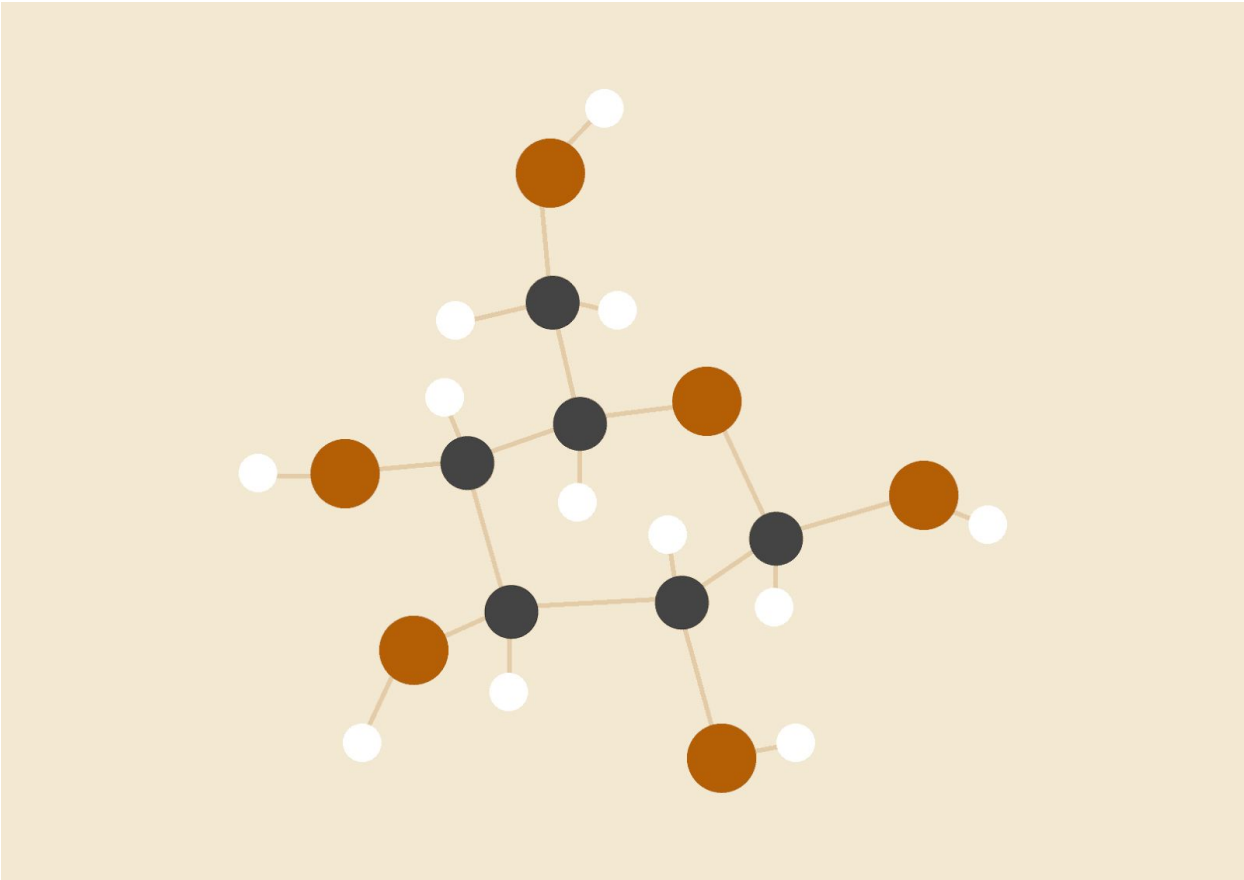
PHARMACY DATABASE DESIGN



P37000018 Luigi Laezza

P37000020 Antonio Nappa

P37000007 Celestino Santagata

UNIVERSITÀ DEGLI STUDI DI NAPOLI - FEDERICO II
DATA MANAGEMENT - M.D. DATA SCIENCE

PART ONE

DATABASE DESIGN

1. INTRODUCTION

The aim of this work is to design a database for the management of a pharmacy warehouse.

The project was implemented through a relational database using Oracle Database version 19c. Then SQL was used for creating, querying and modifying database. The work environment used for is Pycharm, a Python IDE, connected to the database via cx-Oracle, a Python extension module that enables access to Oracle Database and conforms to the Python database API specification; Jetbrains Datagrip was used to display the tables.

In the reality of interest we find eight entities. In particular:

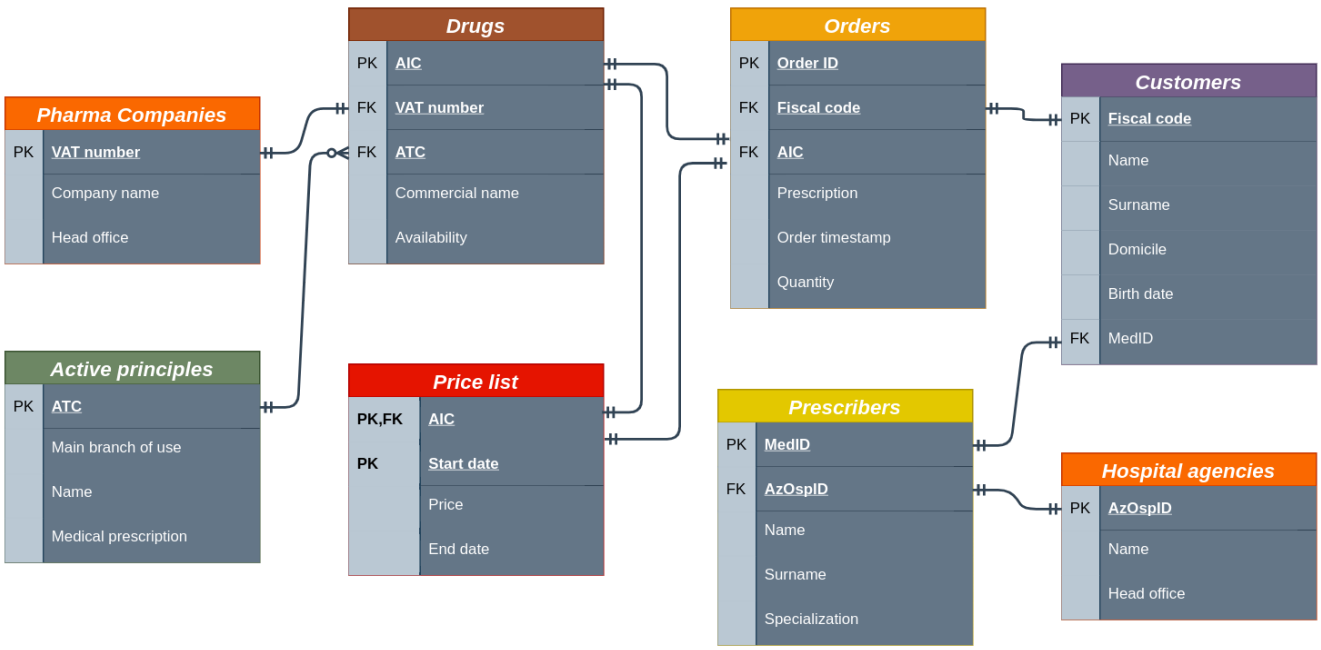
- pharma companies, with name, headquarter and VAT number;
- active principle, with name, unique identification (ATC), pharmaceutical company owning the patent if present and medical prescription;
- drugs, with commercial name, belonging to a specific pharmaceutical company, active principle of which it is composed, unique identifier (AIC) and availability;
- price list;
- orders, identified by an ID and relating to a single drug and a single customer;
- prescribers, the doctor who prescribes the drug which can be generic or specialist and work on its own or for a hospital company;
- customers uniquely identified by the tax code with name, surname, domicile, date of birth and attending physician;
- hospital agencies with ID, name and location.

2. DATABASE DESIGN

The first step in designing the database was the conceptual scheme for a clear view of the reality of interest. Secondly, we moved on to logical design and physical creation of tables in Oracle.

CONCEPTUAL MODEL

The conceptual model allows to represent the reality of interest; it indicates which data is needed to represent. The realization of the conceptual scheme contains the fundamental entities which the database is based on and through which is possible to arrive at the final design of the database.



In particular, in this scheme we can see not only the entities but also the attributes of these entities, which will be then represented in the relational scheme, and the logical relationships between them. The primary keys and foreign keys were also highlighted.

An example of how this would work could be: a customer placing an order with a prescription. In this way, through the recipe it will be possible to buy a specific drug from a specific pharmaceutical company with an active principle that requires a prescription, which was made by a specialist who works for a hospital or is a freelancer. All these features are implemented within the provided script ('database_creation.py').

RELATIONAL MODEL

The next stage is the logical design which foresees the definition of the scheme in a whole of constraints and relationships (or tables). It goes through two preliminary stages, which are:

- transformation: in this phase, some constructs of the conceptual model must be simplified such as compound and multivalued attributes, and the resolution of generalizations;
- translation: it consists in the effective translation of the conceptual model in the relational scheme.

```
import cx_Oracle

def database_connection(hostname, port_number, service_name, username, password):
    dsn = cx_Oracle.makedsn(hostname, port_number, service_name)
    connection = cx_Oracle.connect(username, password, dsn)
    return connection

def tables_deletion(cursor,list_of_tables):
    '''drop tables in list_of_tables'''
    for table in list_of_tables:
        cursor.execute(f"""BEGIN
                             EXECUTE IMMEDIATE 'DROP TABLE {table}';
                           EXCEPTION
                             WHEN OTHERS THEN NULL;
                           END;""")
connection = database_connection('127.0.0.1', '1521', 'orcl', 'SYSTEM', '******')
cursor = connection.cursor()

list_of_tables = ['ORDERS','CUSTOMERS','PRESCRIBERS','HOSPITAL_AGENCIES',
                  'PRICE_LIST','DRUGS','ACTIVE_PRINCIPLES','PHARMA_COMPANIES']
tables_deletion(cursor, list_of_tables)
cursor.execute("""BEGIN
                    EXECUTE IMMEDIATE 'DROP TABLE ODERS';
                  EXCEPTION
                    WHEN OTHERS THEN NULL;
                  END;""")
cursor.execute("""BEGIN
                    EXECUTE IMMEDIATE 'DROP TABLE CUSTOMERS';
                  EXCEPTION
                    WHEN OTHERS THEN NULL;
                  END;""")
cursor.execute("""BEGIN
                    EXECUTE IMMEDIATE 'DROP TABLE PRESCRIBERS';
                  EXCEPTION
                    WHEN OTHERS THEN NULL;
                  END;""")
cursor.execute("""BEGIN
                    EXECUTE IMMEDIATE 'DROP TABLE HOSPITAL_AGENCIES';
                  EXCEPTION
                    WHEN OTHERS THEN NULL;
                  END;""")
```

```
cursor.execute("""BEGIN
                    EXECUTE IMMEDIATE 'DROP TABLE PRICE_LIST';
                  EXCEPTION
                    WHEN OTHERS THEN NULL;
                  END;""")
cursor.execute("""BEGIN
                    EXECUTE IMMEDIATE 'DROP TABLE DRUGS';
                  EXCEPTION
                    WHEN OTHERS THEN NULL;
                  END;""")
cursor.execute("""BEGIN
                    EXECUTE IMMEDIATE 'DROP TABLE ACTIVE_PRINCIPLES';
                  EXCEPTION
                    WHEN OTHERS THEN NULL;
                  END;""")
cursor.execute("""BEGIN
                    EXECUTE IMMEDIATE 'DROP TABLE PHARMA_COMPANIES';
                  EXCEPTION
                    WHEN OTHERS THEN NULL;
                  END;""")
cursor.execute("""
    CREATE TABLE PHARMA_COMPANIES
    (
            Company_name varchar2(60),
            Head_office varchar2(50),
            VAT_number char(11) NOT NULL,
            PRIMARY KEY(VAT_number)
    )
    """)
cursor.execute("""
    CREATE TABLE ACTIVE_PRINCIPLES
    (
        ATC char(7) NOT NULL,
            Name varchar2(50),
            Main_branch_of_use varchar2(20),
            Medical_prescription numeric(1) CHECK(Medical_prescription=0 OR Medical_prescription=1),
            PRIMARY KEY(ATC)
    )
    """)
cursor.execute("""
    CREATE TABLE DRUGS
    (
            AIC char(9) NOT NULL,
            Commercial_name varchar2(70),
            ATC char(7),
            Availability numeric(1) CHECK(Availability=0 OR Availability=1),
            Pharma_company char(11),
        PRIMARY KEY(AIC),
            FOREIGN KEY(Pharma_company) REFERENCES PHARMA_COMPANIES(VAT_number) ON DELETE CASCADE,
            FOREIGN KEY(ATC) REFERENCES ACTIVE_PRINCIPLES(ATC) ON DELETE CASCADE
    )
    """)
cursor.execute("""
    CREATE TABLE PRICE_LIST
    (
            AIC char(9) NOT NULL,
            Start_date date NOT NULL,
            Price numeric(*,2) CHECK(price>0),
            End_date date,
        PRIMARY KEY(AIC,Start_date),
        FOREIGN KEY(AIC) REFERENCES DRUGS(AIC) ON DELETE CASCADE
        )
        """)
```

```
cursor.execute("""
    CREATE TABLE HOSPITAL_AGENCIES
    (
            AzOspID int NOT NULL,   --AUTO_INCREMENT,
            Name varchar2(50),
            Head_office varchar2(50),
            PRIMARY KEY(AzOspID)
    )
    """)
cursor.execute("""
    CREATE TABLE PRESCRIBERS
    (
            MedID int NOT NULL,   --AUTO_INCREMENT,
            Name varchar2(50),
            Surname varchar2(50),
            Specialization varchar2(50),
            Hospital_agency int,
        PRIMARY KEY(MedID),
        FOREIGN KEY(Hospital_agency) REFERENCES HOSPITAL_AGENCIES(AzOspID) ON DELETE SET NULL
    )
    """)
cursor.execute("""
    CREATE TABLE CUSTOMERS
    (
        Fiscal_code char(16) NOT NULL,
            Name varchar2(50),
        Surname varchar2(50),
            Birth_date date,
            Domicile varchar2(50),
            Prescriber int,
        PRIMARY KEY(Fiscal_code),
            FOREIGN KEY(Prescriber) REFERENCES PRESCRIBERS(MedID) ON DELETE SET NULL
    )
    """)
cursor.execute("""
    CREATE TABLE ORDERS
    (
        OrderID int NOT NULL,   --AUTO_INCREMENT,
            Order_date_and_time timestamp(0),     --DATE 'YYYY-MM-DD'
            --Order_date_and_time timestamp(0),   --TIMESTAMP(fractional_seconds_precision)
                                                  --TIMESTAMP 'YYYY-MM-DD HH24:MI:SS.FF'
        Fiscal_code char(16),
        Prescription numeric(1) CHECK(Prescription=0 OR Prescription=1),
            Drug char(9),
            Quantity int,
        PRIMARY KEY(OrderID),
            FOREIGN KEY(Fiscal_code) REFERENCES CUSTOMERS(Fiscal_code), --ON DELETE NO ACTION,
            FOREIGN KEY(Drug) REFERENCES DRUGS(AIC) --ON DELETE NO ACTION
    )
    """)

connection.close()
```

3. FINDER

‘Finder.py’ is a script that searches by drug or by active principle and then: in the case of a drug, provides drug characteristics, pharmaceutical company, availability; instead, with an active principle, provides ATC, main branch of use and prescription requirement.

```
finder_condition = condition_setting()
if finder_condition:
    loop_condition = 'y'
    while loop_condition == 'y':
        selector = input("""What are you looking for?
        - Drug
        - Active principle
        - 0 to exit
        """).lower()
        if selector == 'drug':
            feature = input("Search by name or AIC? ").lower()
            if feature == 'name':
                drug = input("Insert the drug name: ").upper()
                sql_string = f"SELECT * FROM DRUGS WHERE Commercial_name LIKE '%{drug}%'"
                print(sql_string)
                cursor.execute(sql_string)
                for result in cursor.fetchall():
                    print(result)
                    ATC_decoder(result[2])
                loop_condition = input("""Continue? (y/n) """).lower()
            elif feature == 'aic':
                drug = input("Insert the AIC: ")  # .lower()
                sql_string = f"SELECT * FROM DRUGS WHERE AIC = '{drug}'"
                print(sql_string)
                cursor.execute(sql_string)
                for result in cursor.fetchall():
                    print(result)
                    ATC_decoder(result[2])
                loop_condition = input("""Continue? (y/n) """).lower()
            else:
                print("Error, please try again.")
        elif selector == 'active principle':
            feature = input("Search by name or ATC? ").lower()
            if feature == 'name':
                active_principle = input("Insert the active principle name: ")  # .lower()
                sql_string = f"SELECT * FROM ACTIVE_PRINCIPLES WHERE Name LIKE '%{active_principle}%'"
                print(sql_string)
                cursor.execute(sql_string)
                for result in cursor.fetchall():
                    print(result)
                    ATC_decoder(result[0])
                loop_condition = input("""Continue? (y/n) """).lower()
            elif feature == 'atc':
                active_principle = input("Insert the ATC: ")  # .lower()
                sql_string = f"SELECT * FROM ACTIVE_PRINCIPLES WHERE ATC = '{active_principle}'"
                print(sql_string)
                cursor.execute(sql_string)
                for result in cursor.fetchall():
                    print(result)
                    ATC_decoder(result[0])
                loop_condition = input("""Continue? (y/n) """).lower()
            else:
                print("Error, please try again.")
        elif selector == '0':
            loop_condition = 'n'
        else:
            print("Error, please try again.")
```

PART TWO

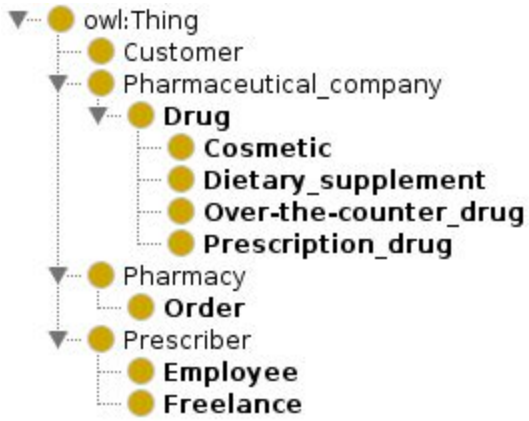
ONTOLOGY

1. INTRODUCTION

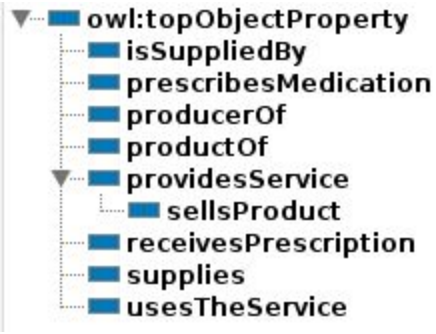
In the field of mathematical logic, a first-order theory is a particular formal system in which it is possible to express sentences and deduce their logical consequences in a completely formal and mechanical way. In computer science, an ontology is a formal, shared and explicit representation of a conceptualization of a domain of interest. More in detail, it is an axiomatic theory of the first order that can be expressed in a descriptive logic. In this work an ontology was used to represent in a different way what was done in the first part through the relational model.

2. CLASSES, OBJECT AND DATA PROPERTIES

Classes are the main aspect of ontologies and describe the concepts of a domain: for example, a class 'drug' can represent all drugs; the specific drugs represent instances of a class while prescription drugs can be seen as subclass. As this field of application is very vast and very complex, in our application we have narrowed the use case.



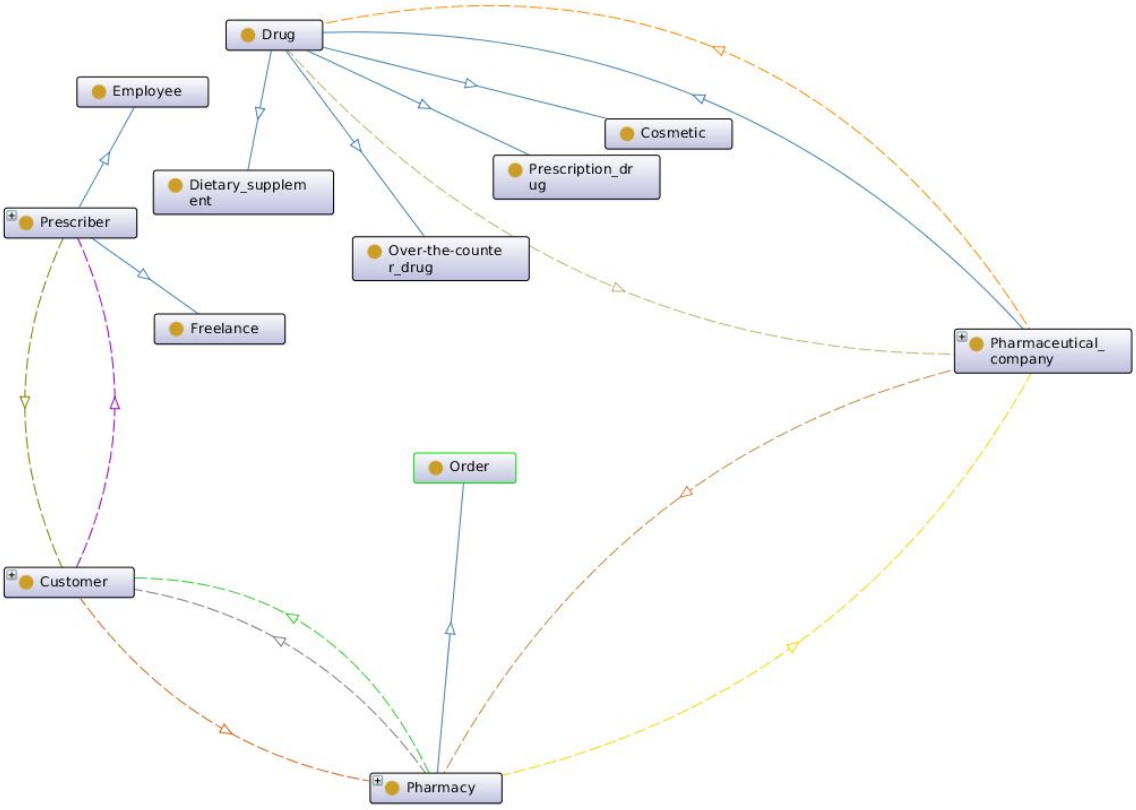
Connections between classes and individuals are represented by "object properties", characterized by a domain and a range and other some intrinsic properties. For example we can say that a drug is produced by a specific pharmaceutical company, or a pharmacy provides the service *`sells product`* to a customer that previously received a prescription.

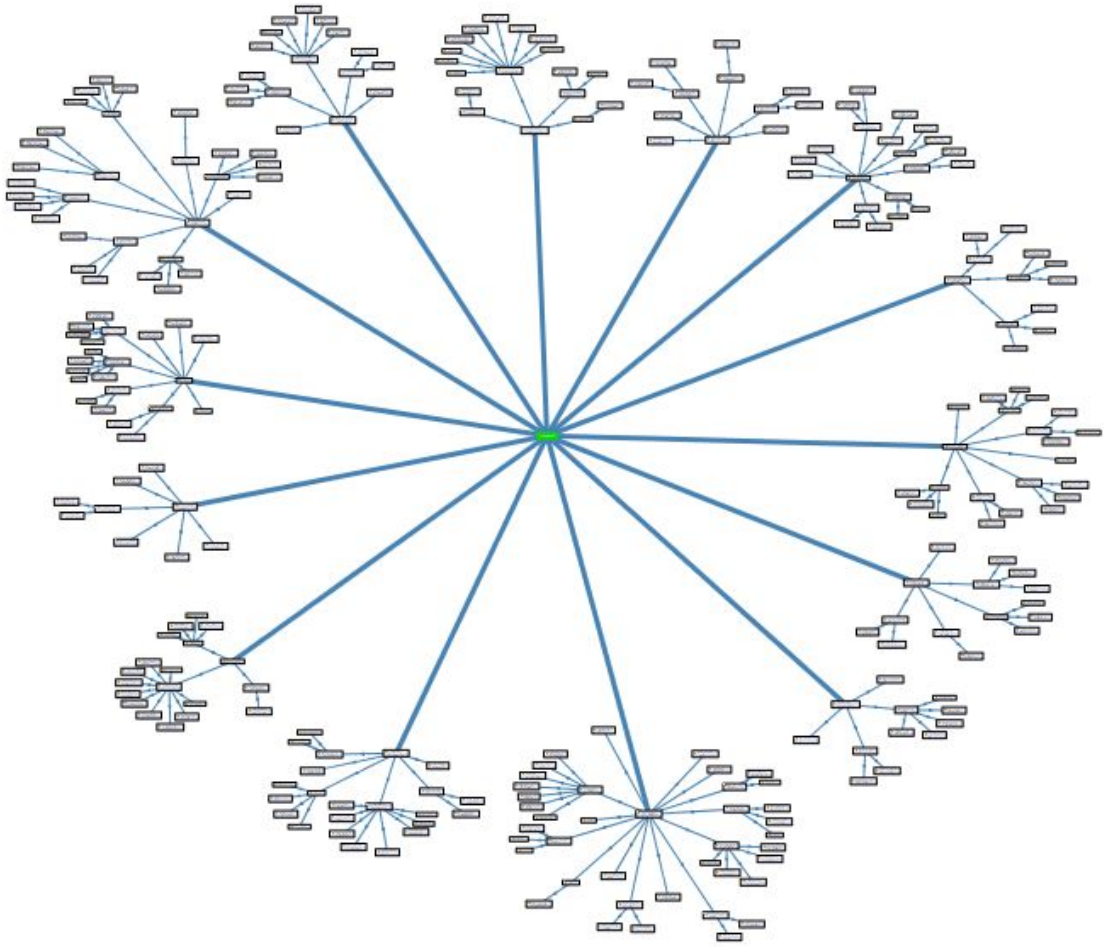


"Data properties" describe the relationships between an individual and a class or subclass and are related to instances. Data properties as well as classes and object properties are also structured hierarchically with their own domain and codomain.

In this work it was also decided to make an ontology only to represent the active principles with the relative graphic representation to show their vastness.

3. GRAPHS


Drug
Employee
Cosmetic
Prescription_dr ug
Dietary_supplem ent
Prescriber
Over-the-counte r_drug
Freelance
Pharmaceutical_ company
Order
Customer
Pharmacy



ATC ontology graph