Modelagem Relacional de Objetos

Tecnologias de Bases de Dados

Catarina Felgueiras
up202403074

José Francisco Veiga
up202108753

Mestrado em Engenharia Informática e Computação
Faculdade de Engenharia da Universidade do Porto

17 de junho de 2025

Resumo
O presente trabalho teve como objetivo principal o desenvolvimento de um modelo objeto-relacional para representar e analisar dados de natureza administrativa e financeira associados a municípios portugueses. Partindo da estrutura de uma base de dados relacional, foi concebido um modelo em conformidade com as extensões do padrão SQL3, permitindo a definição de tipos abstratos, herança, referências e coleções aninhadas (nested tables). Foram criados e populados tipos de objetos que representam entidades como partidos políticos, rubricas orçamentais, entidades geográficas e municípios, incluindo os seus orçamentos e lideranças. Posteriormente, foram definidos métodos úteis no tipo Municipality_T que integram lógica de negócio recorrente. Por fim, foram implementadas várias queries para explorar o modelo, com destaque para o uso das extensões objeto-relacionais e para a análise de despesa pública em diferentes dimensões.

Palavras-chave: Modelo objeto-relacional; SQL3; tipos de objetos; nested tables; referências; métodos; consultas analíticas.

Índice

1 Introdução

Este trabalho tem como principal objetivo a conceção e implementação de um modelo de bases de dados no paradigma object-relational, explorando as potencialidades oferecidas pelas extensões SQL3 face ao modelo relacional clássico. O contexto abordado é a gestão e análise dos orçamentos municipais ao longo de múltiplos anos, contemplando o armazenamento estruturado de dados relativos às receitas e despesas anuais classificadas por categorias específicas (headings), organizadas hierarquicamente através de relações do tipo parent-child.

Cada município é integrado numa hierarquia geográfica, que inclui níveis de agregação como NUT III, NUT II, NUT I e país, sendo registadas informações relevantes como a área geográfica e a população residente em cada nível. Adicionalmente, é considerado o registo histórico relativo à governação política dos municípios, nomeadamente os partidos ou coligações responsáveis pela gestão em diferentes períodos temporais.

Para o desenvolvimento do modelo, foram utilizados tipos de dados abstratos (user-defined types), nested tables, referências tipadas (REFs) e métodos (member functions), facilitando uma representação mais natural e intuitiva dos dados complexos e das estruturas hierárquicas. Este relatório apresenta detalhadamente o modelo conceptual concebido, a sua implementação prática, os métodos desenvolvidos para apoio às consultas mais frequentes e diversos exemplos de queries ilustrativas, destacando assim as vantagens práticas do paradigma object-relational face ao modelo relacional tradicional.

2 Questão 1

Nesta questão foi pedido que se elaborasse um modelo de dados segundo o paradigma object-relational, aproveitando as extensões SQL3 relativamente ao modelo relacional tradicional. Especificamente, pretendeu-se que fossem definidos tipos de dados abstratos (user-defined types), que permitem associar estruturas complexas e métodos encapsulados diretamente às entidades representadas, bem como nested tables, referências tipadas (REFs), e quando apropriado, mecanismos de herança entre tipos. O modelo resultante deveria refletir de forma clara e intuitiva as relações hierárquicas e complexas presentes no contexto dos orçamentos municipais, integrando posteriormente estas definições num esquema prático e implementável numa base de dados object-relational.

A construção deste modelo object-relational teve como principal objetivo tirar partido das extensões do padrão SQL3, integrando conceitos do paradigma orientado a objetos na modelação de dados relacionais. Assim, optou-se por utilizar tipos de objecto (OBJECT TYPES) para representar as principais entidades do domínio, como municípios, partidos políticos, categorias orçamentais (headings) e registos de despesa/receita. Para captar a hierarquia geográfica (país, NUTS I/II/III, município), foi utilizado um tipo abstracto GeoEntity_T, que permite o uso de referências (REF) para modelar relações pai–filho entre entidades, promovendo a reutilização e a generalização.

2.1 Código em SQL

2.1.1 Criação de Tipos

Os seguintes excertos do código apresentam os tipos de dados definidos no âmbito do modelo object-relational, recorrendo às extensões SQL3. O objetivo principal foi representar eficazmente os orçamentos municipais, as suas categorias (headings), períodos temporais, e relações hierárquicas e políticas.

Period_T

```
CREATE TYPE Period_T AS OBJECT (
  year    NUMBER(4),
  quarter VARCHAR2(10) -- '1', '2', '3', '4', 'ANUAL'
);
```

Listing 1: Criação do tipo Period_T em SQL3 para o modelo object-relational

Este tipo é usado para representar períodos específicos no tempo. Cada período inclui um ano (year) e um trimestre (quarter). O trimestre está representado por um único caracter (de '1' a '4').

Party_T

```
CREATE TYPE Party_T AS OBJECT (
  acronym VARCHAR2(10),
  partyName VARCHAR2(100),
  spectrum VARCHAR2(50)
);
```

Listing 2: Criação do tipo Party_T em SQL3 para o modelo object-relational

Este tipo descreve partidos políticos. Inclui um acrónimo (acronym), o nome completo do partido (partyName) e a sua posição no espectro político (spectrum).

Heading_T e Heading_RefList_T

```
CREATE TYPE Heading_T;

CREATE TYPE Heading_RefList_T AS TABLE OF REF Heading_T;

CREATE TYPE Heading_T AS OBJECT (
  id          VARCHAR2(15),
  description VARCHAR2(120),
  type        CHAR(1),        -- 'D' (expense) or 'R' (revenue)
  hlevel      NUMBER(2),
  parent      REF Heading_T,
  children    Heading_RefList_T
);
```

Listing 3: Criação dos tipos Heading_T e Heading_RefList_T em SQL3 para o modelo object-relational

O tipo Heading_T representa categorias orçamentais (headings) para classificar receitas ('R') ou despesas ('D'). Está organizado hierarquicamente, através da referência parent, que aponta para outra categoria superior, e uma lista aninhada de referências (children) que apontam para as categorias filhas, usando uma coleção (Heading_RefList_T).

BudgetEntry_T e BudgetList_T

```sql
CREATE TYPE BudgetEntry_T AS OBJECT (
  entryType CHAR(1), -- 'D' (expense) or 'R' (revenue)
  period    Period_T,
  amount    NUMBER(14,2),
  heading   REF Heading_T
) NOT FINAL;

CREATE TYPE BudgetList_T AS TABLE OF BudgetEntry_T;
```

Listing 4: Criação dos tipos BudgetEntry_T e BudgetList_T em SQL3 para o modelo object-relational

O tipo BudgetEntry_T representa uma entrada individual de orçamento (pode ser despesa ou receita). Cada entrada inclui o tipo de entrada (entryType), o período temporal (period), o montante (amount), e uma referência (heading) para a categoria associada. A coleção aninhada BudgetList_T é usada para guardar múltiplas entradas, permitindo que cada município tenha várias entradas orçamentais associadas.

Leadership_T e LeadershipList_T

```sql
CREATE TYPE Leadership_T AS OBJECT (
  period  Period_T,
  party   REF Party_T
);

CREATE TYPE LeadershipList_T AS TABLE OF Leadership_T;
```

Listing 5: Criação dos tipos Leadership_T e LeadershipList_T em SQL3 para o modelo object-relational

O tipo Leadership_T guarda informação sobre a liderança política em cada município num dado período. Cada registo indica o período temporal (period) e o partido governante (party). LeadershipList_T é uma coleção aninhada que permite associar vários registos de liderança a cada município.

GeoEntity_T

```sql
CREATE TYPE GeoEntity_T AS OBJECT (
  code        VARCHAR2(10),
  name        VARCHAR2(100),
  area        NUMBER,
  population  NUMBER,
  geoLevel    VARCHAR2(20), -- 'country', 'NUTS I', 'NUTS II', 'NUTS III
      ', 'municipality'
  parent      REF GeoEntity_T
) NOT FINAL;
```

Listing 6: Criação do tipo GeoEntity_T em SQL3 para o modelo object-relational

O tipo GeoEntity_T é uma entidade abstrata que representa níveis geográficos diferentes (por exemplo país, NUTS I/II/III e municípios). Inclui informação comum como código (code), nome (name), área (area), população (population) e nível geográfico (geoLevel). A referência parent permite estabelecer uma hierarquia entre as entidades geográficas.

Municipality_T

```
CREATE TYPE Municipality_T AS OBJECT (
  code          VARCHAR2(10),
  name          VARCHAR2(100),
  area          NUMBER,
  population    NUMBER,
  geoLevel      VARCHAR2(20),
  parent        REF GeoEntity_T,
  budgets       BudgetList_T,
  leaderships   LeadershipList_T,
);
```

Listing 7: Criação do tipo Municipality_T em SQL3 para o modelo object-relational

O tipo Municipality_T representa especificamente os municípios. Incorpora todos os atributos semelhantes aos da entidade geográfica geral (GeoEntity_T) mas inclui ainda coleções aninhadas para guardar os seus orçamentos (budgets) e os registos históricos de liderança política (leaderships). A referência parent estabelece a ligação hierárquica com o nível geográfico imediatamente superior.

2.1.2 Criação das Tabelas

Depois de definidos os tipos de dados (OBJECT e coleções), foi necessário criar as tabelas físicas correspondentes no sistema de gestão de base de dados. Estas tabelas armazenam as instâncias dos objetos definidos anteriormente. As instruções a seguir criam essas tabelas, incluindo chaves primárias, definições para referências (REFs) e nested tables.

Tabela Parties

```
CREATE TABLE Parties OF Party_T (
  PRIMARY KEY (acronym)
);
```

Listing 8: Criação da tabela Parties em SQL3

Esta instrução cria uma tabela Parties que irá armazenar objetos do tipo Party_T. Cada partido é identificado unicamente pelo seu acrónimo (acronym), definido aqui como chave primária.

Tabela Headings com Nested Table

```
CREATE TABLE Headings OF Heading_T (
  PRIMARY KEY (id)
)
NESTED TABLE children STORE AS Heading_Children_NT;
```

Listing 9: Criação da tabela Headings em SQL3

Aqui é criada a tabela Headings, que armazena objetos do tipo Heading_T. O identificador de cada heading é o campo id, usado como chave primária. A coleção children, que representa subcategorias (headings filhos), é uma nested table e é armazenada fisicamente com o nome Heading_Children_NT.

```sql
ALTER TABLE Headings
  ADD CONSTRAINT chk_heading_type
  CHECK (type IN ('D','R'));
```

Listing 10: Comando em SQL3

Este comando adiciona uma restrição de domínio para garantir que o campo type apenas aceita os valores 'D' (despesa) ou 'R' (receita), assegurando integridade semântica dos dados.

Tabela GeoEntities com restrição de referência

```sql
CREATE TABLE GeoEntities OF GeoEntity_T (
  PRIMARY KEY (code)
);
```

Listing 11: Criação da tabela GeoEntities em SQL3

Cria-se a tabela GeoEntities, que guarda todas as entidades geográficas, como municípios, NUTS e país, com base no tipo GeoEntity_T. Cada entidade é identificada pelo seu código (code).

```sql
ALTER TABLE GeoEntities
  ADD SCOPE FOR (parent) IS GeoEntities;
```

Listing 12: Comando em SQL3

Este comando especifica que o campo parent, que é um REF GeoEntity_T, só pode referenciar linhas da tabela GeoEntities. Isto ajuda a manter a integridade referencial entre entidades geográficas.

```sql
ALTER TABLE GeoEntities
  ADD CONSTRAINT chk_geo_level
  CHECK (geoLevel IN ('country','NUTS␣I','NUTS␣II','NUTS␣III','
      municipality'));
```

Listing 13: Comando em SQL3

Adiciona-se aqui uma restrição de validação ao campo geoLevel, garantindo que apenas níveis geográficos válidos podem ser atribuídos às entidades.

Tabela Municipalities com nested tables e restrição de referência

```sql
CREATE TABLE Municipalities OF Municipality_T (
  PRIMARY KEY (code)
)
NESTED TABLE budgets      STORE AS Mun_Budgets_NT,
NESTED TABLE leaderships STORE AS Mun_Leaderships_NT;
```

Listing 14: Criação da tabela Municipalities em SQL3

Esta tabela armazena objetos do tipo Municipality_T, cada um identificado por um código único (code). São definidas duas coleções aninhadas:

- budgets, onde se guardam todas as entradas orçamentais de cada município, armazenadas fisicamente na nested table Mun_Budgets_NT.

- leaderships, onde se guarda o histórico de partidos no poder em cada período, com armazenamento na nested table Mun_Leaderships_NT.

```
ALTER TABLE Municipalities
  ADD SCOPE FOR (parent) IS GeoEntities;
```

Listing 15: Comando em SQL3

Define-se o escopo do campo parent (uma referência do tipo REF GeoEntity_T), indicando que só pode referenciar entradas na tabela GeoEntities.

```
ALTER TABLE Municipalities
  ADD CONSTRAINT chk_municipality_level
  CHECK (geoLevel = 'municipality');
```

Listing 16: Comando em SQL3

Por fim, é adicionada uma restrição de verificação para garantir que todos os registos na tabela Municipalities tenham o campo geoLevel com o valor 'municipality', distinguindo-os de outras entidades geográficas.

2.2 Diagrama UML do Modelo Object-Relational

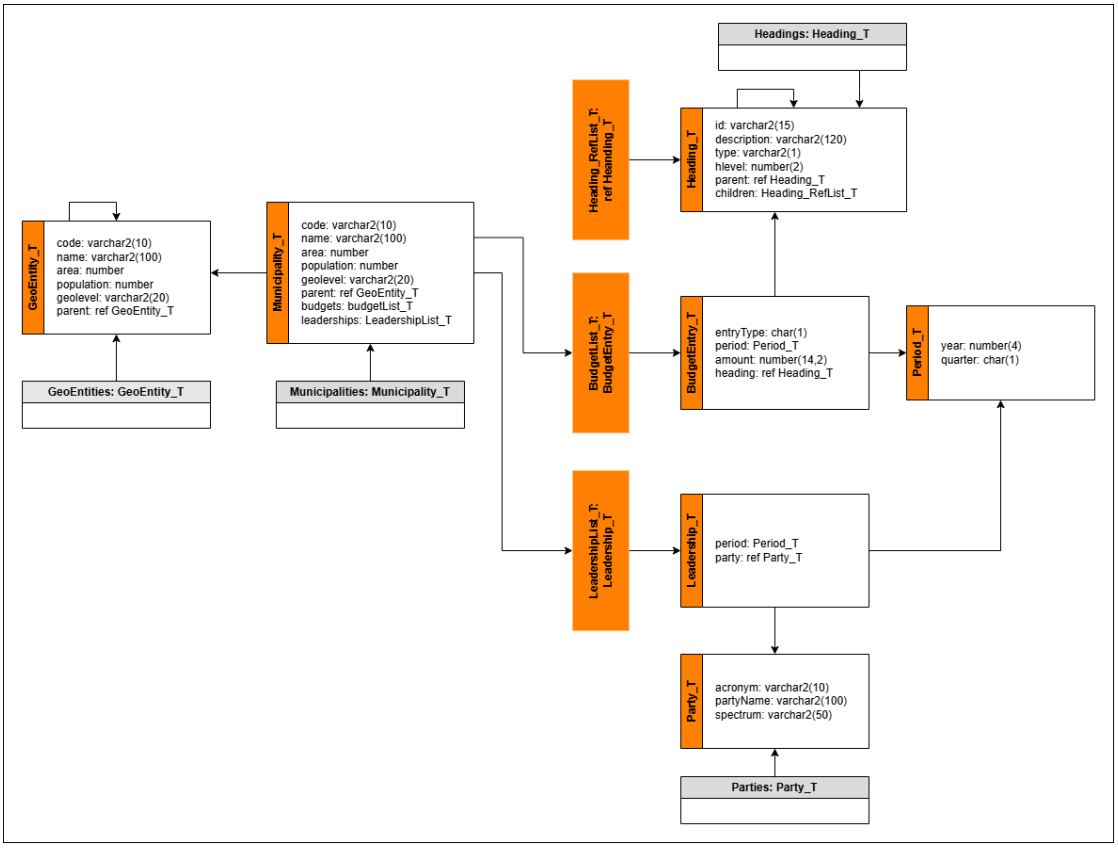


Figura 1: Diagrama UML do Modelo object-relational.

A figura apresenta um diagrama UML do modelo object-relational desenvolvido, onde são representados os tipos de dados abstratos (user-defined types), nested tables, as referências tipadas (REFs) e a correspondência com as tabelas físicas. As caixas a laranja representam as coleções (TABLE OF), as setas indicam dependências e relações entre os tipos, e as caixas cinzentas inferiores correspondem às tabelas OF TYPE instanciadas a partir dos tipos definidos.

3 Questão 2

Para o preenchimento das tabelas do modelo objeto-relacional, optou-se por uma abordagem mista entre dados reais e dados simulados. No caso das entidades geográficas (tabela GeoEntities), foram considerados os diferentes níveis da classificação NUTS (NUTS I, NUTS II e NUTS III), conforme definidos oficialmente para Portugal. Os códigos, designações, áreas e populações destas regiões foram obtidos a partir de fontes oficiais, nomeadamente da página da Wikipedia, com o objetivo de garantir um enquadramento realista e alinhado com os dados estatísticos portugueses.

Para os restantes dados, como os partidos políticos (tabela Parties), os headings orçamentais (tabela Headings) e os municípios (tabela Municipalities), foram utilizados valores simulados e plausíveis para efeitos de teste e demonstração. As rubricas orçamentais foram inspiradas em categorias comuns da contabilidade pública (ex.: salários, investimentos, transferências do Estado), e os valores atribuídos às despesas e receitas foram definidos com coerência interna, mas sem corresponder a dados reais. Os partidos considerados refletem a realidade do sistema político português, e os municípios foram preenchidos com referência a regiões representativas.

Este processo permitiu testar a estrutura e funcionamento do modelo objeto-relacional, bem como executar consultas analíticas sobre os dados, sem a complexidade de integrar um dataset oficial completo. A coerência dos dados foi assegurada, especialmente ao nível das referências (REF) e das coleções aninhadas (NESTED TABLES), de forma a garantir a consistência do modelo e a sua navegabilidade.

O código utilizado nesta questão está presente nos Anexos.

4 Questão 3

Nesta questão, foi solicitado que fossem adicionados métodos ao modelo object-relational para apoiar consultas SQL frequentes. A implementação de métodos diretamente dentro do tipo Municipality_T teve como objetivo principal integrar operações complexas e recorrentes, reduzindo significativamente a complexidade e repetição nas queries efetuadas sobre os municípios. Para isso, antes de se pensar e estruturar os métodos, pensou-se no que seria mais útil para as queries da questão 4.

Os métodos escolhidos procuram responder de forma eficiente às consultas que envolvem agregações financeiras, cálculos per capita ou verificações políticas. Assim, o tipo Municipality_T foi enriquecido com os seguintes métodos (MEMBER FUNCTION):

`total_expenses`

```
MEMBER FUNCTION total_expenses (
  p_period   Period_T,
  p_heading  REF Heading_T DEFAULT NULL
) RETURN NUMBER
```

Listing 17: Função que calcula o total de despesas num determinado período

Este método devolve o total das despesas realizadas pelo município num determinado período. Se for passado um parâmetro de heading, os resultados são filtrados apenas para essa categoria orçamental específica.

`expenses_per_1000_inhabitants`

```
MEMBER FUNCTION expenses_per_1000_inhabitants (
  p_period   Period_T,
  p_heading  REF Heading_T DEFAULT NULL
) RETURN NUMBER
```

Listing 18: Função que calcula a despesa por cada mil habitantes

Este método calcula a despesa por cada mil habitantes do município, num dado período. Reutiliza a função `total_expenses` e divide o resultado pela população do município.

total_investment_per_km2

```
MEMBER FUNCTION total_investment_per_km2 (
  p_period   Period_T
) RETURN NUMBER
```

Listing 19: Função que calcula o investimento por km^2

Esta função calcula o total de despesas de investimento por quilómetro quadrado, filtrando os headings cuja descrição começa por "Inv" e dividindo o total pela área do município.

total_revenues

```
MEMBER FUNCTION total_revenues (
  p_period   Period_T,
  p_heading  REF Heading_T DEFAULT NULL
) RETURN NUMBER
```

Listing 20: Função que calcula o total de receitas

Este método devolve o total das receitas obtidas pelo município num determinado período, opcionalmente filtradas por uma categoria orçamental (heading).

net_balance

```
MEMBER FUNCTION net_balance (
  p_period Period_T
) RETURN NUMBER
```

Listing 21: Função que calcula o saldo orçamental

Este método calcula o saldo orçamental do município num dado período, através da diferença entre receitas e despesas totais.

get_governing_party

```
MEMBER FUNCTION get_governing_party (
  p_period Period_T
) RETURN REF Party_T
```

Listing 22: Função que devolve o partido que governa no período

Devolve uma referência para o partido que liderava o município no período especificado, procurando dentro da coleção de `leaderships`.

is_governed_by

```
MEMBER FUNCTION is_governed_by (
  party REF Party_T,
  p_period Period_T
) RETURN BOOLEAN
```

Listing 23: Função que verifica se o município era governado por um partido específico

Este método devolve `TRUE` se o partido passado como argumento estava no poder no período indicado, e `FALSE` caso contrário.

has_heading

```
MEMBER FUNCTION has_heading (
  p_heading REF Heading_T
) RETURN BOOLEAN
```

Listing 24: Função que verifica se o município possui um heading específico

Verifica se existe pelo menos uma entrada no orçamento do município (budget) associada à categoria orçamental (heading) passada como argumento.

Implementação do TYPE BODY Municipality_T

Para além da definição das assinaturas dos métodos no tipo Municipality_T, foi necessário criar o respetivo TYPE BODY, onde se encontra a lógica de implementação de cada um. Este bloco é essencial para que os métodos funcionem corretamente, permitindo operar sobre os atributos e coleções do objeto.

Cada método foi cuidadosamente desenhado para responder a necessidades de consulta recorrentes.

- total_expenses percorre a nested tables de budgets e acumula os valores que correspondem a despesas do período especificado.
- total_expenses percorre a nested table de budgets e acumula os valores que correspondem a despesas do período especificado.
- expenses_per_1000_inhabitants aproveita o resultado do método anterior e normaliza-o face à população do município.
- total_investment_per_km2 aplica um filtro por categorias que representam investimento (INV%) e divide o total pela área territorial do município.
- net_balance devolve o resultado da subtração entre receitas e despesas no período, funcionando como um indicador financeiro direto.
- get_governing_party e is_governed_by trabalham sobre a coleção leaderships e permitem associar os dados financeiros à força política em exercício.
- has_heading verifica se o município registou entradas orçamentais associadas a uma determinada categoria, útil para filtros avançados.

Ao agrupar toda esta lógica no TYPE BODY, conseguimos obter um modelo mais modular e reutilizável, que torna as futuras consultas mais simples, sem necessidade de escrever blocos repetitivos de código SQL.

```
CREATE OR REPLACE TYPE BODY Municipality_T AS

  MEMBER FUNCTION total_expenses (
    p_period   Period_T,
    p_heading  REF Heading_T DEFAULT NULL
```

```
  ) RETURN NUMBER IS
    total NUMBER := 0;
  BEGIN
    SELECT NVL(SUM(b.amount), 0)
    INTO total
    FROM TABLE(SELF.budgets) b
    WHERE b.entryType = 'D' AND
          b.period.year = p_period.year AND
          (p_heading IS NULL OR b.heading = p_heading);
    RETURN total;
  END;

  MEMBER FUNCTION expenses_per_1000_inhabitants (
    p_period   Period_T,
    p_heading  REF Heading_T DEFAULT NULL
  ) RETURN NUMBER IS
    total NUMBER;
  BEGIN
    total := SELF.total_expenses(p_period, p_heading);
    IF SELF.population > 0 THEN
      RETURN (total / SELF.population) * 1000;
    ELSE
      RETURN NULL;
    END IF;
  END;

  MEMBER FUNCTION total_investment_per_km2 (
    p_period   Period_T
  ) RETURN NUMBER IS
    total NUMBER := 0;
  BEGIN
    SELECT NVL(SUM(b.amount), 0)
    INTO total
    FROM TABLE(SELF.budgets) b
    WHERE b.entryType = 'D' AND
          b.period.year = p_period.year AND
          EXISTS (
            SELECT 1 FROM Headings h
            WHERE h.id LIKE 'D2%' AND b.heading = REF(h)
          );
    IF SELF.area > 0 THEN
      RETURN total / SELF.area;
    ELSE
      RETURN NULL;
    END IF;
  END;

  MEMBER FUNCTION total_revenues (
    p_period   Period_T,
    p_heading  REF Heading_T DEFAULT NULL
  ) RETURN NUMBER IS
    total NUMBER := 0;
  BEGIN
```

```
59      SELECT NVL(SUM(b.amount), 0)
60      INTO total
61      FROM TABLE(SELF.budgets) b
62      WHERE b.entryType = 'R' AND
63            b.period.year = p_period.year AND
64            (p_heading IS NULL OR b.heading = p_heading);
65      RETURN total;
66    END;
67
68    MEMBER FUNCTION net_balance (
69      p_period   Period_T
70    ) RETURN NUMBER IS
71      revenues NUMBER;
72      expenses NUMBER;
73    BEGIN
74      revenues := SELF.total_revenues(p_period);
75      expenses := SELF.total_expenses(p_period);
76      RETURN revenues - expenses;
77    END;
78
79    MEMBER FUNCTION get_governing_party (
80      p_period Period_T
81    ) RETURN REF Party_T IS
82      p REF Party_T;
83    BEGIN
84      FOR i IN 1 .. SELF.leaderships.COUNT LOOP
85        IF SELF.leaderships(i).period.year = p_period.year THEN
86          RETURN SELF.leaderships(i).party;
87        END IF;
88      END LOOP;
89      RETURN NULL;
90    END;
91
92    MEMBER FUNCTION is_governed_by (
93      party REF Party_T,
94      p_period Period_T
95    ) RETURN BOOLEAN IS
96    BEGIN
97      FOR i IN 1 .. SELF.leaderships.COUNT LOOP
98        IF SELF.leaderships(i).period.year = p_period.year AND
99           SELF.leaderships(i).party = party THEN
100         RETURN TRUE;
101       END IF;
102     END LOOP;
103     RETURN FALSE;
104   END;
105
106   MEMBER FUNCTION has_heading (
107     p_heading REF Heading_T
108   ) RETURN BOOLEAN IS
109   BEGIN
110     FOR i IN 1 .. SELF.budgets.COUNT LOOP
111       IF SELF.budgets(i).heading = p_heading THEN
```

```
112         RETURN TRUE;
113       END IF;
114     END LOOP;
115     RETURN FALSE;
116   END;
117
118 END;
```

Listing 25: Implementação completa do TYPE BODY Municipality_T

5 Questão 4

Nesta questão foi solicitado que fossem elaboradas e executadas várias consultas sobre a base de dados construída no modelo object-relational. As queries devem tirar partido das funcionalidades disponibilizadas pelos tipos definidos (OBJECT TYPES), métodos associados e nested tables, de forma a obter diferentes tipos de análises orçamentais e políticas.

De seguida, apresentam-se as consultas desenvolvidas, acompanhadas da explicação e do respetivo resultado.

a) Cálculo de despesas agregadas por período e heading, ordenando os municípios por população.

```
1  SELECT
2      m.name              AS municipio,
3      b.period.year       AS ano,
4      h.description       AS titulo,
5      SUM(b.amount)       AS despesa,
6      m.population        AS habitantes
7  FROM
8      Municipalities m
9      JOIN TABLE(m.budgets) b ON 1 = 1
10     JOIN Headings h         ON b.heading = REF(h)
11 WHERE
12     b.entryType = 'D'
13 GROUP BY
14     m.name,
15     b.period.year,
16     h.description,
17     m.population
18 ORDER BY
19     m.population DESC;
```

Listing 26: Consulta SQL para a alínea a)

Esta query calcula, para cada município, a soma das despesas agrupadas por ano e por heading. Para isso, acede à tabela Municipalities e expande a nested table budgets através de TABLE(m.budgets), que contém os registos orçamentais de cada município. De seguida, junta a tabela Headings usando REF(h) para obter a descrição textual da rubrica associada a cada entrada de orçamento. A cláusula WHERE filtra apenas as entradas de despesa (com entryType = 'D'). A agregação é feita com SUM(b.amount) e o agrupamento por nome do município, ano, rubrica e população, permitindo assim calcular o total de despesa por categoria e período. Por

fim, os municípios são ordenados por número de habitantes em ordem decrescente, permitindo uma análise demográfica da despesa pública. Esta query demonstra o uso de nested tables e referências no modelo object-relational.

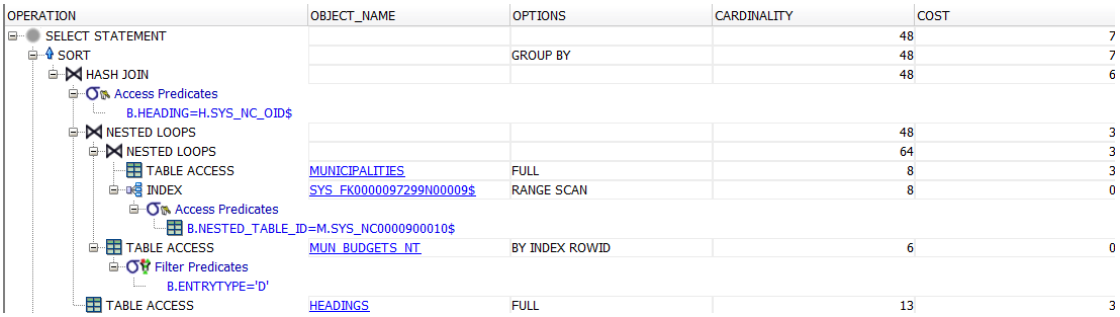
O resultado desta query é o seguinte:

MUNICIPIO	ANO	TITULO	DESPESA	HABITANTES
Lisboa	2022	Investimentos	50000	545923
Lisboa	2022	Salários	80000	545923
Lisboa	2023	Aquisição de Bens e Serviços	20000	545923
Lisboa	2023	Investimentos	105000	545923
Lisboa	2024	Aquisição de Terrenos e Edifícios	75000	545923
Lisboa	2024	Salários	18000	545923
Porto	2022	Investimentos	50000	232125
Porto	2022	Salários	80000	232125
Porto	2023	Aquisição de Bens e Serviços	20000	232125
Porto	2023	Investimentos	105000	232125
Porto	2024	Aquisição de Terrenos e Edifícios	75000	232125

MUNICIPIO	ANO	TITULO	DESPESA	HABITANTES
Porto	2024	Salários	18000	232125
Braga	2022	Investimentos	50000	193333
Braga	2022	Salários	80000	193333
Braga	2023	Aquisição de Bens e Serviços	20000	193333
Braga	2023	Investimentos	105000	193333
Braga	2024	Aquisição de Terrenos e Edifícios	75000	193333
Braga	2024	Salários	18000	193333
Coimbra	2022	Investimentos	50000	143396
Coimbra	2022	Salários	80000	143396
Coimbra	2023	Aquisição de Bens e Serviços	20000	143396
Coimbra	2023	Investimentos	105000	143396

MUNICIPIO	ANO	TITULO	DESPESA	HABITANTES
Coimbra	2024	Aquisição de Terrenos e Edifícios	75000	143396
Coimbra	2024	Salários	18000	143396
Vila Franca de Xira	2022	Investimentos	50000	137659
Vila Franca de Xira	2022	Salários	80000	137659
Vila Franca de Xira	2023	Aquisição de Bens e Serviços	20000	137659
Vila Franca de Xira	2023	Investimentos	105000	137659

Figura 2: Resultado Query a).



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			48	7
SORT		GROUP BY	48	7
HASH JOIN			48	6
Access Predicates				
B.HEADING=H.SYS_NC_OID$				
NESTED LOOPS			48	3
NESTED LOOPS			64	3
TABLE ACCESS	MUNICIPALITIES	FULL	8	3
INDEX	SYS_FK0000097299N00009$	RANGE SCAN	8	0
Access Predicates				
B.NESTED_TABLE_ID=M.SYS_NC0000900010$				
TABLE ACCESS	MUN_BUDGETS_NT	BY INDEX ROWID	6	0
Filter Predicates				
B.ENTRYTYPE='D'				
TABLE ACCESS	HEADINGS	FULL	13	3

Figura 3: Custo Query a).

b) Verificação da consistência entre os valores dos headings de nível superior e a soma dos seus descendentes.

```
1  SELECT
2      m.name                              AS municipio,
3      b.period.year                       AS ano,
4      h.id                                AS heading_pai,
5      h.description                       AS descricao_pai,
6      ROUND(SUM(b.amount), 2)             AS montante_pai,
7      ROUND(SUM(cb.amount), 2)            AS soma_filhos,
8      ROUND(SUM(b.amount) -
9            SUM(cb.amount), 2)            AS dif
10 FROM
11     Municipalities m
12     , TABLE(m.budgets) b
13     , Headings h
14     , Headings c
15     , TABLE(m.budgets) cb
16 WHERE
17         b.heading       = REF(h)
18     AND c.parent        = REF(h)
19     AND cb.heading      = REF(c)
20     AND cb.entryType    = b.entryType
21     AND cb.period.year  = b.period.year
22     AND h.children IS NOT NULL
23 GROUP BY
24     m.name,
25     b.period.year,
26     h.id,
27     h.description
28 HAVING
29     ROUND(SUM(b.amount) - SUM(cb.amount), 2) <> 0
30 ORDER BY
31     municipio,
32     ano,
33     heading_pai;
```

Listing 27: Consulta SQL para a alínea b)

Esta query verifica, para cada município e ano, se os valores orçamentais registados nos headings de nível superior (ou seja, os que têm filhos) são consistentes com a soma dos valores registados nos headings de nível inferior correspondentes. Para isso, começa por aceder à tabela Municipalities e expande duas vezes a nested table budgets: uma vez para obter os valores do heading-pai (b) e outra para obter os valores dos headings-filho (cb). Junta a tabela Headings duas vezes: h representa o heading de nível superior e c os headings de nível inferior, estabelecendo a relação hierárquica através da condição c.parent = REF(h). Em seguida, relaciona os budgets dos filhos (cb) com os headings c, garantindo que são do mesmo tipo de entrada (entryType) e do mesmo ano. A condição h.children IS NOT NULL assegura que apenas são considerados headings com descendência. A cláusula HAVING filtra os casos em que existe diferença entre o valor do heading-pai e a soma dos seus filhos, permitindo identificar inconsistências nos dados. Por fim, os resultados são ordenados por município, ano e identificador do heading. Esta query tira partido da hierarquia de headings e do uso de referências (REF) para percorrer as relações entre objetos no modelo object-relational.

O resultado desta query é o seguinte:

MUNICIPIO	ANO	HEADING_PAI	DESCRICAO_PAI	MONTANTE_PAI	SOMA_FILHOS	DIF
Aveiro	2022	D1	Despesas Correntes	100000	80000	20000
Aveiro	2023	D2	Despesas de Capital	120000	105000	15000
Aveiro	2023	R1	Receitas Correntes	98000	110000	-12000
Aveiro	2024	R2	Receitas de Capital	112000	150000	-38000
Braga	2022	D1	Despesas Correntes	100000	80000	20000
Braga	2023	D2	Despesas de Capital	120000	105000	15000
Braga	2023	R1	Receitas Correntes	98000	110000	-12000
Braga	2024	R2	Receitas de Capital	112000	150000	-38000
Coimbra	2022	D1	Despesas Correntes	100000	80000	20000
Coimbra	2023	D2	Despesas de Capital	120000	105000	15000
Coimbra	2023	R1	Receitas Correntes	98000	110000	-12000

Figura 4: Resultado Query b).

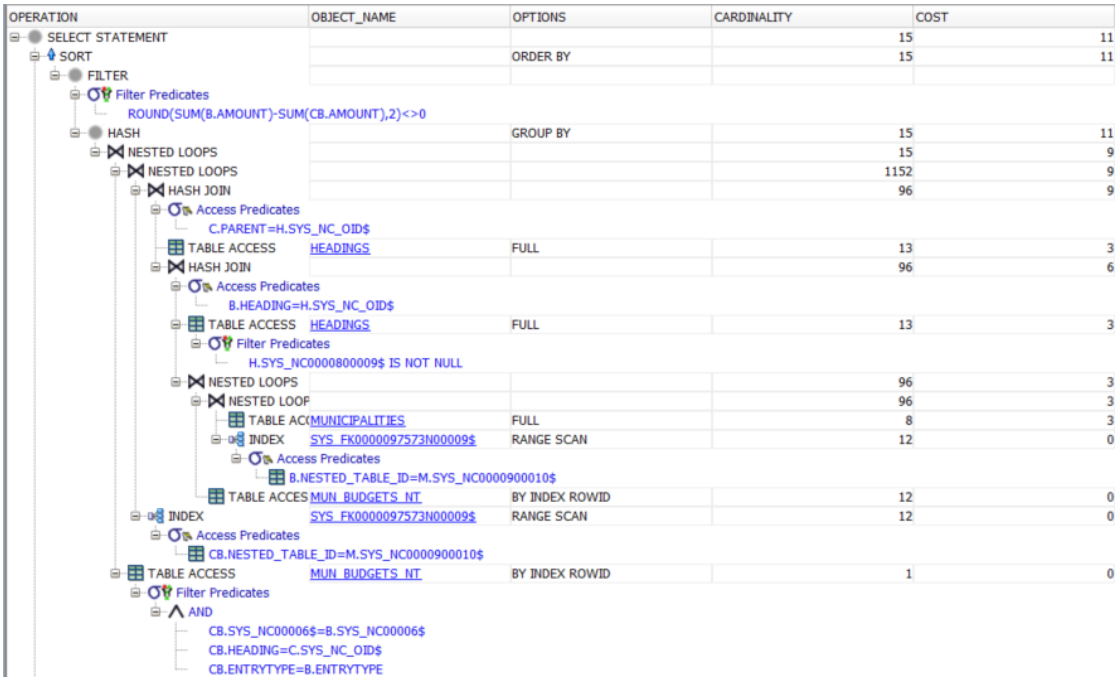


Figura 5: Custo Query b).

c) Análise da despesa média por mil habitantes em cada heading, por partido político.

```sql
SELECT *
FROM (
  SELECT
      p.acronym                              AS partido,
      SUBSTR(h.description, 1, 35)           AS titulo,
      ROUND(
        AVG(
          m.expenses_per_1000_inhabitants(
            l.period,
            REF(h)
          )
        ), 2
      ) AS despesa_media_por_1000_habitantes
  FROM
      Municipalities m
  CROSS JOIN
      Headings h
  JOIN
      TABLE(m.leaderships) l
      ON l.party IS NOT NULL
  JOIN
      Parties p
      ON l.party = REF(p)
  WHERE
      h.type = 'D'
  GROUP BY
      p.acronym,
      h.description
)
WHERE despesa_media_por_1000_habitantes > 0
ORDER BY despesa_media_por_1000_habitantes DESC;
```

Listing 28: Consulta SQL para a alínea c)

Esta query calcula, para cada partido político e para cada rubrica orçamental de despesa (heading), a despesa média por mil habitantes no ano de 2023, considerando os municípios que esse partido lidera nesse ano. A consulta começa por fazer um CROSS JOIN entre os municípios (Municipalities) e os headings (Headings), de forma a percorrer todas as possíveis combinações entre municípios e categorias orçamentais. Em seguida, é feita a expansão da nested table leaderships através de TABLE(m.leaderships), o que permite aceder ao histórico de governação de cada município. Esta tabela é ligada ao ano de 2023 para filtrar apenas a liderança vigente nesse período. Depois, através de REF(p), obtém-se a referência ao partido associado a essa liderança, que é comparada com a tabela Parties para aceder ao acrónimo do partido. A cláusula WHERE garante que apenas são consideradas rubricas do tipo despesa (type = 'D'). A função AVG(...) é aplicada ao método expenses_per_1000_inhabitants(...), definido no tipo Municipality_T, que calcula a despesa nessa rubrica por mil habitantes, normalizando automaticamente em função da população de cada município. A agregação é feita por partido e rubrica, e os resultados são ordenados por partido e depois por rubrica, permitindo comparar o esforço orçamental médio por categoria entre diferentes forças políticas. Esta query demonstra o uso de métodos encapsulados e referências tipadas no modelo object-relational, juntamente com a utilização de nested tables para modelar informação histórica sobre liderança política.

O resultado desta query é o seguinte:

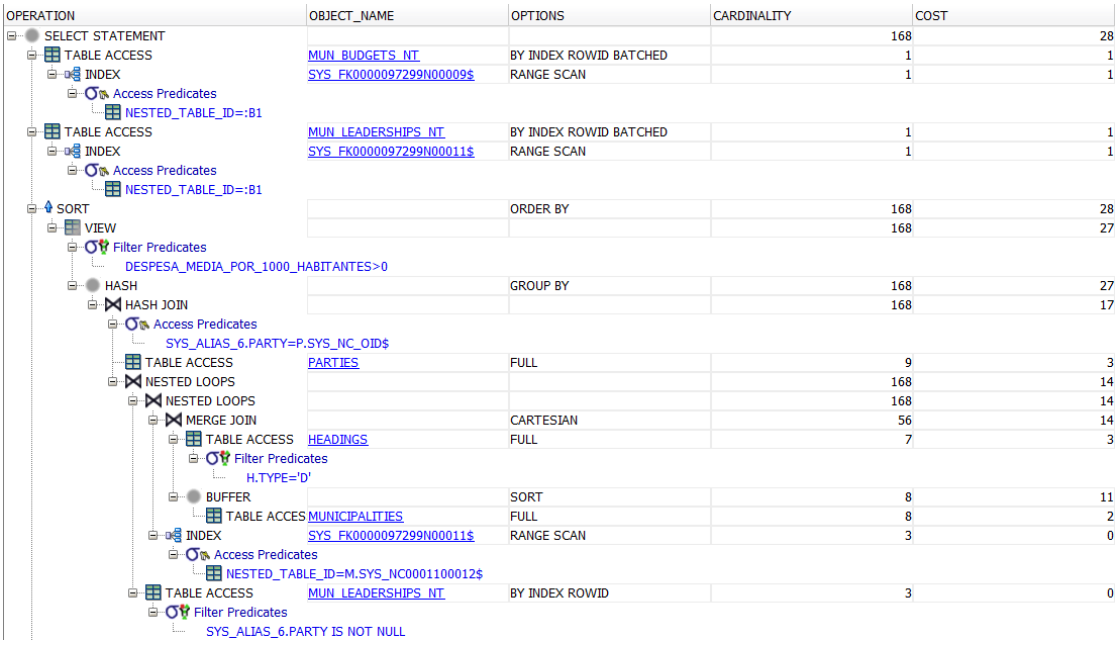
PARTIDO	TITULO	DESPESA_MEDIA_POR_1000_HABITANTES
IL	Investimentos	868,97
L	Investimentos	567
BE	Investimentos	544,15
PAN	Investimentos	508,5
PCP	Investimentos	488,16
IL	Aquisição de Terrenos e Edifícios	310,35
PPD/PSD	Investimentos	301,56
PS	Investimentos	245,15
L	Aquisição de Terrenos e Edifícios	202,5
BE	Aquisição de Terrenos e Edifícios	194,34
PAN	Aquisição de Terrenos e Edifícios	181,61

PARTIDO	TITULO	DESPESA_MEDIA_POR_1000_HABITANTES
PCP	Aquisição de Terrenos e Edifícios	174,34
IL	Aquisição de Bens e Serviços	165,52
L	Aquisição de Bens e Serviços	108
PPD/PSD	Aquisição de Terrenos e Edifícios	107,7
BE	Aquisição de Bens e Serviços	103,65
PAN	Aquisição de Bens e Serviços	96,86
PCP	Aquisição de Bens e Serviços	92,98
PS	Aquisição de Terrenos e Edifícios	87,55
IL	Salários	74,48
PPD/PSD	Aquisição de Bens e Serviços	57,44
L	Salários	48,6

PARTIDO	TITULO	DESPESA_MEDIA_POR_1000_HABITANTES
PS	Aquisição de Bens e Serviços	46,69
BE	Salários	46,64
PAN	Salários	43,59
PCP	Salários	41,84
PPD/PSD	Salários	25,85
PS	Salários	21,01

28 rows selected.

Figura 6: Resultado Query c).



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			168	28
TABLE ACCESS	MUN_BUDGETS_NT	BY INDEX ROWID BATCHED	1	1
INDEX	SYS_FK0000097299N00009$	RANGE SCAN	1	1
Access Predicates				
NESTED_TABLE_ID=:B1				
TABLE ACCESS	MUN_LEADERSHIPS_NT	BY INDEX ROWID BATCHED	1	1
INDEX	SYS_FK0000097299N00011$	RANGE SCAN	1	1
Access Predicates				
NESTED_TABLE_ID=:B1				
SORT		ORDER BY	168	28
VIEW			168	27
Filter Predicates				
DESPESA_MEDIA_POR_1000_HABITANTES>0				
HASH		GROUP BY	168	27
HASH JOIN			168	17
Access Predicates				
SYS_ALIAS_6.PARTY=P.SYS_NC_OID$				
TABLE ACCESS	PARTIES	FULL	9	3
NESTED LOOPS			168	14
NESTED LOOPS			168	14
MERGE JOIN		CARTESIAN	56	14
TABLE ACCESS	HEADINGS	FULL	7	3
Filter Predicates				
H.TYPE='D'				
BUFFER		SORT	8	11
TABLE ACCES	MUNICIPALITIES	FULL	8	2
INDEX	SYS_FK0000097299N00011$	RANGE SCAN	3	0
Access Predicates				
NESTED_TABLE_ID=M.SYS_NC0001100012$				
TABLE ACCESS	MUN_LEADERSHIPS_NT	BY INDEX ROWID	3	0
Filter Predicates				
SYS_ALIAS_6.PARTY IS NOT NULL				

Figura 7: Custo Query c).

d) Qual partido tem mais investimento por km² por ano?

```
1  WITH inv_party AS (
2    SELECT
3        l.period.year                                    AS ano,
4        p.acronym                                        AS partido,
5        SUM(m.total_investment_per_km2(l.period) * m.area) AS inv_total,
             -- euros
6        SUM(m.area)                                      AS area_total
             -- km2
7    FROM
8          Municipalities m
9    JOIN TABLE(m.leaderships) l
10         ON 1 = 1  -- required dummy ON for CROSS JOIN semantics
11   JOIN TABLE(m.budgets) b
12         ON b.period.year = l.period.year
13   JOIN Parties p
14         ON l.party = REF(p)
15   GROUP BY
16         l.period.year,
17         p.acronym
18 )
19 SELECT ano,
20        partido,
21        ROUND(inv_total / area_total, 2) AS investimento_por_km2
22 FROM (
23   SELECT
```

```
24        inv_party.*,
25        RANK() OVER (
26          PARTITION BY ano
27          ORDER BY inv_total / area_total DESC
28        ) AS rnk
29    FROM inv_party
30  )
31  WHERE rnk = 1
32  ORDER BY ano;
```

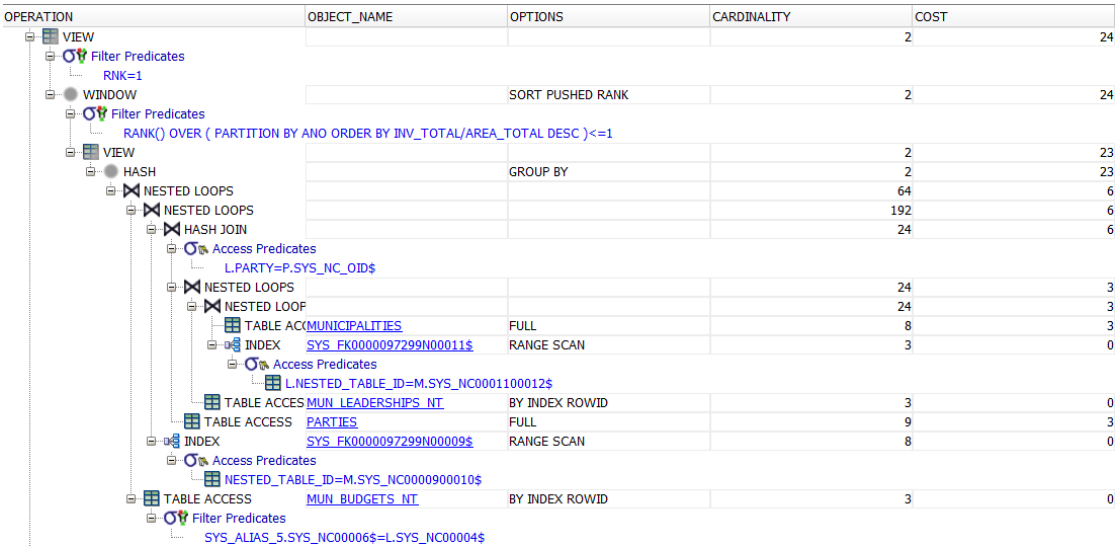
Listing 29: Consulta SQL para a alínea d)

Esta query tem como objetivo determinar, para cada ano, qual o partido político que lidera os municípios com maior investimento por quilómetro quadrado (km²). Para isso, é utilizada uma subquery com 'WITH' denominada 'inv_party', onde se calcula o total de investimento e a área total dos municípios liderados por cada partido, por ano. O cálculo do investimento é feito através da multiplicação do valor devolvido pelo método 'total_investment_per_km2(...)' (que retorna o investimento por km² num dado ano) pela área de cada município, resultando no total investido. Os dados de liderança são obtidos a partir da nested table 'leaderships' usando 'TABLE(m.leaderships)', e são associados a cada partido com 'REF(p)'. Para garantir que os dados de investimento e governação pertencem ao mesmo período, faz-se um 'JOIN' com a nested table 'budgets', filtrando pelo mesmo ano. A subquery 'inv_party' agrega os valores por partido e ano. No passo seguinte, os resultados são ordenados por investimento por km² de forma decrescente, utilizando a função analítica 'RANK()' com 'PARTITION BY ano', permitindo selecionar apenas o partido com melhor desempenho por ano ('WHERE rnk = 1'). A query final devolve então, para cada ano, o partido com maior média de investimento orçamental por quilómetro quadrado, sendo uma demonstração clara da combinação de métodos definidos em tipos, referências e nested tables num modelo object-relational.

O resultado desta query é o seguinte:

ANO	PARTIDO	INVESTIMENTO_POR_KM2
2023	PPD/PSD	2535,01
2024	PPD/PSD	1810,72

Figura 8: Resultado Query d).



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
VIEW			2	24
Filter Predicates				
RNK=1				
WINDOW		SORT PUSHED RANK	2	24
Filter Predicates				
RANK() OVER (PARTITION BY ANO ORDER BY INV_TOTAL/AREA_TOTAL DESC)<=1				
VIEW			2	23
HASH		GROUP BY	2	23
NESTED LOOPS			64	6
NESTED LOOPS			192	6
HASH JOIN			24	6
Access Predicates				
L.PARTY=P.SYS_NC_OID$				
NESTED LOOPS			24	3
NESTED LOOP			24	3
TABLE AC	MUNICIPALITIES	FULL	8	3
INDEX	SYS_FK0000097299N00011$	RANGE SCAN	3	0
Access Predicates				
L.NESTED_TABLE_ID=M.SYS_NC0001100012$				
TABLE ACCES	MUN_LEADERSHIPS_NT	BY INDEX ROWID	3	0
TABLE ACCESS	PARTIES	FULL	9	3
INDEX	SYS_FK0000097299N00009$	RANGE SCAN	8	0
Access Predicates				
NESTED_TABLE_ID=M.SYS_NC0000900010$				
TABLE ACCESS	MUN_BUDGETS_NT	BY INDEX ROWID	3	0
Filter Predicates				
SYS_ALIAS_5.SYS_NC00006$=L.SYS_NC00004$				

Figura 9: Custo Query d).

e) Qual partido tem mais salários por mil habitantes por ano?

```
WITH sal_party AS (
  SELECT
      l.period.year                                          AS ano,
      p.acronym                                              AS partido,
      AVG(
        m.expenses_per_1000_inhabitants(
          l.period,
          (SELECT REF(h)
             FROM Headings h
            WHERE h.id = 'D1.1')
        )
      ) AS sal_por_1000
  FROM
       Municipalities m
  JOIN TABLE(m.leaderships) l
       ON 1 = 1
  JOIN Parties p
       ON l.party = REF(p)
  GROUP BY
       l.period.year,
       p.acronym
)
SELECT
    ano,
    partido,
    ROUND(sal_por_1000, 2) AS salarios_por_1000_habitantes
FROM (
  SELECT
      s.*,
```

```
30        ROW_NUMBER() OVER (
31          PARTITION BY s.ano
32          ORDER BY s.sal_por_1000 DESC
33        ) AS rn
34    FROM sal_party s
35    WHERE s.sal_por_1000 > 0
36  )
37  WHERE rn = 1
38  ORDER BY ano;
```

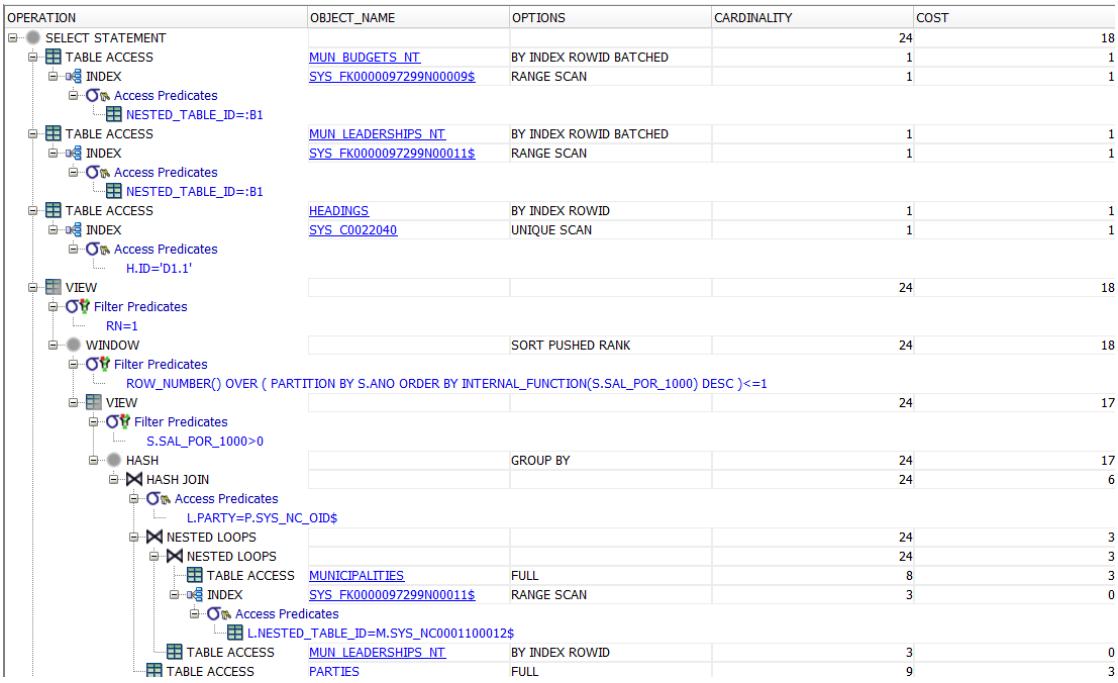
Listing 30: Consulta SQL para a alínea e)

Esta query tem como objetivo identificar, para cada ano, qual o partido político que governa os municípios com maior despesa média em salários por cada mil habitantes. Para tal, utiliza-se uma subquery com 'WITH sal_party AS (...)' onde, para cada partido e ano, se calcula a média das despesas associadas aos headings de tipo salário (cujo 'id' começa por ''SAL'). Esta média é obtida através do método 'expenses_per_1000_inhabitants(...)' da tabela 'Municipalities', que calcula a despesa por mil habitantes para um determinado período e heading. Os períodos e partidos são obtidos através da expansão das nested tables 'leaderships' e 'budgets' com 'TABLE(...)', assegurando a correspondência dos dados pelo campo 'year'. A subquery agrupa os resultados por ano e partido, e calcula a média da despesa. A query principal aplica a função analítica 'RANK()' para ordenar os partidos por despesa e selecionar, através de 'WHERE rnk = 1', apenas o partido com maior valor por mil habitantes em cada ano. O resultado é uma listagem anual dos partidos que mais gastaram com salários por mil habitantes, aproveitando as funcionalidades do modelo object-relational como métodos encapsulados e referências com 'REF(...)'.

O resultado desta query é o seguinte:

ANO	PARTIDO	SALARIOS_POR_1000_HABITANTES
2024	IL	223,45

Figura 10: Resultado Query e).



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			24	18
TABLE ACCESS	MUN_BUDGETS_NT	BY INDEX ROWID BATCHED	1	1
INDEX	SYS_FK0000097299N00009$	RANGE SCAN	1	1
Access Predicates				
NESTED_TABLE_ID=:B1				
TABLE ACCESS	MUN_LEADERSHIPS_NT	BY INDEX ROWID BATCHED	1	1
INDEX	SYS_FK0000097299N00011$	RANGE SCAN	1	1
Access Predicates				
NESTED_TABLE_ID=:B1				
TABLE ACCESS	HEADINGS	BY INDEX ROWID	1	1
INDEX	SYS_C0022040	UNIQUE SCAN	1	1
Access Predicates				
H.ID='D1.1'				
VIEW			24	18
Filter Predicates				
RN=1				
WINDOW		SORT PUSHED RANK	24	18
Filter Predicates				
ROW_NUMBER() OVER (PARTITION BY S.ANO ORDER BY INTERNAL_FUNCTION(S.SAL_POR_1000) DESC)<=1				
VIEW			24	17
Filter Predicates				
S.SAL_POR_1000>0				
HASH		GROUP BY	24	17
HASH JOIN			24	6
Access Predicates				
L.PARTY=P.SYS_NC_OID$				
NESTED LOOPS			24	3
NESTED LOOPS			24	3
TABLE ACCESS	MUNICIPALITIES	FULL	8	3
INDEX	SYS_FK0000097299N00011$	RANGE SCAN	3	0
Access Predicates				
L.NESTED_TABLE_ID=M.SYS_NC0001100012$				
TABLE ACCESS	MUN_LEADERSHIPS_NT	BY INDEX ROWID	3	0
TABLE ACCESS	PARTIES	FULL	9	3

Figura 11: Custo Query e).

f) A alínea f) propunha a criação de uma query que ilustrasse a utilização das extensões object-relacionais (OR). A query apresentada responde a esse objetivo ao tirar partido direto de várias funcionalidades típicas do modelo object-relational, nomeadamente o uso de métodos definidos no tipo Municipality_T, o acesso a atributos encapsulados dentro de objetos, e o uso de referências (REF) e desreferenciações (DEREF).

```
1  SELECT
2      SUBSTR(m.name, 1, 15)                                    AS municipio,
3      DEREF( m.get_governing_party(Period_T(2023,'ANUAL')) ).partyName
4                                                               AS partido_2023
                                                                    ,
5      m.expenses_per_1000_inhabitants( Period_T(2023,'ANUAL') )
6                                                               AS
                                                                    despesa_por_1000habitantes
                                                                    ,
7      m.total_investment_per_km2( Period_T(2023,'ANUAL') )   AS inv_por_km2
8  FROM
9      Municipalities m
10 WHERE
11     EXISTS (
12       SELECT 1
13       FROM TABLE(m.budgets) b
14       JOIN Headings h ON b.heading = REF(h)
15       WHERE LOWER(h.description) LIKE '%inv%'
16     )
17 ORDER BY
```

```
18      inv_por_km2 DESC;
```

Listing 31: Consulta SQL para a alínea f)

Esta query retorna, para cada município com despesas de investimento em 2023, o nome do município, o partido que o governa, a despesa por mil habitantes e o investimento por km².

Esta query exemplifica de forma clara o uso de extensões object-relacionais ao consultar a tabela Municipalities e invocar diretamente métodos definidos no tipo objeto Municipality_T. O objetivo é obter, para cada município com despesas de investimento registadas em 2023, o nome do município, o total de despesas no ano (total_expenses(...)), o nome do partido que governa o município nesse ano (obtido com o método get_governing_party(...) e desreferenciado com DEREF(...)), a despesa total por 1000 habitantes (expenses_per_1000_inhabitants(...)) e o investimento por quilómetro quadrado (total_investment_per_km2(...)). A cláusula WHERE EXISTS (...) filtra apenas os municípios que possuem pelo menos uma entrada orçamental com um heading de investimento (cujo id começa por 'INV'). A utilização de métodos encapsulados, nested tables e referências reflete claramente os mecanismos típicos de um modelo object-relacional, destacando-se como um exemplo prático da integração entre o paradigma relacional e o paradigma orientado a objetos no contexto da linguagem SQL3.

O resultado desta query é o seguinte:

MUNICIPIO	PARTIDO_2023	DESPESA_POR_1000HABITANTES	INV_POR_KM2
Porto	Partido Social Democrata	538,502962	2535,00724
Lisboa	Partido Socialista	228,97002	1049,47526
Braga	Partido Socialista	646,552839	572,519084
Aveiro	Iniciativa Liberal	1551,73484	531,430307
Setúbal	Livre	1012,50648	455,927052
Coimbra	Partido Comunista Português	871,711903	328,74139
Vila Franca de	Pessoas-Animais-Natureza	908,040884	327,817671
Leiria	Bloco de Esquerda	971,70398	185,807822

8 rows selected.

Figura 12: Resultado Query f).

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			10	12
TABLE ACCESS	MUN_BUDGETS_NT	BY INDEX ROWID BATCHED	1	1
INDEX	SYS_FK0000097299N00009$	RANGE SCAN	1	1
Access Predicates				
NESTED_TABLE_ID=:B1				
TABLE ACCESS	MUN_LEADERSHIPS_NT	BY INDEX ROWID BATCHED	1	1
INDEX	SYS_FK0000097299N00011$	RANGE SCAN	1	1
Access Predicates				
NESTED_TABLE_ID=:B1				
TABLE ACCESS	MUN_BUDGETS_NT	BY INDEX ROWID BATCHED	1	1
INDEX	SYS_FK0000097299N00009$	RANGE SCAN	1	1
Access Predicates				
NESTED_TABLE_ID=:B1				
TABLE ACCESS	MUN_LEADERSHIPS_NT	BY INDEX ROWID BATCHED	1	1
INDEX	SYS_FK0000097299N00011$	RANGE SCAN	1	1
Access Predicates				
NESTED_TABLE_ID=:B1				
TABLE ACCESS	MUN_BUDGETS_NT	BY INDEX ROWID BATCHED	1	1
INDEX	SYS_FK0000097299N00009$	RANGE SCAN	1	1
Access Predicates				
NESTED_TABLE_ID=:B1				
TABLE ACCESS	MUN_LEADERSHIPS_NT	BY INDEX ROWID BATCHED	1	1
INDEX	SYS_FK0000097299N00011$	RANGE SCAN	1	1
Access Predicates				
NESTED_TABLE_ID=:B1				
SORT		ORDER BY	10	12
FILTER				
Filter Predicates				
EXISTS (SELECT 0 FROM HEADINGS H,UP202108753.MUN_BUDGETS_NT B WHERE B.NESTED_TABLE_ID=:B1 AND B.HEADING=H.SYS_NC_OID$ AND LOWER(H.DESCRIPTIC				
TABLE ACCESS	MUNICIPALITIES	FULL	8	3
NESTED LOOPS			1	2
NESTED LOOPS			1	2
TABLE ACCESS	MUN_BUDGETS_NT	BY INDEX ROWID BATCHED	1	1
INDEX	SYS_FK0000097299N00009$	RANGE SCAN	1	1
Access Predicates				
B.NESTED_TABLE_ID=:B1				
INDEX	SYS_C0022042	UNIQUE SCAN	1	0
Access Predicates				
B.HEADING=H.SYS_NC_OID$				
TABLE ACCESS	HEADINGS	BY INDEX ROWID	1	1
Filter Predicates				
LOWER(H.DESCRIPTION) LIKE '%inv%'				

Figura 13: Custo Query f).

6 Conclusão

O desenvolvimento deste trabalho permitiu uma aplicação prática e aprofundada dos conceitos do modelo objeto-relacional, evidenciando as vantagens da sua utilização em cenários com estruturas de dados complexas e fortemente inter-relacionadas, como é o caso da administração pública. Através da definição de tipos de objetos, foi possível representar entidades reais com maior expressividade, incorporando não apenas os seus atributos, mas também comportamentos específicos através de métodos membro. Isto resultou num modelo mais coeso e próximo da realidade do domínio, contribuindo para uma base de dados mais semântica e com maior capacidade de abstração.

A utilização de referências (REF) e NESTED TABLES revelou-se fundamental para modelar as hierarquias entre rubricas orçamentais e as relações entre entidades geográficas, nomeadamente os diferentes níveis da nomenclatura NUTS e os municípios. Os métodos definidos no tipo Municipality_T demonstraram uma clara mais-valia, permitindo encapsular lógica de análise orçamental e estatística de forma reutilizável, limpa e integrada com o modelo de dados.

Durante a fase de povoamento da base de dados, foi necessário articular dados reais com estruturas objeto-relacionais, o que evidenciou também as dificuldades práticas associadas à gestão e integração de informação complexa. Por fim, as queries implementadas mostraram a

riqueza do modelo criado, não só por explorarem diferentes dimensões da informação (tempo, tipo de despesa, partido político, território), mas também por destacarem as vantagens das extensões do SQL3, como a invocação direta de métodos e a navegação por referências entre objetos.

Em suma, este trabalho demonstrou que o paradigma objeto-relacional oferece uma abordagem poderosa e expressiva para representar, armazenar e consultar dados complexos, contribuindo para um melhor alinhamento entre o modelo conceptual e a sua implementação física, promovendo simultaneamente boas práticas de modularidade, reutilização e clareza no desenvolvimento de bases de dados.

7 Anexos

```
DECLARE
  TYPE PartyArray_T IS TABLE OF Party_T;
  parties PartyArray_T := PartyArray_T(
    Party_T('PPD/PSD', 'Partido Social Democrata',     'direita' ),
    Party_T('CH',      'CHEGA',                        'direita' ),
    Party_T('PS',      'Partido Socialista',           'esquerda'),
    Party_T('IL',      'Iniciativa Liberal',           'direita' ),
    Party_T('L',       'Livre',                        'esquerda'),
    Party_T('PCP',     'Partido Comunista Portugu s', 'esquerda'),
    Party_T('BE',      'Bloco de Esquerda',            'esquerda'),
    Party_T('PAN',     'Pessoas-Animais-Natureza',     'centro'  ),
    Party_T('JPP',     'Juntos pelo Povo',             'centro'  )
  );
BEGIN
  FOR i IN 1 .. parties.COUNT LOOP
    INSERT INTO Parties VALUES (parties(i));
  END LOOP;
END;
/

DECLARE
  TYPE GeoRec IS RECORD (
    code VARCHAR2(10), name VARCHAR2(100), area NUMBER, pop NUMBER
  );
  TYPE GeoArr IS TABLE OF GeoRec;

  nuts1_arr GeoArr := GeoArr(
    GeoRec('PT1', 'Portugal Continental'      , 88889, 9974165),
    GeoRec('PT2', 'Regi o Aut noma dos A ores', 2322 , 239942  ),
    GeoRec('PT3', 'Regi o Aut noma da Madeira', 801  , 253259  )
  );

  nuts2_arr GeoArr := GeoArr(
    GeoRec('PT11', 'Norte'                     , 21284, 3631502),
    GeoRec('PT15', 'Algarve'                   ,  4997, 472000 ),
    GeoRec('PT19', 'Centro'                    , 28202, 1666684),
    GeoRec('PT1A', 'Pen nsula de Set bal'      ,  1421, 820305 ),
    GeoRec('PT1B', 'Grande Lisboa'             ,  1376, 2079365),
    GeoRec('PT1C', 'Alentejo'                  , 27317, 471322 ),
    GeoRec('PT1D', 'Oeste e Vale do Tejo'      ,  9201, 832987 ),
```

```
    GeoRec('PT20', 'Regi o␣Aut noma␣dos␣A ores',  2322, 239942 ),
    GeoRec('PT30', 'Regi o␣Aut noma␣da␣Madeira',   801, 253259 )
  );

  nuts3_arr GeoArr := GeoArr(
    GeoRec('PT111', 'Alto␣Minho'                  , 2219, 232220 ),
    GeoRec('PT112', 'C vado'                       , 1246, 423377 ),
    GeoRec('PT119', 'Ave'                         , 1453, 419876 ),
    GeoRec('PT11A', ' rea ␣Metropolitana␣do␣Porto', 2040, 1774104),
    GeoRec('PT11B', 'Alto␣T mega␣e␣Barroso'       , 2922, 83463  ),
    GeoRec('PT11C', 'T mega␣e␣Sousa'              , 1832, 408127 ),
    GeoRec('PT11D', 'Douro'                       , 4032, 183418 ),
    GeoRec('PT11E', 'Terras␣de␣Tr s-os-Montes'    , 5544, 106917 ),
    GeoRec('PT150', 'Algarve'                     , 4997, 472000 ),
    GeoRec('PT191', 'Regi o␣de␣Aveiro'             , 1692, 375698 ),
    GeoRec('PT192', 'Regi o␣de␣Coimbra'            , 4335, 439940 ),
    GeoRec('PT193', 'Regi o␣de␣Leiria'             , 2449, 290473 ),
    GeoRec('PT194', 'Viseu␣D o-Laf es'             , 3238, 253154 ),
    GeoRec('PT195', 'Beira␣Baixa'                 , 4614, 99046  ),
    GeoRec('PT196', 'Beiras␣e␣Serra␣da␣Estrela'   , 6305, 208373 ),
    GeoRec('PT1A0', 'Pen nsula␣de␣Set bal'         , 1421, 820305 ),
    GeoRec('PT1B0', 'Grande␣Lisboa'               , 1376, 2079365),
    GeoRec('PT1C1', 'Alentejo␣Litoral'            , 5308, 99111  ),
    GeoRec('PT1C2', 'Baixo␣Alentejo'              , 8505, 115237 ),
    GeoRec('PT1C3', 'Alto␣Alentejo'               , 6230, 104121 ),
    GeoRec('PT1C4', 'Alentejo␣Central'            , 7393, 152853 ),
    GeoRec('PT1D1', 'Oeste'                       , 2220, 376961 ),
    GeoRec('PT1D2', 'M dio␣Tejo'                   , 2283, 212796 ),
    GeoRec('PT1D3', 'Lez ria␣do␣Tejo'              , 4275, 243230 ),
    GeoRec('PT200', 'Regi o␣Aut noma␣dos␣A ores' , 2322, 239942 ),
    GeoRec('PT300', 'Regi o␣Aut noma␣da␣Madeira' ,  801, 253259 )
  );

  parent_ref REF GeoEntity_T;
BEGIN

  -- Insert top-level country (no parent)
  INSERT INTO GeoEntities VALUES (
    GeoEntity_T('PT', 'Portugal', 92012, 10467366, 'country', NULL)
  );

  -- Insert NUTS I
  FOR i IN nuts1_arr.FIRST .. nuts1_arr.LAST LOOP
    SELECT REF(e) INTO parent_ref FROM GeoEntities e WHERE e.code = 'PT';
    INSERT INTO GeoEntities VALUES (
      GeoEntity_T(
        nuts1_arr(i).code,
        nuts1_arr(i).name,
        nuts1_arr(i).area,
        nuts1_arr(i).pop,
        'NUTS␣I',
        NULL
      )
```

```
94      );
95    END LOOP;
96
97    -- Insert NUTS II (parent = appropriate NUTS I)
98    FOR i IN nuts2_arr.FIRST .. nuts2_arr.LAST LOOP
99      SELECT REF(e) INTO parent_ref FROM GeoEntities e WHERE e.code =
          SUBSTR(nuts2_arr(i).code,1,3);
100     INSERT INTO GeoEntities VALUES (
101       GeoEntity_T(
102         nuts2_arr(i).code,
103         nuts2_arr(i).name,
104         nuts2_arr(i).area,
105         nuts2_arr(i).pop,
106         'NUTS II',
107         parent_ref
108       )
109     );
110   END LOOP;
111
112   -- Insert NUTS III (parent from nuts2)
113   FOR i IN nuts3_arr.FIRST .. nuts3_arr.LAST LOOP
114     SELECT REF(e) INTO parent_ref FROM GeoEntities e WHERE e.code =
          SUBSTR(nuts3_arr(i).code,1,4);
115     INSERT INTO GeoEntities VALUES (
116       GeoEntity_T(
117         nuts3_arr(i).code,
118         nuts3_arr(i).name,
119         nuts3_arr(i).area,
120         nuts3_arr(i).pop,
121         'NUTS III',
122         parent_ref
123       )
124     );
125   END LOOP;
126
127   COMMIT;
128 END;
129 /
130
131 -- Top-level EXPENSES
132 INSERT INTO Headings VALUES (
133   Heading_T('D1', 'Despesas Correntes', 'D', 1, NULL, Heading_RefList_T()
        )
134 );
135
136 INSERT INTO Headings VALUES (
137   Heading_T('D2', 'Despesas de Capital', 'D', 1, NULL, Heading_RefList_T
        ())
138 );
139
140 -- Top-level REVENUES
141 INSERT INTO Headings VALUES (
```

```
142   Heading_T('R1', 'Receitas␣Correntes', 'R', 1, NULL, Heading_RefList_T()
          )
143 );
144
145 INSERT INTO Headings VALUES (
146   Heading_T('R2', 'Receitas␣de␣Capital', 'R', 1, NULL, Heading_RefList_T
          ())
147 );
148
149 DECLARE
150   parent REF Heading_T;
151 BEGIN
152   -- Children of D1
153   SELECT REF(h) INTO parent FROM Headings h WHERE h.id = 'D1';
154   INSERT INTO Headings VALUES (Heading_T('D1.1', 'Sal rios', 'D', 2,
          parent, Heading_RefList_T()));
155   INSERT INTO Headings VALUES (Heading_T('D1.2', 'Aquisi  o␣de␣Bens␣e␣
          Servi os', 'D', 2, parent, Heading_RefList_T()));
156   INSERT INTO Headings VALUES (Heading_T('D1.3', 'Juros␣da␣D vida', 'D',
           2, parent, Heading_RefList_T()));
157
158   -- Children of D2
159   SELECT REF(h) INTO parent FROM Headings h WHERE h.id = 'D2';
160   INSERT INTO Headings VALUES (Heading_T('D2.1', 'Investimentos', 'D', 2,
           parent, Heading_RefList_T()));
161   INSERT INTO Headings VALUES (Heading_T('D2.2', 'Aquisi  o␣de␣Terrenos
          ␣e␣Edif cios', 'D', 2, parent, Heading_RefList_T()));
162
163   -- Children of R1
164   SELECT REF(h) INTO parent FROM Headings h WHERE h.id = 'R1';
165   INSERT INTO Headings VALUES (Heading_T('R1.1', 'Impostos␣Diretos', 'R',
           2, parent, Heading_RefList_T()));
166   INSERT INTO Headings VALUES (Heading_T('R1.2', 'Transfer ncias␣do␣
          Estado', 'R', 2, parent, Heading_RefList_T()));
167
168   -- Children of R2
169   SELECT REF(h) INTO parent FROM Headings h WHERE h.id = 'R2';
170   INSERT INTO Headings VALUES (Heading_T('R2.1', 'Venda␣de␣Bens␣de␣
          Investimento', 'R', 2, parent, Heading_RefList_T()));
171   INSERT INTO Headings VALUES (Heading_T('R2.2', 'Transfer ncias␣de␣
          Capital', 'R', 2, parent, Heading_RefList_T()));
172 END;
173 /
174
175 DECLARE
176   TYPE MunRec IS RECORD (
177     code          VARCHAR2(10),
178     name          VARCHAR2(100),
179     area          NUMBER,
180     pop           NUMBER,
181     parent_code   VARCHAR2(10),
182     party_acr     VARCHAR2(10),
183     budget_amt    NUMBER
```

```
184   );
185   TYPE MunArr IS TABLE OF MunRec;
186
187   municipalities MunArr := MunArr(
188     MunRec('LIS', 'Lisboa',    100.05, 545923,  'PT1B0', 'PS'      ,
          250000),
189     MunRec('PRT', 'Porto',      41.42, 232125,  'PT11A', 'PPD/PSD',
          180000),
190     MunRec('BRG', 'Braga',      183.4, 193333,  'PT112', 'PS'      ,
          120000),
191     MunRec('AVR', 'Aveiro',    197.58,  80555,  'PT191', 'IL'      ,
          95000),
192     MunRec('CBR', 'Coimbra',    319.4, 143396,  'PT192', 'PCP'     ,
          105000),
193     MunRec('LEI', 'Leiria',     565.1, 128640,  'PT193', 'BE'      ,
          88000),
194     MunRec('VFX', 'Vila␣Franca␣de␣Xira', 320.3, 137659, 'PT1A0', 'PAN'),
195     MunRec('STS', 'Set bal',   230.3, 123456, 'PT1A0', 'L')
196   );
197
198   parent_ref REF GeoEntity_T;
199   party_ref  REF Party_T;
200   budgets BudgetList_T;
201   leads LeadershipList_T;
202   heading_ref REF Heading_T;
203
204 BEGIN
205   FOR i IN 1 .. municipalities.COUNT LOOP
206     SELECT REF(e) INTO parent_ref FROM GeoEntities e WHERE e.code =
          municipalities(i).parent_code;
207     SELECT REF(p) INTO party_ref FROM Parties p WHERE p.acronym =
          municipalities(i).party_acr;
208
209     -- Create budgets
210     budgets := BudgetList_T();
211     budgets.EXTEND(8);
212
213     SELECT REF(h) INTO heading_ref FROM Headings h WHERE h.id = 'D1.1';
214     budgets(1) := BudgetEntry_T('D', Period_T(2022, 'ANUAL'), 80000,
          heading_ref);
215
216     SELECT REF(h) INTO heading_ref FROM Headings h WHERE h.id = 'D1.2';
217     budgets(2) := BudgetEntry_T('D', Period_T(2023, '1'),     20000,
          heading_ref);
218
219     SELECT REF(h) INTO heading_ref FROM Headings h WHERE h.id = 'D2.1';
220     budgets(3) := BudgetEntry_T('D', Period_T(2023, 'ANUAL'), 105000,
          heading_ref);
221
222     SELECT REF(h) INTO heading_ref FROM Headings h WHERE h.id = 'R1.2';
223     budgets(4) := BudgetEntry_T('R', Period_T(2023, 'ANUAL'), 110000,
          heading_ref);
224
```

```
225     SELECT REF(h) INTO heading_ref FROM Headings h WHERE h.id = 'D1.1';
226     budgets(5) := BudgetEntry_T('D', Period_T(2024, '2'),      18000,
            heading_ref);
227
228     SELECT REF(h) INTO heading_ref FROM Headings h WHERE h.id = 'R2.2';
229     budgets(6) := BudgetEntry_T('R', Period_T(2024, 'ANUAL'), 150000,
            heading_ref);
230
231     SELECT REF(h) INTO heading_ref FROM Headings h WHERE h.id = 'D2.1';
232     budgets(7) := BudgetEntry_T('D', Period_T(2022, 'ANUAL'), 50000,
            heading_ref);
233
234     SELECT REF(h) INTO heading_ref FROM Headings h WHERE h.id = 'D2.2';
235     budgets(8) := BudgetEntry_T('D', Period_T(2024, 'ANUAL'), 75000,
            heading_ref);
236
237     -- Leaderships
238     leads := LeadershipList_T();
239     leads.EXTEND(3);
240     leads(1) := Leadership_T(Period_T(2023, NULL), party_ref);
241     leads(2) := Leadership_T(Period_T(2023, NULL), party_ref);
242     leads(3) := Leadership_T(Period_T(2024, NULL), party_ref);
243
244     -- Insert municipality
245     INSERT INTO Municipalities VALUES (
246       Municipality_T(
247         municipalities(i).code,
248         municipalities(i).name,
249         municipalities(i).area,
250         municipalities(i).pop,
251         'municipality',
252         parent_ref,
253         budgets,
254         leads
255       )
256     );
257   END LOOP;
258
259   COMMIT;
260 END;
261 /
```

Listing 32: Código utilizado na Questão 2