КСиТ

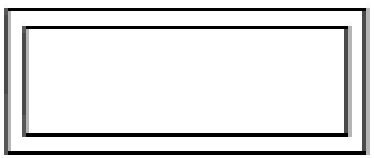
2. Реляционная модель

Базы данных

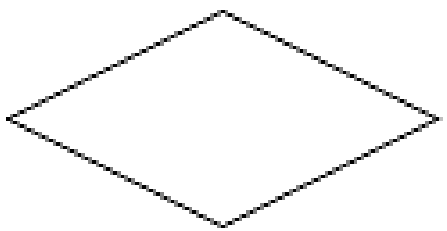
Хисамутдинов М.А.
кафедра №12 НИЯУ МИФИ
2026

Диаграмма сущность связь

Регулярное отношение сущности



Слабое отношение сущности



Связь

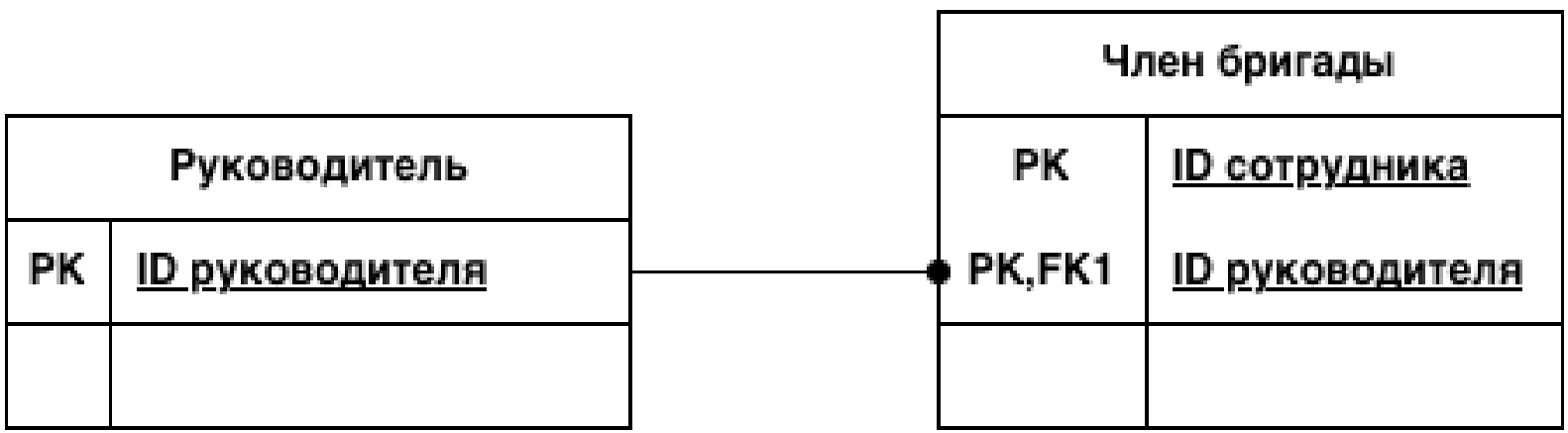
Атрибут

Диаграмма сущность связь

Слабое отноешние сущности

Диаграмма сущность связь

Слабое отноешние сущности



Реляционная модель данных

Реляционная модель данных - это способ организации данных, при котором информация хранится в виде таблиц (отношений), связанных между собой по ключам.

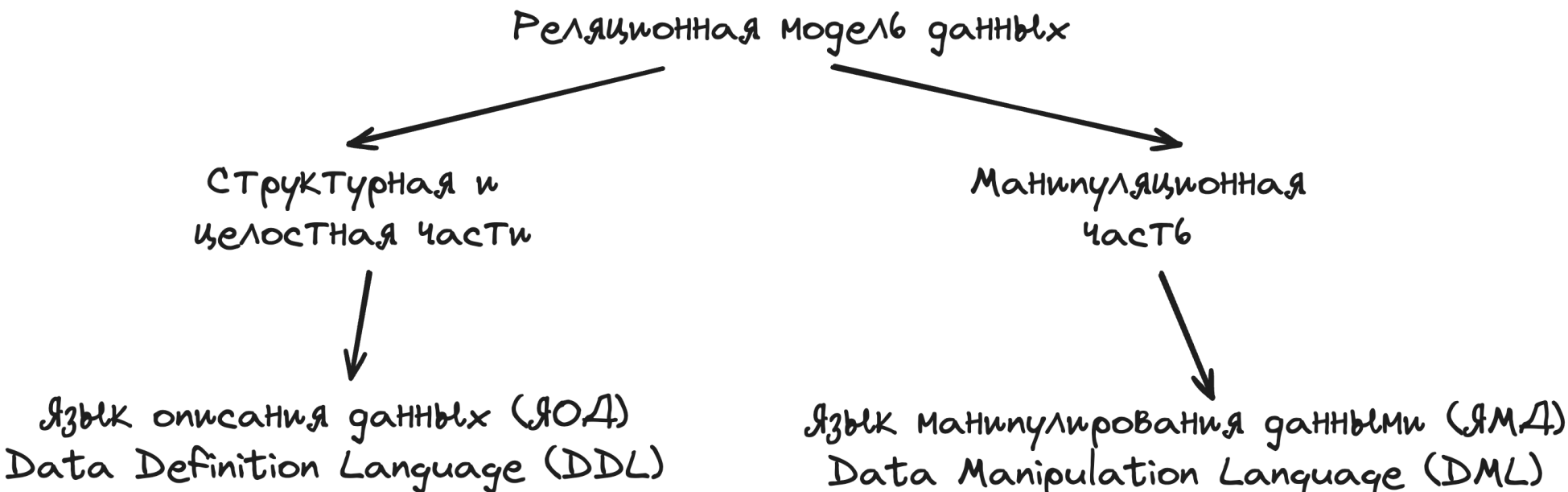
Достоинства:

- небольшой набор абстракций
- мощный математический аппарат
- ненавигационное манипулирование данными

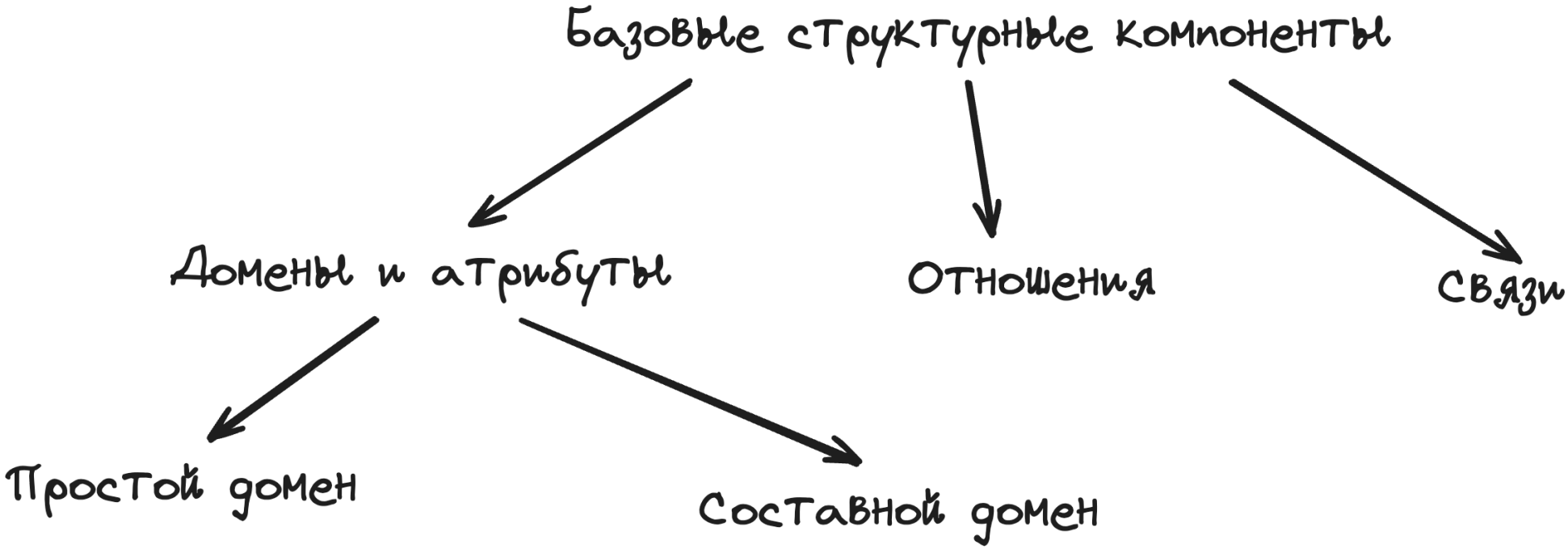
Недостатки:

- ограниченность при использовании в некоторых областях применения
- невозможность адекватного отображения семантики предметной области

Общая характеристика



Структурная часть



ГОД = {1985, 2003, 2000}
ДЕНЬГИ = {500, 1000, 850}

ИСТОРИЯ ЗАРПЛАТЫ =
{{<1985, 500>, <2000, 1000>},
{<2000, 850>}, {<1985, 850>,
<2000, 500>, <2003, 1000>}}

Отношение

Отношение - это множество кортежей одинаковой структуры.

Или чуть менее формально: это таблица в реляционной БД.

Атрибут - это свойство сущности, представленное в таблице (столбец).

Каждый атрибут связан с доменом - множеством допустимых значений.

Схема отношения - именованная совокупность пар $<$ имя атрибута : имя домена $>$

Пример:
Домены: ЧИСЛО, СТРОКА
Схема отношения ОТДЕЛ: ОТДЕЛ(Номер отдела: ЧИСЛО, Название: СТРОКА)

Отношение

Свойства отношения:

- кортежи отношения не упорядочены
- домены внутри кортежей упорядочены

Свойства отношения РМД:

- атрибут отношения уникален
- атрибут определен на домене
- на одном домене - несколько атрибутов
- имя атрибута может совпадать с именем домена
- каждый кортеж уникален
- порядок следования кортежей не устанавливается
- имя отношения уникально

Представление сущности

Ключ - совокупность атрибутов, однозначно идентифицирующих каждый кортеж отношения.

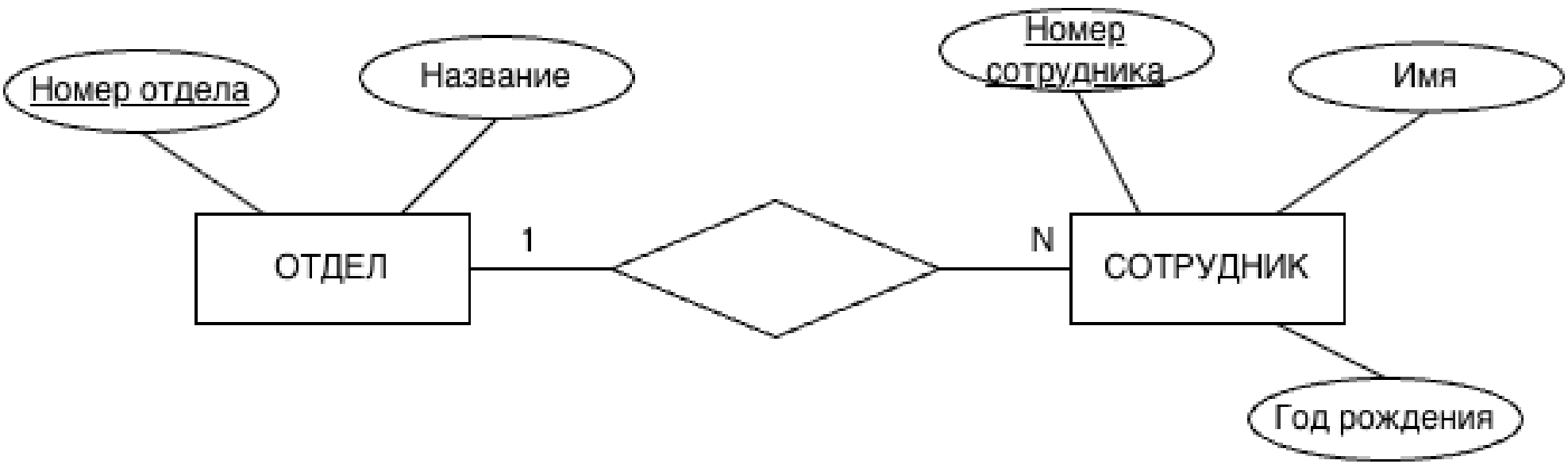
- **Первичный ключ** (PK - Primary Key) - неизбыточный набор атрибутов
  - уникальность
  - неприводимость (невозможно выбросить ни один атрибут, не потеряв уникальность)
- **Альтернативный ключ** (AK - Alternate Key)

Пример:
КАФЕДРА (Номер кафедры, Название (АК))

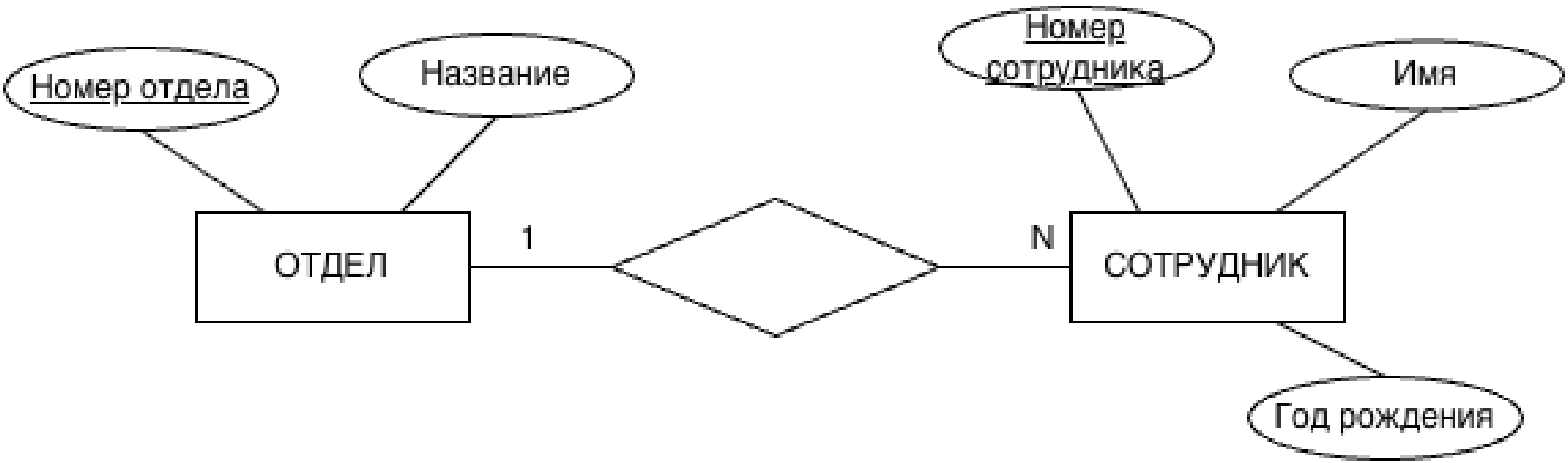
Представление связи

Внешний ключ (FK - Foreign Key) - это атрибут или некоторое множество атрибутов отношения R1, которые не являются собственными атрибутами отношения R1, но их значение совпадает со значениями первичного ключа некоторого отношения R2 (возможность идентичности R1 и R2 не исключается).

Представление связи 1:n



Представление связи 1:n



В частных случаях может появляться связь 1:1

Представление связи 1:n

Отношения:

ОТДЕЛ(Номер отдела, Название (АК))

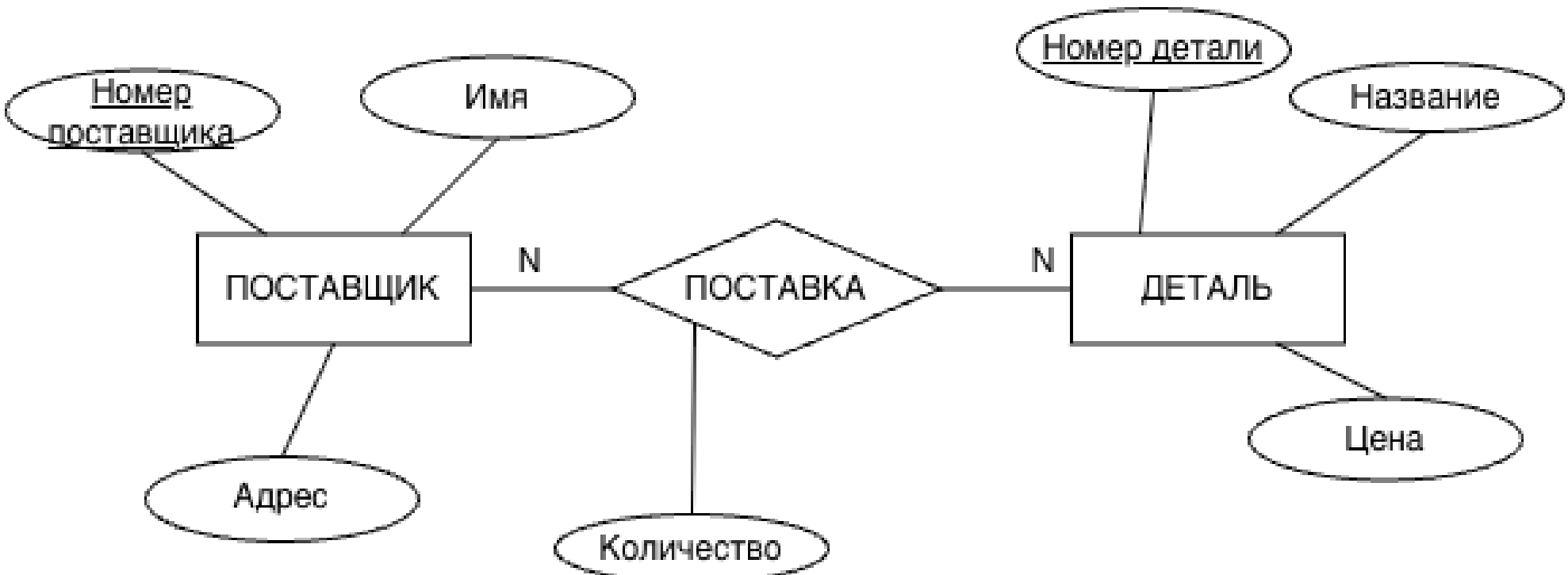
СОТРУДНИК(Номер сотрудника, Имя, Год рождения, Номер отдела (FK))

Номер отдела	**Название**
Первичный ключ	*Альтернативный ключ*
1	Бухгалтерия
2	АХО
3	Библиотека

Номер сотрудника	**Имя**	**Год рождения**	**Номер отдела**
Первичный ключ			*Внешний ключ*
1	Иванов	1953	1
2	Петров	1970	3

Представление связи 1:n

Представление связи n:n



Представление связи n:n

ПОСТАВЩИК(Номер поставщика, Имя, Адрес)

ДЕТАЛЬ(Номер детали, Название, Цена)

№ поставщика	Имя	Адрес
PK		
S1	Иванов	Москва
S2	Сидоров	С. Петербург
S3	Петров	Тюмень

№ детали	Название	Цена
PK		
P1	Болт	18
P2	Винт	14
P3	Гайка	10

Представление связи n:n

Связь ПОСТАВЩИК - ДЕТАЛЬ

ПОСТАВКА(Номер поставщика (FK1), Номер детали (FK2), Количество)

Первичный ключ отношения связи		Количество
Номер поставщика	*Номер детали*	
Внешний ключ отношения ПОСТАВЩИК	Внешний ключ отношения ДЕТАЛЬ	Собственный атрибут связи
S1	P1	100
S1	P2	200
S2	P3	150

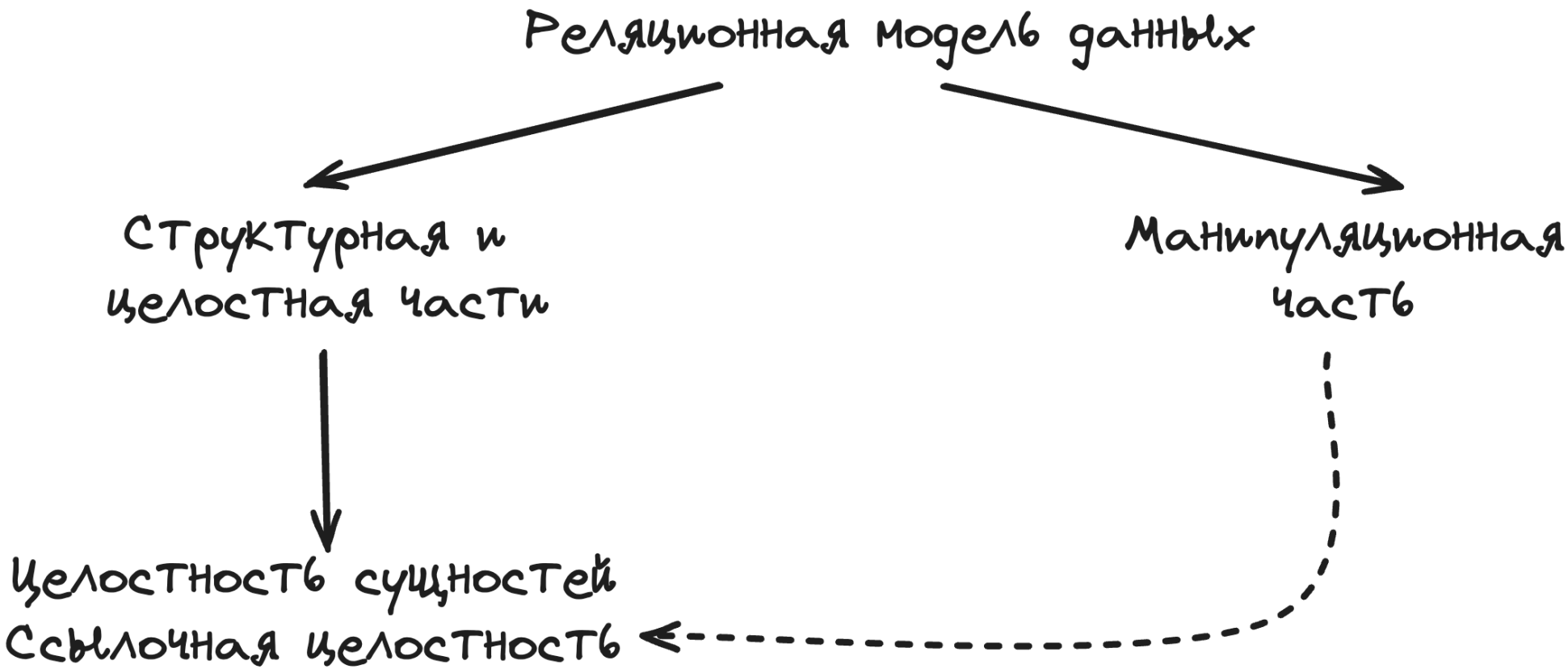
Представление связи n:n

неопределенная связь

Представление связи n:n

преобразование в определенную связь

Ограничения целостности



Ограничения целостности

Целостность сущностей:

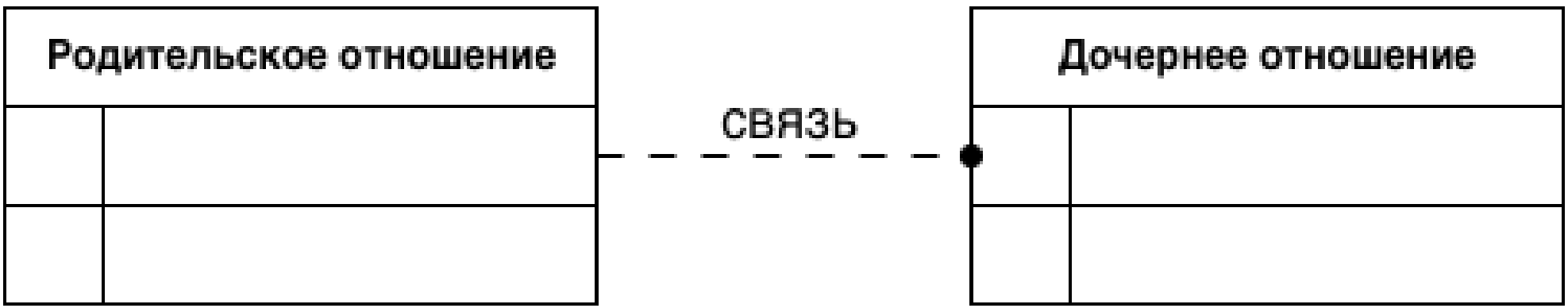
- ограничение первичного ключа (PK)
- уникальность других атрибутов (AK)
- обязательность значений атрибутов (NULL, NOT NULL)
- допустимость значений атрибутов (CHECK)

Ограничения целостности

Ссылочная целостность:

ОТДЕЛ(Номер отдела, Название(АК))

СОТРУДНИК(Номер сотрудника, Имя, Год рождения, Номер отдела (FK))



- вставка
- удаление
- модификация PK

- вставка
- удаление
- модификация FK

Ссылочная целостность

Операции с дочерним отношением:

- Вставка - корректное значение FK
- Удаление - без ограничений
- Модификация FK - корректное новое значение FK

Операции с родительским отношением:

- Вставка - без ограничений
- Удаление - реакция на связанные записи:
  - не удалять
  - удалить все
  - установить пустое значение (отсутствие связи)
- Модификация PK - реакция аналогична

Сущность

1. Название сущности - существительное в единственном числе
2. Две области: первичного ключа и прочие атрибуты
3. Дополнительные ключи определяются как альтернативные

Название сущности	
PK	**Область первичного ключа**
	Область прочих атрибутов

Сущность

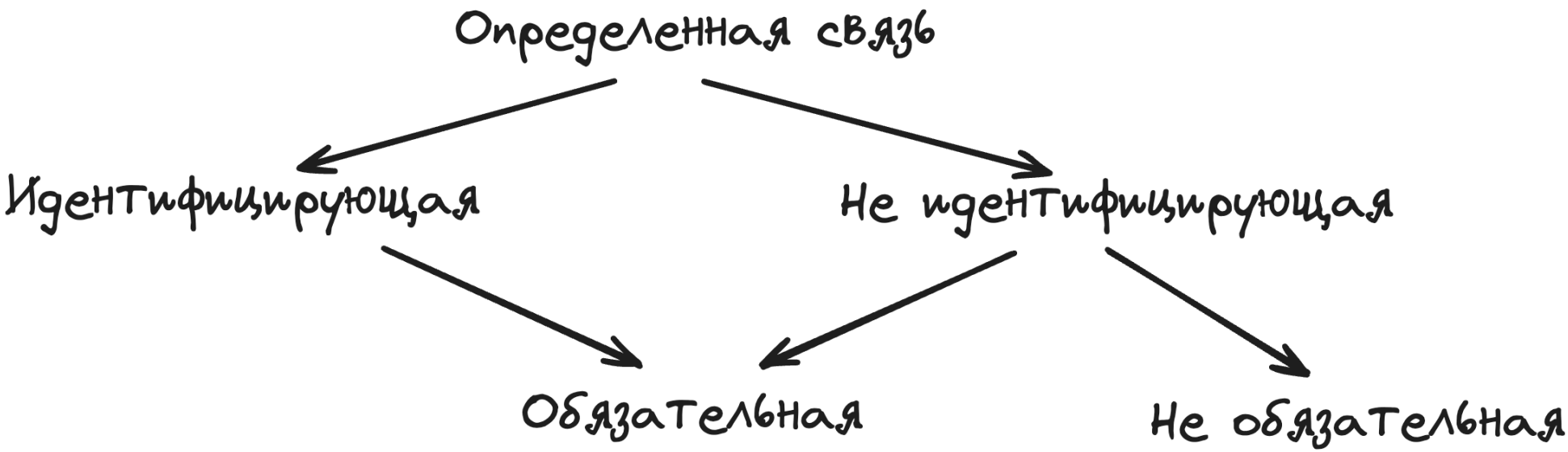
Сущность - Кафедра/E1
Уникальный атрибут: Название кафедры (например, кафедра Кибернетики, или кафедра Компьютерные системы и технологии)
Суррогатный первичный ключ - Номер кафедры;
Название - альтернативный ключ

Кафедра/E1	
PK	**Номер кафедры**
AK1.1	Название кафедры

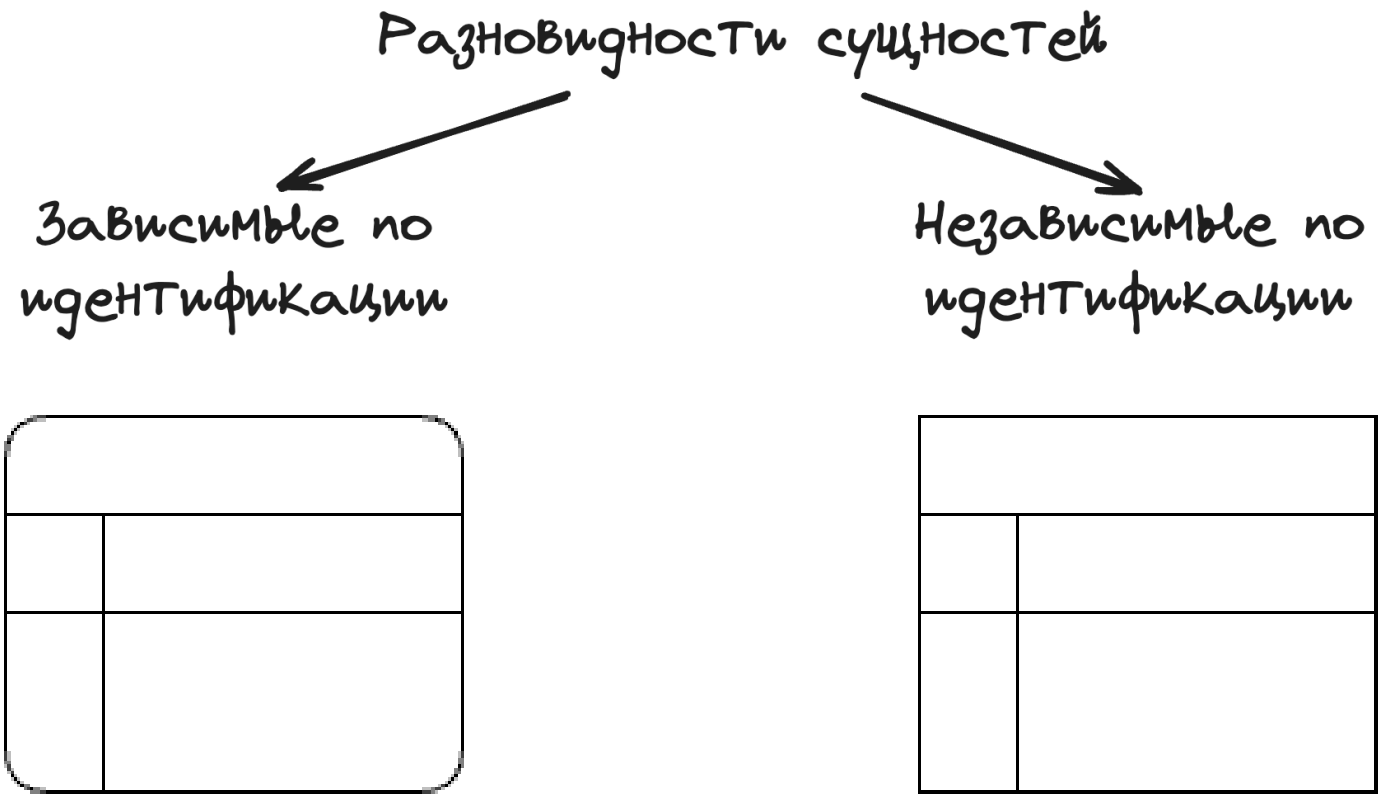
Определенная связь

ER-диаграмма:

Определенная связь

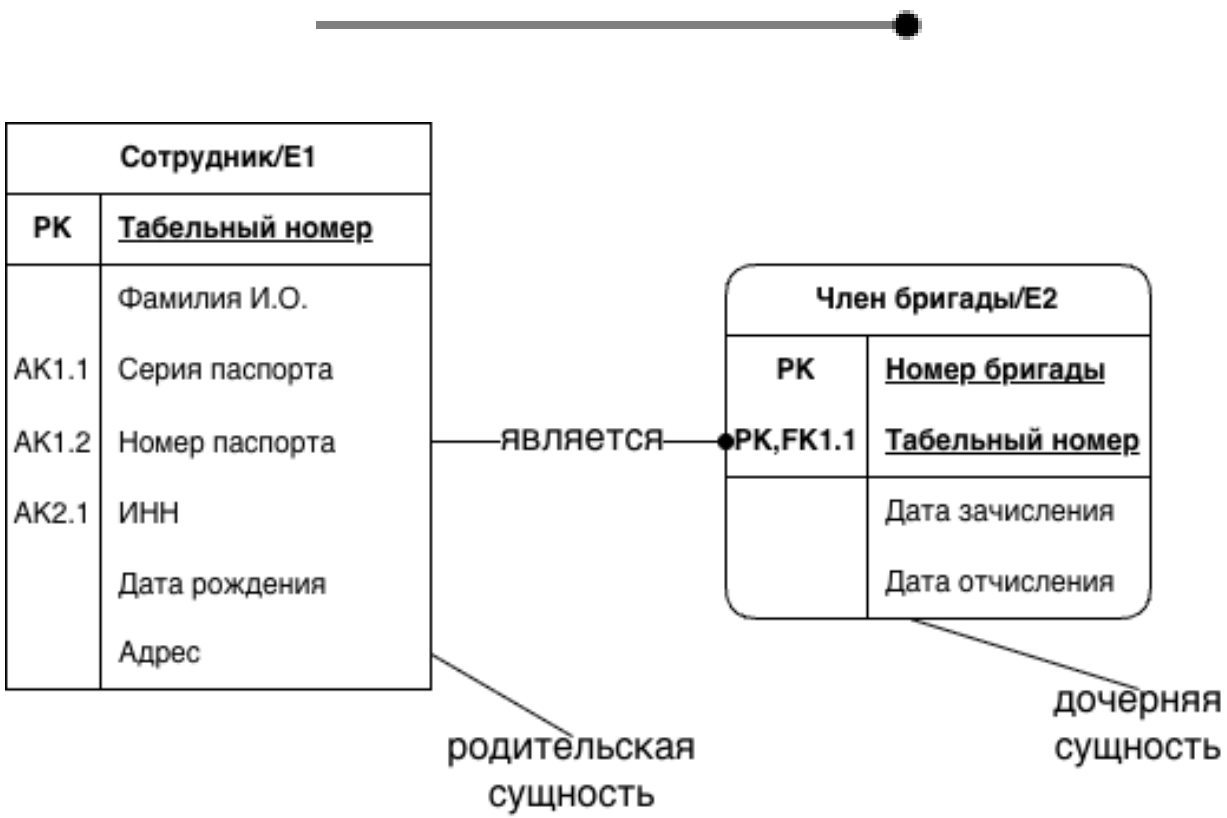


Разновидности сущностей



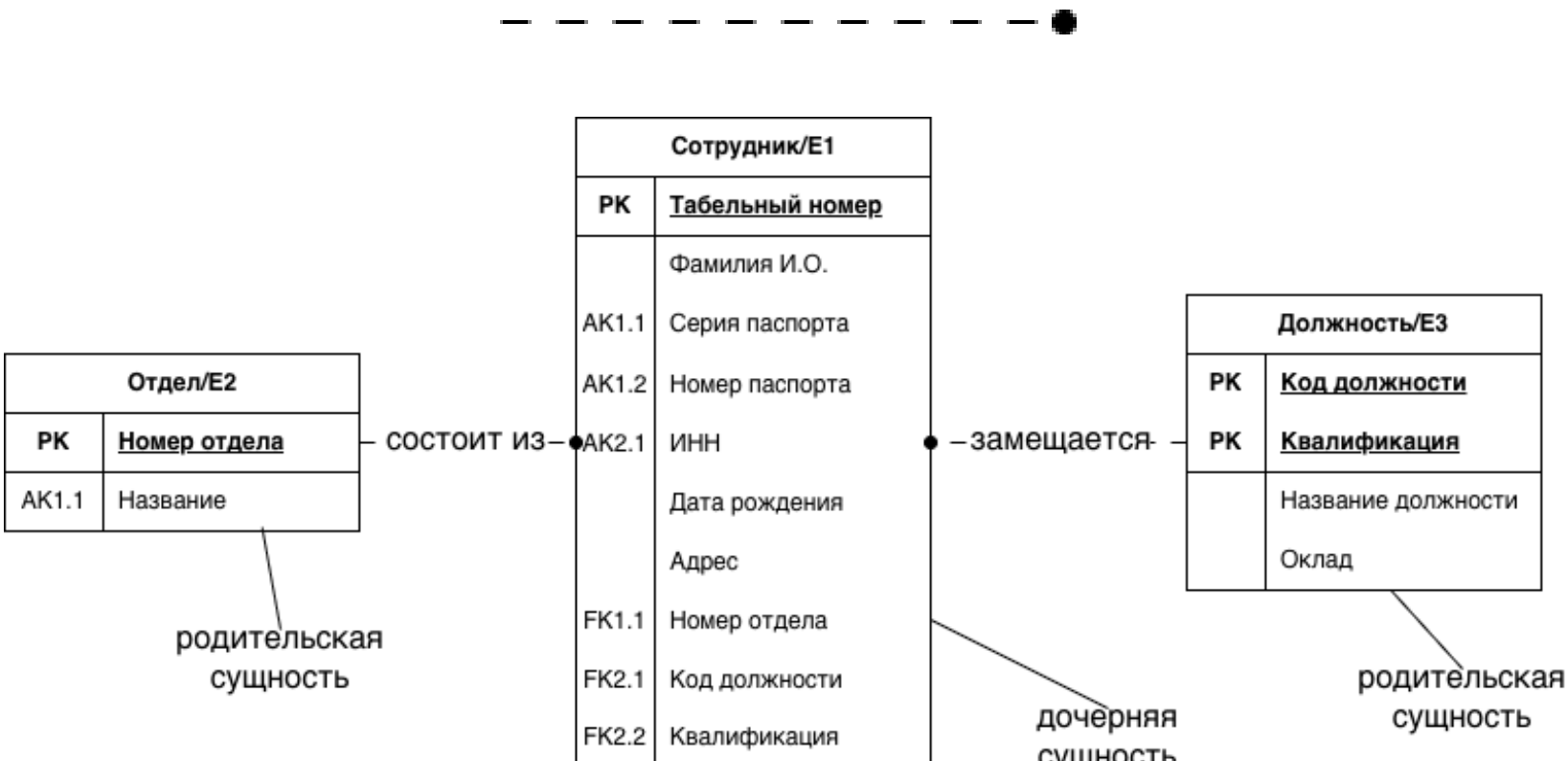
Определенная связь

Определенная идентифицирующая обязательная связь



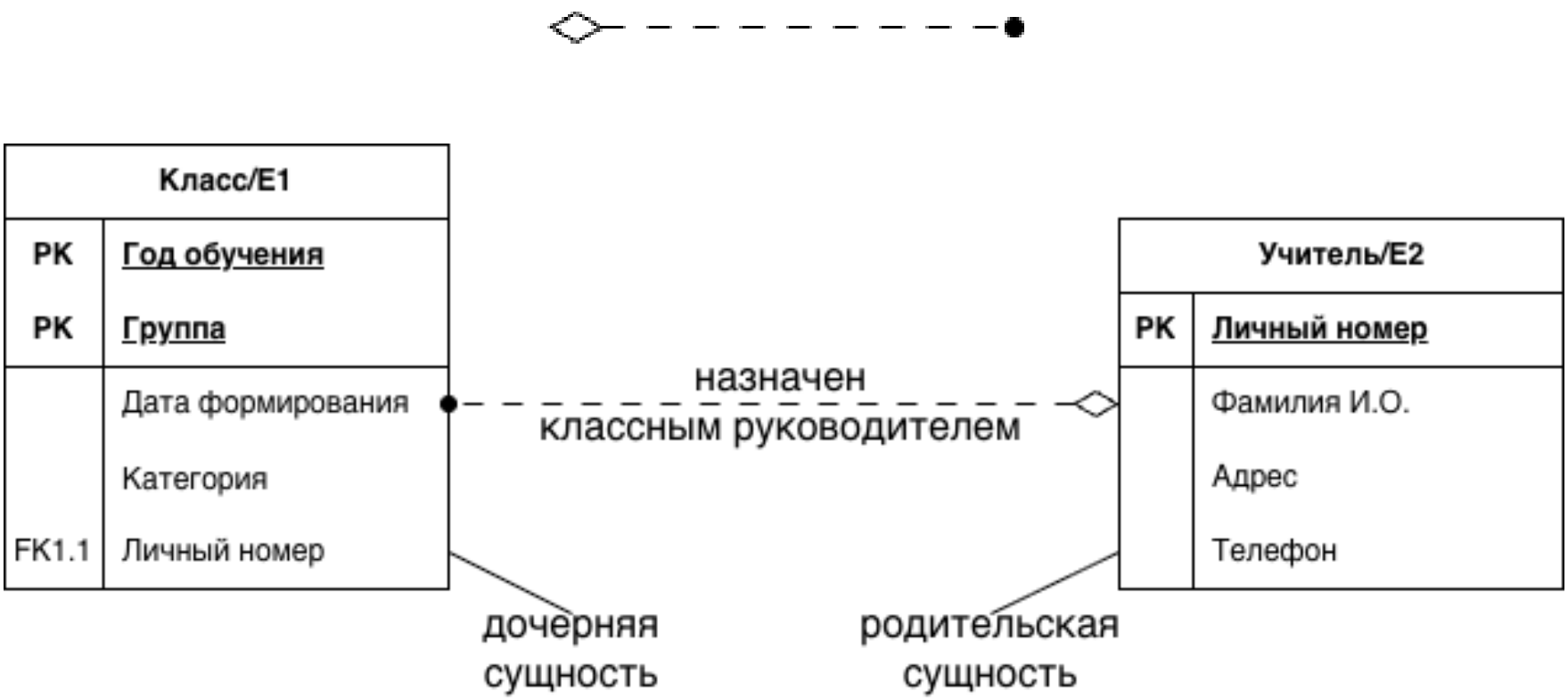
Определенная связь

Определенная не идентифицирующая обязательная связь

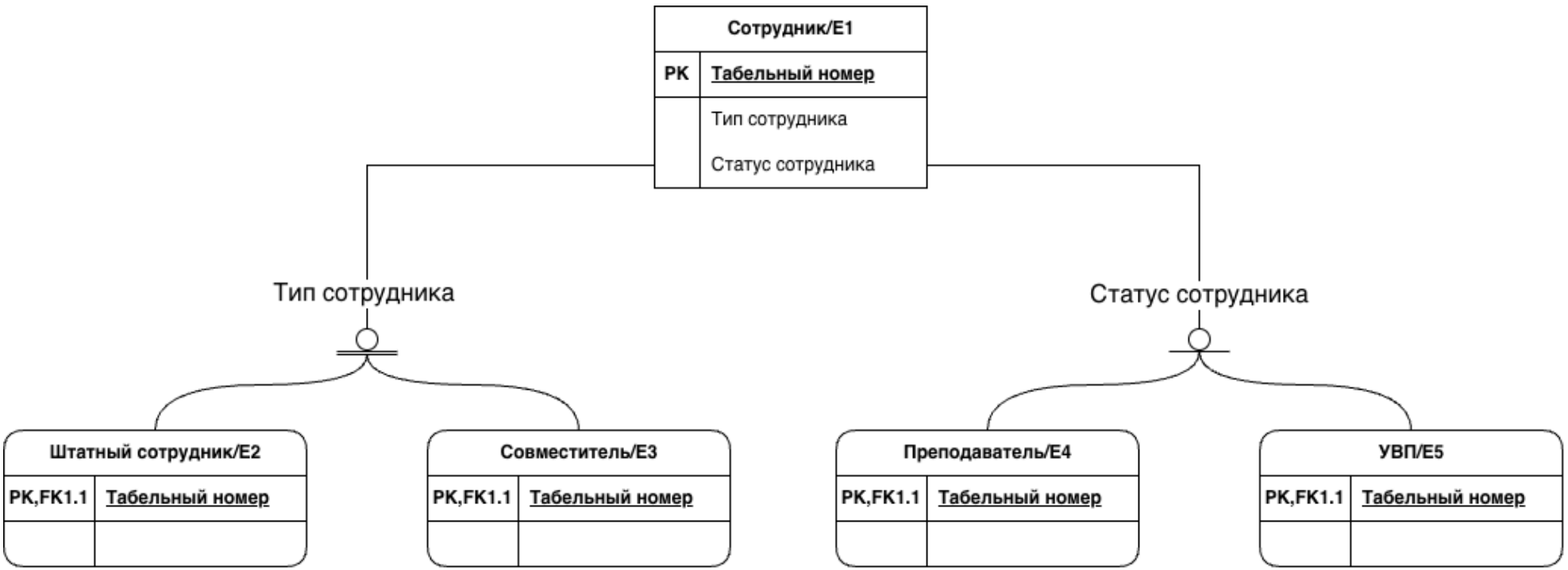


Определенная связь

Определенная не идентифицирующая необязательная связь



Связь типа категория



а) Полная категория

б) Неполная категория

КВ-диаграмма

- выделение первичных и альтернативных ключей
- определение доменов
- разрешение неопределнных связей (типа n:n)
- определение внешних ключей

FA-диаграмма

- выделение атрибутов и доменов
- представление “неудобных” атрибутов в виде отдельных сущностей

Язык описания данных

Язык описания данных (DDL - Data Definition Language) - это часть SQL, которая используется для создания, изменения и удаления структуры базы данных.

Структурный компонент РМД	**Элемент РБД**
Домен	Тип данных
Отношение	Таблица
Атрибут	Колонка таблицы
Кортеж отношения	Строка таблицы

Язык описания данных

Тип ограничения целостности	**Представление в SQL**
Первичный ключ	PRIMARY KEY
Уникальность значения	UNIQUE
Обязательность заполнения	NULL/NOT NULL
Допустимость значения	CHECK
Ссылочные ограничения	FOREIGN KEY

Язык описания данных

- CREATE тип_объекта
- DROP тип_объекта
- ALTER тип_объекта

Типы объектов:

- DATABASE
- TABLE
- VIEW
- INDEX
- TRIGGER
- PROCEDURE
- ...

Объект: таблица

```
CREATE TABLE имя_таблицы (
   имя_колонки тип_данных
   ограничения_на_колонку,
   ...
   табличное_ограничение
   ...
)
```

Ограничение на колонку

Ссылочное ограничение:

```
... REFERENCES родительская_таблица
  (имя_PK в родительской таблице)
  ON DELETE реакция
)
```

Реакция:

```
RESTRICT - запрещает
CASCADE - удаляет все каскадно
SET NULL - ставит NULL
NO ACTION - почти как RESTRICT
SET DEFAULT - ставит DEFAULT
```

Табличное ограничение

Ссылочное ограничение:

```
FOREIGN KEY (список_колонок_таблицы)
  REFERENCES родительская_таблица (PK из
родительской таблицы)
  ON DELETE реакция
)
```

Другие ограничения:

```
имя_ограничения(список_колонок_таблицы)
```

Объект: последовательность

```
CREATE SEQUENCE имя_последовательности
  AS целочисленный_тип
  START WITH константа
  INCREMENT BY константа
  MINVALUE константа / NO MINVALUE
  MAXVALUE константа / NO MAXVALUE
  CYCLE / NO CYCLE
  CACHE константа / NO CACHE
```


Объект: последовательность

```
CREATE SEQUENCE user_seq
  AS bigint
  START WITH 1
  INCREMENT BY 1
  MINVALUE 1
  MAXVALUE 1000000
  NO CYCLE
  CACHE 20;
```


Уникальные значения

Использование последовательности

```
NEXT VALUE FOR имя_последовательности
```

Автоинкрементный тип:

```
IDENTITY (начальное_значение, шаг)
```

Удаление таблицы

```
DROP TABLE имя_таблицы
```

Нельзя удалить родительскую таблицы, если существует хотя бы одна связанная с ней дочерняя таблица.

Измнение таблицы

```
ALTER TABLE имя_таблицы действие
```

Для колонки:

- добавить колонку
- удалить колонку
- изменить колонку

Для ограничений:

- добавить табличное ограничение
- удалить ограничение

Изменение таблицы

Добавить колонку:

```
ALTER TABLE имя_таблицы
  ADD определение_колонки
```

Изменить тип данных колонки:

```
ALTER TABLE имя_таблицы
  ALTER COLUMN имя_колонки TYPE тип_данных
```

Удалить колонку

```
ALTER TABLE имя_таблицы
  DROP COLUMN имя_колонки
```


Изменение таблицы

Добавить табличное ограничение:

```
ALTER TABLE имя_таблицы
   ADD табличное_ограничение
```

Удалить ограничение:

```
ALTER TABLE имя_таблицы
   DROP CONSTRAINT имя_ограничения
```

Типы данных

1 Числовые типы данных

- Целые - `smallint`, `int`, `bigint` (`12`, `100500`)
- С плавающей точкой `real`, `double precision` (`3.14`, `3.1415926535`)
- точный числовой диапазон

```
numeric(p,s), decimal(p,s)
numeric(10,2) → 99999999.99
```

2 Строковые типы

```
varchar(n) ('Hello')
char(n) ('A ')
text 'Большой текст…'
```


Типы данных

3 Дата и время

```
date (2026-02-10)
time (14:30:00)
timestamp (2026-02-10 14:30)
timestamptz (2026-02-10 14:30+03)
interval (2 days 3 hours)
```

4 Логический тип

```
boolean (TRUE / FALSE / NULL)
```

5 Сетевые типы

```
inet (192.168.1.1)
cidr (192.168.1.1/24)
macaddr (08:00:2b:01:02:03)
```


Типы данных

6 UUID

```
uuid (550e8400-e29b-41d4-a716-446655440000)
```

7 JSON и JSONB

JSON - хранит текст (без оптимизации)
JSONB - бинарный, индексируется, работает быстрее

8 Пользовательский типы данных

```
CREATE TYPE status AS ENUM (
  'new','paid','shipped'
);
```

UUID

UUID - это 128-битный идентификатор (16 байт), который обычно пишется как 36 символов с дефисами: 550e8400-e29b-41d4-a716-446655440000.

- UUID v1 (time-based)
  - Основан на времени + идентификаторе узла (исторически MAC) + счётчик.
  - Обычно уникален, частично “упорядочен” по времени.
  - Может утекать информация о времени/узле, не любят за приватность.
- UUID v4 (random)
  - Почти полностью случайный (криптослучайный).
  - Самый популярный, простой, приватный.
  - Плохая локальность для индекса (значения раскиданы случайно)

UUID

- UUID v3 / v5 (name-based)
  - Генерируется как хеш от (namespace + name):
    - v3 — MD5
    - v5 — SHA-1
  - Детерминированный (одно и то же имя → один UUID), удобно для “стабильных ключей”.
  - Не “случайный”, зависит от выбранного namespace и алгоритма.
- UUID v7 (time-ordered)
  - (время + случайность) → лучше для индексов, чем v4, потому что почти монотонный.

В PostgreSQL “из коробки” чаще всего легко получить v4; v7 — через функции/расширения/самописные функции.

UUID

```
CREATE EXTENSION IF NOT EXISTS pgcrypto;

SELECT gen_random_uuid(); -- v4
```

```
CREATE EXTENSION IF NOT EXISTS pgcrypto;

CREATE TABLE orders (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  created_at timestamptz NOT NULL DEFAULT now(),
  payload jsonb NOT NULL DEFAULT '{}'::jsonb
);
```


UUID

В PostgreSQL 18+ есть встроенная функция для UUIDv7

```
SELECT uuidv7();
```

Для PostgreSQL 13-17:

```
-- Но сперва нужно установить pg_uuidv7
CREATE EXTENSION IF NOT EXISTS pg_uuidv7;

SELECT uuid_generate_v7();
```

Необходимо явно приводить к UUID при фильтрации

```
SELECT *
FROM orders
WHERE id = '550e8400-e29b-41d4-a716-446655440000'::uuid;
```

UUID vs BIGINT

- BIGINT IDENTITY
  - компактнее, быстрее индексы, хорошая локальность, но нужен “центр” (одна БД/один генератор)
- UUID v4
  - удобно для распределённых систем, но хуже по индексации
- UUID v7
  - компромисс — почти как UUID, но лучше для индекса

Примеры

```
CREATE TABLE student(
  s_id       INT NOT NULL PRIMARY KEY IDENTITY,
  full_name  TEXT NOT NULL,
  email      TEXT NOT NULL UNIQUE,
  age        SMALLINT NOT NULL CHECK (age > 0 and age < 100),
  created_at TIMESTAMPTZ DEFAULT now()
)
```


Примеры

```
CREATE TABLE enrollment (
  student_id INT NOT NULL REFERENCES student
    ON DELETE NO ACTION,
  course_id INT NOT NULL,
  enrolled_at TIMESTAMPTZ DEFAULT now(),
  grade NUMERIC(3,1),

  PRIMARY KEY (student_id, course_id),
  FOREIGN KEY (course_id) REFERENCES course
    ON DELETE NO ACTION
);
```

Примеры

```
CREATE TABLE app_user (
  user_id bigint GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  email text NOT NULL UNIQUE,
  phone text UNIQUE,
  full_name text NOT NULL,
  is_active boolean NOT NULL DEFAULT true,
  created_at timestamptz NOT NULL DEFAULT now(),
  -- можно создавать вычисляемые поля
  name_len int GENERATED ALWAYS AS (char_length(full_name))
STORED,
  CONSTRAINT chk_phone_format CHECK (phone IS NULL OR phone ~
'^\+?[0-9]{10,15}$')
);
```

Примеры

```
CREATE SEQUENCE product_seq
  AS bigint
  START WITH 1000
  INCREMENT BY 1
  MINVALUE 1000
  NO MAXVALUE
  NO CYCLE
  CACHE 50;
```


Примеры

```
CREATE TABLE product (
  product_id   bigint PRIMARY KEY DEFAULT
nextval('product_seq'),
  sku          text NOT NULL UNIQUE,
  title        text NOT NULL,
  price        numeric(12,2) NOT NULL CHECK (price >= 0),
  meta         jsonb NOT NULL DEFAULT '{}'::jsonb,
  created_at   timestamptz NOT NULL DEFAULT now()
);
```


Примеры

```
CREATE TABLE "order" (
  order_id bigint GENERATED BY DEFAULT AS IDENTITY PRIMARY
KEY,
  user_id bigint NOT NULL,
  status order_status NOT NULL DEFAULT 'new',
  total_amount numeric(12,2) NOT NULL DEFAULT 0 CHECK
(total_amount >= 0),
  created_at timestamptz NOT NULL DEFAULT now(),
  paid_at timestamptz,

  CONSTRAINT fk_order_user
    FOREIGN KEY (user_id) REFERENCES app_user(user_id)
    ON DELETE RESTRICT ON UPDATE CASCADE,
```

Примеры

```
  CONSTRAINT chk_paid_at
    CHECK (paid_at IS NULL OR status IN ('paid','shipped'))
);
```

В данном случае проверяется, что если paid_at заполнено, то статус обязательно должен быть paid или shipped.

Какая проблема?

Примеры

```
  CONSTRAINT chk_paid_at
    CHECK (
      (status IN ('paid','shipped') AND paid_at IS NOT NULL)
      OR
      (status NOT IN ('paid','shipped') AND paid_at IS NULL)
    )
);
```


Примеры

```
CREATE TABLE category (
  category_id bigint GENERATED ALWAYS AS IDENTITY PRIMARY
KEY,
  parent_id bigint,
  name text NOT NULL,

  CONSTRAINT uq_category_name_parent UNIQUE (parent_id,
name),

  CONSTRAINT fk_category_parent
    FOREIGN KEY (parent_id)
    REFERENCES category(category_id)
    ON DELETE SET NULL
);
```


Примеры

```
ALTER TABLE users
ADD COLUMN age int;
```

```
ALTER TABLE users
ADD COLUMN created_at timestamptz DEFAULT now();
```

```
ALTER TABLE users
ADD COLUMN email text NOT NULL;
--  Будет ошибка, если таблица непустая
```


Примеры

```sql
ALTER TABLE users
ADD COLUMN email text;

UPDATE users SET email = 'no@mail.com' WHERE email IS NULL;

ALTER TABLE users
ALTER COLUMN email SET NOT NULL;
```


Примеры

```
ALTER TABLE users
DROP COLUMN age;

ALTER TABLE users
RENAME COLUMN fullname TO full_name;

ALTER TABLE users
ALTER COLUMN age TYPE bigint;

ALTER TABLE users
ALTER COLUMN age TYPE int
USING age::int;
```


Примеры

```
ALTER TABLE users
ALTER COLUMN email SET NOT NULL;

ALTER TABLE users
ALTER COLUMN phone DROP NOT NULL;

ALTER TABLE users
ALTER COLUMN created_at SET DEFAULT now();

ALTER TABLE users
ALTER COLUMN created_at DROP DEFAULT;
```


Примеры

```sql
ALTER TABLE users
RENAME TO app_users;

ALTER TABLE users
ADD CONSTRAINT pk_users PRIMARY KEY (id);

ALTER TABLE users
ADD CONSTRAINT uq_users_email UNIQUE (email);

ALTER TABLE orders
ADD CONSTRAINT chk_total
CHECK (total_amount >= 0);
```

Примеры

```
ALTER TABLE orders
ADD CONSTRAINT fk_orders_user
FOREIGN KEY (user_id)
REFERENCES users(id)
ON DELETE CASCADE;


ALTER TABLE users
DROP CONSTRAINT uq_users_email;
```


Примеры

```
ALTER TABLE users
  ADD COLUMN age int,
  ALTER COLUMN email SET NOT NULL,
  DROP COLUMN temp_field;
```

Конец