UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

"LABORATORIO 05"

ASIGNATURA

Administración de Base de Datos

DOCENTE

Chávez Soto, Jorge Luis

ESTUDIANTE

Carhuaricra Anco, Heidy Nicole - 23200150

Lima, Perú

2025

ÍNDICE

LABORATORIO 05

- **OBJETIVOS**

El presente laboratorio tiene por objetivos:

- Crear triggers PL/SQL.
- Crear paquetes PL/SQL.

- **DESCRIPCIÓN DE LAS TABLAS**

- ***Countries***

Name	Null?	Type
COUNTRY_ID	NOT NULL	CHAR (2)
COUNTRY_NAME		VARCHAR2 (40)
REGION_ID		NUMBER

- ***Departments***

Name	Null?	Type
DEPARTMENT_ID	NOT NULL	NUMBER (4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2 (30)
MANAGER_ID		NUMBER (5)
LOCATION_ID		NUMBER (4)

- ***Employees***

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER (6)
FIRST_NAME		VARCHAR2 (20)
LAST_NAME	NOT NULL	VARCHAR2 (25)
EMAIL	NOT NULL	VARCHAR2 (25)
PHONE_NUMBER		VARCHAR2 (20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2 (10)
SALARY		NUMBER (8, 2)
COMMISSION_PCT		NUMBER (2, 2)
MANAGER_ID		NUMBER (6)
DEPARTMENT_ID		NUMBER (4)

- ***Jobs***

Name	Null?	Type
JOB_ID	NOT NULL	VARCHAR2 (10)
JOB_TITLE	NOT NULL	VARCHAR2 (35)
MIN_SALARY		NUMBER (6)
MAX_SALARY		NUMBER (6)

- ***Job_History***

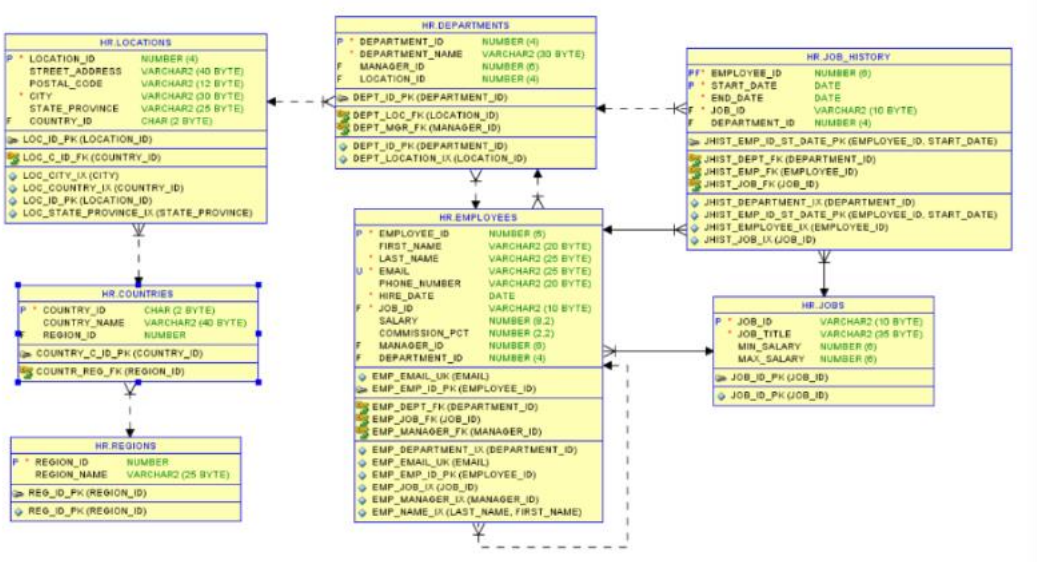
Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER (6)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2 (10)
DEPARTMENT_ID		NUMBER (4)

- ***Locations***

Name	Null?	Type
LOCATION_ID	NOT NULL	NUMBER (4)
STREET_ADDRESS		VARCHAR2 (40)
POSTAL_CODE		VARCHAR2 (12)
CITY	NOT NULL	VARCHAR2 (30)
STATE PROVINCE		VARCHAR2 (25)
COUNTRY_ID		CHAR (2)

- ***Regions***

Name	Null?	Type
RGION_ID	NOT NULL	NUMBER ()
REGION_NAME		VARCHAR2 (25)



- **EJERCICIOS PLANTEADOS**

1. Escriba el paquete respectivo para el objeto almacenado employees con los procedimientos y funciones CRUD necesarias y agregue lo siguiente:
   - Escriba un procedimiento que muestre el código de empleado, apellido y nombre, código de puesto actual y nombre de puesto actual, de los 4 empleados que más han rotado de puesto desde que ingresaron a la empresa. Para cada uno de ellos presente como columna adicional el número de veces que han cambiado de puesto.
   - Escriba una función que muestre un resumen estadístico del número promedio de contrataciones por cada mes con respecto a todos los años que hay información en la base de datos. Debe presentar sólo dos columnas: Nombre del Mes y Número Promedio de Contrataciones en ese Mes. Al final debe retornar por el nombre de la función el total de meses considerados en el listado.
   - Escriba un procedimiento que muestre la información de gastos en salario y estadística de empleados a nivel regional. Presente el nombre de la región, la suma de salarios, cantidad de empleados, y fecha de ingreso del empleado más antiguo.
   - Escriba una función para que calcule el tiempo de servicio de cada uno de sus empleados, para determinar el tiempo de vacaciones que le corresponde a cada empleado. Sabiendo que por un año de trabajo le corresponde un mes de vacaciones de acuerdo con ley. En el nombre de la función retorne el monto total empleado para el tiempo de servicios.

   Escriba las sentencias para crear las tablas de Horario con las columnas (día de la semana, turno, hora de inicio, hora de término), Empleado_Horario con las columnas (día de la semana, turno, código de empleado) y Asistencia_Empleado con las columnas (código de empleado, día de la semana, fecha real, hora de inicio

real y hora de término real). Además de ingresar un conjunto de 10 registros por tabla como mínimo.

- Escriba una función que reciba como parámetro el código de un empleado, el número de mes y el número de año. Calcule la cantidad de horas que laboro en dicho mes. Finalmente, retorne la cantidad de horas por el empleado en el nombre de la función.
- Escriba una función que reciba como parámetro el código de un empleado, el número de mes y el número de año. Calcule la cantidad de horas que falto el empleado en base a la función anterior. Finalmente, retorne en el nombre de la función la cantidad de horas que falto el empleado.
- Escriba un procedimiento que reciba como parámetro el número del mes y el número de año. Calcule para cada empleado en dicho mes y año el monto de sueldo que le corresponde de acuerdo con las horas laboradas y las horas de falta utilizando las funciones anteriores. Finalmente, realice un reporte en el que se muestre el nombre del empleado, el apellido del empleado, el salario que le corresponde en el mes y año.

Escriba un script que me permita crear las tablas de Capacitacion con las columnas (código de capacitación, nombre de la capacitación, horas de capacitacion y una descripción de la capacitación) y la tabla EmpleadoCapacitacion con las columnas (código de empleado y código de capacitación). Además de ingresar un conjunto de 10 registros por tabla.

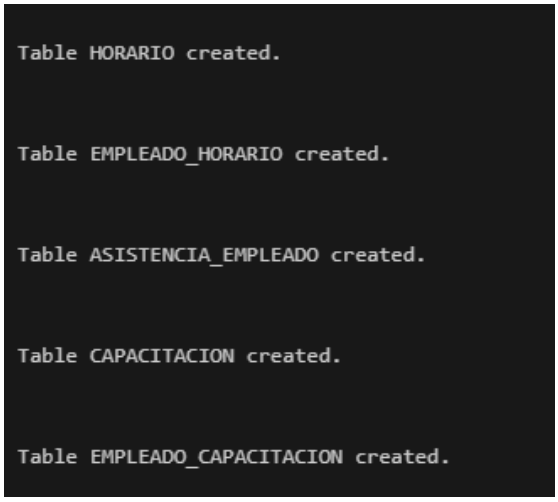
- Escriba una función que calcule la cantidad de horas totales que tiene cada empleado en las capacitaciones desarrolladas por la empresa.
- Elabore un procedimiento que liste todas las capacitaciones desarrolladas por la empresa y muestre los nombres de los empleados junto con la cantidad total de horas que participo cada empleado en las capacitaciones. El orden del listado debe ser por el total de horas de capacitación.

Solución

Creamos las tablas adicionales que van a ser necesarias para resolver este problema:

```
--- Ejercicio 01: Table Horario
CREATE TABLE horario (
    dia_semana      VARCHAR2(10),
    turno           VARCHAR2(10),
    hora_inicio     DATE,
    hora_termino    DATE,
    CONSTRAINT horario_pk PRIMARY KEY (dia_semana, turno)
);

--- Ejercicio 01: Table Empleado_Horario
CREATE TABLE empleado_horario (
    employee_id     NUMBER,
    dia_semana      VARCHAR2(10),
```

```
    turno           VARCHAR2(10),
    CONSTRAINT empleado_horario_pk PRIMARY KEY (employee_id,
dia_semana, turno),
    CONSTRAINT empleado_horario_fk1 FOREIGN KEY (employee_id)
REFERENCES employees(employee_id),
    CONSTRAINT empleado_horario_fk2 FOREIGN KEY (dia_semana,
turno) REFERENCES horario(dia_semana, turno)
);

--- Ejercicio 01: Table Asistencia_Empleado
CREATE TABLE asistencia_empleado (
    employee_id         NUMBER,
    dia_semana          VARCHAR2(10),
    fecha_asistencia    DATE,
    hora_entrada        DATE,
    hora_salida         DATE,
    CONSTRAINT asistencia_empleado_pk PRIMARY KEY (employee_id,
fecha_asistencia),
    CONSTRAINT asistencia_empleado_fk1 FOREIGN KEY
(employee_id) REFERENCES employees(employee_id)
);

--- Ejercicio 01: Capacitación
CREATE TABLE capacitacion (
    capacitacion_id     NUMBER PRIMARY KEY,
    nombre              VARCHAR2(100),
    descripcion         VARCHAR2(255),
    horas               NUMBER
);

--- Ejercicio 01: Empleado_Capacitacion
CREATE TABLE empleado_capacitacion (
    employee_id         NUMBER,
    capacitacion_id     NUMBER,
    fecha_capacitacion  DATE,
    CONSTRAINT empleado_capacitacion_fk1 FOREIGN KEY
(employee_id) REFERENCES employees(employee_id),
    CONSTRAINT empleado_capacitacion_fk2 FOREIGN KEY
(capacitacion_id) REFERENCES capacitacion(capacitacion_id)
);
```



Realizamos la inserción de datos para cada tabla:

```
--- Ejercicio 01: Inserción de datos - Table Horario
INSERT INTO HORARIO VALUES ('MON','MORNING', TO_DATE('2000-01-
01 08:00','YYYY-MM-DD HH24:MI'), TO_DATE('2000-01-01
16:00','YYYY-MM-DD HH24:MI'));
INSERT INTO HORARIO VALUES ('TUE','MORNING', TO_DATE('2000-01-
01 08:00','YYYY-MM-DD HH24:MI'), TO_DATE('2000-01-01
16:00','YYYY-MM-DD HH24:MI'));
INSERT INTO HORARIO VALUES ('WED','MORNING', TO_DATE('2000-01-
01 08:00','YYYY-MM-DD HH24:MI'), TO_DATE('2000-01-01
16:00','YYYY-MM-DD HH24:MI'));
INSERT INTO HORARIO VALUES ('THU','MORNING', TO_DATE('2000-01-
01 08:00','YYYY-MM-DD HH24:MI'), TO_DATE('2000-01-01
16:00','YYYY-MM-DD HH24:MI'));
INSERT INTO HORARIO VALUES ('FRI','MORNING', TO_DATE('2000-01-
01 08:00','YYYY-MM-DD HH24:MI'), TO_DATE('2000-01-01
16:00','YYYY-MM-DD HH24:MI'));
INSERT INTO HORARIO VALUES ('MON','EVENING', TO_DATE('2000-01-
01 16:00','YYYY-MM-DD HH24:MI'), TO_DATE('2000-01-01
20:00','YYYY-MM-DD HH24:MI'));
INSERT INTO HORARIO VALUES ('TUE','EVENING', TO_DATE('2000-01-
01 16:00','YYYY-MM-DD HH24:MI'), TO_DATE('2000-01-01
20:00','YYYY-MM-DD HH24:MI'));
INSERT INTO HORARIO VALUES ('WED','EVENING', TO_DATE('2000-01-
01 16:00','YYYY-MM-DD HH24:MI'), TO_DATE('2000-01-01
20:00','YYYY-MM-DD HH24:MI'));
INSERT INTO HORARIO VALUES ('THU','EVENING', TO_DATE('2000-01-
01 16:00','YYYY-MM-DD HH24:MI'), TO_DATE('2000-01-01
20:00','YYYY-MM-DD HH24:MI'));
INSERT INTO HORARIO VALUES ('FRI','EVENING', TO_DATE('2000-01-
01 16:00','YYYY-MM-DD HH24:MI'), TO_DATE('2000-01-01
20:00','YYYY-MM-DD HH24:MI'));

--- Ejercicio 01: Inserción de datos - Table Empleado_Horario
INSERT INTO EMPLEADO_HORARIO VALUES (101, 'MON','MORNING');
INSERT INTO EMPLEADO_HORARIO VALUES (101, 'TUE','MORNING');
```

```
INSERT INTO EMPLEADO_HORARIO VALUES (102, 'WED','MORNING');
INSERT INTO EMPLEADO_HORARIO VALUES (103, 'THU','MORNING');
INSERT INTO EMPLEADO_HORARIO VALUES (104, 'FRI','MORNING');
INSERT INTO EMPLEADO_HORARIO VALUES (105, 'MON','EVENING');
INSERT INTO EMPLEADO_HORARIO VALUES (106, 'TUE','EVENING');
INSERT INTO EMPLEADO_HORARIO VALUES (107, 'WED','EVENING');
INSERT INTO EMPLEADO_HORARIO VALUES (108, 'THU','EVENING');
INSERT INTO EMPLEADO_HORARIO VALUES (109, 'FRI','EVENING');

--- Ejercicio 01: Inserción de datos - Table
Asistencia_Empleado
INSERT INTO ASISTENCIA_EMPLEADO VALUES (101,'MON',
TO_DATE('2025-09-01','YYYY-MM-DD'), TO_DATE('2025-09-01
08:05','YYYY-MM-DD HH24:MI'), TO_DATE('2025-09-01 16:00','YYYY-
MM-DD HH24:MI'));
INSERT INTO ASISTENCIA_EMPLEADO VALUES (101,'TUE',
TO_DATE('2025-09-02','YYYY-MM-DD'), TO_DATE('2025-09-02
08:00','YYYY-MM-DD HH24:MI'), TO_DATE('2025-09-02 16:00','YYYY-
MM-DD HH24:MI'));
INSERT INTO ASISTENCIA_EMPLEADO VALUES (102,'WED',
TO_DATE('2025-09-03','YYYY-MM-DD'), TO_DATE('2025-09-03
08:00','YYYY-MM-DD HH24:MI'), TO_DATE('2025-09-03 16:00','YYYY-
MM-DD HH24:MI'));
INSERT INTO ASISTENCIA_EMPLEADO VALUES (103,'THU',
TO_DATE('2025-09-04','YYYY-MM-DD'), TO_DATE('2025-09-04
08:00','YYYY-MM-DD HH24:MI'), TO_DATE('2025-09-04 16:00','YYYY-
MM-DD HH24:MI'));
INSERT INTO ASISTENCIA_EMPLEADO VALUES (104,'FRI',
TO_DATE('2025-09-05','YYYY-MM-DD'), TO_DATE('2025-09-05
08:30','YYYY-MM-DD HH24:MI'), TO_DATE('2025-09-05 16:10','YYYY-
MM-DD HH24:MI'));
INSERT INTO ASISTENCIA_EMPLEADO VALUES (105,'MON',
TO_DATE('2025-09-01','YYYY-MM-DD'), TO_DATE('2025-09-01
16:00','YYYY-MM-DD HH24:MI'), TO_DATE('2025-09-01 20:00','YYYY-
MM-DD HH24:MI'));
INSERT INTO ASISTENCIA_EMPLEADO VALUES (106,'TUE',
TO_DATE('2025-09-02','YYYY-MM-DD'), TO_DATE('2025-09-02
16:00','YYYY-MM-DD HH24:MI'), TO_DATE('2025-09-02 20:00','YYYY-
MM-DD HH24:MI'));
INSERT INTO ASISTENCIA_EMPLEADO VALUES (107,'WED',
TO_DATE('2025-09-03','YYYY-MM-DD'), TO_DATE('2025-09-03
16:00','YYYY-MM-DD HH24:MI'), TO_DATE('2025-09-03 20:00','YYYY-
MM-DD HH24:MI'));
INSERT INTO ASISTENCIA_EMPLEADO VALUES (108,'THU',
TO_DATE('2025-09-04','YYYY-MM-DD'), TO_DATE('2025-09-04
16:00','YYYY-MM-DD HH24:MI'), TO_DATE('2025-09-04 20:00','YYYY-
MM-DD HH24:MI'));
INSERT INTO ASISTENCIA_EMPLEADO VALUES (109,'FRI',
TO_DATE('2025-09-05','YYYY-MM-DD'), TO_DATE('2025-09-05
16:00','YYYY-MM-DD HH24:MI'), TO_DATE('2025-09-05 20:00','YYYY-
MM-DD HH24:MI'));
```

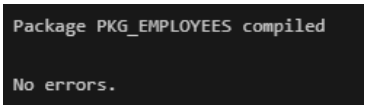
```
--- Ejercicio 01: Inserción de datos - Table Capacitación
INSERT INTO CAPACITACION VALUES (1, 'Excel Avanzado',
'Funciones y tablas dinámicas', 8);
INSERT INTO CAPACITACION VALUES (2, 'Seguridad Informática',
'Buenas prácticas', 6);
INSERT INTO CAPACITACION VALUES (3, 'Atención al Cliente',
'Comunicación efectiva', 4);
INSERT INTO CAPACITACION VALUES (4, 'Liderazgo', 'Gestión de
equipos', 10);
INSERT INTO CAPACITACION VALUES (5, 'SCRUM', 'Metodologías
ágiles', 6);
INSERT INTO CAPACITACION VALUES (6, 'SQL Básico', 'Consultas y
joins', 8);
INSERT INTO CAPACITACION VALUES (7, 'SQL Avanzado',
'Optimizaciones', 12);
INSERT INTO CAPACITACION VALUES (8, 'Power BI',
'Visualización', 10);
INSERT INTO CAPACITACION VALUES (9, 'Inglés Técnico',
'Terminología técnica', 20);
INSERT INTO CAPACITACION VALUES (10,'Gestión de Proyectos',
'Herramientas y planificación', 16);

--- Ejercicio 01: Inserción de datos - Table
Empleado_Capacitacion
INSERT INTO EMPLEADO_CAPACITACION VALUES (101, 1,
TO_DATE('2025-09-01','YYYY-MM-DD'));
INSERT INTO EMPLEADO_CAPACITACION VALUES (101, 6,
TO_DATE('2025-09-02','YYYY-MM-DD'));
INSERT INTO EMPLEADO_CAPACITACION VALUES (102, 2,
TO_DATE('2025-09-03','YYYY-MM-DD'));
INSERT INTO EMPLEADO_CAPACITACION VALUES (103, 3,
TO_DATE('2025-09-04','YYYY-MM-DD'));
INSERT INTO EMPLEADO_CAPACITACION VALUES (104, 4,
TO_DATE('2025-09-05','YYYY-MM-DD'));
INSERT INTO EMPLEADO_CAPACITACION VALUES (105, 5,
TO_DATE('2025-09-06','YYYY-MM-DD'));
INSERT INTO EMPLEADO_CAPACITACION VALUES (106, 7,
TO_DATE('2025-09-07','YYYY-MM-DD'));
INSERT INTO EMPLEADO_CAPACITACION VALUES (107, 8,
TO_DATE('2025-09-08','YYYY-MM-DD'));
INSERT INTO EMPLEADO_CAPACITACION VALUES (108, 9,
TO_DATE('2025-09-09','YYYY-MM-DD'));
INSERT INTO EMPLEADO_CAPACITACION VALUES (109, 10,
TO_DATE('2025-09-10','YYYY-MM-DD'));

COMMIT;
```

Creamos el paquete de empleados

```
--- Ejercicio 01: Creación de Paquete
SET serveroutput ON;
```

```
CREATE OR REPLACE PACKAGE pkg_employees IS
    --- CRUD
    PROCEDURE create_employee(
        p_employee_id       IN employees.employee_id%TYPE,
        p_first_name        IN employees.first_name%TYPE,
        p_last_name         IN employees.last_name%TYPE,
        p_hire_date         IN employees.hire_date%TYPE,
        p_job_id            IN employees.job_id%TYPE,
        p_salary            IN employees.salary%TYPE,
        p_department_id     IN employees.department_id%TYPE
    );

    PROCEDURE get_employee(
        p_employee_id       IN employees.employee_id%TYPE,
        p_rc                OUT SYS_REFCURSOR
    );

    PROCEDURE update_employee(
        p_employee_id       IN employees.employee_id%TYPE,
        p_first_name        IN employees.first_name%TYPE,
        p_last_name         IN employees.last_name%TYPE,
        p_job_id            IN employees.job_id%TYPE,
        p_salary            IN employees.salary%TYPE,
        p_department_id     IN employees.department_id%TYPE
    );

    PROCEDURE delete_employee(
        p_employee_id       IN employees.employee_id%TYPE
    );

    --- Top 4 empleados con más rotación de puesto
    PROCEDURE get_top_rotacion_puesto;

    --- Promedio de contrataciones por mes
    FUNCTION get_promedio_contrataciones_mes RETURN NUMBER;

    --- Estadísticas regionales
    PROCEDURE get_estadisticas_regionales;

    --- Tiempo de servicio y monto total de vacaciones
    FUNCTION get_tiempo_servicio_vacaciones RETURN NUMBER;

    --- Horas trabajadas por empleado en un mes
    FUNCTION get_horas_trabajadas_mes(
        p_employee_id       IN employees.employee_id%TYPE,
        p_mes               IN NUMBER,
        p_anio              IN NUMBER
    ) RETURN NUMBER;

    --- Horas faltadas por empleado en un mes
    FUNCTION get_horas_faltadas_mes(
        p_employee_id       IN employees.employee_id%TYPE,
```

```
        p_mes               IN NUMBER,
        p_anio              IN NUMBER
    ) RETURN NUMBER;

    --- Reporte de nómina para mes/año
    PROCEDURE generate_reporte_nomina(
        p_mes               IN NUMBER,
        p_anio              IN NUMBER
    );

    --- Total de horas de capacitación por empleado
    FUNCTION get_total_horas_capacitacion(
        p_employee_id       IN employees.employee_id%TYPE
    ) RETURN NUMBER;

    --- Lista de capacitaciones por empleado
    PROCEDURE get_lista_capacitaciones_horas_empleado;
END pkg_employees;
/
```



Creamos el cuerpo del paquete de empleados

```
--- Ejercicio 01: Cuerpo del Paquete
CREATE OR REPLACE PACKAGE BODY pkg_employees IS
    --- CRUD
    PROCEDURE create_employee(
        p_employee_id       IN employees.employee_id%TYPE,
        p_first_name        IN employees.first_name%TYPE,
        p_last_name         IN employees.last_name%TYPE,
        p_hire_date         IN employees.hire_date%TYPE,
        p_job_id            IN employees.job_id%TYPE,
        p_salary            IN employees.salary%TYPE,
        p_department_id     IN employees.department_id%TYPE
    ) IS
    BEGIN
        INSERT INTO employees (employee_id, first_name,
last_name, hire_date, job_id, salary, department_id)
        VALUES (p_employee_id, p_first_name, p_last_name,
p_hire_date, p_job_id, p_salary, p_department_id);
        COMMIT;
        dbms_output.put_line('Empleado creado con ID: ' ||
p_employee_id);
    EXCEPTION
        WHEN OTHERS THEN
            dbms_output.put_line('Error al crear empleado: ' ||
SQLERRM);
```

```
            ROLLBACK;
    END create_employee;

    PROCEDURE get_employee(
        p_employee_id       IN employees.employee_id%TYPE,
        p_rc                OUT SYS_REFCURSOR
    ) IS
    BEGIN
        OPEN p_rc FOR
            SELECT * FROM employees WHERE employee_id =
p_employee_id;
    EXCEPTION
        WHEN OTHERS THEN
            dbms_output.put_line('Error al obtener empleado: '
|| SQLERRM);
    END get_employee;

    PROCEDURE update_employee(
        p_employee_id       IN employees.employee_id%TYPE,
        p_first_name        IN employees.first_name%TYPE,
        p_last_name         IN employees.last_name%TYPE,
        p_job_id            IN employees.job_id%TYPE,
        p_salary            IN employees.salary%TYPE,
        p_department_id     IN employees.department_id%TYPE
    ) IS
    BEGIN
        UPDATE employees
        SET first_name = p_first_name,
            last_name = p_last_name,
            job_id = p_job_id,
            salary = p_salary,
            department_id = p_department_id
        WHERE employee_id = p_employee_id;
        COMMIT;
        dbms_output.put_line('Empleado actualizado con ID: ' ||
p_employee_id);
    EXCEPTION
        WHEN OTHERS THEN
            dbms_output.put_line('Error al actualizar empleado:
' || SQLERRM);
            ROLLBACK;
    END update_employee;

    PROCEDURE delete_employee(
        p_employee_id       IN employees.employee_id%TYPE
    ) IS
    BEGIN
        DELETE FROM employees WHERE employee_id =
p_employee_id;
        COMMIT;
        dbms_output.put_line('Empleado eliminado con ID: ' ||
p_employee_id);
```

```
    EXCEPTION
        WHEN OTHERS THEN
            dbms_output.put_line('Error al eliminar empleado: '
|| SQLERRM);
            ROLLBACK;
    END delete_employee;

    --- Ejercicio 01.1: Top 4 empleados con más rotación de
puesto
    PROCEDURE get_top_rotacion_puesto IS
    BEGIN
        FOR registro_cursor IN (
            SELECT e.employee_id,
            e.first_name,
            e.last_name,
            e.job_id AS idPuestoActual,
            j.job_title,
            COUNT(jh.job_id) AS numCambios
        FROM employees e
        JOIN jobs j ON e.job_id = j.job_id
        LEFT JOIN job_history jh ON e.employee_id =
jh.employee_id
        GROUP BY e.employee_id, e.last_name, e.first_name,
e.job_id, j.job_title
        HAVING COUNT(jh.job_id) > 0
        ORDER BY COUNT(jh.job_id) DESC
        FETCH FIRST 4 ROWS ONLY
        ) LOOP
            dbms_output.put_line('Empleado: ' ||
registro_cursor.employee_id ||
                '| Nombres y Apellidos: ' ||
registro_cursor.first_name || ' ' ||
                registro_cursor.last_name || '| Puesto Actual:
' ||
                registro_cursor.idPuestoActual || ' - ' ||
registro_cursor.job_title ||
                '| N° Cambios: ' ||
registro_cursor.numCambios);
        END LOOP;
    EXCEPTION
        WHEN OTHERS THEN
            dbms_output.put_line('Error: ' || SQLERRM);
    END get_top_rotacion_puesto;

    --- Ejercicio 01.2: Promedio de contrataciones por mes
    FUNCTION get_promedio_contrataciones_mes RETURN NUMBER IS
        v_total_meses NUMBER := 0;
    BEGIN
        FOR registro_cursor IN (
            SELECT TO_CHAR(TO_DATE(mes_num, 'MM'), 'Month',
'NLS_DATE_LANGUAGE=SPANISH') AS nombre_mes,
```

```
                    ROUND(AVG(cont), 2) AS
promedio_contrataciones
            FROM (
                SELECT
                    TO_CHAR(hire_date, 'MM') AS mes_num,
                    COUNT(*) AS cont
                FROM employees
                GROUP BY TO_CHAR(hire_date, 'YYYY'),
TO_CHAR(hire_date, 'MM')
            )
            GROUP BY mes_num
            ORDER BY mes_num
        ) LOOP
            v_total_meses := v_total_meses + 1;
            dbms_output.put_line('Mes: ' ||
TRIM(registro_cursor.nombre_mes) ||
                '| Promedio de Contrataciones: ' ||
registro_cursor.promedio_contrataciones);
        END LOOP;
        RETURN v_total_meses;
    EXCEPTION
        WHEN OTHERS THEN
            dbms_output.put_line('Error: ' || SQLERRM);
            RETURN 0;
    END get_promedio_contrataciones_mes;

    --- Ejercicio 01.3: Estadísticas regionales
    PROCEDURE get_estadisticas_regionales IS
    BEGIN
        FOR registro_cursor IN (
            SELECT r.region_name,
                COUNT(e.employee_id) AS total_empleados,
                NVL(SUM(e.salary), 0) AS suma_salarios,
                TO_CHAR(MIN(e.hire_date), 'DD-MM-YYYY') AS
ingreso_mas_antiguo
            FROM regions r
            LEFT JOIN countries c ON r.region_id = c.region_id
            LEFT JOIN locations l ON c.country_id =
l.country_id
            LEFT JOIN departments d ON l.location_id =
d.location_id
            LEFT JOIN employees e ON d.department_id =
e.department_id
            GROUP BY r.region_name
            ORDER BY r.region_name
        ) LOOP
            dbms_output.put_line('Región: ' ||
registro_cursor.region_name);
            dbms_output.put_line(' Salarios Totales: ' ||
registro_cursor.suma_salarios ||
                '| Empleados: ' ||
registro_cursor.total_empleados ||
```

```
                '| Ingreso (Más Antiguo): ' ||
NVL(registro_cursor.ingreso_mas_antiguo,'--'));
        END LOOP;
    EXCEPTION
        WHEN OTHERS THEN
            dbms_output.put_line('Error: ' || SQLERRM);
    END get_estadisticas_regionales;

    --- Ejercicio 01.4: Tiempo de servicio y monto total de
vacaciones
    FUNCTION get_tiempo_servicio_vacaciones RETURN NUMBER IS
        v_total_costo NUMBER := 0;
        v_years NUMBER := 0;
        v_days NUMBER := 0;
    BEGIN
        FOR registro_cursor IN (
            SELECT employee_id,
                first_name,
                last_name,
                hire_date,
                salary
            FROM employees
        ) LOOP
            v_days := TRUNC(SYSDATE -
registro_cursor.hire_date);
            v_years := ROUND(v_days / 365, 2);
            v_total_costo := v_total_costo +
NVL(registro_cursor.salary, 0) * v_years;
            dbms_output.put_line('Empleado: ' ||
registro_cursor.employee_id ||
                '| Nombres y Apellidos: ' ||
registro_cursor.first_name ||
                ' ' || registro_cursor.last_name || '| Años de
Servicio: ' ||
                v_years || '| Salario mensual: ' ||
NVL(registro_cursor.salary, 0));
        END LOOP;

        dbms_output.put_line('Total monto para tiempo de
Vacaciones: ' || v_total_costo);
        RETURN v_total_costo;
    EXCEPTION
        WHEN OTHERS THEN
            dbms_output.put_line('Error: ' || SQLERRM);
            RETURN 0;
    END get_tiempo_servicio_vacaciones;

    --- (Convertir day code en horario en fecha)
    FUNCTION day_code(p_date DATE) RETURN VARCHAR2 IS
    BEGIN
        RETURN UPPER(TO_CHAR(p_date, 'DY',
'NLS_DATE_LANGUAGE=ENGLISH'));
```

```
    END day_code;

    --- Ejercicio 01.5: Horas trabajadas por empleado en un mes
    FUNCTION get_horas_trabajadas_mes(
        p_employee_id   IN employees.employee_id%TYPE,
        p_mes           IN NUMBER,
        p_anio          IN NUMBER
    ) RETURN NUMBER IS
        v_total_horas NUMBER := 0;
        CURSOR asistencia_cursor IS
            SELECT ae.hora_entrada, ae.hora_salida,
ae.fecha_asistencia
                FROM asistencia_empleado ae
                WHERE ae.employee_id = p_employee_id
                AND EXTRACT(MONTH FROM ae.fecha_asistencia) =
p_mes
                AND EXTRACT(YEAR FROM ae.fecha_asistencia) =
p_anio;
        v_hora_inicio DATE;
        v_hora_termino DATE;
    BEGIN
        FOR registro_cursor IN asistencia_cursor LOOP
            v_hora_inicio := registro_cursor.hora_entrada;
            v_hora_termino := registro_cursor.hora_salida;
            IF v_hora_inicio IS NOT NULL AND v_hora_termino IS
NOT NULL THEN
                -- Calcular horas trabajadas
                v_total_horas := v_total_horas +
(v_hora_termino - v_hora_inicio) * 24;
            END IF;
        END LOOP;
        RETURN ROUND(v_total_horas, 2);
    EXCEPTION
        WHEN OTHERS THEN
            dbms_output.put_line('Error al calcular horas
trabajadas: ' || SQLERRM);
            RETURN 0;
    END get_horas_trabajadas_mes;

    --- Ejercicio 01.6: Horas faltadas por empleado en un mes
    FUNCTION get_horas_faltadas_mes(
        p_employee_id   IN employees.employee_id%TYPE,
        p_mes           IN NUMBER,
        p_anio          IN NUMBER
    ) RETURN NUMBER IS
        v_total_horas_faltadas      NUMBER := 0;
        v_horas_trabajadas_mes      NUMBER := 0;
        v_horas_programadas_mes     NUMBER := 0;
    BEGIN
        --- Calcular las horas programadas del empleado según
su horario
        SELECT SUM((h.hora_termino - h.hora_inicio) * 24)
```

```
            INTO v_horas_programadas_mes
        FROM empleado_horario eh
        JOIN horario h ON eh.dia_semana = h.dia_semana AND
eh.turno = h.turno
        WHERE eh.employee_id = p_employee_id;

        --- Calcular las horas trabajadas del empleado en el
mes
        v_horas_trabajadas_mes :=
get_horas_trabajadas_mes(p_employee_id, p_mes, p_anio);

        --- Calcular las horas faltadas
        v_total_horas_faltadas := v_horas_programadas_mes -
v_horas_trabajadas_mes;
        IF v_total_horas_faltadas < 0 THEN
            v_total_horas_faltadas := 0;
        END IF;

        dbms_output.put_line('Empleado: ' || p_employee_id ||
            '| Horas Programadas: ' ||
ROUND(v_horas_programadas_mes, 2) ||
            '| Horas Trabajadas: ' ||
ROUND(v_horas_trabajadas_mes, 2) ||
            '| Horas Faltadas: ' ||
ROUND(v_total_horas_faltadas, 2));

        RETURN ROUND(v_total_horas_faltadas, 2);
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            dbms_output.put_line('No se encontraron datos para
el empleado: ' || p_employee_id);
            RETURN 0;
        WHEN OTHERS THEN
            dbms_output.put_line('Error al calcular horas
faltadas: ' || SQLERRM);
            RETURN 0;
    END get_horas_faltadas_mes;

    --- Ejercicio 01.7: Reporte de nómina para mes/año
    PROCEDURE generate_reporte_nomina(
        p_mes           IN NUMBER,
        p_anio          IN NUMBER
    ) IS
        v_horas_trabajadas_mes      NUMBER;
        v_horas_faltadas_mes        NUMBER;
        v_salario                   NUMBER;
    BEGIN
        dbms_output.put_line('Reporte de Nómina para ' || p_mes
|| '/' || p_anio);
        dbms_output.put_line('----------------------------------
---------');
```

```
        FOR registro_cursor IN (
            SELECT e.employee_id, e.first_name, e.last_name,
e.salary
            FROM employees e
        ) LOOP
            v_horas_trabajadas_mes :=
get_horas_trabajadas_mes(registro_cursor.employee_id, p_mes,
p_anio);
            v_horas_faltadas_mes :=
get_horas_faltadas_mes(registro_cursor.employee_id, p_mes,
p_anio);
            v_salario := NVL(registro_cursor.salary, 0) *
(v_horas_trabajadas_mes / 160);

            dbms_output.put_line('Empleado: ' ||
registro_cursor.employee_id ||
                '| Nombres y Apellidos: ' ||
registro_cursor.first_name || ' ' || registro_cursor.last_name
||
                '| Horas Trabajadas: ' ||
ROUND(v_horas_trabajadas_mes, 2) ||
                '| Horas Faltadas: ' ||
ROUND(v_horas_faltadas_mes, 2) ||
                '| Salario: ' || ROUND(v_salario, 2));
        END LOOP;
    EXCEPTION
        WHEN OTHERS THEN
            dbms_output.put_line('Error al generar reporte de
nómina: ' || SQLERRM);
    END generate_reporte_nomina;

    --- Ejercicio 01.8: Total de horas de capacitación por
empleado
    FUNCTION get_total_horas_capacitacion(
        p_employee_id   IN employees.employee_id%TYPE
    ) RETURN NUMBER IS
        v_total_horas NUMBER := 0;
    BEGIN
        SELECT NVL(SUM(c.horas), 0)
            INTO v_total_horas
        FROM empleado_capacitacion ec
        JOIN capacitacion c ON ec.capacitacion_id =
c.capacitacion_id
        WHERE ec.employee_id = p_employee_id;

        dbms_output.put_line('Empleado: ' || p_employee_id ||
            '| Total Horas de Capacitación: ' ||
v_total_horas);

        RETURN v_total_horas;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
```

```
            dbms_output.put_line('No se encontraron
capacitaciones para el empleado: ' || p_employee_id);
            RETURN 0;
        WHEN OTHERS THEN
            dbms_output.put_line('Error al calcular horas de
capacitación: ' || SQLERRM);
            RETURN 0;
    END get_total_horas_capacitacion;

    --- Ejercicio 01.9: Lista de capacitaciones por empleado
    PROCEDURE get_lista_capacitaciones_horas_empleado IS
    BEGIN
        dbms_output.put_line('Lista de Capacitaciones por
Empleado');
        dbms_output.put_line('----------------------------------
---------');

        FOR registro_cursor IN (
            SELECT e.employee_id,
                e.first_name || ' ' || e.last_name AS
nombres_apellidos,
                NVL(SUM(c.horas), 0) AS total_horas
            FROM employees e
            LEFT JOIN empleado_capacitacion ec ON e.employee_id
= ec.employee_id
            LEFT JOIN capacitacion c ON ec.capacitacion_id =
c.capacitacion_id
            GROUP BY e.employee_id, e.first_name, e.last_name
            ORDER BY total_horas DESC
        ) LOOP
            dbms_output.put_line('Empleado: ' ||
registro_cursor.nombres_apellidos ||
                '| Total Horas de Capacitación: ' ||
registro_cursor.total_horas);
        END LOOP;
    EXCEPTION
        WHEN OTHERS THEN
            dbms_output.put_line('Error al obtener lista de
capacitaciones: ' || SQLERRM);
    END get_lista_capacitaciones_horas_empleado;
END pkg_employees;
/
```

```
Package Body PKG_EMPLOYEES compiled

No errors.
```

2. Escriba un trigger que verifique que sea correcto la inserción de la asistencia del empleado en la tabla AsistenciaEmpleado. Donde, la fecha tiene correspondencia con el día de la semana, la hora de inicio tiene correspondencia con la hora de inicio real, la hora de término corresponda con la hora de término real del empleado que se está registrando su asistencia.

   Solución

```
--- Ejercicio 02

CREATE OR REPLACE TRIGGER trg_validar_asistencia_empleado
BEFORE INSERT ON asistencia_empleado
FOR EACH ROW
DECLARE
    v_dia_semana_horario        horario.dia_semana%TYPE;
    v_hora_inicio               horario.hora_inicio%TYPE;
    v_hora_termino              horario.hora_termino%TYPE;
    v_dia_fecha                 VARCHAR2(10);
BEGIN
    -- Obtener el día de la semana de la fecha de asistencia
    v_dia_fecha := UPPER(TO_CHAR(:NEW.fecha_asistencia, 'DAY',
'NLS_DATE_LANGUAGE=SPANISH'));
    v_dia_fecha := TRIM(v_dia_fecha); --- Eliminar espacios en
blanco

    --- Verificar que coincida el día de la semana con la fecha
    IF UPPER(:NEW.dia_semana) <> v_dia_fecha THEN
        RAISE_APPLICATION_ERROR(-20001,
            'Error: El día de la semana (' || :NEW.dia_semana
||
            ') no corresponde con la fecha indicada (' ||
:NEW.fecha_asistencia || ').');
    END IF;

    --- Buscar el horario del empleado para ese día
    SELECT h.hora_inicio, h.hora_termino
        INTO v_hora_inicio, v_hora_termino
    FROM empleado_horario eh
    JOIN horario h ON eh.dia_semana = h.dia_semana AND eh.turno
= h.turno
    WHERE eh.employee_id = :NEW.employee_id
        AND UPPER(eh.dia_semana) = UPPER(:NEW.dia_semana);

    --- Validar la hora de entrada
    IF TRUNC(:NEW.hora_entrada, 'MI') <> TRUNC(v_hora_inicio,
'MI') THEN
        RAISE_APPLICATION_ERROR(-20002,
            'Error: La hora de entrada (' ||
TO_CHAR(:NEW.hora_entrada, 'HH24:MI') ||
            ') no coincide con la hora de inicio del horario ('
|| TO_CHAR(v_hora_inicio, 'HH24:MI') || ').');
    END IF;
```

```
    --- Validar la hora de salida
    IF TRUNC(:NEW.hora_salida, 'MI') <> TRUNC(v_hora_termino,
'MI') THEN
        RAISE_APPLICATION_ERROR(-20003,
            'Error: La hora de salida (' ||
TO_CHAR(:NEW.hora_salida, 'HH24:MI') ||
            ') no coincide con la hora de término del horario
(' || TO_CHAR(v_hora_termino, 'HH24:MI') || ').');
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20004,
            'Error: No existe horario asignado para el empleado
' || :NEW.employee_id ||
            ' en el día ' || :NEW.dia_semana || '.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20099,
            'Error en trigger de asistencia: ' || SQLERRM);
END trg_validar_asistencia_empleado;
/
```

```
Trigger TRG_VALIDAR_ASISTENCIA_EMPLEADO compiled

No errors.
```

3. Escriba un trigger que verifique que el sueldo asignado o actualizado a un empleado este dentro del rango del mínimo y máximo de acuerdo con el puesto asignado a dicho empleado que se puede validar en la tabla Jobs.
   Solución

```
--- Ejercicio 03

CREATE OR REPLACE TRIGGER trg_validar_sueldo_empleado
BEFORE INSERT OR UPDATE OF salary, job_id ON employees
FOR EACH ROW
DECLARE
    v_min_salary        jobs.min_salary%TYPE;
    v_max_salary        jobs.max_salary%TYPE;
BEGIN
    --- Buscar los límites salariales del puesto asignado
    SELECT min_salary, max_salary
        INTO v_min_salary, v_max_salary
    FROM jobs
    WHERE job_id = :NEW.job_id;

    --- Validar que el sueldo esté dentro del rango permitido
    IF :NEW.salary < v_min_salary OR :NEW.salary > v_max_salary
THEN
```

```
        RAISE_APPLICATION_ERROR(-20010,
            'El sueldo ' || :NEW.salary ||
            ' está fuera del rango permitido para el puesto '
|| :NEW.job_id ||
            '. Rango permitido: ' || v_min_salary || ' - ' ||
v_max_salary);
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20011,
            'No existe información del puesto ' || :NEW.job_id
|| ' en la tabla Jobs.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20099,
            'Error en trigger de validación de sueldo:' ||
SQLERRM);
END trg_validar_sueldo_empleado;
/
```

```
Trigger TRG_VALIDAR_SUELDO_EMPLEADO compiled

No errors.
```

4. Escriba un trigger que restringa el acceso al registro del ingreso sea media hora antes o media hora después de su hora exacta de ingreso y marque inasistencia del empleado, sin que el empleado se dé cuenta.
   Solución

```
--- Ejercicio 04

CREATE OR REPLACE TRIGGER trg_control_asistencia
BEFORE INSERT ON asistencia_empleado
FOR EACH ROW
DECLARE
    v_hora_inicio       horario.hora_inicio%TYPE;
    v_hora_termino      horario.hora_termino%TYPE;
    v_turno             empleado_horario.turno%TYPE;
    v_min_permitido     DATE;
    v_max_permitido     DATE;
BEGIN
    --- Buscar el turno y horario asignado al empleado ese día
    SELECT h.hora_inicio, h.hora_termino
        INTO v_hora_inicio, v_hora_termino
    FROM empleado_horario eh
    JOIN horario h ON eh.dia_semana = h.dia_semana
        AND eh.turno = h.turno
    WHERE eh.employee_id = :NEW.employee_id
        AND eh.dia_semana = :NEW.dia_semana;
```

```
    --- Calcular rango de tolerancia (+/- 30 minutos)
    v_min_permitido := v_hora_inicio - (30/1440);
    v_max_permitido := v_hora_inicio + (30/1440);

    --- Verificar si la hora de entrada está fuera del rango
    IF :NEW.hora_entrada < v_min_permitido OR :NEW.hora_entrada
> v_max_permitido THEN
        --- Marcar inasistencia automáticamente
        :NEW.hora_entrada := 'NULL';
        :NEW.hora_salida := 'NULL';
        dbms_output.put_line('Asistencia fuera de rango,
marcada como inasistencia (no visible para el empleado).');
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        dbms_output.put_line('No se encontró horario asignado
para el empleado ' || :NEW.employee_id || '.');
    WHEN OTHERS THEN
        dbms_output.put_line('Error en trigger de asistencia: '
|| SQLERRM);
END trg_control_asistencia;
/
```