Modelo Objeto-Relacional

Mestrado Integrado em Engenharia Informática e Computação

Tecnologias de Bases de Dados
Ano Letivo 2018/19, 2º Semestre

Francisco Tuna de Andrade - 201503481
João Paulo Madureira Damas - 201504088

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

30 de Abril de 2019

Conteúdo

1 Sumário

No âmbito da cadeira de Tecnologia de Bases de Dados do Mestrado Integrado em Engenharia Informática e Computação, no ano letivo 2018/2019, este trabalho tem como objetivo implementar um modelo que faça uso das extensões proporcionadas pelo modelo objeto-relacional, por exemplo, ao nível de tipos de dados personalizados, funções utilitárias associadas, herança, referências e métodos para ordenação e/ou comparação. Assim, parte de um esquema relacional, a partir do qual um novo modelo, com recurso a estas extensões, é desenvolvido. De seguida, algumas funções comuns úteis são identificadas e algumas interrogações feitas para confirmar a correção e completude do modelo.

2 Modelo objeto-relacional

2.1 Modelo

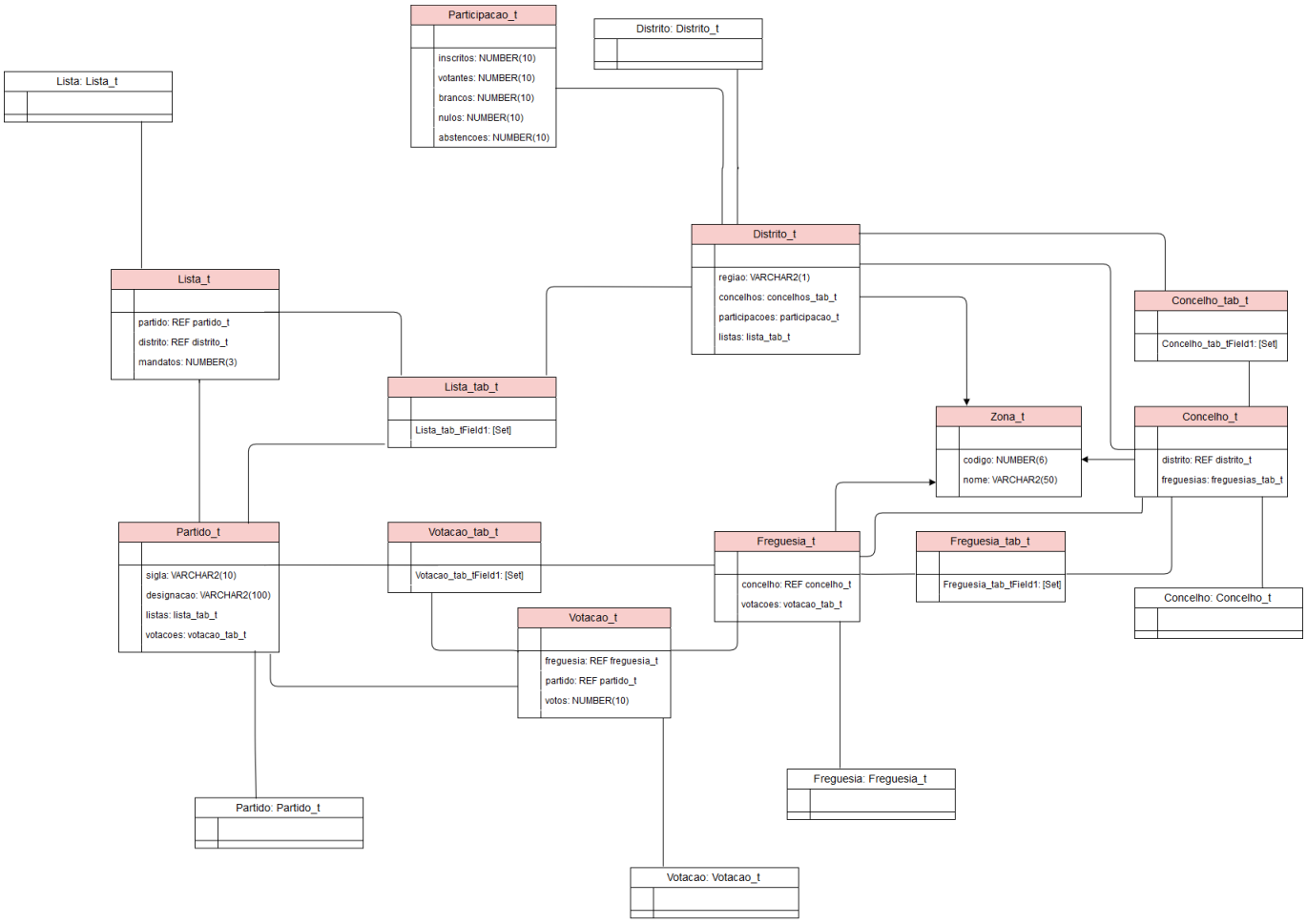

Figura 1: Modelo Objeto-Relacional obtido.

2.2 Decisões relevantes

Ao transpor o modelo relacional para um modelo objeto-relacional, tentou-se fazer um bom uso das extensões SQL3, bem como respeitar as convenções de mapeamento normalmente utilizadas.

Assim, de notar desde logo a criação de tipos para cada entidade identificada no modelo relacional original. Quanto a relações entre as mesmas, é possível identificar três tipos: um para um, um para muitos e muitos para muitos.

A associação Distrito - Participação é de multiplicidade 1-1. Desta forma, considerou-se desnecessária a criação de duas tabelas distintas, preferindo criar apenas uma para guardar a informação relativa aos distritos, já que esta é mais relevante para referências a outras tabelas implementadas. Em Distrito adicionou-se, então, uma coluna do tipo participacao_t para guardar a informação relativa à participação nas eleições daquele distrito.

Nas diferentes decomposições do território nacional (freguesias, concelhos, distritos), verificam-se relações de um para muitos, para além de uma estrutura comum ao nível de código e nome. Nesse sentido, foi criado um tipo base estruturante para ajudar a definir as três entidades. Inicialmente, foi considerada uma única tabela, de forma a fazer uso de interrogações recursivas, no entanto as diferentes associações com outras entidades levaram à implementação de tabelas separadas para cada uma. Quanto às associações em si, optou-se por referências bidirecionais (uma do lado de muitos, uma coleção de referências do lado de um) visto que a alternativa (inclusão) seria bastante limitadora e complexa tendo em conta as associações distintas das diferentes decomposições com outras entidades.

Finalmente, nas associações muitos para muitos (ao nível das listas de cada partido em cada distrito e dos votos de cada partido em cada freguesia), foi utilizada uma tabela intermédia com referências para os objetos envolvidos. A escolha deve-se, sobretudo, ao facto das associações em causa representarem classes de associação com atributos próprios. De forma a estabelecer bidirecionalidade e já que interrogações comuns nestas tabelas envolvem filtragem de tuplos da classe de associação em que um dos lados está envolvido, foram adicionadas coleções de referência a cada uma das tabelas envolvidas.

2.3 *Script* criação

Abaixo encontra-se o *script* SQL para (re)criar o esquema na BD.

```
-- type creation
create or replace type zona_t as object(
    codigo number(6),
    nome varchar2(50),
    not instantiable member function votos_partido(sigla_partido varchar2) return number,
    not instantiable member function total_votos return number
) not instantiable not final;

create or replace type participacao_t as object(
    inscritos number(10),
    votantes number(10),
    abstencoes number(10),
    brancos number(10),
    nulos number(10)
);
```

```
create or replace type distrito_t under zona_t (
    regiao varchar2(1),
    participacao participacao_t,
    overriding member function votos_partido(sigla_partido varchar2) return number,
    overriding member function total_votos return number
);

create or replace type concelho_t under zona_t (
    distrito ref distrito_t,
    overriding member function  votos_partido(sigla_partido varchar2) return number,
    overriding member function total_votos return number
);

create or replace type freguesia_t under zona_t (
    concelho ref concelho_t,
    overriding member function votos_partido(sigla_partido varchar2) return number,
    overriding member function total_votos return number
);

create or replace type partido_t as object (
    sigla varchar2(10),
    designacao varchar2(100)
);

create or replace type lista_t as object (
    distrito ref distrito_t,
    partido ref partido_t,
    mandatos number(3)
);

create or replace type votacao_t as object (
    freguesia ref freguesia_t,
    partido ref partido_t,
    votos number(10)
);

create or replace type freguesia_tab_t as table of ref freguesia_t;

create or replace type concelho_tab_t as table of ref concelho_t;

create or replace type lista_tab_t as table of ref lista_t;

create or replace type votacao_tab_t as table of ref votacao_t;


alter type distrito_t add attribute (listas lista_tab_t, concelhos concelho_tab_t) cascade;

alter type concelho_t add attribute (freguesias freguesia_tab_t) cascade;

alter type freguesia_t add attribute (votacoes votacao_tab_t) cascade;

alter type partido_t add attribute (listas lista_tab_t, votacoes votacao_tab_t) cascade;

-- table creation
create table distrito of distrito_t
    nested table listas store as d_listas_tab
```

```
    nested table concelhos store as concelhos_tab;

create table concelho of concelho_t
    nested table freguesias store as freguesias_tab;

create table freguesia of freguesia_t
    nested table votacoes store as f_votacoes_tab;

create table partido of partido_t
    nested table listas store as p_listas_tab
    nested table votacoes store as p_votacoes_tab;

create table lista of lista_t;

create table votacao of votacao_t;
```

3 Povoamento

Abaixo encontra-se o *script* SQL para povoar a BD.

```
insert into partido (sigla, designacao)
select *
from gtd7.partidos;

insert into distrito (codigo, nome, regiao, participacao)
select d.*, participacao_t(p.inscritos, p.votantes, p.abstencoes, p.brancos, p.nulos)
from gtd7.distritos d
inner join gtd7.participacoes p on d.codigo = p.distrito;

insert into concelho (codigo, nome, distrito)
select c.codigo, c.nome, ref(d)
from gtd7.concelhos c
inner join distrito d on c.distrito = d.codigo;

insert into freguesia (codigo, nome, concelho)
select f.codigo, f.nome, ref(c)
from gtd7.freguesias f
inner join concelho c on f.concelho = c.codigo;

insert into lista (distrito, partido, mandatos)
select ref(d), ref(p), l.mandatos
from distrito d
inner join gtd7.listas l on d.codigo = l.distrito
inner join partido p on l.partido = p.sigla;

insert into votacao (freguesia, partido, votos)
select ref(f), ref(p), v.votos
from freguesia f
inner join gtd7.votacoes v on f.codigo = v.freguesia
inner join partido p on v.partido = p.sigla;

-- atualizacoes (devido a referencias circulares)
```

```
update distrito d
set d.concelhos = cast(multiset(select ref(c) from concelho c where c.distrito.codigo =
↪  d.codigo) as concelho_tab_t),
d.listas = cast(multiset(select ref(l) from lista l where l.distrito.codigo = d.codigo) as
↪  lista_tab_t);

update partido p
set p.votacoes = cast(multiset(select ref(v) from votacao v where v.partido.sigla = p.sigla) as
↪  votacao_tab_t),
p.listas = cast(multiset(select ref(l) from lista l where l.partido.sigla = p.sigla) as
↪  lista_tab_t);

update concelho c
set c.freguesias = cast(multiset(select ref(f) from freguesia f where f.concelho.codigo =
↪  c.codigo) as freguesia_tab_t);

update freguesia f
set f.votacoes = cast(multiset(select ref(v) from votacao v where v.freguesia.codigo =
↪  f.codigo) as votacao_tab_t);
```

4 Métodos úteis

4.1 Zona

Foram implementados os seguintes métodos:

- `NOT INSTANTIABLE MEMBER FUNCTION votos_partido(sigla_partido varchar2) RETURN NUMBER`
  - Retorna o número de votos num determinado partido
- `NOT INSTANTIABLE MEMBER FUNCTION total_votos RETURN NUMBER`
  - Retorna o número total de votos nessa zona
- `MEMBER FUNCTION partido_vencedor RETURN REF partido_t`
  - Retorna uma referência para o partido vencedor nessa zona
- `MEMBER FUNCTION ratio_votos_partido_vencedor`
  - Retorna o rácio de votos obtido pelo partido vencedor

Abaixo, encontra-se o *script* com a implementação dos métodos: (Note-se que os métodos *NOT INSTANTIABLE* foram declarados aquando da definição do tipo zona, no próprio *script* de criação da base de dados)

```
ALTER TYPE zona_t ADD MEMBER FUNCTION partido_vencedor RETURN REF partido_t CASCADE;
ALTER TYPE zona_t ADD MEMBER FUNCTION ratio_votos_partido_vencedor RETURN NUMBER CASCADE;

CREATE OR REPLACE TYPE BODY zona_t AS

    MEMBER FUNCTION partido_vencedor RETURN REF partido_t IS
```

```
    ret_variable REF partido_t;
    BEGIN
        SELECT REF(p) INTO ret_variable FROM partido p WHERE
            NOT EXISTS
                (SELECT * FROM partido p2
                WHERE (p.sigla != p2.sigla AND SELF.votos_partido(p.sigla) <
                 ↪  SELF.votos_partido(p2.sigla)) OR
                (SELF.votos_partido(p.sigla) = SELF.votos_partido(p2.sigla) AND p.sigla >
                 ↪  p2.sigla));

        RETURN ret_variable;
    END partido_vencedor;

    MEMBER FUNCTION ratio_votos_partido_vencedor RETURN NUMBER IS
    ret_variable NUMBER; tv NUMBER;
    BEGIN
        tv := SELF.total_votos();
        IF tv = 0 THEN
            RETURN 0;
        ELSE
            SELECT MAX(SELF.votos_partido(p.sigla) / tv) INTO ret_variable FROM partido p;
        END IF;

        RETURN ret_variable;
    END ratio_votos_partido_vencedor;
END;
```

4.2 Freguesia

Foram implementados os seguintes métodos:

- `OVERRIDING MEMBER FUNCTION votos_partido(sigla_partido VARCHAR2) RETURN NUMBER`
  - Retorna o número de votos num determinado partido
- `OVERRIDING MEMBER FUNCTION total_votos RETURN NUMBER`
  - Retorna o número total de votos nessa freguesia

Abaixo, encontra-se o *script* com a implementação dos métodos: (Note-se que os métodos por serem *OVERRIDING* foram declarados aquando da definição do tipo freguesia, no próprio *script* de criação da base de dados)

```
CREATE OR REPLACE TYPE BODY freguesia_t AS

     OVERRIDING MEMBER FUNCTION votos_partido(sigla_partido VARCHAR2) RETURN NUMBER IS
    ret_variable NUMBER;
    BEGIN
        SELECT NVL(SUM(value(v).votos), 0) INTO ret_variable
        FROM TABLE(SELF.votacoes) v
        WHERE value(v).partido.sigla = sigla_partido;

        RETURN ret_variable;
```

```
    END votos_partido;

    OVERRIDING MEMBER FUNCTION total_votos RETURN NUMBER IS
    ret_variable NUMBER;
    BEGIN
        SELECT SUM(value(v).votos) INTO ret_variable
        FROM TABLE(SELF.votacoes) v;

        RETURN ret_variable;
    END total_votos;
END;
```

4.3 Concelho

Foram implementados os seguintes métodos:

- `OVERRIDING MEMBER FUNCTION votos_partido(sigla_partido VARCHAR2) RETURN NUMBER`
  - Retorna o número de votos num determinado partido
- `OVERRIDING MEMBER FUNCTION total_votos RETURN NUMBER`
  - Retorna o número total de votos nesse concelho

Abaixo, encontra-se o *script* com a implementação dos métodos: (Note-se que os métodos por serem *OVERRIDING* foram declarados aquando da definição do tipo concelho, no próprio *script* de criação da base de dados)

```
CREATE OR REPLACE TYPE BODY concelho_t AS

    OVERRIDING MEMBER FUNCTION votos_partido(sigla_partido VARCHAR2) RETURN NUMBER IS
    ret_variable NUMBER;
    BEGIN
        SELECT NVL(SUM(value(v).votos), 0) INTO ret_variable
        FROM TABLE(SELF.freguesias) f, TABLE(value(f).votacoes) v
        WHERE value(v).partido.sigla = sigla_partido;

        RETURN ret_variable;
    END votos_partido;

    OVERRIDING MEMBER FUNCTION total_votos RETURN NUMBER IS
    ret_variable NUMBER;
    BEGIN
        SELECT SUM(value(v).votos) INTO ret_variable
        FROM TABLE(SELF.freguesias) f, TABLE(value(f).votacoes) v;

        RETURN ret_variable;
    END total_votos;
END;
```

4.4 Distrito

Foram implementados os seguintes métodos:

- `OVERRIDING MEMBER FUNCTION votos_partido(sigla_partido VARCHAR2) RETURN NUMBER`
  - Retorna o número de votos num determinado partido nesse distrito
- `OVERRIDING MEMBER FUNCTION total_votos RETURN NUMBER`
  - Retorna o número total de votos nesse distrito
- `MEMBER FUNCTION integrity RETURN NUMBER`
  - Retorna 1 se este distrito cumprir a regra de integridade, 0 caso contrário
- `MEMBER FUNCTION ratio_mandatos_partido_vencedor RETURN NUMBER`
  - Retorna o rácio de mandatos obtidos pelo partido vencedor nesse distrito

Abaixo, encontra-se o *script* com a implementação dos métodos: (Note-se que os métodos *OVERRIDING* foram declarados aquando da definição do tipo distrito, no próprio *script* de criação da base de dados)

```
CREATE OR REPLACE TYPE BODY distrito_t AS

    OVERRIDING MEMBER FUNCTION total_votos RETURN NUMBER IS
    ret_variable NUMBER;
    BEGIN
        SELECT SUM(value(v).votos) INTO ret_variable
        FROM TABLE(SELF.concelhos) c, TABLE(value(c).freguesias) f, TABLE(value(f).votacoes) v;

        RETURN ret_variable;
    END total_votos;

    OVERRIDING MEMBER FUNCTION votos_partido(sigla_partido VARCHAR2) RETURN NUMBER IS
    ret_variable NUMBER;
    BEGIN
        SELECT NVL(SUM(value(v).votos), 0) INTO ret_variable
        FROM TABLE(SELF.concelhos) c, TABLE(value(c).freguesias) f, TABLE(value(f).votacoes) v
        WHERE value(v).partido.sigla = sigla_partido;

        RETURN ret_variable;
    END votos_partido;

    MEMBER FUNCTION integrity RETURN NUMBER IS
    ret_variable NUMBER;
    BEGIN
        IF SELF.total_votos() + SELF.participacao.abstencoes + SELF.participacao.brancos +
         ↪  SELF.participacao.nulos !=  SELF.participacao.inscritos THEN
            ret_variable := 0;
        ELSE
            ret_variable := 1;
        END IF;

        RETURN ret_variable;
    END integrity;
```

```
    MEMBER FUNCTION ratio_mandatos_partido_vencedor RETURN NUMBER IS
    ret_variable NUMBER; tm NUMBER;
    BEGIN
        SELECT SUM(value(l).mandatos) INTO tm
        FROM TABLE(SELF.listas) l;
        IF tm = 0 THEN
            RETURN 0;
        ELSE
            SELECT MAX(value(l).mandatos / tm) INTO ret_variable
            FROM partido p JOIN TABLE(SELF.listas) l
            ON p.sigla = value(l).partido.sigla;
        END IF;

        RETURN ret_variable;
    END ratio_mandatos_partido_vencedor;

END;
```

4.5 Partido

Foram implementados os seguintes métodos:

- `MAP MEMBER FUNCTION total_votos RETURN NUMBER`

  - Retorna o número total de votos nesse partido

- `MEMBER FUNCTION total_mandatos RETURN NUMBER`

  - Retorna o número total de mandatos nesse partido

- `MEMBER FUNCTION best_ratio_district RETURN best_ratio_ret_t`

  - Retorna o distrito onde este partido obtive o maior rácio de votos assim como o rácio obtido

Abaixo, encontra-se o *script* com a implementação dos métodos:

```
ALTER TYPE partido_t ADD MAP MEMBER FUNCTION total_votos RETURN NUMBER CASCADE;
ALTER TYPE partido_t ADD MEMBER FUNCTION total_mandatos RETURN NUMBER CASCADE;

CREATE OR REPLACE TYPE best_ratio_ret_t AS OBJECT(
    ratio NUMBER,
    dist REF distrito_t
)

ALTER TYPE partido_t ADD MEMBER FUNCTION best_ratio_district RETURN best_ratio_ret_t CASCADE;

CREATE OR REPLACE TYPE BODY partido_t AS

    MAP MEMBER FUNCTION total_votos RETURN NUMBER IS
    ret_variable NUMBER;
    BEGIN
        SELECT NVL(SUM(value(v).votos), 0) INTO ret_variable
```

```
        FROM TABLE(SELF.votacoes) v;

        RETURN ret_variable;
    END total_votos;

   MEMBER FUNCTION total_mandatos RETURN NUMBER IS
   ret_variable NUMBER;
   BEGIN
        SELECT NVL(SUM(value(l).mandatos), 0) INTO ret_variable
        FROM TABLE(SELF.listas) l;

        RETURN ret_variable;
   END total_mandatos;

   MEMBER FUNCTION best_ratio_district RETURN best_ratio_ret_t IS
   ret_variable best_ratio_ret_t;
   BEGIN
       ret_variable := best_ratio_ret_t(NULL, NULL);
       SELECT tmp.refr, tmp.ratio INTO ret_variable.dist, ret_variable.ratio
       FROM
       (SELECT  REF(d) AS refr, d.votos_partido(SELF.sigla) / d.total_votos() AS ratio
       FROM distrito d ORDER BY ratio DESC FETCH FIRST ROW ONLY)tmp;

       RETURN ret_variable;
   END best_ratio_district;
END;
```

5 Perguntas

5.1 Pergunta a

Calcule o número total de deputados que cada partido colocou no Parlamento.

5.1.1 Interrogação SQL

```
SELECT p.sigla AS "Partido", p.total_mandatos() AS "Mandatos"
FROM partido p ORDER BY p.sigla;
```

5.1.2 Resultado da interrogação

	Partido	Mandatos
1	BE	2
2	CDSPP	15
3	MPT	0
4	PCPPEV	17
5	PCTPMRPP	0
6	PDA	0
7	PH	0
8	POUS	0
9	PPDPSD	80
10	PPM	0
11	PS	112
12	PSN	0

Figura 2: Resultado da alíena a)

5.2 Pergunta b

Em cada distrito, quantos votos teve cada partido?

5.2.1 Interrogação SQL

```
SELECT d.nome AS "Distrito", p.sigla AS "Partido", d.votos_partido(p.sigla) AS "Votos"
FROM distrito d, partido p ORDER BY d.nome, p.sigla;
```

5.2.2 Resultado da interrogação

	Distrito	Partido	Votos
1	Aveiro	BE	4676
2	Aveiro	CDSPP	49183
3	Aveiro	MPT	847
4	Aveiro	PCPPEV	12797
5	Aveiro	PCTPMRPP	1511
6	Aveiro	PDA	0
7	Aveiro	PH	968
8	Aveiro	POUS	0
9	Aveiro	PPDPSD	138686
10	Aveiro	PPM	1148
11	Aveiro	PS	145575
12	Aveiro	PSN	660
13	Açores	BE	992
14	Açores	CDSPP	5215
15	Açores	MPT	178
16	Açores	PCPPEV	1612
17	Açores	PCTPMRPP	330
18	Açores	PDA	437
19	Açores	PH	0
20	Açores	POUS	0

Figura 3: Parte do resultado da alíena b) (no total, o resultado contém 240 linhas)

5.3 Pergunta c

Determine o partido vencedor em cada concelho.

5.3.1 Interrogação SQL

```
SELECT c.nome AS "Concelho", c.partido_vencedor().sigla AS "Partido Vencedor"
FROM concelho c ORDER BY c.nome;
```

5.3.2 Resultado da interrogação

	Concelho	Partido Vencedor
1	Abrantes	PS
2	Aguiar da Beira	PPDPSD
3	Alandroal	PS
4	Albergaria-a-Velha	PPDPSD
5	Albufeira	PS
6	Alcanena	PS
7	Alcobaça	PPDPSD
8	Alcochete	PS
9	Alcoutim	PS
10	Alcácer do Sal	PS
11	Alenquer	PS
12	Alfandega da Fé	PPDPSD
13	Alijó	PS
14	Aljezur	PS
15	Aljustrel	PS
16	Almada	PS
17	Almeida	PPDPSD
18	Almeirim	PS
19	Almodovar	PS
20	Alpiarça	PS

Figura 4: Parte do resultado da alíena c) (no total, o resultado contém 308 linhas)

5.4 Pergunta d

Verifique se em algum distrito se viola a seguinte regra de integridade: o somatório dos números de votos das várias listas, votos brancos e nulos, mais o número de abstenções tem que ser igual ao número de eleitores inscritos para essa eleição.

5.4.1 Interrogação SQL

```
SELECT d.nome AS "Distrito", d.participacao.votantes AS "Votantes", d.participacao.abstencoes
 ↪  AS "Abstencoes", d.participacao.inscritos AS "Inscritos",
d.participacao.brancos AS "Brancos", d.participacao.nulos AS "Nulos",
d.participacao.votantes - d.participacao.brancos - d.participacao.nulos AS "Votantes - Brancos
 ↪  - Nulos", d.total_votos() AS "Total Votos"
FROM distrito d WHERE d.integrity() = 0;
```

5.4.2 Resultado da interrogação

	Distrito	Votantes	Abstencoes	Inscritos	Brancos	Nulos	Votantes - Brancos - Nulos	Total Votos
1	Açores	93763	92815	186578	687	802	92274	92275

Figura 5: Resultado da alíena d)

5.5 Pergunta e

Quais as diferenças entre as percentagens de mandatos e de votos obtidos por cada partido, a nível nacional?

5.5.1 Interrogação SQL

```
SELECT perc.sigla AS "Partido", perc.votos AS "Perc. Votos", perc.mandatos AS "Perc. Mandatos",
 ↪  perc.votos - perc.mandatos AS "Votos - Mandatos"
FROM
    (SELECT p.sigla AS sigla, ROUND(100*(p.total_votos() / tv.total_votos), 1) AS votos,
     ↪  ROUND(100*(p.total_mandatos() / tm.total_mandatos), 1) AS mandatos
    FROM partido p,
        (SELECT SUM(d.participacao.votantes) AS total_votos
        FROM distrito d) tv,
        (SELECT SUM(l.mandatos) AS total_mandatos FROM lista l) tm
        ) perc
ORDER BY perc.sigla;
```

5.5.2 Resultado da interrogação

	Partido	Perc. Votos	Perc. Mandatos	Votos - Mandatos
1	BE	2.5	0.9	1.6
2	CDSPP	8.4	6.6	1.8
3	MPT	0.4	0	0.4
4	PCPPEV	9	7.5	1.5
5	PCTPMRPP	0.7	0	0.7
6	PDA	0	0	0
7	PH	0.1	0	0.1
8	POUS	0.1	0	0.1
9	PPDPSD	32.3	35.4	-3.1
10	PPM	0.3	0	0.3
11	PS	44	49.6	-5.6
12	PSN	0.2	0	0.2

Figura 6: Resultado da alíena e)

5.6 Pergunta f

Quais os partidos que colocaram deputados por todos os distritos?

5.6.1 Interrogação SQL

```
SELECT p.sigla AS "Partido"
FROM partido p WHERE NOT EXISTS
    (SELECT *
    FROM distrito d
    WHERE d.codigo
        NOT IN
        (SELECT value(l).distrito.codigo FROM table(p.listas) l)
        OR d.codigo IN (SELECT value(l).distrito.codigo
        FROM table(p.listas) l WHERE value(l).mandatos = 0)
    );
```

5.6.2 Resultado da interrogação

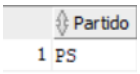

Figura 7: Resultado da alíena f)

As próximas perguntas foram definidas pelo grupo e procuram explorar as potencialidades das extensões OR.

5.7 Pergunta g1

Qual o partido vencedor por distrito?

5.7.1 Interrogação SQL

```
SELECT d.nome, d.partido_vencedor().sigla AS "Partido Vencedor"
FROM distrito d ORDER BY d.nome;
```

5.7.2 Resultado da interrogação

	NOME	Partido Vencedor
1	Aveiro	PS
2	Açores	PS
3	Beja	PS
4	Braga	PS
5	Bragança	PPDPSD
6	Castelo Branco	PS
7	Coimbra	PS
8	Faro	PS
9	Guarda	PS
10	Leiria	PPDPSD
11	Lisboa	PS
12	Madeira	PPDPSD
13	Portalegre	PS
14	Porto	PS
15	Santarém	PS
16	Setúbal	PS
17	Viana do Castelo	PS
18	Vila Real	PPDPSD
19	Viseu	PPDPSD
20	Évora	PS

Figura 8: Resultado da alíena g1)

5.7.3 Aplicação das extensões OR

Devido à aplicação das extensões OR, foi possível reutilizar a função **partido_vencedor()**, definida para o tipo **zona_t** e utilizada na alínea c) para encontrar o vencedor por concelho. A disponibilidade da herança de objetos nas extensões OR proporciona esta reutilização.

Isto indica uma das maiores vantagens das extensões OR. O facto de permitir ao programador poupar trabalho, ao reutilizar as mesmas funções para tipos diferentes derivados da mesma classe base.

5.8 Pergunta g2

5.8.1 Pergunta g21

Quais as freguesias onde houve maioria absoluta por votos?

Interrogação SQL

```
SELECT freg.nome AS "Freguesia", freg.partido_vencedor AS "Partido Vencedor",
 ↪  ROUND(100*freg.ratio, 1) AS "Perc. Votos Partido Vencedor"
FROM
(SELECT f.nome AS nome, f.partido_vencedor().sigla AS partido_vencedor,
 ↪  f.ratio_votos_partido_vencedor() AS ratio FROM freguesia f) freg
WHERE freg.ratio > 0.5
ORDER BY freg.nome;
```

Resultado da interrogação

	Freguesia	Partido Vencedor	Perc. Votos Partido Vencedor
1	A S Gonçalo	PS	54.3
2	A S João	PS	50.1
3	A S Vicente	PS	50.9
4	A Sto Estevão	PS	50.3
5	A dos Negros	PS	51
6	AS Santiago	PS	51.5
7	AS Sta Maria Castelo	PS	50.8
8	Abade de Neiva	PPDPSD	50.4
9	Abadim	PS	57.3
10	Abambres	PPDPSD	67.8
11	Abitureiras	PS	57.4
12	Abiul	PPDPSD	70.9
13	Aboadela	PPDPSD	57.5
14	Aboboreira	PS	52.3
15	Aborim	PS	52
16	Abragão	PPDPSD	52.3
17	Abreiro	PPDPSD	67.1
18	Abrunheira	PS	62.2
19	Achada	PS	56
20	Achete	PS	60.5

Figura 9: Parte do resultado da alíena g21) (no total, o resultado contém 2379 linhas)

5.8.2 Pergunta g22

Quais os concelhos onde houve maioria absoluta por votos?

Interrogação SQL

```sql
SELECT concl.nome AS "Concelho", concl.partido_vencedor AS "Partido Vencedor",
 ↪  ROUND(100*concl.ratio, 1) AS "Perc. Votos Partido Vencedor"
FROM
(SELECT c.nome AS nome, c.partido_vencedor().sigla AS partido_vencedor,
 ↪  c.ratio_votos_partido_vencedor() AS ratio FROM concelho c) concl
WHERE concl.ratio > 0.5
ORDER BY concl.nome;
```

Resultado da interrogação

	Concelho	Partido Vencedor	Perc. Votos Partido Vencedor
1	Abrantes	PS	54.5
2	Aguiar da Beira	PPDPSD	50.6
3	Alcoutim	PS	50.8
4	Alenquer	PS	51.5
5	Aljezur	PS	53.7
6	Almeirim	PS	58.7
7	Almodovar	PS	55.8
8	Alvaiázere	PPDPSD	69.4
9	Amarante	PS	53.6
10	Angra do Heroismo	PS	55.4
11	Ansião	PPDPSD	57.1
12	Arcos de Valdevez	PPDPSD	51
13	Armamar	PPDPSD	57.4
14	Arronches	PS	55.7
15	Arruda dos Vinhos	PS	51
16	Azambuja	PS	52.2
17	Baião	PS	54.3
18	Barrancos	PS	53.4
19	Belmonte	PS	58.7
20	Borba	PS	58.2

Figura 10: Parte do resultado da alíena g22) (no total, o resultado contém 124 linhas)

5.8.3 Pergunta g23

Quais os distritos onde houve maioria absoluta por votos?

Interrogação SQL

```
SELECT dist.nome AS "Distrito", dist.partido_vencedor AS "Partido Vencedor",
 ↪  ROUND(100*dist.ratio, 1) AS "Perc. Votos Partido Vencedor"
FROM
(SELECT d.nome AS nome, d.partido_vencedor().sigla AS partido_vencedor,
 ↪  d.ratio_votos_partido_vencedor() AS ratio FROM distrito d) dist
WHERE dist.ratio > 0.5
ORDER BY dist.nome;
```

Resultado da interrogação

	Distrito	Partido Vencedor	Perc. Votos Partido Vencedor
1	Açores	PS	54.1
2	Castelo Branco	PS	52.8
3	Portalegre	PS	52.3

Figura 11: Resultado da alíena g23)

5.8.4 Aplicação das extensões OR

Em todas as 3 queries acima foi utilizada a função **ratio_votos_partido_vencedor()**, implementada para o tipo **zona_t**. Esta função, por sua vez, chama outra, **total_votos()** que foi declarada como **not instantiable** para o tipo **zona_t** e apenas implementada nas classes derivadas. Conseguimos, assim, com as 3 queries acima, aplicar uma vantagem das extensões OR, o polimorfismo.

5.9 Pergunta g3

Quais os distritos onde houve maioria absoluta por mandatos?

5.9.1 Interrogação SQL

```
SELECT dist.nome AS "Distrito", dist.partido_vencedor AS "Partido Vencedor",
 ↪  ROUND(100*dist.ratio, 1) AS "Perc. Mandatos Partido Vencedor"
FROM
(SELECT d.nome AS nome, d.partido_vencedor().sigla AS partido_vencedor,
 ↪  d.ratio_mandatos_partido_vencedor() AS ratio FROM distrito d) dist
WHERE dist.ratio > 0.5
ORDER BY dist.nome;
```

5.9.2 Resultado da interrogação

	Distrito	Partido Vencedor	Perc. Mandatos Partido Vencedor
1	Açores	PS	60
2	Beja	PS	66.7
3	Castelo Branco	PS	60
4	Coimbra	PS	60
5	Faro	PS	62.5
6	Madeira	PPDPSD	60
7	Portalegre	PS	66.7
8	Porto	PS	51.4
9	Vila Real	PPDPSD	60

Figura 12: Resultado da alíena g3)

5.9.3 Aplicação das extensões OR

Nesta query, utilizou-se as funções **partido_vencedor()** e **ratio_mandatos_partido_vencedor()** para simplificar consideravelmente a escrita da query. Ela ilustra, portanto, uma das vantagens das extensões OR, a simplificação da escrita de queries.

5.10 Pergunta g4

Qual o distrito onde cada partido obteve a maior percentagem de votos?

5.10.1 Interrogação SQL

```
SELECT tmp.sigla AS "Partido", tmp.ret.dist.nome AS "Melhor distrito", ROUND(100*tmp.ret.ratio,
 ↪  1) AS "Perc. Votos"
FROM
(SELECT p.sigla AS sigla, p.best_ratio_district() AS ret FROM  partido p) tmp
ORDER BY tmp.sigla;
```

5.10.2 Resultado da interrogação

	Partido	Melhor distrito	Perc. Votos
1	BE	Lisboa	5
2	CDSPP	Viana do Castelo	14.3
3	MPT	Faro	0.7
4	PCPPEV	Beja	28.9
5	PCTPMRPP	Beja	2
6	PDA	Açores	0.5
7	PH	Viseu	0.3
8	POUS	Leiria	0.2
9	PPDPSD	Madeira	47.5
10	PPM	Vila Real	0.5
11	PS	Açores	54.1
12	PSN	Madeira	0.5

Figura 13: Resultado da alíena g4)

5.10.3 Aplicação das extensões OR

Nesta query utilizou-se a função **best_ratio_district()**, implementada no tipo **partido_t**. Esta função retorna um tipo especialmente criado **best_ratio_ret_t**, com vários membros, permitindo assim que ela seja chamada uma única vez para cada partido, o que diminui a complexidade temporal da query.

Foi, assim, possível ilustrar uma vantagem das extensões OR, a diminuição da complexidade temporal.

5.11 Pergunta g5

Qual a percentagem de votos de cada partido? Apresente os partidos por ordem descendente do número de votos obtidos.

5.11.1 Interrogação SQL

```
SELECT p.sigla AS "Partido", ROUND(100*p.total_votos()/tv.total_votos, 1) AS "Perc. Votos"
FROM
partido p,
(SELECT SUM(d.participacao.votantes) AS total_votos FROM distrito d) tv
ORDER BY value(p) DESC;
```

5.11.2 Resultado da interrogação

	Partido	Perc. Votos
1	PS	44
2	PPDPSD	32.3
3	PCPPEV	9
4	CDSPP	8.4
5	BE	2.5
6	PCTPMRPP	0.7
7	MPT	0.4
8	PPM	0.3
9	PSN	0.2
10	PH	0.1
11	POUS	0.1
12	PDA	0

Figura 14: Resultado da alíena g5)

5.11.3 Aplicação das extensões OR

Nesta query definiu-se a função **total_votos()** como **MAP** para o tipo **partido_t**, o que permitiu obter os partidos ordenados pelo seu número de votos. Foi, assim, possível ilustrar a aplicação das funções de ordenação nas extensões OR.

6 Conclusões

As extensões Objeto-Relacional revelam-se extremamente úteis na modelação de dados devido à proximidade de um paradigma mais próximo do pensamento (Object Oriented), como representação de estruturas hierárquicas (*herança*), mantendo as principais funcionalidades de um modelo relacional. A definição de tipos complexos e funções utilitárias associadas (*encapsulamento*, em vez de guardar apenas informação, como em bases de dados puramente relacionais) permitem facilitar interrogações existentes, abre a possibilidade a novas interrogações mais complexas e frequentemente evita a necessidade de realizar junções (sempre à custa de alguma redundância, o que deve ter sido em conta), uma operação, embora forte, sempre custosa. Por exemplo, relações parte-todo (*agregação*), que em bases de dados relacionais frequentemente são implementadas recorrendo a mecanismos de chaves estrangeiras com restrições adicionais, num modelo objeto-relacional podem simplesmente ser representadas como uma propriedade extra no objeto correspondente ao todo.

Em suma, é uma extensão com bastante potencial que, se corretamente utilizada, facilita o processo de desenvolvimento e abre novas possibilidades para funcionalidades anteriormente muito complicadas de obter.