

Языки прикладного программирования

Лекция 13

```
    for object to mirror
    mirror_mod.mirror_object = object

    if operation == "MIRROR_X":
        mirror_mod.use_x = True
        mirror_mod.use_y = False
        mirror_mod.use_z = False
    elif operation == "MIRROR_Y":
        mirror_mod.use_x = False
        mirror_mod.use_y = True
        mirror_mod.use_z = False
    elif operation == "MIRROR_Z":
        mirror_mod.use_x = False
        mirror_mod.use_y = False
        mirror_mod.use_z = True

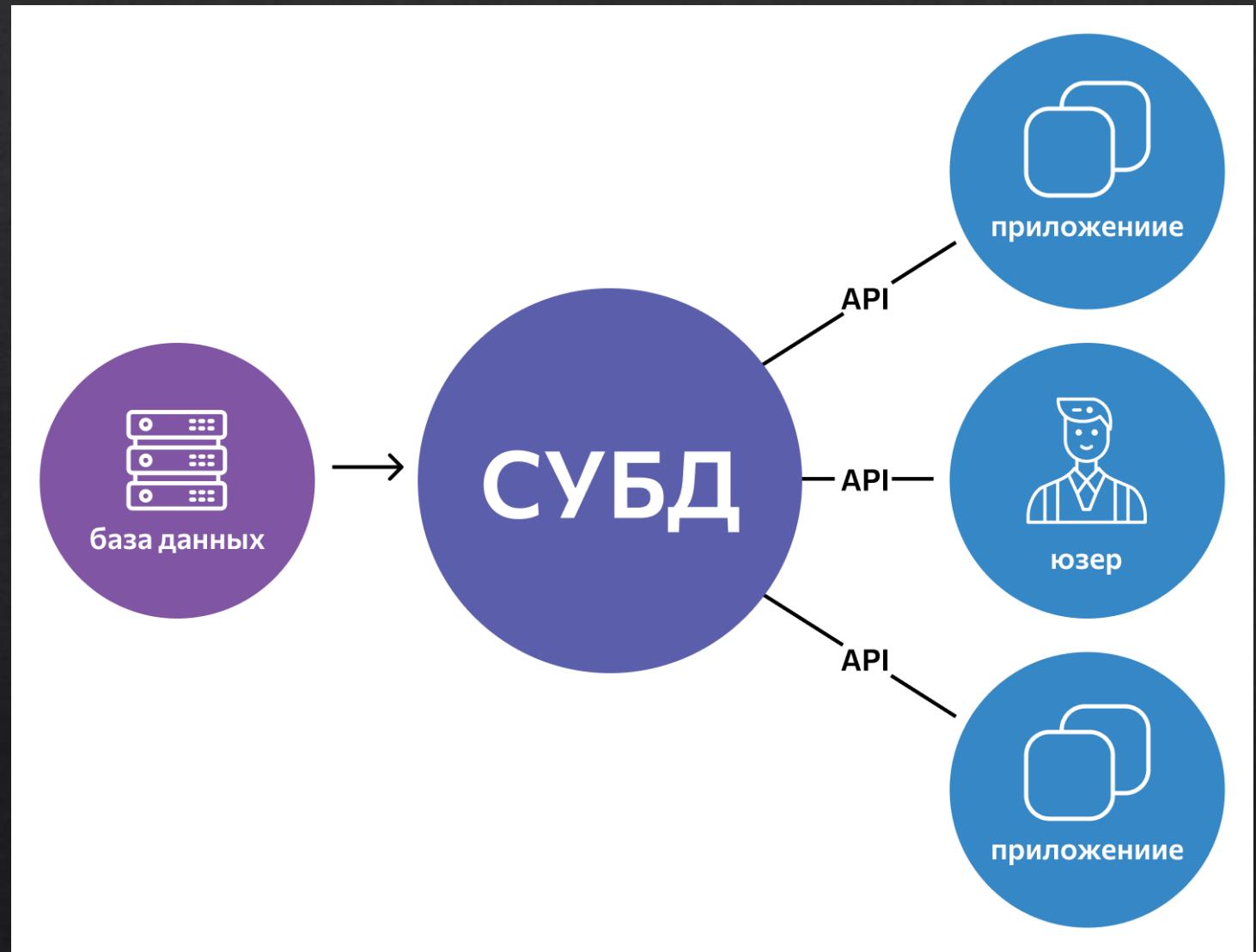
    selection at the end - add
    mirror_ob.select= 1
    mirror_ob.select=1
    context.scene.objects.active = mirror_ob
    ("Selected" + str(modifier))
    mirror_ob.select = 0
    bpy.context.selected_objects.append(mirror_ob)
    data.objects[one.name].select = 1
    print("please select exactly one object")

- OPERATOR CLASSES -
types.Operator:
    X mirror to the selected object.mirror_mirror_x"
    "mirror X"
```

Базы Данных

Что такое База данных?

- ❖ База данных (БД) это:
 - ❖ имеющая название совокупность данных, которая отражает состояние объектов и их отношений в рассматриваемой предметной области.
 - ❖ Упорядоченный набор структурированной информации или данных, которые хранятся в электронном виде в компьютерной системе.
 - ❖ ...
- ❖ Определений много, но важно понимать, что БД:
 - ❖ Это структурированная совокупность данных
 - ❖ Хранится в информационной системе (зачастую в долговременной памяти)
- ❖ Полезная [статья](#) на хабре



Что такое СУБД?

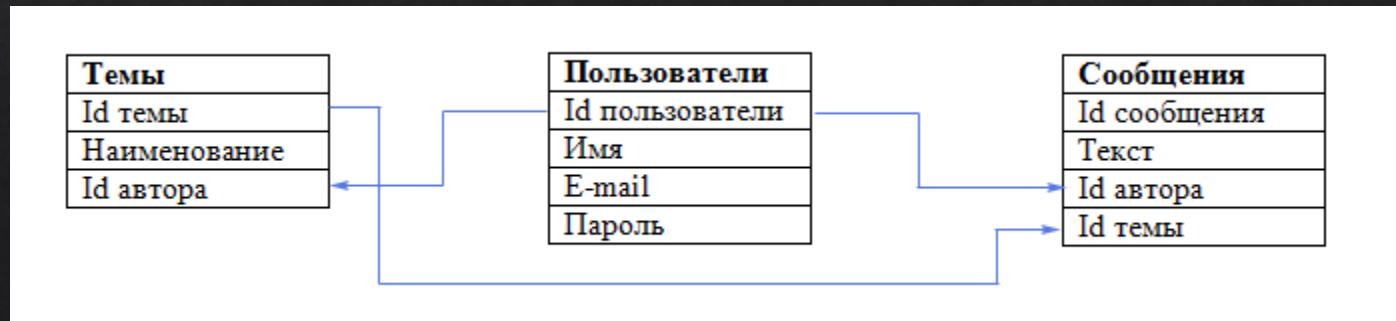
- ❖ СУБД (система управления базами данных) – это ПО, предназначенное для работы с базами данных.
- ❖ СУБД:
 - ❖ Это интерфейс между БД и пользователем/ПО использующим эту БД.
 - ❖ Предоставляет возможность получать, обновлять информацию, а также управлять ее упорядочением и оптимизацией.
 - ❖ Обеспечивает контроль и управление данными (администрирование, контроль, мониторинг, настройка, восстановление и т.п.).

Типы баз данных

- ❖ Реляционные (SQL)
- ❖ Не реляционные (NoSQL)
 - ❖ Документные
 - ❖ Графовые
 - ❖ Колоночные
 - ❖ Ключ-значение
 - ❖ ...

Реляционные БД

- ❖ Данные организуются в виде набора таблиц, состоящих из столбцов и строк.
- ❖ Каждый столбец имеет строго определенный тип данных.
- ❖ Каждая строка таблицы - набор связанных значений, относящихся к одному объекту или сущности.
- ❖ Ячейка – значение атрибута сущности.

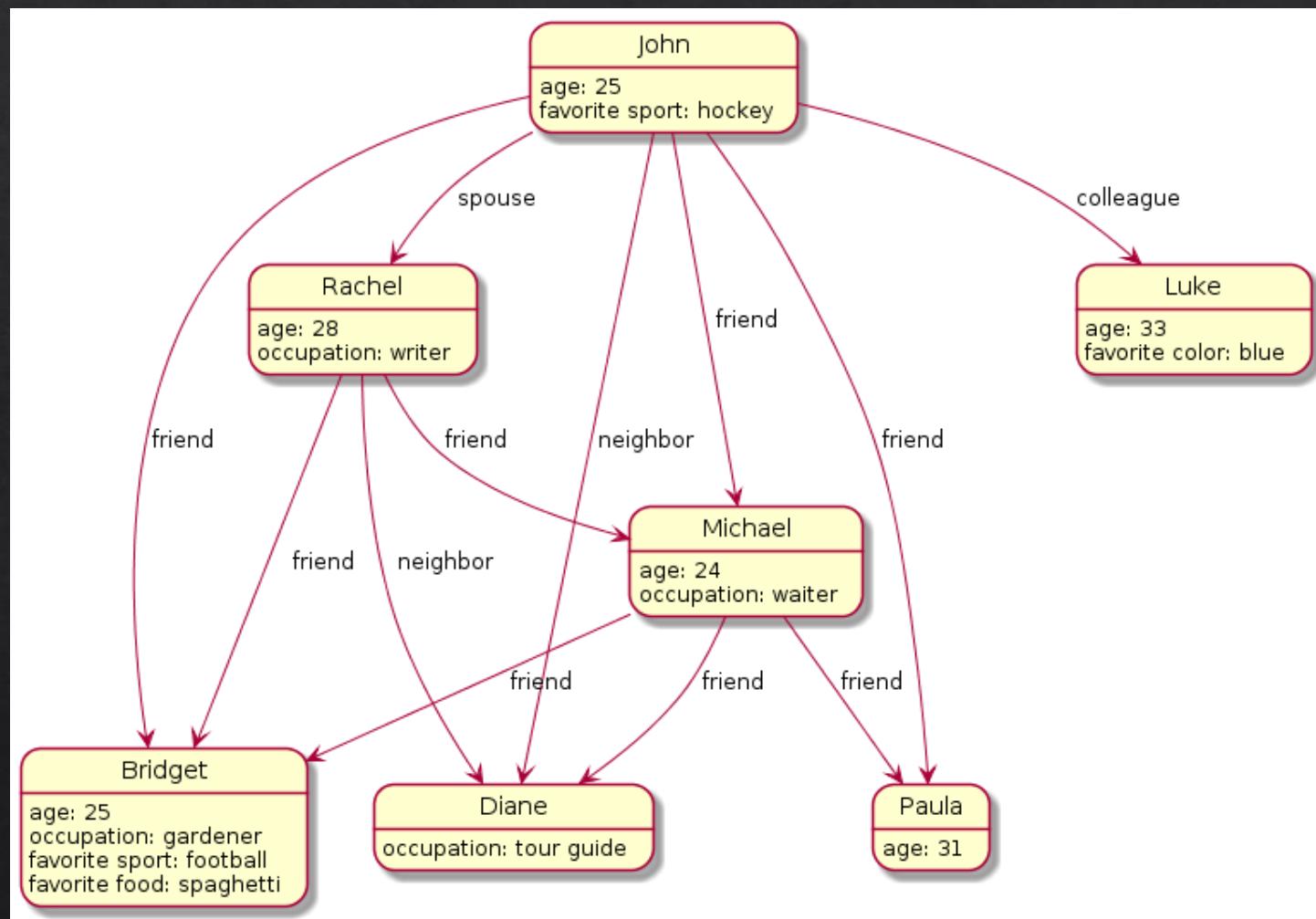


Документные/объектные БД

```
1  {
2      "address": {
3          "building": "1007",
4          "coord": [ -73.856077, 40.848447 ],
5          "street": "Morris Park Ave",
6          "zipcode": "10462"
7      },
8      "borough": "Bronx",
9      "cuisine": "Bakery",
10     "grades": [
11         { "date": { "$date": 1393804800000 }, "grade": "A", "score": 2 },
12         { "date": { "$date": 1378857600000 }, "grade": "A", "score": 6 },
13         { "date": { "$date": 1358985600000 }, "grade": "A", "score": 10 },
14         { "date": { "$date": 1322006400000 }, "grade": "A", "score": 9 },
15         { "date": { "$date": 1299715200000 }, "grade": "B", "score": 14 }
16     ],
17     "name": "Morris Park Bake Shop",
18     "restaurant_id": "30075445"
19 }
```

- ❖ Хранятся коллекции документов/объектов
- ❖ В общем случае в одной коллекции могут быть разные типы/форматы объектов
- ❖ Хранят данные в форматах JSON (BSON), XML и т.п.
- ❖ Характерные представители:
 - ❖ MongoDB
 - ❖ Amazon DocumentDB
 - ❖ RethinkDB

Графовые БД



- ❖ Данные хранятся в виде графа
- ❖ Фокус на отношениях между сущностями
- ❖ Примеры:
 - ❖ Neo4j
 - ❖ JanusGraph
 - ❖ Dgraph



Колоночные БД

- ❖ Данные хранятся по колонкам, а не строкам.
 - ❖ Дешевое чтение, добавление компактное хранение
 - ❖ Дорогое удаление и изменение
-
- ❖ Примеры:
 - ❖ ClickHouse
 - ❖ Cassandra

БД типа ключ-значение

- ❖ Совокупность пар ключ-значение (словарь/hash-map).
- ❖ Как ключи, так и значения могут представлять собой что угодно: от простых до сложных составных объектов.

- ❖ Примеры:
 - ❖ Redis
 - ❖ DynamoDB

Реляционные БД

Чуть подробнее...

Реляционная Модель

- ❖ Данные организованы в виде набора таблиц, состоящих из столбцов и строк.
- ❖ В таблицах хранится информация об «объектах», представленных в БД.
- ❖ Каждая строка таблицы представляет собой набор связанных значений, относящихся к одному «объекту», или сущности.

ACID

- ❖ ACID - набор требований к системе, обеспечивающий наиболее надёжную и предсказуемую её работу:
 - ❖ атомарность,
 - ❖ согласованность,
 - ❖ изоляция,
 - ❖ надежность;
- ❖ Требования сформулированы в конце 1970-х годов.
- ❖ Реляционные СУБД соответствуют требованиям ACID, не реляционные, зачастую – нет.
- ❖ <https://habr.com/ru/post/555920/>

Атомарность

- ❖ **Транзакция** – это одно или несколько действий, выполненных в виде последовательности операций, представляющих собой единую логическую задачу.
- ❖ **Атомарность** – это условие, при котором либо транзакция успешно выполняется целиком, либо, если какая-либо из ее частей не выполняется, вся транзакция отменяется.
- ❖ Т.е. транзакция – неделимое (атомарное) действие.

Согласованность

- ❖ Согласованность – это условие, при котором данные, записываемые в базу данных в рамках транзакции, должны быть целостными т.е.
 - ❖ соответствовать набору ограничений, определенных предметной областью.
 - ❖ Исключать дублирование данных, относящихся к одним сущностям.
- ❖ Поддерживается за счет:
 - ❖ Первичных ключей,
 - ❖ Внешних ключей,
 - ❖ Ограничений на значения столбцов
- ❖ Ограничения позволяют применять правила предметной области к данным в таблицах и гарантировать точность и надежность данных.
- ❖ Большинство СУБД также поддерживает триггеры - SQL-скрипты, которые выполняются в ответ на определенные операции в БД (вставку, изменение).

Первичный и внешний ключи

- ❖ Каждая сущность (строка) имеет уникальный идентификатор, называемый **первичным ключом**.
- ❖ Стока одной таблицы может быть связана с данными из другой при помощи **внешнего ключа**.

Изоляция

- ❖ **Изоляция** необходима для контроля над согласованностью и гарантирует независимость транзакций друг от друга.

Надежность

- ❖ **Надежность** подразумевает, что все внесенные в базу данных изменения на момент успешного завершения транзакции считаются постоянными.

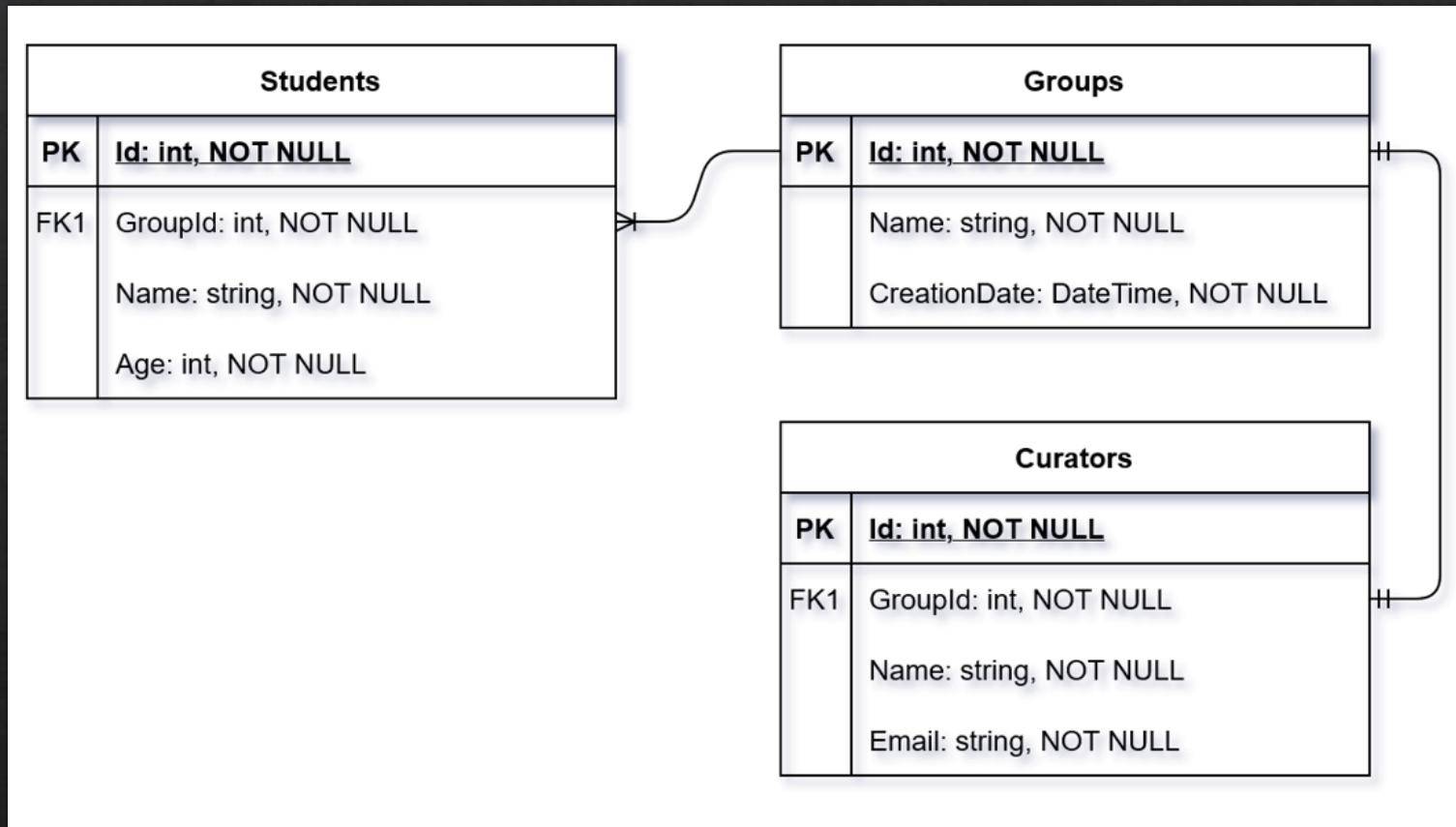
Типы данных

Совместимость: В стандарте SQL описаны следующие типы (или их имена): bigint, bit, bit varying, boolean, char, character varying, character, varchar, date, double precision, integer, interval, numeric, decimal, real, smallint, time (с часовым поясом и без), timestamp (с часовым поясом и без), xml.

Описание Базы данных

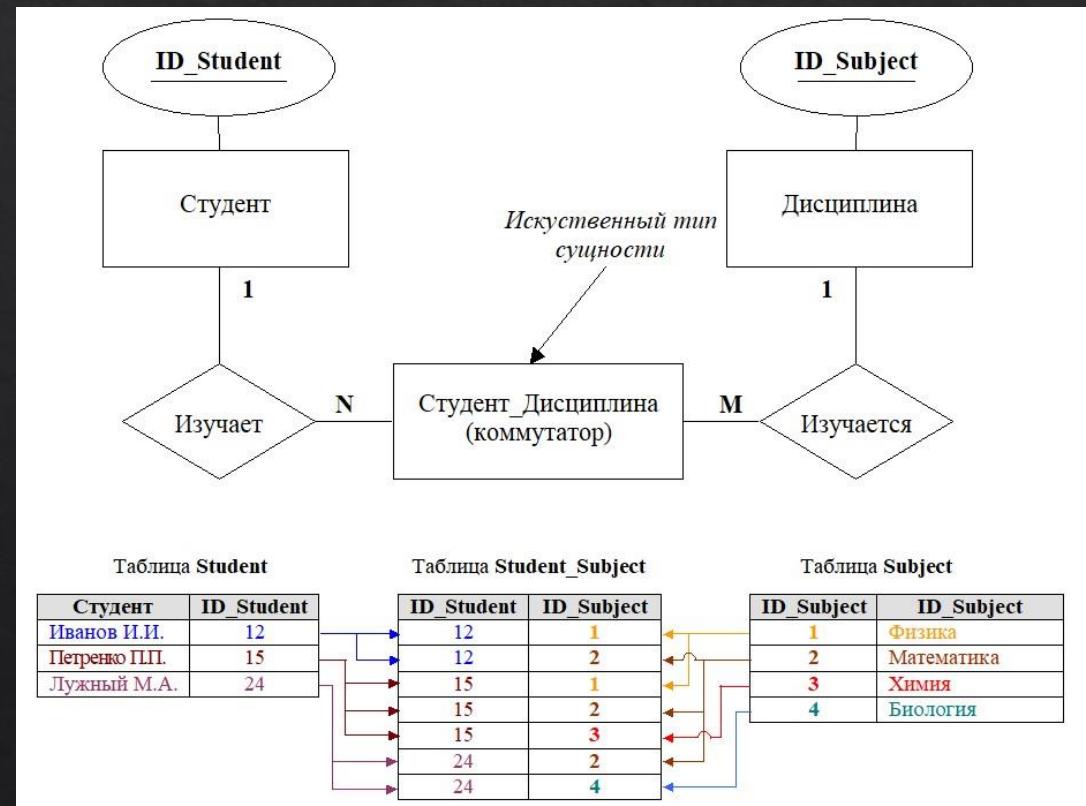
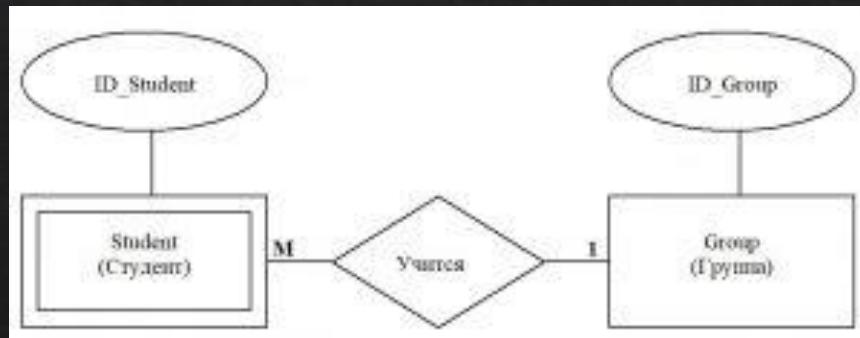
- ❖ Схема базы данных (Database schema) — её структура, описанная на формальном языке, поддерживаемом СУБД. В реляционных базах данных схема определяет таблицы, поля в каждой таблице (обычно с указанием их названия, типа, обязательности), и ограничения целостности (первичный, потенциальные и внешние ключи и другие ограничения).
- ❖ ER-диаграмма (Entity-Relationship. Сущность-Связь) — диаграмма для отображения связи между сущностями в предметной области.

Описание Базы данных



Виды отношений между сущностями

- ❖ Один к одному
- ❖ Один ко многим
- ❖ Многие ко многим



SQL (Structured Query Language)

- ❖ SQL – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

SQL запросы

- ❖ **SELECT** – выбрать строки из таблиц;
- ❖ **INSERT** – добавить строки в таблицу;
- ❖ **UPDATE** – изменить строки в таблице;
- ❖ **DELETE** – удалить строки в таблице;

<https://www.w3schools.com/sql>

SELECT

```
SELECT ('столбцы или * для выбора всех столбцов; обязательно')
FROM ('таблица; обязательно')
WHERE ('условие/фильтрация, например, city = 'Moscow'; необязательно')
GROUP BY ('столбец, по которому хотим сгруппировать данные; необязательно')
HAVING ('условие/фильтрация на уровне сгруппированных данных; необязательно')
ORDER BY ('столбец, по которому хотим отсортировать вывод; необязательно')
```

```
select City, count(CustomerID) from Customers
WHERE Country = 'Germany'
GROUP BY City
```

INSERT

```
INSERT INTO table_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);
```

```
INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)  
VALUES  
('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway'),  
('Greasy Burger', 'Per Olsen', 'Gateveien 15', 'Sandnes', '4306', 'Norway'),  
('Tasty Tee', 'Finn Egan', 'Streetroad 19B', 'Liverpool', 'L1 0AA', 'UK');
```

UPDATE

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

Delete

```
DELETE FROM table_name WHERE condition;
```

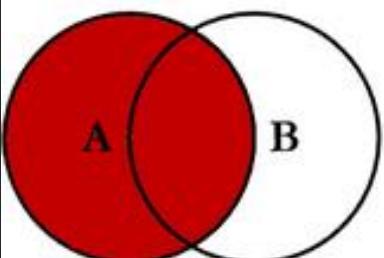
```
DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';
```

JOIN

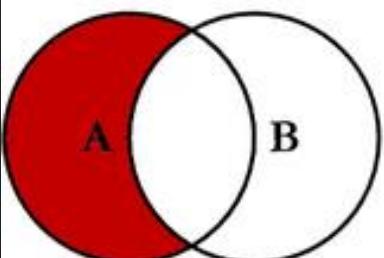
- ❖ JOIN — необязательный элемент, используется для объединения таблиц по ключу, который присутствует в обеих таблицах. Перед ключом ставится оператор ON.

```
select * from Orders
join Customers on Orders.CustomerID = Customers.CustomerID
where Customers.CustomerID >10
```

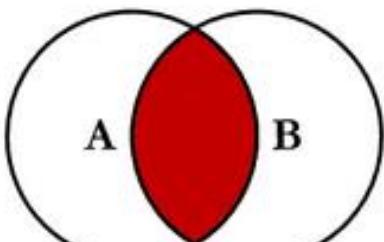
SQL JOINS



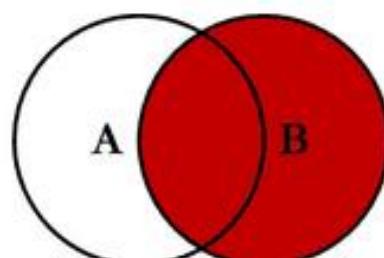
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



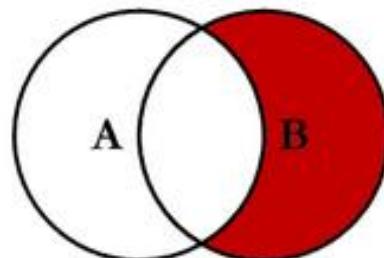
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



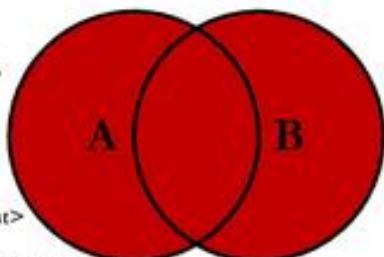
```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



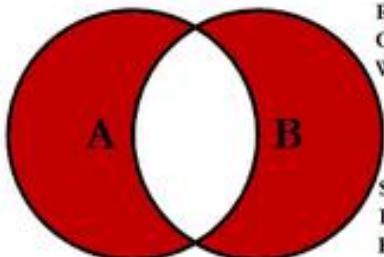
```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Миграция

- ❖ Это план перехода базы данных от старой схемы к новой.

ORM (Object-Relational Mapping)

- ❖ технология, суть которой заключается в создании «виртуальной объектной базы данных» в коде приложения.
- ❖ ORM берет на себя взаимодействие с БД выступая в качестве слоя абстракции, выполняя оптимизации, упрощая программисту работу с БД.

Подходы к работе ORM

- ❖ Database first – подключаемся к существующей БД, по базе данных создаем классы
- ❖ Model first – описывается модель данных, на основании которой создается БД и классы (EDMX)
- ❖ Code first – создание БД на основании описанной в коде модели данных.

Преимущества

- ❖ Модель описана в одном месте. Это упрощает поддержку и повторное использование кода.
- ❖ Разработчику не нужно писать SQL и настраивать структуру БД, ускоряет разработку
- ❖ Защита от стандартных уязвимостей приложений (SQL injection)
- ❖ Дополнительный слой абстракции (зачастую только в рамках реляционных баз) – позволяющий с легкостью изменять СУБД не изменяя код

Недостатки

- ❖ Медленнее чем хорошо написанный SQL
- ❖ В высоконагруженных сценариях необходимо изучить нюансы ORM фреймворка который используете

EF Core

- ❖ Entity Framework (EF) Core — кроссплатформенная и расширяемая технология доступа к данным с открытым исходным кодом для .NET
- ❖ Более новая версия Entity Framework, разработанного для .NET Framework

Модель в EF Core

- ❖ Модель состоит из классов сущностей и объекта контекста, который представляет сеанс взаимодействия с базой данных. Объект контекста позволяет выполнять запросы и сохранять данные.
- ❖ Способы создания моделей:
 - ❖ Data annotation – настройка сопоставления моделей и таблиц с помощью атрибутов.
 - ❖ Fluent API – настройка сопоставления моделей и таблиц с помощью набора методов специализированного API.

А дальше
проще
показывать...

❖ Почитать можно:

- ❖ <https://learn.microsoft.com/ru-ru/ef/core/>
- ❖ <https://metanit.com/sharp/entityframeworkcore/>

Навигационные свойства

- ❖ Применяются для связи между моделями
- ❖ По умолчанию они не будут заполнены при получении данных, для подгрузки – используется `Include`, если уровней много – `ThenInclude`, это приводит к выполнению JOIN запросов!

```
var elrond = context.Characters.Include(x => x.ReceivedDuels).First(x => x.Race == Race.Elf);

foreach (var character in elrond.ReceivedDuels!.ToList())
    Console.WriteLine(character);
```