Informatique - ECG 2 - mathématiques appliquées 10 au 21 novembre 2025

Bases de données - TP 1 (2 séances)

Corrigé Code de partage avec Capytale : a192-7980124

Voir en annexe pour les commandes initiales permettant de créer la base de données (création et remplissage des tables).

Travail préparatoire

Sur une feuille, représenter le schéma relationnel de cette base de données (on indiquera les tables, les noms des différents attributs, on identifiera les clés primaires et étrangères et on mettra en évidence les liens entre les tables).

Le schéma relationnel de cette base de données peut être donné comme suit (on peut aussi faire un schéma pour représenter) :
Les différentes tables sont (en souligné, les clés primaires et avec un #, les clés étrangères) :
Clients(Id, Nom, Prenom, Adresse, CodePoste, Ville, Tel)
Livreurs(Id, Nom)
Pizzas(Id, Nom, PU)
Commandes(Num, Date, Heure, #IdClient, #IdLivreur)
Preparer(#IdCmde, #IdPizza, Qte)

Enfin, on précise les liens (relations) :

- dans la table `Commandes`, `IdClient` pointe vers la clé primaire de la table `Clients` et `IdLivreur` pointe vers la la clé primaire de la table `Livreurs` ;
- dans la table `Preparer`, `IdCmde` pointe vers la clé primaire de la table `Commandes` et `IdPizza` pointe vers la la clé primaire de la table `Pizzas`.

Exercice 1- compléter la base de données

Nous allons créer des requêtes pour ajouter des informations relatives à une nouvelle commande :

Nouveau client :

▷ Nathan DUCHEMIN
▷ 2 square de la Raquette 49500 Nyoiseau 02 44 27 13 12

Nouvelle commande passée le 20 novembre à 20h30

▷ 2 Pizzas Régina
▷ 1 Pizza Calzonne

La commande sera livrée par Tudal

1. Pour déterminer la valeur maximale pour l'`Id` de la table `Clients`, on exécute :

   ```
   SELECT MAX(Id) FROM Clients
   ```

2. Pour ajouter un client dans la table `Clients`, on exécute :

   ```
   INSERT INTO Clients ( ‘Id‘, ‘Nom‘, ‘Prenom‘, ‘Adresse‘, ‘CodePoste‘, ‘Ville‘, ‘Tel‘)
   VALUES ( (SELECT MAX(Id)+1 FROM Clients), "DUCHEMIN", "Nathan", "2 square de la
   Raquette", "49500", "Nyoiseau", "02 44 27 13 12")
   ```

3. Sur le même modèle, ajouter la commande dans la table `Commandes`

   On adapte les commandes précédentes, tout d'abord pour connaitre l'identifiant de la nouvelle commande, on trouve le maximum des identifiants de commandes (c'est ici l'attribut `Num`)

`SELECT MAX(Num) FROM Commandes`

On trouve alors 78 et on crée donc la ligne d'identifiant (`Num`) 78 et avec les informations correspondantes : date, heure, l'identifiant du client (nouvellement créé à la question précédente) est 6, et celui du livreur (Tudal) est 2 :

```
INSERT INTO Commandes ( `Num`, `Date`, `Heure`, `IdClient`, `IdLivreur`) VALUES
( 78, "2023-11-20", "20:30", 6,2)
```

On peut vérifier le résultat de notre requête avec `SELECT * FROM Commandes`

4. Sur le même modèle, ajouter la commande dans la table `Preparer`

   Ici pas besoin de connaitre le maximum, car nous n'insérons pas de clé primaire, il faut juste trouver les identifiants pour les pizzas correspondantes (0 pour la Regina et 5 pour la Calzone), puis d'exécuter :

   ```
   INSERT INTO Preparer (`IdCmde`, `IdPizza`, `Qte` ) VALUES ( (SELECT MAX(Num) FROM
   Commandes) , 0, 2), ( (SELECT MAX(Num) FROM Commandes) , 5, 1)
   ```

Exercice 2

Donner les requêtes qui permettront d'obtenir les informations recherchées :

1. afficher l'intégralité de la table `Commandes`

   C'est tout simplement `SELECT * FROM Commandes`

2. afficher tous les Id des différentes pizzas commandées dans la commande n°2

   Il faut alors chercher dans la table `Preparer` qui contient les identifiants des pizzas pour chaque commande et il suffit d'insérer la condition `IdCmde=2`

   ```
   SELECT * FROM Preparer WHERE IdCmde=2
   ```

3. à l'aide d'une jointure, afficher l'`Id`, le nom, la quantité et le prix de chaque pizza commandée dans la commande n°2

   La table `Preparer` ne contient pas les noms des pizzas, il faut donc joindre la table `Pizzas` en précisant que la jointure s'effectue sur les identifiants Preparer.IdPizza=Pizzas.Id et on précise les attributs que l'on souhaite voir apparaitre (en indiquant les noms des tables pour éviter les ambiguïtés) :

   ```
   SELECT Pizzas.Id, Pizzas.Nom, Preparer.Qte, Pizzas.PU FROM Preparer INNER JOIN
   Pizzas ON Preparer.IdPizza=Pizzas.Id WHERE IdCmde=2
   ```

   Les opérations arithmétiques usuelles pour les nombres : `+`, `-`, `*`, `/`, `MIN`, `MAX`,... sont naturelles avec SQL.

4. en déduire la requête qui permettra de calculer le montant de la commande n°2

   En utilisant la jointure de la requête précédente comme base, on va ici multiplier les colonnes de la quantité (`Preparer.Qte`) et du prix unitaire (`Pizzas.PU`)et en fait la somme : `SELECT SUM(Preparer.Qte*Pizzas.PU) FROM Preparer INNER JOIN Pizzas ON Preparer.IdPizza=Pizzas.Id WHERE IdCmde=2`

Exercice 3

1. Afficher toutes les dates de toutes les commandes

   ```
   select Date from Commandes
   ```

2. Afficher combien de pizza Regina ont été commandées entre deux dates définie. On pourra utiliser dans le 'WHERE' la commande 'BETWEEN' ... 'AND' ...

   On peut d'abord tester la requête avec `BETWEEN ...` :
   `SELECT Date FROM Commandes WHERE Date BETWEEN '2023-09-01' AND '2023-09-30'`
   (attention aux guillemets : on peut mettre "..." ou '...')
   puis pour trouver l'identifiant de la pizza Regina, on peut exécuter : `select Id from Pizzas where Nom="Regina"` (on trouve 0)
   et enfin pour répondre à la question, on doit joindre la table **Preparer** et utiliser la fonction d'agrégation `SUM`. Attention il y a deux conditions ici, d'où le `AND` entre les deux et il vaut mieux mettre des parenthèses quand la condition comporte plusieurs chaines de caractères (i.e. quand il y a des espaces).

   ```
   select sum(Qte) from Commandes INNER JOIN Preparer ON Preparer.IdCmde=Commandes.Num
   where (Date between '2023-09-01' and '2023-09-30') and Preparer.IdPizza=0
   ```

   on doit trouver 27

3. Afficher toutes les pizzas commandées par Paul BAS, puis le nombre total de chacune des pizzas qu'il a commandées.
   On pourra utiliser la commande 'GROUP BY'en fin de requête qui permet de regouper selon un champ (ici `Pizzas.Nom`, à défaut on peut faire avec le IdPizza de la table `Preparer`).

   On peut décomposer la requête en plusieurs étapes pour vérifier les différentes commandes. Voilà une requête finale possible :

   ```
   SELECT Pizzas.Nom, SUM(Qte) FROM Clients
   INNER JOIN Commandes ON Commandes.IdClient = Clients.Id
   INNER JOIN Preparer ON Preparer.IdCmde = Commandes.Num
   INNER JOIN Pizzas ON Pizzas.Id = Preparer.IdPizza
   WHERE Clients.Nom LIKE "BAS" AND Clients.Prenom LIKE "Paul"
   GROUP BY Pizzas.Nom
   ```

   On remarque que plusieurs jointures successives sont possibles.

4. Afficher le nom de la pizza la plus commandée par les clients

   On utilise ici la fonction d'agrégation `MAX` après avoir sélectionné les données demandées. Pour sélectionner ces données, on comprend que l'on est obligé de faire deux requêtes imbriquées : d'abord sélectionner les données de manière analogue à la requête de la question précédente (sélectionner le total pour chaque pizzas), puis sélectionner le maximum de ce résultat. On remarque que l'on peut donner un nom à la table de résultats, ici `total`, on appelle cela un alias.

   ```
   SELECT NomPizz, MAX(total) FROM ( SELECT Nom as NomPizz, SUM(Qte) as total FROM
   Pizzas INNER JOIN Preparer ON Preparer.IdPizza = Pizzas.Id GROUP BY Nom )
   ```

5. Paul DUCHEMIN a déménagé, il faut mettre à jour sa fiche client :
   ▷ Paul DUCHEMIN
   ▷ 18 rue de la Marquetterie 49520 Noyant-La-Gravoyère

   C'est un cas d'usage typique de la fonction `UPDATE` (mettre à jour) :

   ```
   update Clients set Adresse="18 rue de la Marquetterie" , CodePoste="49520" ,
   Ville="Noyant-La-Gravoyère" where Id=3
   ```

6. Afficher le nombre de pizzas « Regina » qui ont été commandées

```
select sum(Qte) from Preparer inner join Pizzas on Pizzas.Id=Preparer.IdPizza
where Nom ="Regina"
```

7. la pizzeria a un problème d'intoxication alimentaire avec la pizza « Jaurais Pas Du ». Afficher le nom et le numéro de téléphone de tous les clients qui ont commandé une pizza « Jaurais Pas Du ».

   On a « triché » ici en utilisant l'identifiant de la pizza « Jaurais Pas Du »(sinon, il faut joindre la table `Pizzas`). On a regroupé les informations par clients à la fin car il n'est pas nécessaire de les avoir plusieurs fois.

```
select Clients.Nom, Clients.Tel from Commandes
inner join Clients on Commandes.IdClient=Clients.Id
inner join Preparer on Preparer.IdCmde=Commandes.Num
where IdPizza=4 group by Clients.Nom
```

Exercice 4

La pizzeria souhaite mettre en place une carte de fidélité avec l'acquisition d'un point par tranche de 10 euros pour chaque commande de pizza.

1. Modifier le modèle relationnel pour insérer une table `Points` qui permette de mémoriser le nombre de points pour chaque client.

   On se contente de deux attributs, un identifiant client et le nombre de points :

```
CREATE TABLE 'Points' (
'IdClient' INTEGER,
'Points' INTEGER,
FOREIGN KEY ('IdClient') REFERENCES Clients('Id'));
```

2. Afficher le montant total des commandes effectuées par chaque client au mois de novembre.

   Requête qui nécessite le recours à de nombreuses tables pour avoir les informations demandées (le client, le nombre et le type de pizzas par commande et le prix de chacune d'elle) :

```
select Clients.Id, Clients.nom, Clients.prenom, sum(Qte*PU) from Clients
inner join Commandes on Commandes.IdClient=Clients.Id - pour faire le lien
clients/commande
inner join Preparer on Preparer.IdCmde=Commandes.Num - pour connaitre le contenu
des commandes
inner join Pizzas on Pizzas.Id=Preparer.IdPizza - pour avoir le prix des pizzas
where Date BETWEEN '2023-11-01' AND '2023-11-30' group by Clients.Id
```

   Nota bene : on pourrait se passer de la table `Clients` en n'affichant que l'identifiant client (on n'aurait alors pas le prénom ni le nom). C'est ce qu'on fait ci-dessous.

3. En déduire chaque requête pour ajouter le nombre de points à chaque client pour le mois de novembre.

   Il est en fait plus simple de faire la création de table à ce moment en créant une table « en tant que » (*as*) un résultat de requête. C'est ce qu'on fait en reprenant la requête précédente (sans la table `Clients` comme évoqué ci-dessus) et en convertissant en points le total en euros (divisé par 10 puis on ne garde que la partie entière) :

```
create table Points as
select Commandes.IdClient as ClientId, floor(sum(Qte*PU)/10) as Points
from Commandes
```

```
inner join Preparer on Preparer.IdCmde=Commandes.Num - pour connaitre le contenu
des commandes
inner join Pizzas on Pizzas.Id=Preparer.IdPizza - pour avoir le prix des pizzas
where Date BETWEEN '2023-11-01' AND '2023-11-30'
group by Commandes.IdClient
```

4. Afficher le client ou la cliente qui a le plus de points.

   Il suffit juste de prendre le maximum de la colonne `Points` de la table éponyme qui vient d'être créée. On peut joindre la table `Clients` pour connaitre les nom et prénom du client en question :

   ```
   select ClientId, nom, prenom,max(Points) from Points - cela suffit
   inner join Clients on Clients.Id=Points.ClientId - pour savoir qui c'est
   ```

5. Afficher le nombres de points au total pour toute la clientèle.

   Question analogue (encore plus simple) avec une somme :

   ```
   select sum(Points) as "total de points de la clientèle pour le mois de novembre
   from Points"
   ```

6. Afficher le nombre de pizzas offertes si tous les clients et clientes utilisent leur carte de fidélité fin novembre.

   Il semble qu'il manque une information : combien de points faut-il pour avoir une pizza offerte ? Disons qu'il en faut 10 et il faut alors compter pour chaque client puis faire le total (et non diviser par10 le total précédent) :

   ```
   select sum(floor(Points/10)) as "total potentiel de pizzas offertes pour le mois
   de novembre" from Points
   ```

Annexe - création et remplissage des tables

Pour commencer, on exécute les deux séries de commandes ci-dessous, respectivement, pour créer les tables et les remplir.

On exécute les requêtes suivantes pour créer les tables.

```
CREATE TABLE `Clients` (`Id` INTEGER PRIMARY KEY NOT NULL, `Nom` TEXT, `Prenom` TEXT,
`Adresse` TEXT, `CodePoste` TEXT, `Ville` TEXT, `Tel` TEXT) ;

CREATE TABLE `Livreurs` (`Id` INTEGER PRIMARY KEY NOT NULL, `Nom` TEXT);

CREATE TABLE `Pizzas` (`Id` INTEGER PRIMARY KEY NOT NULL, `Nom` TEXT, `PU` FLOAT);

CREATE TABLE `Commandes` (`Num` INTEGER PRIMARY KEY NOT NULL, `Date` date, `Heure` time,
`IdClient` INTEGER, `IdLivreur` INTEGER, FOREIGN KEY (`IdClient`) REFERENCES Clients(`Id`)
FOREIGN KEY (`IdLivreur`) REFERENCES Livreurs(`Id`) );

CREATE TABLE `Preparer` (`IdCmde` INTEGER NOT NULL, `IdPizza` INTEGER, `Qte` INTEGER,
FOREIGN KEY (`IdCmde`) REFERENCES Commandes(`Num`), FOREIGN KEY (`IdPizza`) REFERENCES
Pizzas(`Id`));
```

On exécute les requêtes suivantes pour remplir les tables qui ont été créées.

```
INSERT INTO `Pizzas` (`Id`, `Nom`, `PU`) VALUES
( 0, "Regina", 7.50),
( 1, "Quatre fromages", 8.50),
( 2, "Margarita", 8.00),
( 3, "Arrabiata", 8.50),
( 4, "Jaurais Pas Du", 9.50),
( 5, "Calzonne", 10.00),
( 6, "Speciale chef", 15.00),
( 7, "Farsus", 12.00) ;

INSERT INTO `Livreurs` ( `Id`, `Nom` ) VALUES
( 0, "Klervi"),
( 1, "Gwendal"),
( 2, "Tudal"),
( 3, "Mona") ;

INSERT INTO `Clients` ( `Id`, `Nom`, `Prenom`, `Adresse`, `CodePoste`, `Ville`, `Tel`)
VALUES
(0, "BAS", "Paul", "2 rue des Fosses", "49475", "Setré", "02 41 25 12 37"),
(1, "HAUT", "Paul", "4 av. des Sommets", "49510", "Segre-En-Anjou-Bleu", "06 12 45 27
13"),
(2, "SALOMON", "Pierre", "4 chemin du Bois", "49630", "Tourlaville", "02 41 25 12 37"),
(3, "DUCHEMIN", "Paul", "4 av. des Sommets", "49500", "Segre-En-Anjou-Bleu", "06 12
45 27 13"),
(4, "BAS", "Mireille", "2 rue des Fosses", "49475", "Setré", "02 41 25 12 37"),
(5, "SALOMON", "Renée", "4 chemin du Bois", "49630", "Tourlaville", "06 12 45 27 13") ;

INSERT INTO Commandes (`Num`, `Date`, `Heure`, `IdClient`, `IdLivreur`) VALUES
( 0, "2023-09-05", "20:46", 2, 0 ), ( 1, "2023-09-05", "20:46", 0, 0 ),
....
- voir sur Capytale pour la suite des données
INSERT INTO Preparer (`IdCmde`, `IdPizza`, `Qte` ) VALUES
( 0, 7, 1 ), ( 1, 0, 3 ), ( 2, 7, 1 ), ( 2, 5, 1 ),
....
- voir sur Capytale pour la suite des données
```