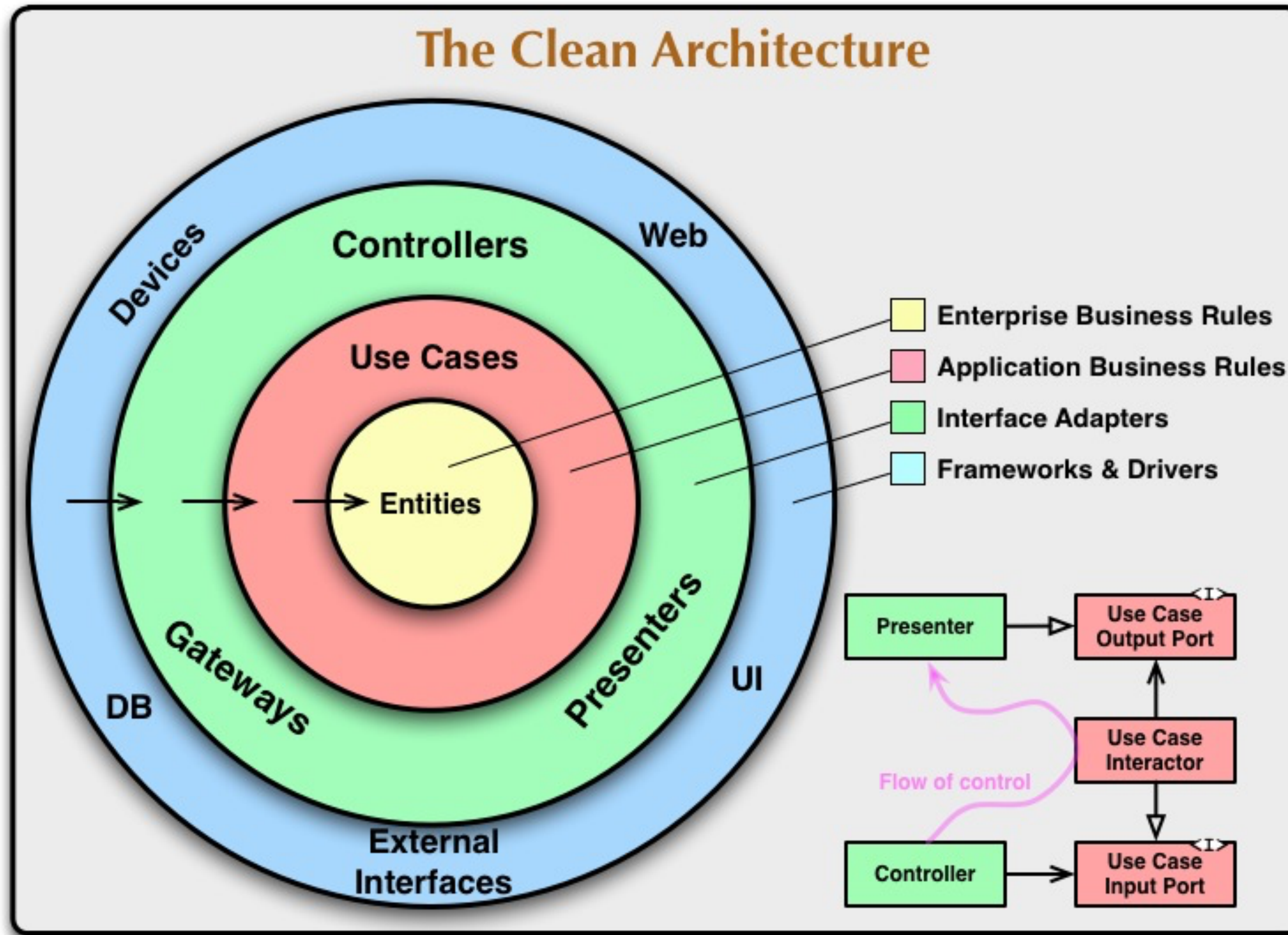


Entity Framework

+ чистая архитектура

14.11.2025, ФКН ПИ

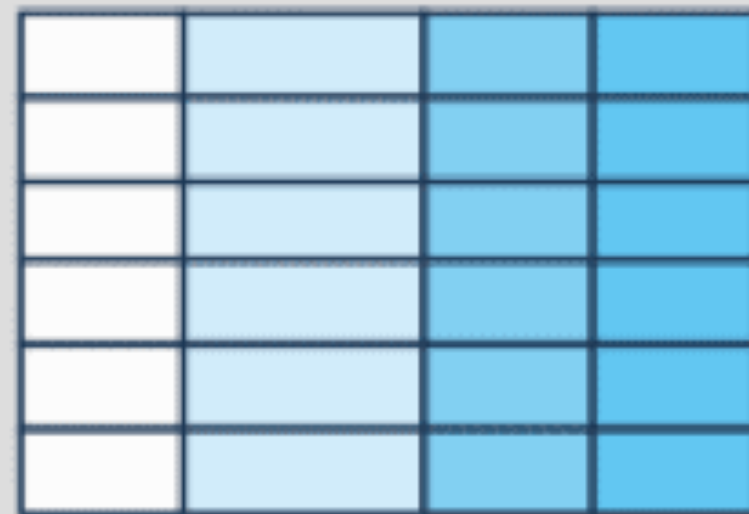
Clean Architecture



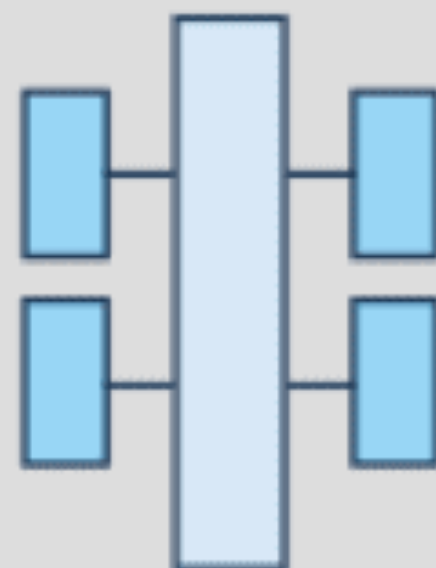
SQL базы данных, реляционные базы данных

SQL

Relational



Analytical (OLAP)



NoSQL

Key-Value



Column-Family



Graph



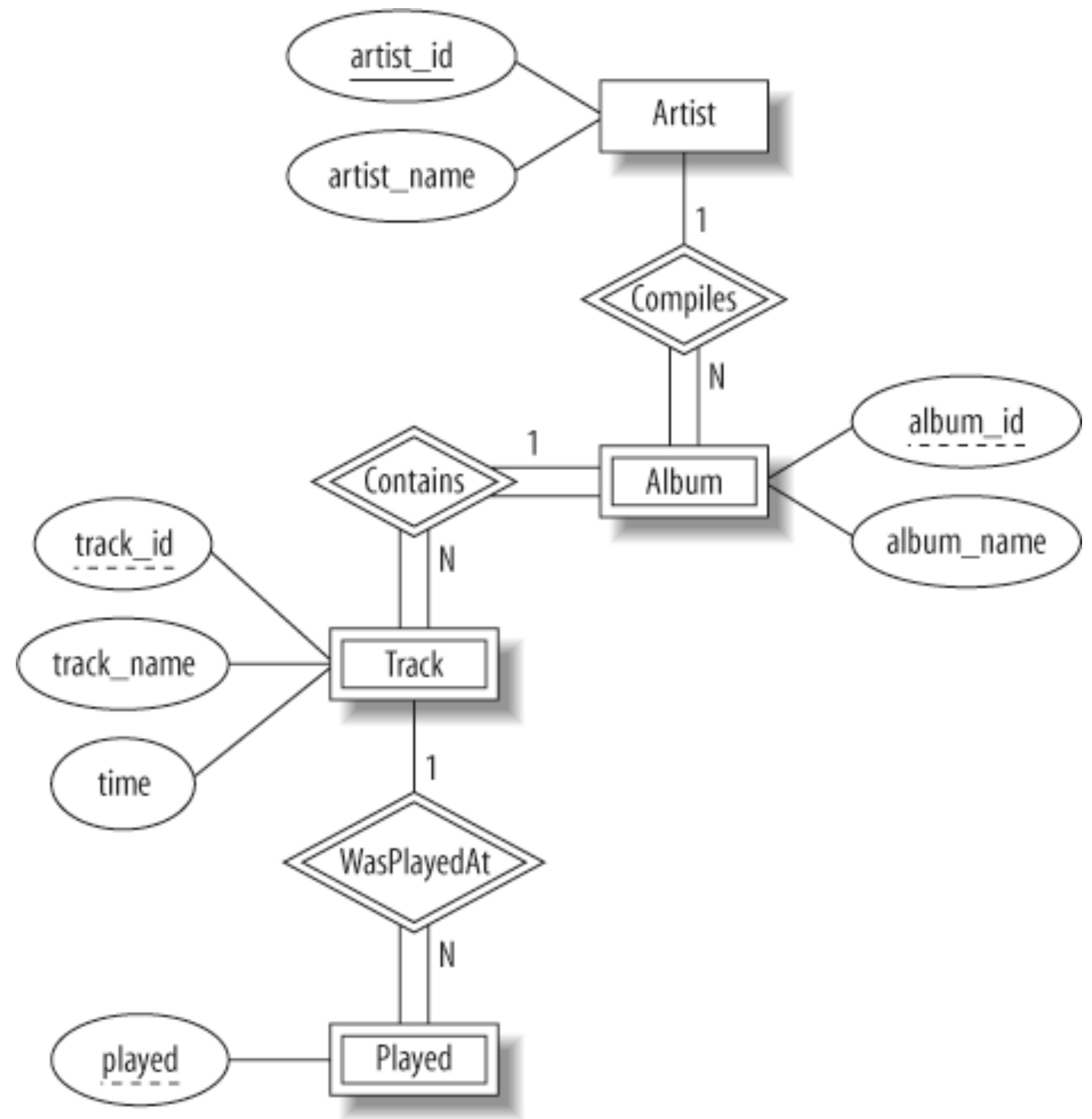
Document



ER-моделирование

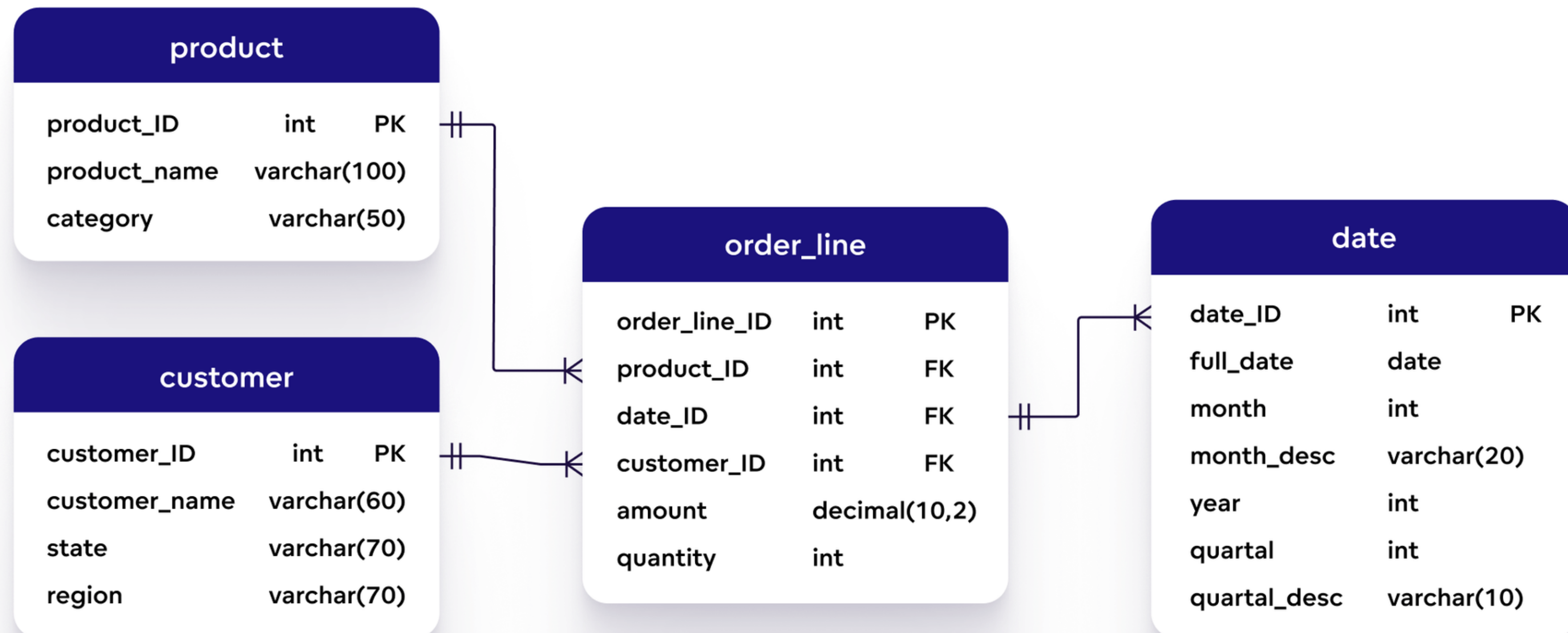
Описывает сущности и отношения между ними.

Элементы: сущности, атрибуты (некоторые из них – ключи), отношения, кратности связей.



Physical Data Model

Физическая модель данных показывает реальное устройство таблиц в БД, сущности и связи.



PK - Primary key FK - Foreign key
int, decimal, varchar, date - data types

SQL. Create

```
CREATE TABLE Persons (  
    PersonID int,  
    LastName varchar(255),  
    FirstName varchar(255),  
    Address varchar(255),  
    City varchar(255)  
);
```

PersonID	LastName	FirstName	Address	City

SQL. Insert

```
INSERT INTO Customers (CustomerName, ContactName, Address, City,
PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger',
'4006', 'Norway');
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
92	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway

SQL. Insert

```
INSERT INTO Customers (CustomerName, ContactName, Address, City,
PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger',
'4006', 'Norway');
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
92	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway

SQL. Select

SELECT CustomerName, City FROM Customers;

SELECT * FROM Customers;

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

SQL. Update

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

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

SQL. Delete

DELETE FROM Customers;

DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Описание проекта

API, которое позволяет работать с песнями и музыкальными альбомами: осуществлять по ним поиск и получать информацию о них.

Кроме того, это API содержит функциональность, связанную с пользователями (возможность вести список понравившихся пользователю альбомов)*.

Теперь нам пора перенести все хранимые в память сущности из памяти в БД.



Настройка



- Убедиться, что на машине установлена и запущена СУБД
- Найти драйвер для конкретной СУБД, добавить его в зависимости проекта
- Можно работать!

Заполняем БД с помощью SQL-запросов

Этот шаг необязателен, но хорошо бы, чтобы на момент работы с БД в ней что-то уже лежало.

А еще это возможность посмотреть на то, как можно выполнить SQL-запрос в IDEA. Для этого нужно создать файл .sql, ввести запрос (или несколько запросов), и отправить файл на выполнение.

```
1  create table album(  
2     id text primary key,  
3     title text,  
4     artist text,  
5     image_url text  
6 );  
7  
8  create table songs(  
9     id text primary key,  
10    title text,  
11    album_id text,  
12    constraint fk_album  
13        foreign key(album_id)  
14        references album(id)  
15 );  
16
```

```
1  insert into album(id, title, artist, image_url)  
2 values ('id1', 'Блэкаут', 'Операция Пластин', 'some url'),  
3 ('id2', 'Scaled and Icy', 'Twenty One Pilots', 'some url1'),  
4 ('id3', 'Punisher', 'Phoebe Bridgers', 'some url2');  
5  
6  insert into songs(id, title, album_id)  
7 values ('song1', 'Хоровод', 'id1'),  
8 ('song2', 'Saturday', 'id2'),  
9 ('song3', 'Moon Song', 'id3');
```

Statement, prepared statement

```
private fun getConnection() =  
    DriverManager.getConnection(PropertiesUtil.get(URL_KEY))
```

```
fun performExecute(sql: String): Boolean {  
    val c = getConnection()  
    try {  
        c.use {  
            val query = c.prepareStatement(sql)  
            return query.execute()  
        }  
    } catch (e: Exception) {  
        c.close()  
  
        println(e.message)  
        return false  
    }  
}
```

boolean execute() - create/drop
ResultSet executeQuery() - select
int executeUpdate() - update/insert/delete и т.д.

Выполняем запросы

```
val createTableRequest = """
    create table if not exists account (
        id serial primary key,
        email varchar(256),
        address varchar(256));
    """.trimIndent();
```

```
val selectByAddressRequest = """
    select * from account
    where address = ?
    """.trimIndent()
```

```
val insertRequest = """
    insert into account (email, address)
    values ('test0@hse.ru', 'random str.'),
    ('test1@hse.ru', 'random str.'),
    ('test2@hse.ru', 'random2 str.'),
    ('test3@hse.ru', 'random3 str. ');
    """.trimIndent()
```

ORM (Object Relational Mapping)

Relational database (such as PostgreSQL or MySQL)

ID	FIRST_NAME	LAST_NAME	PHONE
1	John	Connor	+16105551234
2	Matt	Makai	+12025555689
3	Sarah	Smith	+19735554512
...

Python objects

```
class Person:  
    first_name = "John"  
    last_name = "Connor"  
    phone_number = "+16105551234"
```

```
class Person:  
    first_name = "Matt"  
    last_name = "Makai"  
    phone_number = "+12025555689"
```

```
class Person:  
    first_name = "Sarah"  
    last_name = "Smith"  
    phone_number = "+19735554512"
```

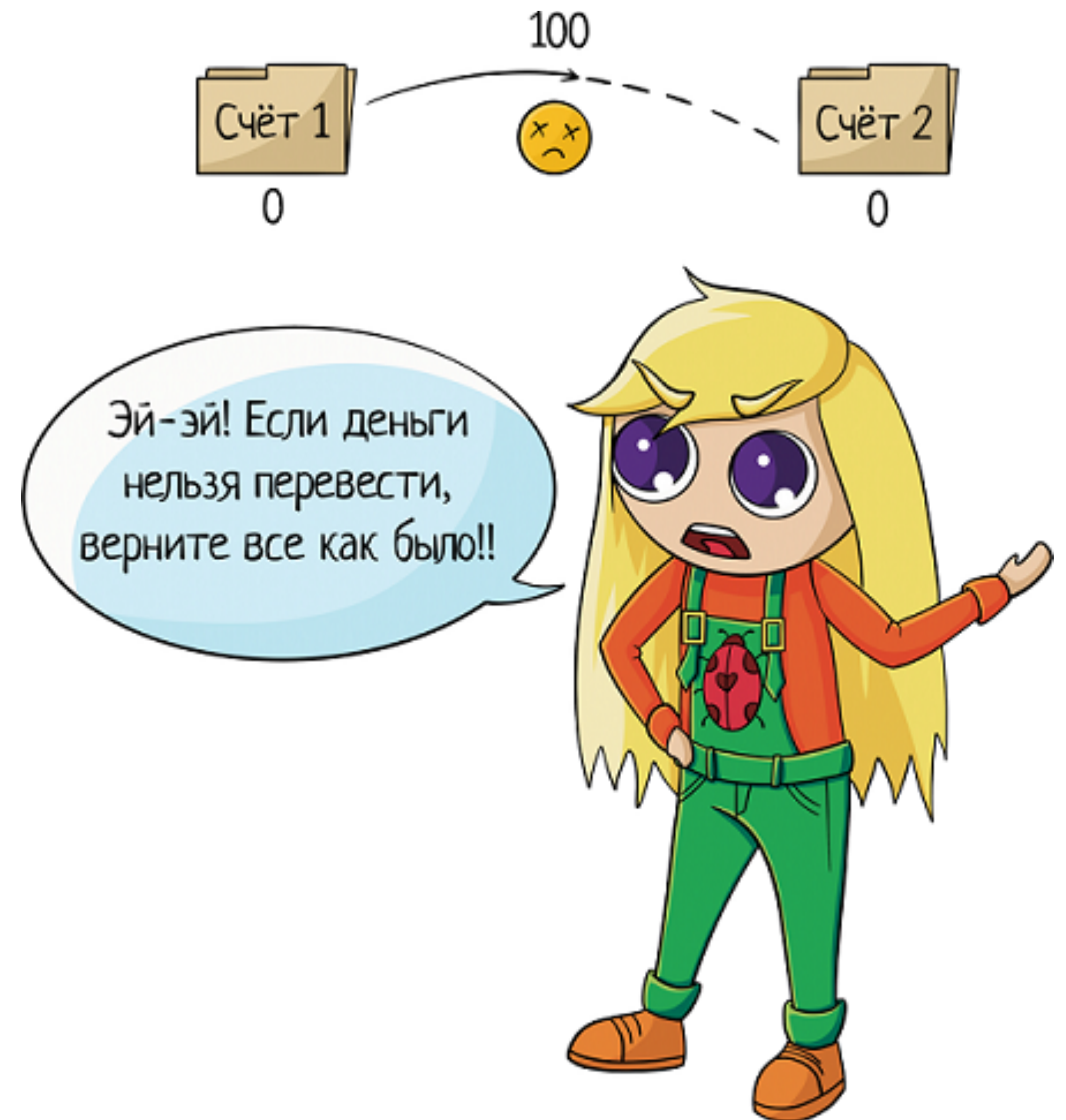
ORMs provide a bridge between
**relational database tables, relationships
and fields** and **Python objects**

Транзакции

Транзакции

Транзакция – набор операций над базой данных, которые объединены в одну атомарную операцию.

Статья: <https://habr.com/ru/articles/537594/>



Пример транзакции

C#

Copy

```
using var context = new BloggingContext();
await using var transaction = await context.Database.BeginTransactionAsync();

try
{
    context.Blogs.Add(new Blog { Url = "https://devblogs.microsoft.com/dotnet/" });
    await context.SaveChangesAsync();

    await transaction.CreateSavepointAsync("BeforeMoreBlogs");

    context.Blogs.Add(new Blog { Url = "https://devblogs.microsoft.com/visualst" });
    context.Blogs.Add(new Blog { Url = "https://devblogs.microsoft.com/aspnet/" });
    await context.SaveChangesAsync();

    await transaction.CommitAsync();
}
catch (Exception)
{
    // If a failure occurred, we rollback to the savepoint and can continue the
    await transaction.RollbackToSavepointAsync("BeforeMoreBlogs");

    // TODO: Handle failure, possibly retry inserting blogs
}
```

