Grupo Nº 26

TÉCNICO LISBOA

Base de Dados

1.º Semestre 2015/2016

Projecto Bloco de Notas

Relatório de Projecto Bloco de Notas

João Marçal Nº78471
João Alves Nº79155
José Semedo Nº78294

Índice

1 Queries

1.1 Quais são os utilizadores que falharam o login mais vezes do que tiveram sucesso?

```
SELECT DISTINCT U.userid
FROM utilizador U,
     login L
WHERE
    (SELECT COUNT(sucesso)
     FROM login L
     WHERE sucesso = 0
       AND U.userid = L.userid) >
    (SELECT COUNT (sucesso)
     FROM login L
     WHERE sucesso = 1
       AND U.userid = L.userid)
ORDER BY userid;
```

1.2 Quais são os registos que aparecem em todas as páginas de um utilizador?

```
SELECT u_rp_nbp.userid, u_rp_nbp.regid
FROM
(SELECT u.userid,
          rp.regid,
          COUNT(rp.pageid) AS nbp
   FROM utilizador u, reg_pag rp,
        pagina p,
        registo r,
        tipo_registo tr
   WHERE u.userid = rp.userid
     AND u.userid = p.userid
     AND u.userid = r.userid
     AND u.userid = tr.userid
     AND rp.pageid = p.pagecounter
     AND r.typecounter = tr.typecnt
     AND tr.typecnt = rp.typeid
     AND rp.regid = r.regcounter
     AND rp.typeid = r.typecounter
     AND r.ativo = 1
     AND rp.ativa = 1
     AND p.ativa = 1
     AND tr.ativo = 1
   GROUP BY(rp.regid)
   ORDER BY u.userid) AS u_rp_nbp,
  (SELECT u.userid, COUNT(p.pagecounter) AS nbp
   FROM utilizador u, pagina p
   WHERE u.userid = p.userid
     AND p.ativa = 1
   GROUP BY(u.userid)) AS u_nbp
WHERE u_nbp.userid = u_rp_nbp.userid
  AND u_nbp.nbp = u_rp_nbp.nbp;
```

1.3 Quais os utilizadores que tem o maior número médio de registos por página?

Quando interpretamos esta query achamos que era para fazer para cada utilizador a média dos registos em página pelas páginas desse utilizador, mais tarde chamaram-nos a atenção que era a média dos registos pelas páginas acabamos por deixar a ultima no relatório mas deixamos aqui uma nota para o docente que caso nos tenhamos enganado implementamos ambas as versões.

```
SELECT u_rpp.userid
FROM
  (SELECT u_nbpag.userid,
          u_nbreg.nb_reg/u_nbpag.nb_pag AS reg_p_pag
   FROM
     (SELECT u.userid, COUNT(p.pagecounter) AS nb_pag
      FROM utilizador u,
           pagina p
      WHERE u.userid = p.userid
        AND p.ativa = 1
      GROUP BY(u.userid)) AS u_nbpag,
     (SELECT u.userid, COUNT(r.regcounter) AS nb_reg
      FROM utilizador u,
           registo r,
           tipo_registo tr
      WHERE r.userid = u.userid
        AND tr.userid = u.userid
        AND tr.typecnt = r.typecounter
        AND r.ativo = 1
        AND tr.ativo = 1
      GROUP BY u.userid) AS u_nbreg
   WHERE u_nbpag.userid = u_nbreg.userid
   GROUP BY u_nbpag.userid) AS u_rpp
WHERE u_rpp.reg_p_pag =
    (SELECT MAX(u_rpp.reg_p_pag)
     FROM
       (SELECT u_nbpag.userid,
               u_nbreg.nb_reg/u_nbpag.nb_pag AS reg_p_pag
```

```
FROM
  (SELECT u.userid,
          COUNT(p.pagecounter) AS nb_pag
   FROM utilizador u,
        pagina p
   WHERE u.userid = p.userid
     AND p.ativa = 1
   GROUP BY(u.userid)) AS u_nbpag,

  (SELECT u.userid, COUNT(r.regcounter) AS nb_reg
   FROM utilizador u,
        registo r,
        tipo_registo tr
   WHERE r.userid = u.userid
     AND tr.userid = u.userid
     AND tr.typecnt = r.typecounter
     AND r.ativo = 1
     AND tr.ativo = 1
   GROUP BY u.userid) AS u_nbreg
WHERE u_nbpag.userid = u_nbreg.userid
GROUP BY u_nbpag.userid) AS u_rpp);
```

1.4 Quais os utilizadores que, em todas as suas páginas, têm registos de todos os tipos de registos que criaram?

```
SELECT u_nbpg.userid

FROM

/*Query com utilizador e numero total de paginas que tenham todos os t
ipos de reg do utilizador em si*/

  (SELECT u_p_nbtr.userid, COUNT(u_p_nbtr.pagecounter) AS nbpg

   FROM

       /*Query com utilizador paginas e quantos tipos de registo

       essa pagina tem*/

     (SELECT u.userid, p.pagecounter, COUNT(DISTINCT rp.typeid) AS nbt
r

      FROM reg_pag rp, utilizador u, pagina p, registo r, tipo_registo
tr

      WHERE rp.userid = u.userid

        AND p.userid = u.userid

        AND r.userid = u.userid

        AND tr.userid = u.userid

        AND tr.typecnt = r.typecounter

        AND tr.typecnt = rp.typeid

        AND rp.pageid = p.pagecounter

        AND rp.regid = r.regcounter

        AND rp.typeid = r.typecounter

        AND r.ativo = 1

        AND tr.ativo = 1

        AND p.ativa = 1

        AND rp.ativa = 1

      GROUP BY(p.pagecounter)

      ORDER BY u.userid) AS u_p_nbtr,

       /*Query com utilizador e quantos tipos de registo ele tem*/
```

```
      (SELECT u.userid, COUNT(t.typecnt) AS nbtr
       FROM utilizador u, tipo_registo t
       WHERE u.userid = t.userid
         AND t.ativo = 1
       GROUP BY (u.userid)) AS u_nbtr
    WHERE u_p_nbtr.userid = u_nbtr.userid
      AND u_p_nbtr.nbtr = u_nbtr.nbtr
    GROUP BY (u_p_nbtr.userid)) AS u_nbpg_tr,
  /*Query com utilizador e o nr total de paginas de um utilizador*/
  (SELECT u.userid,
          COUNT(p.pagecounter) AS nbpg
    FROM utilizador u,
         pagina p
    WHERE u.userid = p.userid
      AND p.ativa = 1
    GROUP BY(u.userid)) AS u_nbpg
WHERE u_nbpg.userid = u_nbpg_tr.userid
  AND u_nbpg.nbpg = u_nbpg_tr.nbpg;
```

2 Restrições de Integridade

2.1 Todo o valor de contador sequência existente na relação sequência existe numa e uma vez no universo das relações tipo registo, pagina, campo, registo e valor.

A restrição de integridade pedia na realidade tem dois aspetos, temos de garantir que cada valo r do contador de sequência só pode aparecer uma vez nas tabelas tipo registo, pagina, campo, r egisto, valor e reg_pag e temos de garantir que para todo o valor contador de sequência da tabe la sequência existe um valor nas tabelas tipo registo, pagina, campo, registo, valor e registo em página.

Esta segunda condição não pode ser assegurada com um trigger pois isto iria impossibilitarmo-nos de adicionar entradas a tabela de sequência e as que possuem *foreign keys* para a tabela se quência, assim esta restrição de integridade não é trivial de assegurar, a única maneira que tính amos de assegurar esta restrição de integridade seria ao nível de aplicação sempre que fizéssem os um *insert* na tabela sequência tínhamos de estar dentro de um bloco *start transaction / end t ransaction* para sempre que fizéssemos um *insert* na tabela sequencia casso algo corresse mal p oêssemos fazer *rollback* e desta forma evitávamos ter contadores de sequencia sem estarem pr esentes em nenhuma tabela.

A primeira condição é bastante mais simples de assegurar, podemos assegurar esta condição atr avés de doze triggers no total, um trigger *BEFORE INSERT* e um *BEFORE UPDATE* em cada um da s cinco tabelas (tipo_registo, pagina, campo, registo, valor e reg_pag) segue-se um exemplo de c ada um desses dois triggers para a tabela tipo_registo os triggers são exatamente iguais para as outras tabelas apenas é necessário mudar o nome da tabela ao qual se vai aplicar o trigger.

```
/*Campos: Instituto Superior Tecnico Alameda

Disciplina: Bases de Dados 15/16

Data: 10/12/2015

Grupo: 26

Elementos do Grupo: Joao Marcal N°78471, Joao Alves N°79155, Jose Semed
o N°78294

Trigger desenvolvido por: Joao Marcal N°78471

Este trigger verifica em todas as tabelas que contem um idseq se existe
algum idseq igual ao do campo idseq do registo que vamos inserir

RI: Todo o valor de contador sequência existente na relação sequência e
xiste numa e uma vez no universo das relações tipo registo, pagina, cam
po, registo e valor.*/

DELIMITER //

CREATE TRIGGER contador_sequencia_duplicado_i BEFORE INSERT ON tipo_re
gisto

FOR EACH ROW BEGIN IF

  (SELECT COUNT(*) FROM registo WHERE new.idseq = idseq) > 0 OR
```

```
  (SELECT COUNT(*) FROM tipo_registo WHERE new.idseq = idseq) > 0 OR
  (SELECT COUNT(*) FROM pagina WHERE new.idseq = idseq) > 0 OR
  (SELECT COUNT(*) FROM campo WHERE new.idseq = idseq) > 0 OR
  (SELECT COUNT(*) FROM valor WHERE new.idseq = idseq) > 0 OR
  (SELECT COUNT(*) FROM reg_pag WHERE new.idseq = idseq) > 0 THEN
      CALL ContadorDeSequenciaDuplicado;
   END IF;
 END//
DELIMITER ;
/*Campos: Instituto Superior Tecnico Alameda
Disciplina: Bases de Dados 15/16
Data: 10/12/2015
Grupo: 26
Elementos do Grupo: Joao Marcal N°78471, Joao Alves N°79155, Jose Semed
o N°78294
Trigger desenvolvido por: Joao Marcal N°78471
Este trigger verifica em todas as tabelas que contem um idseq se existe
algum idseq igual ao do campo idseq do registo que vamos actualizar
RI: Todo o valor de contador sequência existente na relação sequência e
xiste numa e uma vez no universo das relações tipo registo, pagina, cam
po, registo e valor.*/
DELIMITER //
CREATE TRIGGER contador_sequencia_duplicado_u BEFORE UPDATE ON tipo_re
gisto
FOR EACH ROW BEGIN IF
  (SELECT COUNT(*) FROM registo WHERE new.idseq = idseq) > 0 OR
  (SELECT COUNT(*) FROM tipo_registo WHERE new.idseq = idseq) > 0 OR
  (SELECT COUNT(*) FROM pagina WHERE new.idseq = idseq) > 0 OR
  (SELECT COUNT(*) FROM campo WHERE new.idseq = idseq) > 0 OR
  (SELECT COUNT(*) FROM valor WHERE new.idseq = idseq) > 0 OR
  (SELECT COUNT(*) FROM reg_pag WHERE new.idseq = idseq) > 0 THEN
      CALL ContadorDeSequenciaDuplicado;
   END IF;
 END// DELIMITER ;
```

3 Desenvolvimento da aplicação

Para desenvolver a aplicação foram utilizadas um conjunto de scripts PHP:

- index.php
- main.php
- config.php
- mostraregistos.php
- camposregistos.php
- inserircampo.php
- inserirpagina.php
- inserirregisto.php
- inserirtiporegisto.php

A aplicação foi desenvolvida usando como base um development evironment chamado XAMPP, disponível para download gratuito online. Este ambiente permitiu-nos desenvolver a aplicação sem ser necessário configurar um host para utilizar a base de dados do sigma remotamente. Para o fazer é necessário configurar o config.php.

Para usar qualquer uma das versões da aplicação (com ou sem transacções) deverão ser alteradas as credenciais do dono da base de dados. É ideal iniciar o browser Mozilla Firefox em sessão privada para testar o programa. Antes de iniciar sessão com um outro utilizador é necessário fechar o browser.

A primeira página pela qual o utilizador passa é o index.php, na qual são pedidos um email e uma password (figura 1). Estas credenciais são verificadas na base de dados, e só é concedida passage à próxima página se forem um par válido. Este processo é necessário pricipalmente para inicializar valores em variáveis que são necessárias em todas as etapas da aplicação.

Figura 1- Log in da aplicação

Após um login com sucesso é apresentada a página main.php (figura 2). Esta é a página principal da aplicação que apresenta todas as páginas e tipos de registo ativos do utilizador. A partir desta página é possível adicionar tanto páginas como tipos de registo novos, assim como apagá-los.

Figura 2- Vista principal da aplicação

Escolher a opção apagar para qualquer uma das entidades, faz com que deixe de ser apresentada na aplicação.

Ao escolher a opção adicionar para uma página é apresentada a página inserirpagina.php, em que é pedido ao utilizador o nome que deseja dar à página (figura 3). Após escrever o nome e escolher inserir, o utilizador é levado de volta à página main.php em que já se apresenta também a página nova.

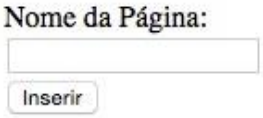

Figura 3- Ecrã para adicionar uma nova página

Ao escolher a opção adicionar para um tipo de registo é apresentada a página inserirtiporeg.php, em que é pedido ao utilizador o nome que deseja dar ao tipo de registo (figura 4). Após escrever e escolher

Inserir, o utilizador é levado de volta à página main.php em que já se apresenta também o novo tipo de registo.

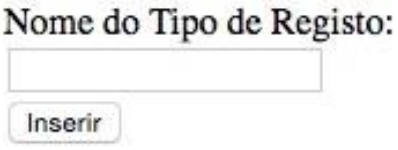

Figura 4- Ecrã para adicionar um novo tipo de registo

Ao escolher a opção Registos da Página para uma página é apresentada a página mostraregistos.php, em que são apresentados os nomes dos registos dessa página, e a opção de adicionar um registo novo (figura 5)

Figura 5- Ecrã para adicionar um novo tipo de registo

Para adicionar um registo novo o utilizador deve escolher do menu dropdown o tipo de registo que pretende que o seu registo novo seja (figura 5). Após escolhido o tipo, é apresentada a página inserirregisto.php onde o utilizador deve introduzir o nome de registo desejado, e o valor desejado para cada campo do registo (figura 6). Após isto é apresentada de novo a página main.php.

Figura 6- Ecrã para preencher os valores dos campos e o nome do registo

Para alterar os campos de um tipo de registo, o utilizador deve escolher alterar campos, após isso é apresentada a página camposregistos.php (figura 7). A partir desta página é possível apagar os campos de um tipo de registo

Figura 7-Ecrã para gerir os campos de um tipo de registo

4 Formas Normais

4.1 Em que forma normal se encontra a relação *utilizador*?

A relação utilizador encontrasse na forma normal Boyce–Codd (FNBC) isto porque segundo o enunciado apenas temos duas dependências funcionais {userid} -> {email, nome, password, questao1, resposta1, questao2, resposta2, pais, categoria} e {email} -> { userid, nome, password, questao1, resposta1, questao2, resposta2, pais, categoria} assim apenas o email e o userid é que podiam ter alguma relação que fizesse com que a relação não se encontrasse na FNBC mas como ambos são chaves candidatas não existe nenhuma implicação que faça a relação não se encontrar em FNBC.

4.2 Dependência Funcional adicionada

4.2.1 Em que forma normal se encontra a relação utilizador depois de adicionarmos a dependência funcional no enunciado.

A relação utilizador depois da dependência funcional {nome, password, questao2, resposta2, questao1, resposta1} → {email} encontrasse na 3FN porque o *email* passa a ser determina por um conjunto de atributos que não são chave candidata.

4.2.2 Repartição da tabela utilizador para ficar na FNBC

A tabela utilizador vai originar duas tabelas:

{userid, email, pais, categoria} e {userid, nome, password, questao2, resposta2, questao1, resposta1}

Desta forma não há perda de informação, ambas as tabelas ficam na FNBC mas perdemos a dependência funcional {nome, password, questao2, resposta2, questao1, resposta1} → {email}.

5 Índices

Criamos um índices b-tree pois estes são o único tipo de índices que o mySQL suporta e por esta razão era o único tipo de índice que podíamos realmente testar e observar se tinha algum efeito no tempo que a query demorava a correr.

5.1 Devolver a média do número de registos por página de um utilizador

A query que fizemos para conseguirmos escolher um índice devidamente é a seguinte:

```
SELECT u_nbpag.userid,
       u_nbreg.nb_reg/u_nbpag.nb_pag AS reg_p_pag
FROM
  (SELECT u.userid, COUNT(p.pagecounter) AS nb_pag
   FROM utilizador u,
        pagina p
   WHERE u.userid = p.userid
     AND p.ativa = 1
   GROUP BY(u.userid)) AS u_nbpag,

  (SELECT u.userid, COUNT(r.regcounter) AS nb_reg
   FROM utilizador u,
        registo r,
        tipo_registo tr
   WHERE r.userid = u.userid
     AND tr.userid = u.userid
     AND tr.typecnt = r.typecounter
     AND r.ativo = 1
     AND tr.ativo = 1
   GROUP BY u.userid) AS u_nbreg
WHERE u_nbpag.userid = u_nbreg.userid
GROUP BY u_nbpag.userid;
```

Os índices que criamos de forma a aumentar a rapidez desta query são:

- Um índice para a tabela página sobre a coluna activa, isto porque na nossa query vamos apenas querer considerar as paginas que estejam activas.

  ```
  CREATE INDEX a ON pagina (ativa);
  ```

- Um Índice para a tabela tipo de registo sobre a coluna activo, isto porque mais uma vez apenas vamos querer os registos cujos tipos de registo estejam activos.

```
CREATE INDEX a_u ON tipo_registo (ativo, userid);
```

Não criamos mais índices para as outras pois estas tabelas quando eram criadas vinham com instruções para criar índices que já eram os melhores índices para esta query ou então o mySQL apenas utilizava os índices que ele cria através das *PRIMARY KEYS*.

Na figura seguinte (figura 8) conseguimos observar o *speedup* que a query sofreu apos termos adicionada-do os nosso índices. Do numero 1 ao 3 corremos a query sem os nossos índices e do numero 6 ao 8 corremos a query com os nossos índices.

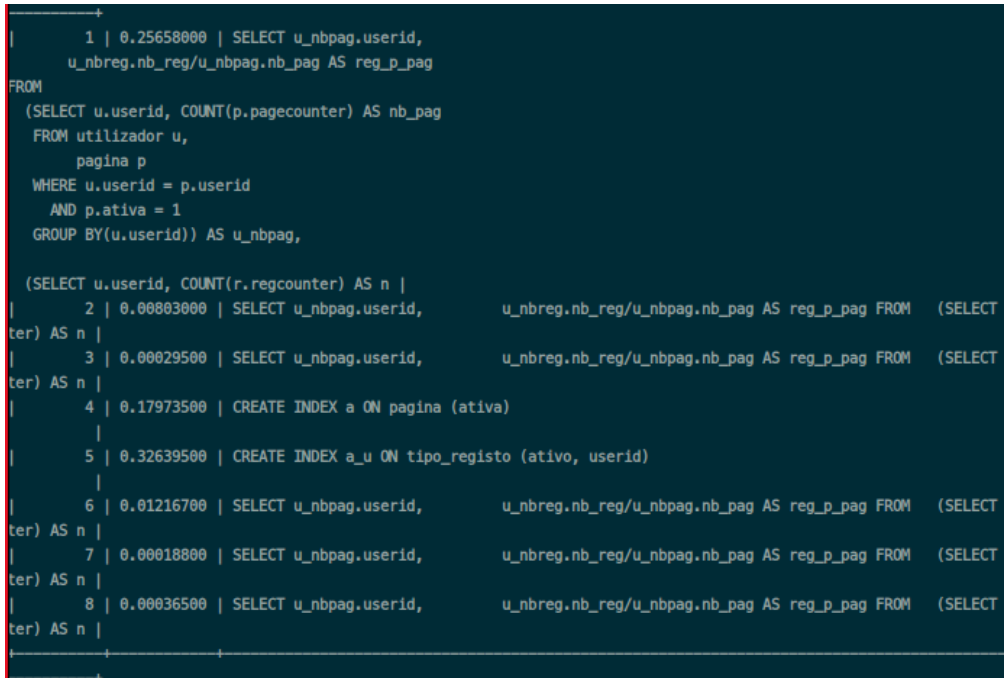

Figura 8 – Ecrã com o output do commando show profiles apos corer as queries

5.2 Ver o nome dos registos associados todas as páginas de um utilizador

A query que fizemos para conseguirmos escolher um índice devidamente é a seguinte:

```
SELECT u_rp_nbp.userid, u_rp_nbp.nome
FROM
  (SELECT u.userid, r.nome, COUNT(rp.pageid) AS nbp
   FROM utilizador u,
        reg_pag rp,
        pagina p,
        registo r,
        tipo_registo tr
   WHERE u.userid = rp.userid
     AND u.userid = p.userid
     AND u.userid = r.userid
     AND u.userid = tr.userid
     AND rp.pageid = p.pagecounter
     AND r.typecounter = tr.typecnt
     AND tr.typecnt = rp.typeid
     AND rp.regid = r.regcounter
     AND rp.typeid = r.typecounter
     AND r.ativo = 1
     AND rp.ativa = 1
     AND p.ativa = 1
     AND tr.ativo = 1
   GROUP BY(r.nome)
   ORDER BY u.userid) AS u_rp_nbp,

  (SELECT u.userid, COUNT(p.pagecounter) AS nbp
   FROM utilizador u,
        pagina p
   WHERE u.userid = p.userid
     AND p.ativa = 1
```

```
   GROUP BY(u.userid)) AS u_nbp
WHERE u_nbp.userid = u_rp_nbp.userid
  AND u_nbp.nbp = u_rp_nbp.nbp;
```

Os índices que criamos de forma a aumentar a velocidade da query foram os seguintes:

- Um índice para a tabela página sobre a coluna ativa, isto porque apenas queremos as paginas activas e no mySQL o comando *EXPLAIN* mostrava que o mySQL não estava a usar nenhuma key para esta tabela.

  ```
  CREATE INDEX a ON pagina (ativa);
  ```

- O outro índice que criamos foi um índice para a tabela *reg_pag* (registos em pagina) sobre as colunas *ativa e userid*, criamos este índice porque mais uma vez no comando *EXPLAIN* o mySQL não estava a usar nenhuma key para esta tabela:

  ```
  CREATE INDEX a_u ON reg_pag (ativa, userid);
  ```

Na criação destes índices começamos por criar um índice para o nome na tabela registo mas este índice não se provou muito beneficiante, assim acabamos por analisar o que o camando *EXPLAIN* nos fornecia e tentamos criar os melhores índices nesta situação.

Na imagem seguinte conseguimos observar o *speedup* que a query sofreu apos termos adicionado os nossos índices. Do numero 11 até ao 13 corremos a nossa query sem os nossos índices e do numero 16 ao 18 corremos a query com os nossos indices.

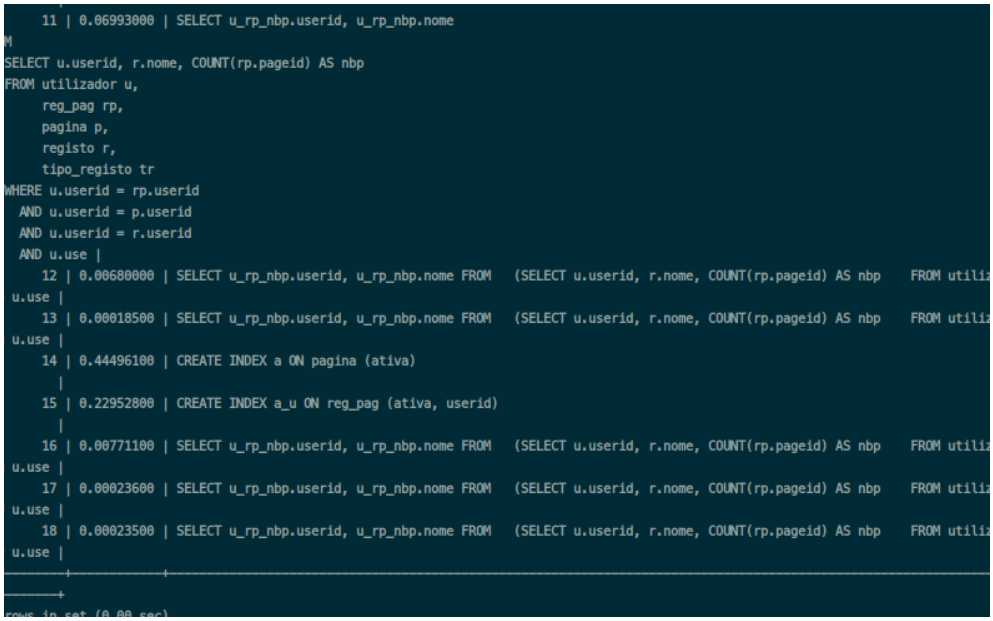

Figura 9 – Ecrã com o output do comando show profiles apos corer as queries

6 Transações

Implementou-se também uma versão da aplicação com transactions. São implementadas primitivas (beginTransaction(), commit(), rollback()), que garantem as propriedades ACID das várias acções.
Para todas as interacções com a base de dados usamos um esqueleto de try/catch com as primitivas de transacções (figura 8).

```
try {
    //Comecar a transaccao
    $db->beginTransaction();

    //Set de queries, se uma falhar deve ser lancada uma excepcao
    $db->query('primeira query');
    $db->query('segunda query');
    $db->query('terceira query');

    //Se chegar aqui, significa que nao se lancou nenhuma excepcao
    //ou seja, nenhuma query falhou
    $db->commit();
} catch (Exception $e) {
    //Foi lancada uma excepcao
    //Deve haver rollback da transaccao
    $db->rollback();
}
```

Figura 10 – Trecho de código a demonstrar como foram implemetadas as transações

7 Data Warehouse

7.1 Considerações acerca da Data Warehouse

Quando estávamos a construir o modelo em estrela pedido reparamos que as tabelas de dimensões que nos eram pedias não podiam apenas ter os atributos que estavam no formulário pois se apenas considerássemos esses atributos não iriamos desenvolver um modelo em estrela como nos foi ensinado nas aulas prática.

Assim começamos por criar a tabela *d_utilizador* esta tabela de grau 6 cujos atributos são *duserid, userid, email, nome, pais, categoria*, adicionamos o atributo userid pois é o identificador principal de um utilizador que é usado na base de dados transacional por esta razão adicionamos este atributo a dimensão utilizador, o atributo duserid foi adicionado pois foi-nos chamado a atenção o facto de dentro da dimensão a *PRIMARY KEY* não poder ser a mesma que era na tabela utilizador caso isto acontecesse íamos ter perdas de informação cada vez que um utilizador fizesse um update a um atributo que constasse na tabela *d_utilizador.*

De seguida criamos a tabela *d_tempo* uma tabela de grau 7 cujos atributos são *timeid, dia, mês, ano, horas, minutos, segundos* mais uma vez criamos estes atributos todos para suportar um modelo em estrela sem qualquer tipo de perda de informação.

A nível da nossa tabela de factos *f_login,* esta tabela é uma tabela de grau 3 cujos atributos são *duserid, timeid* e *sucesso*, os atributos *duserid* e *timeid* são respetivamente *foreigh key* para as tabelas *d_utilizador* e *d_tempo.*

Em anexo com este relatório vai um ficheiro chamado bd_sql.sql que contem o código necessário para carregar este modelo em estrela e inicializar os *triggers* necessários para base de dados OLAP ser atualizada.

7.2 Obter a média de tentativas de login para todos os utilizadores de Portugal, em cada categoria, com rollup por ano e mês.

```
SELECT dt.ano, dt.mes, COUNT(fl.sucesso)/COUNT(DISTINCT du.userid) AS
media
FROM f_login fl, d_utilizador du, d_tempo dt
WHERE fl.duserid = du.duserid
AND   fl.timeid = dt.timeid
AND   du.pais = "Portugal"
GROUP BY du.categoria, dt.ano, dt.mes with ROLLUP;
```