NoSQL Assignment

Lab assignment no. 3

Version 3 – Student applications

Integrated Master in Informatics and Computing Engineering

Database Technology

Group elements:
André Pires - 201207106 - ei12058@fe.up.pt
João Bandeira - 201200615 - ei12022@fe.up.pt

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

June 3, 2016

Contents

1 Existing database

Database schema presented in the lab assignment[1].
Table lics (list of programs)

- codigo NUMBER(4) NOT NULL primary key,
- sigla VARCHAR2(5) NOT NULL,
- nome VARCHAR2(80) NOT NULL

Table cands (all the applications)

- bi VARCHAR2(12) NOT NULL,
- curso NUMBER(4) NOT NULL references lics,
- ano_lectivo NUMBER(4) NOT NULL,
- resultado VARCHAR2(1), – C – accepted; E – excluded; S – waiting list
- media NUMBER(5,2),
- PRIMARY KEY (bi, curso, ano_lectivo)

Table alus (the annual enrolments) [1]

- numero VARCHAR2(9) primary key,
- bi VARCHAR2(12) NOT NULL,
- curso NUMBER(4) references lics,
- a_lect_matricula NUMBER(4) NOT NULL,
- estado VARCHAR2(2), – F – following; C – concluded
- a_lect_conclusao NUMBER(4),
- med_final NUMBER(4,2) – final grade
- FOREIGN KEY (bi, curso, a_lect_matricula) REFERENCES cands

Table anos (list of years with enrolments)

- ano NUMBER(4)

[1]Notice that if a person changes between programs, he will get two different student number, but the same personal id (BI). Academic years are represented by the first year, i. e., 1999 means '1999/2000'.

2 Mongo document model

2.1 Table lics (list of programs)

```
{
    codigo: {type: Number, required: true, unique: true},
    sigla: {type: String, required: true, maxlength: 5},
    nome: {type: String, required: true, maxlength: 80}
}
```

2.2 Table cands

```
{
    bi: {type: String, required: true, maxlength: 12},
    curso: {type: Number, required: true},
    ano_lectivo: {type: Number, required: true},
    resultado: {type: String, required: true, maxlength: 1},
    media: {type: Number, required: true}
}
```

We thought of adding the whole course (see lics in 2.1) in this collection but we decided not to, because the benefit which this redundancy would bring wouldn't make up for the extra space in memory.

2.3 Table alus (the annual enrolments)

```
{
   numero: {type: String, required: true, maxlength: 9,unique:
       true},
    bi: {type: String, required: true, maxlength: 12},
    curso: {type: Number},
    a_lect_matricula: {type: Number, required: true},
    estado: {type: String, maxlength: 2},
    a_lect_conclusao: {type: Number},
    med_cand: {type: Number},
    med_final: {type: Number}
}
```

We added *med_cand* to this collection because it was the only information about the *cand table* (see 2.2) which was not available in this collection.

2.4 Table anos (list of years with enrolments)

```
{
   ano: {type: Number}
}
```

3 Database migration

3.1 Table Lics (list of programs)

3.1.1 PL/SQL - Generating XML data

```
PROCEDURE export_lics_xml AS
BEGIN
  htp.p('<?xml version="1.0" encoding="UTF-8"?>');
  htp.p('<lics>');
  for c in (select * from gtd2.lics) loop
    htp.p('<lic><cod>'||c.codigo||'</cod><sigla>'||c.sigla||'</sigla>
      <nome>'||c.nome||'</nome></lic>');
  end loop;
  htp.p('</lics>');
END export_lics_xml;
```

3.1.2 XSL - From XML to Mongo insert script

```
<xsl:stylesheet version="2.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="text" omit-xml-declaration="yes" indent="no"/>

   <xsl:template match="lics">
      <xsl:text>db.lics.insert([</xsl:text>
         <xsl:apply-templates select="//lic" />
      <xsl:text>]);</xsl:text>
   </xsl:template>

   <xsl:template match="lic">
      <xsl:text>{codigo: </xsl:text><xsl:value-of select="./cod"
      /><xsl:text>, </xsl:text>
      <xsl:text>sigla: "</xsl:text><xsl:value-of select="./sigla"
      /><xsl:text>", </xsl:text>
      <xsl:text>nome: "</xsl:text><xsl:value-of select="./nome"
      /><xsl:text>"},</xsl:text>
   </xsl:template>

</xsl:stylesheet>
```

3.2 Table Cands (candidates)

3.2.1 PL/SQL - Generating XML data

```
PROCEDURE export_cands_xml AS
BEGIN
  htp.p('<?xml version="1.0" encoding="UTF-8"?>');
  htp.p('<cands>');
  for c in (select * from gtd2.cands) loop
    htp.p('<cand><bi>'||c.bi||'</bi><curso>'||c.curso||'</curso>
      <ano_lectivo>'||c.ano_lectivo||'</ano_lectivo>
      <resultado>'||c.resultado||'</resultado>
      <media>'||c.media||'</media></cand>');
  end loop;
  htp.p('</cands>');
END export_cands_xml;
```

3.2.2 XSL - From XML to Mongo insert script

```
<xsl:stylesheet version="2.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="text" omit-xml-declaration="yes" indent="no"/>

   <xsl:template match="cands">
      <xsl:text>db.cands.insert([</xsl:text>
         <xsl:apply-templates select="//cand" />
      <xsl:text>]);</xsl:text>
   </xsl:template>

   <xsl:template match="cand">
      <xsl:text>{bi: "</xsl:text><xsl:value-of select="./bi"
      /><xsl:text>", </xsl:text>
      <xsl:text>curso: </xsl:text>
         <xsl:choose>
            <xsl:when test="./curso/text() != '' ">
               <xsl:value-of select="./curso" />
            </xsl:when>
            <xsl:otherwise>
               <xsl:text>null</xsl:text>
            </xsl:otherwise>
         </xsl:choose>
         <xsl:text>, </xsl:text>
      <xsl:text>ano_lectivo: </xsl:text><xsl:value-of
      select="./ano_lectivo" /><xsl:text>, </xsl:text>
      <xsl:text>resultado: "</xsl:text><xsl:value-of
      select="./resultado" /><xsl:text>", </xsl:text>
      <xsl:text>media: </xsl:text>
         <xsl:choose>
            <xsl:when test="./media/text() != ''">
               <xsl:value-of select="./media" />
            </xsl:when>
            <xsl:otherwise>
               <xsl:text>null</xsl:text>
            </xsl:otherwise>
         </xsl:choose>
      <xsl:text>}, </xsl:text>
   </xsl:template>

</xsl:stylesheet>
```

3.3 Table alus (students)

3.3.1 PL/SQL - Generating XML data

```
PROCEDURE export_alus_xml AS
BEGIN
  htp.p('<?xml version="1.0" encoding="UTF-8"?>');
  htp.p('<alunos>');
  for c in (select a.NUMERO, a.BI, a.CURSO, a.A_LECT_MATRICULA,
      a.ESTADO, a.A_LECT_CONCLUSAO, a.MED_FINAL,
      ca.RESULTADO, ca.MEDIA
    from gtd2.alus a, gtd2.cands ca
    where a.bi = ca.bi and a.curso = ca.curso and a.a_lect_matricula =
        ca.ano_lectivo and ca.resultado = 'C') loop
    htp.p('<aluno><numero>'||c.numero||'</numero><bi>'||c.bi||'</bi>
      <curso>'||c.curso||'</curso>
      <a_lect_matricula>'||c.a_lect_matricula||'</a_lect_matricula>
      <estado>'||c.estado||'</estado>
      <a_lect_conclusao>'||c.a_lect_conclusao||'</a_lect_conclusao>
      <med_final>'||c.med_final||'</med_final>
      <media_candidatura>'||c.media||'</media_candidatura>'||'</aluno>');
  end loop;
  htp.p('</alunos>');
END export_alus_xml;
```

3.3.2 XSL - From XML to Mongo insert script

```
<xsl:stylesheet version="2.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="text" omit-xml-declaration="yes" indent="no"/>

   <xsl:template match="alunos">
      <xsl:text>db.alus.insert([</xsl:text>
         <xsl:apply-templates select="//aluno" />
      <xsl:text>]);</xsl:text>
   </xsl:template>

   <xsl:template match="aluno">
      <xsl:text>{numero: </xsl:text><xsl:value-of select="./numero"
      /><xsl:text>, </xsl:text>
      <xsl:text>bi: "</xsl:text><xsl:value-of select="./bi"
      /><xsl:text>", </xsl:text>
      <xsl:text>curso: </xsl:text>
         <xsl:choose>
            <xsl:when test="./curso/text() != '' ">
               <xsl:value-of select="./curso" />
            </xsl:when>
            <xsl:otherwise>
               <xsl:text>null</xsl:text>
            </xsl:otherwise>
         </xsl:choose>
         <xsl:text>, </xsl:text>

      <xsl:text>a_lect_matricula: </xsl:text>
         <xsl:choose>
```

```xml
        <xsl:when test="./a_lect_matricula/text() != '' ">
          <xsl:value-of select="./a_lect_matricula" />
        </xsl:when>
        <xsl:otherwise>
          <xsl:text>null</xsl:text>
        </xsl:otherwise>
      </xsl:choose>
      <xsl:text>, </xsl:text>

    <xsl:text>estado: "</xsl:text><xsl:value-of select="./estado" /><xsl:text>", </xsl:text>

    <xsl:text>a_lect_conclusao: </xsl:text>
      <xsl:choose>
        <xsl:when test="./a_lect_conclusao/text() != '' ">
          <xsl:value-of select="./a_lect_conclusao" />
        </xsl:when>
        <xsl:otherwise>
          <xsl:text>null</xsl:text>
        </xsl:otherwise>
      </xsl:choose>
      <xsl:text>, </xsl:text>

    <xsl:text>med_cand: </xsl:text>
      <xsl:choose>
        <xsl:when test="./media_candidatura/text() != '' ">
          <xsl:value-of select="./media_candidatura" />
        </xsl:when>
        <xsl:otherwise>
          <xsl:text>null</xsl:text>
        </xsl:otherwise>
      </xsl:choose>
      <xsl:text>, </xsl:text>

    <xsl:text>med_final: </xsl:text>
      <xsl:choose>
        <xsl:when test="./med_final/text() != '' ">
          <xsl:value-of select="./med_final" />
        </xsl:when>
        <xsl:otherwise>
          <xsl:text>null</xsl:text>
        </xsl:otherwise>
      </xsl:choose>
      <xsl:text>}, </xsl:text>

  </xsl:template>

</xsl:stylesheet>
```

4 Database queries

4.1 Query 1

Calculate the total number of students enrolled in each program, in each year, after 1991.

```sql
select a.A_LECT_MATRICULA year, l.nome, count(*) as numAlus
from alus a, lics l
where a.curso = l.codigo
and a.A_LECT_MATRICULA > 1991
group by l.nome, a.a_lect_matricula
order by year;
```

```
db.alus.aggregate([
   {$match : { a_lect_matricula : {$gt: 1991}}},
   {$group : {_id : {curso: "$curso", ano: "$a_lect_matricula"}, num :
       {$sum : 1}}},
   { $sort: { "_id.ano": 1 } }
])
```

4.2 Query 2

Obtain the BI and the student number of the students with a final grade (med_final) higher than the application grade (media).

```sql
select a.bi, a.numero
from alus a, cands c
where a.bi = c.bi
and a.med_final > c.media;
```

```
db.alus.aggregate( [
    { $project: {
        bi: 1,
        numero: 1,
        med_final: 1,
        med_cand: 1,
        eq: { $cond: [ { $and : [
                          {$ne : [ "$med_cand" , null ]},
                          {$ne : [ "$med_final" , null ]},
                          {$gt: [ "$med_final", "$med_cand" ]}
                       ]}, 1, 0 ] }
    } },
    { $match: { eq: 1 } }
] )
```

4.3 Query 3

Find the average of the final grades of all the students finishing their program in a certain number of years, 5 years, 6 years, . . .

```sql
select (a_lect_conclusao - a_lect_matricula) as diff, avg(med_final)
from alus
where (a_lect_conclusao - a_lect_matricula) > 4
group by (a_lect_conclusao - a_lect_matricula)
order by (a_lect_conclusao - a_lect_matricula);
```

```
db.alus.aggregate([
    { $match: { a_lect_conclusao: { $ne: 'null' } }},
    { $project: {
        "time": { $subtract: [ "$a_lect_conclusao", "$a_lect_matricula"
             ] },
        "med_final" : "$med_final"
    }},
      { $match: { "time": { $gt: 4 } }},
    {$group : {_id : {diff : "$time"}, avg : { $avg : "$med_final"} }},
    {$sort :{ "_id.diff": 1 } }
])
```

5 Mongo and Oracle comparison

Although Oracle SQL and MongoDB are both implementations of database technologies, they have little in common. While the first is an implementation of standard SQL, the latter implements NoSQL. Until this assignment, the group had only had contact with SQL, so some of the tasks were requested took longer than expected to complete because the group wasn't familiar with either the NoSQL paradigm, or the MongoDB syntax.

Regarding the data size, both implementations have its advantages. The main advantage of Oracle (and SQL in general) is the fact that it allows joining tables easily, so that the information on several tables can be combined. In Mongo, without the possibility of joining collections, some workarounds have to be made. One of the possible workarounds is to create some redundancy of data, by storing some data fields in more than one collection, resulting in more space used. On the other hand, in Mongo there are also ways of saving space, for example, for attributes that have a *null* value, one can simply not store that attribute in the collection.

About the processing time, in the queries that were developed in this assignment, there were no visible differences between both technologies. Possibly if the datasets were larger, there would be some noticeable variations.

The syntax of both technologies, as said before, is also one of the big differences between them. SQL provides a simpler syntax, with not many variations (mostly follows the *SELECT a, b, ... FROM c, d, ... WHERE ...* pattern). Mongo provides more than one syntax to query the data (*find*, *aggregate*, ...), so its syntax is more difficult to learn, because not only the combination of different methods allows numerous ways of making the same query, but also different methods have different syntax for the same operations. Writing queries in Mongo would be easier if the syntax was more uniform.

References

[1] Gabriel David. Nosql assignment. Lab pdf, `https://moodle.up.pt/pluginfile.php/122884/mod_resource/content/0/67-Assign3-Mongo-v3-en.pdf`, 2016.