL).

# Ловушка LEFT JOIN: Ошибка ценою в часы отладки

## 1 Ожидание (Чистый LEFT JOIN)


Status: Safe, outline in cyan.

## 2 Ловушка (Условие в WHERE)

```
WHERE task.status = 'Active'
```

`NULL = 'Active'` → **FALSE ✕**

## 3 Удаление

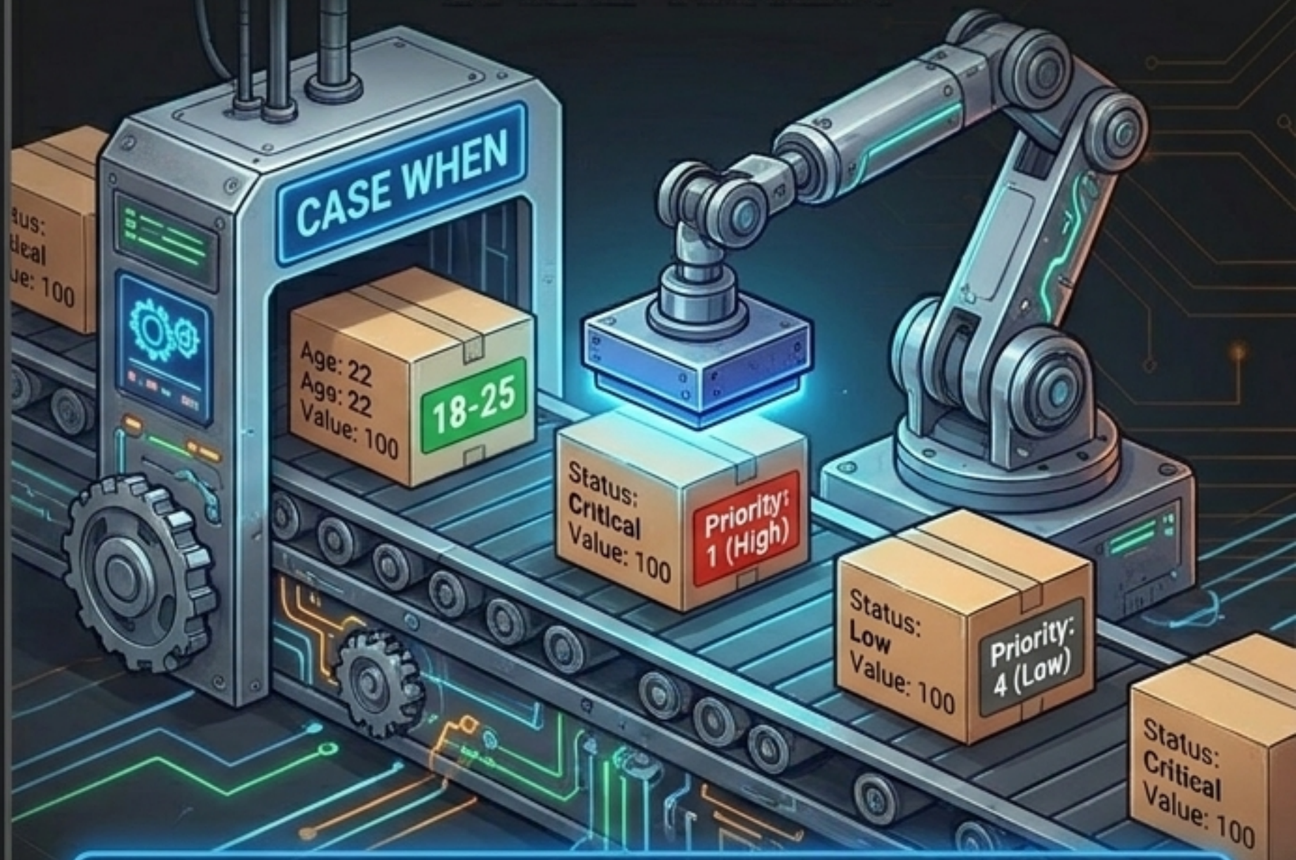

Строка физически удаляется. Ваш LEFT JOIN случайно превратился в INNER JOIN. Проект без задач исчез из отчёта.

**Как исправить:** Перенесите условия для правой таблицы из WHERE в ON (ON t.project_id = p.id AND t.status = 'Active').

# Механика GROUP BY


Все неагрегированные колонки из **SELECT** обязаны быть в **GROUP BY**!

# Сортировочная машина CASE WHEN




**Use Cases:**

1. **Бакетизация:** Группировка возрастов или чеков.
2. **Кастомная сортировка:** Обход алфавитного порядка (Critical=1, Low=4).
3. **Pivot:** Условная агрегация столбцов (SUM + CASE).

# Математика Пустоты (Danger)

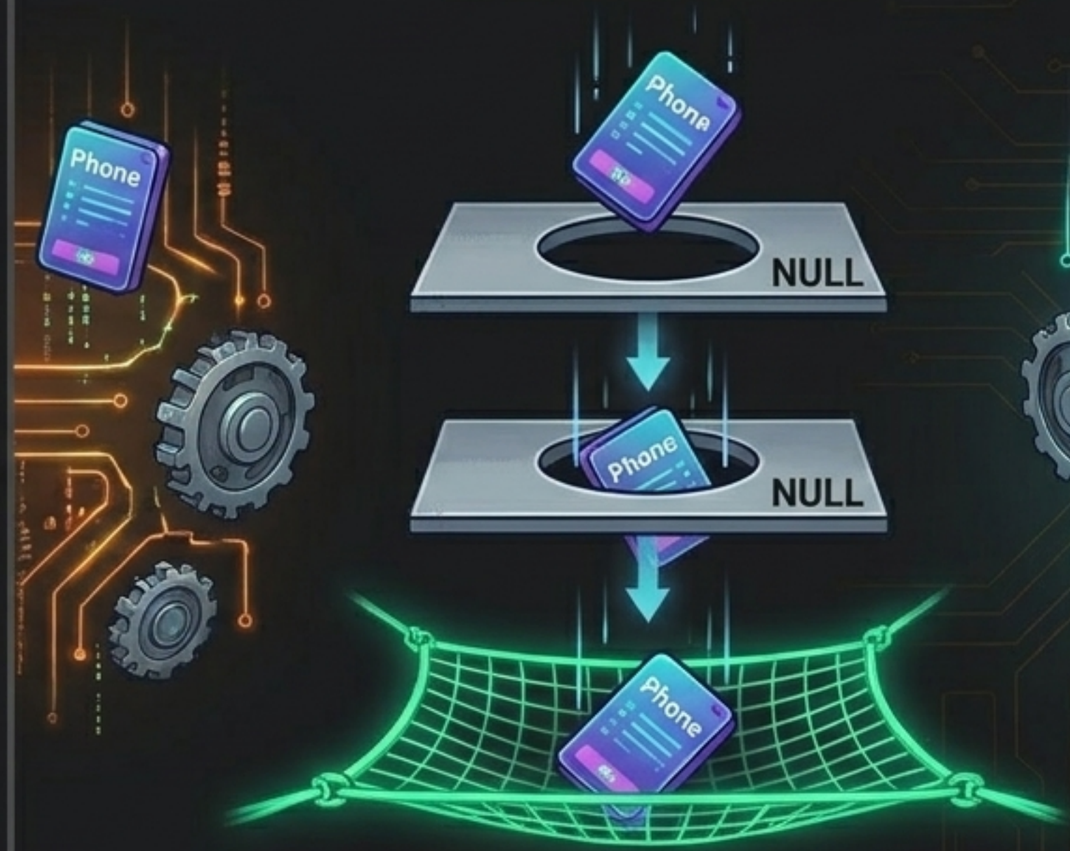
**NULL + 1 = → NULL**

**NULL = NULL → FALSE ✕**

**NULL IN (1, 2) → FALSE ✕**

**NULL** — это **отсутствие значения**.
Оно не участвует в логических сравнениях и ломает математику.

# Спасательная сетка: COALESCE




**COALESCE** (‘Нет контакта’)

Функция возвращает первое не-NULL значение.
Незаменима для защиты агрегатов:
**COALESCE**(SUM(price), 0).

# Чек-лист Аналитика

## Архитектура (ERD)

- [ ] Двигайтесь от Концептуальной модели (язык бизнеса) к Физической (SQL).
- [ ] Сущности — это существительные. Списки в одной колонке недопустимы.
- [ ] Все N:M связи требуют промежуточной таблицы (ассоциативной сущности).

## Структура (Нормализация)

- [ ] **1НФ**: Только атомарные значения.
- [ ] **2НФ/3НФ**: Избавьтесь от дублирования, вынося логические справочники.
- [ ] Денормализуйте только ради Highload производительности, осознав риски рассинхрона.

## Извлечение (SQL)

- [ ] Пишите запрос с оглядкой на логический порядок выполнения (FROM -> JOIN -> WHERE).
- [ ] Никогда не фильтруйте правую таблицу LEFT JOIN в WHERE. Переносите в ON.
- [ ] Оборачивайте опасные метрики в **COALESCE** для защиты от пустоты.