Exposiciones Enero 2017

Se desea diseñar la base de datos para la gestión de las exposiciones temporales que se realizan en todos los museos de España. Los requisitos de los que disponemos para diseñar dicha base de datos son los siguientes:

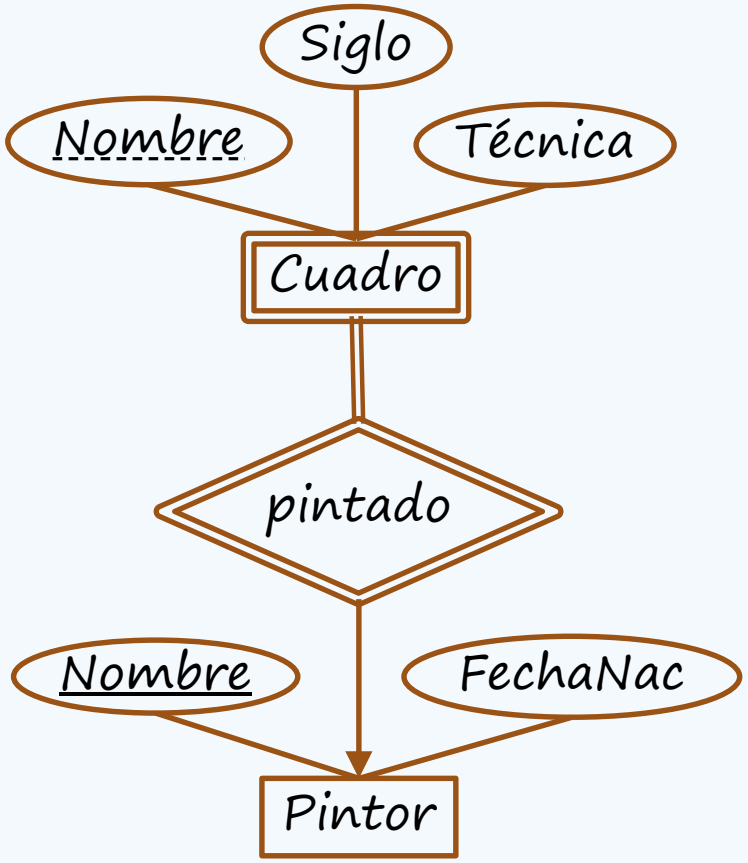
- ✓ De cada cuadro se registrará el nombre, el siglo en el que fue pintado y la técnica utilizada (óleo, acuarela, acrílico o carboncillo). Habrá que tener en cuenta que hay títulos de cuadros muy ampliamente utilizados (por ejemplo, muchos pintores tienen un autorretrato entre sus obras). Sin embargo, los títulos no se repiten entre las obras de un mismo pintor.
- ✓ Todos los cuadros son pintados por un pintor. De cada pintor guardaremos: el nombre (único) y su fecha de nacimiento.
- ✓ Un pintor puede tener (o no) a otro como maestro y un pintor puede ser maestro como máximo de 4 pintores. Solamente almacenaremos pintores que tengan cuadros en nuestra base de datos.
- ✓ De cada exposición se desea almacenar el nombre (único) y las fechas de inicio y fin, así como el museo en el que se realizan. Todas las exposiciones se celebran en algún museo de los registrados en la base de datos y no son itinerantes, es decir, solo se celebran en un museo.
- ✓ De cada museo se registrará el nombre (único para cada museo) y la ciudad en la que se encuentra. En la base de datos pueden aparecer museos que no tengan aún ninguna exposición temporal.
- ✓ Cada exposición temporal tiene asignados los cuadros que en ella se exhiben (al menos uno). Todos los cuadros que aparecen en la base de datos están asignados, al menos, a una de las exposiciones almacenadas. Cada vez que un cuadro se asigna a una exposición temporal es necesario suscribir una póliza de seguro cuyo número necesitamos almacenar.

Crea el esquema Entidad-Relación de la base de datos descrita. Debes de incluir atributos, claves y restricciones de cardinalidad y participación.

Alberto de la Encina Vara

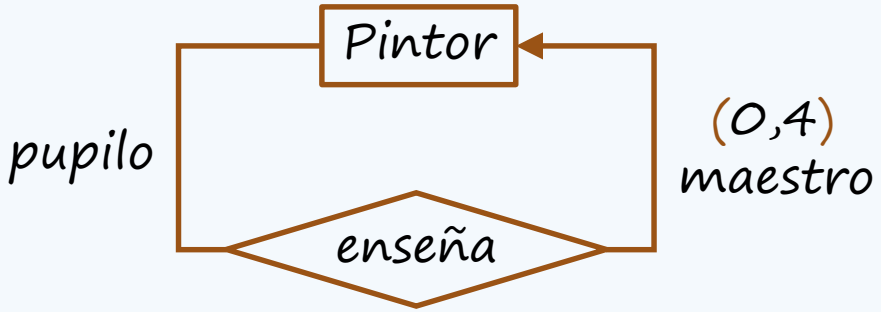
Exposiciones Enero 2017

- ✓ De cada cuadro se registrará el nombre, el siglo en el que fue pintado y la técnica utilizada (óleo, acuarela, acrílico o carboncillo). Habrá que tener en cuenta que hay títulos de cuadros muy ampliamente utilizados (por ejemplo, muchos pintores tienen un autorretrato entre sus obras). Sin embargo, los títulos no se repiten entre las obras de un mismo pintor.
- ✓ Todos los cuadros son pintados por un pintor. De cada pintor guardaremos: el nombre (único) y su fecha de nacimiento.



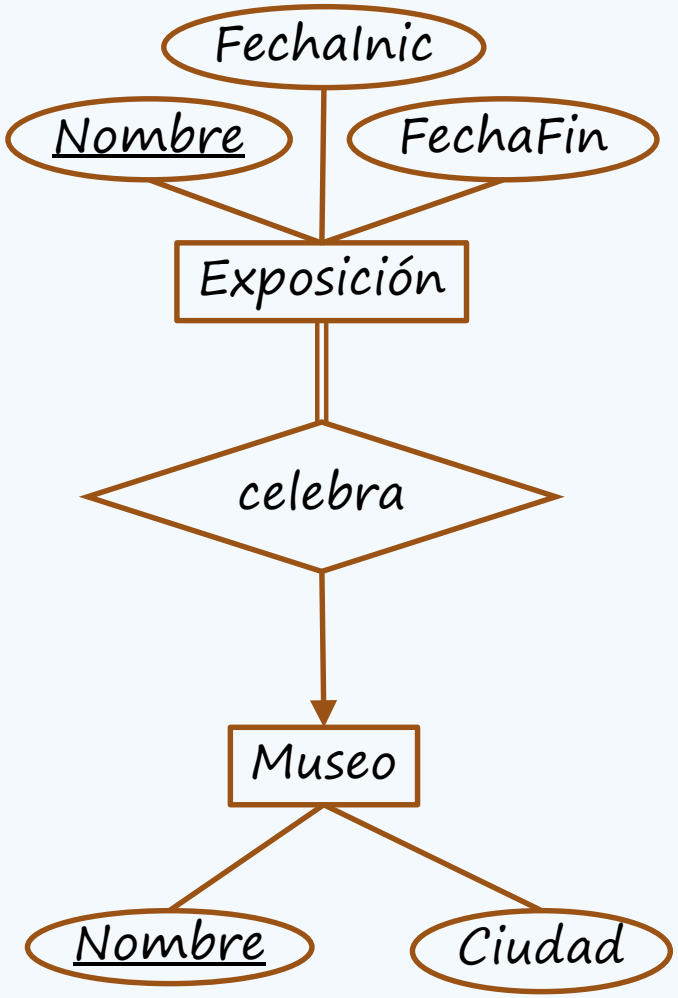
Exposiciones Enero 2017

- ✓Un pintor puede tener (o no) a otro como maestro y un pintor puede ser maestro como máximo de 4 pintores. Solamente almacenaremos pintores que tengan cuadros en nuestra base de datos.



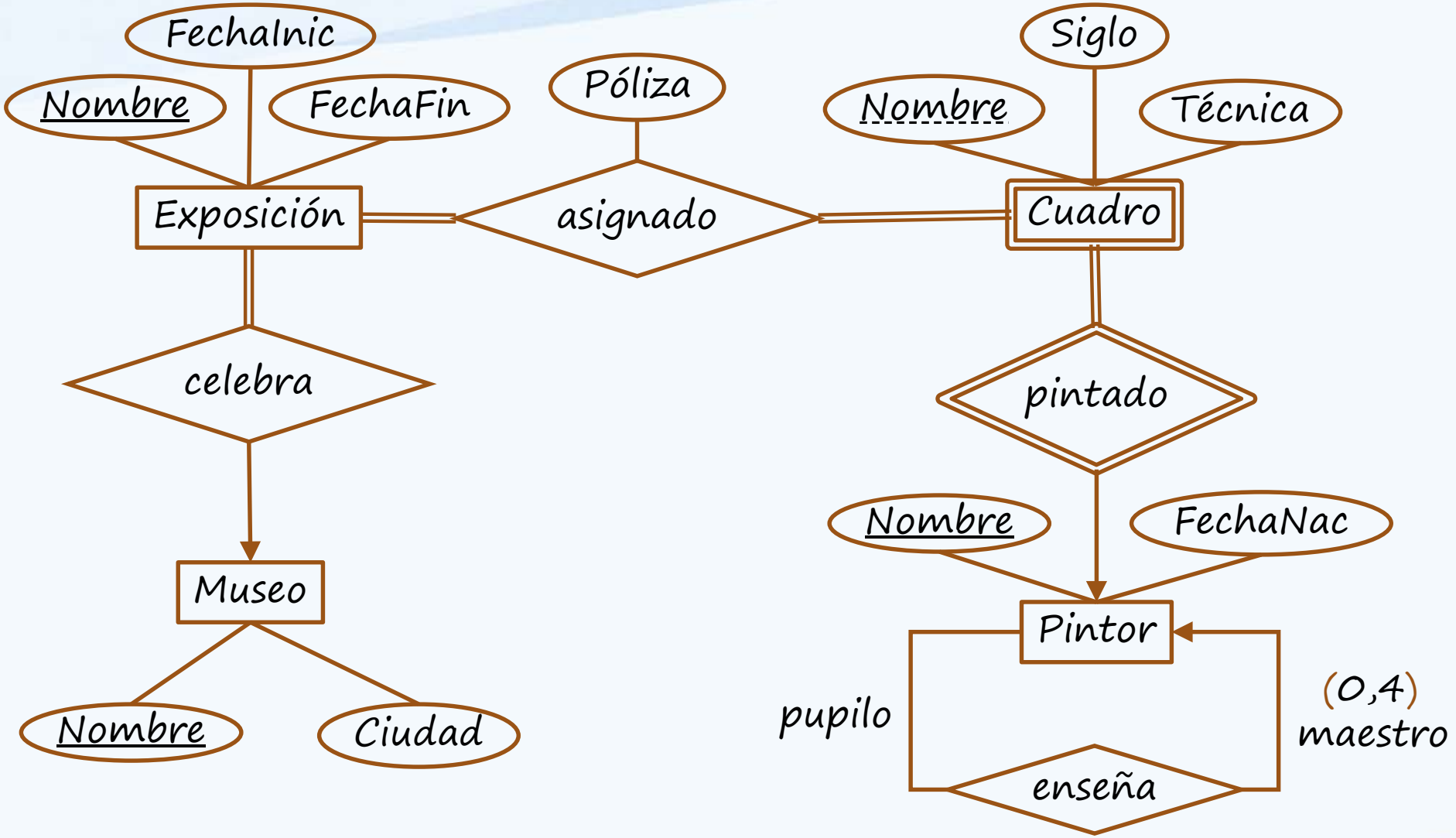
Exposiciones Enero 2017

- ✓ De cada exposición se desea almacenar el nombre (único) y las fechas de inicio y fin, así como el museo en el que se realizan. Todas las exposiciones se celebran en algún museo de los registrados en la base de datos y no son itinerantes, es decir, solo se celebran en un museo.
- ✓ De cada museo se registrará el nombre (único para cada museo) y la ciudad en la que se encuentra. En la base de datos pueden aparecer museos que no tengan aún ninguna exposición temporal.



Exposiciones Enero 2017

- ✓ Cada exposición temporal tiene asignados los cuadros que en ella se exhiben (al menos uno). Todos los cuadros que aparecen en la base de datos están asignados, al menos, a una de las exposiciones almacenadas. Cada vez que un cuadro se asigna a una exposición temporal es necesario suscribir una póliza de seguro cuyo número necesitamos almacenar.


FechaInic
Nombre
FechaFin
Póliza
Siglo
Nombre
Técnica
Exposición
asignado
Cuadro
celebra
pintado
Museo
Nombre
FechaNac
Pintor
Nombre
Ciudad
pupilo
(0,4)
maestro
enseña

Esquema R:

Etapa 1:

Puede ser nulo

Exposiciones(nombre, fechaIni*, fechaFin*)

Museos(nombre, ciudad)

Pintores(nombre, fechaNac)

Cuadros(nombre, nombrePintor, siglo, técnica)

Etapa 2:

Exposiciones(nombre, fechaIni*, fechaFin*, nombreMuseo)

Museos(nombre, ciudad)

Pintores(nombre, fechaNac, nombreMaestro*)

Cuadros(nombrePintor, nombre, siglo, técnica)

Asignados(nombreExp, nombrePintor, nombreCuad, póliza)

Esquema R:

Etapa 1:

Exposiciones(nombre, fechaIni*, fechaFin*)

Museos(nombre, ciudad)

Pintores(nombre, fechaNac)

Cuadros(nombre, nombrePintor, siglo, técnica)

C.E. nombrePintor de Pintores(nombre)

Etapa 2:

Exposiciones(nombre, fechaIni*, fechaFin*, nombreMuseo)

C.E. nombreMuseo de Museo(nombre)

Museos(nombre, ciudad)

Pintores(nombre, fechaNac, nombreMaestro*)

C.E. nombreMaestro de Pintores(nombre)

Cuadros(nombre, nombrePintor, siglo, técnica)

C.E. nombrePintor de Pintores(nombre)

Asignados(nombreExp, nombrePintor, nombreCuad, póliza)

C.E. nombreExp de Exposiciones(nombre)

C.E. (nombrePintor, nombreCuad) de Cuadros(nombrePintor, nombre)

SQL creación tablas

Las restricciones de cardinalidad total no han sido pasadas.
Así, como tampoco el cardinal del maestro (0,4).
Técnicas de pintura posibles pasadas a Oracle (óleo, acuarela, acrílico o carboncillo).

```
CREATE TABLE Enero17_Museos(
        nombre VARCHAR2(40) PRIMARY KEY,
        ciudad VARCHAR2(20) NOT NULL
);

CREATE TABLE Enero17_Exposiciones(
        nombre VARCHAR2(40) CONSTRAINT Enero17_PK_exposciones PRIMARY KEY,
        fechaIni DATE,
        fechaFin DATE,
        nombreMuseo VARCHAR2(40) REFERENCES Enero17_Museos(nombre) NOT NULL
);

CREATE TABLE Enero17_Pintores(
        nombre VARCHAR2(30),
        fechaNac DATE NOT NULL,
        nombreMaestro VARCHAR2(30),
        CONSTRAINT Enero17_PK_pintores PRIMARY KEY (nombre),
        CONSTRAINT Enero17_FK_pintores FOREIGN KEY (nombreMaestro)
                                REFERENCES Enero17_Pintores(nombre)
);
```

SQL creación tablas

```
CREATE TABLE Enero17_Cuadros(
        nombrePintor VARCHAR2(30),
        nombre VARCHAR2(25),
        siglo int NOT NULL,
        técnica VARCHAR2(20) NOT NULL
);
ALTER TABLE  Enero17_Cuadros ADD CONSTRAINT Enero17_PK_cuadros PRIMARY KEY (nombrePintor, nombre);
ALTER TABLE  Enero17_Cuadros ADD CONSTRAINT Enero17_FK_cuadros FOREIGN KEY (nombrePintor)
        REFERENCES Enero17_Pintores(nombre);
ALTER TABLE Enero17_Cuadros ADD CONSTRAINT Enero17_CK_Tecnica
        CHECK (técnica IN ('óleo', 'acuarela', 'acrílico', 'carboncillo', 'desconocida'));

CREATE TABLE Enero17_Asignados(
        nombreExp VARCHAR2(40),
        nombrePintor VARCHAR2(30),
        nombreCuad VARCHAR2(25),
        póliza int NOT NULL,
        CONSTRAINT Enero17_PK_asignados PRIMARY KEY (nombreExp, nombrePintor, nombreCuad),
        CONSTRAINT Enero17_FK_asignados_expo
                FOREIGN KEY (nombreExp) REFERENCES Enero17_Exposiciones(nombre),
        CONSTRAINT Enero17_FK_asignados_cuad
                FOREIGN KEY (nombrePintor, nombreCuad) REFERENCES Enero17_Cuadros
)
```

Consultas:

Exposiciones(nombre, fechaIni*, fechaFin*, nombreMuseo)

Museos(nombre, ciudad)

Pintores(nombre, fechaNac, nombreMaestro*)

Cuadros(nombrePintor, nombre, siglo, técnica)

Asignados(nombreExp, nombrePintor, nombreCuad, póliza)

1. Pintores cuyos cuadros no han sido expuestos en ningún museo.
2. Todos los cuadros que han sido expuestos en alguna exposición.
3. Museos en los que no se ha realizado ninguna exposición.
4. Las exposiciones sin cuadros. Debería ser vacía pero no se metió esa restricción.
5. Los pupilos sin maestros.
6. Los pintores que tiene maestro.
7. Los maestros sin pupilos.
8. Los maestros de si mismos.
9. Pintores que han pintado un cuadro titulado "autorretrato".
10. Nombre de los cuadros pintados por el pintor "Dali" o "Sofonisba".
11. Nombre de los cuadros titulados iguales pintados por el pintor "Dali" o "Sofonisba"

Consultas álgebra relacional:

1. Pintores cuyos cuadros no han sido expuestos en ningún museo.

   Π_{nombre} Pintores \ $\Pi_{nombrePintor}$ Asignados

2. Todos los cuadros que han sido expuestos en alguna exposición.

   $\Pi_{nombrePintor,\ nombreCuad}$ Asignados

3. Museos en los que no se ha realizado ninguna exposición.

   Π_{nombre} Museos \ $\Pi_{nombreMuseo}$ Exposiciones

4. Las exposiciones sin cuadros. Debería ser vacía pero no se metió esa restricción.

   Π_{nombre} Exposiciones \ $\Pi_{nombreExp}$ Asignados

5. Los pintores sin maestros.

   $\sigma_{(nombreMaestro == NULL)}$ Pintores

6. Los pintores que tiene maestro.

   $\sigma_{(nombreMaestro != NULL)}$ Pintores

7. Los maestros sin pupilos.

   Esta consulta no se puede hacer. Si eres maestro estás enseñando a algún pintor, así está almacenado en al BBDD.
   Salvo que se almacenara dicha información como: Los maestros que sólo se enseñan a si mismos.

8. Los maestros de si mismos.

   Π_{nombre} ($\sigma_{(nombreMaestro = nombre)}$ Pintores)

9. Pintores que han pintado un cuadro titulado "autorretrato".

   $\Pi_{nombrePintor}$ ($\sigma_{(nombre = \text{"autorretrato"})}$ Cuadros)

10. Nombre de los cuadros pintados por el pintor "Dalí" o "Sofonisba".

    Π_{nombre} ($\sigma_{(nombrePintor = \text{"Dalí"} \vee nombrePintor = \text{"Sofonisba"})}$ Cuadros)

11. Nombre de los cuadros titulados igual pintados por el pintor "Dalí" o "Sofonisba".

    Π_{nombre} ($\sigma_{(nombrePintor = \text{"Dalí"})}$ Cuadros) $\cap$ Π_{nombre} ($\sigma_{(nombrePintor = \text{"Sofonisba"})}$ Cuadros)

Consultas:

Exposiciones(nombre, fechaIni*, fechaFin*, nombreMuseo)

Museos(nombre, ciudad)

Pintores(nombre, fechaNac, nombreMaestro*)

Cuadros(nombrePintor, nombre, siglo, técnica)

Asignados(nombreExp, nombrePintor, nombreCuad, póliza)

1. La fecha de nacimiento de los pintores que han pintado un cuadro titulado "autorretrato".
2. La fecha de nacimiento de los pintores que son maestros.
3. La fecha de nacimiento de los maestros que tienen maestro.
4. La fecha de nacimiento de los pintores que han pintado algún cuadro al "oleo".
5. La fecha de nacimiento de los pintores que han pintado todos sus cuadros al "oleo".
6. La fecha de nacimiento de los pintores que han pintado un cuadro titulado "autorretrato" y que éste ha sido expuesto en alguna exposición.
7. Cuadros que han sido expuestos en al menos 2 exposiciones.
8. Los cuadros que han sido expuestos en alguna de las exposiciones del museo "Pardo".
9. Los maestros que enseñan a solo un pintor.
10. Los maestros que enseñan a más de un pintor.
11. Los cuadros que han sido expuestos en todas las exposiciones del museo "Pardo".

Consultas álgebra relacional:

1. La fecha de nacimiento de los pintores que han pintado un cuadro titulado "autorretrato".

   $\Pi_{fechaNac}((\Pi_{nombrePintor}(\sigma_{(nombre = \text{"autorretrato"})} Cuadros)) \bowtie_{(nombrePintor = nombre)} Pintores)$

2. La fecha de nacimiento de los pintores que son maestros.

   $\Pi_{fechaNac}(\rho_{(nombreMaestro2)}(\Pi_{nombreMaestro} Pintores) \bowtie_{(nombreMaestro2 = nombre)} Pintores)$

   $\Pi_{fechaNac}(\Pi_{nombreMaestro} Pintores \bowtie_{(nombreMaestro = nombre)} \Pi_{(nombre, fechaNac)} Pintores)$

3. La fecha de nacimiento de los maestros que tienen maestro.

   $\Pi_{fechaNac}(\rho_{(nombreMaestro2)}(\Pi_{nombreMaestro} Pintores) \bowtie_{(nombreMaestro2 = nombre)} \sigma_{(nombreMaestro != NULL)} Pintores)$

4. La fecha de nacimiento de los pintores que han pintado algún cuadro al "oleo".

   $\Pi_{fechaNac}(\sigma_{(técnica = \text{"oleo"})}(\Pi_{nombreMaestro} Pintores \bowtie_{(nombre = nombrePintor)} \rho_{(nombrePintor, nombreCuadro, siglo, técnica)} Cuadros)$

5. La fecha de nacimiento de los pintores que han pintado todos sus cuadros al "oleo".

   $\Pi_{fechaNac}(Pintores \bowtie_{(nombre == nombre1)} \rho_{(nombre1)}(\Pi_{nombre} Pintores \setminus \Pi_{nombrePintor}(\sigma_{(técnica != \text{"oleo"})} Cuadros))))$

6. La fecha de nacimiento de los pintores que han pintado un cuadro titulado "autorretrato" y que éste ha sido expuesto en alguna exposición.

   $\Pi_{fechaNac}(Pintores \bowtie_{(nombre = nombrePintor)}(\Pi_{nombrePintor}(\sigma_{(nombreCuad = \text{"autorretrato"})} Asignados))$

7. Cuadros que han sido expuestos en al menos 2 exposiciones.

   $\Pi_{nP1,nC1}\sigma_{(nE1 != nE2)}((\rho_{(nE1, nP1, nC1, p1)} Asignados) \bowtie_{(nP1 == nP2 \wedge nC1 == nC2)}(\rho_{(nE2, nP2, nC2, p2)} Asignados))$

8. Los cuadros que han sido expuestos en alguna de las exposiciones del museo "Pardo".

   $\Pi_{nombrePintor, nombreCuad}(Asignados \bowtie_{(nombreExp == nombre)}(\Pi_{nombre} \sigma_{(nombreMuseo == \text{'Pardo'})} Exposiciones))$

9. Los maestros que enseñan a más de un pintor.

   $\Pi_{nM1}\sigma_{(nP1 != nP2)}((\rho_{(nP1, f1, nM1)} Pintores) \bowtie_{(nM1 == nM2)}(\rho_{(nP2, f2, nM2)} Pintores))$

10. Los maestros que enseñan a solo un pintor.

    $\Pi_{nombreMaestro} Pintores \setminus (\Pi_{nM1}\sigma_{(nP1 != nP2)}((\rho_{(nP1, f1, nM1)} Pintores) \bowtie_{(nM1 == nM2)}(\rho_{(nP2, f2, nM2)} Pintores)))$

11. Los cuadros que han sido expuestos en todas las exposiciones del museo "Pardo".

    $(\Pi_{nombreExp, nombrePintor, nombreCuad} Asignados) \% (\rho_{(nombreExp)} \Pi_{nombre} \sigma_{(nombreMuseo == \text{'Pardo'})} Exposiciones)$

Consultas SQL:

La fecha de nacimiento de los pintores que son maestros.

```
-- Opción igual que la del álgebra relacional.
SELECT p2.fechaNac
FROM Pintores p1 JOIN Pintores p2 ON p2. nombre = p1.nombreMaestro;

-- Opción igual que la del álgebra relacional. SIN DUPLICADOS
SELECT DISTINCT p2.fechaNac
FROM Pintores p1 JOIN Pintores p2 ON p2. nombre = p1.nombreMaestro;

-- Opción igual que la del álgebra relacional, un poco mejor.
SELECT fechaNac
FROM Pintores p1 JOIN
        (SELECT nombreMaestro FROM Pintores) p2 ON p1. nombre = p2.nombreMaestro;

-- Opción más eficiente y sin duplicados. BUENA
SELECT fechaNac
FROM Pintores p
WHERE p.nombre IN (SELECT nombreMaestro
                   FROM Pintores);
```

Consultas SQL:

Pintores cuyos cuadros no han sido expuestos en ningún museo.

```
-- Opción igual que la del álgebra relacional.
SELECT nombre
FROM Pintores
   MINUS
SELECT nombrePintor
FROM Asignados;
```

```
-- Opción más eficiente.  BUENA
SELECT nombre
FROM Pintores p
WHERE p.nombre NOT IN
            (SELECT nombrePintor
             FROM Asignados);
```

Cuadros que han sido expuestos en al menos 2 exposiciones.

```
-- Opción igual que la del álgebra relacional.
SELECT a1.nombrePintor, a1.nombreCuad
FROM Asignados a1
        JOIN Asignados a2
            ON (a1.nombrePintor = a2.nombrePintor
                AND a1.nombreCuad = a2.nombreCuad)
WHERE a1.nombreExp != a2. nombreExp;
```

```
-- Opción más eficiente.  BUENA
SELECT nombrePintor, nombreCuad
FROM Asignados
GROUP BY nombrePintor, nombreCuad
HAVING count(*) >1;
```

Consultas SQL:

Todos los cuadros que han sido expuestos en alguna exposición.

```
-- Opción igual que la del álgebra relacional.
SELECT nombrePintor, nombreCuad
FROM Asignados;
```

Los cuadros que han sido expuestos en alguna de las exposiciones del museo "Pardo".

```
SELECT nombrePintor, nombreCuad
FROM Asignados a
WHERE a.nombreExp IN    (SELECT nombre
                         FROM Exposiciones
                         WHERE nombreMuseo = 'Pardo');
```

Los cuadros que han sido expuestos en todas las exposiciones del museo "Pardo"

```
SELECT nombrePintor, nombreCuad
FROM Asignados JOIN Exposiciones
            ON Asignados.nombreExp = Exposiciones.nombre
WHERE nombreMuseo = 'Pardo'
GROUP BY nombrePintor, nombreCuad
HAVING COUNT(*) =    (SELECT count(*)
                      FROM Exposiciones
                      WHERE nombreMuseo = 'Pardo');
```

Consultas SQL:

Las exposiciones sin cuadros.

```
-- Opción igual que la del álgebra relacional.
SELECT nombre
FROM Exposiciones
   MINUS
SELECT nombreExp
FROM ENERO17_Asignados;
```

```
-- Opción más eficiente.  BUENA
SELECT nombre
FROM Exposiciones e
WHERE e.nombre NOT IN (SELECT nombreExp
                       FROM Asignados);
```

Los pupilos sin maestros.

```
-- Opción igual que la del álgebra relacional.
SELECT nombre
FROM Pintores
WHERE nombreMaestro IS NULL;
```

Los pintores sin maestros.

```
-- Opción igual que la del álgebra relacional.
SELECT nombre
FROM Pintores
WHERE nombreMaestro IS NULL;
```

Restricciones de datos / Disparadores

Ni las técnicas de pintura posibles (óleo, acuarela, acrílico o carboncillo).

```
ALTER TABLE Cuadros ADD
    CONSTRAINT CK_Tecnica
    CHECK (TECNICA IN ('óleo', 'acuarela', 'acrílico', 'carboncillo',
'desconocida'));
```

Procedimientos (PL/SQL)

Crea una tabla de maestros con el número de alumnos que tiene cada maestro. Realiza un procedimiento que inicialice los datos de dicha tabla utilizando los datos de la tabla Enero17_Pintores.

```
CREATE TABLE MaestrosLog(
    nombre VARCHAR2(30) PRIMARY KEY
                    CONSTRAINT FK_maestrosLog REFERENCES Pintores (nombre)
                                                        ON DELETE CASCADE
    ,numAlumnos INTEGER
    );

 CREATE OR REPLACE PROCEDURE inicializaMaestrosLog IS
      CURSOR listaMaestros IS
            SELECT nombreMaestro, count(*) AS numAlumnos
            FROM Pintores
            WHERE nombreMaestro IS NOT NULL
            GROUP BY nombreMaestro;
      BEGIN
          FOR maestro IN listaMaestros LOOP
                INSERT INTO MaestrosLog VALUES maestro;
          END LOOP;
      END Enero17_insertarMaestrosLog;
/

BEGIN inicializaMaestrosLog; END; /
```

Funciones (PL/SQL)

Crea una función simple que dado un nombre devuelva el número de alumnos de dicho maestro.
Debe devolver 0 tanto si no se encuentra en la base de datos como si no es maestro de ningún alumno.

```
-- FUNCION SIMPLE Y SU PRUEBA: Cardinal del maestro de la tabla Pintores (0,4).
CREATE OR REPLACE FUNCTION numAlumnos(maestro Pintores.nombreMaestro%TYPE)
     RETURN INT IS
  numPupilos INT;
BEGIN
   SELECT COUNT(*) INTO numPupilos
   FROM Pintores p
   WHERE p.nombreMaestro = maestro;

   RETURN (numPupilos);
 END numAlumnos;
/
-- OPCIÓN 1: Prueba de la función
BEGIN
  DBMS_OUTPUT.put_line('Alberto = '|| TO_CHAR(numAlumnos('Alberto'),'999'));
  DBMS_OUTPUT.put_line('TestP0  = '|| TO_CHAR(numAlumnos('TestP0'),'999'));
  DBMS_OUTPUT.put_line('TestP10 = '|| TO_CHAR(numAlumnos('TestP10'),'999'));
END;
/
```

Trigger (Disparadores)

Manteniendo la tabla MaestrosLog actualizada con las inserciones y actualizaciones.

```
CREATE OR REPLACE TRIGGER CardinalMaestro
  AFTER INSERT OR UPDATE OF nombreMaestro
     ON Pintores
  FOR EACH ROW -- trigger
 DECLARE
  numAlumnosNEW INT := numPupilos(:new.nombreMaestro);
  numAlumnosOLD INT := numPupilos(:old.nombreMaestro);
 BEGIN
  IF INSERTING AND :new.nombreMaestro IS NOT NULL THEN
                        Update_MaestrosLog(:new.nombreMaestro, numAlumnosNEW + 1);
  ELSIF UPDATING THEN
    IF :new.nombreMaestro IS NULL AND :old.nombreMaestro IS NOT NULL THEN
                        Update_MaestrosLog(:old.nombreMaestro, numAlumnosOLD - 1);
    ELSIF :new.nombreMaestro IS NOT NULL AND :old.nombreMaestro IS NULL THEN
                        Update_MaestrosLog(:new.nombreMaestro, numAlumnosNEW + 1);
    ELSIF :new.nombreMaestro IS NOT NULL AND :old.nombreMaestro IS NOT NULL
        AND :new.nombreMaestro != :old.nombreMaestro THEN
                        Update_MaestrosLog(:old.nombreMaestro, numAlumnosOLD - 1);
                        Update_MaestrosLog(:new.nombreMaestro, numAlumnosNEW + 1);
    END IF;
  END IF;
 END Enero17_CardinalMaestro;
 /
```

PL/SQL y Trigger (Disparadores)

Manteniendo la tabla MaestrosLog actualizada con las inserciones y actualizaciones.

```
CREATE OR REPLACE PROCEDURE Update_MaestrosLog_v1
          (maestro Pintores.nombreMaestro%TYPE, nAlumnos INT) IS
   excepcionMuchosPupilos EXCEPTION; -- Creación de excepción de usuario
BEGIN
  IF  nAlumnos > 4 THEN
           RAISE  excepcionMuchosPupilos; -- Lanzar la excepción de usuario
  ELSIF nAlumnos = 0 THEN
       DELETE FROM Pintores p
       WHERE p.nombre = maestro;
  ELSIF nAlumnos = 1 AND numPupilos(maestro) = 0 THEN
    INSERT INTO MaestrosLog VALUES (maestro, nAlumnos);
  ELSIF nAlumnos > 0 THEN
     UPDATE MaestrosLog p
       SET p.numAlumnos = nAlumnos
     WHERE p.nombre = maestro;
  END IF;

  EXCEPTION -- Control de excepciones lanzadas
        WHEN excepcionMuchosPupilos THEN
          DBMS_OUTPUT.put_line('Demasiados pupilos para el maestro de nombre: '||maestro);
          -- RAISE  excepcionMuchosPupilos; -- Se puede relanzar o lanzar excepción de aplicación
          RAISE_APPLICATION_ERROR(-20201, 'Demasiados pupilos para el maestro de nombre: '||maestro);
END Update_MaestrosLog_v1;
/
```

PL/SQL y Trigger (Disparadores)

Manteniendo la tabla MaestrosLog actualizada con las inserciones y actualizaciones.

```
CREATE OR REPLACE PROCEDURE Update_MaestrosLog_v21 (maestro Enero17_Pintores.nombreMaestro%TYPE, nAlumnos INT) IS
   excepcionMuchosPupilos EXCEPTION; -- Creación de excepción de usuario
   numAlums NUMBER;
BEGIN
  -- Si no está no se realiza la actualización pero NO lanza excepcion
  UPDATE MaestrosLog p
     SET p.numAlumnos = p.numAlumnos + nAlumnos
  WHERE p.nombre = maestro;

  -- En Oracle SQL%ROWCOUNT hace referencia a cuantas filas fueron con la última instrucción. En este caso cuantas han sido actualizadas.
  -- Si no se actualiza ninguna fila lo inserto
  IF SQL%ROWCOUNT = 0 THEN
    INSERT INTO MaestrosLog VALUES (maestro, nAlumnos);
  END IF;
  -- Puedo asegurar que el maestro ya es encuentra en la base de datos. Puede que con valores negativos o 0, pero ya está.

  numAlums := numPupilos(maestro);

  IF  numAlums > 4 THEN
            RAISE  excepcionMuchosPupilos; -- Lanzar la excepción de usuario
            -- RAISE_APPLICATION_ERROR(-20201, 'Demasiados pupilos para el maestro de nombre: '||:new.nombreMaestro);
  ELSIF numAlums <= 0 THEN
       DELETE FROM MaestrosLog
       WHERE nombre = maestro;
  END IF;
    EXCEPTION -- Control de excepciones lanzadas
         WHEN excepcionMuchosPupilos THEN
           DBMS_OUTPUT.put_line('Demasiados pupilos para el maestro de nombre: '||maestro);
           -- RAISE  excepcionMuchosPupilos; -- Se puede relanzar o lanzar excepción de aplicación
           -- Las excepciones deshacen todo el proceso (ROLLBACK)
           RAISE_APPLICATION_ERROR(-20201, 'Demasiados pupilos para el maestro de nombre: '||maestro);
       WHEN OTHERS THEN DBMS_OUTPUT.put_line('No existe el maestro de nombre: '||maestro);
END Update_MaestrosLog_v21;
/
```

PL/SQL y Trigger (Disparadores)

Manteniendo la tabla MaestrosLog actualizada con las inserciones y actualizaciones.

```
CREATE OR REPLACE PROCEDURE Update_MaestrosLog_v22 (maestro Pintores.nombreMaestro%TYPE, nAlumnos INT) IS
   excepcionMuchosPupilos EXCEPTION; -- Creación de excepción de usuario
   numAlums NUMBER;
BEGIN
   numAlums := Enero17_numPupilos(maestro);
   -- Como ya he calculado el numAlums decido si añadirlo, actualizarlo, borrarlo o no hacer NADA
   IF numAlums = 0 THEN -- No está incluido en tabla MaestrosLog
      IF 0< nAlumnos AND nAlumnos <= 4 THEN
         -- Puede que falle ya que el maestro no este en la tabla Pintores
         INSERT INTO MaestrosLog VALUES (maestro, nAlumnos);
      END IF;
   ELSIF 0 < numAlums + nAlumnos AND numAlums + nAlumnos <= 4 THEN
       UPDATE MaestrosLog p
          SET p.numAlumnos = p.numAlumnos + nAlumnos
        WHERE p.nombre = maestro;
   ELSIF 0 = numAlums + nAlumnos THEN
      DELETE FROM MaestrosLog
      WHERE nombre = maestro;
   ELSE
      RAISE excepcionAddPupilos; -- Lanzar la excepción de usuario
   END IF;

   EXCEPTION -- Control de excepciones lanzadas
      WHEN excepcionAddPupilos THEN
         DBMS_OUTPUT.put_line('Demasiados pupilos para el maestro de nombre: '||maestro);
         -- RAISE; -- Se puede relanzar o lanzar excepción de aplicación
         -- Las excepciones deshacen todo el proceso (ROLLBACK)
         RAISE_APPLICATION_ERROR(-20201, 'Demasiados pupilos para el maestro de nombre: '||maestro);
      WHEN OTHERS THEN DBMS_OUTPUT.put_line('No existe el maestro de nombre: '||maestro);
END Update_MaestrosLog_v22;
/
```

Trigger compuesto (Disparador)

Manteniendo la tabla MaestrosLog actualizada con las eliminaciones. (Compound trigger)

```
CREATE OR REPLACE TRIGGER CardinalMaestro_DELETE
  FOR DELETE ON Pintores
WHEN (old.nombreMaestro IS NOT NULL)
COMPOUND TRIGGER -- Trigger compuesto, ya que necesita consultar una tabla en modificación
   numAlumnosOLD INT :=0;
   maestro Pintores.nombreMaestro%TYPE;

AFTER EACH ROW IS -- trigger
 BEGIN
      maestro := :old.nombreMaestro;
 END AFTER EACH ROW;

AFTER STATEMENT IS -- A realizar después de la modificación
 BEGIN
    numAlumnosOLD := numPupilos(maestro);
   Update_MaestrosLog(maestro, numAlumnosOLD - 1);
 END AFTER STATEMENT;

END CardinalMaestro_DELETE;
/
```