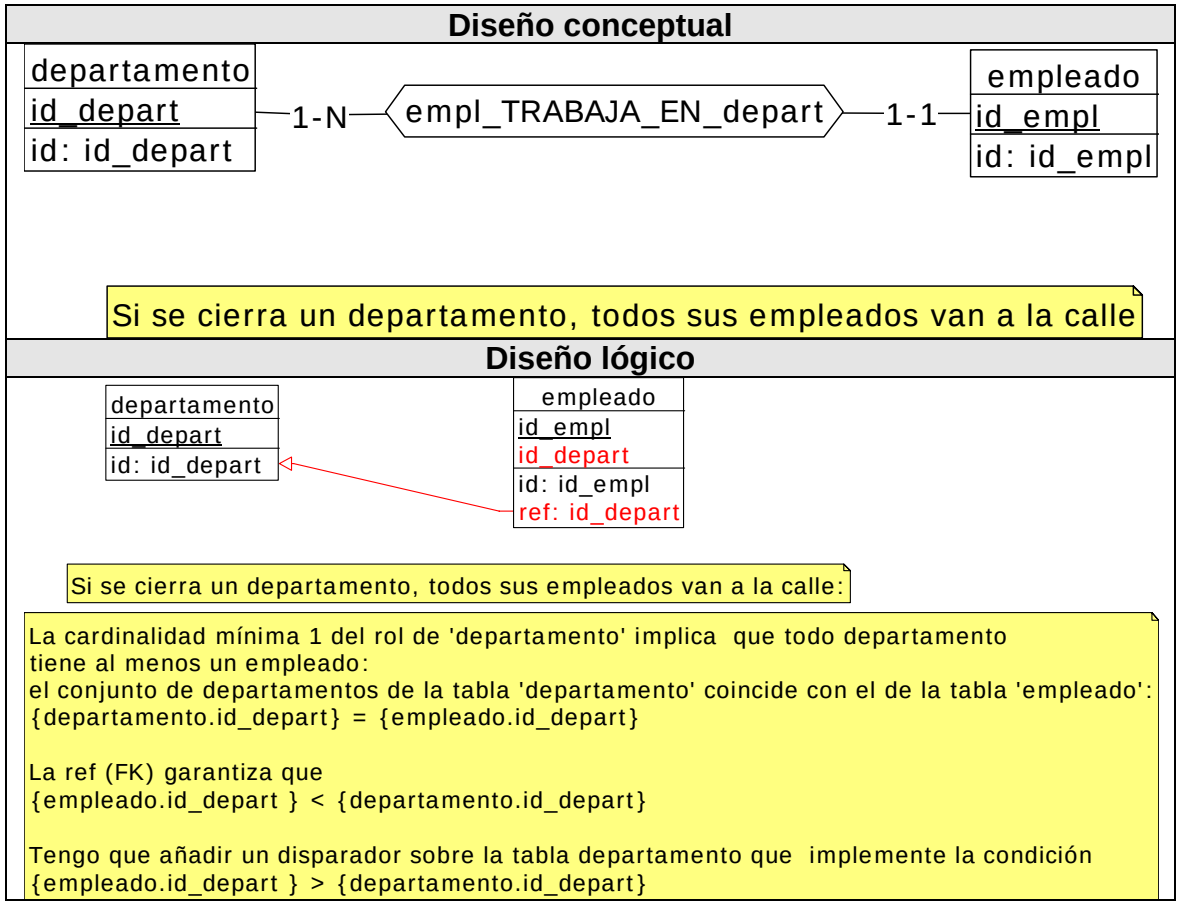
El objetivo de este documento es analizar cómo se implementa en ORACLE la restricción EQU.



La restricción EQU significa que los conjuntos de valores de ‘id_depart’ de las tablas ‘empleado’ y ‘departamento’ coinciden:

$$\{departamento.id\_depart\} = \{empleado.id\_depart\}$$

Esta igualdad implica dos condiciones, cuya implementación trataremos a continuación:

Todo empleado está asignado a un departamento

La referencia definida en el esquema lógico tiene el mismo significado que una clave foránea en el nivel físico: garantiza que

$$\{empleado.id\_depart\} \text{ es un subconjunto de } \{departamento.id\_depart\}$$

Todo departamento tiene al menos un empleado

Esto es, para cada fila de la tabla ‘departamento’ debe corresponderle al menos una fila de la tabla ‘empleado’:

$$\{departamento.id\_depart\} \text{ es un subconjunto de } \{empleado.id\_depart\}$$

Podemos observar que, en este caso, no se puede declarar una clave foránea de la tabla 'departamento' a 'empleado', ya que 'empleado.id_depart' no tiene la restricción UNIQUE: puede haber varios empleados en el mismo departamento.

Para imponer la restricción es necesario recurrir a disparadores, que reaccionarán ante los eventos que pueden suponer una violación de la misma:

1. La inserción de un departamento nuevo, sin empleados.
2. La modificación del identificador de un departamento, de modo que en la tabla 'empleado' no haya empleados asignados al departamento con el nuevo identificador.
3. El borrado de todos los empleados de un departamento.

Los eventos 1 y 2 se tratarán mediante un disparador sobre la tabla 'departamento':

```
CREATE OR REPLACE TRIGGER Disp_empl_depart
BEFORE INSERT OR UPDATE OF id_depart ON departamento
FOR EACH ROW
DECLARE
  v_contador integer:=0;
BEGIN
  select count(*) into v_contador
  from empleado
  where :new.id_depart= id_depart;

IF v_contador=0 THEN
    RAISE_APPLICATION_ERROR(-20007,'El departamento ' || :new.id_depart ||
                                                    ' no tiene empleados');
  END IF;

END;
```

La gestión del evento tercero requerirá un disparador sobre la tabla 'empleado': habría que ver, tras cada borrado, si todavía quedan en la tabla empleados para el departamento afectado. En este caso, 'empleado' es una tabla mutante (el disparador lanza una consulta sobre una tabla en la que se están realizando operaciones LMD), por lo que no podemos declarar disparadores a nivel de fila. Sin embargo, necesitamos conocer el identificador del departamento del que hemos borrado empleados (:old.id_depart), y sólo podemos acceder al valor de :old.id_depart desde un disparador a nivel de fila.

El problema puede resolverse del siguiente modo:

- Crear un paquete con una tabla PL/SQL, *tabla_depart_borrad*, en la que almacenaremos los identificadores de los departamentos para los que se han borrado empleados.
- Declarar un disparador a nivel de fila sobre la tabla 'empleado' que, cada vez que se dé de baja a un empleado, guarde el valor de su departamento en la tabla *tabla_depart_borrad*. Este disparador no realiza ninguna consulta sobre la tabla 'empleado' y, por tanto, no presenta problemas de tabla mutante.

- Declarar un disparador a nivel de sentencia sobre la tabla ‘empleado’ que, para cada uno de los departamentos de la tabla *tabla_depart_borrad*, consulte si le quedan empleados.

La implementación queda del siguiente modo:

```
CREATE OR REPLACE PACKAGE  depart_empl AS

TYPE t_depart_borrad IS TABLE OF  departamento.id_depart%TYPE
                INDEX BY BINARY_INTEGER;
        tabla_depart_borrad t_depart_borrad;

        numer_depart_borrad_s BINARY_INTEGER := 0;

END depart_empl;


CREATE OR REPLACE  TRIGGER R_NUMERO_EMPLEADOS
 BEFORE DELETE ON EMPLEADO
FOR EACH ROW
BEGIN
  depart_empl.numer_depart_borrad_s:= depart_empl.numer_depart_borrad_s+1;
  depart_empl.tabla_depart_borrad (depart_empl.numer_depart_borrad_s) := :new.id_depart;

END;



CREATE OR REPLACE TRIGGER S_NUMERO_EMPLEADOS
 AFTER DELETE ON EMPLEADO
DECLARE
 cuenta_empleados INTEGER;
 depart_buscad departamento.id_depart%TYPE;

BEGIN

  FOR id_depart IN 1..depart_empl.numer_depart_borrad_s LOOP

    depart_buscad := depart_empl.tabla_depart_borrad (id_depart);

    SELECT  COUNT (*)
    INTO    cuenta_empleados
    FROM    empleado
    WHERE   id_depart = depart_buscad;

    IF (cuenta_empleados = 0 AND depart_buscad IS NOT NULL) THEN
      RAISE_APPLICATION_ERROR (-20001,       'No se puede borrar el empleado: ‘ ||
                                             ‘el departamento ' || depart_buscad ||
                                             ' se quedaría vacío ' );
```

```
   END IF;
END LOOP;

END;
```

Queda por resolver un problema: ¿Cómo dar de alta un nuevo departamento? No podemos empezar insertando una fila en la tabla ‘empleado’ (se violaría la clave foránea declarada en el apartado anterior) ni en ‘departamento’ (el disparador *Disp_empl_depart* lanzaría una excepción). La solución es declarar la clave foránea como diferida: el SGBD la verifica no después de cada inserción, sino al finalizar una transacción:

```
ALTER TABLE empleado
ADD CONSTRAINT FKempleado_id_depart
FOREIGN KEY (id_depart) REFERENCES departamento
ON DELETE CASCADE
DEFERRABLE INITIALLY DEFERRED;
```

Hacemos notar que esta solución sólo es posible en un SGBD que permita la gestión de transacciones y que admita un lenguaje procedural. **Si no, no es posible implementar el esquema conceptual.**