EXERCICIOS FUNCIÓNS E PROCEDEMENTOS

Índice

Exercicios funcións e procedementos almacenados MySQL.....3
1. Detalles do empregado por nome e apelidos.....3
2. Estatísticas salariales dun departamento.....4
3. Actualización ou inserción do título do empregado.....6
4. Aplicar bonus en función da antigüidade.....7
5. Función para calcular anos de servizo.....9
6. Listado de empregados con salarios por riba da media.....10
7. Axuste de salarios mediante bucle.....11
8. Función para obter o número de empregados dun departamento.....12
9. Transferir un empregado a outro departamento.....13
10. Informe completo de empregados con clasificación salarial.....14

Exercicios funcións e procedementos almacenados MySQL

1. Detalles do empregado por nome e apelidos

Crear un procedemento almacenado que, mediante parámetros de entrada, reciba o nome e apelidos dun empregado e devolva información detallada sobre el.

- **Parámetros de entrada:**
  - `p_first_name` (VARCHAR): Primeiro nome do empregado.
  - `p_last_name` (VARCHAR): Apelidos do empregado.
- **Parámetros de saída:**
  - `p_emp_no` (INT): Número do empregado.
  - `p_hire_date` (DATE): Data de contratación.
  - `p_current_salary` (INT): Salario actual (rexistro na táboa *salaries* con `to_date = '9999-01-01'`).
- **Restricións:**
  - Non se debe empregar JOIN; utilízanse consultas SELECT con variables locais para recoller intermedios.
  - Se non se atopa ningún empregado que corresponda aos criterios, establecer `p_emp_no` a un valor especial (p.ex. 0) e devolver unha mensaxe de erro mediante un SELECT final.

```
DELIMITER $$

CREATE PROCEDURE employee_Nombre_Apellidos(

IN p_first_name VARCHAR(50),
IN p_last_name VARCHAR(50),
OUT p_emp_no INT,
OUT p_hire_date DATE,
OUT p_current_salary INT)
BEGIN
DECLARE v_emp_no INT;
SELECT emp_no INTO v_emp_no
FROM employees
WHERE first_name = p_first_name AND last_name = p_last_name
LIMIT 1;
IF v_emp_no IS NULL THEN
SET p_emp_no = 0;
SELECT 'Error: Empleado no encontrado' AS message;
ELSE
SET p_emp_no = v_emp_no;
SELECT hire_date INTO p_hire_date FROM employees WHERE emp_no = p_emp_no;
SELECT salary INTO p_current_salary FROM salaries
WHERE emp_no = p_emp_no AND to_date = '9999-01-01';
END IF;
END $$
DELIMITER ;

call employees.employee_Nombre_Apellidos('Georgi', 'Facello', @p_emp_no, @p_hire_date,
@p_current_salary);
select @p_emp_no, @p_hire_date, @p_current_salary;
```

2. Estatísticas salariales dun departamento

Crear un procedemento chamado `department_salaryStats` que, dado o código dun departamento, calcule e devolva estatísticas básicas sobre os salarios dos empregados actualmente asignados.

- **Parámetro de entrada:**
  - `p_dept_no` (CHAR(4)): Código do departamento.
- **Parámetros de saída:**
  - `p_min_salary` (INT): Salario mínimo.
  - `p_max_salary` (INT): Salario máximo.
  - `p_avg_salary` (DECIMAL): Salario medio.
- **Implementación:**
  - Utilizar variables locais para almacenar o resultado das funcións de agregado (MIN, MAX, AVG) aplicadas á táboa *salaries* filtrando rexistros con `to_date = '9999-01-01'` e que o empregado apareza en *dept_emp* con o mesmo criterio.
  - Se o departamento non ten empregados asignados, devolver -1 en cada parámetro de saída.

```
DELIMITER //
CREATE PROCEDURE department_salaryStats(
    IN p_dept_no CHAR(4),
    OUT p_min_salary INT,
    OUT p_max_salary INT,
    OUT p_avg_salary DECIMAL(10,2)
)
BEGIN
    DECLARE minSalary INT;
    DECLARE maxSalary INT;
    DECLARE avgSalary DECIMAL(10,2);
    DECLARE employeeCount INT;
    SELECT COUNT(DISTINCT de.emp_no)
    INTO employeeCount
    FROM dept_emp de
    JOIN salaries s ON de.emp_no = s.emp_no
    WHERE de.dept_no = p_dept_no
    AND s.to_date = '9999-01-01';
    IF employeeCount = 0 THEN
        SET p_min_salary = -1;
        SET p_max_salary = -1;
        SET p_avg_salary = -1;
    ELSE
        SELECT MIN(s.salary), MAX(s.salary), AVG(s.salary)
        INTO minSalary, maxSalary, avgSalary
        FROM dept_emp de
        JOIN salaries s ON de.emp_no = s.emp_no
        WHERE de.dept_no = p_dept_no
        AND s.to_date = '9999-01-01';
        SET p_min_salary = minSalary;
        SET p_max_salary = maxSalary;
        SET p_avg_salary = avgSalary;
    END IF;
END //
DELIMITER ;


CALL department_salaryStats('d009', @p_min_salary, @p_max_salary, @p_avg_salary);
SELECT @p_min_salary AS MinSalary, @p_max_salary AS MaxSalary, @p_avg_salary AS
AvgSalary;
```

3. Actualización ou inserción do título do empregado

Crear un procedemento denominado `updateEmployeePosition` que permita actualizar o título actual dun empregado na táboa *titles* ou, se non existe un rexistro actual (con `to_date = '9999-01-01'`), inserir un novo rexistro.

- **Parámetros de entrada:**
  - `p_emp_no` (INT): Número do empregado.
  - `p_new_title` (VARCHAR): Novo título a asignar.
- **Parámetro de saída:**
  - `p_status` (INT): Valor 0 se a operación é exitosa ou -1 se ocorre algún erro.
- **Implementación:**
  - Verificar se existe un rexistro actual para o empregado na táboa *titles*.
  - Se existe, actualizar o campo `title`; se non, realizar unha inserción cunha data de inicio definida (por exemplo, a data actual) e `to_date = '9999-01-01'`.
  - Incorporar un bloque de xestión de erros (HANDLER) para capturar posíbeis excepciones na actualización ou inserción.

```
DELIMITER //
CREATE PROCEDURE updateEmployeePosition(
   IN p_emp_no INT,
   IN p_new_title VARCHAR(50),
   OUT p_status INT)
BEGIN
   DECLARE currentTitleExists INT DEFAULT 0;
   DECLARE EXIT HANDLER FOR SQLEXCEPTION
   BEGIN
      SET p_status = -1;
   END;
   SELECT COUNT(*)
   INTO currentTitleExists
   FROM titles
   WHERE emp_no = p_emp_no AND to_date = '9999-01-01';
   IF currentTitleExists > 0 THEN
      UPDATE titles
      SET title = p_new_title
      WHERE emp_no = p_emp_no AND to_date = '9999-01-01';
      SET p_status = 0;
   ELSE
      INSERT INTO titles (emp_no, title, from_date, to_date)
      VALUES (p_emp_no, p_new_title, CURDATE(), '9999-01-01');
      SET p_status = 0;
   END IF;
END //
DELIMITER ;


CALL updateEmployeePosition(10001, 'Senior Engineer', @p_status);
SELECT @p_status AS Status;
```

4. Aplicar bonus en función da antigüidade

Obxectivo:
Crear un procedemento chamado `calculateBonus` que, dado o número dun empregado e unha cantidade de bonus, verifique se o empregado ten máis de 10 anos de antigüidade e, se é o caso, engada o bonus ao seu salario actual.

- **Parámetros de entrada:**
  - `p_emp_no` (INT): Número do empregado.
  - `p_bonus` (INT): Cantidade de bonus a aplicar.
- **Parámetro de saída:**
  - `p_new_salary` (INT): O novo salario despois de aplicar o bonus, ou -1 se o empregado non cumpre os criterios.
- **Implementación:**
  - Utilizar a función `TIMESTAMPDIFF` para calcular os anos de servizo a partir de `hire_date` na táboa *employees*.
  - Se o empregado ten 10 anos ou máis, actualizar o rexistro actual en *salaries* (con `to_date = '9999-01-01'`) engadindo o bonus.
  - Devolver o novo salario ou -1 se non se aplica o bonus.

```
DELIMITER //
CREATE PROCEDURE calculateBonus(
    IN p_emp_no INT,
    IN p_bonus INT,
    OUT p_new_salary INT)
BEGIN
    DECLARE yearsOfService INT DEFAULT 0;
    DECLARE currentSalary INT DEFAULT 0;
    DECLARE employeeExists INT DEFAULT 0;
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SET p_new_salary = -1;
    END;
    SELECT COUNT(*)
    INTO employeeExists
    FROM employees
    WHERE emp_no = p_emp_no;
    IF employeeExists = 0 THEN
        SET p_new_salary = -1;
    ELSE
        SELECT TIMESTAMPDIFF(YEAR, hire_date, CURDATE())
        INTO yearsOfService
        FROM employees
        WHERE emp_no = p_emp_no;
        IF yearsOfService < 10 THEN
            SET p_new_salary = -1;
        ELSE
            SELECT salary
            INTO currentSalary
            FROM salaries
            WHERE emp_no = p_emp_no AND to_date = '9999-01-01';
            UPDATE salaries
            SET salary = currentSalary + p_bonus
            WHERE emp_no = p_emp_no AND to_date = '9999-01-01';
            SELECT salary
            INTO p_new_salary
            FROM salaries
            WHERE emp_no = p_emp_no AND to_date = '9999-01-01';
        END IF;
    END IF;
END //
DELIMITER ;


CALL calculateBonus(10001, 5000, @p_new_salary);
SELECT @p_new_salary AS NewSalary;
```

5. Función para calcular anos de servizo

Crear unha función almacenada chamada `employee_yearsOfService` que, dado o número dun empregado, devolva o número de anos que lle leva na empresa.

- **Parámetro de entrada:**
  - `p_emp_no` (INT): Número do empregado.
- **Valor de retorno:**
  - Anos de servizo (INT): Calculados como a diferenza en anos entre a data actual e o campo `hire_date` da táboa *employees*.
- **Implementación:**
  - Se o empregado non existe, a función devolverá -1.
  - Utilizar a función `CURDATE()` e `TIMESTAMPDIFF`.

```
DELIMITER //
CREATE FUNCTION employee_yearsOfService(
    p_emp_no INT
) RETURNS INT
READS SQL DATA
BEGIN
    DECLARE yearsOfService INT;
    DECLARE employeeExists INT;
    SELECT COUNT(*)
    INTO employeeExists
    FROM employees
    WHERE emp_no = p_emp_no;

    IF employeeExists = 0 THEN
        RETURN -1;
    ELSE
        SELECT TIMESTAMPDIFF(YEAR, hire_date, CURDATE())
        INTO yearsOfService
        FROM employees
        WHERE emp_no = p_emp_no;
        RETURN yearsOfService;
    END IF;
END //
DELIMITER ;


SELECT employee_yearsOfService(10001) AS YearsOfService;
```

6. Listado de empregados con salarios por riba da media

Crear un procedemento chamado `listHighEarners` que recolla e amose os empregados que teñen un salario superior á media xeral de todos os salarios.

- **Sen parámetros de entrada ou saída.**
- **Implementación:**
  - Primeiro, calcular a media dos salarios actuais (utilizando unha consulta SELECT con AVG na táboa *salaries* onde `to_date = '9999-01-01'`).
  - Utilizar un cursor para recorrer os rexistros dos empregados que teñen un salario superior a esa media.
  - Para cada empregado, amosar o número de empregado e o salario mediante un SELECT.
  - Incluir a xestión de fin de cursor mediante `CONTINUE HANDLER FOR NOT FOUND`.

```
DELIMITER //
CREATE PROCEDURE listHighEarners()
BEGIN
    DECLARE currentEmpNo INT;
    DECLARE currentSalary INT;
    DECLARE averageSalary DECIMAL(10, 2);
    DECLARE finished INT DEFAULT 0;
    DECLARE highEarnersCursor CURSOR FOR
        SELECT emp_no, salary
        FROM salaries
        WHERE to_date = '9999-01-01' AND salary > averageSalary;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;
    SELECT AVG(salary)
    INTO averageSalary
    FROM salaries
    WHERE to_date = '9999-01-01';
    OPEN highEarnersCursor;
    read_loop: LOOP
        FETCH highEarnersCursor INTO currentEmpNo, currentSalary;
        IF finished = 1 THEN
            LEAVE read_loop;
        END IF;
        SELECT currentEmpNo AS EmployeeNumber, currentSalary AS Salary;
    END LOOP;
    CLOSE highEarnersCursor;
END //
DELIMITER ;

CALL listHighEarners();
```

7. Axuste de salarios mediante bucle

Obxectivo:
Crear un procedemento chamado `adjustSalariesByPercentage` que aplique un incremento percentual a todos os salarios actuais.

- **Parámetro de entrada:**
  - `p_percentage` (DECIMAL): Porcentaxe a incrementar (por exemplo, 10 para un 10% de aumento).
- **Parámetro de saída (opcional):**
  - Mensaxe de confirmación ou de erro.
- **Implementación:**
  - Se `p_percentage` é negativo ou maior que 50, o procedemento debe saír inmediatamente sen facer modificacións e devolver unha mensaxe de erro.
  - De outro xeito, utilizar un bucle WHILE para obter mediante un SELECT os IDs dos empregados (ou recoller nunha variable tipo cursor ou mediante un array de IDs) e actualizar individualmente cada salario actual (na táboa *salaries* con `to_date = '9999-01-01'`) incrementando o valor polo porcentaje indicado.
  - Ao final, devolver unha mensaxe de éxito.

```
DELIMITER //
CREATE PROCEDURE adjustSalariesByPercentage(
    IN p_percentage DECIMAL())
BEGIN

DELIMITER ;

CALI ;
```

8. Función para obter o número de empregados dun departamento

Obxectivo:
Crear unha función almacenada denominada `getDepartmentEmployeeCount` que, dado o código dun departamento, devolva o número de empregados actualmente asignados.

- **Parámetro de entrada:**
  - `p_dept_no` (CHAR(4)): Código do departamento.
- **Valor de retorno:**
  - Número de empregados (INT) que aparecen na táboa *dept_emp* cun rexistro actual (`to_date = '9999-01-01'`).
- **Implementación:**
  - Se o departamento non existe ou non ten empregados, devolver 0.
  - Utilizar a función COUNT(*) nunha consulta SELECT.

```
DELIMITER $$
CREATE FUNCTION getDepartmentEmployeeCount( p_dept_no CHAR(4) )
RETURNS INT
DETERMINISTIC
BEGIN
   DECLARE emp_count INT;
   SELECT COUNT(*) INTO emp_count
   FROM dept_emp
   WHERE dept_no = p_dept_no
     AND to_date = '9999-01-01';
   IF emp_count IS NULL THEN
      SET emp_count = 0;
   END IF;
   RETURN emp_count;
END$$
DELIMITER ;

SELECT getDepartmentEmployeeCount('d004');
```

9. Transferir un empregado a outro departamento

Crear un procedemento almacenado chamado `transferEmployee` que permita trasladar un empregado dun departamento a outro, actualizando a táboa *dept_emp*.

- **Parámetros de entrada:**
  - `p_emp_no` (INT): Número do empregado.
  - `p_new_dept_no` (CHAR(4)): Código do novo departamento.
- **Parámetro de saída:**
  - `p_message` (VARCHAR): Mensaxe que indique se a transferencia foi exitosa ou se se produciu algún erro (por exemplo, empregado ou departamento non existente).
- **Implementación:**
  - Actualizar o rexistro actual en *dept_emp* definindo unha data de fin (por exemplo, a data actual).
  - Inserir un novo rexistro en *dept_emp* para o empregado, con a data de inicio como a data actual e `to_date = '9999-01-01'`.
  - Incorporar comprobación da existencia do empregado e do departamento, e xestionar os erros correspondentes.

```
DELIMITER $$
CREATE PROCEDURE transferEmployee(
   IN p_emp_no INT,
   IN p_new_dept_no CHAR(4),
   OUT p_message VARCHAR(255))
BEGIN
   DECLARE emp_exists INT DEFAULT 0;
   DECLARE dept_exists INT DEFAULT 0;
   DECLARE current_date_var DATE;
   SET current_date_var = CURDATE();
   SELECT COUNT(*) INTO emp_exists
   FROM employees
   WHERE emp_no = p_emp_no;
   SELECT COUNT(*) INTO dept_exists
   FROM departments
   WHERE dept_no = p_new_dept_no;
         IF emp_exists = 0 THEN
      SET p_message = 'Error: O empregado non existe.';
   ELSEIF dept_exists = 0 THEN
      SET p_message = 'Error: O departamento non existe.';
   ELSE
      UPDATE dept_emp
      SET to_date = current_date_var
      WHERE emp_no = p_emp_no AND to_date = '9999-01-01';
      INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
      VALUES (p_emp_no, p_new_dept_no, current_date_var, '9999-01-01');
        SET p_message = CONCAT('Transferencia exitosa. O empregado ', p_emp_no, ' foi
trasladado ao departamento ', p_new_dept_no, '.');
   END IF;
END $$
DELIMITER ;

CALL transferEmployee(10001, 'd004', @message);
SELECT @message;
```

10. Informe completo de empregados con clasificación salarial

Crear un procedemento chamado `generateEmployeeReport` que xere un informe consolidado cos datos dos empregados, integrando información de varias táboas.

- **Sen parámetros de entrada ou saída (a saída será mediante un SELECT final).**
- **O informe debe incluír:**
  - Número do empregado.
  - Nome completo (concatenación de `first_name` e `last_name`).
  - Nome do departamento (obtenido da táboa *departments* a través de *dept_emp*).
  - Salario actual (de *salaries* con `to_date = '9999-01-01'`).
  - Unha clasificación do salario empregando a instrución CASE:
    - Se o salario é menor de 30000 → 'BAIXO'.
    - Se o salario está entre 30000 e 60000 → 'MEDIO'.
    - Se o salario é superior a 60000 → 'ALTO'.
- **Implementación:**
  - Realizar as combinacións necesarias entre as táboas *employees*, *dept_emp*, *departments* e *salaries*.
  - Pode usarse variables locais se se desexa procesar algún dato intermedio, e a instrución CASE debe formar parte do SELECT final.
  - O resultado final debe ser amosado nun SELECT con columnas ben identificadas.

```
DELIMITER $$
CREATE PROCEDURE generateEmployeeReport()
BEGIN
   SELECT
      e.emp_no AS "Número de Empregado",
      CONCAT(e.first_name, ' ', e.last_name) AS "Nome Completo",
      d.dept_name AS "Departamento",
      s.salary AS "Salario Actual",
      CASE
         WHEN s.salary < 30000 THEN 'BAIXO'
         WHEN s.salary BETWEEN 30000 AND 60000 THEN 'MEDIO'
         WHEN s.salary > 60000 THEN 'ALTO'
      END AS "Clasificación Salarial"
   FROM
      employees e
   JOIN
      dept_emp de ON e.emp_no = de.emp_no
   JOIN
      departments d ON de.dept_no = d.dept_no
   JOIN
      salaries s ON e.emp_no = s.emp_no
   WHERE
      s.to_date = '9999-01-01';
END $$
DELIMITER ;

CALL generateEmployeeReport();
```