1. Contenido adicional

implementación de Casos PRÁCTICOs (CAPÍTULO 6) en el SGBD Oracle

En el capítulo 6 se ha presentado el diseño y desarrollo de una base de datos de dos casos prácticos. Se ha seguido metodología vista a lo largo del libro, llevando a cabo todas las fases del modelado: esquema Entidad-Interrelación (E/R) y Relacional antes de pasar a la implementación en el Sistema Gestor de Base de datos (SGBD) que aquí se proporciona como un contenido adicional.

Aunque el lenguaje SQL admitido por ORACLE es muy similar al estándar visto, existen algunas diferencias. Las más importantes a destacar son los tipos de datos:

* Creación de dominios. ORACLE no admite dominios, los cuales deben simularse mediante *checks* o tablas auxiliares.
* Aserciones. ORACLE no cuenta con aserciones como están entendidas en el estándar, hay que implementarlas por medio de disparadores, llamados en TRIGGERs ORACLE.
* Opciones de borrado y modificación de claves ajenas. ORACLE 10g sólo admite opciones en el caso de borrado de claves ajenas a diferencia del estándar. Las únicas opciones admitidas son en el caso de:
  + Borrado: en cascada, con puesta a nulos y restringido, que es la opción por defecto en el caso de no poner nada.
  + Modificacion: con puesta a nulos y restringido, opción por defecto.

Según esta distinción de ORACLE en las opciones de borrado y modificación, habrá que implementar por medio de TRIGGER todas las opciones de modificación en cascada que tengamos.

A continuación se detallan los script de definición de tablas con restricciones de integridad: claves primarias (pk), claves alternativas (uk), claves ajenas (fk), aceptación de NULL y control de dominios por medio de *CHECK*.

NOTA: Para facilitar la notación en los script, y por características específicas de ORACLE respecto a la longitud máxima permitida para identificar objetos de la base de datos, se han modificado algunos nombres de tablas con la finalidad de reducir la longitud de sus nombres.

SCRIPT DE IMPLEMENTACIÓN DE LA BD DEL CASO PRACTICO 1: Gestión fondos de un museo

CREATE TABLE PLANTAS (

Numero VARCHAR2(4) NOT NULL,

D1\_N\_salas INTEGER NOT NULL,

CONSTRAINT pk\_plantas PRIMARY KEY (numero)

);

CREATE TABLE SALAS (

num\_sala VARCHAR2(7) NOT NULL,

num\_planta VARCHAR2(4) NOT NULL,

max\_cuadros INTEGER NOT NULL,

D2\_N\_cuadros INTEGER NOT NULL,

CONSTRAINT pk\_salas PRIMARY KEY (num\_sala,num\_planta),

CONSTRAINT fk\_plantas\_salas FOREIGN KEY (num\_planta)

REFERENCES PLANTAS

ON DELETE CASCADE

);

CREATE TABLE PINTORES (

cod\_pintor INTEGER NOT NULL,

nombre VARCHAR2(50),

nom\_artistico VARCHAR2(20) NOT NULL,

nacionalidad VARCHAR2(20) NOT NULL,

fecha\_nac DATE NOT NULL,

fecha\_fall DATE,

CONSTRAINT pk\_pintores PRIMARY KEY (cod\_pintor),

CONSTRAINT uk\_pintores UNIQUE (nom\_artistico),

CHECK (fecha\_nac<fecha\_fall)

);

CREATE TABLE ESTILOS (

nombre VARCHAR2(20) NOT NULL,

fecha\_ini DATE NOT NULL,

fecha\_fin DATE,

CONSTRAINT pk\_estilos PRIMARY KEY (nombre),

CHECK ( fecha\_ini<fecha\_fin )

);

CREATE TABLE COLECCIONES (

codigo INTEGER NOT NULL,

nombre VARCHAR2(20) NOT NULL,

direccion VARCHAR2(20) NOT NULL,

CONSTRAINT pk\_coleccion PRIMARY KEY (codigo)

);

CREATE TABLE CUADROS (

titulo VARCHAR2(50) NOT NULL,

tecnica VARCHAR2(20) NOT NULL,

fecha DATE NOT NULL,

pintor INTEGER NOT NULL,

estilo VARCHAR2(20) NOT NULL,

tipo\_cuadro VARCHAR2(20) NOT NULL,

CONSTRAINT pk\_cuadros PRIMARY KEY (pintor,titulo),

CONSTRAINT fk\_pintores\_cuadros FOREIGN KEY (pintor)

REFERENCES PINTORES

ON DELETE CASCADE,

CONSTRAINT fk\_estilos\_cuadros FOREIGN KEY (estilo)

REFERENCES ESTILOS,

CHECK (tecnica IN ('oleo', 'temple', 'acrílico', 'esmalte')),

CHECK (tipo\_cuadro IN ('propio', 'ajeno'))

);

CREATE TABLE AJENOS (

titulo VARCHAR2(50) NOT NULL,

pintor INTEGER NOT NULL,

CONSTRAINT pk\_ajenos PRIMARY KEY (pintor,titulo),

CONSTRAINT fk\_cuadros\_ajenos FOREIGN KEY (pintor,titulo)

REFERENCES CUADROS

ON DELETE CASCADE

);

CREATE TABLE PROPIOS (

titulo VARCHAR2(50) NOT NULL,

pintor INTEGER NOT NULL,

per\_restauracion VARCHAR2(5) NOT NULL,

coleccion INTEGER,

CONSTRAINT pk\_propios PRIMARY KEY (pintor,titulo),

CONSTRAINT fk\_cuadros\_propios FOREIGN KEY (pintor,titulo)

REFERENCES CUADROS

ON DELETE CASCADE,

CONSTRAINT fk\_colecciones\_propios FOREIGN KEY (coleccion)

REFERENCES COLECCIONES

);

CREATE TABLE EXPUESTOS (

num\_sala VARCHAR2(7) NOT NULL,

num\_planta VARCHAR2(4) NOT NULL,

titulo VARCHAR2(50) NOT NULL,

pintor INTEGER NOT NULL,

fecha\_ini DATE NOT NULL,

fecha\_fin DATE,

CONSTRAINT pk\_expuestos PRIMARY KEY (pintor,titulo,fecha\_ini),

CONSTRAINT uk\_expuestos UNIQUE (pintor, titulo, fecha\_fin),

CONSTRAINT fk\_salas\_expuestos FOREIGN KEY (num\_sala,num\_planta)

REFERENCES SALAS,

CONSTRAINT fk\_propios\_expuestos FOREIGN KEY (pintor, titulo)

REFERENCES PROPIOS,

CHECK ( fecha\_ini<fecha\_fin )

);

CREATE TABLE PERSONAL (

codigo INTEGER NOT NULL,

nombre VARCHAR2(50) NOT NULL,

tipo\_profesion VARCHAR2(20) NOT NULL,

CONSTRAINT pk\_personal PRIMARY KEY (codigo),

CHECK (tipo\_profesion IN ('conservador', 'restaurador'))

);

CREATE TABLE CONSERVADORES (

codigo INTEGER NOT NULL,

estilo VARCHAR2(20) NOT NULL,

CONSTRAINT pk\_conservadores PRIMARY KEY (codigo),

CONSTRAINT fk\_estilos\_conservadores FOREIGN KEY (estilo)

REFERENCES ESTILOS,

CONSTRAINT fk\_personal\_conservadores FOREIGN KEY (codigo)

REFERENCES PERSONAL

ON DELETE CASCADE

);

CREATE TABLE RESTAURADORES (

codigo INTEGER NOT NULL,

CONSTRAINT pk\_restauradores PRIMARY KEY (codigo),

CONSTRAINT fk\_personal\_restauradores FOREIGN KEY (codigo)

REFERENCES PERSONAL

ON DELETE CASCADE

);

CREATE TABLE REST (

codigo INTEGER NOT NULL,

pintor INTEGER NOT NULL,

titulo VARCHAR2(50) NOT NULL,

fecha\_ini DATE NOT NULL,

fecha\_fin DATE NOT NULL,

CONSTRAINT pk\_rest PRIMARY KEY (codigo,pintor,titulo, fecha\_ini),

CONSTRAINT fk\_restauradores\_rest FOREIGN KEY (codigo)

REFERENCES RESTAURADORES,

CONSTRAINT FK\_propios\_rest FOREIGN KEY (pintor,titulo)

REFERENCES PROPIOS,

CHECK ( fecha\_ini<fecha\_fin )

);

CREATE TABLE PUEDE\_REST (

restaurador INTEGER NOT NULL,

pintor INTEGER NOT NULL,

CONSTRAINT pk\_puede\_rest PRIMARY KEY (restaurador, pintor),

CONSTRAINT fk\_puede\_rest\_restauradores FOREIGN KEY (restaurador)

REFERENCES RESTAURADORES,

CONSTRAINT fk\_puede\_rest\_pintores FOREIGN KEY (pintor)

REFERENCES PINTORES

);

CREATE TABLE REST\_PROXIMAS (

restaurador INTEGER NOT NULL,

pintor INTEGER NOT NULL,

titulo VARCHAR2(50) NOT NULL,

D4\_fecha\_prox DATE NOT NULL,

CONSTRAINT pk\_rest\_proximas PRIMARY KEY (restaurador),

CONSTRAINT uk\_rest\_proximas UNIQUE (pintor,titulo),

CONSTRAINT fk\_propios\_rest\_proximas FOREIGN KEY (pintor,titulo)

REFERENCES PROPIOS

);

CREATE TABLE EXPO\_TEMP (

Titulo VARCHAR2(50) NOT NULL,

D3\_N\_cuadros INTEGER NOT NULL,

fecha\_ini DATE NOT NULL,

fecha\_fin DATE NOT NULL,

CONSTRAINT pk\_expo\_temp PRIMARY KEY (titulo),

CHECK ( fecha\_ini<fecha\_fin )

);

CREATE TABLE PREST\_INTERNOS (

pintor INTEGER NOT NULL,

titulo VARCHAR2(50) NOT NULL,

exposicion VARCHAR2(40) NOT NULL,

CONSTRAINT pk\_prest\_internos PRIMARY KEY (pintor, titulo, exposicion),

CONSTRAINT fk\_propios\_prest\_internos FOREIGN KEY (pintor,titulo)

REFERENCES PROPIOS,

CONSTRAINT fk\_expo\_temp\_prest\_internos FOREIGN KEY (exposicion)

REFERENCES EXPO\_TEMP

);

CREATE TABLE TELEFONOS (

telefono VARCHAR2(20) NOT NULL,

codigo INTEGER NOT NULL,

CONSTRAINT pk\_telefonos PRIMARY KEY (telefono),

CONSTRAINT fk\_colecciones\_telefonos FOREIGN KEY (codigo)

REFERENCES COLECCIONES

);

CREATE TABLE PREST\_EXT\_EXPO (

coleccion INTEGER NOT NULL,

pintor INTEGER NOT NULL,

titulo VARCHAR2(50) NOT NULL,

exposicion VARCHAR2(40) NOT NULL,

CONSTRAINT pk\_prest\_ext\_expo

PRIMARY KEY (pintor,titulo,exposicion),

CONSTRAINT fk\_colecciones\_prest\_ext\_expo FOREIGN KEY (coleccion)

REFERENCES COLECCIONES,

CONSTRAINT fk\_expo\_temp\_prest\_ext\_expo FOREIGN KEY (exposicion)

REFERENCES EXPO\_TEMP,

CONSTRAINT fk\_ajenos\_prest\_ext\_expo FOREIGN KEY (pintor,titulo)

REFERENCES AJENOS

);

Para implementar la base de datos con todas sus restricciones en el SGBD ORACLE, hay que añadir al script de creación expuesto TRIGGER como los que se indican en la tabla 7.1 del capítulo 6. A continuación, como ejemplo al lector, se va a mostrar el código de dos TRIGGER representativos.

A continuación se muestra el TRIGGER “Trigger\_Calculo\_cuadros\_sala”, que calcula el número de cuadros por sala, valor de *D2\_N\_cuadros* de la tabla SALAS. Como se indicó en la tabla 6.1 del capítulo 6 será un TRIGGER que se ejecutará después de insertar, borrar o modificar en la tabla EXPUESTOS.

CREATE OR REPLACE TRIGGER Trigger\_Calculo\_cuadros\_sala

AFTER INSERT OR DELETE OR UPDATE ON Expuestos FOR EACH ROW

DECLARE

fecha DATE;

actual BOOLEAN;

BEGIN

SELECT SYSDATE INTO fecha FROM dual;

IF INSERTING

THEN BEGIN

IF (:new.fecha\_ini<fecha) AND

(fecha < :new.fecha\_finor :new.fecha\_fin = null)

THEN

UPDATE salas SET D2\_N\_cuadros=D2\_N\_cuadros+1

WHERE (num\_sala=:new.num\_sala) AND

(num\_planta=:new.num\_planta);

END IF;

END;

ELSIF DELETING

THEN BEGIN

IF (:old.fecha\_ini<fecha) AND

(fecha < :old.fecha\_finor :old.fecha\_fin = null)

THEN

UPDATE salas SET D2\_N\_cuadros=D2\_N\_cuadros-1

WHERE (num\_sala=:old.num\_sala) AND

(num\_planta=:old.num\_planta);

END IF;

END;

ELSE BEGIN

IF (:old.fecha\_ini<fecha) AND

(fecha < :old.fecha\_finor :old.fecha\_fin = null)

THEN

UPDATE salas SET D2\_N\_cuadros=D2\_N\_cuadros-1

WHERE (num\_sala=:old.num\_sala) AND

(num\_planta=:old.num\_planta);

END IF;

IF (:new.fecha\_ini< fecha) AND

(fecha < :new.fecha\_finor :new.fecha\_fin = null)

THEN

UPDATE salas SET D2\_N\_cuadros=D2\_N\_cuadros+1

WHERE (num\_sala=:new.num\_sala) AND

(num\_planta=:new.num\_planta);

END IF;

END;

END IF;

END;

/

Por segundo disparador se muestra como ejemplo de TRIGGER, uno que comprueba que un restaurador sólo trabaja en restauraciones de cuadros de pintores que tenga asignado. Se ejecutará antes de insertar en la tabla REST. Consulta la tabla PUEDE\_REST para hacer la comprobación, en el caso de que el pintor del cuadro a restaurar no sea de los asignados al restautador, el TRIGGER no permitirá realizar la operación.

CREATE OR REPLACE TRIGGER Trigger\_puede\_restaurar

BEFORE INSERT ON rest FOR EACH ROW

DECLARE

pintor\_asignado INTEGER;

BEGIN

SELECT COUNT(\*) INTO Trigger\_pintor\_asignado

FROM puede\_rest

WHERE (pintor=:new.pintor) and (restaurador=:new.codigo);

IF pintor\_asignado = 0

THEN

RAISE\_application\_error (-20001, 'No se puede, ese cuadro es de un pintor

no asignado al restaurador');

END IF;

END;

/

SCRIPT DE IMPLEMENTACIÓN DE LA BD DEL CASO PRACTICO 2: Editorial de libros de texto escolar

CREATE TABLE VENDEDORES(

codigo VARCHAR2(10) NOT NULL,

correo VARCHAR2(50) NOT NULL,

NIF VARCHAR2(9) NOT NULL,

Nombre VARCHAR2(15) NOT NULL,

Apellidos VARCHAR2(30) NOT NULL,

tipo VARCHAR2(10) NOT NULL,

CONSTRAINT pk\_vendedores PRIMARY KEY (codigo),

CONSTRAINT uk\_vendedores1 UNIQUE (correo),

CONSTRAINT uk\_vendedores2 UNIQUE (NIF),

CHECK (tipo IN ('contratado','promotor'))

);

CREATE TABLE CONTRATADOS(

codigo VARCHAR2(10) NOT NULL,

modelo\_coche VARCHAR2(15) NOT NULL,

categoria VARCHAR2(7) NOT NULL,

fecha\_alta DATE NOT NULL,

fecha\_baja DATE,

CONSTRAINT pk\_contratados PRIMARY KEY (codigo),

CONSTRAINT fk\_contratados FOREIGN KEY (codigo)

REFERENCES VENDEDORES ON DELETE CASCADE,

CHECK (categoria IN ('1','2','3')),

CHECK ('fecha\_alta'<'fecha\_baja')

);

CREATE TABLE PROMOTORES(

codigo VARCHAR2(10) NOT NULL,

categoria VARCHAR2(1) NOT NULL,

contratado VARCHAR2(10) NOT NULL,

CONSTRAINT pk\_promotores PRIMARY KEY (codigo),

CONSTRAINT fk\_contratados1 FOREIGN KEY (codigo)

REFERENCES VENDEDORES ON DELETE CASCADE,

CONSTRAINT fk\_contratados2 FOREIGN KEY (contratado) REFERENCES CONTRATADOS,

CHECK (categoria IN ('A','B', 'C'))

);

CREATE TABLE DELEGACIONES(

nombre VARCHAR2(11) NOT NULL,

correo VARCHAR2(50) NOT NULL,

CONSTRAINT uk\_delegacion UNIQUE (correo),

CONSTRAINT pk\_delegación PRIMARY KEY (nombre)

);

CREATE TABLE CENTROS(

codigo VARCHAR2(10) NOT NULL,

titularidad VARCHAR2(10) NOT NULL,

nombre VARCHAR2(10) NOT NULL,

delegacion VARCHAR2(11) NOT NULL,

calle VARCHAR2(20) NOT NULL,

num NUMBER(3) NOT NULL,

poblacion VARCHAR2(20) NOT NULL,

provincial VARCHAR2(20) NOT NULL,

CP NUMBER(5) NOT NULL,

CONSTRAINT pk\_centros PRIMARY KEY (codigo),

CONSTRAINT fk\_centros FOREIGN KEY (delegacion) REFERENCES DELEGACIONES ON DELETE CASCADE

);

CREATE TABLE ASIGNADOS(

vendedor VARCHAR2(10) NOT NULL,

centro VARCHAR2(10) NOT NULL,

CONSTRAINT pk\_asignados1 PRIMARY KEY (vendedor, centro),

CONSTRAINT fk\_asignados1 FOREIGN KEY (vendedor)

REFERENCES VENDEDORES,

CONSTRAINT fk\_asignados2 FOREIGN KEY (centro)

REFERENCES CENTROS

);

CREATE TABLE VISITAS (

centro VARCHAR2(10) NOT NULL,

dia DATE NOT NULL,

hora\_ini DATE NOT NULL,

descripcion LONG NOT NULL,

hora\_fin DATE,

total\_gastos NUMBER (5,2) DEFAULT (0.00),

CONSTRAINT pk\_visitas PRIMARY KEY (centro,dia,hora\_ini),

CONSTRAINT fk\_visitas FOREIGN KEY (centro)

REFERENCES CENTROS ON DELETE CASCADE,

CHECK (hora\_ini<hora\_fin)

);

CREATE TABLE VISITAS\_VENDEDORES (

vendedor VARCHAR2(10) NOT NULL,

centro VARCHAR2(10) NOT NULL,

dia DATE NOT NULL,

hora\_ini DATE NOT NULL,

CONSTRAINT pk\_visitas\_vendedores PRIMARY KEY(vendedor,centro,dia,hora\_ini),

CONSTRAINT fk\_visitas\_vendedores1 FOREIGN KEY (vendedor) REFERENCES VENDEDORES,

CONSTRAINT fk\_visitas\_vendedores2 FOREIGN KEY (centro,dia,hora\_ini) REFERENCES VISITAS

);

CREATE TABLE GASTOS(

Codigo VARCHAR2(10) NOT NULL,

Precio NUMBER(10,2) NOT NULL,

Tipo VARCHAR2(15) NOT NULL,

Vendedor VARCHAR2(10) NOT NULL,

centro VARCHAR2(10) NOT NULL,

dia DATE NOT NULL,

hora\_ini DATE NOT NULL,

CONSTRAINT pk\_gastos PRIMARY KEY (codigo),

CONSTRAINT fk\_gastos1 FOREIGN KEY (vendedor) REFERENCES VENDEDORES,

CONSTRAINT fk\_gastos2 FOREIGN KEY (centro,dia,hora\_ini) REFERENCES VISITAS,

CHECK (tipo IN ('manutencion','transporte','alojamiento'))

);

CREATE TABLE TRANSPORTES (

codigo VARCHAR2(10) NOT NULL,

medio VARCHAR2(7) NOT NULL,

Kms NUMBER(4) DEFAULT (0),

CONSTRAINT pk\_transportes PRIMARY KEY (codigo),

CONSTRAINT fk\_transportes FOREIGN KEY (codigo) REFERENCES GASTOS ON DELETE CASCADE,

CHECK (medio IN ('coche','taxi','avion','tren','autobus','metro'))

);

CREATE TABLE MANUTENCIONES (

codigo VARCHAR2(8) NOT NULL,

hora VARCHAR2(8) NOT NULL,

CONSTRAINT pk\_manutenciones PRIMARY KEY (codigo),

CONSTRAINT fk\_manutenciones FOREIGN KEY (codigo)

REFERENCES GASTOS ON DELETE CASCADE,

CHECK (hora IN ('desayuno','comida','cena'))

);

Para implementar la base de datos con todas sus restricciones en el SGBD ORACLE, hay que añadir al script de creación expuesto, TRIGGER como los que se indican en la tabla 6.2 del capítulo 6. A continuación, como ejemplo al lector, se va a mostrar el código de dos TRIGGER representativos.

A continuación se muestra el TRIGGER “Calculo\_total\_gastos” que actualiza la columna “total\_gastos” de la tabla VISITAS.

CREATE OR REPLACE TRIGGER Calculo\_total\_gastos

BEFORE INSERT OR UPDATE OF precio ON GASTOS FOR EACH ROW

BEGIN

IF INSERTING

THENUPDATE VISITAS

SET total\_gastos = (VISITAS.total\_gastos + :new.precio)

WHERE

(VISITAS.centro=:new.centro) AND

(VISITAS.dia =:new.dia) AND

(VISITAS.hora\_ini =:new.hora\_ini);

ELSIF UPDATING

THEN

UPDATE VISITAS

SET total\_gastos = (VISITAS.total\_gastos +:new.precio -:old.precio)

WHERE

(VISITAS.centro =:new.centro) AND

(VISITAS.dia =:new.dia) AND

(VISITAS.hora\_ini =:new.hora\_ini);

END IF;

END;

/

Para controlar que un contratado supervisa como máximo a tres promotores, está el siguiente TRIGGER, “Trigger\_maximo\_supervisiones”, que se ejecutará en la tabla PROMOTORES cuando se inserte un nuevo vendedor promotor. Consulta la misma tabla PROMOTORES. No permitirá realizar la operación en el caso de que el vendedor contratado que se le quiera asignar como supervisor tenga ya tres promotores a supervisar.

CREATE OR REPLACE TRIGGER Trigger\_maximo\_supervisiones

BEFORE INSERT ON PROMOTORES FOR EACH ROW

DECLARE

n\_supervisiones INTEGER;

BEGIN

SELECT COUNT (\*) INTO n\_supervisiones

FROM PROMOTORES

WHERE (contratado =:new.contratado);

IF n\_supervisiones>=3

THEN

RAISE\_application\_error (-20001, 'No se puede asignar este

Contratado como supervisor, ya esta supervisando ya a

tresvendedorespromotores');

END IF;

END;

/