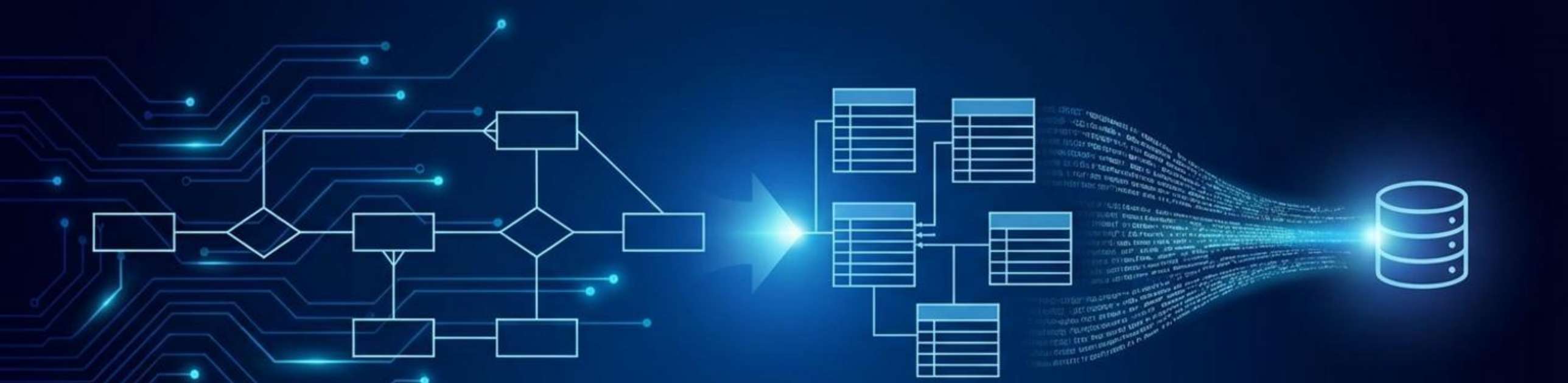


РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ

От концептуального дизайна к SQL

Неделя 2 | Лекция 2

Что такое Entity-Relationship Model?

Определение: ER-модель (Петер Чен, 1976) — это «способ рисовать структуру данных человеческим языком».

БИЗНЕС
(Аналитик)

«Один пользователь может любить много фильмов».

Универсальный язык между бизнесом и разработкой

Любит (1:*)

РАЗРАБОТКА
(Инженер)

```
CREATE TABLE films (
 .., user_id INT
 REFERENCES users(id));
```

Цель: Показать, что ER — это универсальный язык общения.

ER-нотации: Нотация Чена (концептуальная)

🎯 **Нотация Чена**: “Описываем идею”. Цель — научить читать базовые диаграммы и понимать кардинальность.

Элементы:

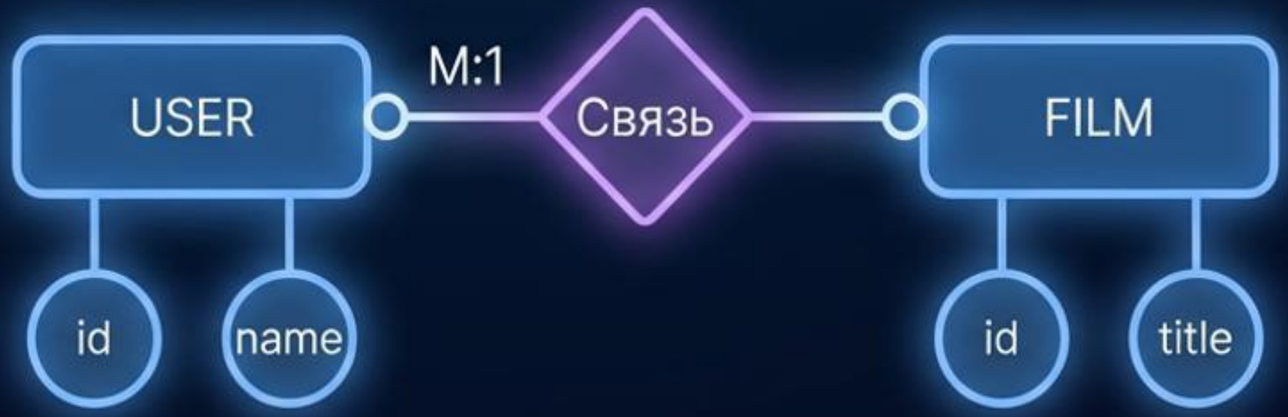
= Сущность

= Связь

= Атрибут сущности

1, *, M, N = Кардинальность

Пример:



Один USER (много) связан с FILM (один)

Нотация Crow’s Foot (логическая)

Символы:

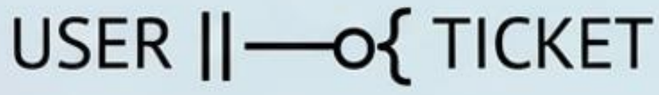
= Ровно один (1)

= Ноль или один (0..1)

= Много (*)

= Ноль или много (0..*)

Пример:



Пояснение: лапки воробья показывают обязательность и кратность.
Цель — перевести студента с концептуального на логический уровень.

Шпаргалка по нотации Crow's Foot

Одна сторона (One Side)	Другая сторона (Other Side)
`||` Ровно один (Exactly One)	`}|` Много (Many)
`|o` Ноль или один (Zero or One)	`}o` Ноль или много (Zero or Many)

Примеры связей:

`User ||—<o{ Order`

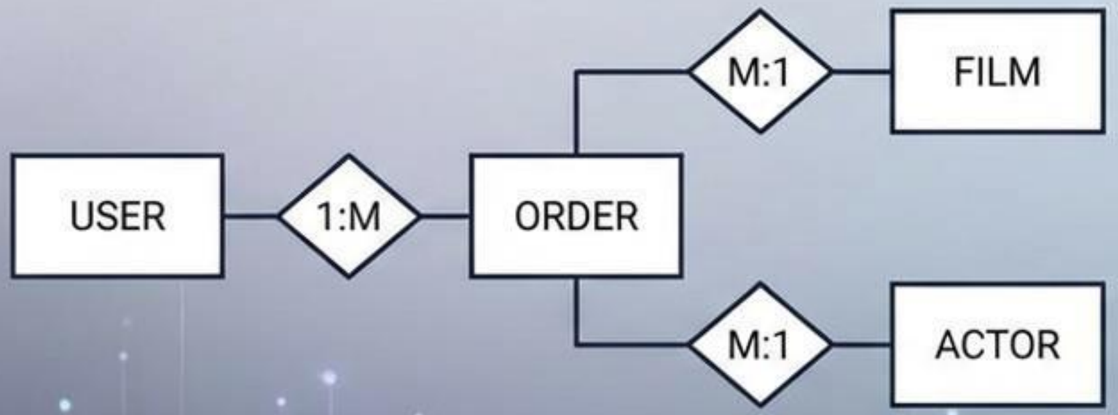
(один юзер, ноль или много заказов)

`Order ||—|| Invoice`

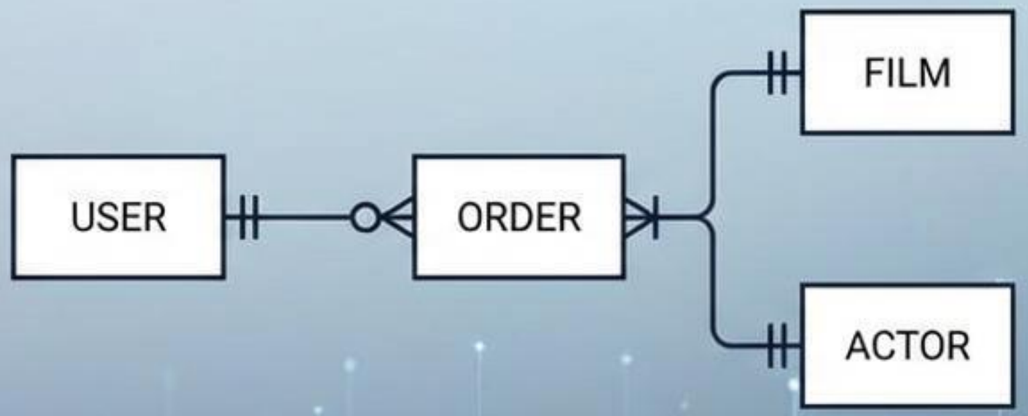
(один заказ, ровно один счет)

СРАВНЕНИЕ ER-НОТАЦИЙ: КИНОТЕАТР

Нотация Чена



Нотация Crow's Foot

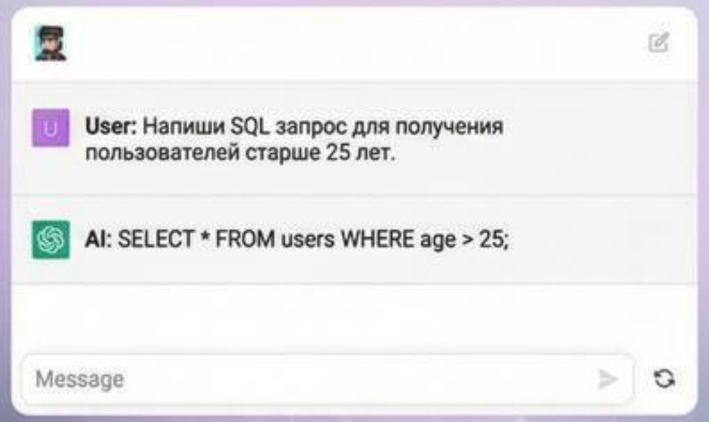


Вывод: одна предметная область — два способа записи, оба полезны.

ИИ меняет разработку БД

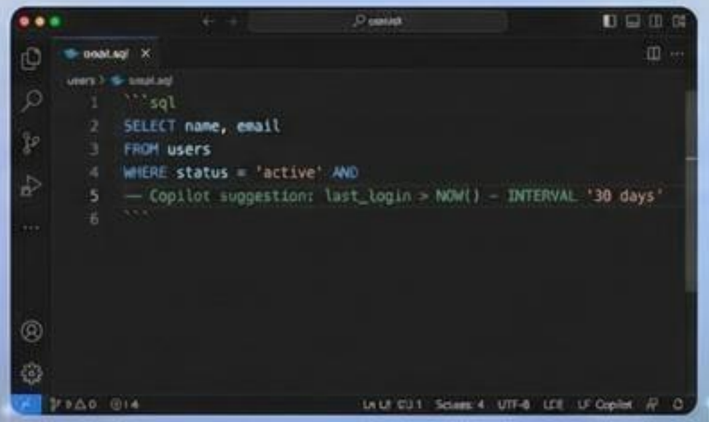
От чат-ботов к генерации схем: Эволюция инструментов

2023: ChatGPT



Генерация кода по запросу (chat)

2024: GitHub Copilot для SQL



Инлайн-подсказки в IDE (code inline)

2025: Целые схемы из промтов

Генерация полной схемы (full schema)

Цель: Автоматизация рутинных ER-задач возможна, но требует качественного промта для точного результата.

domain.md: Техническое ТЗ для ИИ-ассистента

domain.md

```
# System Name: Кинотеатр

## Entities
User[1]————*Ticket
Ticket————*Film
Film————*Actor

## Business Rules
- Ticket.price must be greater than 0
- User can reserve up to 5 tickets
- Film duration is in minutes

## Legacy Queries
-- Get popular films
SELECT f.title, COUNT(t.id) AS tickets_sold
FROM Film f
JOIN Ticket t ON f.id = t.film_id
GROUP BY f.title
ORDER BY tickets_sold DESC;
```

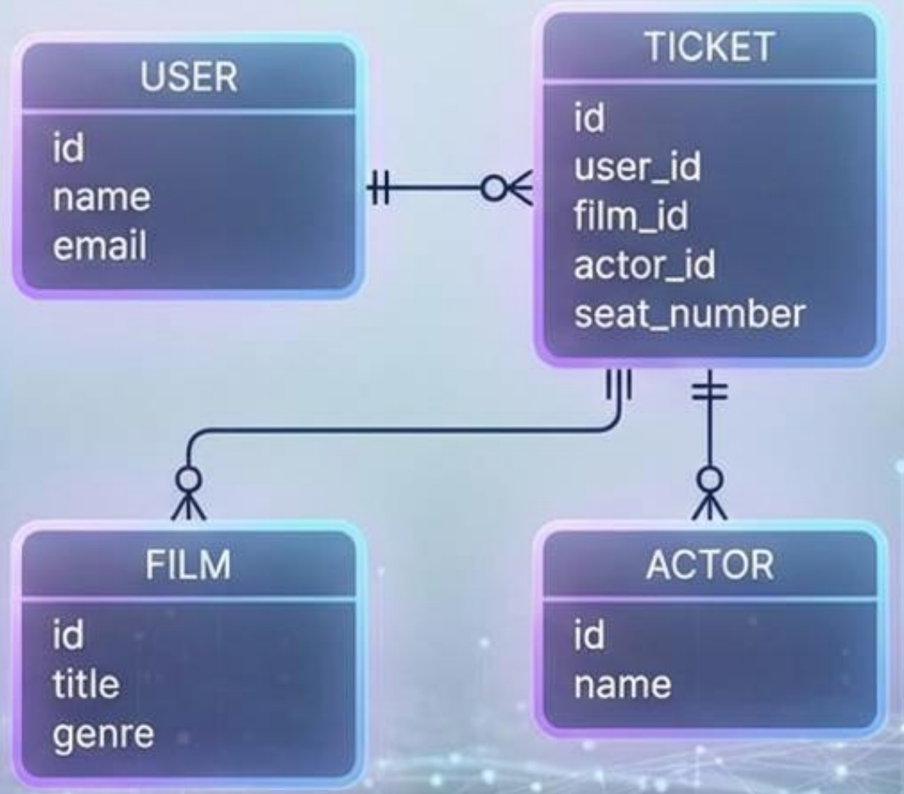
Цель: Хорошее ТЗ = половина работы ИИ-ассистента

domain.md (VS Code) | ИИ-промт

domain.Store > JetBrains Mono

```
1   # Movie Store
2
3   ## Entities
4   User[1]—*Ticket
5   Ticket——*Film
6   Film———*Actor
7
```

ER (draw.io) | Визуальная модель



ИИ превращает текстовое ТЗ в визуальную модель для обсуждения.

ПОЧЕМУ РЕЛЯЦИОННАЯ МОДЕЛЬ ДОМИНИРУЕТ?

1. НАДЕЖНОСТЬ

ACID-гарантии, целостность данных

2.СТАНДАРТИЗАЦИЯ

SQL везде работает одинаково

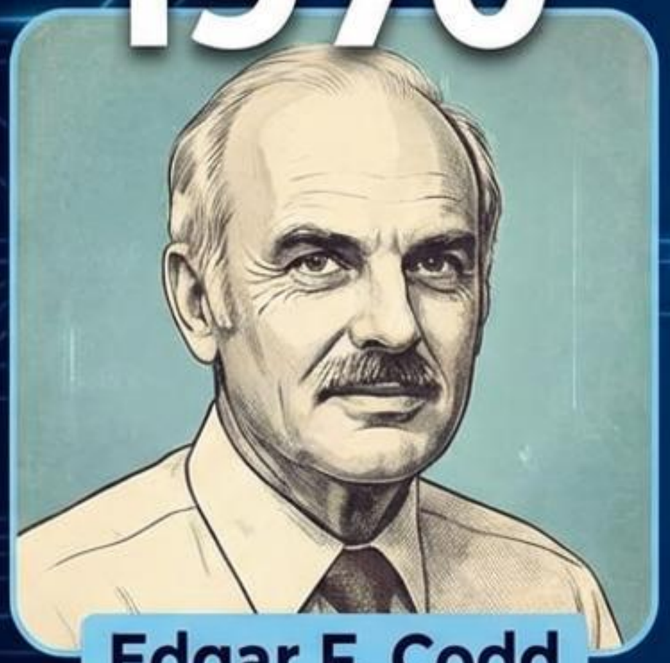
3. ЭФФЕКТИВНОСТЬ

Оптимизаторы запросов работают хорошо

4. М ТАБИРУЕМОСТЬ

Справляется с миллиардами строк

1970



Edgar F. Codd

A Relational Model of Data
for Large Shared Data Banks

Революционная идея:

- Данные как таблицы
- Теория множеств
- Универсальный язык запросов (SQL)

Результат: Стандарт де-факто для 50+ лет

1970 — 1980 — 1990 — 2010 — 2010 — 2020s+

ЭВОЛЮЦИЯ ПОСЛЕ КОДДА: ОТ SQL К ИИ

1975

SQL-72: Стандартизация языка запросов

1990s

ORACLE, IBM DB2: Корпоративные СУБД, масштабирование

2000s

POSTGRESQL, MYSQL: Открытый исходный код, веб-приложения

2020s+

POSTGRESQL+AI: Распределенные системы, ИИ-интеграция, Big Data

Идеи Кодда масштабируются, адаптируясь к современным вызовам данных и искусственного интеллекта.

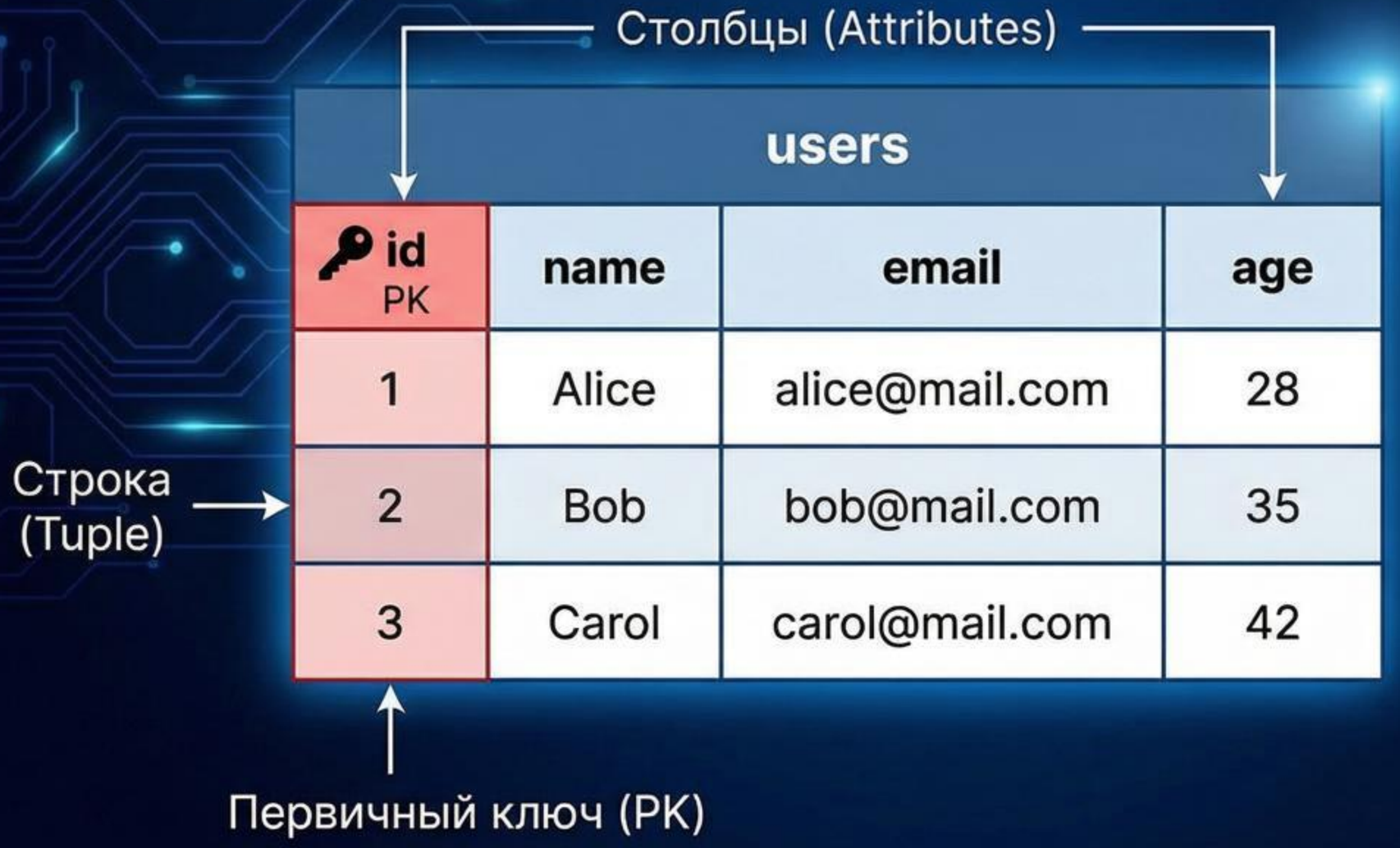
Неделя 2 | Лекция 2

ПОЧЕМУ POSTGRESQL ДЛЯ КУРСА

- ✅ **ACID:** Надежно (банки используют)
- ✅ **ACID:** Standard SQL (везде работает)
- ✅ **Extension:** pg_vector, json, hstore
- ✅ **Open source:** Скачать → запустить
- ✅ **Jobs:** 40% вакансий требуют Postgres

Все выучим. Курс на PostgreSQL → принципы везде.

ТАБЛИЦА = ОТНОШЕНИЕ (RELATION)



✅ **Правила:**
Каждый столбец имеет **ТИП** (INTEGER, TEXT, DATE)

✅ Все строки имеют **ОДИНАКОВУЮ** структуру

🔑 Строки **УНИКАЛЬНЫ** (no duplicates)

СТРУКТУРА ТАБЛИЦЫ: ОПРЕДЕЛЕНИЕ И SQL

АБСТРАКТНОЕ ОПРЕДЕЛЕНИЕ

Имя таблицы: users	
Атрибут (Столбец)	Тип данных
id	INTEGER
name	VARCHAR(100)
email	VARCHAR(255)
created_at	DATE

Атрибуты с типами

РЕАЛИЗАЦИЯ В SQL

```
CREATE TABLE users (
    id INTEGER PRIMARY KEY, -- Уникальный ID
    name VARCHAR(100) NOT NULL,
    email VARCHAR(255) UNIQUE,
    created_at DATE DEFAULT CURRENT_DATE
);
```

ПРАВИЛА:

- Каждый столбец имеет **ТИП** (INTEGER, TEXT, DATE)
- Все строки имеют **ОДИНАКОВУЮ** структуру
- Строки **УНИКАЛЬНЫ** (без дубликатов)

СЛАЙД 8: Пример — таблица Заказов

Практический SQL и данные

SQL Код (DDL)

SERIAL: Автоинкремент

Уникальный ключ

```
CREATE TABLE orders (
  id SERIAL PRIMARY KEY,          -- Уникальный ID
  customer_id INTEGER NOT NULL,   -- ID клиента (FK)
  order_date DATE NOT NULL,       -- Дата заказа
  total_amount DECIMAL(10, 2),    -- Сумма с 2 знаками
  status VARCHAR(20) DEFAULT 'pending' -- Статус
);
```

Точность валюты

Значение по умолчанию

Пример Данных

id	cust_id	order_date	total	status
1	101	2025-01-10	150.50	pending
2	102	2025-01-11	250.00	shipped

Автоматически сгенерированные ID

Значение по умолчанию или измененное

ШПАРГАЛКА ПО ТИПАМ ДАННЫХ

От концептуального дизайна к SQL

ТИП ДАННЫХ	ОПИСАНИЕ/РАЗМЕР	ПРИМЕРЫ
INTEGER	Целое число (4 байта), диапазон до 2 млрд	100, -5, 42
BIGINT	Большое целое (8 байт), огромный диапазон	123456789012345
VARCHAR(n)	Текст переменной длины, до n символов	'Москва', 'Иван Иванов'
TEXT	Текст неограниченной длины	Большая статья...
DATE	Дата (год, месяц, день)	'2023-10-27'
TIMESTAMP	Дата и время с точностью до долей секунд	'2023-10-27 14:30:00'
BOOLEAN	Логическое значение (TRUE/FALSE/NULL)	TRUE, FALSE
DECIMAL	Число с фиксированной точностью, для денег	123.45, 0.99

ПЛОХО: Все в VARCHAR

age (VARCHAR)	price (VARCHAR)
'28'	'99.99'
'thirty'	'abc' ← ошибка!

Данные некорректны, СУБД пропустит

Типы помогают предотвращать ошибки

ХОРОШО: Правильные типы

age (INTEGER)	price (DECIMAL)
28	99.99
ERROR!	ERROR! ← защита!

СУБД блокирует некорректные данные

Типы данных — первый уровень валидации на уровне СУБД.

Экономия и производительность

Влияние типов данных на размер и скорость

Все-VARCHAR (Неправильно) ❌

TABLE
Tfottttrershoridadeddtinstehornstennapvoanrtzionsstringgo...
Tiotwisertidankaversenthashtsbsotticomsitsteangotringang...
Tfohtisersidensafenuwshownmicatacwrsnc tslangsstrimgo...
Tfotwisertidankaversenthashtsbsotticomsitsteangotrmgang...
Tfoicwsertidancmisemkshcrrnridosldkkveint tslangsstrimgo...
Tintwisertidanicvarsenthochtsbsattcomsitsteangotringang...

Размер: **8 ГБ**

Скорость поиска: **Медленно**

Правильный выбор типа экономит ресурсы

и ускоряет работу

Правильные типы (Правильно) ✅

INTEGER	DECIMAL	DATE	VARCHAR
1	20.00	03/09/2022	monta
2	54.00	03/09/2022	john
3	80.95	03/09/2022	varchar
4	20.00	03/09/2022	mary
5	3.00	03/09/2022	hissin
6	30.00	03/09/2022	linn

Размер: **1 ГБ** (8x экономия!)

Скорость поиска: **Быстро** (Индексы работают)

Выбор типа — это вопрос корректности, экономии ресурсов и производительности, что критично для высоконагруженных систем.

Практический шаблон: CREATE TABLE

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,                  -- INTEGER PK, автоинкремент
    username VARCHAR(50) NOT NULL,          -- NOT NULL, обязательное поле
    email VARCHAR(100) UNIQUE NOT NULL,-- UNIQUE, уникальный адрес
    birth_date DATE,                        -- DATE, дата рождения
    is_active BOOLEAN DEFAULT TRUE,         -- BOOLEAN DEFAULT, значение по умолчанию
    balance DECIMAL(10, 2)                  -- DECIMAL(10,2), денежный формат
);
```

Ключевые ограничения:

- 🔑 PRIMARY KEY: Уникальный идентификатор
- ⚠️ NOT NULL: Обязательное значение
- ⭐ UNIQUE: Отсутствие дубликатов