DB Lab
PostgreSQL & SQL3

Alberto Belussi

2019/2020

POSTRESQL & SQL3

PostgreSQL implements a subset of the standard SQL3, in particular it allows one to:

- Define structured used-defined types with
  - CREATE TYPE statements
- Define typed tables with
  - CREATE TABLE <table name> OF <type name> statement
- Define multivalued and strucutred attributes with
  - ARRAY construct and using structured used-defined types as attribute types
- The UNNEST function can be applied in the FROM clause of an SQL query.

PostreSQL & SQL3

- Inheritance of type is not allowed, but inheritance of table is allowed by means of the clause:
  - INHERITS (parent_table, …)

- The ONLY clause can be used in the FROM clause of a query to limit the query only to the TABLE excluding its descendants.

POSTRESQL & SQL3

PostgreSQL does not implement the following constructs of SQL3:

- Reference type: the REF clause of CREATE TABLE does not exist in PostgreSQL.
- Thus also in the query expression the symbol -> cannot be used to ”jump” from one tuple to another one.

Exercise

FIRST STEP

- I have activated a user and a database with PostGIS extension for each one of you on the server dbserver.scienze.univr.it
- In order to activate your user and database you need to connect the server: dbserver.scienze.univr.it using GIA login.
- Then you have to read the instruction to execute the scripts that physically generate the database.
- Using pgAdmin 4, that should be available among the tools you find on your account in the university lab, you can now access to this database.

Exercise

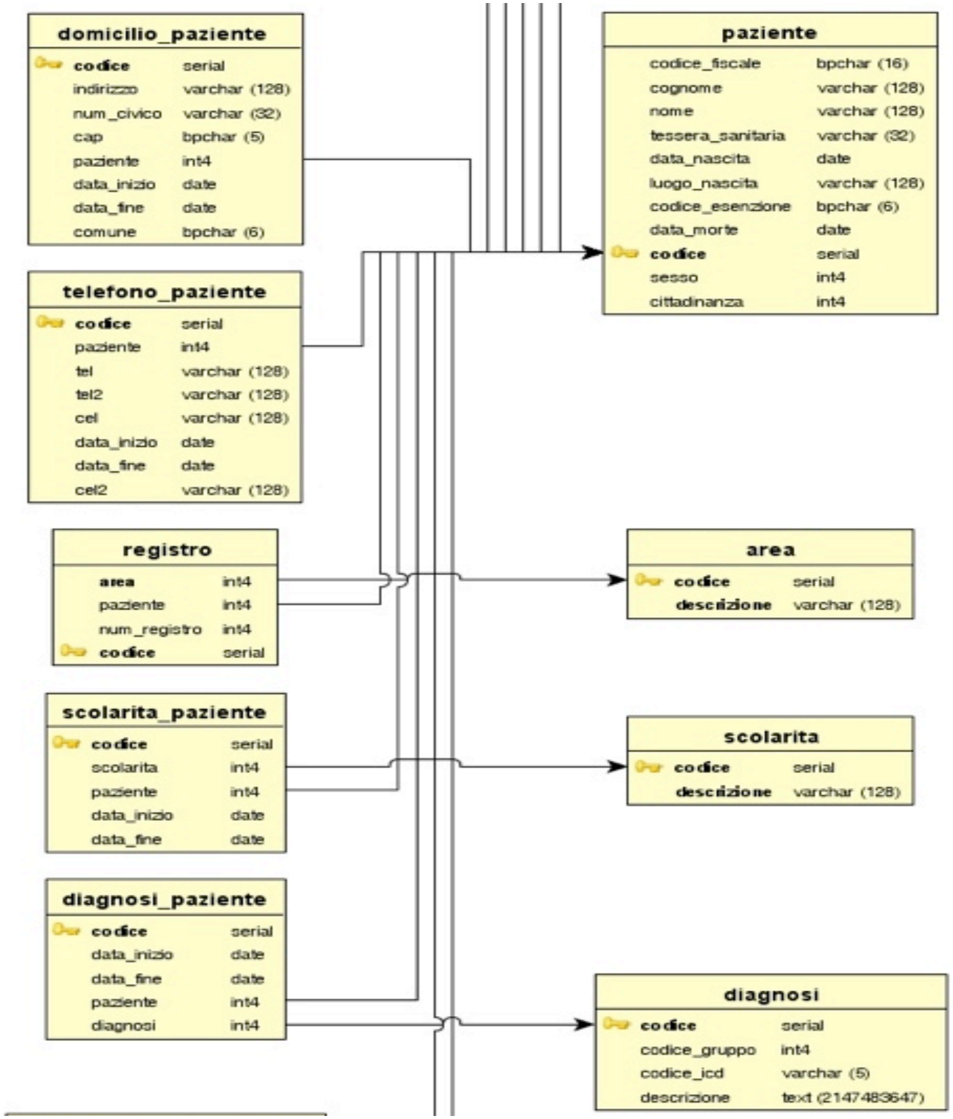
SECOND STEP

- Download from the e-learning platform the file psychoDB.backup.zip
- Unzip the file.
- From pgAdmin 4, perform a restore on your database (choose the correct option in the pop up menu on the database name).
- Refresh the database content.
- A new schema called psycho_db should appear in your database.

Exercise

THIRD STEP

- Understand the database content



domicilio_paziente	
🔑 **codice**	serial
indirizzo	varchar (128)
num_civico	varchar (32)
cap	bpchar (5)
paziente	int4
data_inizio	date
data_fine	date
comune	bpchar (6)

telefono_paziente	
🔑 **codice**	serial
paziente	int4
tel	varchar (128)
tel2	varchar (128)
cel	varchar (128)
data_inizio	date
data_fine	date
cel2	varchar (128)

registro	
area	int4
paziente	int4
num_registro	int4
🔑 **codice**	serial

scolarita_paziente	
🔑 **codice**	serial
scolarita	int4
paziente	int4
data_inizio	date
data_fine	date

diagnosi_paziente	
🔑 **codice**	serial
data_inizio	date
data_fine	date
paziente	int4
diagnosi	int4

paziente	
codice_fiscale	bpchar (16)
cognome	varchar (128)
nome	varchar (128)
tessera_sanitaria	varchar (32)
data_nascita	date
luogo_nascita	varchar (128)
codice_esenzione	bpchar (6)
data_morte	date
🔑 **codice**	serial
sesso	int4
cittadinanza	int4

area	
🔑 **codice**	serial
descrizione	varchar (128)

scolarita	
🔑 **codice**	serial
descrizione	varchar (128)

diagnosi	
🔑 **codice**	serial
codice_gruppo	int4
codice_icd	varchar (5)
descrizione	text (2147483647)

EXERCISE

FOURTH STEP

- Apply some encapsulations in order to reduce the number of tables and generate tuples with complex structure
- In particular, it is required to encapsulate in the patient the information describing her/his telephone numbers (table telefono_paziente), the place where she/he lives (table domicilio_paziente), the type of job (table professione_paziente) and the diagnosis (table diagnosi_paziente).

EXERCISE

FIFTH STEP

- In order to produce this result, it is necessary that we:
    - Define a new representation of the patient by means of a new structured user-defined type
    - Define the necessary types for the information that we need to encapsulate: telephone numbers, place where they live, type of job and diagnosis
    - Define a new typed table patient_tt
    - Insert the data from the original tables into the new table patient_tt

EXERCISE

SIXTH STEP

- Write the following queries on the new table patient_tt:
  - Find the code, the date and place of birth of the patients that lived or are living in one municipality that does not belong to the province of Verona.
  - Find the code and the telephone numbers of the patients that had a diagnosis of the disease "Episodio depressivo"
  - Group the patient for type of job and produce for each type: the name, the number of patients, the average number of diseases per patient and the average number of telephone per patient.

Connection to PostgreSQL

- How can you connect to your database using psql?
  - **export PGUSER=loginGIA**
  - **psql** –h <server> –d <database>

    psql –h dbserver –d XXXX