

## Tecnologia de Bases de Dados

Mestrado Integrado em

Engenharia Informática e Computação



## **OBJECT RELATIONAL ASSIGNMENT**

Trabalho realizado por:

Fabiola Figueira da Silva - UP201502850

Pedro Filipe Agrela Faria - UP201406992

Pedro Manuel Monteiro Albano - UP201008982

Faculdade de Engenharia da Universidade do Porto  
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

14 de Maio de 2018

# 1 Conteúdo

<b>2 Objetivos do trabalho</b>	2
<b>3 Criação da Base de Dados</b>	2
3.1 Esquema	2
3.2 Definição da linguagem (DDL)	3
3.3 Povoamento dos objectos	3
3.4 Funções criadas	3
3.4.1 list_activities	3
3.4.2 supports_activity	4
<b>4 Execução das Queries em OR BD</b>	4
4.1 Query (a)	4
4.1.1 Descrição e código sql	4
4.1.2 Resultado da query	4
4.2 Query (b)	5
4.2.1 Descrição e código sql	5
4.2.2 Resultado da query	5
4.3 Query (c)	5
4.3.1 Descrição e código sql	5
4.3.2 Resultado da query	6
4.4 Query (d)	6
4.4.1 Descrição e código sql	6
4.4.2 Resultado da query	6
4.5 Query (e)	7
4.5.1 Descrição e código sql	7
4.5.2 Resultado da query	8
4.6 Query (f)	8
4.6.1 Descrição e código sql	8
4.6.2 Resultado da query	9
<b>5 Anexos</b>	9
5.1 Anexo 1 - Código em SQL usado na criação do modelo de objetos relacionais.	9
5.2 Anexo 2 - Código em SQL do povoamento do modelo de objetos relacionais	12

## 2 Objetivos do trabalho

O objetivo principal deste trabalho é o desenvolvimento de uma base de dados em SQL3 (Modelo objeto-relacional), fazendo uso assim de objetos que contém dados estruturados, funções, heranças, tabelas aninhadas e referências de objetos. Um outro objetivo deste trabalho é popular o nosso modelo de objetos relacionais com os dados existentes na base de dados relacional fornecida pelo docente, disponíveis em GTD8. Foi ainda desenhado um esquema que representa o nosso modelo de objetos relacionais.

## 3 Criação da Base de Dados

### 3.1 Esquema

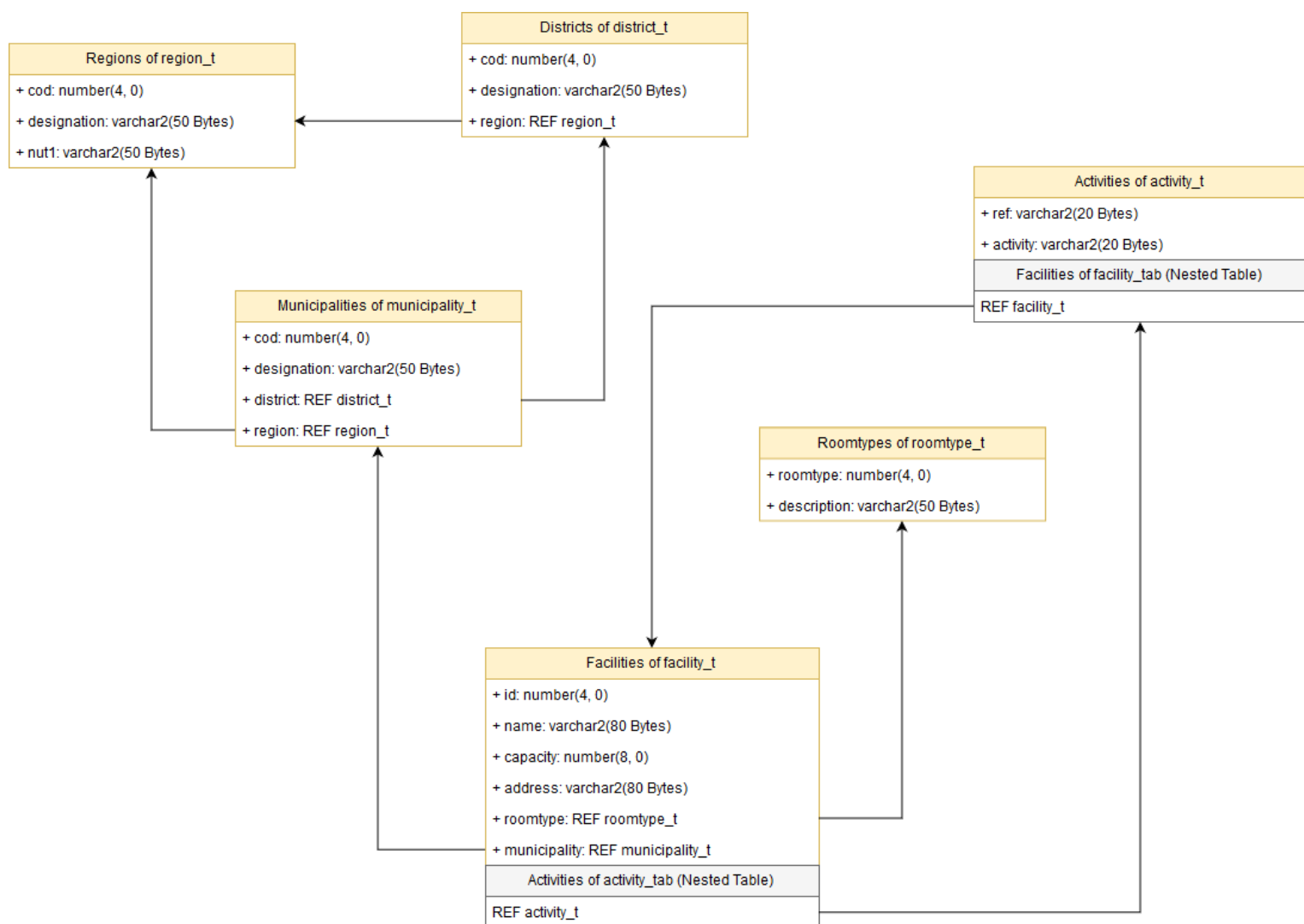


Figura 1 - Esquema do modelo da base de dados

## 3.2 Definição da linguagem (DDL)

O trabalho baseia-se na manipulação de eventos/atividades em recintos presentes em municípios e regiões. Desta forma criamos os seguintes objetos:

- **Regions do tipo region\_t:** informação das regiões.
- **Districts do tipo dsitRICT\_t:** informação dos distritos (um distrito pode estar contido em uma ou duas regiões).
- **Municipalities do tipo municipatILITY\_t:** informação dos municípios.
- **Facilities do tipo facility\_t:** informação dos recintos.
- **Roomtypes do tipo roomtype\_t:** informação do tipo de recinto.
- **Activities do tipo activity\_t:** informação das actividades existentes no recinto.

A tabela Uses indicado no esquema do enunciado foi implementada com representação assimétrica.

O anexo 1, representa o código em SQL usado na criação destes objetos.

## 3.3 Povoamento dos objectos

No anexo 2, é possível verificar o código em SQL do povoamento do modelo de objetos relacionais com os dados existentes na base de dados relacional fornecida pelo docente.

## 3.4 Funções criadas

### 3.4.1 list\_activities

Devolve uma lista de actividades numa string de um recinto.

```
create or replace function list_activities(facility in facility_t)
return varchar2 as

cursor activity_cursor is (select value(t).activity as activity from table(facility.activities) t);
actList varchar2(200) := '';
fst number := 0;
begin
    for i in activity_cursor
    loop
        if fst = 0 then
            actList := i.activity;
            fst := 1;
        else
            actList := actList || ', ' || i.activity;
        end if;
    end loop;
    return(actList);
end;
```

Figura 2 - Função list\_activities

### 3.4.2 supports\_activity

Verifica se o recinto(facility) tem a capacidade (users) e a actividade (activity) de entrada. Retorna 1 se verifica estas condições, -1 caso contrário.

```
create or replace function supports_activity(facility in facility_t, users in number, activity in activity_t)
return number as
found number := 0;
begin
select count(*) into found from table(facility.activities) t where value(t).ref = activity.ref;
if found > 0 and facility.capacity >= users then
return(1);
else
return(-1);
end if;
end;
```

Figura 3 - Função supports\_activity

## 4 Execução das Queries em OR BD

### 4.1 Query (a)

#### 4.1.1 Descrição e código sql

Mostrar o id, nome, descrição e actividade na qual os recintos têm "touros" nas descrição do room type e "teatro" como actividade

```
--4.a)
SELECT f.id as id, f.name as name, f.roomtype.description as description, value(act).activity as activity
FROM facilities f, table(f.activities) act
WHERE value(act).activity = 'teatro' and f.roomtype.description Like '%touros%'
ORDER BY f.id;
```

Figura 4 - Query (a)

#### 4.1.2 Resultado da query

	ID	NAME	DESCRIPTION	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

Figura 5 - Resultado da query (a)

## 4.2 Query (b)

### 4.2.1 Descrição e código sql

Contagem dos recintos com room type "touros" por região

```
--4.b)
SELECT f.municipality.region.designation as Region, count(f.id) as Count
FROM facilities f
WHERE f.roomtype.description Like '%touros%'
GROUP by f.municipality.region.designation;
```

Figura 6 - Query (b)

### 4.2.2 Resultado da query

REGION	COUNT
1 Norte	3
2 Algarve	1
3 Lisboa	6
4 Centro	11
5 Alentejo	43

Figura 7 - Resultado da query (b)

## 4.3 Query (c)

### 4.3.1 Descrição e código sql

Contagem dos municípios que não têm recintos cuja atividade é "cinema"

```
--4.c)
SELECT count(*) as "Municipalities without cinema"
FROM municipalities m
WHERE ref(m) NOT IN (SELECT DISTINCT f.municipality
FROM facilities f, table(f.activities) a
WHERE value(a).activity = 'cinema');
```

Figura 8 - Query (c)

### 4.3.2 Resultado da query

Municipalities without cinema	
1	100

Figura 9 - Resultado da query (c)

## 4.4 Query (d)

### 4.4.1 Descrição e código sql

Mostrar a atividade, o nome do município e o número correspondente de recintos, no qual calcula quais os município que têm mais recintos em cada uma dos 6 tipos de atividades

```
--4.d)
with a1 as(
  select f.municipality.designation as municipality, value(a).activity as activity, count(value(f)) as ammount
  from facilities f, table(f.activities) a
  group by f.municipality, value(a).activity
), a2 as(
  select activity, max(ammount) as ammount from a1 group by activity
) select a1.municipality, a1.activity, a1.ammount
from a1 inner join a2 on a1.ammount = a2.ammount and a1.activity = a2.activity;
```

Figura 10 - Query (d)

### 4.4.2 Resultado da query

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

Figura 11 - Resultado da query (d)

## 4.5 Query (e)

### 4.5.1 Descrição e código sql

Mostrar o código e designação dos distritos com recintos em todos os municípios

```
--4.e)
with a1 as (
  -- Facility -> Municipality, District
  select
    f.id as facility,
    f.municipality as municipality,
    f.municipality.district as district
  from facilities f
), a2 as (
  -- Number of Facilities per Municipality/District
  select
    count(facility) as facilities,
    municipality,
    district
  from a1
  group by municipality, district
), a3 as (
  -- Number of Municipalities with Facilities per District
  select
    district,
    count(municipality) as mwfacilities
  from a2
  group by district
), a4 as (
  -- Number of Municipalities per District
  select
    m.district,
    count(m.cod) as municipalities
  from municipalities m
  group by m.district
)
select
  q.district.cod as code,
  q.district.designation as designation
from a3 q
where exists(select * from a4)
order by code;
```

Figura 12 - Query (e)



## 4.5.2 Resultado da query

	CODE	DESIGNATION
1	1	Aveiro
2	2	Beja
3	3	Braga
4	4	Bragança
5	5	Castelo Branco
6	6	Coimbra
7	7	Évora
8	8	Faro
9	9	Guarda
10	10	Leiria
11	11	Lisboa
12	12	Portalegre
13	13	Porto
14	14	Santarém
15	15	Setúbal
16	16	Viana do Castelo
17	17	Vila Real
18	18	Viseu

Figura 13 - Resultado da query (e)

## 4.6 Query (f)

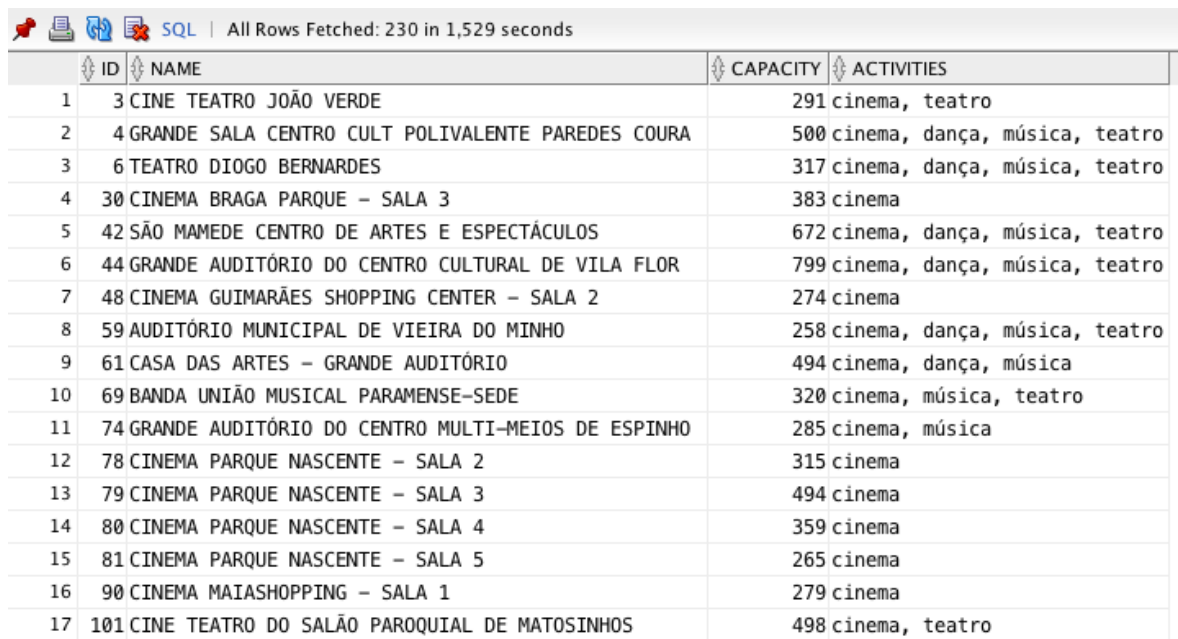
### 4.6.1 Descrição e código sql

Obter os recintos com capacidade igual ou superior a 200, cuja actividade contém cinema.

```
--4.f)
-- Facilities with 200 or more capacity, that include 'cinema' as an activity
select f.id, f.name, f.capacity, list_activities(value(f)) as activities
from facilities f
where supports_activity(
    value(f),
    250,
    (select value(a) from activities a where a.activity = 'cinema')
) > 0
order by f.id;
```

Figura 14 - Query (f)

## 4.6.2 Resultado da query



ID	NAME	CAPACITY	ACTIVITIES
1	3 CINE TEATRO JOÃO VERDE	291	cinema, teatro
2	4 GRANDE SALA CENTRO CULT POLIVALENTE PAREDES COURA	500	cinema, dança, música, teatro
3	6 TEATRO DIOGO BERNARDES	317	cinema, dança, música, teatro
4	30 CINEMA BRAGA PARQUE – SALA 3	383	cinema
5	42 SÃO MAMEDE CENTRO DE ARTES E ESPECTÁCULOS	672	cinema, dança, música, teatro
6	44 GRANDE AUDITÓRIO DO CENTRO CULTURAL DE VILA FLOR	799	cinema, dança, música, teatro
7	48 CINEMA GUIMARÃES SHOPPING CENTER – SALA 2	274	cinema
8	59 AUDITÓRIO MUNICIPAL DE VIEIRA DO MINHO	258	cinema, dança, música, teatro
9	61 CASA DAS ARTES – GRANDE AUDITÓRIO	494	cinema, dança, música
10	69 BANDA UNIÃO MUSICAL PARAMENSE–SEDE	320	cinema, música, teatro
11	74 GRANDE AUDITÓRIO DO CENTRO MULTI-MEIOS DE ESPINHO	285	cinema, música
12	78 CINEMA PARQUE NASCENTE – SALA 2	315	cinema
13	79 CINEMA PARQUE NASCENTE – SALA 3	494	cinema
14	80 CINEMA PARQUE NASCENTE – SALA 4	359	cinema
15	81 CINEMA PARQUE NASCENTE – SALA 5	265	cinema
16	90 CINEMA MAIA SHOPPING – SALA 1	279	cinema
17	101 CINE TEATRO DO SALÃO PAROQUIAL DE MATOSINHOS	498	cinema, teatro

Figura 15 - Resultado da query (f)

## 5 Anexos

### 5.1 Anexo 1 - Código em SQL usado na criação do modelo de objetos relacionais.

```
-- 1) Define Data Models and Import Data
-- Types
create type roomtype_t as object(
    roomtype number(4, 0),
    description varchar2(50)
);
/
create type region_t as object(
    cod number(4, 0),
    designation varchar2(50),
    nut1 varchar2(50)
);
/
create type district_t as object(
    cod number(4, 0),
    designation varchar2(50),
    region ref region_t
);
/
```

```

create type municipality_t as object(
    cod number(4, 0),
    designation varchar2(50),
    district ref district_t,
    region ref region_t
);
/
create type activity_t as object(
    ref varchar2(20),
    activity varchar2(20)
);
/
create type activity_tab as table of ref activity_t;
/
create type facility_t as object(
    id number(4, 0),
    name varchar2(80),
    capacity number(8, 0),
    address varchar2(80),
    roomtype ref roomtype_t,
    municipality ref municipality_t,
    activities activity_tab
);
/
create type facility_tab as table of ref facility_t;
/
alter type activity_t add attribute facilities facility_tab cascade;
/

-- Tables
create table roomtypes of roomtype_t(
    primary key(roomtype)
);
/
create table regions of region_t(
    primary key(cod)
);
/
create table districts of district_t(
    primary key(cod)
);
/
create table municipalities of municipality_t(
    primary key(cod)
);
/
create table activities of activity_t(
    primary key(ref)
) nested table facilities store as facilities_nt;
/

```

```

create table facilities of facility_t(
    primary key(id)
) nested table activities store as activities_nt;
/

-- 2) Inserts
insert into roomtypes select roomtype_t(roomtype, description) from rl_roomtypes;
/
insert into regions select region_t(cod, designation, nut1) from rl_regions;
/
insert into districts
with aux1 as (
    select rl_districts.cod, rl_districts.designation, ref(r) as region
    from rl_districts
    left outer join regions r
    on rl_districts.region = r.cod
)
select district_t(cod, designation, region) from aux1;
/
insert into municipalities
with aux1 as (
    select rl_municipalities.cod, rl_municipalities.designation,
    ref(d) as district, rl_municipalities.region
    from rl_municipalities
    left outer join districts d
    on rl_municipalities.district = d.cod
), aux2 as (
    select aux1.cod, aux1.designation, aux1.district, ref(r) as region
    from aux1
    left outer join regions r
    on aux1.region = r.cod
)
select municipality_t(cod, designation, district, region) from aux2;
/
insert into activities select activity_t(ref, activity, facility_tab()) from
rl_activities;
/
insert into facilities
with aux1 as (
    select rl_facilities.id, rl_facilities.name, rl_facilities.capacity,
    rl_facilities.address,
    ref(rt) as roomtype, rl_facilities.municipality
    from rl_facilities
    left outer join roomtypes rt
    on rl_facilities.roomtype = rt.roomtype
), aux2 as (
    select id, name, capacity, address, roomtype, ref(m) as municipality
    from aux1
    left outer join municipalities m
    on aux1.municipality = m.cod
)

```

```

select facility_t(id, name, capacity, address, roomtype, municipality,
activity_tab()) from aux2;
/
begin
  for u in (select id, ref from rl_uses)
  loop
    insert into table(select f.activities from facilities f where u.id = f.id)
    select ref(a) from activities a where u.ref = a.ref;
    insert into table(Select a.facilities from activities a where u.ref =
a.ref)
    select ref(f) from facilities f where u.id = f.id;
  end loop;
end;
/

```

## 5.2 Anexo 2 - Código em SQL do povoamento do modelo de objetos relacionais

```

-- Clone Relational Table Structure + Data
create table rl_activities as (select * from gtd8.activities);
create table rl_uses as (select * from gtd8.uses);
create table rl_facilities as (select * from gtd8.facilities);
create table rl_roomtypes as (select * from gtd8.roomtypes);
create table rl_districts as (select * from gtd8.districts);
create table rl_municipalities as (select * from gtd8.municipalities);
create table rl_regions as (select * from gtd8.regions);

-- Install Constraints
alter table rl_activities add constraint pk_rl_activities primary key(ref);
alter table rl_uses add constraint pk_rl_uses primary key(id, ref);
alter table rl_facilities add constraint pk_rl_facilities primary key(id);
alter table rl_roomtypes add constraint pk_rl_roomtypes primary key(roomtype);
alter table rl_municipalities add constraint pk_rl_municipalities primary key(cod);
alter table rl_districts add constraint pk_rl_districts primary key(cod);
alter table rl_regions add constraint pk_rl_regions primary key(cod);

alter table rl_uses add constraint fk_rl_uses_activities foreign key(ref)
references rl_activities(ref);
alter table rl_uses add constraint fk_rl_uses_facilities foreign key(id) references
rl_facilities(id);
alter table rl_facilities add constraint fk_rl_facilities_roomt foreign
key(roomtype) references rl_roomtypes(roomtype);
alter table rl_facilities add constraint fk_rl_facilities_munip foreign
key(municipality) references rl_municipalities(cod);
alter table rl_municipalities add constraint fk_rl_municipalities_dist foreign
key(district) references rl_districts(cod);
alter table rl_municipalities add constraint fk_rl_municipalities_reg foreign
key(region) references rl_regions(cod);
alter table rl_districts add constraint fk_rl_districts_reg foreign key(region)
references rl_regions(cod);

```