Base de donnée pour la gestion d’un musée

Groupe 9

Aurore Martorana - 22105397

Spécifications

Cette base de données a été pensée pour la gestion des œuvres d'un musée imaginaire, ainsi que de ses visiteurs. Je ne me suis pas penchée sur le personnel qui pourrait y travailler, ou des événements qui pourraient y être organisés par exemple.

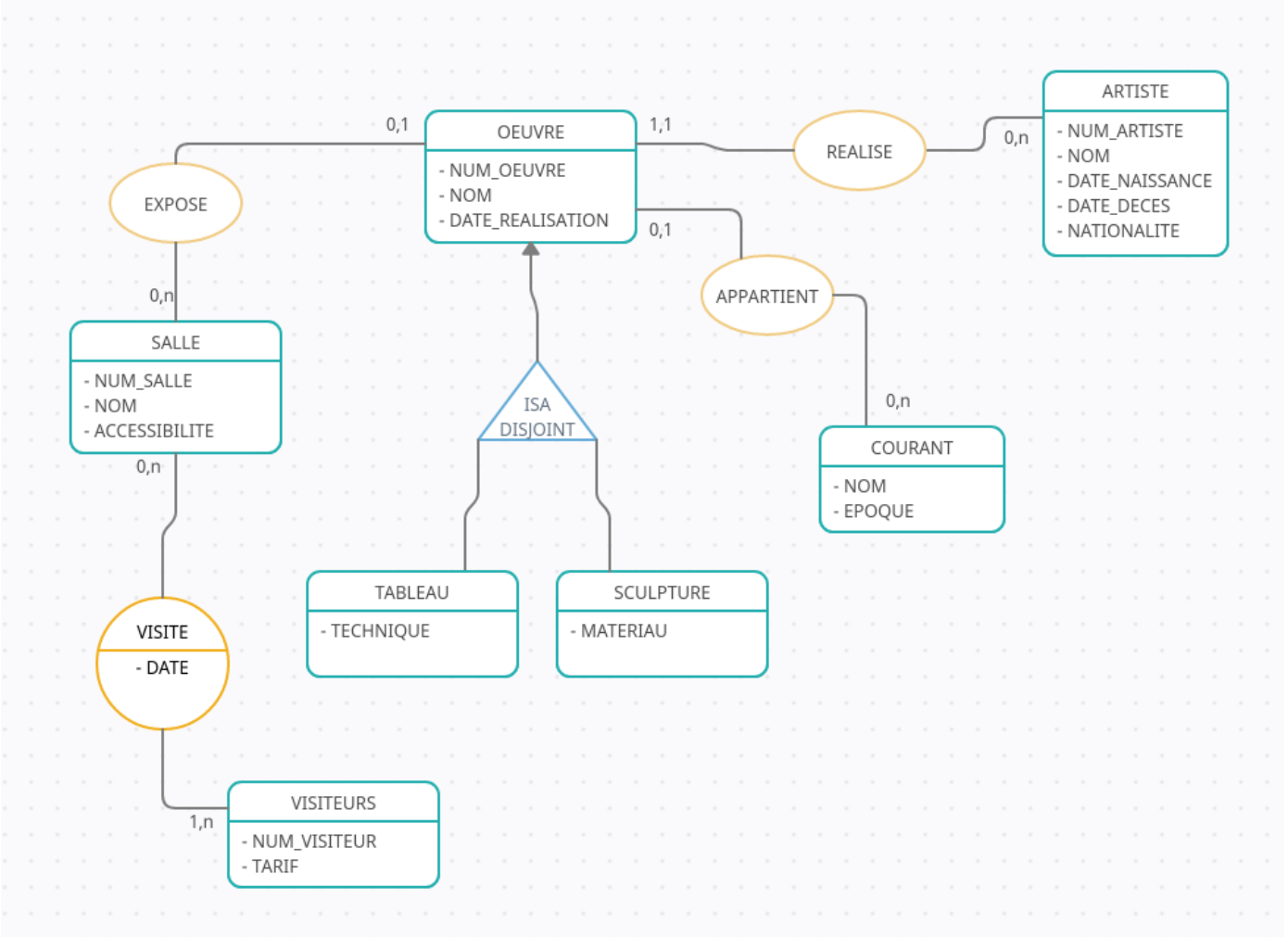
Concernant les spécifications de mon modèle :
Le musée comporte des salles qui sont identifiées par un numéro, un nom et leur accessibilité (ouverte ou fermée) que les visiteurs, identifiés par un numéro et un tarif, peuvent visiter à une certaine date. Une œuvre est identifiée par un numéro, un nom, une date de réalisation et ses dimensions. Elle peut être exposée dans une salle, et est soit une sculpture, avec l'attribut matériau, soit un tableau avec l'attribut technique. Une oeuvre peut appartenir à un courant artistique caractérisé par son nom et une époque, et a été réalisée par un artiste, qui a un numéro, un nom, une date de naissance et de décès, et une nationalité.

Dictionnaire des données

NOM	DESCRIPTION	TYPE	LONGUEUR	NATURE	CONTRAINTE
NUM_OEUVRE	Identifiant d'une oeuvre	varchar	10	Unique	identifiant
OEUVRE.NOM	Nom de l'oeuvre si elle en a un	varchar	25	Facultatif	
DATE_REALISATION	Date de réalisation d'une oeuvre	numeric	4,0	Obligatoire	not null
DIMENSION	Dimensions d'une oeuvre (en cm)	varchar	25	Obligatoire	not null
TECHNIQUE	Technique utilisée pour un tableau	varchar	25	Obligatoire	not null
MATÉRIAU	Matériau d'une sculpture	varchar	25	Obligatoire	not null
NUM_ARTISTE	Identifiant de l'artiste	varchar	10	Unique	identifiant
ARTISTE.NOM	Nom de l'artiste	varchar	25	Obligatoire	not null
DATE_NAISSANCE	Date de naissance de l'artiste	date		Facultatif	
DATE_DECES	Date de décès de l'artiste	date		Facultatif	

NATIONALITE	Nationalité de l'artiste au format "Pays, Continent"	varchar	25	Obligatoire	not null
COURANT.NOM	Nom du courant artistique	varchar	25	Unique	identifiant
EPOQUE	Epoque	varchar	25	Obligatoire	not null
NUM_SALLE	Identifiant de la salle	varchar	10	Unique	identifiant
SALLE.NOM	Nom de la salle ("Europe" ou "Asie" par exemple)	varchar	25	Facultatif	
ACCESSIBILITÉ	Indique si la salle est "ouverte" ou "fermée" au public	varchar	10	Obligatoire	not null
NUM_VISITEUR	Identifiant d'un visiteur	varchar	10	Unique	identifiant
TARIF	Intitulé du tarif ("normal", "étudiant",...)	varchar	25	Obligatoire	not null
DATE_VISITE	Date de la visite d'un visiteur	date		Obligatoire	not null

Schéma entité-association



Explications :

Les œuvres ou courant n'ont pas de description, et les artistes n'ont pas de biographie, par souci de temps et de complexité.
Aussi, j'ai fait le choix qu'une œuvre ne pouvait avoir qu'un seul artiste et appartenir à un seul courant maximum, pour plus de simplicité.
Enfin, les visiteurs visitent des salles et non pas le musée entier dans la relation "Visite", afin d'avoir des statistiques sur le succès des différentes salles.

Schéma Relationnel

Oeuvre(numOeuvre, nom, dateRealisation, dimensions, *#numSalle, #numArtiste, #courant*)
Courant(nom, époque)
Tableau(technique)
Sculpture(matériau)
Artiste(numAriste, nom, dateNaissance, dateDécès, nationalité)
Salle(numSalle, accessibilité)
Visiteur(numVisiteur, tarif)
Visite(*#numVisiteur, #numSalle*, dateVisite)

Schéma Physique

```
CREATE TABLE ARTISTE(
    NUM_ARTISTE VARCHAR(10),
    NOM VARCHAR(25) CONSTRAINT NOM_ARTISTE_NN NOT NULL,
    DATE_NAISSANCE DATE,
    DATE_DECES DATE,
    NATIONALITE VARCHAR(50) CONSTRAINT NAT_ARTISTE_NN NOT NULL,

    CONSTRAINT PK_ARTISTE PRIMARY KEY (NUM_ARTISTE)
);

CREATE TABLE COURANT(
    NOM VARCHAR(25),
    EPOQUE VARCHAR(25) CONSTRAINT EPOQUE_COURANT_NN NOT NULL,

    CONSTRAINT PK_COURANT PRIMARY KEY (NOM)
);

CREATE TABLE SALLE(
    NUM_SALLE VARCHAR(10),
    NOM VARCHAR(25),
    ACCESSIBILITE VARCHAR(10) CONSTRAINT ACCESS_SALLE_NN NOT NULL,

    CONSTRAINT PK_SALLE PRIMARY KEY (NUM_SALLE),
    CONSTRAINT DOM_ACCESS_SALLE CHECK (ACCESSIBILITE IN
('OUVERTE','FERMEE'))
);

CREATE TABLE VISITEUR(
    NUM_VISITEUR VARCHAR(10),
```

```
    TARIF VARCHAR(25) CONSTRAINT TARIF_NN NOT NULL,

    CONSTRAINT PK_VISITEUR PRIMARY KEY (NUM_VISITEUR),
    CONSTRAINT DOM_TARIF CHECK (TARIF IN
('NORMAL','ETUDIANT','MINEUR','ENFANT'))
);

CREATE TABLE VISITE(
    NUM_VISITEUR VARCHAR(10),
    NUM_SALLE VARCHAR(10),
    DATE_VISITE DATE CONSTRAINT DATE_VISITE_NN NOT NULL,

    CONSTRAINT PK_VISITE PRIMARY KEY (NUM_VISITEUR, NUM_SALLE),

    CONSTRAINT FK_VISITE_VISITEUR FOREIGN KEY (NUM_VISITEUR) REFERENCES
VISITEUR(NUM_VISITEUR) ON DELETE CASCADE,
    CONSTRAINT FK_VISITE_SALLE FOREIGN KEY (NUM_SALLE) REFERENCES
SALLE(NUM_SALLE) ON DELETE CASCADE
);

CREATE TABLE OEUVRE(
    NUM_OEUVRE VARCHAR(10),
    NOM VARCHAR(75),
    DATE_REALISATION NUMERIC(4,0),
    DIMENSIONS VARCHAR(25) CONSTRAINT DIM_NN NOT NULL,

    NUM_SALLE VARCHAR(10),
    NUM_ARTISTE VARCHAR(10),
    COURANT VARCHAR(25),

    CONSTRAINT PK_OEUVRE PRIMARY KEY (NUM_OEUVRE),
    CONSTRAINT FK_OEUVRE_SALLE FOREIGN KEY (NUM_SALLE) REFERENCES
SALLE(NUM_SALLE) ON DELETE CASCADE,
    CONSTRAINT FK_OEUVRE_ARTISTE FOREIGN KEY (NUM_ARTISTE) REFERENCES
ARTISTE(NUM_ARTISTE) ON DELETE CASCADE,
    CONSTRAINT FK_OEUVRE_COURANT FOREIGN KEY (COURANT) REFERENCES
COURANT(NOM) ON DELETE CASCADE
);

CREATE TABLE SCULPTURE(
    NUM_OEUVRE VARCHAR(10),
    MATERIAU VARCHAR(25) CONSTRAINT MAT_NN NOT NULL,
```

```
    CONSTRAINT PK_SCULPTURE PRIMARY KEY (NUM_OEUVRE),
    CONSTRAINT FK_SCULPTURE_OEUVRE FOREIGN KEY (NUM_OEUVRE) REFERENCES
OEUVRE(NUM_OEUVRE) ON DELETE CASCADE
);

CREATE TABLE TABLEAU(
    NUM_OEUVRE VARCHAR(10),
    TECHNIQUE VARCHAR(50) CONSTRAINT TECH_NN NOT NULL,

    CONSTRAINT PK_PEINTURE PRIMARY KEY (NUM_OEUVRE),
    CONSTRAINT FK_TABLEAU_OEUVRE FOREIGN KEY (NUM_OEUVRE) REFERENCES
OEUVRE(NUM_OEUVRE) ON DELETE CASCADE
);
```

Requêtes

Requête 1:
Le nom et le matériau de toutes les sculptures dont l'artiste est Africain.

```
SELECT OEUVRE.NOM AS NOM_OEUVRE, MATERIAU, ARTISTE.NOM AS NOM_ARTISTE
FROM SCULPTURE JOIN OEUVRE ON SCULPTURE.NUM_OEUVRE = OEUVRE.NUM_OEUVRE
JOIN ARTISTE ON OEUVRE.NUM_ARTISTE = ARTISTE.NUM_ARTISTE AND
NATIONALITE LIKE '%Afrique';
```

- Résultat attendu :

NOM_OEUVRE	MATERIAU	NOM_ARTISTE
Anyanwu	Bronze	Ben Enwonwu
Atlas	Bronze	Ben Enwonwu

- Explication :

Comme la nationalité est au format 'Pays, Continent', on peut facilement retrouver les artistes africains avec LIKE '%Afrique'.

Requête 2:
Le nom du courant artistique ainsi que le nombre d'œuvres qui y appartiennent, pour les courants représentés par au moins 2 œuvres.

```
SELECT COURANT, COUNT(COURANT) AS NOMBRE_D_OEUVRES FROM OEUVRE GROUP BY
COURANT HAVING COUNT(COURANT) >= 2 ORDER BY NOMBRE_D_OEUVRES DESC;
```

- Résultat attendu :

COURANT NOMBRE_D_OEUVRES

Surréalisme 4
Contemporain 3
Modernisme 2

- Explication

On GROUP BY le nom des courants puis on utilise HAVING sur le résultat pour retirer du résultat les courants qui ne sont pas représentés par deux œuvres ou plus. Aussi, on ordonne le résultat par le nombre d'œuvres descendant.

Requête 3 :

Le nom des artistes qui ont réalisé des oeuvres de différents courants artistiques :

```
SELECT A1.NOM FROM ARTISTE A1 JOIN OEUVRE O1 ON A1.NUM_ARTISTE =
O1.NUM_ARTISTE WHERE EXISTS ( SELECT * FROM ARTISTE A2 JOIN OEUVRE O2
ON  A2.NUM_ARTISTE = O2.NUM_ARTISTE AND A1.NUM_ARTISTE = A2.NUM_ARTISTE
AND O1.COURANT <> O2.COURANT) GROUP BY A1.NUM_ARTISTE, A1.NOM;
```

- Résultat attendu :

NOM

Pablo Picasso

- Explication :

On utilise une sous requête corrélative pour sélectionner l'artiste pour lequel il existe deux œuvres au moins avec des courants différents, et on GROUP BY le numéro et le nom au cas où il existe des homonymes

Requête 4:

Le nom des oeuvres exposées dans la salle qui expose le plus d'oeuvres

```
SELECT NOM FROM OEUVRE WHERE NUM_SALLE IN (SELECT NUM_SALLE FROM OEUVRE
GROUP BY NUM_SALLE HAVING COUNT(NUM_SALLE) >= ALL(SELECT
COUNT(NUM_SALLE) FROM OEUVRE GROUP BY NUM_SALLE));
```

- Résultat attendu :

NOM

Untitled
Judith II
Les amants
Tempete de neige en mer
Nu bleu II
Mordor VII
La Celestina
Nu couche I
Jean de la Fontaine

9 lignes sélectionnées

- Explication :

On utilise deux requêtes imbriquées, la dernière sous requête donne le nombre d'œuvres exposées dans chaque salle, et la sous requête dans laquelle elle est imbriquée donne le numéro de la salle, ou des salles, qui expose le plus d'œuvres avec >= ALL. On aurait pu utiliser MAX() dans la deuxième sous requête imbriquée.

Requête 5:
Les visiteurs qui ont visité toutes les salles du musée

```
SELECT * FROM VISITEUR V1 WHERE NOT EXISTS ( SELECT * FROM SALLE S JOIN
VISITE ON VISITE.NUM_SALLE = S.NUM_SALLE WHERE NOT EXISTS ( SELECT *
FROM VISITEUR V2 JOIN VISITE ON V2.NUM_VISITEUR = VISITE.NUM_VISITEUR
AND S.NUM_SALLE = VISITE.NUM_SALLE AND V1.NUM_VISITEUR =
V2.NUM_VISITEUR));
```

- Résultat attendu

NUM_VISITEUR TARIF

VIS6 NORMAL

- Explication :

On effectue une division avec un double NOT EXISTS. On veut sélectionner les visiteurs pour lesquels il n’existe pas de salle pour laquelle il n’existe pas de visite qui aie été faite dans cette salle par ces visiteurs en question. Autrement dit on souhaite les visiteurs telle qu’il existe forcément des visites pour chacune des salles.

Procédures, fonctions et triggers

Fonction 1:
Cette fonction renvoie le nombre de visiteurs de la journée pour une certaine date passée en paramètre.

```
CREATE OR REPLACE FUNCTION nbr_visite(date_cible IN DATE)
    RETURN NUMBER
AS
    total NUMBER;
BEGIN
    SELECT COUNT(DISTINCT NUM_VISITEUR) INTO total
        FROM VISITE
        WHERE TO_CHAR(DATE_VISITE, 'YYYY-MM-DD') = TO_CHAR(date_cible,
'YYYY-MM-DD');

    RETURN(total);
END;
/
```

Fonction 2:
Cette fonction renvoie les recettes de la journée pour une certaine date passée en paramètre.

```
CREATE OR REPLACE FUNCTION recette_journee(date_cible IN DATE)
    RETURN NUMBER
AS
    total NUMBER;
BEGIN
    total := 0;

    FOR visiteur IN (
        SELECT DISTINCT VISITEUR.NUM_VISITEUR, TARIF
        FROM VISITEUR INNER JOIN VISITE ON VISITEUR.NUM_VISITEUR =
VISITE.NUM_VISITEUR
        WHERE TO_CHAR(DATE_VISITE, 'YYYY-MM-DD') = TO_CHAR(date_cible,
'YYYY-MM-DD'))
    LOOP
        IF visiteur.TARIF = 'NORMAL' THEN
            total := total + 12;
        ELSIF visiteur.TARIF = 'ETUDIANT' THEN
            total := total + 10;
```

```
        ELSIF visiteur.TARIF = 'MINEUR' THEN
            total := total + 8;
        END IF;
    END LOOP;

    RETURN(total);
END;
/
```

Procédure 1:

Cette procédure affiche le nombre de visiteurs, ainsi que les recettes de la journée pour une date donnée en paramètre.

```
CREATE OR REPLACE PROCEDURE statistiques (date_cible IN DATE) AS
    visites NUMBER;
    recette NUMBER;
BEGIN
    recette := recette_journee(date_cible);
    visites := nbr_visite(date_cible);

    dbms_output.put_line('Statistiques du ' || TO_CHAR(date_cible,
'DL', 'NLS_DATE_LANGUAGE = FRENCH') || ':');
    dbms_output.put_line('Nombre de visiteurs: ' || visites);
    dbms_output.put_line('Recettes totales: ' || recette);
END;
/
```

Trigger 1:

Le trigger vérifie à l'insertion d'un artiste que sa date de naissance ne dépasse pas sa date de décès. On compare donc ces deux attributs.

```
CREATE OR REPLACE TRIGGER check_age_artiste
    BEFORE INSERT
    ON ARTISTE
    FOR EACH ROW

DECLARE
    erreur_age_artiste EXCEPTION;

BEGIN
```

```
    IF :new.DATE_NAISSANCE > :new.DATE_DECES
        THEN RAISE erreur_age_artiste;
    END IF;

    EXCEPTION
        WHEN erreur_age_artiste THEN RAISE_APPLICATION_ERROR(-20001,
'La date de naissance est superieure a la date de deces');

END;
/
```

Trigger 2:

Le trigger vérifie que si on INSERT une nouvelle visite à la date d'aujourd'hui, il y a une exception si la visite s'effectue dans une salle 'FERMEE'.

```
CREATE OR REPLACE TRIGGER check_salle_ouverte
    BEFORE INSERT
    ON VISITE
    FOR EACH ROW
DECLARE
    etat SALLE.ACCESSIBILITE%TYPE;
    erreur_salle_indisponible EXCEPTION;
BEGIN
    IF :new.DATE_VISITE = SYSDATE THEN

        SELECT ACCESSIBILITE INTO etat FROM SALLE WHERE NUM_SALLE =
:new.NUM_SALLE;
        IF etat = 'FERMEE'
            THEN RAISE erreur_salle_indisponible;
        END IF;
    END IF;

    EXCEPTION
        WHEN erreur_salle_indisponible THEN
RAISE_APPLICATION_ERROR(-20002, 'La visite est impossible dans une
salle fermee');
END;
/
```

Tests

- Test de la procédure 1 : statistiques

On veut afficher les statistiques pour la date d'aujourd'hui

```
EXEC statistiques(SYSDATE);
```

- Test de la fonction 1 et 2 : nbr_visite et recettes_journee

Ces fonctions sont appelées dans la procédure 'statistique'.

- Test du trigger 1 : check_age_artiste

On INSERT un nouvel artiste avec une date de naissance supérieure à sa date de décès.

```
INSERT INTO ARTISTE VALUES ('ART100', 'Artiste errone', '01-01-2020',
'01-01-1200', 'France, Europe');
```

- Test du trigger 2 : check_salle_ouverte

On INSERT une nouvelle visite dans la salle 2 (fermée) pour le visiteur 10 à la date d'aujourd'hui

```
INSERT INTO VISITE VALUES ('VIS10', 'SAL2', SYSDATE);
```