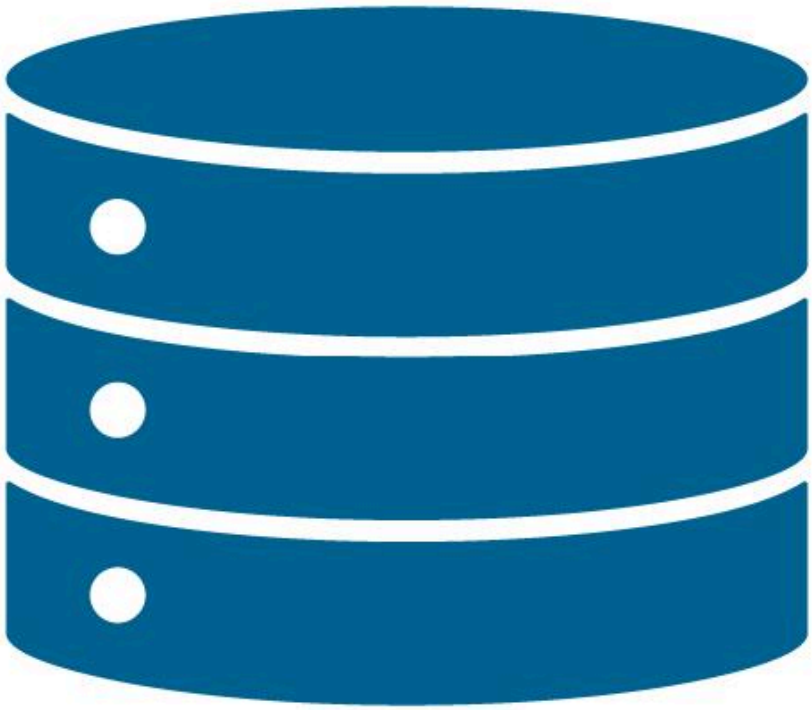

Trabajo Práctico N°2

—

Integrantes:

- Rea, Alejandro
- Coronel, Maximiliano
- Sosa, Matías
- Díaz, Lucas
- Bruno, Leonel

- **Actividad 1**

```
DROP PROCEDURE IF EXISTS pa_empleados_getSueldo;

DELIMITER //


CREATE PROCEDURE pa_empleados_getSueldo ( IN numero INT, OUT nombre
VARCHAR(50), OUT sueldo DECIMAL)


BEGIN


 SELECT e.nombre, s.sueldo INTO nombre, sueldo FROM empleados e JOIN sueldos s ON
e.empleado_id = s.empleado_id
 WHERE e.empleado_id = numero ;


 END //


 DELIMITER ;


 CALL pa_empleados_getSueldo(10007, @nombre, @sueldo);
 SELECT @nombre, @sueldo;
```

- **Actividad 2**

```
DELIMITER //
CREATE PROCEDURE pa_empresa_getMaxSueldo(IN valor INT)
BEGIN
   DECLARE nombre_completo VARCHAR(100);


   SELECT CONCAT(e.nombre, ' ', e.apellido)
   INTO nombre_completo
```

```
    FROM sueldos s
    JOIN empleados e ON s.empleado_id = e.empleado_id
    WHERE s.sueldo = valor
    LIMIT 1;

    IF nombre_completo IS NOT NULL THEN
        SELECT nombre_completo AS 'Empleado con ese sueldo';
    ELSE
        SELECT 'NO EXISTE' AS mensaje;
    END IF;
END //
DELIMITER ;
CALL pa_empresa_getMaxSueldo(45000);
```

- **Actividad 3**

```
DELIMITER $$
CREATE PROCEDURE pa_empleados_Add(IN p_nombre VARCHAR(50), IN p_apellido
VARCHAR(50), IN p_sueldo DECIMAL(10,2), IN p_departamento_id INT)
BEGIN
DECLARE existe_dep INT;
DECLARE error_ocurrido INT DEFAULT 0;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET error_ocurrido = 1;

SELECT COUNT(*) INTO existe_dep FROM departamentos WHERE id = p_departamento_id;

START TRANSACTION;
    IF existe_dep = 0 THEN
        ROLLBACK;
```

```
        SELECT 'Error: El departamento no existe' AS mensaje;
    ELSE
        INSERT INTO empleados (nombre, apellido, sueldo, departamento_id)
        VALUES (p_nombre, p_apellido, p_sueldo, p_departamento_id);
    END IF;

    IF error_ocurrido = 1 THEN
        ROLLBACK;
        SELECT 'Error: Clave duplicada o fallo en inserción' AS mensaje;
    ELSE
        COMMIT;
        SELECT 'Empleado agregado correctamente' AS mensaje;
    END IF;
END$$
DELIMITER ;
CALL pa_empleados_Add()
```

- **Actividad 4:**

Se agregó una columna en departamentos que haga referencia al empleado que es jefe, para probar:

```
ALTER TABLE departamentos ADD COLUMN id_empleado INT;
ALTER TABLE departamentos ADD CONSTRAINT fk_jefe FOREIGN KEY (id_empleado) REFERENCES empleados(empleado_id);
UPDATE departamentos SET id_empleado = 10001 WHERE departamento_id = "d001";
UPDATE departamentos SET id_empleado = 10005 WHERE departamento_id = "d002";
UPDATE departamentos SET id_empleado = 10007 WHERE departamento_id = "d003";
UPDATE departamentos SET id_empleado = 10003 WHERE departamento_id = "d004";
DELIMITER $$

CREATE PROCEDURE pa_staff_getJefes(IN p_fecha DATE, OUT numJefes INT)
```

```
BEGIN
SELECT COUNT(*) INTO numJefes
FROM empleados e
INNER JOIN departamentos d ON d.id_empleado = e.empleado_id
WHERE DATEDIFF(p_fecha, fecha_contratacion) > (10 * 365);
END$$

DELIMITER ;
call pa_staff_getJefes('2024-12-15', @numJefes);
SELECT @numJefes;
```

- **Actividad 5**

```
DROP PROCEDURE IF EXISTS pa_departamentos_getNombre;
DELIMITER //
CREATE PROCEDURE pa_departamentos_getNombre (IN p_numero INT, OUT
nomDepa_idDepa VARCHAR(300))
BEGIN
IF p_numero BETWEEN 1 AND 10 then
SELECT group_concat(concat(departamento_id, '-', nombre_departamento) SEPARATOR ', ')
INTO nomDepa_idDepa FROM
 (SELECT departamento_id, nombre_departamento
 FROM departamentos
 LIMIT p_numero )
 AS sub;

END if ;

END //
DELIMITER ;
```

```
CALL pa_departamentos_getNombre (1, @nom_id);
SELECT @nom_id;
```

- **Actividad 6**

```
DELIMITER //
CREATE PROCEDURE pa_departamentos_getNumEmpleados(IN dept1 CHAR(4), IN dept2
CHAR(4))
BEGIN
   DECLARE nombre1 VARCHAR(40);
   DECLARE nombre2 VARCHAR(40);
   DECLARE total1 INT DEFAULT 0;
   DECLARE total2 INT DEFAULT 0;


   SELECT nombre_departamento INTO nombre1
   FROM departamentos
   WHERE departamento_id = dept1;


   SELECT nombre_departamento INTO nombre2
   FROM departamentos
   WHERE departamento_id = dept2;


   IF nombre1 IS NULL OR nombre2 IS NULL THEN
      SELECT 'Ingreso incorrecto' AS mensaje;
   ELSE
```

```
        SELECT COUNT(*) INTO total1
        FROM emp_dept
        WHERE departamento_id = dept1;

        SELECT COUNT(*) INTO total2
        FROM emp_dept
        WHERE departamento_id = dept2;



        SELECT CONCAT(nombre1, ': ', total1, ' empleados') AS info;
        SELECT CONCAT(nombre2, ': ', total2, ' empleados') AS info;
    END IF;
END //
DELIMITER ;
```

– Actividad 7

```
DROP PROCEDURE IF EXISTS pa_departamentos_cargarResumen;
DELIMITER $$

CREATE PROCEDURE pa_departamentos_cargarResumen(IN min_empleados INT)
BEGIN
  DECLARE fin_cursor BOOLEAN DEFAULT FALSE;
  DECLARE v_nombre_departamento VARCHAR(100);
  DECLARE v_num_empleados INT;

  DECLARE c CURSOR FOR
    SELECT d.nombre_departamento, COUNT(e.empleado_id) AS num_empleados
    FROM departamentos d
```

```
    INNER JOIN emp_dept ed ON d.departamento_id = ed.departamento_id
    INNER JOIN empleados e ON ed.empleado_id = e.empleado_id
    GROUP BY d.departamento_id;

  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
    BEGIN
      SELECT 'Ocurrió un error durante la ejecución del procedimiento.' AS mensaje_error;
    END;

  DECLARE CONTINUE HANDLER FOR NOT FOUND SET fin_cursor = TRUE;

  DROP TEMPORARY TABLE IF EXISTS tmp_resumen_departamentos;
  CREATE TEMPORARY TABLE tmp_resumen_departamentos (nombre VARCHAR(100),
num_empleados INT);
  OPEN c;

FETCH c INTO v_nombre_departamento, v_num_empleados;

WHILE NOT fin_cursor DO

  IF v_num_empleados > min_empleados THEN
    INSERT INTO tmp_resumen_departamentos(nombre, num_empleados)
    VALUES (v_nombre_departamento, v_num_empleados);
  END IF;

  FETCH c INTO v_nombre_departamento, v_num_empleados;

END WHILE;

CLOSE c;
```

```
SELECT * FROM tmp_resumen_departamentos;

END $$

DELIMITER ;

CALL pa_departamentos_cargarresumen(3);
```

- **Actividad 8**

```
DROP PROCEDURE IF EXISTS pa_departamentos_Add;

DELIMITER //

CREATE PROCEDURE pa_departamentos_Add(INOUT cantidad INT)
BEGIN
  DECLARE total_actual INT;
  DECLARE contador INT;
  DECLARE max_departamentos INT DEFAULT 990;
  DECLARE msg TEXT;

  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
    BEGIN
      SET msg = 'Clave duplicada';
      SELECT msg AS mensaje;
      SET cantidad = -1;
    END;

  etiqueta_proc: BEGIN
```

```
    SELECT COUNT(*) INTO total_actual FROM departamentos;
    IF total_actual + cantidad > max_departamentos THEN
      SET cantidad = -1;
      LEAVE etiqueta_proc;
    END IF;

    SET contador = total_actual + 1;

    WHILE cantidad > 0 DO
      INSERT INTO departamentos(departamento_id, nombre_departamento)
      VALUES (CONCAT('d', LPAD(contador, 3, '0')), CONCAT('Dept. creado
manualmente ', ' ', contador));

      SET contador = contador + 1;
      SET cantidad = cantidad - 1;
    END WHILE;

    SELECT 'Alta exitosa' AS mensaje;

  END etiqueta_proc;
END //

DELIMITER ;

SET @x = 10;
CALL pa_departamentos_add(@x);
```

- **Actividad 9.**

```
DELIMITER $$
CREATE PROCEDURE pa_crear_email(nombre VARCHAR(50), apellido
VARCHAR(50), dominio VARCHAR(20))
BEGIN
        DECLARE nuevo_mail TEXT;
   SET nuevo_mail = lower(fn_eliminar_acentos(concat(apellido, "_", nombre, "@",
dominio)));
   SELECT nuevo_mail;
END $$
DELIMITER ;

CALL pa_crear_email("Álejándró", "ahí", "gmaíl.com");

– Función
DELIMITER $$

CREATE FUNCTION fn_eliminar_acentos(cadena VARCHAR(50))
RETURNS VARCHAR(50) DETERMINISTIC
BEGIN
   DECLARE i INT DEFAULT 1;
   DECLARE resultado VARCHAR(50) DEFAULT '';
   DECLARE letra CHAR(1);

   WHILE i <= LENGTH(cadena) DO
      SET letra = SUBSTRING(cadena, i, 1);

      SET letra = CASE
         WHEN letra IN ('á', 'Á') THEN 'a'
         WHEN letra IN ('é', 'É') THEN 'e'
         WHEN letra IN ('í', 'Í') THEN 'i'
```

```
            WHEN letra IN ('ó', 'Ó') THEN 'o'
            WHEN letra IN ('ú', 'Ú') THEN 'u'
            ELSE letra
        END;

        SET resultado = CONCAT(resultado, letra);
        SET i = i + 1;
    END WHILE;

    RETURN resultado;
END;
DELIMITER ;

-- Trigger
DELIMITER $$
CREATE TRIGGER trigger_crear_email_before_insert
BEFORE INSERT
ON empleados
FOR EACH ROW
BEGIN
IF NEW.email IS NULL THEN
        SET NEW.email = "correo_por_defecto@gmail.com";
END IF;
END ;
DELIMITER ;
```

- **Actividad 10**

Creación de la tabla log_cambios_email:

CREATE TABLE log_cambios_email (id INT AUTO_INCREMENT PRIMARY KEY, empleado_id INT, fecha_hora DATETIME, old_email VARCHAR(255), new_email VARCHAR(255));

```
DELIMITER $$
CREATE TRIGGER trigger_guardar_email_after_update
AFTER UPDATE ON empleados
FOR EACH ROW
BEGIN
        INSERT INTO log_cambios_email(empleado_id, fecha_hora, old_email, new_email)
    VALUES (OLD.empleado_id, NOW(), OLD.email, NEW.email);
END ;
DELIMITER ;
```

- **Actividad 11**

CREATE TABLE log_empleados_eliminados(id INT AUTO_INCREMENT PRIMARY KEY, empleado_id INT, fecha_hora DATETIME, nombre VARCHAR(200), apellido VARCHAR(200));

```
DELIMITER $$
CREATE TRIGGER trigger_guardar_empleados_eliminados
AFTER DELETE ON empleados
FOR EACH ROW
BEGIN
        INSERT into log_empleados_eliminados(empleado_id, fecha_hora, nombre, apellido)
    VALUES (OLD.empleado_id, NOW(), OLD.nombre, OLD.apellido, OLD.email);
END$$
DELIMITER ;
```

- **Actividad 12**

```
ALTER TABLE empleados ADD COLUMN email VARCHAR(100);

CREATE UNIQUE INDEX idx_email_unique ON empleados(email);

DELIMITER $$

CREATE PROCEDURE pa_insertar_empleado_email(IN p_empleado_id INT, IN
p_nombre VARCHAR(14), IN p_apellido VARCHAR(16), IN p_fecha_nacimiento DATE,
IN p_genero ENUM('M','F'), IN p_fecha_contratacion DATE, IN p_email VARCHAR(100))
BEGIN
   DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
   BEGIN
      SELECT 'Email duplicado, operación cancelada' AS mensaje;
   END;

   INSERT INTO empleados(empleado_id, nombre, apellido, fecha_nacimiento, genero,
fecha_contratacion, email)
   VALUES(p_empleado_id, p_nombre, p_apellido, p_fecha_nacimiento, p_genero,
p_fecha_contratacion, p_email);
END $$

DELIMITER ;

CALL pa_insertar_empleado_email(10012, 'Lucas', 'Díaz', '2000-01-01', 'M',
'2025-06-09', 'georgi_facello@gmail.com');
CALL pa_insertar_empleado_email(10013, 'Juan', 'Pérez', '1995-02-15', 'M',
'2025-06-09', 'pedro_garcía@gmail.com');
EXPLAIN SELECT * FROM empleados WHERE email = 'pedro_garcía@gmail.com';
```