Faculdade de Engenharia da Universidade do Porto

Relatório TBDA

2.º Projeto

Objeto-Relacional - Teaching Service

AssignOR-2

Turma 1

João Augusto dos Santos Lima, up201605314@fe.up.pt

Susana Maria de Sousa Lima, up201603634@fe.up.pt

Índice

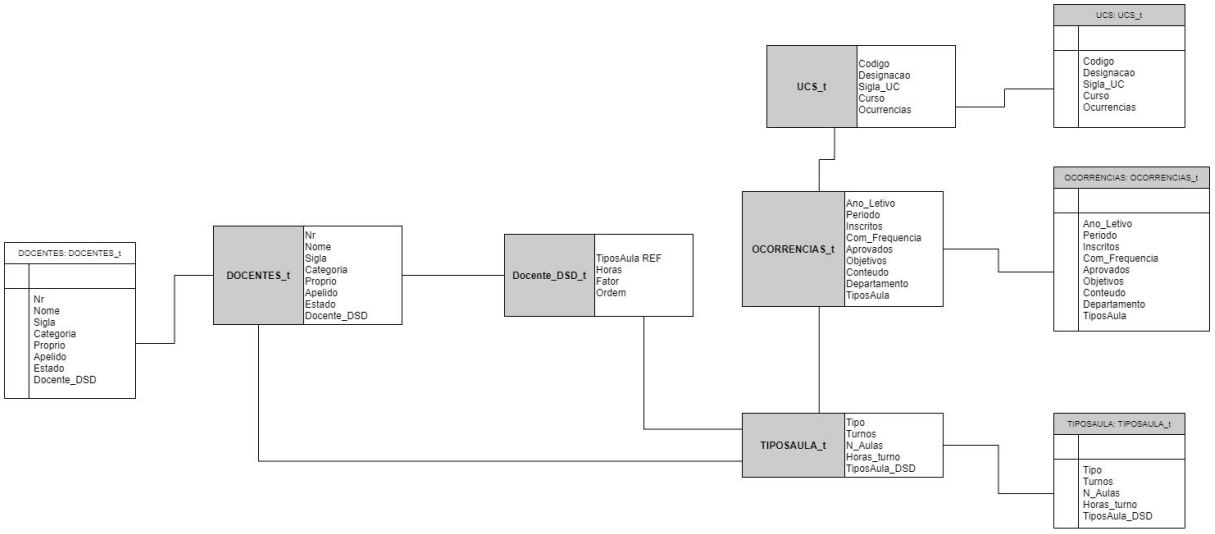
Introdução

O projeto consiste na exploração das possibilidades proporcionadas pelo uso do esquema Objeto-Relacional, em relação ao esquema relacional, tendo em consideração o uso de tipos definidos pelo utilizador, com objetos que combinam dados e estruturas, bem como funções para manipulá-los, tabelas e vetores aninhados, referências a objetos e diferentes métodos.

No desenvolvimento do projeto foram consideradas duas abordagens diferentes para a implementação do modelo Objeto-Relacional, tendo-se optado pela segunda por se revelar mais simples e adequada ao contexto do problema.

Primeira abordagem

Modelo Objeto-Relacional



Image

Criação

```
create or replace type tiposAula_t as object(
   id number(10),
    tipo varchar(2),
    turnos number(2),
    n_aulas number(5),
    horas_turno number(5)
);

create or replace type dsd_t as object(
   nr number(10),
    id number(10),
    horas number(10),
    fator number(5),
    ordem number(5)
);

create or replace type docentes_t as object(
   nr number(10),
    nome varchar(80),
    sigla varchar(10),
```

```
    categoria varchar(50),
    proprio varchar(50),
    apelido varchar(50),
    estado varchar(10)
);

create or replace type tiposAula_dsd_tab_t as table of ref docentes_t;

alter type tiposAula_t add attribute tiposAula_dsd tiposAula_dsd_tab_t
cascade;

create or replace type tiposAula_tab_t as table of tiposAula_t;

create or replace type ocorrencias_t as object(
   codigo varchar(10),
    ano_letivo varchar(10),
    periodo varchar(5),
    inscritos number(8),
    com_frequencia number(8),
    aprovados number(8),
    objetivos varchar(4000),
    conteudo varchar(4000),
    departamento varchar(10),
    tiposAula tiposAula_tab_t
);

create or replace type ocorrencias_tab_t as table of ocorrencias_t;

create or replace type ucs_t as object(
   codigo varchar(10),
    designacao varchar(120),
    sigla_uc varchar(10),
    curso varchar(30),
    ocorrencias ocorrencias_tab_t
);

alter type dsd_t add attribute ucs ref ucs_t cascade;

create or replace type docente_dsd_tab_t as table of dsd_t;


alter type docentes_t add attribute docente_dsd docente_dsd_tab_t
cascade;

create table docentes of docentes_t
    nested table docente_dsd store as docente_dsd_tab;
```

```
create table ucs of ucs_t
    nested table ocorrencias store as ocorrencias_tab
        (nested table tiposAula store as tiposAula_tab
            (nested table tiposAula_dsd store as tiposAula_dsd_tab));
```

Nesta abordagem foram usadas tabelas aninhadas em vários níveis (como pode ser visto no último *create*). No fim, obtém-se apenas duas tabelas, ***docentes*** e ***ucs***, que permitem o acesso à totalidade dos dados, removendo a necessidade de mais tabelas.

Um dos objetivos do modelo apresentado passa pela *nested table* ***docente_dsd_tab*** referenciar a *nested table* ***tiposAula***, no entanto, verificamos que isso não é possível no *sql developer*. Consequentemente, foi necessário alterar a ***docente_dsd_tab*** para, em vez de referenciar a tabela ***tiposAula***, referenciar a tabela ***ucs***. A partir da tabela ***ucs*** é possível aceder à tabela ***tiposAula*** e assim obter os seu dados.

Povoamento

```
insert into ucs (codigo,designacao,sigla_uc,curso,ocorrencias)
select codigo,designacao,sigla_uc,curso,cast(multiset(
    select ocorrencias_t(o.codigo, o.ano_letivo, o.periodo, o.inscritos,
o.com_frequencia, o.aprovados, o.objetivos, o.conteudo,
o.departamento,cast(multiset(
        select tiposAula_t(a.id, a.tipo, a.turnos, a.n_aulas,
a.horas_turno,null)
        from GTD10.xtiposaula a
        where a.ano_letivo = o.ano_letivo and a.periodo = o.periodo and
a.codigo = o.codigo
    ) as tiposAula_tab_t))
    from GTD10.xocorrencias o
    where o.codigo = u.codigo
)as ocorrencias_tab_t)
from GTD10.xucs u;
```

```
insert into docentes
(nr,nome,sigla,categoria,proprio,apelido,estado,docente_dsd)
select nr, nome, sigla, categoria, proprio, apelido,
estado,cast(multiset(
    select dsd_t(nr, id, horas, fator, ordem, (
        select ref(u)
        from ucs u , table(u.ocorrencias) o, table(o.tiposAula) ta
        where ta.id = s.id
```

```
    ))
    from GTD10.xdsd s
    where d.nr = s.nr
) as docente_dsd_tab_t)
from GTD10.xdocentes d;


update table(select o.tiposAula from table(select u.ocorrencias from ucs
u) o) ta
   set ta.tiposAula_dsd= cast(multiset(
       select ref(d)
       from docentes d, table(d.docente_dsd) dsd
       where ta.id = dsd.id
   ) as tiposAula_dsd_tab_t );
```

No povoamento das tabelas na abordagem em questão, encontraram-se dois problemas significativos:

- O segundo *insert* revelou-se muito demorado;
- Não foi possível realizar o *update* necessário para atualizar as referências de ***tiposAula*** para ***docentes***.

O segundo *insert* é extremamente demorado uma vez que, para cada linha de *GTD10.xdsd* (27385 linhas), é necessário aceder a ***ucs*** e comparar o seu *nr* com o *nr* em ***xdsd***. Por sua vez, a tabela ***ucs*** tem 5396 linhas, cada linha contém várias ocorrências e cada ocorrência tem vários tipos de aula. Deste modo, foi possível concluir que futuras interrogações poderiam ter o mesmo problema.

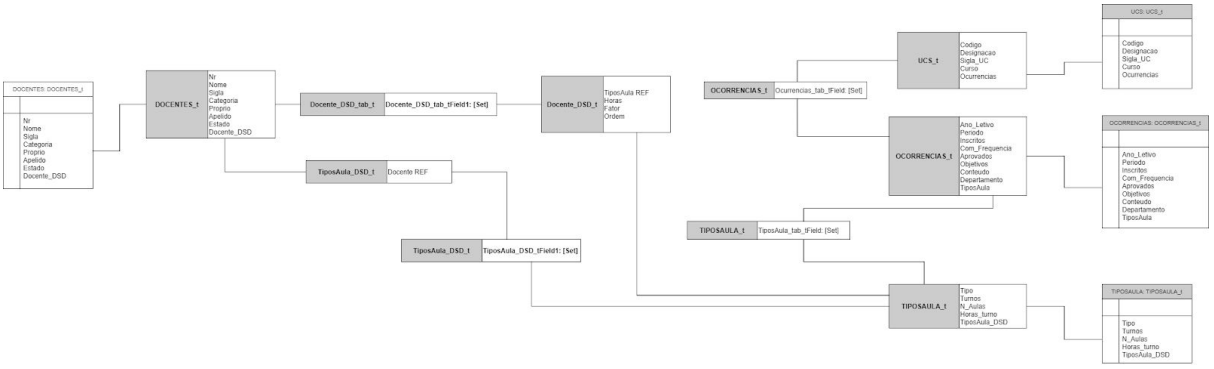
Por outro lado, não foi possível realizar o *update* às referências ***tiposAula***. Foram encontrados vários problemas com o comando *update* e não foi possível puxar para primeiro nível a tabela aninhada ***tiposAula***.

Comentário

Não foi possível implementar esta abordagem pelos problemas referidos anteriormente. No entanto, o grupo concluiu que não seria a abordagem mais benéfica para o problema em questão e para as *queries* propostas. Por conseguinte, optou-se por uma abordagem semelhante que, em vez de recorrer a tabelas aninhadas em vários níveis, utiliza referências a estas mesmas tabelas. As principais diferenças passam pela existência de várias tabelas (à semelhança do modelo relacional) e pelas referências às tabelas que precisam de ser “desreferenciadas” antes de serem utilizadas.

Segunda abordagem

Modelo Objeto-Relacional



Image

Criação

```
create or replace type tiposAula_t as object(
   id number(10),
    tipo varchar(2),
    turnos number(2),
    n_aulas number(5),
    horas_turno number(5),
    map member function getClassHours return number
);

create or replace type dsd_t as object(
   nr number(10),
    id number(10),
    horas number(10),
    fator number(5),
    ordem number(5),
    tiposAula ref tiposAula_t,
    map member function getHorasFator return number
);

create or replace type docente_dsd_tab_t as table of ref dsd_t;

create or replace type docentes_t as object(
   nr number(10),
    nome varchar(80),
    sigla varchar(10),
```

```
    categoria varchar(50),
    proprio varchar(50),
    apelido varchar(50),
    estado varchar(10),
    docente_dsd docente_dsd_tab_t
);

create or replace type tiposAula_dsd_tab_t as table of ref docentes_t;

alter type tiposAula_t add attribute tiposAula_dsd tiposAula_dsd_tab_t
cascade;

create or replace type tiposAula_tab_t as table of ref tiposAula_t;

create or replace type ocorrencias_t as object(
   codigo varchar(10),
    ano_letivo varchar(10),
    periodo varchar(5),
    inscritos number(8),
    com_frequencia number(8),
    aprovados number(8),
    objetivos varchar(4000),
    conteudo varchar(4000),
    departamento varchar(10),
    tiposAula tiposAula_tab_t,
    member function calculatePercentage(a number,b number) return number
);

create or replace type ocorrencias_tab_t as table of ref ocorrencias_t;

create or replace type ucs_t as object(
   codigo varchar(10),
    designacao varchar(120),
    sigla_uc varchar(10),
    curso varchar(30),
    ocorrencias ocorrencias_tab_t
);

create table dsd of dsd_t;

create table tiposAula of tiposAula_t
    nested table tiposAula_dsd store as tiposAula_dsd_tab;

create table docentes of docentes_t
    nested table docente_dsd store as docente_dsd_tab;
```

```
create table ocorrencias of ocorrencias_t
    nested table tiposAula store as tiposAula_tab return as locator;

create table ucs of ucs_t
    nested table ocorrencias store as ocorrencias_tab return as locator;
```

Esta abordagem de criação das tabelas é semelhante à anteriormente apresentada. Por um lado, são criados os tipos ***_tab_t*** como referências às tabelas e, por outro lado, são criadas várias tabelas para a representação do modelo.

Esta abordagem também permite o acesso às tabelas individualmente, obtendo os dados na totalidade. Isto revela-se útil quando se pretende fazer uma *query* a partir de uma tabela "*nested*" (por exemplo: *ocorrências*).

Povoamento

```
insert into docentes (nr, nome, sigla, categoria, proprio, apelido,
estado)
select nr, nome, sigla, categoria, proprio, apelido, estado
from GTD10.xdocentes;

insert into ucs (codigo, designacao, sigla_uc, curso)
select codigo, designacao, sigla_uc, curso
from GTD10.xucs;

insert into ocorrencias (codigo, ano_letivo, periodo, inscritos,
com_frequencia, aprovados, objetivos, conteudo, departamento)
select codigo, ano_letivo, periodo, inscritos, com_frequencia,
aprovados, objetivos, conteudo, departamento
from GTD10.xocorrencias;

insert into tiposAula (id, tipo, turnos, n_aulas, horas_turno)
select id, tipo, turnos, n_aulas, horas_turno
from GTD10.xtiposaula;

insert into dsd (nr, id, horas, fator, ordem, tiposAula)
select nr, xdsd.id, horas, fator, ordem, ref(ta)
from GTD10.xdsd xdsd, tiposAula ta
where ta.id = xdsd.id;

update docentes d
set d.docente_dsd = cast(multiset(
    select ref(x)
```

```
    from dsd x
    where d.nr = x.nr) as docente_dsd_tab_t);

update ucs u
set u.ocorrencias = cast(multiset(
    select ref(o)
    from ocorrencias o
    where o.codigo = u.codigo) as ocorrencias_tab_t);

update ocorrencias o
set o.tiposAula = cast(multiset(
    select ref(ta)
    from tiposAula ta, GTD10.xtiposaula xta
    where ta.id = xta.id
    and o.codigo = xta.codigo
    and o.ano_letivo = xta.ano_letivo
    and o.periodo = xta.periodo
    ) as tiposAula_tab_t);

update tiposAula ta
set ta.tiposAula_dsd = cast(multiset(
    select ref(d)
    from docentes d, dsd x
    where ta.id = x.id
    and d.nr = x.nr
    ) as tiposAula_dsd_tab_t);
```

Para o povoamento da base de dados, além dos *inserts*, são precisos vários *updates* para atribuir as referências para cada tabela. Isto facilita a execução das *queries*, não sendo necessário fazer várias comparações.

Métodos úteis

```
create type body tiposAula_t as
    map member function getClassHours return number is
        begin
        return horas_turno * turnos;
        end getClassHours;
end;


create type body dsd_t as
    map member function getHorasFator return number is
        begin
        return horas*fator;
        end getHorasFator;
end;

create type body ocorrencias_t as
    member function calculatePercentage(a number,b number) return number
is
        begin
        return b*100/a;
        end calculatePercentage;
end;
```

De modo a tirar partido da potencialidade de associar métodos a tipos, e para facilitar algumas das *queries* implementadas, foram criados três métodos. O primeiro método, ***getClassHours,*** é utilizado nas *queries a* e *b* para calcular o total de horas de cada tipo de aulas. Por outro lado, o segundo método, ***getHorasFator,*** é utilizado na pergunta *c* para, através do *fator*, calcular a quantidade fatorizada de horas que um docente dá aulas. Por último, foi adicionado o método ***calculatePercentage***, que calcula a percentagem entre dois números recebidos como parâmetros. Este método é utilizado na *query f* para obter os resultados desejados.

Não surgiu a necessidade de adicionar outros métodos uma vez que não existem outras operações relevantes ou repetidas que as *queries* implementadas realizem.

Comentário

Comparativamente à abordagem anterior, esta abordagem revela-se (além de possível de implementar) mais simples no que diz respeito ao povoamento e mais adequada às *queries* implementadas.

Perguntas e respostas

Pergunta a

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

Formulação

```
select value(ta).tipo as tipo,  sum(value(ta).getClassHours()) as
classHours
from ucs u, table(value(u).ocorrencias) o, table(value(o).tiposAula) ta
where u.curso = 233
and value(o).ano_letivo = '2004/2005'
group by value(ta).tipo;
```

Resultados

	TIPO	CLASSHOURS
1	P	587
2	TP	703
3	T	308

Comentário

Para responder a esta interrogação, selecionam-se os tipos de aula (***tiposAula***) das ocorrências (***ocorrencias***) das unidades curriculares (***ucs***) do ano letivo de 2004/2005 e calcula-se a soma total das *class hours* dos tipos de aula. De modo a obter o valor das *class hours* para cada tipo de aula recorre-se à função ***getClassHours*** definida para o tipo ***tiposAula_t***, que retorna o resultado da multiplicação do número de horas por turno (***horas_turno***) pelo número de turnos (***turnos***).

Pergunta 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?"

Formulação

```
create or replace view requiredHours as
select value(u).codigo as codigo, sum(value(ta).getClassHours()) as
```

```
horas
from ucs u, table(value(u).ocorrencias) o, table(value(o).tiposAula) ta
where value(o).ano_letivo = '2003/2004'
group by value(u).codigo;

create or replace view assignedHours as
select u.codigo as codigo, sum(value(x).horas) as horas
from ucs u, table(value(u).ocorrencias) o, table(value(o).tiposAula) ta,
table(value(ta).tiposAula_dsd) d, table(value(d).docente_dsd) x
where value(o).ano_letivo = '2003/2004'
and value(ta).id = value(x).id
group by u.codigo

select r.codigo, r.horas as requiredHours, a.horas as assignedHours
from requiredHours r, assignedHours a
where r.codigo = a.codigo and r.horas <> a.horas;
```

Resultados

	CODIGO	REQUIREDHOURS	ASSIGNEDHOURS
1	EEC5020	1	0
2	MGI1204	5	3
3	EMG2204	7	8
4	EIC5101	4	6
5	MEA208	5	6
6	EM631	10	9
7	EMG4105	5	6
8	MTM109	3	4
9	EIC5202	48	30
10	MEM170	2	4
11	MEM185	2	3
12	EMG5205	5	6
13	MEST208	2	3
14	MEST209	2	3
15	MRSC1104	1	2
16	MEEC1076	3	4
17	MEEC1075	3	4
18	EEC4277	4	1
19	EM613	4	6
20	EEC3161	13	14
21	EC4104	20	22
22	EMG2202	7	6

	CODIGO	REQUIREDHOURS	ASSIGNEDHOURS
22	EMG2202	7	6
23	MEA400	8	9
24	MEEC1085	3	4
25	EEC5180	4	5
26	MTM100	6	5
27	EEC3265	13	14
28	MEEC1055	3	4
29	EEC4248	9	10
30	MEEC2103	3	4
31	MFAMF1101	2	3
32	EI1100	4	3
33	MRSC1204	3	4
34	MMCCE1214	3	4
35	MEB100	3	2
36	EEC5145	4	6
37	MRPE1204	2	3
38	EEC5040	11	0
39	EEC4161	15	16
40	EMG1103	10	11
41	MMI1201	3	4
42	MMI1102	3	4

	CODIGO	REQUIREDHOURS	ASSIGNEDHOURS
43	EMG2001	6	3
44	EMG4103	2	5
45	MGI1210	3	2
46	MEA301	3	2
47	EIC4213	1	2
48	MEEC1051	3	4
49	EMG5204	5	6
50	EEC5278	5	6
51	EQ101	28	8
52	EI1101	4	5
53	EI1201	3	4
54	EI1206	4	1
55	EQ502	12	0
56	EM128	24	25
57	EM229	34	28
58	MMI1204	3	13
59	MMI1101	3	4
60	EIC5102	7	4
61	MEA406	3	2
62	MMCCE1205	3	4
63	GEI205	4	5
64	EEC4290	4	6

	CODIGO	REQUIREDHOURS	ASSIGNEDHOURS
64	EEC4290	4	6
65	MEEC1054	3	1
66	EEC4145	8	9
67	MEEC1088	3	4
68	MGI1203	3	4
69	EEC5127	4	5
70	MEST314	2	4
71	EQ103	17	18
72	MMCCE1213	3	4
73	EC1108	60	65
74	EEC5277	4	6
75	EEC4244	4	6
76	MEB200	3	4
77	EMG1202	7	8
78	MMI1103	3	4
79	MEA407	3	2
80	EEC4250	5	7
81	EEC5250	4	6
82	MEEC1089	3	4
83	EM337	36	33
84	EEC5060	11	0

	CODIGO	REQUIREDHOURS	ASSIGNEDHOURS
85	MEST211	2	3
86	EEC4101	18	19
87	EMM527	40	0
88	EQ109	17	18
89	MRSC1201	5	3
90	EMG3105	5	6
91	MEAM1200	3	4
92	EI1202	4	5
93	EI1203	4	5
94	EM114	39	26
95	MEEC1050	3	4
96	MEEC2102	3	4
97	EI1208	3	4
98	EM335	31	26
99	EIC4101	24	14
100	EEC4273	5	6
101	MEA311	5	6
102	MEEC2094	3	4
103	EEC5021	16	17

	CODIGO	REQUIREDHOURS	ASSIGNEDHOURS
104	EMG4202	5	6
105	EIC4212	4	5
106	MAIC1101	3	2
107	MAIC1107	3	4
108	MEST115	2	3
109	EEC5243	4	6
110	EI1209	4	5
111	GEI210	4	2
112	MMCCE1102	3	4
113	GEI213	9	8
114	MEA306	3	2
115	MEEC2095	3	4
116	MEEC1078	3	4
117	MEEC1072	3	4
118	MEB107	3	5
119	MEM176	2	3
120	EMG1001	12	6
121	EIC4217	4	5
122	EIC5120	4	6
123	EC5183	6	3
124	MEST302	2	3
125	EMG3102	7	5

Comentário

De modo a simplificar a *query* para esta questão, foram definidas duas *views.* A primeira, ***requiredHours***, permite obter o número de *class hours* requeridas para cada unidade curricular no ano de 2003/2004. Por outro lado, a segunda *view* criada, ***assignedHours,*** permite obter o número de horas atribuídas para cada unidade curricular no ano em questão. Por fim, a *query* em si, apenas compara os valores das horas das duas *views* para a mesma uc e retorna as que diferem.

Pergunta 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."

Formulação

```
create or replace view horasPorTipo as
select value(d).nr as nr, value(d).nome as nome, value(ta).tipo as tipo,
sum(value(x).getHorasFator()) as horas
from ocorrencias o, table(value(o).tiposAula) ta,
table(value(ta).tiposAula_dsd) d, table(value(d).docente_dsd) x
where value(o).ano_letivo = '2003/2004'
and value(ta).id = value(x).id
group by value(d).nr, value(d).nome, value(ta).tipo
order by value(d).nr;

create or replace view maxHorasPorTipo as
select  ht.tipo as tipo, max(ht.horas) as maxHoras
from horasPorTipo ht
group by ht.tipo;

select ht.nr, ht.nome, ht.tipo, ht.horas as horasFator
from horasPorTipo ht, maxHorasPorTipo mh
where ht.tipo  = mh.tipo
and ht.horas = mh.maxHoras;
```

Resultados

	NR	NOME	TIPO	HORASFATOR
1	207638	Fernando Francisco Machado Veloso Gomes	T	34
2	208187	António Almerindo Pinheiro Vieira	P	30
3	210006	João Carlos Pascoal de Faria	OT	4
4	249564	Cecília do Carmo Ferreira da Silva	TP	26

Comentário

À semelhança da pergunta anterior, também nesta interrogação foram criadas duas *views* com o objetivo de simplificar a implementação da *query* em questão. A primeira, ***horasPorTipo,*** retorna a soma das *horasFator* para cada docente e para cada tipo de aula no ano letivo 2003/2004. Para obter o valor das *horasFator* para um docente, recorre-se à função ***getHorasFator*** definida no tipo ***dsd_t***. Por sua vez, a segunda *view* criada, ***maxHorasPorTipo,*** determina, para cada tipo de aula, o maior valor de *horasFator*. Por fim,

a *query* apenas seleciona o docente com o valor correspondente de *horasFator* máximo, para cada tipo, obtido na *view* ***maxHorasPorTipo.***

Pergunta d

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

Formulação

```
select value(o).ano_letivo as ano_letivo, value(d).categoria as
categoria,  round(avg(value(x).horas),3) as avgHours
from ocorrencias o, table(value(o).tiposAula) ta,
table(value(ta).tiposAula_dsd) d, table(value(d).docente_dsd) x
where regexp_like (value(o).ano_letivo, '^200[1-4]')
and value(ta).id = value(x).id
and value(d).categoria is not null
group by value(o).ano_letivo, value(d).categoria
order by value(o).ano_letivo,  value(d).categoria
```

Resultados

	ANO_LETIVO	CATEGORIA	AVGHOURS
1	2001/2002	103	2.25
2	2001/2002	107	2.258
3	2001/2002	110	2.62
4	2001/2002	11005	5.75
5	2001/2002	111	2.688
6	2001/2002	112	1.5
7	2001/2002	116	3.283
8	2001/2002	117	3.673
9	2001/2002	119	4.755
10	2001/2002	120	3.869
11	2001/2002	122	4.5
12	2001/2002	124	4
13	2001/2002	144	2
14	2001/2002	19995	1.667
15	2001/2002	19997	2.231
16	2001/2002	19999	3.081
17	2001/2002	374	5
18	2001/2002	519	3.9
19	2001/2002	520	6.333
20	2001/2002	565	2
21	2002/2003	103	2

	ANO_LETIVO	CATEGORIA	AVGHOURS
25	2002/2003	11005	4
26	2002/2003	11007	4
27	2002/2003	111	2.722
28	2002/2003	112	5.2
29	2002/2003	116	3.231
30	2002/2003	117	3.404
31	2002/2003	119	4.386
32	2002/2003	120	3.831
33	2002/2003	122	4.5
34	2002/2003	125	4
35	2002/2003	144	2.111
36	2002/2003	19995	1.857
37	2002/2003	19997	2.375
38	2002/2003	19999	2.263
39	2002/2003	374	3.75
40	2002/2003	519	3.333
41	2002/2003	520	6.5
42	2002/2003	565	2
43	2002/2003	903	2
44	2003/2004	10108	2
45	2003/2004	10119	2

	ANO_LETIVO	CATEGORIA	AVGHOURS
46	2003/2004	103	2.167
47	2003/2004	107	2.312
48	2003/2004	110	2.464
49	2003/2004	11005	4.667
50	2003/2004	11007	0
51	2003/2004	111	2.625
52	2003/2004	112	4.125
53	2003/2004	116	3.051
54	2003/2004	117	3.366
55	2003/2004	119	5.104
56	2003/2004	120	3.57
57	2003/2004	122	4.5
58	2003/2004	144	1.833
59	2003/2004	19995	2.2
60	2003/2004	19997	1.778
61	2003/2004	19999	2.228
62	2003/2004	374	3
63	2003/2004	519	3.143
64	2003/2004	565	2.167

	ANO_LETIVO	CATEGORIA	AVGHOURS
65	2003/2004	903	2.25
66	2004/2005	10108	2
67	2004/2005	10119	3
68	2004/2005	103	2.143
69	2004/2005	107	2.466
70	2004/2005	110	2.37
71	2004/2005	11007	2
72	2004/2005	111	4.438
73	2004/2005	112	3.143
74	2004/2005	116	3.034
75	2004/2005	117	3.081
76	2004/2005	119	6
77	2004/2005	120	3.785
78	2004/2005	122	3.667
79	2004/2005	144	1.706
80	2004/2005	19995	4.333
81	2004/2005	19997	2.143
82	2004/2005	19999	2.169
83	2004/2005	374	8
84	2004/2005	519	3.071
85	2004/2005	565	4
86	2004/2005	903	1.667

Comentário

Nesta query apenas se calcula o número de horas médio dos docentes para cada categoria nos anos compreendidos entre 2001/2002 e 2004/2005.

Pergunta 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.”

Formulação

```
select value(o).ano_letivo as ano_letivo, value(o).periodo as periodo,
sum(value(ta).horas_turno) as horas
from ocorrencias o, table(value(o).tiposAula) ta
where value(ta).n_aulas is not null and value(ta).turnos is not null and
value(o).periodo like '%S'
group by value(o).ano_letivo, value(o).periodo
order by value(o).ano_letivo, value(o).periodo;
```

Resultados

	ANO_LETIVO	PERIODO	HORAS
1	1996/1997	1S	6
2	1996/1997	2S	6
3	2002/2003	1S	1438
4	2002/2003	2S	1317
5	2003/2004	1S	1431
6	2003/2004	2S	1402
7	2004/2005	1S	1723
8	2004/2005	2S	1483
9	2005/2006	1S	1745
10	2005/2006	2S	1510
11	2006/2007	1S	1538
12	2006/2007	2S	979

Comentário

Para responder a esta interrogação apenas foram consideradas, para todos os anos letivos, as ocorrências cujos períodos correspondem a semestres, ou seja, que tenham o valor de “*1S*” ou “*2S*”. A partir destas ocorrências selecionam-se os tipos de aulas correspondentes e calcula-se a soma de todas as horas por turno (***horas_turno***), uma vez que se assume que o estudante apenas faria um dos turnos.

Pergunta f

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

Uma vez que foi dada liberdade para escolher uma questão que ilustre o uso de extensões OR, o grupo decidiu obter a percentagem de alunos com frequência e a percentagem de alunos aprovados no ano letivo de 2003/2004 em qualquer cadeira cujo código comece com “CI” (com qualquer número de 3 casas à frente).

Formulação

```
select value(u).codigo as codigo,value(o).ano_letivo as anoLetivo,
round(value(o).calculatePercentage(value(o).inscritos,value(o).com_frequ
encia),3) as PercentagemComFrequencia
,round(value(o).calculatePercentage(value(o).inscritos,value(o).aprovado
s),3) as PercentagemAprovados
from ucs u, table(value(u).ocorrencias) o
where value(o).inscritos is not null and value(o).com_frequencia is not
null and value(o).aprovados is not null
and value(o).ano_letivo = '2003/2004'
and regexp_like (value(u).codigo, '^CI[0-9][0-9][0-9]')
order by value(u).codigo,value(o).ano_letivo;
```

Resultados

	CODIGO	ANOLETIVO	PERCENTAGEMCOMFREQUENCIA	PERCENTAGEMAPROVADOS
1	CI001	2003/2004	80,392	60,784
2	CI002	2003/2004	69,811	54,717
3	CI003	2003/2004	81,633	77,551
4	CI004	2003/2004	76,596	72,34
5	CI005	2003/2004	66	60
6	CI006	2003/2004	70,909	63,636
7	CI007	2003/2004	68,627	60,784
8	CI008	2003/2004	68,421	52,632
9	CI009	2003/2004	69,388	65,306
10	CI010	2003/2004	61,538	51,923
11	CI011	2003/2004	88,462	88,462
12	CI012	2003/2004	96	96
13	CI013	2003/2004	92,308	88,462
14	CI014	2003/2004	100	88,889
15	CI015	2003/2004	84	64
16	CI016	2003/2004	96,154	96,154
17	CI017	2003/2004	80,769	80,769
18	CI018	2003/2004	96,154	92,308
19	CI019	2003/2004	92,308	88,462
20	CI020	2003/2004	92,593	88,889
21	CI021	2003/2004	88,235	70,588
22	CI022	2003/2004	100	100
23	CI023	2003/2004	100	100
24	CI024	2003/2004	100	100
25	CI025	2003/2004	100	100
26	CI026	2003/2004	23,529	23,529
27	CI027	2003/2004	100	100
28	CI028	2003/2004	100	94,118
29	CI036	2003/2004	100	100
30	CI037	2003/2004	64,706	64,706
31	CI038	2003/2004	100	100

Comentário

A *query* em questão utiliza o método ***calculatePercentage*** definido no tipo ***ocorrencias_t*****,** que recebe como argumentos dois números e calcula uma percentagem entre eles. Este método é utilizado para calcular as percentagens de alunos com frequência e de alunos aprovados. A *query* em si é bastante simples, apenas filtra as unidades curriculares com o código desejado e as ocorrências com o ano em questão.

Conclusão

Este trabalho permite concluir que as extensões Objeto-Relacional, se corretamente utilizadas, podem ter bastante potencial, facilitando o processo de implementação do modelo.

No entanto, é de referir, que o grupo deparou-se com algumas dificuldades na implementação de um modelo inicial, que simultaneamente, tirasse benefício das potencialidades proporcionadas por esta extensão, e fosse simples e intuitivo de modo a facilitar a implementação das *queries* propostas.

No final, considerou-se o modelo definido adequado ao problema em questão, sendo que a implementação do mesmo e das *queries* propostas permitiu aprofundar o conhecimento, adquirido nas aulas, deste tipo de representação.