FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

OBJECT RELATIONAL ASSIGNMENT

Database Technologies 2020/2021

Group G:

André Mori - up201700493@edu.fe.up.pt
Diogo Silva - up201405742@edu.fe.up.pt
Jiahao Zhang - up202010263@edu.fe.up.pt
Pasit Khantigul - up202010272@edu.fe.up.pt

Index

Introduction **2**

Object-relational Data Model **2**

Populate **4**

Creation of object types 4

Creation of tables 5

Populating the tables 6

Methods **8**

isActivity 8

descriptionContains 8

municipalityHasActivity and totalFacilities 9

totalFacilities - region_t 10

Questions **10**

Question 1 10

Query 10

Result 11

Question 2 11

Query 11

Result 12

Question 3 12

Query 12

Result 12

Question 4 12

Query 12

Result 13

Question 5 13

Query 13

Result 14

Question 6 14

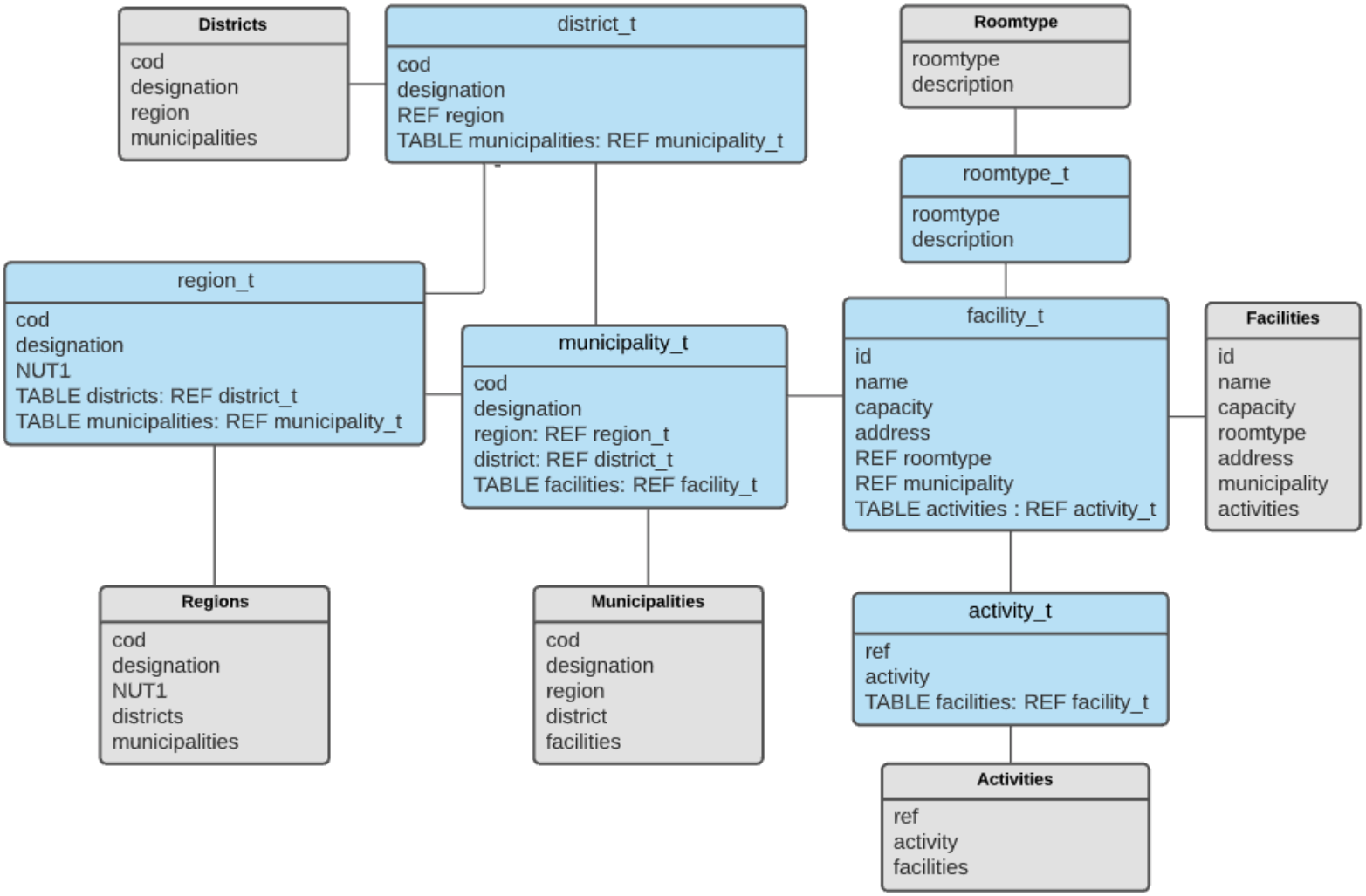
Query 14

Conclusion **15**

Introduction

The main goal of this project is to develop a database using the SQL3 Object Model and present some special characteristics of the OR database like the use of objects, references, nested tables, methods and functions.
Note: scripts containing all the necessary queries to create, populate and query the database can also be found in the **sql/** folder delivered.

Object-relational Data Model



We defined the following types to be used on the creation of the tables of the database (colored blue in the image above):

- **region_t:** represents a region of Portugal, contains a reference to its districts and its municipalities.
- **district_t:** represents a district of Portugal and contains a reference to its municipalities.

- **districtsref_tab_t:** represents a table of references to districts.
- **municipality_t:** represents a municipality of Portugal and contains reference to its facilities.
- **municipalitiesref_tab_t:** represents a table of references to municipalities.
- **roomtype_t:** represents a room type of a facility.
- **facility_t:** represents a facility in a municipality and contains reference to its room type and activities.
- **facilitiesref_tab_t:** represents a table of references to facilities.
- **activity_t:** represents an activity done in a facility and contains reference to its facilities.
- **activitiesref_tab_t:** represents a table of references to activities.

The tables created based on the types are (colored gray in the image above):

- Regions
- Districts
- Municipalities
- Roomtypes
- Activities
- Facilities

Populate

Before starting with the population of the database, we should firstly identify the object types where we populate the database:

Creation of object types

```
CREATE OR REPLACE TYPE region_t AS OBJECT (
    cod NUMBER(4, 0),
    designation   VARCHAR2(50 BYTE),
    nut1 VARCHAR2(50 BYTE)
)
/
CREATE OR REPLACE TYPE district_t AS OBJECT
(
    cod     NUMBER(4,0),
    designation   VARCHAR2(50 BYTE),
    region REF region_t,
    MEMBER FUNCTION facilitiesInAllMunicipalities RETURN NUMBER
)
/
CREATE OR REPLACE TYPE districtsref_tab_t AS TABLE OF REF district_t
/
CREATE OR REPLACE TYPE municipality_t AS OBJECT
(
    cod     NUMBER(4,0),
    designation   VARCHAR2(50 BYTE),
    district REF district_t,
    region REF region_t,
    MEMBER FUNCTION municipalityHasActivity(word STRING) RETURN NUMBER

)
/
CREATE OR REPLACE TYPE municipalitiesref_tab_t AS TABLE OF REF
municipality_t
/
CREATE OR REPLACE TYPE roomtype_t AS OBJECT
(
    roomtype NUMBER(4,0),
    description VARCHAR2(50 BYTE),
    MEMBER FUNCTION descriptionContains(word STRING) RETURN STRING
)
/
CREATE OR REPLACE TYPE facility_t AS OBJECT
(
```

```
    id NUMBER(4,0),
    name VARCHAR2(80),
    capacity NUMBER(8,0),
    roomtype REF roomtype_t,
    address VARCHAR2(80),
    municipality REF municipality_t
)
/
CREATE OR REPLACE TYPE facilitiesref_tab_t AS TABLE OF REF facility_t
/
CREATE OR REPLACE TYPE activity_t AS OBJECT
(
    ref VARCHAR2(20),
    activity VARCHAR2(20),
    MEMBER FUNCTION isActivity(word STRING) RETURN STRING
)
/
CREATE OR REPLACE TYPE activitiesref_tab_t AS TABLE OF REF activity_t
/

ALTER TYPE region_t
ADD ATTRIBUTE (districts districtsref_tab_t) CASCADE;

ALTER TYPE region_t
ADD ATTRIBUTE (municipalities municipalitiesref_tab_t) CASCADE;

ALTER TYPE district_t
ADD ATTRIBUTE (municipalities municipalitiesref_tab_t) CASCADE;

ALTER TYPE municipality_t
ADD ATTRIBUTE (facilities facilitiesref_tab_t) CASCADE;

ALTER TYPE facility_t
ADD ATTRIBUTE (activities activitiesref_tab_t) CASCADE;

ALTER TYPE activity_t
ADD ATTRIBUTE (facilities facilitiesref_tab_t) CASCADE;
```

After creating the Object type for each object, we start filling the database with tables using the object type as shown below:

Creation of tables

```
CREATE TABLE regions OF region_t
    NESTED TABLE districts STORE AS region_districts
    NESTED TABLE municipalities STORE AS region_municipalities;

CREATE TABLE districts OF district_t
    NESTED TABLE municipalities STORE AS district_municipalities;

CREATE TABLE municipalities OF municipality_t
    NESTED TABLE facilities STORE AS municipality_facilities;

CREATE TABLE roomtypes OF roomtype_t;

CREATE TABLE facilities OF facility_t
    NESTED TABLE activities STORE AS facility_activities;

CREATE TABLE activities OF activity_t
    NESTED TABLE facilities STORE AS activity_facilities;
```

And finally, we are ready to populate the tables.

Populating the tables

```
DELETE FROM regions;
DELETE FROM districts;
DELETE FROM municipalities;
DELETE FROM roomtypes;
DELETE FROM facilities;
DELETE FROM activities;

INSERT INTO regions (COD, DESIGNATION, NUT1)
SELECT COD, DESIGNATION, NUT1 FROM gtd8.regions;

INSERT INTO districts (COD, DESIGNATION, REGION)
SELECT d.COD, d.DESIGNATION, (SELECT ref(r) FROM regions r WHERE r.COD =
d.REGION) FROM gtd8.districts d;

INSERT INTO municipalities (COD, DESIGNATION, DISTRICT, REGION)
SELECT m.COD, m.DESIGNATION, (SELECT ref(d) FROM districts d WHERE d.COD
= m.DISTRICT), (SELECT ref(r) FROM regions r WHERE r.COD = m.REGION)
FROM gtd8.municipalities m;

INSERT INTO roomtypes (ROOMTYPE, DESCRIPTION)
SELECT rt.ROOMTYPE, rt.DESCRIPTION FROM gtd8.roomtypes rt;
```

```
INSERT INTO facilities (ID, NAME, CAPACITY, ROOMTYPE, ADDRESS,
MUNICIPALITY)
SELECT f.ID, f.NAME, f.CAPACITY, (SELECT ref(rt) FROM roomtypes rt WHERE
rt.ROOMTYPE = f.ROOMTYPE), f.ADDRESS, (SELECT ref(m) FROM municipalities
m WHERE m.COD = f.MUNICIPALITY) FROM gtd8.facilities f;

INSERT INTO activities (REF, ACTIVITY) SELECT a.REF, a.ACTIVITY
FROM gtd8.activities a;

------------------------------
------- Nested Tables ---------

-- Facilities Activities
UPDATE facilities f
SET f.activities =
CAST(MULTISET(SELECT REF(a) FROM activities a JOIN gtd8.Uses u ON a.ref
= u.ref WHERE u.id = f.id) AS activitiesref_tab_t);

-- Activities Facilities
UPDATE activities a
SET a.facilities =
CAST(MULTISET(SELECT REF(f) FROM facilities f JOIN gtd8.Uses u ON a.ref
= u.ref WHERE u.id = f.id) AS facilitiesref_tab_t);

-- Municipalities Facilities
UPDATE municipalities m
SET m.Facilities =
CAST(MULTISET(SELECT REF(f) FROM facilities f WHERE m.cod =
f.municipality.cod) AS facilitiesref_tab_t);

-- Districts Municipalities
UPDATE districts d
SET d.Municipalities =
CAST(MULTISET(SELECT REF(m) FROM municipalities m WHERE d.cod =
m.district.cod) AS municipalitiesref_tab_t);

-- Regions Districts
UPDATE regions r
SET r.Districts =
CAST(MULTISET(SELECT REF(d) FROM districts d WHERE r.cod = d.region.cod)
AS districtsref_tab_t);

-- Regions Municipalities
UPDATE regions r
SET r.Municipalities =
CAST(MULTISET(SELECT REF(m) FROM municipalities m WHERE r.cod =
```

```
m.region.cod) AS municipalitiesref_tab_t);
```

Methods

In this database, we have found useful some methods that simplify the lecture of the queries. Here are shown some possibilities of the methods that are used in the queries:

isActivity

```
CREATE OR REPLACE TYPE BODY activity_t AS
    MEMBER FUNCTION isActivity(word STRING) RETURN STRING IS
    BEGIN
        IF activity = word THEN
            RETURN 'TRUE';
        ELSE
            RETURN 'FALSE';
        END IF;
    END isActivity;
END;
```

This function is used to check if the activity of the object matches with the activity inserted as parameter of the method (word STRING)

descriptionContains

```
CREATE OR REPLACE TYPE BODY roomtype_t AS
    MEMBER FUNCTION descriptionContains(word STRING) RETURN STRING
IS
    BEGIN
        IF description LIKE '%' || word || '%' THEN
            RETURN 'TRUE';
        ELSE
            RETURN 'FALSE';
        END IF;
    END descriptionContains;
END;
```

This method does a very similar thing as the previous function but it checks if the attribute activity of the object roomtype contains the word sent as parameter to the method.

municipalityHasActivity and totalFacilities

```
CREATE OR REPLACE TYPE BODY municipality_t AS
    MEMBER FUNCTION municipalityHasActivity(word STRING) RETURN
INTEGER IS
    activityCount INTEGER;

    BEGIN
        SELECT COUNT(*) INTO activityCount
        FROM TABLE(facilities) f, TABLE(VALUE(f).activities) a
        WHERE VALUE(a).isActivity(word) = 'TRUE';

        RETURN activityCount;

    END municipalityHasActivity;


    MEMBER FUNCTION totalFacilities RETURN INTEGER IS
    nFacilities INTEGER;

    BEGIN
        SELECT COUNT(*) INTO nFacilities
        FROM municipalities m, TABLE(m.facilities) f
        WHERE SELF.cod = m.cod;

        RETURN nFacilities;
    END totalFacilities;
END;
```

These two functions of the municipality type are very simple:

- the 1st one checks if a municipality contains a facility that contains an activity that matches with the parameter word
- the 2nd method returns the count of municipalities (and facilities inside the specific municipality) that has the same cod as the object where it’s calling the function

totalFacilities - region_t

```
alter type region_t
add member function totalFacilities return integer cascade;

CREATE OR REPLACE TYPE BODY region_t AS
    MEMBER FUNCTION totalFacilities RETURN INTEGER IS
    totalFacilities INTEGER;

    BEGIN
        SELECT SUM(m.totalFacilities()) INTO totalFacilities
        FROM municipalities m
        WHERE m.region.cod = self.cod;

        RETURN totalFacilities;
    END totalFacilities;
END;
```

Note: the method was not tested due to a problem regarding the Oracle server at FEUP.
The method returns the count of facilities inside of a specific region that has the same cod as the object where it's calling the function.

Questions

Question 1

Which are the facilities where the room type description contains 'touros' and have 'teatro' as one of their activities? Show the id, name, description and activity.

Query

```
-- 1
SELECT f.id, f.name, f.roomtype.description, VALUE(a).activity
FROM facilities f, TABLE(f.activities) a
WHERE f.roomtype.description LIKE '%touros%' AND VALUE(a).activity =
'teatro';

---- 1 with functions
SELECT f.id, f.name, f.roomtype.description, VALUE(a).activity
FROM facilities f, TABLE(f.activities) a
WHERE f.roomtype.descriptionContains('touros') = 'TRUE'
AND a.isActivity('teatro') = 'TRUE';
```

Result

	ID	NAME	ROOMTYPE.DESCRIPTION	VALUE(A).ACTIVITY
1	916	COLISEU JOSÉ RONDÃO DE ALMEIDA-EX PRAÇA DE TOIROS	Praça de touros multiusos	teatro
2	940	ARENA DE ÉVORA - EX PRAÇA DE TOIROS	Praça de touros multiusos	teatro
3	957	COLISEU DE REDONDO - EX PRAÇA DE TOIROS	Praça de touros multiusos	teatro

Question 2

How many facilities with 'touros' in the room type description are there in each region?

Query

```
-- 2
SELECT r.cod, r.designation, COALESCE(info.n_facilities, 0) n_facilities
FROM regions r LEFT OUTER JOIN
(
    SELECT r.cod, r.designation, COUNT(VALUE(f).ID) n_facilities
    FROM regions r, TABLE(r.municipalities) m,
TABLE(VALUE(m).facilities) f
    WHERE VALUE(f).roomtype.description LIKE '%touros%'
    GROUP BY r.cod, r.designation
    ORDER BY r.cod
) info
ON r.cod = info.cod;

---- 2 with functions
SELECT r.cod, r.designation, COALESCE(info.n_facilities, 0) n_facilities
FROM regions r LEFT OUTER JOIN
(
    SELECT r.cod, r.designation, COUNT(VALUE(f).ID) n_facilities
    FROM regions r, TABLE(r.municipalities) m,
TABLE(VALUE(m).facilities) f
    WHERE VALUE(f).roomtype.descriptionContains('touros') = 'TRUE'
    GROUP BY r.cod, r.designation
    ORDER BY r.cod
) info
ON r.cod = info.cod;
```

Result

	COD	DESIGNATION	N_FACILITIES
1	1	Norte	3
2	2	Centro	11
3	3	Lisboa	6
4	4	Alentejo	43
5	5	Algarve	1
6	6	Açores	0
7	7	Madeira	0

Question 3

How many municipalities do not have any facility with an activity of 'cinema'?

Query

```
-- 3
SELECT COUNT(m.cod)
FROM municipalities m
WHERE m.cod NOT IN
(
    SELECT m.cod FROM municipalities m, TABLE(m.facilities) f,
TABLE(VALUE(f).activities) a
    WHERE VALUE(a).activity = 'cinema'
);

---- 3 with functions
SELECT COUNT(m.cod) FROM municipalities m WHERE
m.municipalityHasActivity('cinema') = 0;
```

Result

	COUNT(M.COD)
1	100

Question 4

Which is the municipality with more facilities engaged in each of the six kinds of activities? Show the activity, the municipality name and the corresponding number of facilities.

Query

```
-- 4
```

```
SELECT t1.activity, t1.designation, t1.n_facilities
FROM
(
    SELECT VALUE(a).ACTIVITY activity, m.COD, m.DESIGNATION,
COUNT(VALUE(f).ID) n_facilities
    FROM municipalities m, TABLE(m.facilities) f,
TABLE(VALUE(f).activities) a
    GROUP BY VALUE(a).ACTIVITY, m.COD, m.DESIGNATION
) t1
LEFT OUTER JOIN
(
    SELECT VALUE(a).ACTIVITY activity, m.COD, m.DESIGNATION,
COUNT(VALUE(f).ID) n_facilities
    FROM municipalities m, TABLE(m.facilities) f,
TABLE(VALUE(f).activities) a
    GROUP BY VALUE(a).ACTIVITY, m.COD, m.DESIGNATION
) t2 ON (t1.activity = t2.activity AND t1.n_facilities <
t2.n_facilities)
WHERE t2.ACTIVITY IS NULL
ORDER BY t1.ACTIVITY;
```

Result

	ACTIVITY	DESIGNATION	N_FACILITIES
1	cinema	Lisboa	96
2	circo	Lisboa	2
3	dança	Lisboa	47
4	música	Lisboa	77
5	tauromaquia	Moura	4
6	teatro	Lisboa	66

Question 5

Which are the codes and designations of the districts with facilities in all the municipalities?

Query

```
-- 5
SELECT cod, designation
FROM districts
WHERE cod NOT IN
(
    SELECT DISTINCT d.cod
    FROM districts d, TABLE(d.municipalities) m
```

```
    WHERE (d.cod, VALUE(m).cod) NOT IN
    (
        SELECT d.cod, VALUE(m).cod
        FROM districts d, TABLE(d.municipalities) m,
TABLE(VALUE(m).facilities) f
    )
) ORDER BY cod;

---- 5 with functions
SELECT cod, designation
FROM districts
WHERE cod NOT IN
(
    SELECT DISTINCT d.cod
    FROM districts d, TABLE(d.municipalities) m
    WHERE m.totalFacilities() = 0
) ORDER BY cod;
```

Result

	COD	DESIGNATION
1	7	Évora
2	11	Lisboa
3	12	Portalegre
4	15	Setúbal

Question 6

Add a query that illustrates the use of OR extensions.

Note: the group was developing another query with more complexity but due to the problems faced regarding the Oracle server, it remained unfinished. The query below was not tested for the same reason, leaving it without the results.

Query

```
SELECT r.designation, r.totalFacilities() as  totalFacilities
FROM regions r
ORDER BY totalFacilities DESC;
```

Conclusion

With the development of the project, the group learned about the importance of combining Object-Oriented technology with the relational database. It allows the definition of more complex types, removes the need to perform join operations by using references to access other tables and also allows the use of member functions, which reduces the overall cost.
Due to problems related to the Oracle server at FEUP, the group was unable to test the queries and functions created for Question 6.