Sujet : Réservation de matériel

Présentation du projet

Dans cet exercice, nous allons créer une base de données pour gérer la réservation de matériel pour une école d'ingénieur. La base de données contiendra des informations sur les étudiants, le matériel disponible et les réservations effectuées

Révisions :

Auteur	Version	Date	Commentaire
A. PIVIDORI	1.0	01/03/2023	Création dû TP
A. PIVIDORI	1.1	01/03/2024	Simplification de l'exercice 8.

Consignes de rendu :

- Date de rendu : À définir lors du premier TP.
- Format du rendu :
    - 1 Fichier contenant les éléments d'analyses demandés en première partie :
        - Analyse des besoins
        - Modélisation des données
        - Modélisation logique
    - 1 dossier contenant 1 script par exercice. Les fichiers devront être nommés comme suit
        - 2024_BDD_NOM_PRENOM_EX3
        - 2024_BDD_NOM_PRENOM_EX1_2 ; (pour les sous-parties)
    - Ces fichiers devront se trouver sur un GIT (public) : vous devrez juste envoyer le lien du GIT en fin de TP à andrey.pividori@worldline.com avec comme Objet : [BDD4][Nom][Prenom] - Rendu de TP

Analyse des besoins :

Identifiez les entités principales impliquées dans le système de réservation de matériel, par exemple : Utilisateur, Matériel, Réservation. Identifiez les attributs nécessaires pour chaque entité, en tenant compte des informations à stocker et à gérer, par exemple : nom, prénom, email pour Utilisateur. Identifiez les associations entre les entités, par exemple : une Réservation est liée à un Utilisateur et à un Matériel.

Modélisation des données :

Utilisez les symboles MERISE pour représenter les entités, les attributs et les associations identifiées. Dessinez un schéma entité-association (ER) représentant les entités, les attributs et les associations, ainsi que les cardinalités entre elles. Identifiez les clés primaires pour chaque entité.

Modélisation logique :

Transformez le schéma entité-association (ER) en un schéma relationnel utilisant les tables pour représenter les entités et les associations. Identifiez les clés primaires et les clés étrangères nécessaires pour définir les relations entre les tables. Documentez le schéma relationnel en spécifiant les types de données appropriés pour chaque attribut.

Conception de la base de données :

La base de données peut être créée en utilisant un logiciel de gestion de base de données relationnelle, tel que MySQL.

Implémentation de la base de données :

Après avoir conçu la base de données, mettez en place en créant les tables et les contraintes de clé étrangère. Les exercices ci-dessous vous permettront de mettre en place des requêtes avec des niveaux de difficultés progressives.

Exercice 1 : Insérer des données dans les tables

Insérez des données dans les tables pour créer des utilisateurs, du matériel, des réservations et des emprunts. Effectuez des requêtes SELECT pour vérifier que les données ont bien été insérées.

1.1 Ajout de données dans la table « Utilisateur »

- Proposez un contenu de la table utilisateur (et documentez le)
- Créez la table Utilisateur
- Complétez là avec au moins 10 utilisateurs

1.2 Ajout de données dans la table « Matériel »

- Proposez un contenu de la table matériel (et documentez le)
- Créez la table Matériel
- Complétez là avec au moins 10 matériel

1.3 Ajout de données dans la table « Reservation »

- Proposez un contenu de la table réservation (et documentez le)
- Une réservation est encadrée par une date de début et une date de fin, proposez le tableau de description.
- Créez la table Reservation
- Complétez là avec au moins 5 réservations

Exercice 2 : Effectuer des requêtes de sélection

Effectuez des requêtes SELECT pour récupérer des informations sur les utilisateurs, le matériel, les réservations et les emprunts. Utilisez des clauses WHERE pour filtrer les résultats en fonction de critères spécifiques.

- Au moins 3 requêtes sont attendues (avec au moins une par table et comprenant des clauses WHERE)

Exercice 3 : Effectuer des requêtes de jointure

Effectuez des requêtes de jointure pour combiner les données de plusieurs tables.

Exemple : Récupérez les informations sur les utilisateurs ayant effectué une réservation donnée, ou encore les informations sur le matériel emprunté par un utilisateur donné.

- Une requête de jointure sur les utilisateurs ayant effectué une réservation
- Une requête de jointure pour récupérer les informations sur le matériel emprunté par un utilisateur donné

Exercice 4 : Effectuer des requêtes d'agrégation

Effectuez des requêtes d'agrégation pour calculer des statistiques sur les données.

Exemple : Calculez le nombre total de réservations effectuées sur une période donnée, ou encore le nombre d'utilisateurs ayant emprunté du matériel donné.

- Requête d'aggrégation pour calculer le nombre total de réservations effectuées sur une période donnée
- Requête d'aggrégation pour calculer le nombre d'utilisateur ayant emprunté du matériel

Exercice 5 : Mettre à jour les données

Effectuez des requêtes UPDATE pour mettre à jour les données dans les tables.

Exemple : Modifiez la quantité disponible d'un matériel après un emprunt, ou encore annuler une réservation existante.

- Requête de modification de la quantité disponible d'un matériel après un emprunt

Exercice 6 : Supprimer des données

Effectuez des requêtes DELETE pour supprimer des données dans les tables.

Exemple : supprimez une réservation existante ou encore retirer un utilisateur du système.

- Requête de suppression d'une réservation existante

Exercice 7 : Requêtes avancées

Effectuer les requêtes suivantes :

- Afficher tous les utilisateurs ayant emprunté au moins un équipement
- Afficher les équipements n'ayant jamais été empruntés
- Afficher les équipements ayant été emprunté plus de 3 fois
- Afficher le nombre d'emprunt pour chaque utilisateur

Aide : Assurez-vous d'avoir tous les pré-requis avant de penser à tester vos requêtes.

Exercice 8 : Gestion des réservations avec contraintes de disponibilité

Contexte : nous souhaitons ajouter des contraintes de disponibilité sur les réservations de matériel. Un matériel ne peut être réservé que s'il est disponible pendant la période spécifiée. Après avoir terminé le TP de "base", vous devez mettre à jour le schéma de la base de données et implémenter les fonctionnalités nécessaires pour gérer ces contraintes.

Instructions :

1. Mettez à jour le modèle de données existant en ajoutant une nouvelle table "disponibilite" avec les colonnes suivantes :

id_disponibilite (clé primaire)
id_materiel (clé étrangère référençant la table "materiel")
date_debut (date de début de la disponibilité)
date_fin (date de fin de la disponibilité)

2. Modifiez la table "reservation" en ajoutant une nouvelle colonne "id_disponibilite" (clé étrangère référençant la table "disponibilite").

3. Modifiez les contraintes SQL existantes pour prendre en compte les contraintes de disponibilité lors de la création et de la mise à jour des réservations.

4. Implémentez une fonctionnalité permettant de vérifier la disponibilité d'un matériel pour une période donnée avant de permettre la réservation. Si le matériel n'est pas disponible, affichez un message d'erreur approprié.

5. Implémentez une fonctionnalité permettant de gérer les disponibilités du matériel. Les administrateurs doivent pouvoir ajouter, modifier et supprimer des périodes de disponibilité pour chaque matériel.

6. Testez votre application en effectuant des réservations avec différentes périodes pour vérifier que les contraintes de disponibilité sont correctement appliquées.

SQL CHEAT SHEET http://www.sqltutorial.org

QUERYING DATA FROM A TABLE

```
SELECT c1, c2 FROM t;
```
Query data in columns c1, c2 from a table

```
SELECT * FROM t;
```
Query all rows and columns from a table

```
SELECT c1, c2 FROM t
WHERE condition;
```
Query data and filter rows with a condition

```
SELECT DISTINCT c1 FROM t
WHERE condition;
```
Query distinct rows from a table

```
SELECT c1, c2 FROM t
ORDER BY c1 ASC [DESC];
```
Sort the result set in ascending or descending order

```
SELECT c1, c2 FROM t
ORDER BY c1
LIMIT n OFFSET offset;
```
Skip *offset* of rows and return the next n rows

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1;
```
Group rows using an aggregate function

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1
HAVING condition;
```
Filter groups using HAVING clause

QUERYING FROM MULTIPLE TABLES

```
SELECT c1, c2
FROM t1
INNER JOIN t2 ON condition;
```
Inner join t1 and t2

```
SELECT c1, c2
FROM t1
LEFT JOIN t2 ON condition;
```
Left join t1 and t1

```
SELECT c1, c2
FROM t1
RIGHT JOIN t2 ON condition;
```
Right join t1 and t2

```
SELECT c1, c2
FROM t1
FULL OUTER JOIN t2 ON condition;
```
Perform full outer join

```
SELECT c1, c2
FROM t1
CROSS JOIN t2;
```
Produce a Cartesian product of rows in tables

```
SELECT c1, c2
FROM t1, t2;
```
Another way to perform cross join

```
SELECT c1, c2
FROM t1 A
INNER JOIN t2 B ON condition;
```
Join t1 to itself using INNER JOIN clause

USING SQL OPERATORS

```
SELECT c1, c2 FROM t1
UNION [ALL]
SELECT c1, c2 FROM t2;
```
Combine rows from two queries

```
SELECT c1, c2 FROM t1
INTERSECT
SELECT c1, c2 FROM t2;
```
Return the intersection of two queries

```
SELECT c1, c2 FROM t1
MINUS
SELECT c1, c2 FROM t2;
```
Subtract a result set from another result set

```
SELECT c1, c2 FROM t1
WHERE c1 [NOT] LIKE pattern;
```
Query rows using pattern matching %, _

```
SELECT c1, c2 FROM t
WHERE c1 [NOT] IN value_list;
```
Query rows in a list

```
SELECT c1, c2 FROM t
WHERE  c1 BETWEEN low AND high;
```
Query rows between two values

```
SELECT c1, c2 FROM t
WHERE  c1 IS [NOT] NULL;
```
Check if values in a table is NULL or not

SQL CHEAT SHEET http://www.sqltutorial.org

MANAGING TABLES

```
CREATE TABLE t (
    id INT PRIMARY KEY,
    name VARCHAR NOT NULL,
    price INT DEFAULT 0
);
```
Create a new table with three columns

```
DROP TABLE t ;
```
Delete the table from the database

```
ALTER TABLE t ADD column;
```
Add a new column to the table

```
ALTER TABLE t DROP COLUMN c ;
```
Drop column c from the table

```
ALTER TABLE t ADD constraint;
```
Add a constraint

```
ALTER TABLE t DROP constraint;
```
Drop a constraint

```
ALTER TABLE t1 RENAME TO t2;
```
Rename a table from t1 to t2

```
ALTER TABLE t1 RENAME c1 TO c2 ;
```
Rename column c1 to c2

```
TRUNCATE TABLE t;
```
Remove all data in a table

USING SQL CONSTRAINTS

```
CREATE TABLE t(
   c1 INT, c2 INT, c3 VARCHAR,
   PRIMARY KEY (c1,c2)
);
```
Set c1 and c2 as a primary key

```
CREATE TABLE t1(
   c1 INT PRIMARY KEY,
   c2 INT,
   FOREIGN KEY (c2) REFERENCES t2(c2)
);
```
Set c2 column as a foreign key

```
CREATE TABLE t(
   c1 INT, c1 INT,
   UNIQUE(c2,c3)
);
```
Make the values in c1 and c2 unique

```
CREATE TABLE t(
  c1 INT, c2 INT,
  CHECK(c1> 0 AND c1 >= c2)
);
```
Ensure c1 > 0 and values in c1 >= c2

```
CREATE TABLE t(
    c1 INT PRIMARY KEY,
    c2 VARCHAR NOT NULL
);
```
Set values in c2 column not NULL

MODIFYING DATA

```
INSERT INTO t(column_list)
VALUES(value_list);
```
Insert one row into a table

```
INSERT INTO t(column_list)
VALUES (value_list),
       (value_list), ....;
```
Insert multiple rows into a table

```
INSERT INTO t1(column_list)
SELECT column_list
FROM t2;
```
Insert rows from t2 into t1

```
UPDATE t
SET c1 = new_value;
```
Update new value in the column c1 for all rows

```
UPDATE t
SET c1 = new_value,
    c2 = new_value
WHERE condition;
```
Update values in the column c1, c2 that match the condition

```
DELETE FROM t;
```
Delete all data in a table

```
DELETE FROM t
WHERE condition;
```
Delete subset of rows in a table

SQL CHEAT SHEET http://www.sqltutorial.org

MANAGING VIEWS

```
CREATE VIEW v(c1,c2)
AS
SELECT c1, c2
FROM t;
```
Create a new view that consists of c1 and c2

```
CREATE VIEW v(c1,c2)
AS
SELECT c1, c2
FROM t;
WITH [CASCADED | LOCAL] CHECK OPTION;
```
Create a new view with check option

```
CREATE RECURSIVE VIEW v
AS
select-statement -- anchor part
UNION [ALL]
select-statement; -- recursive part
```
Create a recursive view

```
CREATE TEMPORARY VIEW v
AS
SELECT c1, c2
FROM t;
```
Create a temporary view

```
DROP VIEW view_name;
```
Delete a view

MANAGING INDEXES

```
CREATE INDEX idx_name
ON t(c1,c2);
```
Create an index on c1 and c2 of the table t

```
CREATE UNIQUE INDEX idx_name
ON t(c3,c4);
```
Create a unique index on c3, c4 of the table t

```
DROP INDEX idx_name;
```
Drop an index

SQL AGGREGATE FUNCTIONS

AVG returns the average of a list

COUNT returns the number of elements of a list

SUM returns the total of a list

MAX returns the maximum value in a list

MIN returns the minimum value in a list

MANAGING TRIGGERS

```
CREATE OR MODIFY TRIGGER trigger_name
WHEN EVENT
ON table_name TRIGGER_TYPE
EXECUTE stored_procedure;
```
Create or modify a trigger

WHEN
- **BEFORE** – invoke before the event occurs
- **AFTER** – invoke after the event occurs

EVENT
- **INSERT** – invoke for INSERT
- **UPDATE** – invoke for UPDATE
- **DELETE** – invoke for DELETE

TRIGGER_TYPE
- **FOR EACH ROW**
- **FOR EACH STATEMENT**

```
CREATE TRIGGER before_insert_person
BEFORE INSERT
ON person FOR EACH ROW
EXECUTE stored_procedure;
```
Create a trigger invoked before a new row is inserted into the person table

```
DROP TRIGGER trigger_name;
```
Delete a specific trigger