Projeto de Bases de Dados, Parte 2

Grupo 2

Turno BD22517L10

António Tavares, 78122
Luís Borges, 78349

Paulo Ritto, 78929

Índice

1 - Consultas SQL

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

```
SELECT L.userid FROM login L, utilizador U

WHERE L.userid=U.userid

GROUP BY L.userid HAVING AVG (L.sucesso) < 0.5;
```

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

```
SELECT DISTINCT R.regcounter from registo R

WHERE NOT EXISTS (SELECT P.pagecounter FROM pagina P WHERE R.userid = P.userid
AND NOT EXISTS (SELECT * FROM reg_pag RP WHERE P.pagecounter = RP.pageid and
R.regcounter = RP.regid));
```

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

```
SELECT DISTINCT P.userid FROM pagina P, reg_pag RP

WHERE P.ativa=1 AND RP.ativa=1 AND P.userid=RP.userid

GROUP BY P.userid

HAVING (COUNT(DISTINCT RP.idregpag)/COUNT(DISTINCT P.pagecounter)) >=
ALL(SELECT (COUNT(DISTINCT RP2.idregpag)/COUNT(DISTINCT P2.pagecounter)) FROM
reg_pag RP2, pagina P2 WHERE RP2.ativa=1 AND P2.ativa=1 AND RP2.userid=P2.userid
GROUP BY P2.userid);
```

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

```
SELECT U.userid AS userid FROM utilizador U, pagina P

WHERE P.ativa =1 AND U.userid = P.userid AND P.userid

NOT IN (SELECT DISTINCT U2.userid AS userid2 FROM utilizador U2, pagina P2 WHERE
U2.userid = P2.userid AND P2.ativa=1 AND EXISTS (SELECT * FROM tipo_registo T
WHERE U2.userid = T.userid AND T.ativo=1 AND NOT EXISTS (SELECT * FROM reg_pag
A WHERE P2.pagecounter = A.pageid AND T.typecnt = A.typeid AND A.ativa=1)));
```

2 - Restrição de Integridade

Para resolver a questão da restrição de integridade, decidiu-se criar de antemão um procedimento, o conta-idseq, que recebe um id de uma sequência e devolve o número de vezes que se encontra na tabela sequência. Se essa função alguma vez retornar um número maior que 0, sabe-se que o que estamos a inserir/atualizar é inválido, levantando assim um erro.

A restrição de integridade pedida para ser resolvida foi a seguinte: *Todo o valor de contador_sequencia existente na relação sequencia existe numa e uma vez no universo das relações tipo registo, pagina, campo, registo e valor*

/ Procedimento que perante um idseq, retorna o número de vezes que se encontra na tabela sequência */*

```
Delimiter //
create function conta-idseq (_idseq int(11))
        returns integer
begin
        declare idseq_count integer;
        select count(*) into idseq_count
                from sequencia
                where contador_sequencia = _idseq;
        return idseq_count;
End //
Delimiter ;
```

*/*Trigger que verifica o insert na tabela tipo_registo*/*

```
Delimiter //
CREATE TRIGGER insert_check_tipo_registo BEFORE INSERT ON tipo_registo
FOR EACH ROW
BEGIN
IF conta-idseq(NEW.idseq) > 0 THEN
CALL PROC-QUE-NAO-EXISTE()
END IF;
END //
Delimiter ;
```

```
/*Trigger que verifica o insert na tabela pagina*/

Delimiter //
CREATE TRIGGER insert_check_pagina BEFORE INSERT ON pagina
FOR EACH ROW
BEGIN
IF conta-idseq(NEW.idseq) > 0 THEN
CALL PROC-QUE-NAO-EXISTE()
END IF;
END //
Delimiter ;
/*Trigger que verifica o insert na tabela campo*/

Delimiter //
CREATE TRIGGER insert_check_campo BEFORE INSERT ON campo
FOR EACH ROW
BEGIN
IF conta-idseq(NEW.idseq) > 0 THEN
CALL PROC-QUE-NAO-EXISTE()
END IF;
END //
Delimiter ;

/*Trigger que verifica o insert na tabela registo*/

Delimiter //
CREATE TRIGGER insert_check_registo BEFORE INSERT ON registo
FOR EACH ROW
BEGIN
IF conta-idseq(NEW.idseq) > 0 THEN
CALL PROC-QUE-NAO-EXISTE()
END IF;
END //
Delimiter ;

/*Trigger que verifica o insert na tabela valor*/

Delimiter //
CREATE TRIGGER insert_check_valor BEFORE INSERT ON valor
FOR EACH ROW
BEGIN
IF conta-idseq(NEW.idseq) > 0 THEN
CALL PROC-QUE-NAO-EXISTE()
END IF;
END //
Delimiter ;
```

```
/*Trigger que verifica o update na tabela tipo_registo*/

Delimiter //
CREATE TRIGGER insert_check_tipo_registo BEFORE UPDATE ON tipo_registo
FOR EACH ROW
BEGIN
IF conta-idseq(NEW.idseq) > 0 THEN
CALL PROC-QUE-NAO-EXISTE()
END IF;
END //
Delimiter ;
/*Trigger que verifica o update na tabela pagina*/

Delimiter //
CREATE TRIGGER insert_check_pagina BEFORE UPDATE ON pagina
FOR EACH ROW
BEGIN
IF conta-idseq(NEW.idseq) > 0 THEN
CALL PROC-QUE-NAO-EXISTE()
END IF;
END //
Delimiter ;

/*Trigger que verifica o update na tabela campo*/

Delimiter //
CREATE TRIGGER insert_check_campo BEFORE UPDATE ON campo
FOR EACH ROW
BEGIN
IF conta-idseq(NEW.idseq) > 0 THEN
CALL PROC-QUE-NAO-EXISTE()
END IF;
END //
Delimiter ;

/*Trigger que verifica o update na tabela registo*/

Delimiter //
CREATE TRIGGER insert_check_registo BEFORE UPDATE ON registo
FOR EACH ROW
BEGIN
IF conta-idseq(NEW.idseq) > 0 THEN
CALL PROC-QUE-NAO-EXISTE()
END IF;
END //
Delimiter ;
```

*/*Trigger que verifica o update na tabela valor*/*

```
Delimiter //
CREATE TRIGGER insert_check_valor BEFORE UPDATE ON valor
FOR EACH ROW
BEGIN
IF conta-idseq(NEW.idseq) > 0 THEN
CALL PROC-QUE-NAO-EXISTE()
END IF;
END //
Delimiter ;
```

3 - Desenvolvimento da Aplicação

Para o desenvolvimento da aplicação, criou-se um conjunto de páginas em HTML, que pediam ao utilizador determinada informação, sendo que também existe um ficheiro PHP correspondente, que processa esse pedido. Todos os ficheiros foram enviados, juntamente com este relatório, pelo que no relatório iremos apenas apresentar o PHP.

a) *Inserir uma nova página*

```
<?php
	$servername = "db.ist.utl.pt";
	$username = "ist178349";
	$password = "=luispedro9=";
	$dbname = "ist178349";
	$pagename = $_GET['pagename'];

	try {
		$db = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);
		$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

		foreach($db->query("select userid from login where
contador_login=(select max(contador_login) from login);") as $row) {
			$userid = $row['userid'];
		}

		foreach($db->query("select max(pagecounter) from pagina;") as $row) {
			$pagecounter = $row['max(pagecounter)'] + 1;
		}

		foreach($db->query("select max(S.contador_sequencia) from sequencia
S, login L where L.contador_login=(select max(contador_login) from login) and
S.userid=L.userid;") as $row) {
			$idseq = $row['max(S.contador_sequencia)'];
		}

		$stmt = $db->prepare("INSERT INTO pagina (userid, pagecounter, nome,
idseq, ativa) VALUES (?, ?, ? ,?, true)");
		$stmt->execute(array($userid, $pagecounter, $pagename, $idseq));

	}

	catch(PDOException $e){
		echo $sql . "<br>" . $e->getMessage();
	}
	$db = null;   ?>
```

b) *Inserir um novo tipo de registo*

```
<?php
	$servername = "db.ist.utl.pt";
	$username = "ist178349";
	$password = "=luispedro9=";
	$dbname = "ist178349";
	$regtype = $_GET['regtype'];

	try {
		$db = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);

		$db->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

		foreach($db->query("select userid from login where
contador_login=(select max(contador_login) from login);") as $row) {
			$userid = $row['userid'];
		}

		foreach($db->query("select max(typecnt) from tipo_registo;") as
$row) {
			$typecounter = $row['max(typecnt)'] + 1;
		}


		foreach($db->query("select max(S.contador_sequencia) from
sequencia S, login L where L.contador_login=(select max(contador_login) from login)
and S.userid=L.userid;") as $row) {
			$idseq = $row['max(S.contador_sequencia)'];
		}

		$stmt = $db->prepare("INSERT INTO tipo_registo (userid,
typecnt, nome, ativo, idseq) VALUES (?, ?, ? ,true, ?)");
		$stmt->execute(array($userid, $typecounter, $regtype, $idseq));

	}

	catch(PDOException $e){
		echo $sql . "<br>" . $e->getMessage();
	}
	$db = null;
?>
```

c) *Inserir novos campos para um tipo de registo*

```
<?php
	$servername = "db.ist.utl.pt";
	$username = "ist178349";
	$password = "=luispedro9=";
	$dbname = "ist178349";
	$regtype = $_GET['regtype'];
	$camponame = $_GET['camponame'];

	try {
		$db = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);

		$db->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

		foreach($db->query("select userid from login where
contador_login=(select max(contador_login) from login);") as $row) {
			$userid = $row['userid'];
		}




		foreach($db->query("select C.typecnt from campo C,
tipo_registo TP where TP.ativo=1 and TP.nome='$regtype' and TP.userid=C.userid;") as
$row) {
			$typecounter = $row['typecnt'];
		}

		foreach($db->query("select max(campocnt) from campo;") as
$row) {
			$campocounter = $row['max(campocnt)'] + 1;
		}

		foreach($db->query("select max(S.contador_sequencia) from
sequencia S, login L where L.contador_login=(select max(contador_login) from login)
and S.userid=L.userid;") as $row) {
			$idseq = $row['max(S.contador_sequencia)'];
		}



		  $stmt = $db->prepare("INSERT INTO campo (userid, typecnt,
campocnt, idseq, ativo, nome) VALUES (?, ?, ?, ?, true, ?)");
```

```
                        $stmt->execute(array($userid, $typecounter, $campocounter,
$idseq, $camponame));

                        echo 'Feito!';
                }


        catch(PDOException $e){
                echo $sql . "<br>" . $e->getMessage();
        }
        $db = null;
?>
```

d) *Retirar uma página*

```
<?php
        $servername = "db.ist.utl.pt";
        $username = "ist178349";
        $password = "=luispedro9=";
        $dbname = "ist178349";
        $pagename = $_GET['pagename'];

        try {
                        $db = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);

                        $db->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                        foreach($db->query("select userid from login where
contador_login=(select max(contador_login) from login);") as $row) {
                                $userid = $row['userid'];
                        }

                        foreach($db->query("select max(pagecounter) from pagina;") as
$row) {
                                $pagecounter = $row['max(pagecounter)'] + 1;
                        }




                        $stmt = $db->prepare("update pagina set ativa=0 where nome=
? and userid=?;");
                        $stmt->execute(array($pagename, $userid));
```

```
                        echo 'Feito';
                }

        catch(PDOException $e){
                echo $sql . "<br>" . $e->getMessage();
        }
        $db = null;
?>
```

e) *Retirar um tipo de registo*

```
<?php
        $servername = "db.ist.utl.pt";
        $username = "ist178349";
        $password = "=luispedro9=";
        $dbname = "ist178349";
        $regname = $_GET['regname'];

        try {
                        $db = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);

                        $db->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                        foreach($db->query("select userid from login where
contador_login=(select max(contador_login) from login);") as $row) {
                                $userid = $row['userid'];
                        }

                        foreach($db->query("select typecnt from tipo_registo where
nome='$regname' and userid='$userid';") as $row) {
                                $typecounter = $row['typecnt'];
                        }

                        $stmt = $db->prepare("update tipo_registo set ativo=0 where
nome = ? and userid = ?;");
                        $stmt->execute(array($regname, $userid));



                        $stmt = $db->prepare("update registo set ativo=0 where
typecounter = ? and userid = ?;");
                        $stmt->execute(array($typecounter, $userid));
```

```
                echo 'Feito';
        }

    catch(PDOException $e){
        echo $sql . "<br>" . $e->getMessage();
    }
    $db = null;
?>
```

f) *Inserir um registo e os respectivos valores dos campos associados*

```
<?php
    $servername = "db.ist.utl.pt";
    $username = "ist178349";
    $password = "=luispedro9=";
    $dbname = "ist178349";
    $regtype = $_GET['regtype'];
    $camponame = $_GET['camponame'];

    try {
        $db = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);

        $db->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

        foreach($db->query("select userid from login where
contador_login=(select max(contador_login) from login);") as $row) {
            $userid = $row['userid'];
        }

        foreach($db->query("select C.typecnt from campo C,
tipo_registo TP where TP.ativo=1 and TP.nome='$regtype' and TP.userid=C.userid;") as
$row) {
            $typecounter = $row [‘typecnt’];
        }

        $stmt = $db->prepare("update campo set ativo=0 where typecnt
= ? and nome= ? and userid = ?;");
        $stmt->execute(array($typecounter, $camponame, $userid));

        echo 'Feito!';
    }

    catch(PDOException $e){
        echo $sql . "<br>" . $e->getMessage();
```

```
	}
	$db = null;
?>
```

h) *Ver uma página com os registos nela contidos*

```
<?php
	$servername = "db.ist.utl.pt";
	$username = "ist178349";
	$password = "=luispedro9=";
	$dbname = "ist178349";
	$pagename = $_GET['pagename'];

	try {
			$db = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);

			$db->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

			foreach($db->query("select userid from login where
contador_login=(select max(contador_login) from login);") as $row) {
				$userid = $row['userid'];
			}

			foreach($db->query("select pagecounter from pagina where
nome= '$pagename';") as $row) {
				$pagecounter = $row['pagecounter'];
			}

			foreach($db->query("select nome from registo R, reg_pag RP
where R.regcounter=RP.regid and RP.pageid='$pagecounter';") as $row) {
				echo $row['nome'] . "<br><br>";
			}

		}

	catch(PDOException $e){
		echo $sql . "<br>" . $e->getMessage();
	}
	$db = null;
?>
```

4 - Formas Normais

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

Para esta relação pertencer à Primeira Forma Normal (1FN), todos os atributos de tabela têm de ser definidos em domínios com valores atómicos. Nenhum dos atributos é processado como um conjunto de partes. Desta forma, esta condição é satisfeita, pelo que a relação se encontra pelo menos na 1FN. Para pertencer à Segunda Forma Normal (2FN), os atributos que não pertencem à chave têm de depender dela como um todo. Tendo em conta que as chaves candidatas não são compostas e, por conseguinte, nenhum dos elementos dessa hipotética chave composta determina outro de forma separada, pode-se também afirmar que a relação também se encontra na 2FN. Para pertencer à Terceira Forma Normal (3FN), todos os atributos que não pertencem à chave têm de ser mutuamente independentes. Ora, este requisito é cumprido, dado que as relações de dependência existentes são as chaves candidatas em questão. Logo, a relação encontra-se na 3FN. Também se pode afirmar que a relação pertence à BCNF, dado que todos os determinantes da relação são as suas chaves candidatas: *userid* e *email*.

b) *Com o trigger considerado, em que forma normal se encontra agora a relação utilizador?*

Com este novo trigger, há que rever as condições de normalização desta relação. Neste novo contexto, passa a existir uma terceira relação de dependência, em que a combinação de nome, password, questao2, resposta2, questao1 e resposta1 formam um novo determinante, que determina email. Desta forma, existe um determinante que não é chave, pelo que a relação não pode estar na BCNF. No entanto, esta continua a obedecer às outras três (nenhum dos componentes do conjunto determina outro separadamente). Assim sendo, há que fazer uma decomposição para ter esta relação na BCNF. Pegando nos atributos de utilizador e no novo trigger, podem-se obter duas relações: uma relação utilizadorA, com os atributos do novo determinante como chave primária (neste caso, composta); e uma relação utilizadorB, com todos os atributos da relação utilizador, excluindo o atributo email. userid será a chave primária. email passará assim a ser determinado, em vez de ser determinante (deixa de ser chave candidata). Desta forma, voltam-se a ter todos os determinantes como chaves, pelo que as relações voltam à BCNF.

5 – Índices

Para se encontrarem índices adequados para melhorar e optimizar o desempenho das interrogações pedidas, há que as definir primeiro em SQL e calcular métricas de desempenho (nota: tempos medidos no MySQL Workbench 6.5).

Comece-se então pela primeira interrogação: *Devolver a média do número de registos por página de um utilizador*. Esta pode ser traduzida a partir da seguinte query:

SELECT (COUNT(DISTINCT RP.idregpag) /COUNT(DISTINCT P.pagecounter)) FROM pagina P, registo R, reg_pag RP

WHERE P.userid = 'userid' AND R.userid = 'userid' AND RP.userid = 'userid' AND P.pagecounter = RP.pageid AND R.regcounter = RP.regid AND P.ativa = 1 AND R.ativo = 1 AND RP.ativa = 1;

Através da medição de tempos (recurso a PROFILING), obteve-se um tempo inicial de 0,00048500s (há que notar que esta execução não foi a primeira, pelo que já existiam valores em cache. A primeira execução resultou num valor de 0,00119675s). A tabela obtida através do comando EXPLAIN mostrava que o maior número de consultas era feito na tabela RP, pelo que a chave de pesquisa teria de ser atributo desta.

Assim sendo, definiu-se um índice com as seguintes características:

- Secundário, cujas chaves de pesquisa são os atributos *userid* e *regid* de *reg-_pag.* Ambos os atributos não são chaves primárias. Logo, este índice não poderá ser primário;
- Desagrupado, pois tanto a ordem dos registos de dados de *userid* como a de *regid* não são próximas das das entradas de dados.
- Denso, já que tanto *userid* como *regid* possuem uma entrada de dados por cada valor que assumem;
- Hash-based, já que se está a lidar com interrogações (recurso a B+-Trees seria mais vantajogo em consultas com valores situados entre intervalos definidos) (Nota: apesar de esta ser a resposta teoricamente mais adequada, os índices Hash-based só são suportados pelo Memory Storage Engine, pelo que não se registarão as devidas melhorias influenciadas pelo tipo de indexação. De qualquer forma, os índices serão criados com o comando USING HASH).

Este foi o índice obtido:

CREATE INDEX id_reg_pag1 USING HASH ON reg_pag(userid, regid);

No que toca a medição de tempos, as melhorias foram ligeiras, tendo havido uma ligeira redução de 0,00002000s em relação ao tempo original. O número de linhas consultadas na tabela RP também não diminuiu. O suporte para Hash poderia resultar em desempenhos superiores quando comparados com este.

Passe-se então para a segunda interrogação: *Ver o nome dos registos associados à página de um utilizador.* A query obtida foi a seguinte:

SELECT DISTINCT R.nome AS regname FROM pagina P, registo R, reg_pag RP

WHERE R.regcounter = RP.regid AND P.pagecounter = 'pagecounter' AND R.userid = 'userid' AND P.userid = 'userid' AND RP.ativa = 1 AND P.ativa = 1 AND R.ativo = 1

ORDER BY R.nome;

O tempo medido foi de 0,00062475s (mesmas condições do primeiro, sendo que o valor inicial foi de 0,00147600s). A tabela obtida por EXPLAIN apresenta resultados semelhantes, embora com valores muito diferentes nas linhas consultadas em RP: 200, ao todo:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	P	NULL	const	PRIMARY,userid	PRIMARY	8	const,const	1	100.00	Using temporary
1	SIMPLE	R	NULL	ref	PRIMARY,userid,userid_2	PRIMARY	4	const	3	10.00	Using where
1	SIMPLE	RP	NULL	ALL	NULL	NULL	NULL	NULL	200	1.00	Using where; Distinct; Using join buffer (Block N...

O índice para este caso específico terá de ter as seguintes características.

- Secundário, cujas chave de pesquisa é o atributo *regid* de *reg_pag*. Tal como no primeiro, como *regid* não é chave primária, este índice não poderá ser primário;
- Desagrupado, porque a ordem de registos de dados de *regid* difere da das entradas de dados;
- Denso, porque *regid* tem uma entrada de dados para cada valor que assume;
- Hash-based, pelas razões enunciadas anteriormente (recurso a interrogações).

Obter-se-á então o seguinte índice:

CREATE INDEX id_reg_pag2 USING HASH ON reg_pag(regid);

Com este índice, o número de linhas consultadas de RP diminui para 3:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	P	NULL	const	PRIMARY,userid	PRIMARY	8	const,const	1	100.00	Using temporary
1	SIMPLE	R	NULL	ref	PRIMARY,userid,userid_2	PRIMARY	4	const	3	10.00	Using where
1	SIMPLE	RP	NULL	ref	id_reg_pag2	id_reg_pag2	4	newproj.R.regcounter	1	10.00	Using where; Distinct

No que toca aos tempos, mediram-se valores abaixo dos 0,00040000s (considere-se uma medição oficial de 0,00038800s). Registaram-se assim melhorias no desempenho, com tempos de execução equivalentes a 66% dos originais.

6 – Transações

A funcionalidade pedida nesta secção é conseguida através do uso de 3 funções do PDO em PHP: *beginTransaction()*, antes das queries; *commit(),* após a execução das queries; e *rollBack(),* caso ocorra algum erro. De seguida, exemplifica-se esta funcionalidade, na inserção de uma nova página.

```
<?php
	$servername = "db.ist.utl.pt";
	$username = "ist178349";
	$password = "=luispedro9=";
	$dbname = "ist178349";
	$pagename = $_GET['pagename'];

	try {
		$db = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
		$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

		$db->beginTransaction();

		foreach($db->query("select userid from login where contador_login=(select max(contador_login) from login);") as $row) {
			$userid = $row['userid'];
		}

		foreach($db->query("select max(pagecounter) from pagina;") as $row) {
			$pagecounter = $row['max(pagecounter)'] + 1;
		}

		foreach($db->query("select max(S.contador_sequencia) from sequencia S, login L where L.contador_login=(select max(contador_login) from login) and S.userid=L.userid;") as $row) {
			$idseq = $row['max(S.contador_sequencia)'];
		}
```

```
                $stmt = $db->prepare("INSERT INTO pagina (userid, pagecounter, nome,
idseq, ativa) VALUES (?, ?, ? ,?, true)");
                $stmt->execute(array($userid, $pagecounter, $pagename, $idseq));

                $db->commit();

        }



        catch(PDOException $e){
                $db->rollBack();
                echo $sql . "<br>" . $e->getMessage();
        }
        $db = null;
?>
```

7 - Data Warehouse

Para este item, criámos três tabelas, a d_utilizador e a d_tempo, acrescentando um userid ao d_utilizador:

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.*

```
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,
FOREIGN KEY (userid) REFERENCES utilizador(userid),
UNIQUE INDEX email_UNIQUE (email)
);

CREATE TABLE IF NOT EXISTS d_tempo (
   dia TINYINT(31),
   mes TINYINT(12),
   ano INT NOT NULL,
);

CREATE TABLE IF NOT EXISTS login_fact (
   userid INT NOT NULL,
   dia TINYINT(31),
   mes TINYINT(12),
   ano INT NOT NULL,
   numero_logins INT NOT NULL,
PRIMARY KEY (userid),
);
```

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.*

```
SELECT AVG(numero_logins) FROM login_fact LF, d_utilizador U, d_tempo T
WHERE LF.userid = U.userid AND U.pais = 'Portugal'
GROUP BY ROLLUP (U.categoria, T.ano, T.mes);
```