

UD4 – Examen (Modelo B)

LEE LAS INSTRUCCIONES ATENTAMENTE. DEBERÁS ENTREGAR ESTE MISMO DOCUMENTO EN FORMATO PDF (CUALQUIER OTRO FORMATO DISTINTO DE PDF NO SERÁ CORREGIDO). SE ESPERA QUE PUEDAS RESOLVER ESTAS ACTIVIDADES EMPLEANDO EXCLUSIVAMENTE TUS CONOCIMIENTOS Y LOS MATERIALES QUE HAYAS DESCARGADO PREVIAMENTE. EN CASO DE CUALQUIER EVIDENCIA DE COPIA, EL EXAMEN SE EVALUARÁ COMO SUSPENSO (0).

PARA ESTE EXAMEN, REUTILIZAREMOS EL SCRIPT QUE USAMOS PARA LA ACTIVIDAD 3.5 QUE PUEDES ENCONTRAR EN MOODLE. POR LO TANTO, NO OLVIDES CORREGIR EL ERROR (SUGERENCIA: ORDEN INCORRECTO PARA LOS INSERTS).

Pregunta 1. Para empezar, vamos a crear algunos usuarios y unas vistas (3 puntos):

1. Crea las siguientes cuentas de usuario. (0,5 puntos)
 - a. Usuario: alan | Contraseña: turing
 - b. Usuario: steve | Contraseña: wozniak

```
CREATE USER 'alan'@'localhost' IDENTIFIED BY 'alan';
CREATE USER 'steve'@'localhost' IDENTIFIED BY 'wozniak';
```

```
✓ 17 14:36:22 CREATEUSER'alan'@'localhost'IDENTIFIEDBY'alan'
✓ 18 14:36:22 CREATEUSER'steve'@'localhost'IDENTIFIEDBY'wozniak'
```

2. Crea las siguientes vistas. (2 puntos)
 - a. Crea una vista llamada pobreza que muestra el nombre, apellidos y salario del empleado mejor pagado en ese momento.

```
CREATE VIEW pobreza AS
SELECT e.first_name, e.last_name, s.salary
FROM employees e
JOIN salaries s ON e.emp_no = s.emp_no
WHERE s.to_date = '9999-01-01'
ORDER BY s.salary DESC
LIMIT 1;
```



```
✓ 35 14:39:55 CREATE VIEW pobreza AS SELECT e.first_name, e.last_name, s.salary FROM employees e JOIN salaries s ON e.emp_no = s.emp_no WHERE s.to_da...
```

Result Grid			
	first_name	last_name	salary
▶	Charlene	Brattka	113229

- b. Crea una vista llamada RRHH que muestre el nombre y apellido de todos los empleados que actualmente trabajen en dicho departamento.

```
CREATE VIEW RRHH AS
SELECT e.first_name, e.last_name
FROM employees e
JOIN dept_emp de ON e.emp_no = de.emp_no
JOIN departments d ON de.dept_no = d.dept_no
WHERE d.dept_name = 'Human Resources';
```

✓ 89 14:42:56 CREATE VIEW RRHH AS SELECT e.first_name, e.last_name FROM employees e JOIN dept_emp de ON e.emp_no = de.emp_no JOIN departments d...

Result Grid   Filter Rows		
	first_name	last_name
▶	Kyoichi	Maliniak
	Eberhardt	Terkki
	Adamantios	Portugali
	Alejandro	Brender
	Mayumi	Schueller
	Hisao	Lipner
	Mona	Azuma
	Premal	Baek
	Somnath	Foote
	Hironobu	Haraldson

3. Otorga a los usuarios los siguientes privilegios. (0,5 puntos)
- a. alan – todos los privilegios para ambas vistas.

```
GRANT ALL PRIVILEGES ON pobreza TO 'alan'@'localhost';
GRANT ALL PRIVILEGES ON RRHH TO 'alan'@'localhost';
```

✓ 144 14:44:35 GRANT ALL PRIVILEGES ON pobreza TO 'alan'@'localhost'

✓ 145 14:44:35 GRANT ALL PRIVILEGES ON RRHH TO 'alan'@'localhost'

- b. steve – sólo se le permite consultar sobre la tabla de empleados, sin embargo, debe poder modificar la columna last_name.

```
GRANT SELECT (emp_no, first_name, last_name, gender, hire_date, birth_date) ON employees TO 'steve'@'localhost';
GRANT UPDATE (last_name) ON employees TO 'steve'@'localhost';
```

✓ 164 14:45:40 GRANT SELECT (emp_no, first_name, last_name, gender, hire_date, birth_date) ON employees TO 'steve'@'localhost'

✓ 165 14:45:40 GRANT UPDATE (last_name) ON employees TO 'steve'@'localhost'

No olvides mostrar tu código y capturas de pantalla de sus resultados, incluido el código necesario para lograr la funcionalidad.

Pregunta 2. Sigamos. A continuación, encontrarás 5 enunciados de scripts diferentes para implementar. **Elige sólo 3 de ellos** e implementa esos 3. (4,5 puntos)

1. **Enchufados.** Crea una función que busque el primer salario de un empleado dado su emp_no y retorne un mensaje de error si dicha cuantía es inferior a \$100.000. En caso contrario, debe devolver el primer salario del empleado. (1,5 puntos)

```
DELIMITER $$
CREATE FUNCTION primer_salario(emp INT) RETURNS INT
DETERMINISTIC
BEGIN
    declare salario INT;

    -- Obtener el slario mas antiguo

    SELECT salary INTO salario
    FROM salaries
    WHERE emp_no = emp
    ORDER BY from_date ASC
    LIMIT 1;

    IF salario < 100000 THEN
        SIGNAL SQLSTATE '45000'
        SET message_text = 'Salario mas de 100.000';
    END IF;

    RETURN salario;
END $$
```

2. **Reducción de jornada.** Crea un procedimiento que baje el salario un 40% a un trabajador dado su emp_no. El procedimiento debe asegurarse de cerrar correctamente el registro del salario anterior y abrir un nuevo registro para el nuevo salario. (1,5 puntos)

```

CREATE PROCEDURE reducir_jornada(IN emp INT)
BEGIN
    DECLARE salario_actual INT;
    DECLARE fecha_actual DATE;

    SET fecha_actual = CURDATE();

    -- bbtener salario actual
    SELECT salary INTO salario_actual
    FROM salaries
    WHERE emp_no = emp AND to_date = '9999-01-01';

    -- cerrar salario anterior
    UPDATE salaries
    SET to_date = fecha_actual
    WHERE emp_no = emp AND to_date = '9999-01-01';

    -- insertar nuevo salario reducido
    INSERT INTO salaries (emp_no, salary, from_date, to_date)
    VALUES (emp, salario_actual * 0.6, fecha_actual, '9999-01-01');
END $$

```

3. **La extra.** Crea un trigger que evite que un empleado tenga dos salarios superpuestos en salaries. Si un nuevo salario se solapa en fechas con otro salario existente para el mismo empleado, muestra un mensaje de error. (1,5 puntos)

```

CREATE TRIGGER evitar_solapamiento_salarios
BEFORE INSERT ON salaries
FOR EACH ROW
BEGIN
    IF EXISTS (
        SELECT 1
        FROM salaries
        WHERE emp_no = NEW.emp_no
        AND (
            NEW.from_date BETWEEN from_date AND to_date OR
            NEW.to_date BETWEEN from_date AND to_date OR
            from_date BETWEEN NEW.from_date AND NEW.to_date
        )
    ) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Ya existe un salario en ese rango de fechas';
    END IF;
END $$

```

4. **Migajas.** Crea un trigger que evite que un empleado pueda recibir un nuevo salario que sea un 5% más alto que su último salario, mostrando un mensaje de error en tal caso.

(1,5 puntos)

```
DELIMITER $$
CREATE TRIGGER evitar_aumento_excesivo
BEFORE INSERT ON salaries
FOR EACH ROW
BEGIN
    DECLARE ultimo_salario INT;

    SELECT salary INTO ultimo_salario
    FROM salaries
    WHERE emp_no = NEW.emp_no
    ORDER BY to_date DESC
    LIMIT 1;

    IF NEW.salary > ultimo_salario * 1.05 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'El nuevo salario excede el aumento permitido (5%)';
    END IF;
END $$
```

5. **Las mujeres facturan.** Escribe un procedimiento almacenado que devuelva una lista de las empleadas (mujeres) que actualmente cobran más una determinada cuantía, la cual se pasa como parámetro. (1,5 puntos)

```
DELIMITER $$
CREATE PROCEDURE mujeres_facturan(IN cantidad DECIMAL(10,2))
BEGIN
    SELECT e.emp_no, e.first_name, e.last_name, s.salary
    FROM employees e
    JOIN salaries s ON e.emp_no = s.emp_no
    WHERE e.gender = 'F'
        AND s.to_date = '9999-01-01'
        AND s.salary > cantidad;
END $$
```

No olvides mostrar tu código y capturas de pantalla de sus resultados, incluido el código necesario para mostrar la funcionalidad.

Pregunta 3. Nuestro superintendente quiere que cada vez que un empleado reciba un nuevo título, se registren el emp_no, la fecha del nombramiento, el nuevo título y, especialmente, el emp_no del manager del departamento en una tabla llamada "titulines". Considera las siguientes preguntas. (2,5 puntos)

1. Para lograr la funcionalidad requerida, ¿qué herramienta sería la más apropiada en este caso? ¿Trigger, función o procedimiento? Justifica tu respuesta. (1 punto)

La herramienta es un TRIGGER, ya que queremos registrar automáticamente cada nuevo título en la tabla titulines en el momento en que se inserta en la tabla titles. Los triggers permiten ejecutar lógica reactiva ante operaciones INSERT, UPDATE o DELETE.

2. Desarrolla la funcionalidad solicitada y demuestra su funcionamiento. (1,5 puntos)

```
CREATE TABLE titulines (
  emp_no INT,
  fecha_nom DATE,
  titulo VARCHAR(50),
  manager_no INT
)
;

DELIMITER $$
CREATE TRIGGER log_titulo
AFTER INSERT ON titles
FOR EACH ROW
BEGIN
  DECLARE dept_id CHAR(4);
  DECLARE manager_id INT;

  -- obtener el departamento actual del empleado
  SELECT dept_no INTO dept_id
  FROM dept_emp
  WHERE emp_no = NEW.emp_no AND to_date = '9999-01-01';

  -- obtener el manager del departamento
  SELECT emp_no INTO manager_id
  FROM dept_manager
  WHERE dept_no = dept_id AND to_date = '9999-01-01';

  -- insertar en titulines
  INSERT INTO titulines (emp_no, fecha_nom, titulo, manager_no)
  VALUES (NEW.emp_no, NEW.from_date, NEW.title, manager_id);
END $$
```

NO OLVIDES AÑADIR CAPTURAS DE PANTALLA DE TU CÓDIGO, ASÍ COMO DE TODOS TUS RESULTADOS