# Лекция 2:
# создание Базы Данных

# 1. Построение БД

ИТМО ВТ

# С чего начать построение БД?

**Предметная область** - часть реального мира, данные о которой необходимо разместить в базе данных (учебный процесс в университете, магазин и т.п.)

ИТМО ВТ

# Пользовательский уровень

Обычный пользователь системы не знает о:

- фактическом размещении данных в памяти компьютера;
- механизмах для оптимизации операций с данными;
- обработке запроса и осуществлении поиска;

ИТМО ВТ

# Пользовательский уровень

Как представляет предметную область (часть предметной области) пользователь?

- Пользователь обычно:
  - взаимодействует с частью системы;
  - имеет неполное представление о системе в целом;
- Сбор информации в виде: текста, диаграмм и т.д;

ИТМО ВТ

# Инфологический уровень

- Инфологическая модель:
  - обобщенное представление предметной области;
  - собирается на основе анализа пользовательских представлений;
  - не зависит от «физического» хранилища;
  - есть стандартные средства для описания (например, ER-диаграмма);

ИТМО ВТ

# Даталогическая модель

- Представление инфологической модели с  учетом:
  - используемой модели данных;
  - могут появляться детали, относящиеся к СУБД (типы данных);

ИТМО ВТ

# Физический уровень

- Реализация даталогической модели средствами СУБД:
  - зависит от особенностей конкретной СУБД.
  - описывается на языке, поддерживаемом СУБД.

- Пример: SQL-код описания таблиц:

  CREATE TABLE STUDENT ( … )

ИТМО ВТ

# Проектирование базы данных

1. Пользовательское представление о предметной области.
2. Инфологическая модель.
3. Физическая реализация.

Проектирование «сверху-вниз» (top-down approach)

# 2. Инфологическая модель

ИТМО ВТ

# Модель «сущность-связь»

- Один из вариантов для построения инфологической модели: ER-диаграммы (Entity-Relationship - сущность-связь).
- Питер Чен (1976, IBM).

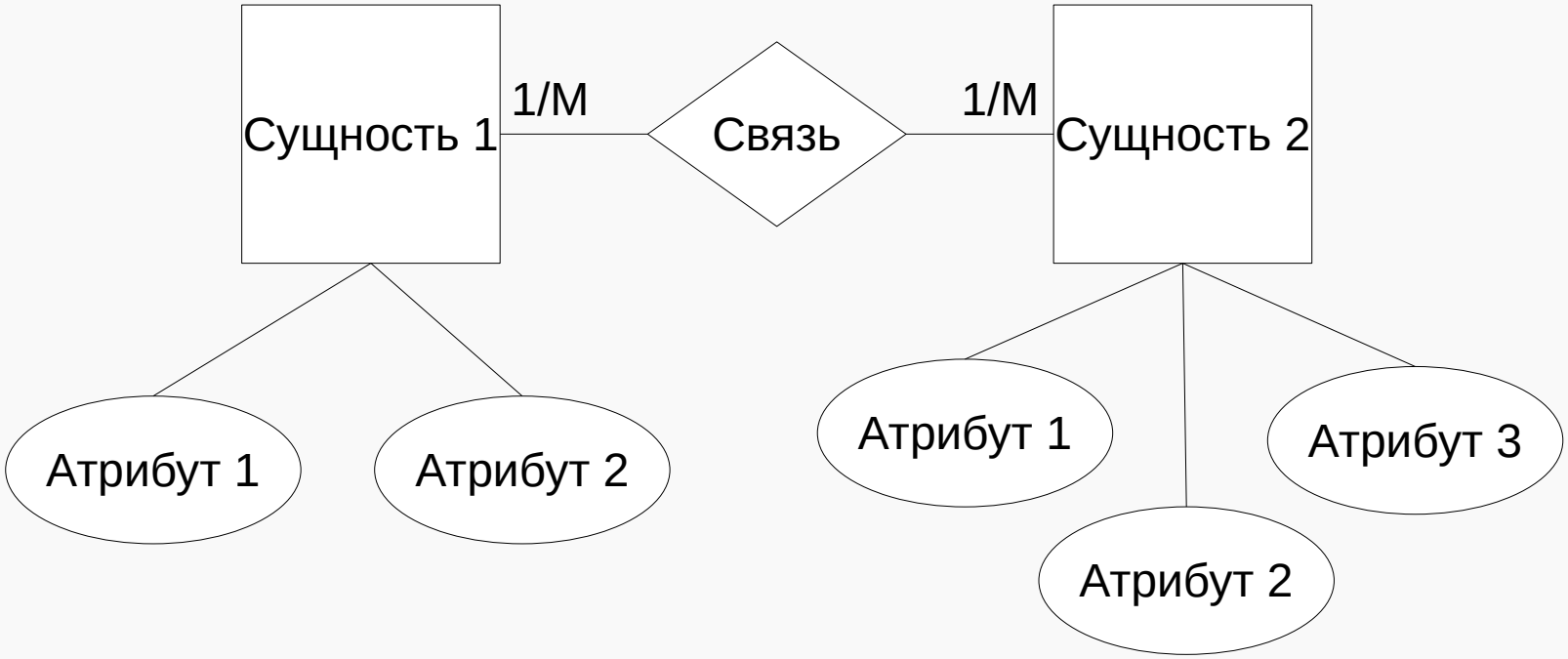
ИТМО ВТ

# Основные элементы ИМ

- *Сущность* — класс объектов, фактов, явлений, предметов, элементы которых будут храниться в базе данных.
- *Экземпляр сущности* относится к конкретной вещи в наборе. Например, типом сущности (сущностью) может быть СТУДЕНТ , а экземпляром — Иван Иванов и т. д.
- *Атрибут* — важная характеристика (свойство) сущности, которой присваивается имя.
- *Связь* — ассоциирование двух или более сущностей, выражающая форму взаимодействия между ними.

ИТМО ВТ

# Изображение основных элементов




Над связями - степень связи (1 или  M - "много")

ИТМО ВТ

# Связи между сущностями

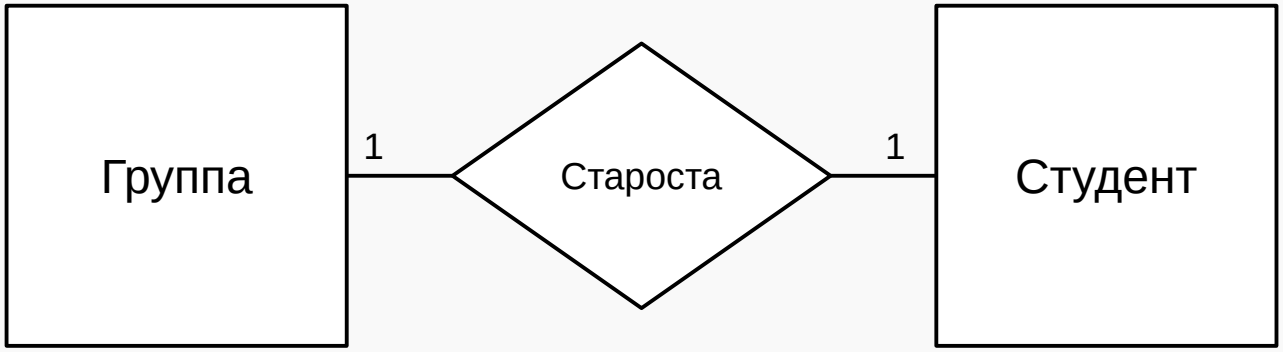
4 (3) вида связей:

- "один-к-одному" (1:1)
- "один-ко-многим" (1:M)
- "многие-к-одному" (M:1)
- "многие-ко-многим" (M:M)

ИТМО ВТ

# Один-к-одному (1:1)

В каждый момент времени каждому экземпляру первой сущности соответствует 1 или 0 экземпляров второй сущности:



ИТМО ВТ

# Один-ко-многим (1:M)

Одному экземпляру первой сущности соответствуют 0, 1 или несколько экземпляров второй сущности;

Одному экземпляру второй сущности соответствует 0 или 1 экземпляров первой сущности;

ИТМО ВТ

# Многие-к-одному (M:1)

Обратная связь к 1:M (рассматриваем со стороны второй сущности).


Связь сущностей Студент-Группа - M:1

ИТМО ВТ

# Многие-ко-многим (M:M)

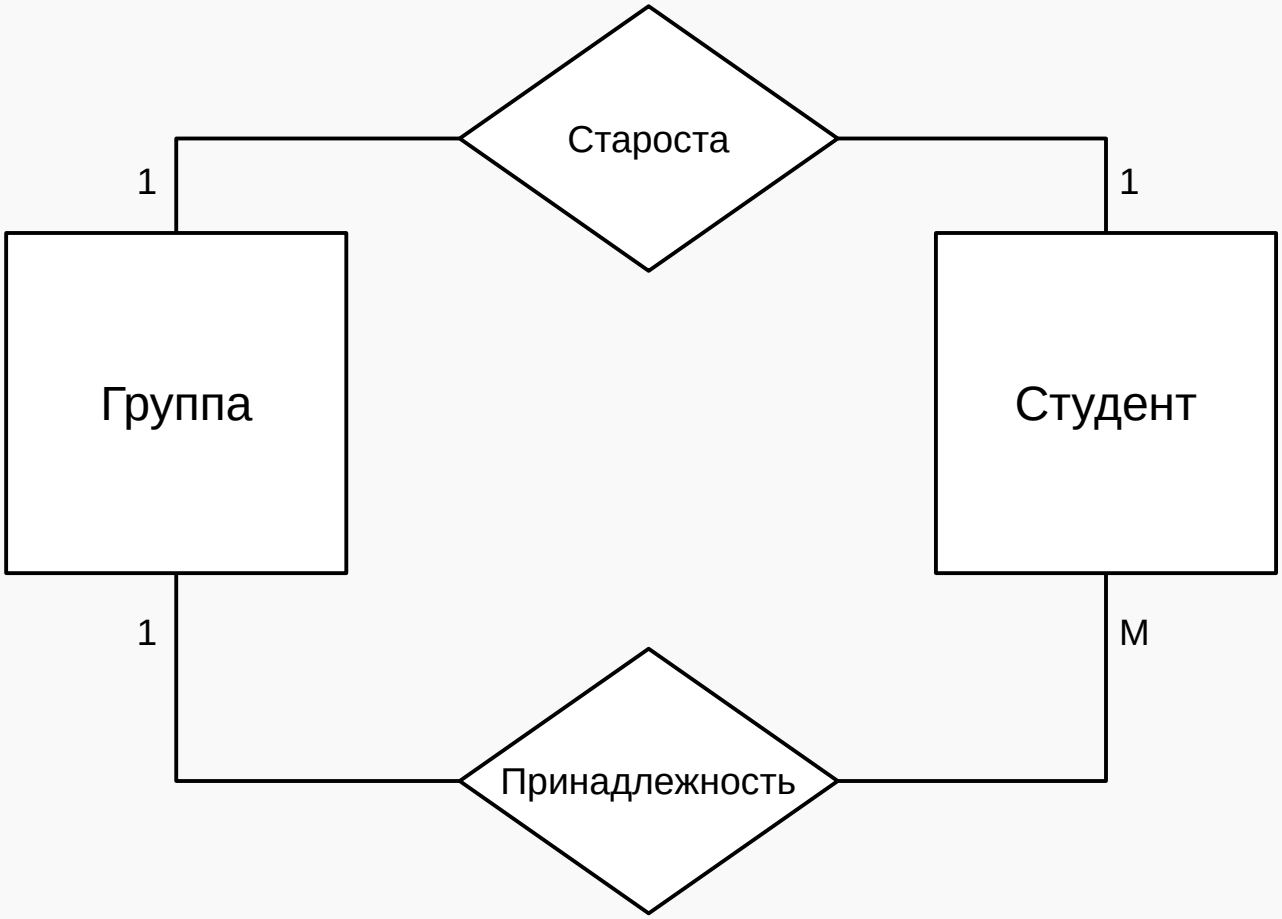
Одному экземпляру первой сущности соответствуют 0, 1 или несколько экземпляров второй сущности;

Одному экземпляру второй сущности соответствует 0, 1 или несколько экземпляров первой сущности;

ИТМО ВТ

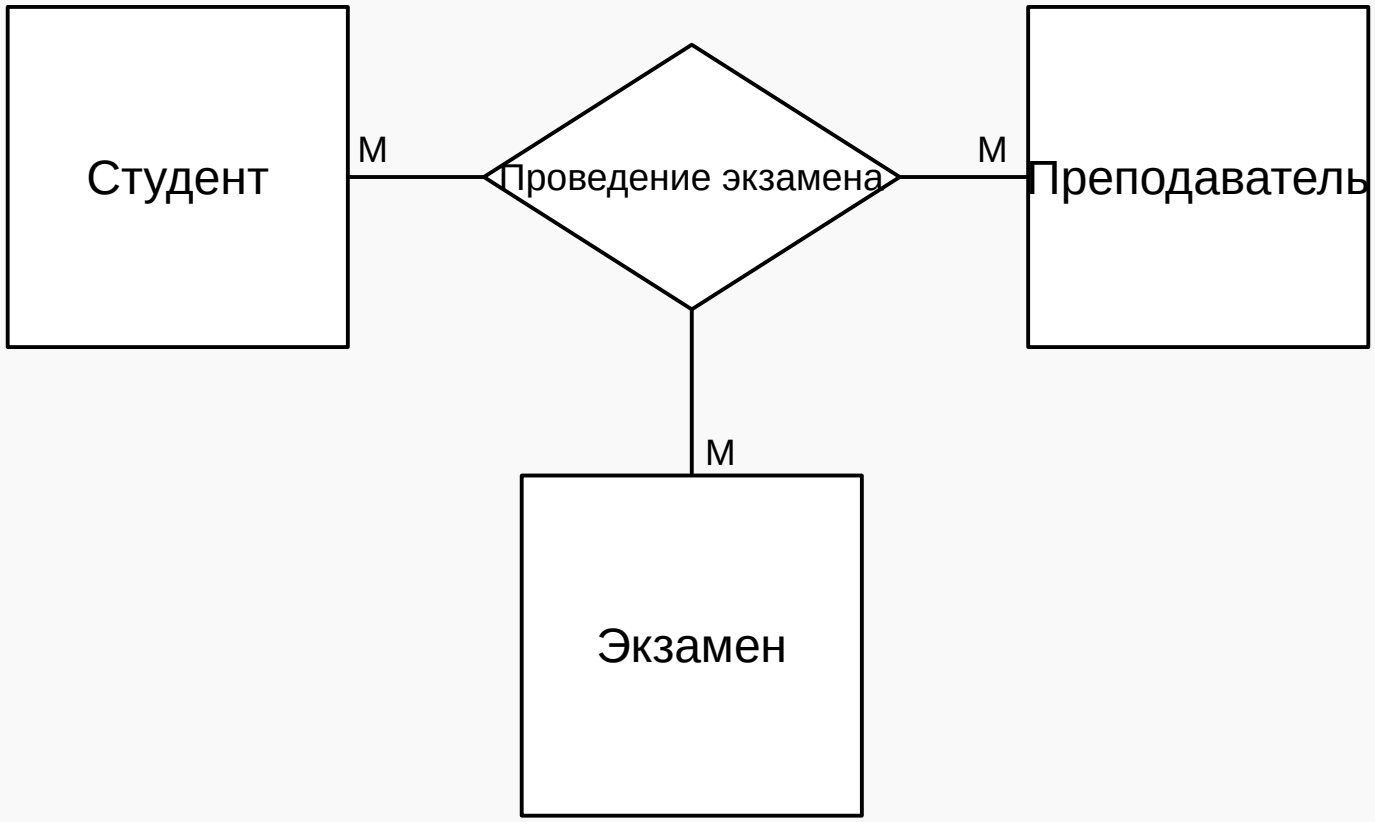
# Сложные связи

Несколько связей между двумя сущностями;



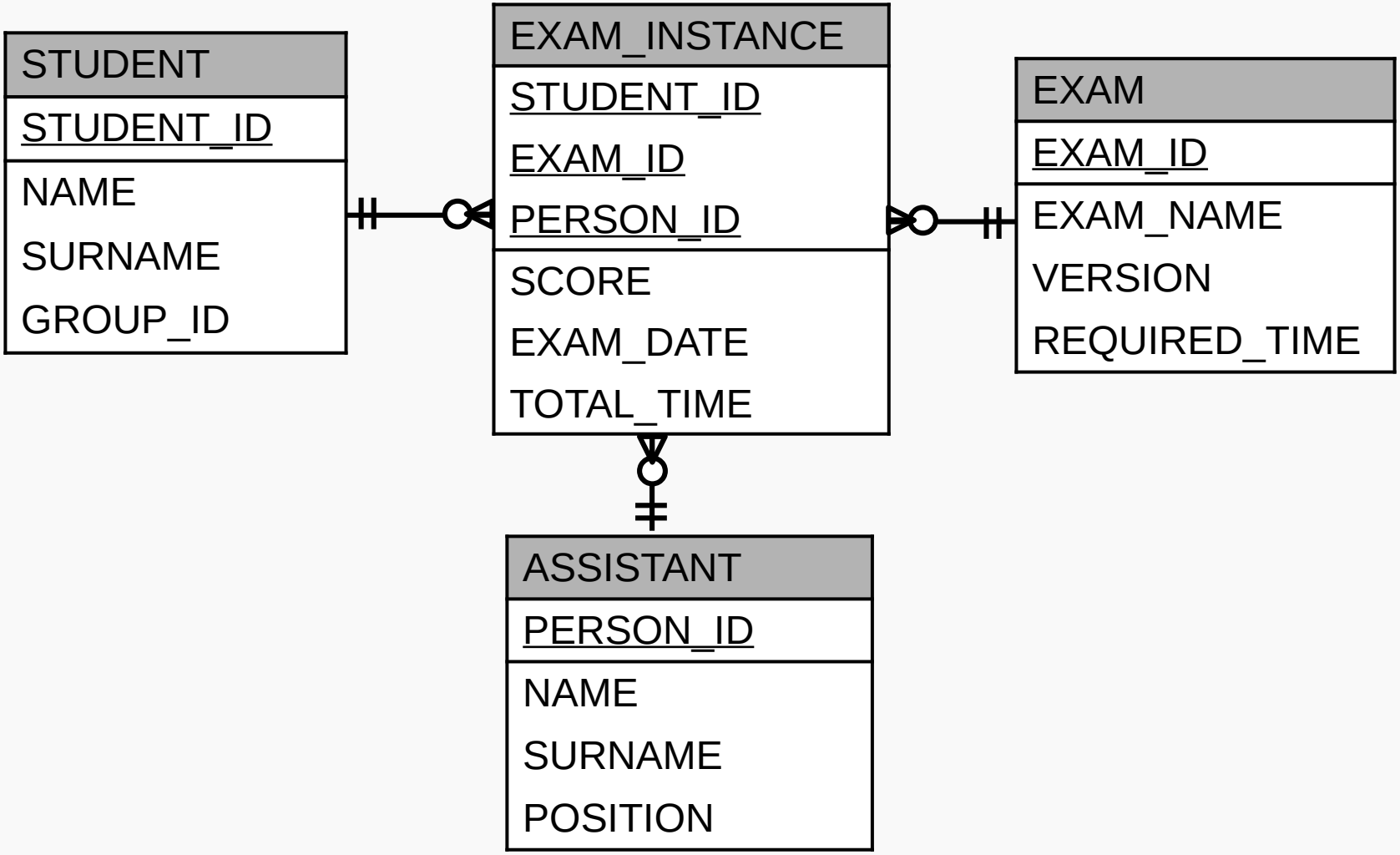
ИТМО ВТ

# Тренарные связи



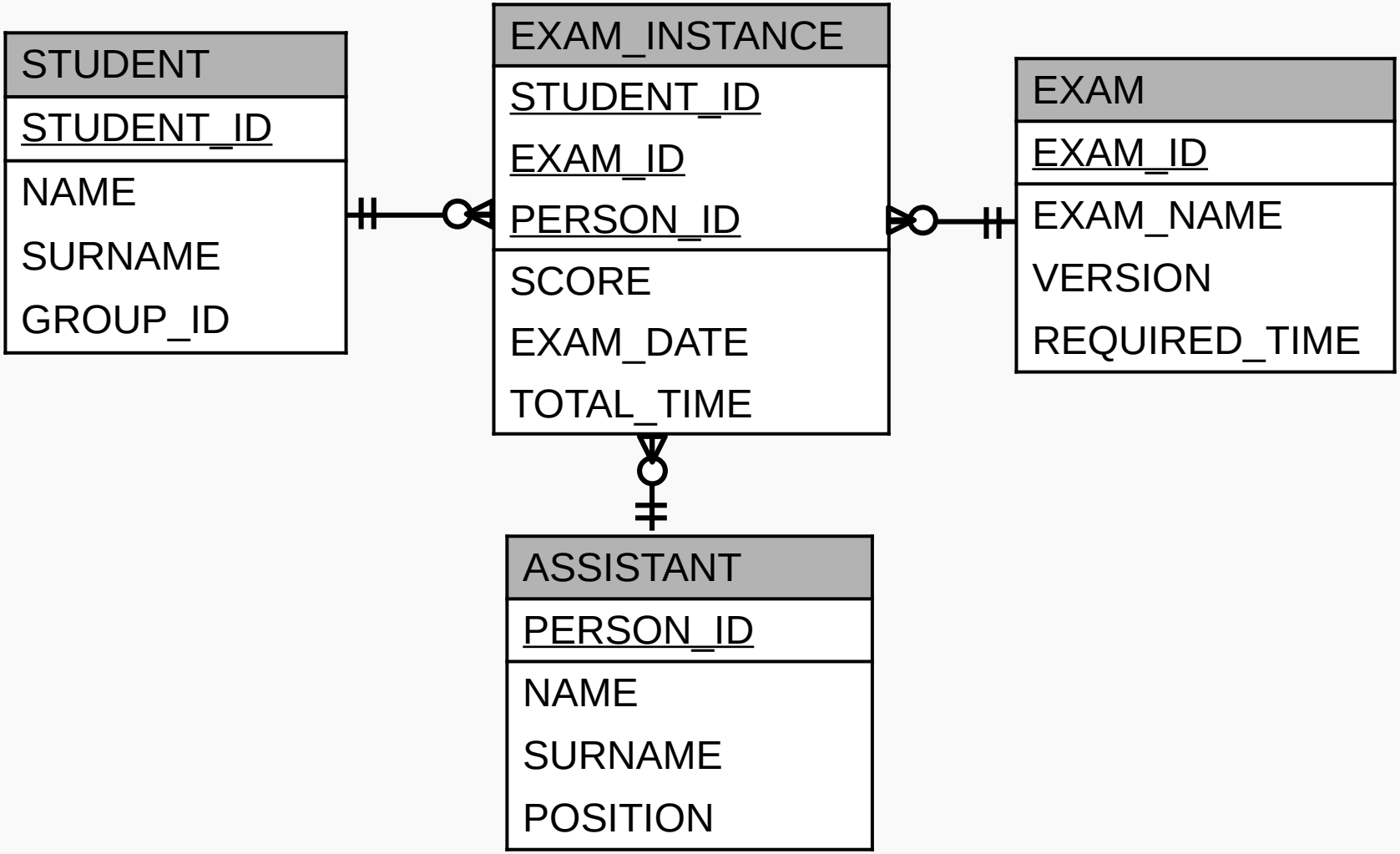
ИТМО ВТ

# Дополненное представление



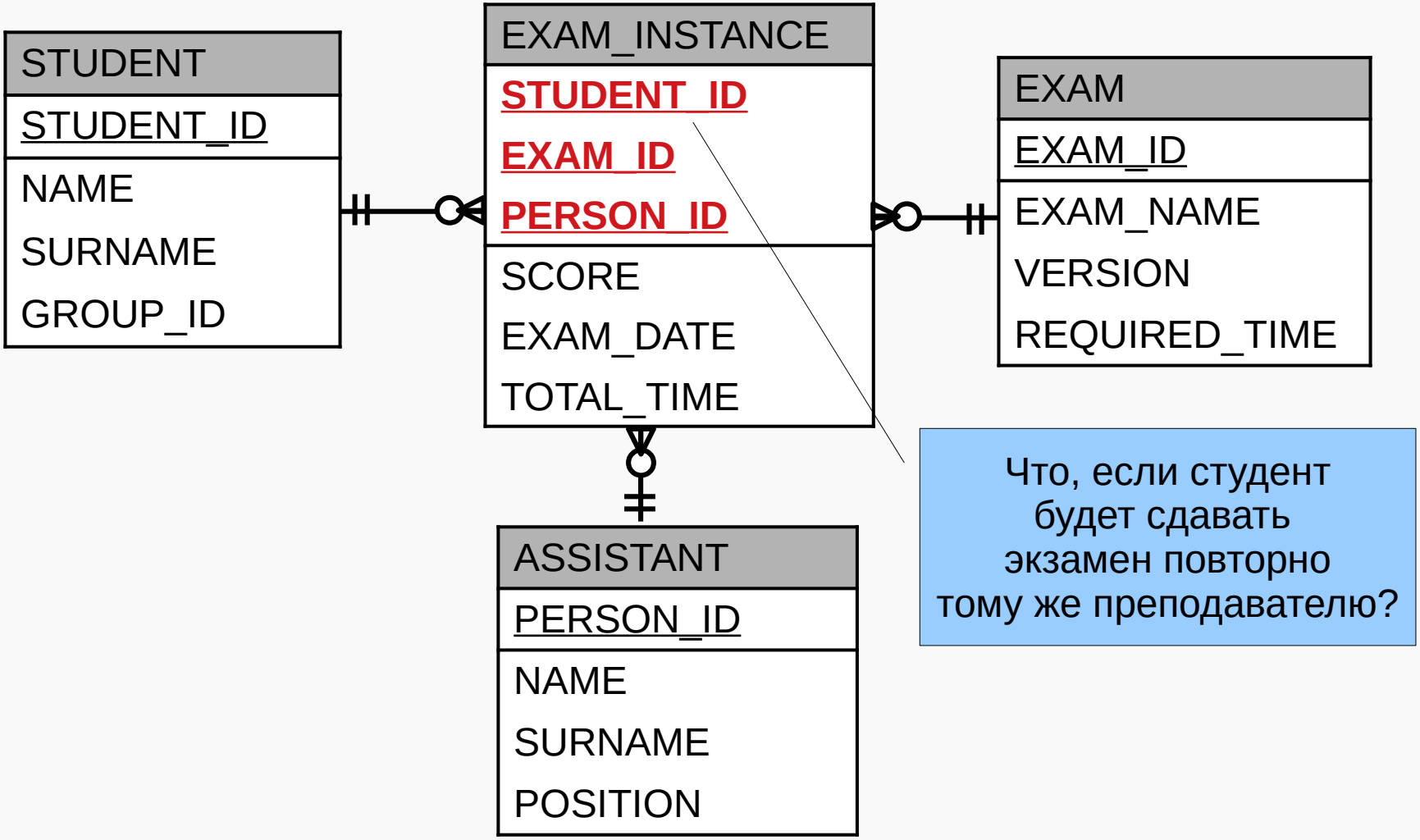
ИТМО ВТ

# Какие недостатки в модели?



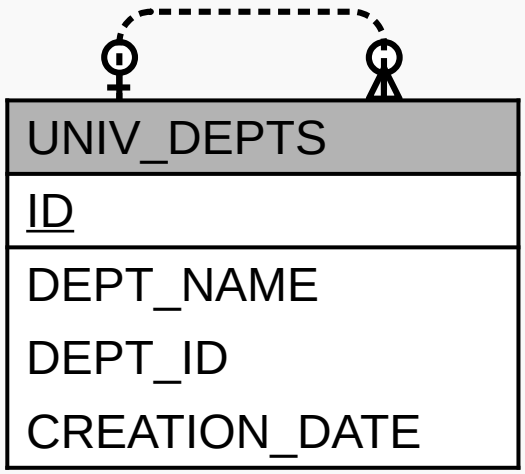
ИТМО ВТ

# Какие недостатки в модели?



ИТМО ВТ

# Рекурсивная связь



ИТМО ВТ

# Классификация сущностей

Э. Кодд (1979, RM/T):

- стержневые
- ассоциативные
- арактеристические

- обычные
- слабые

- типы
- подтипы

ИТМО ВТ

# Классификация сущностей

- **Стержневая сущность** (стержень) — независимая, базовая сущность (Студент, Группа)
- **Ассоциативная сущность** (ассоциация) — связь вида "многие-ко-многим" ("*-ко-многим" и т. д.) между двумя или более сущностями
- **Характеристическая сущность** (характеристика) — связь вида "многие-к-одной" или "одна-к-одной" между двумя сущностями (частный случай ассоциации). Цель характеристики - описание или уточнении некоторой другой сущности.

ИТМО ВТ

# Ключ

- ***Ключ*** — минимальный набор атрибутов, по значениям которых однозначно определяется требуемый экземпляр сущности.
- ***Суррогатный ключ*** - автоматически сгенерированный атрибут для однозначной идентификации экземпляра сущности в рамках данной сущности.

# 3. Создание реляционной БД

ИТМО ВТ

# Объекты базы данных

- **Объекты** базы данных - таблицы, представления, процедуры, триггеры.
- Для работы с объектами БД используется **SQL**:
  - Предложения манипуляции данными (Data Manipulation Language, **DML**)
  - Предложения определения данных (Data Definition Language, **DDL**)
  - Предложения определения доступа к данным (Data Control Language, **DCL**)

ИТМО ВТ

# Составляющие языка SQL

- Предложения;
- Идентификаторы (имена);
- Константы;
- Операторы;
- Зарезервированные и ключевые слова;

ИТМО ВТ

# Предложения SQL

**Предложение** — команда, которая состоит из ключевых и зарезервированных слов, определяемых пользователем в соответствии с синтаксическими правилами SQL:

```
SELECT * FROM STUDENTS;
```

ИТМО ВТ

# Идентификаторы

- Имена объектов баз данных (определенные пользователем или системные).
- Создается при определении объекта.
- Используется для обращения к объекту:

```
SELECT * FROM STUDENTS;
```

ИТМО ВТ

# Константы

Любые значения, которые не являются идентификаторами или ключевыми словами:

- числовые значения: 9999, 5E6
- строки символов: 'Пример строки'
- значения, связанные с представлением времени (дата и время): 06-03-2017 10:06:54
- булевы значения: TRUE

ИТМО ВТ

# Операторы

- Служат для обозначения действия над одним или несколькими выражениями.

- Делятся на категории: арифметические, логические, присваивания, сравнения и тд.

- SELECT * FROM STUDENTS WHERE AGE > 19;

# Зарезервированные и ключевые слова

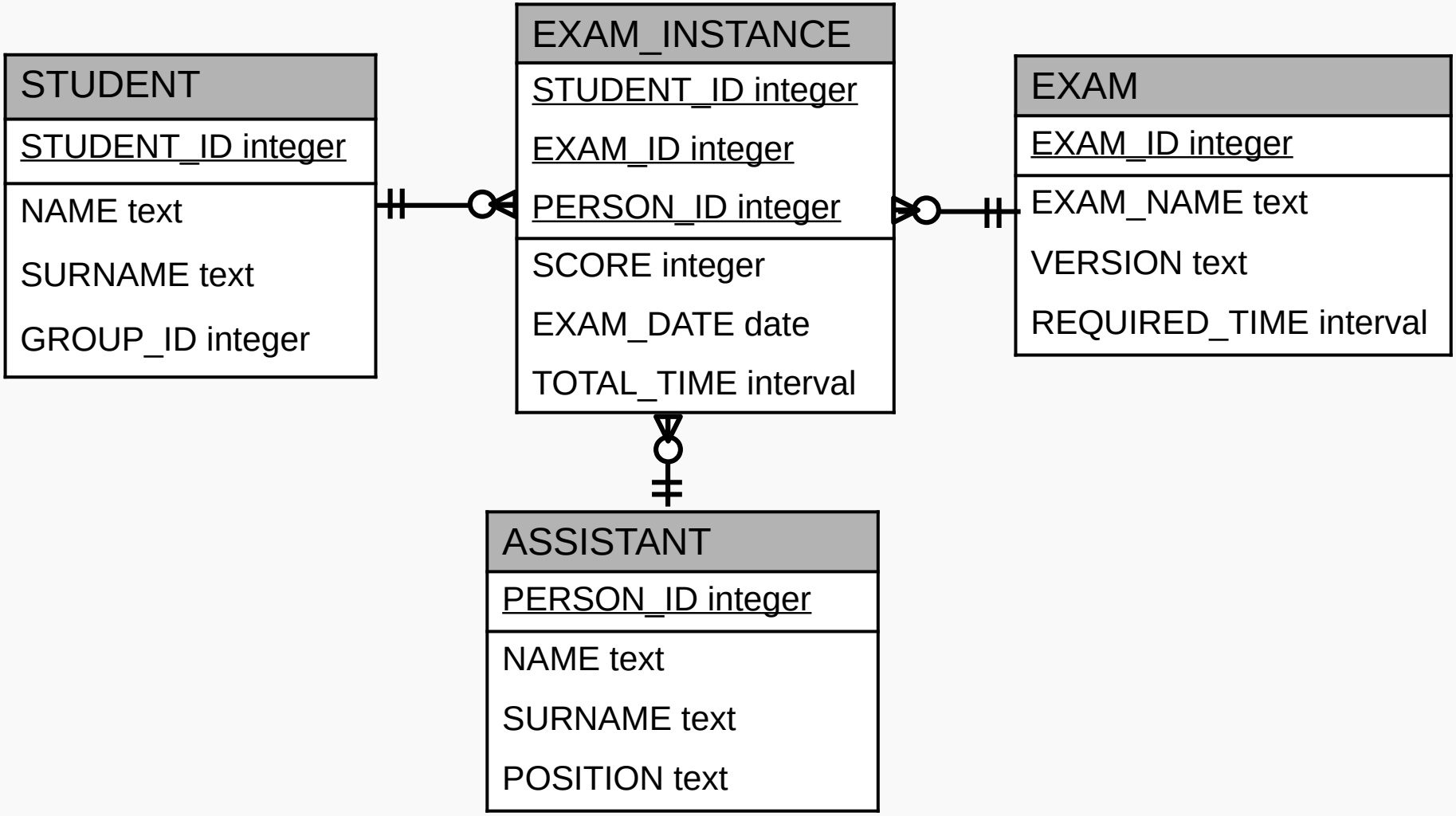
Слова и фразы для задания конструкций и использования возможностей языка SQL;

```
SELECT * FROM STUDENTS WHERE AGE > 19;
```

# 4. Работа с таблицами

ИТМО ВТ

# Даталогическая модель



ИТМО ВТ

# DDL

Операторы определения данных *(Data Definition Language, DDL)*:

- CREATE создает объект БД (саму базу, таблицу, представление, пользователя и т. д.);
- ALTER изменяет объект;
- DROP удаляет объект.

ИТМО ВТ

# Создание таблиц

Таблицы:

- *Базовые* — в действительности существующие, хранящиеся в физической памяти.

- *Виртуальные* — представления, курсоры, неименованные таблицы (таблицы, которые не существуют постоянно в базе данных).

# Создание базовой таблицы

Используется предложение CREATE TABLE:

```
CREATE TABLE STUDENTS (
    STUD_ID integer,
    STUD_NAME text,
    BIRTH_DATE date
);
```

ИТМО ВТ

# Удаление базовой таблицы

Используется предложение DROP TABLE:

**DROP TABLE** *STUDENTS;*

ИТМО ВТ

# Изменение таблиц

Используется предложение ALTER:

**ALTER TABLE** STUDENTS **ADD COLUMN** STUD_GROUP text;

**ALTER TABLE** STUDENTS **DROP COLUMN** BIRTH_DATE;

ИТМО ВТ

# Типы данных

Каждая колонка имеет свой **тип данных:**

- Ограничивает список возможных значений, которые могут находится в этой колонке
- Определяет «смысл» данных, хранящихся в колонке, чтобы данные могли использоваться при вычислениях.

ИТМО ВТ

# Типы данных

Тип данных задается при создании:

```
CREATE TABLE STUDENTS (
    STUD_ID integer,
    STUD_NAME text,
    BIRTH_DATE date
);
```

ИТМО ВТ

# Числовые типы

PostgreSQL:

- smallint — целые числа (2 байта);
- integer — целые числа (4 байта);
- bigint — целые числа (8 байт);

ИТМО ВТ

# Символьные типы

PostgreSQL:

- varchar(n) — переменной длины с ограничением;
- char(n) — фиксированной длины, остаток заполняется пробелами;
- text — переменной длины без задаваемого ограничения;

где n — положительное число.

```
CREATE TABLE STUDENTS (ST_NAME char(25));
INSERT INTO STUDENTS VALUES ('Valery');
```

ИТМО ВТ

# Логический тип

Тип boolean:

- состояния: «true», «false» (третье состояние, «unknown» представляется через SQL-значение NULL);
- «true» может задаваться следующими значениями: **TRUE**, 'true', 't', 'yes', 'y', 'on', '1';
- «false» может задаваться следующими значениями: **FALSE**, 'false', 'f', 'no', 'n', 'off', '0';

ИТМО ВТ

# Типы даты/времени

PostgreSQL:

- time — время суток;
- timestamp — дата и время;
- date — дата;
- interval — временной интервал;

**TIMESTAMP** '2016-11-23 11:13:44+01'

ИТМО ВТ

# NULL-значения

- NULL — специальное значение (пустое, несуществующее значение), которое может быть записано в поле таблицы базы данных.

- NULL-значения нужно рассматривать как "отсутствие информации" (**не как** пустые строки, пробелы и тд).

# Что будет в результате запроса?

STUDENT

| id | name | surname | gr_id |
|---|---|---|---|
| 1 | Григорий | Иванов | 34 |
| 2 | Григорий | Петров | NULL |
| 3 | Иван | Сидоров | NULL |

```
SELECT COUNT(gr_id) FROM STUDENT
WHERE gr_id is NULL;
```

ИТМО ВТ

# NULL-значения

- Значения типа NULL **не равны** друг другу
- Столбец, содержащий значение NULL , **игнорируется** при вычислениях агрегатных значений
- В предложениях с DISTINCT, ORDER BY, GROUP BY, значения NULL **не отличаются** друг от друга.

# Что будет в результате запроса?

STUDENT

| id | name | surname | gr_id |
|---|---|---|---|
| 1 | Григорий | Иванов | 34 |
| 2 | Григорий | Петров | NULL |
| 3 | Иван | Сидоров | NULL |

SELECT **COUNT(*)** FROM STUDENT
WHERE gr_id is NULL;

ИТМО ВТ

# Значения по умолчанию

```
CREATE TABLE GROUPS (
    GR_ID integer,
    GR_NAME text,
    GR_COUNT integer DEFAULT 0
);
```

Значение по умолчанию может быть **выражением**

ИТМО ВТ

# Ограничения целостности

- Типы данных — один из способов ограничивать данные, но его не всегда **достаточно**.

- SQL позволяет определять ограничения для колонок и таблиц.

ИТМО ВТ

# CHECK

```
CREATE TABLE STUDENTS (
    ST_ID integer,
    ST_NAME text,
    FAILED_COURSES integer
                    CHECK (FAILED_COURSES >= 0)
);
```

ИТМО ВТ

# CHECK

- Позволяет задать для определённой колонки, выражение, которое будет осуществлять проверку, помещаемого в эту колонку значения.
- Выражение должно возвращать логическое значение
- Проверка CHECK проходит, если выражение, указанное в ограничении возвращает значение истина или null

ИТМО ВТ

# CHECK

Ограничению можно задать имя (чтобы на него ссылаться):

```
CREATE TABLE STUDENTS (
    ST_ID integer,
    ST_NAME text,
    FAILED_COURSES integer CONSTRAINT fcrs
                    CHECK (FAILED_COURSES >= 0)
);
```

ИТМО ВТ

# CHECK

Ограничение можно накладывать на таблицу:

```
CREATE TABLE STUDENTS (
    ST_ID integer,
    FAILED_MAX integer,
    FAILED_COURSES integer,
    CONSTRAINT fcrs
        CHECK
            (FAILED_COURSES >= 0 AND
             FAILED_COURSE <= FAILED_MAX)
);
```

ИТМО ВТ

# NOT NULL

Колонка не должна содержать значение **null**:

```
CREATE TABLE STUDENTS (
    ST_ID integer NOT NULL,
    FAILED_MAX numeric NOT NULL,
    FAILED_COURSES integer NOT NULL
);
```

Логически эквивалентно созданию ограничения:

```
CHECK (column_name IS NOT NULL)
```

ИТМО ВТ

# Ограничение уникальности

Данные в колонке или группе колонок являются уникальными по отношению к данным из той же колонки/группы колонок для других строк в той же таблице:

```
CREATE TABLE STUDENTS (
    ST_ID integer NOT NULL UNIQUE,
    ST_NAME text NOT NULL,
    ST_SURNAME text NOT NULL,
    ST_BIRTH date NOT NULL
);
```

ИТМО ВТ

# Ограничение уникальности

```
CREATE TABLE STUDENTS (
    ST_ID integer UNIQUE NOT NULL,
    ST_NAME text NOT NULL,
    ST_SURNAME text NOT NULL,
    ST_BIRTH date NOT NULL,
    UNIQUE (ST_NAME, ST_SURNAME, ST_BIRTH)
);
```

ИТМО ВТ

# Первичный ключ

```
CREATE TABLE STUDENTS (
    ST_ID integer PRIMARY KEY,
    ST_NAME text NOT NULL,
    ST_SURNAME text NOT NULL,
    ST_BIRTH date NOT NULL,
    UNIQUE (ST_NAME, ST_SURNAME, ST_BIRTH)
);
```

ИТМО ВТ

# Первичный ключ

- Первичный ключ означает, что колонка (группа колонок) используются как уникальный идентификатор строки в таблице
- Ограничение первичного ключа эквивалентно комбинации ограничений уникальности и не-null
- Таблица может иметь не более одного первичного ключа.

ИТМО ВТ

# Первичный ключ (PostgreSQL)

- Добавление первичного ключа автоматически создает уникальный b-tree индекс на колонку.

- PostgreSQL позволяет не создавать первичный ключ для таблицы (согласно теории **требуется** наличие первичного ключа для каждой таблицы).

ИТМО ВТ

# Внешний ключ

- Позволяет отобразить связи между различными сущностями.
- Поддержка целостности на уровне связей.

ИТМО ВТ

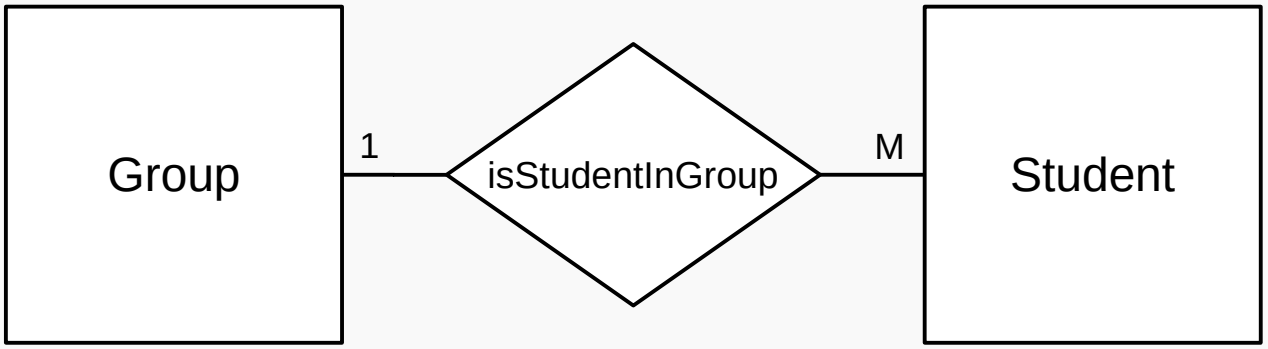
# Внешний ключ

***Внешний ключ*** — множество атрибутов сущности для организации связей с экземплярами другой сущности (должен соответствовать *первичному* ключу в первой сущности)

- Если сущность связывает две другие сущности, то она должна включать внешние ключи, соответствующие первичным ключам связываемых сущностей.
- Если одна сущность характеризует другую сущность, то она должна включать внешний ключ, соответствующий первичному ключу первой сущности.

ИТМО ВТ

# Внешний ключ




```
CREATE TABLE GROUP (
    GR_ID integer PRIMARY KEY,
    GR_NAME text
);
CREATE TABLE STUDENT (
    ST_ID integer PRIMARY KEY,
    ST_NAME text,
    GR_ID integer REFERENCES GROUP (GR_ID)
);
```

# Внешний ключ

```
CREATE TABLE GROUP (
    GR_ID integer PRIMARY KEY,
    GR_NAME text
);
```

| GR_ID | GR_NAME |
|---|---|
| 1 | 3119 |

```
CREATE TABLE STUDENT (
    ST_ID integer PRIMARY KEY,
    ST_NAME text,
    GR_ID integer REFERENCES GROUP (GR_ID)
);
```

| ST_ID | ST_NAME | GR_ID |
|---|---|---|
| 42 | Chuck Norris | 1 |

# Внешний ключ

```
CREATE TABLE GROUP (
    GR_ID integer PRIMARY KEY,
    GR_NAME text
);

CREATE TABLE STUDENT (
    ST_ID integer PRIMARY KEY,
    ST_NAME text,
    GR_ID integer REFERENCES GROUP
);
```

| GR_ID | GR_NAME |
|---|---|
| 1 | 3119 |

| ST_ID | ST_NAME | GR_ID |
|---|---|---|
| 42 | Chuck Norris | 1 |

*(сокр. Запись) — будет использовано значение первичного ключа из таблицы GROUP

# Внешний ключ (?)

```
CREATE TABLE GROUP (
    GR_ID integer PRIMARY KEY,
    GR_NAME text
);
```

| GR_ID | GR_NAME |
|---|---|
| 1 | 3119 |

```
CREATE TABLE STUDENT (
    ST_ID integer PRIMARY KEY,
    ST_NAME text,
    GR_ID integer REFERENCES GROUP
);
```

| ST_ID | ST_NAME | GR_ID |
|---|---|---|
| 42 | Chuck Norris | 1 |

**Что произойдет при удалении группы?**

```
DELETE FROM GROUP WHERE GR_ID = 1;
```

ИТМО ВТ

# Варианты

1. Ничего не делать.
2. Удалить каскадно.
3. Установить в null.
4. Установить значение по умолчанию.

# Внешний ключ

```
CREATE TABLE GROUP (
    GR_ID integer PRIMARY KEY,
    GR_NAME text
);
```

| GR_ID | GR_NAME |
|---|---|
| 1 | 3119 |

```
CREATE TABLE STUDENT (
    ST_ID integer PRIMARY KEY,
    ST_NAME text,
    GR_ID integer REFERENCES GROUP ON DELETE CASCADE
);
```

| ST_ID | ST_NAME | GR_ID |
|---|---|---|
| 42 | Chuck Norris | 1 |

*другие варианты: ON DELETE RESTRICT,
ON DELETE SET DEFAULT, ON DELETE SET NULL
**аналогично ON UPDATE ...

# 5. Отображение ER-диаграмм в БД

ИТМО ВТ

# Отображение

- *Сущность* — таблица
- *Экземпляр сущности* — строка таблицы
- *Атрибут* — столбец в таблице
- *Связь* — внешний ключ, таблица

ИТМО ВТ

# Связь «один-к-*»

- Реализуется путем добавления в таблицу внешнего ключа.
- Внешний ключ обычно добавляется в сущность-ассоциацию или в сущность-характеристику.
- Для моделирования характера связи на внешний ключ вводятся доп. ограничения (например, «один-к-одному» - UNIQUE).

ИТМО ВТ

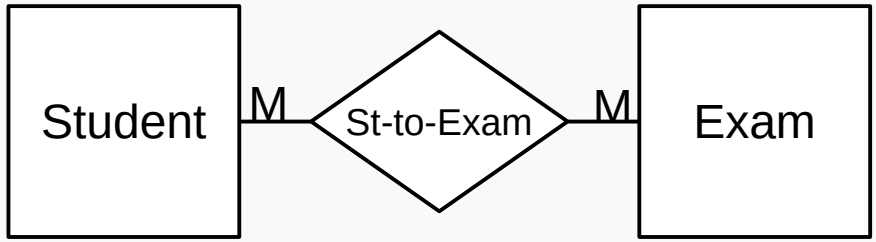
# Связь «многие-ко-многим»

Для реализации связи вида «многие-ко-многим» создается вспомогательная таблица:

# Связь «многие-ко-многим»

```
CREATE TABLE STUDENT (
    ST_ID integer PRIMARY KEY,
    ST_NAME text
);

CREATE TABLE EXAM (
    EX_ID integer PRIMARY KEY,
    EX_NAME text
);
```




```
CREATE TABLE STUD_TO_EXAM (
    ST_ID integer REFERENCES STUDENTS,
    EX_ID integer REFERENCES EXAMS,
    PRIMARY KEY (ST_ID, EX_ID)
);
```

# 6. Работа с таблицами

ИТМО ВТ

# Создание таблицы с LIKE

Предположим, что есть таблица:

```
CREATE TABLE APPLICANT (
        StudentID int,
        Name text,
        Surname text
);
```

Можно создать таблицу на основе уже созданной таблицы.

ИТМО ВТ

# Создание таблицы с LIKE

```
CREATE TABLE STUDENT LIKE APPLICANT (
        GroupID int
);
```

Копируются атрибуты с типами данных, NOT NULL ограничения.

Для настройки копируемых частей:

INCLUDING DEFAULTS, INCLUDING CONSTRAINTS, ...

ИТМО ВТ

# Последовательности

Используются для генерации значений на основе заданной логики:

CREATE [ options ] SEQUENCE INDEX_NAME

[ INCREMENT [ BY ] incValue ]

[ MINVALUE minValue | NO MINVALUE ]

[ MAXVALUE maxValue | NO MAXVALUE ]

[ START [ WITH ] startValue ]

[ CACHE cache ] [ [ NO ] CYCLE ]

...

# Последовательности

```
CREATE SEQUENCE studentIDSeq
        START WITH 1
        INCREMENT BY 2
        MAXVALUE 10000
        NOCACHE;
```

Функции для управления: currval, nextval, setval, lastval

```
SELECT currval('studentIDSeq');
```

ИТМО ВТ

# Последовательности

```
INSERT INTO STUDENT(StudentID, Name, Surname)
VALUES (
        nextval('studentIDSeq'),
        'Ivan',
        'Ivanov'
);
```

ИТМО ВТ

# Последовательности

```
CREATE TABLE STUDENT(
    StudentID int DEFAULT nextval('studentIDSeq')
                    PRIMARY KEY,
    Name text,
    Surname text
);
```

ИТМО ВТ

# serial

Тип для генерации значений на основе последовательности.

3 вида:

- smallserial — на основе smallint;
- serial — на основе int;
- bigserial — на основе bigint;

ИТМО ВТ

# serial

```
CREATE TABLE STUDENT(
        StudentID serial,
        Name text,
        Surname text
);
```

На самом деле создается последовательность, связанная с таблицей STUDENT.

ИТМО ВТ

# Виртуальные таблицы

***Виртуальные*** — представления, неименованные таблицы (таблицы, которые не существуют постоянно в базе данных).

Можно построить на основе **запроса**.

ИТМО ВТ

# Литература

При подготовке презентации использовались материалы из:

- Введение в реляционные базы данных / В. В. Кириллов, Г. Ю. Громов, Издательство: BHV, 2009 г.
- Документация PostgreSQL.

  https://www.postgresql.org/about/licence/