



TÉCNICO LISBOA

RELATÓRIO DE PROJECTO DE BASES DE DADOS Parte 2

1º Semestre 2015/2016

Grupo 11
Turno : 4ª Feira 8:30h

Filipe Sardinha – 71035
José Dias – 76397
Inês Percheiro - 76397

Conteúdo

Consultas em SQL.....	2
Restrições de Integridade	6
Desenvolvimento da Aplicação.....	9
Formas Normais	9
Índices	10
Transações	12
Data warehouse	12

Consultas em SQL

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

```
SELECT userid
FROM
  (SELECT userid,
           SUM(CASE WHEN sucesso = 1 THEN 1 ELSE 0 END) AS sucessos,
           SUM(CASE WHEN sucesso = 0 THEN 1 ELSE 0 END) AS insucessos
   FROM login
   GROUP BY userid) AS auxtable
WHERE insucessos > sucessos;
```

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

```
SELECT aux.userid,
       aux2.regid
FROM
  (SELECT P.userid,
         COUNT(*) AS numpag
   FROM pagina P
  WHERE P.ativa = TRUE
  GROUP BY P.userid) AS aux,
  (SELECT RP.userid,
         RP.regid,
         COUNT(*) AS numreg
   FROM reg_pag RP,
        pagina PG,
        registo R,
        tipo_registo TR
  WHERE RP.ativa = TRUE
        AND RP.userid = PG.userid
        AND RP.pageid = PG.pagecounter
        AND PG.ativa = TRUE
        AND RP.userid = TR.userid
        AND RP.typeid = TR.typecnt
        AND TR.ativo = TRUE
        AND RP.userid = R.userid
        AND RP.regid = R.regcounter
        AND R.ativo = TRUE
  GROUP BY RP.regid
  ORDER BY RP.userid) AS aux2
WHERE aux2.numreg = aux.numpag
      AND aux.userid = aux2.userid;
```

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

```
SELECT tab1.userid
FROM
  ( SELECT userid,
        med_reg_ppag AS nummedreg
  FROM
    ( SELECT userid,
          avg(nreg) AS med_reg_ppag
    FROM
      (SELECT rp.userid,
              count(*) AS nreg,
              rp.pageid
      FROM reg_pag rp
      WHERE rp.ativa = TRUE
      GROUP BY pageid
      ORDER BY userid) AS tablea
    GROUP BY userid) AS aux ) AS tab1
WHERE tab1.nummedreg >= ALL
  ( SELECT med_reg_ppag AS nummedreg
  FROM
    ( SELECT userid,
          avg(nreg) AS med_reg_ppag
    FROM
      (SELECT rp.userid,
              COUNT (*) AS nreg,
              rp.pageid
      FROM reg_pag rp
      WHERE rp.ativa = TRUE
      GROUP BY pageid
      ORDER BY userid) AS tablea
    GROUP BY userid) AS aux);
```

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

```
SELECT aux.userid,
       aux3.typeid
FROM
  (SELECT P.userid,
         COUNT(*) AS numpag
   FROM pagina P
   WHERE P.ativa = TRUE
   GROUP BY P.userid) AS aux,
  (SELECT TIP.userid,
         COUNT(*) AS numtr
   FROM tipo_registro TIP
   WHERE TIP.ativo = TRUE
   GROUP BY TIP.userid) aux2,
  (SELECT RP.userid,
         RP.typeid,
         COUNT(*) AS numtypereg
   FROM reg_pag RP,
        registro R,
        tipo_registro TR,
        pagina PG
   WHERE RP.ativa = TRUE
        AND RP.userid = R.userid
        AND RP.regid = R.regcounter
        AND R.ativo = TRUE
        AND RP.userid = TR.userid
        AND RP.typeid = TR.typecnt
        AND TR.ativo = TRUE
        AND RP.userid = PG.userid
        AND RP.pageid = PG.pagecounter
        AND PG.ativa = TRUE
   GROUP BY typeid
   ORDER BY userid) AS aux3
WHERE aux3.numtypereg = aux.numpag * aux2.numtr
      AND aux.userid = aux2.userid
      AND aux.userid = aux3.userid;
```

Restrições de Integridade

A) 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.

Nota: Para facilitar a leitura foram feitas algumas tabulações de forma a não dividir a query em zonas mais sensíveis.

```
CREATE OR REPLACE TRIGGER trigger_p
BEFORE
INSERT ON pagina
FOR EACH ROW DECLARE idp number; DECLARE idr number; DECLARE idc
number; DECLARE idv number; DECLARE idt number; DECLARE tot
number; BEGIN
SELECT count(*) INTO idp
FROM pagina
WHERE idseq = new.idseq;
    SELECT count(*) INTO idv
    FROM valor WHERE idseq = new.idseq;
    SELECT count(*) INTO idc
    FROM campo WHERE idseq = new.idseq;
    SELECT count(*) INTO idr
    FROM registo WHERE idseq = new.idseq;
    SELECT count(*) INTO idt
    FROM tipo_registo WHERE idseq = new.idseq;
    SET tot:=: idp + idv + idc + idr + idt; IF tot > 0 THEN
    ROLLBACK raiseerror('Duplicate sequencia\n'); END IF; END;
CREATE OR REPLACE TRIGGER trigger_v
BEFORE
INSERT ON valor
```

```

FOR EACH ROW DECLARE idp number; DECLARE idr number; DECLARE
idc number; DECLARE idv number; DECLARE idt number; DECLARE tot
number; BEGIN

SELECT count(*) INTO idp
FROM pagina WHERE idseq = new.idseq;
SELECT count(*) INTO idv
FROM valor WHERE idseq = new.idseq;
SELECT count(*) INTO idc
FROM campo WHERE idseq = new.idseq;
SELECT count(*) INTO idr
FROM registro WHERE idseq = new.idseq;
SELECT count(*) INTO idt
FROM tipo_registro WHERE idseq = new.idseq;
SET tot=: idp + idv + idc + idr + idt; IF tot > 0 thenthen
ROLLBACK raiseerror('Duplicate sequencia\n'); END IF; END;
CREATE OR REPLACE TRIGGER trigger_c
BEFORE
INSERT ON campo

FOR EACH ROW DECLARE idp number; DECLARE idr number; DECLARE
idc number; DECLARE idv number; DECLARE idt number; DECLARE tot
number; BEGIN

SELECT count(*) INTO idp
FROM pagina WHERE idseq = new.idseq;
SELECT count(*) INTO idv
FROM valor WHERE idseq = new.idseq;
SELECT count(*) INTO idc
FROM campo WHERE idseq = new.idseq;
SELECT count(*) INTO idr
FROM registro WHERE idseq = new.idseq;
SELECT count(*) INTO idt
FROM tipo_registro WHERE idseq = new.idseq;
SET tot=: idp + idv + idc + idr + idt; IF tot > 0 THEN
ROLLBACK raiseerror('Duplicate sequencia\n'); END IF; END;
CREATE OR REPLACE TRIGGER trigger_r
BEFORE
INSERT ON registro

```



```

FOR EACH ROW DECLARE idp number; DECLARE idr number; DECLARE
idc number; DECLARE idv number; DECLARE idt number; DECLARE tot
number; BEGIN

SELECT count(*) INTO idp
FROM pagina WHERE idseq = new.idseq;
SELECT count(*) INTO idv
FROM valor WHERE idseq = new.idseq;
SELECT count(*) INTO idc
FROM campo WHERE idseq = new.idseq;
SELECT count(*) INTO idr
FROM registro WHERE idseq = new.idseq;
SELECT count(*) INTO idt
FROM tipo_registro WHERE idseq = new.idseq;
SET tot=: idp + idv + idc + idr + idt; IF tot > 0 THEN
ROLLBACK raiseerror('Duplicate sequencia\n'); END IF; END;
CREATE OR REPLACE TRIGGER trigger_t
BEFORE
INSERT ON tipo_registro

FOR EACH ROW DECLARE idp number; DECLARE idr number; DECLARE
idc number; DECLARE idv number; DECLARE idt number; DECLARE tot
number; BEGIN

SELECT count(*) INTO idp
FROM pagina WHERE idseq = new.idseq;
SELECT count(*) INTO idv
FROM valor WHERE idseq = new.idseq;
SELECT count(*) INTO idc
FROM campo WHERE idseq = new.idseq;
SELECT count(*) INTO idr
FROM registro WHERE idseq = new.idseq;
SELECT count(*) INTO idt
FROM tipo_registro WHERE idseq = new.idseq;
SET tot=: idp + idv + idc + idr + idt; IF tot > 0 THEN
ROLLBACK raiseerror('Duplicate sequencia\n'); END IF; END;

```

Desenvolvimento da Aplicação

Os ficheiros relativos ao desenvolvimento da Aplicação encontram-se dentro do ficheiro .ZIP submetido no Sistema Fénix na pasta de nome “Desenvolvimento da Aplicação”.

Assim como foi aconselhado no Piazza.

Formas Normais

A) Em que forma normal se encontra a relação utilizador?

A relação utilizador encontra-se na 3ª forma normal. Justificaremos esta resposta passo a passo. A relação encontra-se na 1ª forma porque todos os seus atributos são valores escalares, sendo que em cada coluna apenas existe um valor. Encontra-se também na 2ª forma pois cada atributo não chave é completamente dependente dos atributos da chave, sem esta dependência não era possível identificar os atributos de um utilizador.

Esta relação encontra-se também na 3ª forma normal pois todos os atributos não chave são independentes entre si.

Esta constatação causou-nos algumas dúvidas pois pusemos em causa se o atributo *questao1* determinaria o atributo *resposta1*, e o atributo *questao2* determinaria o atributo *resposta2* mas colocando a hipótese de dois utilizadores distintos terem o atributo *questao1* igual mas depois os atributos *resposta1* diferentes esclareceu-nos, pois apesar de existir uma relação na prática entre estes dois atributos não lhe podemos chamar uma relação de dependência.

B) Considere que existe um trigger que garante que é sempre verdade que:

nome, password, questao2, resposta2, questao1, resposta1 → *email*

– Em que forma normal se encontra agora a relação utilizador?

– Caso esta não se encontre na BCNF, proponha uma decomposição (sem perdas de informação) da mesma de forma a que todas as relações obtidas estejam na BCNF

Com a alteração proposta, podemos dizer que a relação utilizador passa a estar na 2ª forma normal, pois continuam-se a verificar a 1ª e 2ª formas normais, mas deixa-se de garantir que todos os atributos não chave são independentes entre si uma vez que o *email* passa a ser determinado pelo *nome, password, questao2, resposta2, questao1* e *resposta1*.

Divisão proposta:

R1 = (nome, password, questao2, resposta2, questao1, resposta1, email)

R2 = (userid, nome, password, questao2, resposta2, questao1, resposta1, email)

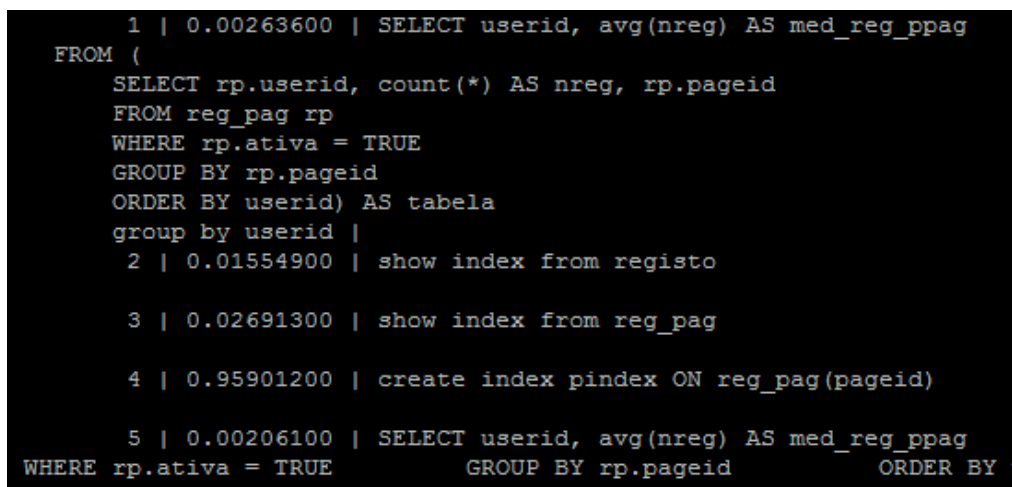
Índices

A) Devolver a média do número de registos por página de um utilizador.

```
SELECT userid,
       avg(nreg) AS med_reg_ppag
FROM
  (SELECT rp.userid,
         count(*) AS nreg,
         rp.pageid
   FROM reg_pag rp
  WHERE rp.ativa = TRUE
  GROUP BY rp.pageid
  ORDER BY userid) AS tabela
GROUP BY userid;
```

Com esta consulta em SQL, é possível obter o resultado da média do número de registos por página de um utilizador, ainda assim é possível tornar o processo de procura mais eficiente através da criação de um índice primário, sobre o atributo pageid para aceder diretamente ao pageid pretendido sem ter de percorrer a tabela reg_pag por completo.

Com a figura seguinte é possível observar a diferença de tempo entre a consulta com índice e sem índice.



```
1 | 0.00263600 | SELECT userid, avg(nreg) AS med_reg_ppag
FROM (
  SELECT rp.userid, count(*) AS nreg, rp.pageid
  FROM reg_pag rp
  WHERE rp.ativa = TRUE
  GROUP BY rp.pageid
  ORDER BY userid) AS tabela
group by userid |
2 | 0.01554900 | show index from registo

3 | 0.02691300 | show index from reg_pag

4 | 0.95901200 | create index pindex ON reg_pag(pageid)

5 | 0.00206100 | SELECT userid, avg(nreg) AS med_reg_ppag
WHERE rp.ativa = TRUE      GROUP BY rp.pageid      ORDER BY t
```

Figura 1 – Tempo observado alínea a)

B) Ver o nome dos registos associados à página de um utilizador.

```
SELECT userid,
       regid,
       nome FROM(
                (SELECT RP.userid, RP.pageid, RP.regid
                 FROM reg_pag RP
                 WHERE RP.ativa = TRUE
                 ORDER BY userid) AS aux1
                NATURAL JOIN
                (SELECT R.userid, R.regcounter, R.nome
                 FROM registo R
                 WHERE R.ativo = TRUE) AS aux2)
ORDER BY userid;
```

Esta consulta em mySQL permite obter os nomes de todos os registos associados às páginas de um utilizador, implementando um índice na tabela de registo sobre o atributo regcounter é possível acelerar a procura, pois o índice primário facilita o acesso aos registos na tabela registo que faz com que não seja necessário percorrer toda a tabela

Consulta sem índice:

```
|          1 | 0.00026000 | SELECT aux2.userid, aux2.regcounter, aux2.nome
FROM(
(SELECT RP.userid, RP.pageid, RP.regid
FROM reg_pag RP
ORDER BY userid) as aux1
JOIN (
SELECT R.userid, R.regcounter, R.nome
FROM registo R
ORDER BY R.userid) as aux2
ON aux1.userid = aux2.userid
AND aux1.regid = aux2.regcounter)
ORDER BY aux2.us |
```

Consulta com índice:

```
|          4 | 0.00017300 | SELECT aux2.userid, aux2.regcounter, aux2.nome
FROM(
(SELECT RP.userid, RP.pageid, RP.regid
FROM reg_pag RP
ORDER BY userid) as aux1
JOIN (
SELECT R.userid, R.regcounter, R.nome
FROM registo R
ORDER BY R.userid) as aux2
ON aux1.userid = aux2.userid
AND aux1.regid = aux2.regcounter)
ORDER BY aux2.us |
```

Transações

As transações implementadas permitem garantir um acesso simultâneo à base de dados. Esta transação permite que sempre que um utilizador pretenda fazer uma alteração na base de dados a query start transaction impeça que até todas as modificações estejam realizadas e seja feito um commit das mesmas elas apareçam na base de dados. Quando falha uma transação existe um mecanismo inerente á mesma denominado de rollback que restaura a database para um estado anterior à transação falhada.

As transações encontram-se aplicadas nos ficheiros PHP dentro do ficheiro .ZIP submetido no Sistema Fénix na pasta de nome “Desenvolvimento da Aplicação”.

Data warehouse

- (a) Crie na base de dados o esquema de uma estrela com informação de número de tentativas de login tendo como dimensões: d_utilizador(email, nome, país, categoria) e d_tempo(dia, mes, ano). Escreva as instruções SQL necessárias para carregar o esquema em estrela a partir das tabelas existentes.
- (b) Considerando o esquema da estrela criado em (a), escreva a interrogação em MySQL para obter a média de tentativas de login para todos os utilizadores de Portugal, em cada categoria, com rollup por ano e mês

```
CREATE TABLE IF NOT EXISTS d_utilizador ( userid INT NOT NULL
AUTO_INCREMENT, email VARCHAR(255) NOT NULL, nome VARCHAR(255)
NOT NULL, pais VARCHAR(45) NOT NULL, categoria VARCHAR(45) NOT
NULL, PRIMARY KEY (userid), UNIQUE INDEX email_UNIQUE (email));
```

```
CREATE TABLE IF NOT EXISTS d_tempo ( timeid INT NOT NULL
AUTO_INCREMENT, dia VARCHAR(255) NOT NULL, mes VARCHAR(255) NOT
NULL, ano VARCHAR(255) NOT NULL, PRIMARY KEY (timeid));
```

```
CREATE TABLE IF NOT EXISTS star ( facid INT NOT NULL
AUTO_INCREMENT, userid INT NOT NULL, timeid INT NOT NULL,
PRIMARY KEY (facid),
FOREIGN KEY (userid) REFERENCES
d_utilizador (userid) ON
DELETE CASCADE ON
UPDATE CASCADE,
FOREIGN KEY (timeid) REFERENCES
```

```
d_tempo (timeid) ON
```

```
DELETE CASCADE ON  
UPDATE CASCADE);
```

```
INSERT INTO d_utilizador (userid, email, nome, pais, categoria)  
SELECT userid,  
       email,  
       nome,  
       pais,  
       categoria  
FROM utilizador;
```

```
INSERT INTO d_tempo (dia, mes, ano)  
SELECT DAY(moment) AS dia,  
       MONTH(moment) AS mes,  
       YEAR(moment) AS ano  
FROM login;
```

```
INSERT INTO star (facid, userid, timeid)  
SELECT t.timeid AS timeid,  
       u.userid AS userid  
FROM star s  
INNER JOIN d_utilizador u ON (s.userid = u.userid)  
INNER JOIN d_tempo t ON (s.timeid = t.timeid);
```