UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Database Management Systems, A.Y. 2019/2020
Master Degree in Computer Engineering
Master Degree in Telecommunication Engineering

Homework 3 – Physical Design

Deadline: April, 2020

Group SPRITZ	Project ExaMode Decision Support System	
Last Name Marchesin	First Name Stefano	Last Name Marchesin

Variations to the Relational Schema

Figure 1 shows the relational schema. The attributes 'start date' and 'end date' in relation 'TherapySession' have been merged into attribute 'Duration', with data type 'datarange'. Additionally, the data type of 'Visit Date' and 'Test Date' has been changed to 'tstzrange' – in this way we can model dates for medical visits and medical tests as scheduled time slots.

[Notice: report here variations to the relational schema of the previous homework, if present, otherwise include only the relational schema.]

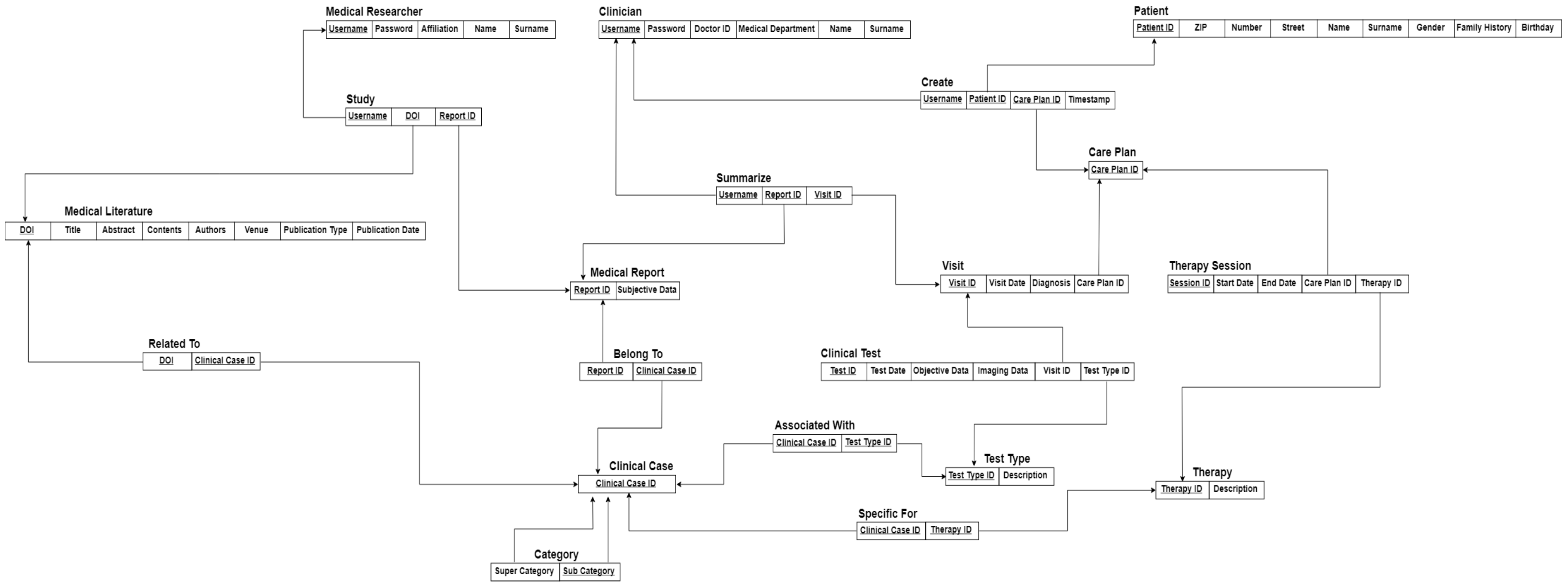

Medical Researcher
Username | Password | Affiliation | Name | Surname
Clinician
Username | Password | Doctor ID | Medical Department | Name | Surname
Patient
Patient ID | ZIP | Number | Street | Name | Surname | Gender | Family History | Birthday
Study
Username | DOI | Report ID
Create
Username | Patient ID | Care Plan ID | Timestamp
Care Plan
Care Plan ID
Summarize
Username | Report ID | Visit ID
Medical Literature
DOI | Title | Abstract | Contents | Authors | Venue | Publication Type | Publication Date
Medical Report
Report ID | Subjective Data
Visit
Visit ID | Visit Date | Diagnosis | Care Plan ID
Therapy Session
Session ID | Start Date | End Date | Care Plan ID | Therapy ID
Related To
DOI | Clinical Case ID
Belong To
Report ID | Clinical Case ID
Clinical Test
Test ID | Test Date | Objective Data | Imaging Data | Visit ID | Test Type ID
Associated With
Clinical Case ID | Test Type ID
Test Type
Test Type ID | Description
Therapy
Therapy ID | Description
Clinical Case
Clinical Case ID
Specific For
Clinical Case ID | Therapy ID
Category
Super Category | Sub Category

Figure 1Relational Schema

Physical Schema

In the following the SQL instructions to build the database in Figure 1 are reported. Note that the tables should be created in the correct order, as reported thereafter.

```
-- Database Creation
CREATE DATABASE examode OWNER POSTGRES ENCODING = 'UTF8';

-- Connect to examode db to create data for its 'public' schema
\c examode

-- Generating uuid values requires a plug-in. In Postgres, a plug-in is
known as an extension. To install an extension, call CREATE EXTENSION. To
avoid re-installing, add IF NOT EXISTS.
-- The extension for uuids is an open-source library built in C: uuid-
ossp.
-- When creating uuids, use function 'uuid_generate_v4' that generates an
uuid deriving it entirely from random numbers. See:
https://www.postgresql.org/docs/devel/uuid-ossp.html for details
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

-- Create new domains
CREATE DOMAIN pwd AS character varying(254)
      CONSTRAINT properpassword CHECK (((VALUE)::text ~* '[A-Za-z0-9._%-
]{5,}'::text));

-- Create new data types
CREATE TYPE gendertype AS ENUM (
      'Male',
      'Female'
);

CREATE TYPE pubtype AS ENUM (
      'Journal',
      'Conference'
);

-- Table Creation
-- CAVEAT: remember to create table in the right order wrt foreign keys

-- ZIP has VARCHAR(5) because postal code length is always 5 (in Italy)
-- Patient
CREATE TABLE Patient(
      PatientId UUID,
      Name VARCHAR NOT NULL,
      Surname VARCHAR NOT NULL,
      Gender GENDERTYPE,
      Birthday DATE NOT NULL,
      FamilyHistory TEXT,
      ZIP VARCHAR(5) NOT NULL,
      Street VARCHAR,
      Number SMALLINT,
      CONSTRAINT Address CHECK ((Street is NULL AND Number is NULL) OR
(Street is NOT  NULL AND Number is NOT NULL)),
      PRIMARY KEY (PatientId)
);

-- Clinician
CREATE TABLE Clinician(
      Username VARCHAR,
      Password PWD,
      DocId UUID NOT NULL,
      MedDep UUID NOT NULL,
      Name VARCHAR NOT NULL,
      Surname VARCHAR NOT NULL,
      PRIMARY KEY (Username)
);

-- MedicalResearcher
CREATE TABLE MedicalResearcher(
      Username VARCHAR,
      Password PWD,
      Affiliation VARCHAR NOT NULL,
      Name VARCHAR NOT NULL,
      Surname VARCHAR NOT NULL,
      PRIMARY KEY (Username)
);

-- CarePlan
CREATE TABLE CarePlan(
      CarePlanId UUID,
      PRIMARY KEY (CarePlanId)
);

-- TIMESTAMPTZ is the abbreviation of TIMESTAMP WITH TIME-ZONE
-- Generate
CREATE TABLE Generate(
      Username VARCHAR,
      PatientId UUID,
      CarePlanId UUID,
      TimeStamp TIMESTAMPTZ NOT NULL,
      PRIMARY KEY (Username, PatientId, CarePlanId),
      FOREIGN KEY (Username) REFERENCES Clinician(Username),
      FOREIGN KEY (PatientId) REFERENCES Patient(PatientId),
      FOREIGN KEY (CarePlanId) REFERENCES CarePlan(CarePlanId)
);

-- TIMESTAMP is used for Date as multiple visits can be done in the same
day (at different times)
-- Visit
CREATE TABLE Visit(
      VisitId UUID,
      CarePlanId UUID NOT NULL,
      Diagnosis TEXT NOT NULL,
      Date TSTZRANGE NOT NULL,
      PRIMARY KEY (VisitId),
      FOREIGN KEY (CarePlanId) REFERENCES CarePlan(CarePlanId)
);
```

```
-- Therapy
CREATE TABLE Therapy(
     TherapyId VARCHAR,
     Description TEXT NOT NULL,
     PRIMARY KEY (TherapyId)
);

-- TherapySession
CREATE TABLE TherapySession(
     SessionId UUID,
     TherapyId VARCHAR NOT NULL,
     CarePlanId UUID NOT NULL,
     Duration DATERANGE NOT NULL,
     PRIMARY KEY (SessionId),
     FOREIGN KEY (TherapyId) REFERENCES Therapy(TherapyId),
     FOREIGN KEY (CarePlanId) REFERENCES CarePlan(CarePlanId)
);

-- TestType
CREATE TABLE TestType(
     TestTypeId VARCHAR,
     Description TEXT NOT NULL,
     PRIMARY KEY (TestTypeId)
);

-- ClinicalTest
CREATE TABLE ClinicalTest(
     ClinicalTestId UUID,
     Date TSTZRANGE NOT NULL,
     ObjData TEXT NOT NULL,
     ImgData TEXT,
     VisitId UUID NOT NULL,
     TestTypeId VARCHAR NOT NULL,
     PRIMARY KEY (ClinicalTestId),
     FOREIGN KEY (VisitId) REFERENCES Visit(VisitId),
     FOREIGN KEY (TestTypeId) REFERENCES TestType(TestTypeId)
);

-- ClinicalCase
CREATE TABLE ClinicalCase(
     ClinicalCaseId VARCHAR,
     PRIMARY KEY (ClinicalCaseId)
);

-- SpecificFor
CREATE TABLE SpecificFor(
     ClinicalCaseId VARCHAR,
     TherapyId VARCHAR,
     PRIMARY KEY(ClinicalCaseId, TherapyId),
     FOREIGN               KEY(ClinicalCaseId)                REFERENCES
ClinicalCase(ClinicalCaseId),
     FOREIGN KEY(TherapyId) REFERENCES Therapy(TherapyId)
);

-- AssociatedWith
```

```
CREATE TABLE AssociatedWith(
      ClinicalCaseId VARCHAR,
      TestTypeId VARCHAR,
      PRIMARY KEY(ClinicalCaseId, TestTypeId),
      FOREIGN                KEY(ClinicalCaseId)                REFERENCES
ClinicalCase(ClinicalCaseId),
      FOREIGN KEY(TestTypeId) REFERENCES TestType(TestTypeId)
);

-- Category
CREATE TABLE Category(
      SubCategory VARCHAR,
      SuperCategory VARCHAR NOT NULL,
      PRIMARY KEY (SubCategory),
      FOREIGN KEY (SubCategory) REFERENCES ClinicalCase(ClinicalCaseId),
      FOREIGN KEY (SuperCategory) REFERENCES ClinicalCase(ClinicalCaseId)
);

-- MedicalReport
CREATE TABLE MedicalReport(
      ReportId UUID,
      SubjData TEXT NOT NULL,
      PRIMARY KEY (ReportId)
);

-- BelongTo
CREATE TABLE BelongTo(
      ReportId UUID,
      ClinicalCaseId VARCHAR,
      PRIMARY KEY (ReportId, ClinicalCaseId),
      FOREIGN KEY (ReportId) REFERENCES MedicalReport(ReportId),
      FOREIGN           KEY          (ClinicalCaseId)          REFERENCES
ClinicalCase(ClinicalCaseId)
);

-- Summarize
CREATE TABLE Summarize(
      Username VARCHAR,
      ReportId UUID,
      VisitId UUID,
      PRIMARY KEY (Username, ReportId, VisitId),
      FOREIGN KEY (Username) REFERENCES Clinician(Username),
      FOREIGN KEY (ReportId) REFERENCES MedicalReport(ReportId),
      FOREIGN KEY (VisitId) REFERENCES Visit(VisitId)
);

-- MedicalLiterature
CREATE TABLE MedicalLiterature(
      DOI VARCHAR,
      Title VARCHAR NOT NULL,
      Abstract VARCHAR,
      Contents VARCHAR,
      Authors VARCHAR NOT NULL,
      Venue VARCHAR NOT NULL,
      PublicationType PUBTYPE NOT NULL,
      PublicationDate DATE NOT NULL,
	PRIMARY KEY (DOI)
);

-- RelatedTo
CREATE TABLE RelatedTo(
	DOI VARCHAR,
	ClinicalCaseId VARCHAR,
	PRIMARY KEY (DOI, ClinicalCaseId),
	FOREIGN KEY (DOI) REFERENCES MedicalLiterature(DOI),
	FOREIGN KEY (ClinicalCaseId) REFERENCES
ClinicalCase(ClinicalCaseId)
);

-- Study
CREATE TABLE Study(
	Username VARCHAR,
	DOI VARCHAR,
	ReportId UUID,
	PRIMARY KEY (Username, DOI, ReportId),
	FOREIGN KEY (Username) REFERENCES MedicalResearcher(Username),
	FOREIGN KEY (DOI) REFERENCES MedicalLiterature(DOI),
	FOREIGN KEY (ReportId) REFERENCES MedicalReport(ReportId)
);
```

Populate the Database: Example

In the following, there are some examples of SQL instructions to insert data within relation MedicalReport. Whenever a new visit for a given care plan is performed, a clinician updates the MedicalReport relation and all the relations associated with it.

```
-- MedicalReport

-- The process of inserting medical reports within the db is as follows:
-- 1: The clinician writes the subjective data related to the outcomes of
a visit
-- 2: The MedicalReport is associated with one (or more) ClinicalCase ids
-- 3: The relation Summarize is filled with the Clinician Username, the
MedicalReport Id and the Visit Id
-- 4: The SubjData attribute of MedicalReport, for the given ReportId, is
modified if an additional visit is performed
-- 5: The relation Summarize is filled with the (potentially new)
Clinician Username, the (same) ReportId and the (new) Visit Id
-- 6: Repeat steps 1-5

-- INSERT OPERATIONS for MedicalReport: 3e636e45-034a-4269-b868-
a0a8958086d4
INSERT INTO MedicalReport (ReportId, SubjData) VALUES
	('3e636e45-034a-4269-b868-a0a8958086d4', 'Written on 2010-05-03
17:22
                                                         \n Found
hearth problems. A 6 months therapy is required to reduce the risk of
heart attacks.');

INSERT INTO BelongTo (ReportId, ClinicalCaseId) VALUES
```

```
      ('3e636e45-034a-4269-b868-a0a8958086d4', 'T190');

INSERT INTO Summarize (Username, ReportId, VisitId) VALUES
      ('DocDre',  '3e636e45-034a-4269-b868-a0a8958086d4',  '712f38a9-16c9-
4ff1-ad8b-45ce46ad4e53');

--   UPDATE   OPERATIONS   for   MedicalReport:   3e636e45-034a-4269-b868-
a0a8958086d4
UPDATE MedicalReport
      SET SubjData = 'Written on 2010-05-03 17:22
                          \n Found hearth problems. A 6 months therapy
is required to reduce the risk of heart attacks.
                          \n Written on 2010-11-03 13:22
                          \n Hearth problems resolved but found a strong
sepsis blood infection. An additional 3 months therapy is required to
eradicate the infection.'
      WHERE
            ReportId = '3e636e45-034a-4269-b868-a0a8958086d4';

INSERT INTO BelongTo (ReportId, ClinicalCaseId) VALUES
      ('3e636e45-034a-4269-b868-a0a8958086d4', 'T045');

INSERT INTO Summarize (Username, ReportId, VisitId) VALUES
      ('DocDre',  '3e636e45-034a-4269-b868-a0a8958086d4',  '3775bb0b-130b-
4a70-9c88-470b457c1340');

UPDATE MedicalReport
      SET SubjData = 'Written on 2010-05-03 17:22
                          \n Found hearth problems. A 6 months therapy
is required to reduce the risk of heart attacks.
                          \n\n Written on 2010-11-03 13:22
                          \n Hearth problems resolved but found a strong
sepsis blood infection. An additional 3 months therapy is required to
eradicate the infection.
                          \n\n Written on 2011-02-03 13:22
                          \n  Blood  infection  eradicated.  Patient  can
leave the hospital.'
      WHERE
            ReportId = '3e636e45-034a-4269-b868-a0a8958086d4';

INSERT INTO Summarize (Username, ReportId, VisitId) VALUES
      ('DocDre',  '3e636e45-034a-4269-b868-a0a8958086d4',  '102bcee0-501b-
4c63-a5c0-31a82a14a8a5');


--   INSERT   OPERATIONS   for   MedicalReport:   fe2ef85c-752b-4af5-ae67-
dfc1f247f979
INSERT INTO MedicalReport (ReportId, SubjData) VALUES
      ('fe2ef85c-752b-4af5-ae67-dfc1f247f979',  'Written  on  2012-06-05
11:00
                                                              \n   Patient
presents respiratory problems related to a backlash of mild sepsis. A 2-
weeks therapy is required.');

INSERT INTO BelongTo (ReportId, ClinicalCaseId) VALUES
      ('fe2ef85c-752b-4af5-ae67-dfc1f247f979', 'T045');
```

```
INSERT INTO Summarize (Username, ReportId, VisitId) VALUES
      ('DocDre',  'fe2ef85c-752b-4af5-ae67-dfc1f247f979',  'f45d74ee-2e30-
4f09-a4ae-37df7102f65a');

--   UPDATE   OPERATIONS   for   MedicalReport:   fe2ef85c-752b-4af5-ae67-
dfc1f247f979
UPDATE MedicalReport
      SET SubjData = 'Written on 2012-06-05 11:00
                              \n   Patient   presents   respiratory   problems
related to a backlash of mild sepsis. A 2-weeks therapy is required.
                              \n\n Written on 2012-06-21 09:00
                              \n Patient problems are debellated.'
      WHERE
            ReportId = 'fe2ef85c-752b-4af5-ae67-dfc1f247f979';

INSERT INTO Summarize (Username, ReportId, VisitId) VALUES
      ('DocDre',  'fe2ef85c-752b-4af5-ae67-dfc1f247f979',  'f8321357-c86c-
403c-8887-aeeafce50db4');


--   INSERT   OPERATION   for   MedicalReport:   ccdfba04-6725-49cb-8814-
c214c35fe3ea
INSERT INTO MedicalReport (ReportId, SubjData) VALUES
      ('ccdfba04-6725-49cb-8814-c214c35fe3ea',   'Written   on   2017-03-05
17:00
                                                                      \n   Patient
complains back pain. Patient has to take pain killers for one week. An
additional visit will be scheduled only if patient still suffers from
back pain.');

INSERT INTO BelongTo (ReportId, ClinicalCaseId) VALUES
      ('ccdfba04-6725-49cb-8814-c214c35fe3ea', 'T391');

INSERT INTO Summarize (Username, ReportId, VisitId) VALUES
      ('AvjJim',  'ccdfba04-6725-49cb-8814-c214c35fe3ea',  '02cc96bb-5276-
42d7-82e4-ff262aa60efd');
```

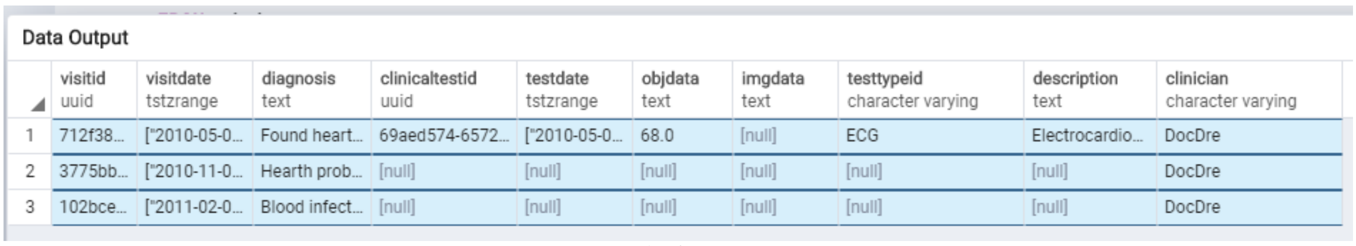
Principal Queries

In this section, we report three queries to navigate the database:

1. Retrieve all the visit-related information for a given care plan;
2. Retrieve all the research-related information for clinical cases;
3. Retrieve total number of visits per care plan.

1.

```
-- Retrieve all the visit-related information for a given care plan

-- Subquery to retrieve all the visit-related information
SELECT Visit.VisitId,
         Visit.Date AS VisitDate,
         Visit.Diagnosis,
         ClinicalTest.ClinicalTestId,
         ClinicalTest.Date AS TestDate,
         ClinicalTest.ObjData,
         ClinicalTest.ImgData,
         TestType.TestTypeId,
         TestType.Description
FROM Visit
      LEFT OUTER JOIN ClinicalTest ON Visit.VisitId =
ClinicalTest.VisitId
      LEFT OUTER JOIN TestType ON ClinicalTest.TestTypeId =
TestType.TestTypeId
WHERE Visit.CarePlanId = 'de7c222e-98f0-4eae-b690-7fb37a246bdd';

-- Subquery to retrieve clinician-related information
SELECT Summarize.Username AS Clinician,
         Summarize.VisitId
FROM Summarize
      INNER JOIN MedicalReport ON Summarize.ReportId =
MedicalReport.ReportId;

-- Query to retrieve complete information about visits for a given care
plan
SELECT VisitInfo.VisitId,
         VisitInfo.VisitDate,
         VisitInfo.Diagnosis,
         VisitInfo.ClinicalTestId,
         VisitInfo.TestDate,
         VisitInfo.ObjData,
         VisitInfo.ImgData,
         VisitInfo.TestTypeId,
         VisitInfo.Description,
         ClinicianInfo.Clinician
FROM
      (SELECT Visit.VisitId,
                    Visit.Date AS VisitDate,
                    Visit.Diagnosis,
                    ClinicalTest.ClinicalTestId,
                    ClinicalTest.Date AS TestDate,
                    ClinicalTest.ObjData,
                    ClinicalTest.ImgData,
                    TestType.TestTypeId,
                    TestType.Description
            FROM Visit
                  LEFT OUTER JOIN ClinicalTest ON Visit.VisitId =
ClinicalTest.VisitId
                  LEFT OUTER JOIN TestType ON ClinicalTest.TestTypeId =
TestType.TestTypeId
            WHERE Visit.CarePlanId = 'de7c222e-98f0-4eae-b690-
7fb37a246bdd')
      AS VisitInfo
      INNER JOIN
      (SELECT Summarize.Username AS Clinician,
                  Summarize.VisitId
            FROM Summarize
                  INNER JOIN MedicalReport ON Summarize.ReportId =
MedicalReport.ReportId)
      AS ClinicianInfo ON VisitInfo.VisitId = ClinicianInfo.VisitId;
```



Data Output

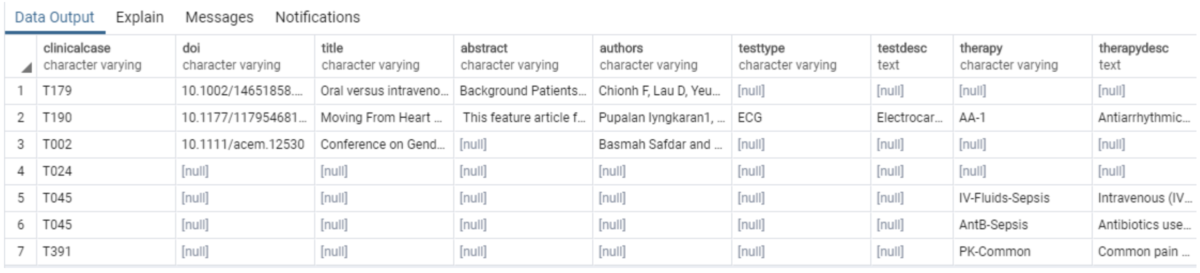
	visitid uuid	visitdate tstzrange	diagnosis text	clinicaltestid uuid	testdate tstzrange	objdata text	imgdata text	testtypeid character varying	description text	clinician character varying
1	712f38...	["2010-05-0...	Found heart...	69aed574-6572...	["2010-05-0...	68.0	[null]	ECG	Electrocardio...	DocDre
2	3775bb...	["2010-11-0...	Hearth prob...	[null]	[null]	[null]	[null]	[null]	[null]	DocDre
3	102bce...	["2011-02-0...	Blood infect...	[null]	[null]	[null]	[null]	[null]	[null]	DocDre

Figure 2 Results for Query 1

2.

```
-- Retrieve all the research information related to clinical cases

-- Subquery to retrieve medical literature info
SELECT RT.ClinicalCaseId,
        ML.DOI,
        ML.Title,
        ML.Abstract,
        ML.Authors
FROM MedicalLiterature AS ML
      NATURAL JOIN RelatedTo AS RT;

-- Subquery to retrieve medical test info
SELECT AW.ClinicalCaseId,
        AW.TestTypeId,
        TT.Description
FROM AssociatedWith AS AW
      NATURAL JOIN TestType AS TT;

-- Subquery to retrieve therapy info
SELECT SF.ClinicalCaseId,
        SF.TherapyId,
        TH.Description
FROM SpecificFor AS SF
      NATURAL JOIN Therapy AS TH;

-- Complete query
SELECT CC.ClinicalCaseId AS ClinicalCase,
       MedLit.DOI,
        MedLit.Title,
        MedLit.Abstract,
        MedLit.Authors,
        TestInfo.TestTypeId AS TestType,
        TestInfo.Description AS TestDesc,
        TherapyInfo.TherapyId AS Therapy,
        TherapyInfo.Description AS TherapyDesc
FROM
      ClinicalCase AS CC
      LEFT OUTER JOIN
      (SELECT RT.ClinicalCaseId,
                ML.DOI,
                ML.Title,
                ML.Abstract,
                ML.Authors
           FROM MedicalLiterature AS ML
                NATURAL JOIN RelatedTo AS RT) AS MedLit
```

```
    ON CC.ClinicalCaseId = MedLit.ClinicalCaseId
    LEFT OUTER JOIN
    (SELECT AW.ClinicalCaseId,
                AW.TestTypeId,
                TT.Description
          FROM AssociatedWith AS AW
                NATURAL JOIN TestType AS TT) AS TestInfo
    ON CC.ClinicalCaseId = TestInfo.ClinicalCaseId
    LEFT OUTER JOIN
    (SELECT SF.ClinicalCaseId,
                SF.TherapyId,
                TH.Description
          FROM SpecificFor AS SF
                NATURAL JOIN Therapy AS TH) AS TherapyInfo
    ON CC.ClinicalCaseId = TherapyInfo.ClinicalCaseId;
```

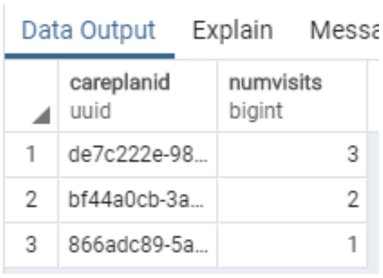

Data Output Explain Messages Notifications

	clinicalcase character varying	doi character varying	title character varying	abstract character varying	authors character varying	testtype character varying	testdesc text	therapy character varying	therapydesc text
1	T179	10.1002/14651858.…	Oral versus intraveno…	Background Patients…	Chionh F, Lau D, Yeu…	[null]	[null]	[null]	[null]
2	T190	10.1177/117954681…	Moving From Heart …	This feature article f…	Pupalan Iyngkaran1, …	ECG	Electrocar…	AA-1	Antiarrhythmic…
3	T002	10.1111/acem.12530	Conference on Gend…	[null]	Basmah Safdar and …	[null]	[null]	[null]	[null]
4	T024	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]
5	T045	[null]	[null]	[null]	[null]	[null]	[null]	IV-Fluids-Sepsis	Intravenous (IV…
6	T045	[null]	[null]	[null]	[null]	[null]	[null]	AntB-Sepsis	Antibiotics use…
7	T391	[null]	[null]	[null]	[null]	[null]	[null]	PK-Common	Common pain …

Figure 3 Results of Query 2

3.

```
-- Retrieve total number of visits per care plan

SELECT CarePlanId, COUNT(VisitId) AS NumVisits

FROM Visit

GROUP BY CarePlanId

ORDER BY NumVisits DESC;
```


Data Output Explain Messa

	careplanid uuid	numvisits bigint
1	de7c222e-98…	3
2	bf44a0cb-3a…	2
3	866adc89-5a…	1

Figure 4 Results of Query 3

JDBC Implementations of the Principal Queries and Visualization

Hereafter, we report a java class which read the data from the database and print the results on screen.

```
/*
 * Copyright 2019 University of Padua, Italy
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

/**
 * Lists all the visits in the database
 *
 * @author Stefano Marchesin
 * @version 1.00
 */
public class PrintVisit {

      /**
       * The JDBC driver to be used
       */
      private static final String DRIVER = "org.postgresql.Driver";

      /**
       * The URL of the database to be accessed
       */
      private static final String DATABASE =
"jdbc:postgresql://localhost/examode";

      /**
       * The username for accessing the database
       */
      private static final String USER = "postgres";

      /**
       * The password for accessing the database
       */
      private static final String PASSWORD = "pwd";

      /**
```

```
	 * The SQL statement to be executed
	 */
	private static final String SQL = "SELECT VisitId, Diagnosis, Date
FROM Visit;";


	/**
	 * List all the authors in the database
	 *
	 * @param args
	 *            command-line arguments (not used).
	 */
	public static void main(String[] args) {

		// the connection to the DBMS
		Connection con = null;

		// the statement to be executed
		Statement stmt = null;

		// the results of the statement execution
		ResultSet rs = null;

		// start time of a statement
		long start;

		// end time of a statement
		long end;


		// "data structures" for the data to be read from the database

		// the data about the visit
		String visitId = null;
		String diagnosis = null;
		String date = null;

		try {
			// register the JDBC driver
			Class.forName(DRIVER);

			System.out.printf("Driver         %s         successfully
registered.%n", DRIVER);
		} catch (ClassNotFoundException e) {
			System.out.printf(
					"Driver   %s   not   found:   %s.%n",   DRIVER,
e.getMessage());

			// terminate with a generic error code
			System.exit(-1);
		}

		try {

			// connect to the database
			start = System.currentTimeMillis();
```

```
                con    =    DriverManager.getConnection(DATABASE,    USER,
PASSWORD);

                end = System.currentTimeMillis();

                System.out.printf(
                           "Connection   to   database   %s   successfully
established in %,d milliseconds.%n",
                           DATABASE, end-start);

                // create the statement to execute the query
                start = System.currentTimeMillis();

                stmt = con.createStatement();

                end = System.currentTimeMillis();

                System.out.printf(
                           "Statement   successfully   created   in   %,d
milliseconds.%n",
                           end-start);

                // execute the query
                start = System.currentTimeMillis();

                rs = stmt.executeQuery(SQL);

                end = System.currentTimeMillis();

                System.out
                           .printf("Query  %s  successfully  executed  %,d
milliseconds.%n",
                                      SQL, end - start);

                System.out
                           .printf("Query results:%n");

                // cycle on the query results and print them
                while (rs.next()) {

                     // read visit identifier
                     visitId = rs.getString("visitId");

                     // read the diagnosis
                     diagnosis = rs.getString("diagnosis");

                     // read the date scheduled for the visit
                     date = rs.getString("date");

                     System.out.printf("- %s, %s, %s%n",
                               visitId, diagnosis, date);

                }
           } catch (SQLException e) {
                System.out.printf("Database access error:%n");
                // cycle in the exception chain
                while (e != null) {
                     System.out.printf("-        Message:         %s%n",
e.getMessage());
                     System.out.printf("-   SQL   status   code:   %s%n",
e.getSQLState());
                     System.out.printf("-   SQL   error   code:   %s%n",
e.getErrorCode());
                     System.out.printf("%n");
                     e = e.getNextException();
                }
           } finally {
                try {

                     // close the used resources
                     if (rs != null) {

                          start = System.currentTimeMillis();

                          rs.close();

                          end = System.currentTimeMillis();

                          System.out
                          .printf("Result set successfully closed in %,d
milliseconds.%n",
                                    end-start);
                     }

                     if (stmt != null) {

                          start = System.currentTimeMillis();

                          stmt.close();

                          end = System.currentTimeMillis();

                          System.out
                          .printf("Statement successfully closed in %,d
milliseconds.%n",
                                    end-start);
                     }

                     if (con != null) {

                          start = System.currentTimeMillis();

                          con.close();

                          end = System.currentTimeMillis();

                          System.out
                          .printf("Connection successfully closed in %,d
milliseconds.%n",
                                    end-start);
                }

                System.out.printf("Resources successfully
released.%n");

            } catch (SQLException e) {
                System.out.printf("Error while releasing
resources:%n");

                // cycle in the exception chain
                while (e != null) {
                    System.out.printf("- Message: %s%n",
e.getMessage());
                    System.out.printf("- SQL status code: %s%n",
e.getSQLState());
                    System.out.printf("- SQL error code: %s%n",
e.getErrorCode());
                    System.out.printf("%n");
                    e = e.getNextException();
                }

            } finally {

                // release resources to the garbage collector
                rs = null;
                stmt = null;
                con = null;

                System.out.printf("Resources released to the
garbage collector.%n");
            }
        }

        System.out.printf("Program end.%n");

    }
}
```