Object Relational – Teaching Service

Assignment 2 Report

Integrated Masters in Informatics and Computing Engineering

Database Technologies

Grupo A:
Carlos Albuquerque - up201706735@edu.fe.up.pt
Gonçalo Marantes - up201706917@edu.fe.up.pt
Tito Griné - up201706732@edu.fe.up.pt

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

May 17, 2021

Summary

The object oriented paradigm (OOP) is one of the most (if not the most) adopted design concept in software engineering. And for a good reason: it is a simple abstraction of the real world, it provides useful mechanisms for mutability and it is the basis for many design patterns and industry best practices.

Because the OOP principles are so ubiquitous, SQL also embraced them since SQL3. This version of SQL provides extensions to the original query language, supporting the creation of Object Relational Databases (ORDB). Even though they are not very common, they have some interesting features that change the way a database is conceived and queried.

The goal of this project is to experiment and use these ORDB features to implement an actual database model. In this report, we go through each step of process and describe our approach and considerations regarding the model. We also explore different ways of querying the database and try to make the most of the SQL3 extensions.

By the end of the project, we were able to successfully explore the available object oriented functionalities and implement a useful model that became incredibly easier to query. However, it does seem slower and is somewhat harder to update and populate. Nevertheless, our knowledge of the object relational possibilities and limitations greatly increased and, as such, we deem this a very successful project.

Contents

1 Object Relational Data Model **4**

1.1 Type creation . . . 4

1.2 Table creation . . . 6

2 Populate Model **7**

2.1 SQL Script . . . 7

3 Object Methods and Functions **10**

3.1 SQL Script . . . 10

4 Object-Relational Database Queries **11**

4.1 Question a) . . . 11

4.2 Question b) . . . 11

4.3 Question c) . . . 12

4.4 Question d) . . . 13

4.5 Question e) . . . 14

4.6 Question f) . . . 15

5 Conclusion **16**

References **17**

List of Figures

1 Object relational model . . . 4

2 Question a) answer . . . 11

3 Question b) answer . . . 12

4 Question c) answer . . . 13

5 Question d) answer . . . 14

6 Question e) answer . . . 15

Listings

1 Type creation SQL script . . . 4

2 Table creation SQL script . . . 6

3 Model population SQL script . . . 7

4 SQL script for method creation . . . 10

5 Question a) SQL script . . . 11

6 Question b) SQL script . . . 12

7 Question c) SQL script . . . 13

8 Question d) SQL script . . . 13

9 Question e) SQL script . . . 14

10 Question f) SQL script . . . 15

1 Object Relational Data Model

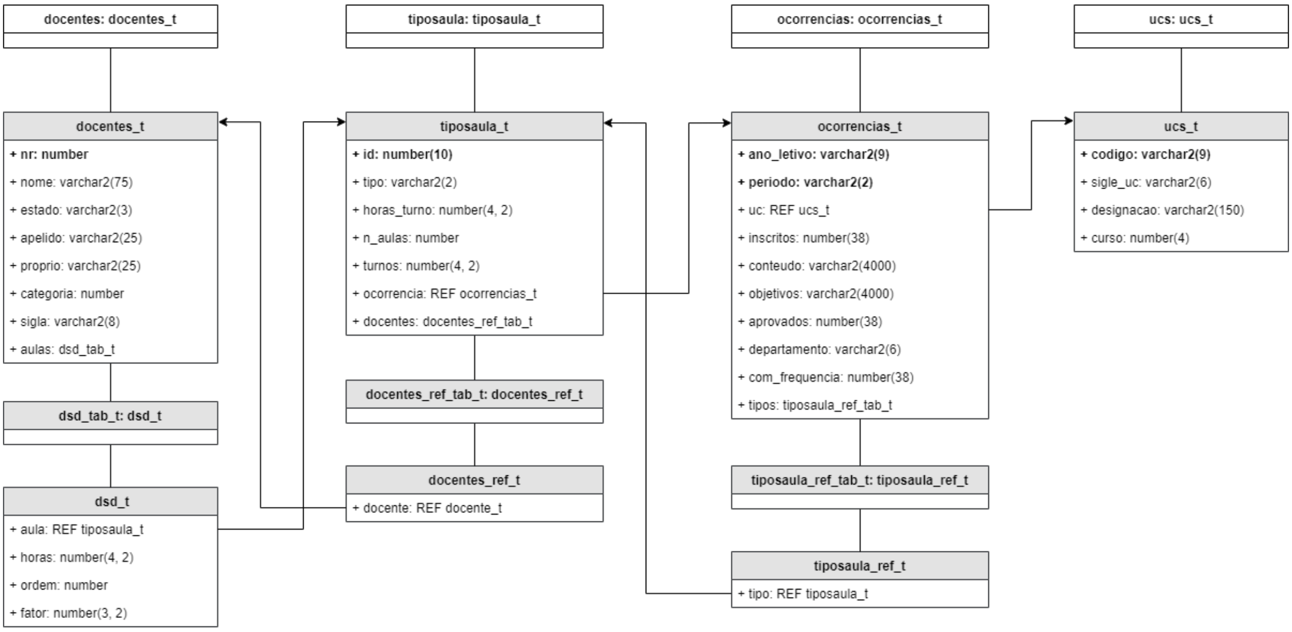


Figure 1: Object relational model

The schema now only has four tables: DOCENTES; TIPOSAULA; OCORRENCIAS; UCS.

OCORRENCIAS, now represented as the type ocorrencias_t, holds a reference to an object ucs_t, analogous to the previous foreign key to UCS. It also has a nested table of references to tiposaula_t, thus facilitating accessing the corresponding types of each occurrence. In turn, the object tiposaula_t has a reference to the corresponding occurrence, through the ocorrencias_t type.

To model the original DSD table, that acted as an intermediate table between DOCENTES and TIPOSAULA, we used the asymmetrical representation approach. As such, the object docentes_t has a nested table of dsd_t, that in turn holds the attributes previously in DSD and a reference to the corresponding object tiposaula_t instead of the previous foreign key. Finally, tiposaula_t has a nested table of references to docentes_t. We opted for this solution since the information previously stored in the DSD table is most relevant for the professors, thus rendering it more useful on docentes_t, and therefore avoiding the need for an extra table.

1.1 Type creation

```
DROP TYPE ucs_t FORCE;
DROP TYPE ocorrencias_t FORCE;
DROP TYPE docentes_t FORCE;
DROP TYPE docentes_ref_t FORCE;
DROP TYPE docentes_ref_tab_t FORCE;
DROP TYPE tiposaula_t FORCE;
DROP TYPE tiposaula_ref_t FORCE;
DROP TYPE tiposaula_ref_tab_t FORCE;
DROP TYPE dsd_t FORCE;
DROP TYPE dsd_tab_t FORCE;

CREATE TYPE ucs_t AS object
(
  codigo     VARCHAR2(9),
  sigla_uc   VARCHAR2(6),
  designacao VARCHAR2(150),
  curso      NUMBER(4),
  -- methods --
  MAP MEMBER FUNCTION get_avg_approval_rate
    RETURN NUMBER
);
/
CREATE TYPE ocorrencias_t AS object
(
  uc             REF ucs_t,
  ano_letivo     VARCHAR2(9),
```

```sql
  periodo        VARCHAR2(2),
  inscritos      NUMBER(38),
  conteudo       VARCHAR2(4000),
  objetivos      VARCHAR2(4000),
  aprovados      NUMBER(38),
  departamento   VARCHAR2(6),
  com_frequencia NUMBER(38)
  /* tipos tiposaula_ref_tab_t to be added later */
);
/
CREATE TYPE docentes_t AS object
(
  nr        NUMBER,
  nome      VARCHAR2(75),
  estado    VARCHAR2(3),
  apelido   VARCHAR2(25),
  proprio   VARCHAR2(25),
  categoria NUMBER,
  sigla     VARCHAR2(8),
  /* aulas dsd_tab_t to be added later */
);
/
CREATE TYPE docentes_ref_t AS object
(
  docente REF docentes_t
);
/
CREATE TYPE docentes_ref_tab_t AS TABLE OF docentes_ref_t;
/
CREATE TYPE tiposaula_t AS object
(
  id          NUMBER(10),
  tipo        VARCHAR2(2),
  horas_turno NUMBER(4,2),
  n_aulas     NUMBER,
  turnos      NUMBER(4,2),
  ocorrencia  REF ocorrencias_t,
  docentes    docentes_ref_tab_t,
  -- methods--
  MAP MEMBER FUNCTION class_hours
    RETURN NUMBER,
  STATIC FUNCTION total_hours(v_curso number, v_periodo varchar2, v_ano_letivo varchar2)
    RETURN NUMBER
);
/
CREATE TYPE tiposaula_ref_t AS object
(
  tipo REF tiposaula_t
);
/
CREATE TYPE tiposaula_ref_tab_t AS TABLE OF tiposaula_ref_t;
/
CREATE TYPE dsd_t AS object
(
  aula  REF tiposaula_t,
  horas NUMBER(4,2),
  ordem NUMBER,
  fator NUMBER(3,2)
);
/
CREATE TYPE dsd_tab_t AS TABLE OF dsd_t;
/
-- Insert missing fields due to circularity
ALTER TYPE ocorrencias_t ADD attribute
(
  tipos tiposaula_ref_tab_t
) CASCADE;

ALTER TYPE docentes_t ADD attribute
(
  aulas dsd_tab_t
) CASCADE;
```

Listing 1: Type creation SQL script

1.2 Table creation

```sql
DROP TABLE docentes CASCADE CONSTRAINTS;
DROP TABLE ocorrencias CASCADE CONSTRAINTS;
DROP TABLE ucs CASCADE CONSTRAINTS;
DROP TABLE tiposaula CASCADE CONSTRAINTS;

CREATE TABLE ucs OF ucs_t
(
    designacao NOT NULL,
    CONSTRAINT ucs_pk PRIMARY KEY (codigo)
);

CREATE TABLE docentes OF docentes_t
(
    nome   NOT NULL,
    sigla  NOT NULL,
    estado NOT NULL,
    CONSTRAINT docentes_pk PRIMARY KEY (nr)
) nested TABLE aulas store AS aulas_tab;

CREATE TABLE ocorrencias OF ocorrencias_t
(
    uc          NOT NULL,
    ano_letivo NOT NULL,
    periodo    NOT NULL,
    CONSTRAINT ocorrencias_ucs_fk FOREIGN KEY (uc) REFERENCES ucs
) nested TABLE tipos store AS tipos_tab;

CREATE TABLE tiposaula OF tiposaula_t
(
    tipo NOT NULL,
    CONSTRAINT tiposaula_pk PRIMARY KEY (id),
    CONSTRAINT tiposaula_ocorrencias_fk FOREIGN KEY (ocorrencia) REFERENCES ocorrencias
) nested TABLE docentes store AS docentes_tab;
```

Listing 2: Table creation SQL script

2 Populate Model

2.1 SQL Script

To populate the tables, we begin with the tables ucs, ocorrencias and docentes by simply selecting from the analogous original relational database and directly inserting in the new tables the necessary fields. On ocorrencias we have to get the reference to the ucs_t object, which can be done by simply matching the codigo from the original database to the newly created one. Also, in order to facilitate inserting the nested tables, we insert empty tables along with the data from the original table in ocorrenciass and docentes.

In order to populate the table tiposaula we make use of a procedure. It starts by declaring a cursor for the results of querying the original database for the necessary fields for tiposaula_t. It then has a loop that will iterate through the cursor, inserting the values into tiposaula keeping a reference to the created object. With this reference it also inserts it in the corresponding ocorrencias_t object.

Finally, another procedure is used to populate the missing attributes for the objects in docentes and tiposaula. Much like the previous procedure, it declares a cursor for the results of querying the original database to get the necessary values from DSD, but also two types for holding references to objects docentes_t and tiposaula_t. The procedure will then loop through the cursor, retrieving the reference to the docentes_t object and the tiposaula_t object. With it, the reference to docente_t is added to the respective tiposaula_t, and the values from DSD are added to the corresponding docente_t, together with the reference to the tiposaula_t object.

```sql
-- ucs
INSERT INTO ucs
SELECT codigo,
       sigla_uc,
       designacao,
       curso
FROM   xucs;

-- ocorrencias
INSERT INTO ocorrencias
SELECT ref(u),
       ano_letivo,
       periodo,
       inscritos,
       conteudo,
       objetivos,
       aprovados,
       departamento,
       com_frequencia,
       tiposaula_ref_tab_t() AS tipos
FROM   xocorrencias,
       ucs u
WHERE  u.codigo = xocorrencias.codigo;

-- docentes
INSERT INTO docentes
SELECT nr,
       nome,
       estado,
       apelido,
       proprio,
       categoria,
       sigla,
       dsd_tab_t() AS aulas
FROM   xdocentes;

-- tipos aula
DECLARE
  CURSOR tiposaula_it IS
    SELECT id,
           tipo,
           horas_turno,
           n_aulas,
           turnos,
           ref(o) AS ocorrencia
    FROM   xtiposaula
    join   ocorrencias o
    ON     xtiposaula.ano_letivo = o.ano_letivo
    AND    xtiposaula.periodo = o.periodo
    AND    xtiposaula.codigo = o.uc.codigo;

    tipos_ref REF TIPOSAULA_T;
BEGIN
  FOR tiposaula_rec IN tiposaula_it
  LOOP
    INSERT INTO tiposaula t VALUES
                (
                    tiposaula_rec.id,
                    tiposaula_rec.tipo,
                    tiposaula_rec.horas_turno,
                    tiposaula_rec.n_aulas,
                    tiposaula_rec.turnos,
                    tiposaula_rec.ocorrencia,
                    docentes_ref_tab_t()
                )
    returning   ref(t)
    INTO        tipos_ref;

    INSERT INTO table
                (
                    SELECT tipos
                    FROM   ocorrencias o
                    WHERE  tiposaula_rec.ocorrencia = ref(o)
                )
                VALUES
                (
                    tiposaula_ref_t(tipos_ref)
                );
  END LOOP;

  COMMIT;
EXCEPTION
WHEN OTHERS THEN
  ROLLBACK;
  RAISE;
END;

-- docentes - dsd - tiposaula
DECLARE
  CURSOR dsd_it IS
    SELECT nr,
           id,
           horas,
           ordem,
           fator
    FROM   xdsd;

  docente_ref REF DOCENTES_T;
  tiposaula_ref REF TIPOSAULA_T;
BEGIN
  FOR dsd_rec IN dsd_it
  LOOP
    SELECT ref(d)
    INTO   docente_ref
    FROM   docentes d
    WHERE  d.nr = dsd_rec.nr;

    SELECT ref(t)
    INTO   tiposaula_ref
    FROM   tiposaula t
    WHERE  t.id = dsd_rec.id;

    INSERT INTO table
                (
                    SELECT docentes
                    FROM   tiposaula t
                    WHERE  t.id = dsd_rec.id
                )
                VALUES
                (
                    docente_ref
                );

    INSERT INTO table
                (
                    SELECT aulas
                    FROM   docentes d
```

```sql
                    WHERE  d.nr = dsd_rec.nr
                )
                VALUES
                (
                    tiposaula_ref,
                    dsd_rec.horas,
                    dsd_rec.ordem,
                    dsd_rec.fator
                );
  END LOOP;

  COMMIT;
EXCEPTION
WHEN OTHERS THEN
  ROLLBACK;
  RAISE;
END;
```

Listing 3: Model population SQL script

3 Object Methods and Functions

To further explore the object relational capabilities we developed a couple of methods useful for solving the queries proposed in the assignment. One of the most common is the calculation of `class_hours`, which is simply the number of shifts times the amount of hours for each shift. For question e), we developed the `total_hours` function that calculates the total amount of weekly hours of classes for the specified `curso`, in the specified `ano_letivo` and `periodo`. Finally, for question f) we came up with the `get_avg_approval_rate` map method for the `ucs_t` type that calculates the average approval rate of the curricular unit. This allows us to quickly order ucs by their approval rates.

We did not find any more relevant methods, which limited our ability to explore and further develop this part of the model.

3.1 SQL Script

```sql
CREATE OR replace TYPE BODY tiposaula_t AS
    -- class_hours
    MAP MEMBER FUNCTION class_hours
      RETURN NUMBER
    IS
    BEGIN
      RETURN horas_turno * turnos;
    END class_hours;

    -- total_hours
    STATIC FUNCTION total_hours(v_curso      NUMBER,
                              v_periodo    VARCHAR2,
                              v_ano_letivo VARCHAR2)
      RETURN NUMBER
    IS
    v_total NUMBER;
    BEGIN
      SELECT COALESCE(SUM(t.class_hours()), 0) AS horas_semanais
      INTO   v_total
      FROM   tiposaula t
      WHERE  t.ocorrencia.uc.curso = v_curso
             AND t.ocorrencia.periodo = v_periodo
             AND t.ocorrencia.ano_letivo = v_ano_letivo;
      RETURN v_total;
    END total_hours;
END;

CREATE OR REPLACE TYPE BODY ucs_t AS
    MAP MEMBER FUNCTION get_avg_approval_rate
      RETURN NUMBER
    IS
      approval_rate NUMBER;
    BEGIN
      SELECT avg(o.aprovados/o.inscritos)
      INTO   approval_rate
      FROM   ocorrencia o
      WHERE  o.uc = ref(self);

      RETURN approval_rate;
    END get_avg_approval_rate;
END;
```

Listing 4: SQL script for method creation

4 Object-Relational Database Queries

4.1 Question a)

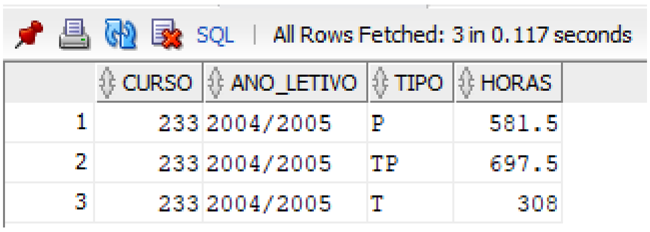
"How many class hours of each type did the program 233 got in year 2004/2005?"

As we will see along the various sections in this report, there is more than one way to achieve a certain result. Since we are using nested tables and bidirectional references, we can start from more than one table. For this query, most of the information we need and the filters we apply are on the `ocorrencias` table. Therefore we decided to start the query on that table and then fetch the values we need from the referenced objects in the `tipos` nested table. However, it is possible to also start from the `tiposaula` table, that option is also available in the listing 5.

```sql
-- starting with the ocorrencias table
SELECT o.uc.curso                 AS curso,
       o.ano_letivo               AS ano_Letivo,
       t.tipo.tipo                AS tipo,
       SUM(t.tipo.class_hours())  AS horas
FROM   ocorrencias o,
       TABLE(o.tipos) t
WHERE  o.uc.curso = 233
       AND o.ano_letivo = '2004/2005'
GROUP  BY t.tipo.tipo,
          ano_letivo,
          o.uc.curso;

-- starting with the tiposaula table
SELECT t.ocorrencia.uc.curso    AS curso,
       t.ocorrencia.ano_letivo  AS ano_Letivo,
       t.tipo                   AS tipo,
       SUM(t.class_hours())     AS horas
FROM   tiposaula t
WHERE  t.ocorrencia.uc.curso = 233
       AND t.ocorrencia.ano_letivo = '2004/2005'
GROUP  BY t.tipo,
          t.ocorrencia.ano_letivo,
          t.ocorrencia.uc.curso;
```

Listing 5: Question a) SQL script



All Rows Fetched: 3 in 0.117 seconds

	CURSO	ANO_LETIVO	TIPO	HORAS
1	233	2004/2005	P	581.5
2	233	2004/2005	TP	697.5
3	233	2004/2005	T	308

Figure 2: Question a) answer

4.2 Question b)

"Which courses (show the code, total class hours required, total classes assigned) have a difference between total class hours required and the service actually assigned in year 2003/2004?"

Since we are using nested tables we can start this query from any of the tables `ocorrencias`, `tiposaula` or `docentes` as we can see in listing 6. Depending on where we start there may be a need to verify some relations on the where clause. For example, if we start with `tiposaula`, we move on to the `docentes` table which have a nested table `aulas`, which is a table of type `dsd_t`. Now we only have to make sure that the objects `aulas` in reference the same `tiposaula` tuple, which can be verified by the `a.aula = ref(t)` clause.

On the other hand, if we start from the `docentes` table, then we do not need to make this verification as it is explicitly done simply by using a cursor (i.e. the "." notation in objects). For example, `a.aula.ocorrencia .ano_letivo = '2003/2004'`, we are starting for a `dsd_t` object, that must have access to a `tiposaula_t` object, that itsleft has a reference to a `ocorrencia_t` object in which the `ano_letivo` attribute corresponds to the string

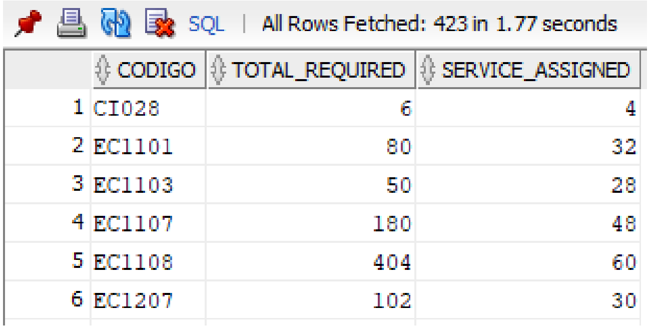
'2003/2004'.

```sql
-------------------------------
-- Starting at ocorrencias
SELECT o.uc.codigo               AS codigo,
       SUM(t.tipo.class_hours()) AS total_required,
       SUM(a.horas)              AS service_assigned
FROM   ocorrencias o,
       TABLE(o.tipos) t,
       TABLE(t.tipo.docentes) d,
       TABLE(d.docente.aulas) a
WHERE  o.ano_letivo = '2003/2004'
       AND a.aula = t.tipo
GROUP  BY o.uc.codigo
HAVING SUM(t.tipo.class_hours()) != SUM(a.horas)
ORDER  BY o.uc.codigo;

---------------------------------
-- Starting at tiposaula
SELECT t.ocorrencia.uc.codigo AS codigo,
       SUM(t.class_hours())   AS total_required,
       SUM(a.horas)           AS service_assigned
FROM   tiposaula t,
       TABLE(t.docentes) d,
       TABLE(d.docente.aulas) a
WHERE  t.ocorrencia.ano_letivo = '2003/2004'
       AND a.aula = ref(t)
GROUP  BY t.ocorrencia.uc.codigo
HAVING SUM(t.class_hours()) != SUM(a.horas)
ORDER  BY t.ocorrencia.uc.codigo;

-------------------------------------
-- Starting at docentes
SELECT a.aula.ocorrencia.uc.codigo AS codigo,
       SUM(a.aula.class_hours())   AS total_required,
       SUM(a.horas)                AS service_assigned
FROM   docentes d,
       TABLE(d.aulas) a
WHERE  a.aula.ocorrencia.ano_letivo = '2003/2004'
GROUP  BY a.aula.ocorrencia.uc.codigo
HAVING SUM(a.aula.class_hours()) != SUM(a.horas)
ORDER  BY a.aula.ocorrencia.uc.codigo;
```

Listing 6: Question b) SQL script



All Rows Fetched: 423 in 1.77 seconds

	CODIGO	TOTAL_REQUIRED	SERVICE_ASSIGNED
1	CI028	6	4
2	EC1101	80	32
3	EC1103	50	28
4	EC1107	180	48
5	EC1108	404	60
6	EC1207	102	30

Figure 3: Question b) answer

4.3 Question c)

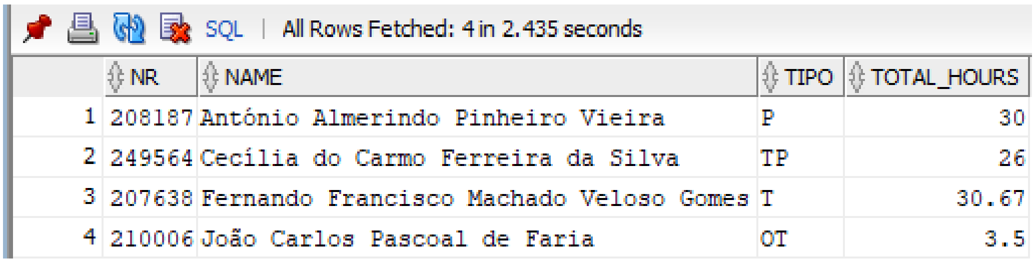
"Who is the professor with more class hours for each type of class, in the academic year 2003/2004? Show the number and name of the professor, the type of class and the total of class hours times the factor."

To start with, we can simply aggregate the sum of weekly hours for each professor and class type. Then it is only necessary to find the maximum number of weekly hours for each class type, which is also aggregated by professor. In order to do this we have created a VIEW called total_per_type which stores the weekly hours per professor and class type. This VIEW is then used in the main query to fetch the maximum weekly hours in the total_per_type VIEW.

```
CREATE OR replace VIEW total_per_type
AS
  SELECT d.docente.nr    AS nr,
         d.docente.nome  AS name,
         t.tipo.tipo     AS tipo,
         SUM(a.horas)    AS total_hours
  FROM   ocorrencias o,
         TABLE(o.tipos) t,
         TABLE(t.tipo.docentes) d,
         TABLE(d.docente.aulas) a
  WHERE  o.ano_letivo = '2003/2004'
         AND a.aula = t.tipo
  GROUP  BY t.tipo.tipo,
            d.docente.nr,
            d.docente.nome
  ORDER  BY d.docente.nome;

SELECT nr,
       name,
       total_per_type.tipo,
       total_hours
FROM   (SELECT tipo,
               MAX(total_hours) AS max_hours
        FROM   total_per_type
        GROUP  BY tipo) max_per_type
       JOIN total_per_type
         ON max_per_type.tipo = total_per_type.tipo
WHERE  max_per_type.max_hours = total_per_type.total_hours;
```

Listing 7: Question c) SQL script



All Rows Fetched: 4 in 2.435 seconds

	NR	NAME	TIPO	TOTAL_HOURS
1	208187	António Almerindo Pinheiro Vieira	P	30
2	249564	Cecília do Carmo Ferreira da Silva	TP	26
3	207638	Fernando Francisco Machado Veloso Gomes	T	30.67
4	210006	João Carlos Pascoal de Faria	OT	3.5

Figure 4: Question c) answer

4.4 Question d)

"Which is the average number of hours by professor by year in each category, in the years between 2001/2002 and 2004/2005?"

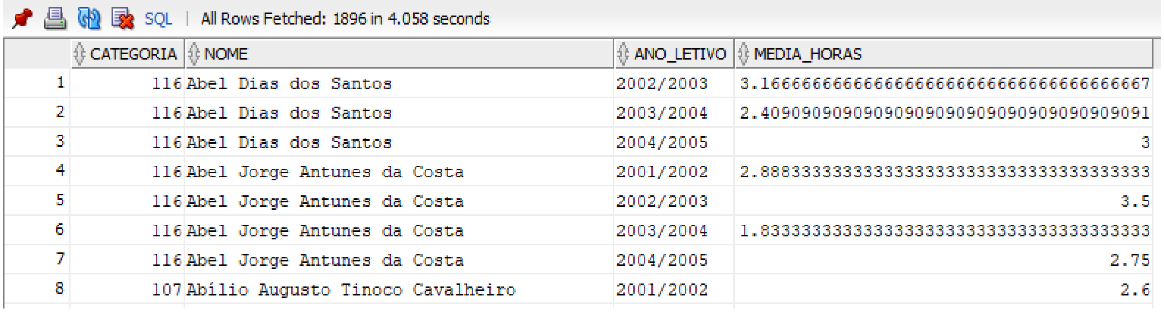
This query can be done in several ways, we can start at either one of the border tables, i.e `docentes` or `ocorrencia`. However, if we perform the query starting in the `docentes` table, the query will be smaller and simpler. This is because transversing the schema defined in figure 1 from left to right gives us more control on which objects are being accessed due to the fact that the relations are one-to-many. On the other hand, if we start by accessing the `ocorrencias` table, the many-to-one relations might become complex, as we need to treat these references as tables, which will require the use of the `TABLE()` function more often. This can be verified in the SQL code in listing 8.

```
-- docentes -> ocorrencia
SELECT d.categoria                     AS categoria,
       d.nome                          AS nome,
       a.aula.ocorrencia.ano_letivo    AS ano_letivo,
       AVG(a.horas)                    AS media_horas
FROM   docentes d,
       TABLE(d.aulas) a
WHERE  a.aula.ocorrencia.ano_letivo IN ( '2001/2002', '2002/2003', '2003/2004', '2004/2005' )
GROUP  BY d.categoria,
          a.aula.ocorrencia.ano_letivo,
          d.nome
ORDER  BY nome,
          ano_letivo;

-- ocorrencia -> docentes
```

```
SELECT d.docente.categoria  AS categoria,
       d.docente.nome       AS nome,
       o.ano_letivo         AS ano_letivo,
       AVG(a.horas)         AS media_horas
FROM   ocorrencias o,
       TABLE(o.tipos) t,
       TABLE(t.tipo.docentes) d,
       TABLE(d.docente.aulas) a
WHERE  o.ano_letivo IN ( '2001/2002', '2002/2003', '2003/2004', '2004/2005' )
       AND a.aula = t.tipo
GROUP  BY d.docente.categoria,
          o.ano_letivo,
          d.docente.nome
ORDER  BY nome,
          ano_letivo;
```

Listing 8: Question d) SQL script



SQL | All Rows Fetched: 1896 in 4.058 seconds

	CATEGORIA	NOME	ANO_LETIVO	MEDIA_HORAS
1	116	Abel Dias dos Santos	2002/2003	3.16666666666666666666666666666666666667
2	116	Abel Dias dos Santos	2003/2004	2.40909090909090909090909090909090909091
3	116	Abel Dias dos Santos	2004/2005	3
4	116	Abel Jorge Antunes da Costa	2001/2002	2.88333333333333333333333333333333333333
5	116	Abel Jorge Antunes da Costa	2002/2003	3.5
6	116	Abel Jorge Antunes da Costa	2003/2004	1.83333333333333333333333333333333333333
7	116	Abel Jorge Antunes da Costa	2004/2005	2.75
8	107	Abílio Augusto Tinoco Cavalheiro	2001/2002	2.6

Figure 5: Question d) answer

4.5 Question e)

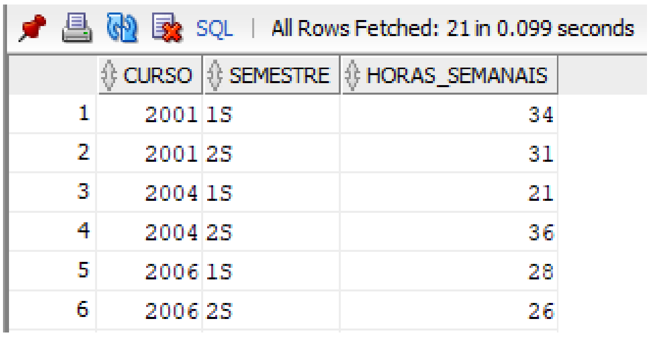
"Which is the total hours per week, on each semester, that an hypothetical student enrolled in every course of a single curricular year from each program would get."

In order to answer this query we can simply use the `horas_turno` column of the `tiposaula` table, which represents the number of week hours for each class. Then we simply need to sum these columns grouped by both semesters and program.

```
SELECT t.ocorrencia.uc.curso  AS curso,
       t.ocorrencia.periodo   AS semestre,
       SUM(t.horas_turno)     AS horas_semanais
FROM   tiposaula t
WHERE  t.ocorrencia.periodo IN ( '1S', '2S' )
       AND t.ocorrencia.ano_letivo = (-- this may be an argument (2009/2010)
                                     SELECT MAX(o.ano_letivo)
                                     FROM   ocorrencias o)
GROUP  BY t.ocorrencia.uc.curso,
          t.ocorrencia.periodo
ORDER  BY t.ocorrencia.uc.curso,
          t.ocorrencia.periodo;

-----------------------------------
-- using static function in tiposaula_t
SELECT t.ocorrencia.uc.curso     AS curso,
       t.ocorrencia.periodo      AS semestre,
       tiposaula_t.total_hours(t.ocorrencia.uc.curso, t.ocorrencia.periodo,
       t.ocorrencia.ano_letivo) AS horas_semanais
FROM   tiposaula t
WHERE  t.ocorrencia.periodo IN ( '1S', '2S' )
       AND t.ocorrencia.ano_letivo = '2009/2010'
GROUP  BY t.ocorrencia.uc.curso,
          t.ocorrencia.periodo,
          t.ocorrencia.ano_letivo
ORDER  BY t.ocorrencia.uc.curso,
          t.ocorrencia.periodo;
```

Listing 9: Question e) SQL script



All Rows Fetched: 21 in 0.099 seconds

	CURSO	SEMESTRE	HORAS_SEMANAIS
1	2001	1S	34
2	2001	2S	31
3	2004	1S	21
4	2004	2S	36
5	2006	1S	28
6	2006	2S	26

Figure 6: Question e) answer

4.6 Question f)

“Add a query that illustrates the use of OR extensions.”

We believe that most of the queries we have made so far make use of OR extensions. Nonetheless, we have opted for sorting `ucs` based on their average approval rate of all their `occorencias` object. Since every `occorencias` has a reference to the designated `uc`, then we filter for every `ocorrencia` with reference to `SELF`, since this SELF is the reference to the calling `uc` object. This can be seen on listing 4.

After that it was only a matter of querying the `ucs` table and grouping the average approval rate by their `sigla_uc`.

```sql
SELECT    u.sigla_uc,
          u.get_avg_approval_rate()
FROM      ucs u
GROUP BY u.sigla_uc
ORDER BY u.get_avg_approval_rate() DESC;
```

Listing 10: Question f) SQL script

Unfortunately we could not verify the results as the Oracle DBMS was malfunctioning and did not allow type creation or update.

5 Conclusion

After trying out some of the ORDB capabilities we can say that they are indeed interesting, but also require a whole different mindset towards developing databases.

Although the definition of the data does not change much, associations take a lot more though into as there are many ways of implementing them. That was one of our first problems when creating the model, but after some research and consideration we got a reasonable solution for our model.

Querying the data was quite challenging at first. As we are used to think of data as tables and not objects, most of the times we were not able to get the results in our first attempt. As we got used to it and understood how to use the references better, it became much faster than writing all the conditions for the join operations. Therefore, we believe this is a clear advantage of the object relational paradigm over the traditional relational databases.

Although an important part of objects, we did not find much use for the methods. There are certainly cases where the methods come in handy, but most of the computation is done through the queries. Complex and more powerful methods are harder to develop and do not provided much in return. So we do think this is one of the downsides of the object relational model. We also did not explore the type inheritance features, mostly because they were not appropriate to our model.

Concluding, we saw great potential in these ORDB features with the successful implementation of the model and queries. It does help in some cases, but overall, using objects for database implementation requires more effort on the programmer with little added value in comparison to a relational database. It seems appropriate only for very specific use cases. Nevertheless, this project was quite interesting and greatly improved our skills on object relational capabilities.

References

[1] Oracle. *Oracle® Database Object-Relational Developer's Guide 11g Release 2 (11.2)*. 2011. URL: `https://docs.oracle.com/cd/E24693_01/appdev.11203/e11822/toc.htm`.

[2] Oracle. *PL/SQL User's Guide and Reference 10g Release 1 (10.1)*. 2003. URL: `https://docs.oracle.com/cd/B14117_01/appdev.101/b10807/toc.htm`.