TÉCNICO LISBOA

# Projeto de Bases de Dados

Parte 2


**Alunos:**

77956 - Pedro Oliveira

78058 - João Tiago

79756 - Luís Duarte


**Grupo:** 17 **Turno:** 4ª Feira 11:30h (BD817L05)

# Consultas SQL:

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

```
SELECT userid, email, nome, SUM(sucesso) SUCESSO,
        COUNT(sucesso) N_TENTATIVAS
FROM login NATURAL JOIN utilizador
GROUP BY userid, email, nome
HAVING SUM(sucesso) < COUNT(sucesso);
```

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

```
SELECT R.userid, R.typecounter, R.regcounter, R.nome
FROM registo R
WHERE NOT EXISTS (
        SELECT *
        FROM pagina P
        WHERE P.userid = R.userid
                AND NOT EXISTS (
                        SELECT *
                        FROM reg_pag RP
                        WHERE R.userid = RP.userid
                                AND R.typecounter = RP.typeid
                                AND R.regcounter = RP.regid
                                AND P.pagecounter = RP.pageid));
```

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

```
SELECT U.userid, U.nome, U.email, COUNT(RP.regid) AS numRegs,
        COUNT(DISTINCT P.pagecounter) AS numPags,
        COUNT(RP.regid)/COUNT(DISTINCT P.pagecounter) registosPorPagina
FROM utilizador U,
        pagina P LEFT JOIN reg_pag RP
                ON P.pagecounter=RP.pageid
                        AND P.userid=RP.userid
WHERE U.userid = P.userid
GROUP BY userid, nome, email
ORDER BY registosPorPagina DESC;
```

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

```
SELECT DISTINCT U.userid, U.email, U.nome
FROM utilizador U, tipo_registo A
WHERE U.userid = A.userid
        AND NOT EXISTS (
                SELECT *
                FROM tipo_registo T
                WHERE U.userid = T.userid
                        AND NOT EXISTS (
                                SELECT *
                                FROM reg_pag R
                                WHERE T.userid = R.userid
                                        AND T.typecnt = R.typeid));
```

# Restrições de Integridade

**(a) 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.**

```
DELIMITER $$
DROP FUNCTION IF EXISTS check_seq_func $$
CREATE FUNCTION check_seq_func(seq_id INT) RETURNS INT
BEGIN
        RETURN EXISTS (
                SELECT *
                FROM (SELECT idseq FROM tipo_registo
                        UNION
                        SELECT idseq FROM pagina
                        UNION
                        SELECT idseq FROM campo
                        UNION
                        SELECT idseq FROM registo
                        UNION
                        SELECT idseq FROM valor
                        UNION
                        SELECT idseq FROM reg_pag) A
                WHERE A.idseq=seq_id);
END$$
```

```
DROP TRIGGER IF EXISTS check_seq_tipo_registo $$
CREATE TRIGGER check_seq_tipo_registo
	BEFORE INSERT ON tipo_registo FOR EACH ROW
BEGIN
	IF (SELECT check_seq_func(NEW.idseq))
		THEN set NEW=null;
	END IF;
END$$


DROP TRIGGER IF EXISTS check_seq_pagina $$
CREATE TRIGGER check_seq_pagina
	BEFORE INSERT ON pagina FOR EACH ROW
BEGIN
	IF (SELECT check_seq_func(NEW.idseq))
		THEN set NEW=null;
	END IF;
END$$


DROP TRIGGER IF EXISTS check_seq_campo $$
CREATE TRIGGER check_seq_campo
	BEFORE INSERT ON campo FOR EACH ROW
BEGIN
	IF (SELECT check_seq_func(NEW.idseq))
		THEN set NEW=null;
	END IF;
END$$

DROP TRIGGER IF EXISTS check_seq_registo $$
CREATE TRIGGER check_seq_registo
	BEFORE INSERT ON registo FOR EACH ROW
BEGIN
	IF (SELECT check_seq_func(NEW.idseq))
		THEN set NEW=null;
	END IF;
END$$
```

```
DROP TRIGGER IF EXISTS check_seq_valor $$
CREATE TRIGGER check_seq_valor
        BEFORE INSERT ON valor FOR EACH ROW
BEGIN
        IF (SELECT check_seq_func(NEW.idseq))
                THEN set NEW=null;
        END IF;
END$$


DROP TRIGGER IF EXISTS check_seq_reg_pag $$
CREATE TRIGGER check_seq_reg_pag
        BEFORE INSERT ON reg_pag FOR EACH ROW
BEGIN
        IF (SELECT check_seq_func(NEW.idseq))
                THEN set NEW=null;
        END IF;
END$$
DELIMITER ;
```

# Desenvolvimento da aplicação

Código em anexo.

# Formas Normais

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

A relação utilizador encontra-se na 2ª Forma Normal. Isto deve-se à não existência de atributos multivalor (Forma Normal 1) e todos os atributos não chave dependerem funcionalmente da totalidade da chave (Forma Normal 2), isto é, para cada chave primária (email/userid), os atributos não chave (nome, password, questão1, resposta1, questão2, resposta2, pais, categoria) não têm mais que um valor e estes dependem da chave. Utilizador não se encontra na 3ª Forma Normal ou superior, uma vez que Resposta1 é dependente da Pergunta1 e Resposta2 é dependente da Pergunta2.

**(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?**

Continua na Forma Normal 2 devido ao mesmo problema enunciado na resposta da pergunta anterior.

**– 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.**

É preciso separar as perguntas e repostas da tabela do utilizador, passando a estar essa informação numa nova tabela:

**utilizador** (<u>userid</u>, email, nome, password, país, categoria)
**validacao** (<u>userid, questao</u>, resposta)

userid → nome, password, questão1, questão2, país, categoria
userid → email
email → userid
questão1 → resposta1
questão2 → resposta2

# Índices

**(a) Devolver a média do número de registos por página de um utilizador**
**Indíce:**

```
CREATE INDEX mediaRegPag
ON reg_pag (userid ASC, pageid ASC);
```

**Classificação do Índice:** Índice desagrupado e esparço.

**Query de teste:**

```
SELECT SQL_NO_CACHE U.userid, U.nome, U.email, COUNT(RP.regid) numRegs,
        COUNT(DISTINCT P.pagecounter) numPags,
        COUNT(RP.regid)/COUNT(DISTINCT P.pagecounter) RegistosPorPagina
FROM utilizador U,
        pagina P LEFT JOIN reg_pag RP
                ON P.pagecounter=RP.pageid
                        AND P.userid=RP.userid
WHERE U.userid = P.userid
GROUP BY userid, nome, email
ORDER BY 6 DESC;
```

**(b) Ver o nome dos registos associados à página de um utilizador**
**Indíce:**

```
CREATE INDEX nomesRegPag
        ON reg_pag (userid, regid);
```

**Classificação do Índice:** Índice desagrupado e esparço.

**Query de teste:**

```
SELECT SQL_NO_CACHE RP.userid, RP.pageid, RP.regid, R.nome Nome_Registo
FROM reg_pag RP, registo R
WHERE RP.userid=R.userid AND RP.regid=R.regcounter
ORDER BY RP.userid, RP.pageid, RP.regid;
```

# Transações

Código em anexo.

# 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, pais, categoria) e d_tempo(dia, mes, ano). Escreva as instruções SQL necessárias para carregar o esquema em estrela a partir das tabelas existentes.**

```
DROP TABLE IF EXISTS d_utilizador;
CREATE TABLE d_utilizador(
        email VARCHAR(255) NOT NULL,
        nome VARCHAR(255) NOT NULL,
        pais VARCHAR(45) NOT NULL,
        categoria VARCHAR(45) NOT NULL,
PRIMARY KEY (email)
);

DROP TABLE IF EXISTS d_tempo;
CREATE TABLE d_tempo(
        dia INT NOT NULL,
        mes INT NOT NULL,
        ano INT NOT NULL,
PRIMARY KEY (dia, mes, ano)
);
```

```
DROP TABLE IF EXISTS dataWarehouse_login;
CREATE TABLE dataWarehouse_login(
        email VARCHAR(255) NOT NULL,
        dia INT NOT NULL,
        mes INT NOT NULL,
        ano INT NOT NULL,
        numero_tentativas_login INT NOT NULL,
PRIMARY KEY (email, dia, mes, ano),
FOREIGN KEY (email) REFERENCES d_utilizador (email) ON DELETE CASCADE,
FOREIGN KEY (dia, mes, ano) REFERENCES d_tempo (dia, mes, ano)
);


INSERT d_utilizador SELECT email, nome, pais, categoria FROM utilizador;

INSERT d_tempo (
        SELECT DISTINCT DATE_FORMAT(moment, '%d'), DATE_FORMAT(moment, '%m'),
                DATE_FORMAT(moment, '%Y')
         FROM login);

INSERT dataWarehouse_login (
        SELECT B.email, DATE_FORMAT(A.moment, '%d'), DATE_FORMAT(A.moment, '%m'),
                DATE_FORMAT(A.moment, '%Y'), COUNT(*)
        FROM login A, utilizador B
        WHERE A.userid=B.userid
        GROUP BY A.moment, B.email);
```

**(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 U.categoria, U.email, D.mes, D.ano, D.Media_Tentativas_Login
FROM d_utilizador U,
        (SELECT A.email, A.mes, A.ano,
                AVG(A.numero_tentativas_login) Media_Tentativas_Login
        FROM dataWarehouse_login A, d_utilizador B
        WHERE A.email = B.email
        GROUP BY A.email, A.ano, A.mes WITH ROLLUP) D
WHERE U.email = D.email
        AND U.pais = 'Portugal'
ORDER BY U.categoria, U.email, -D.ano DESC, -D.mes DESC;
```