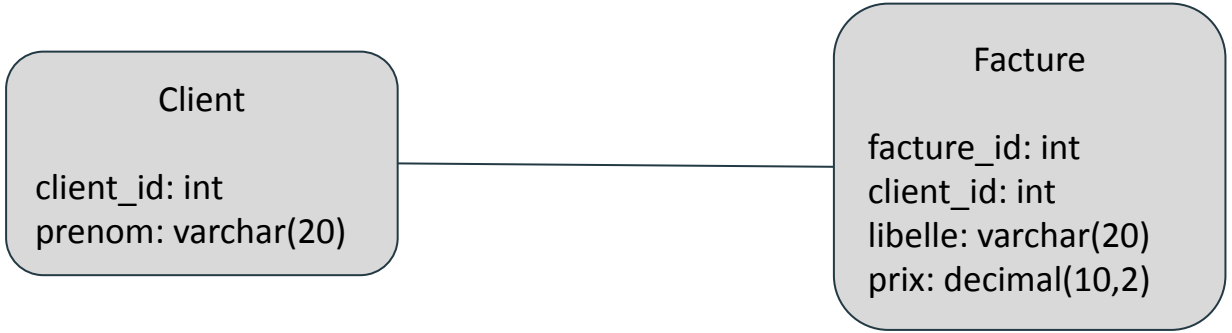
Exercices

Exercice 1.1: création de tables

Créez les tables suivantes. Puis ajoutez les contraintes correspondantes.
client_id (de Client) et facture_id sont des clés primaires générées automatiquement.
client_id (de Facture) est une clé étrangère.



Client(client_id, prenom)
Facture(facture_id, #client_id, libelle, prix)

Exercice 1.2: insertion de données

Insérez les donnés suivantes dans les tables précédentes.

Dans la table client, insérez les clients

- John.
- Jane.

Dans la table facture, insérez les factures suivantes

Pour John

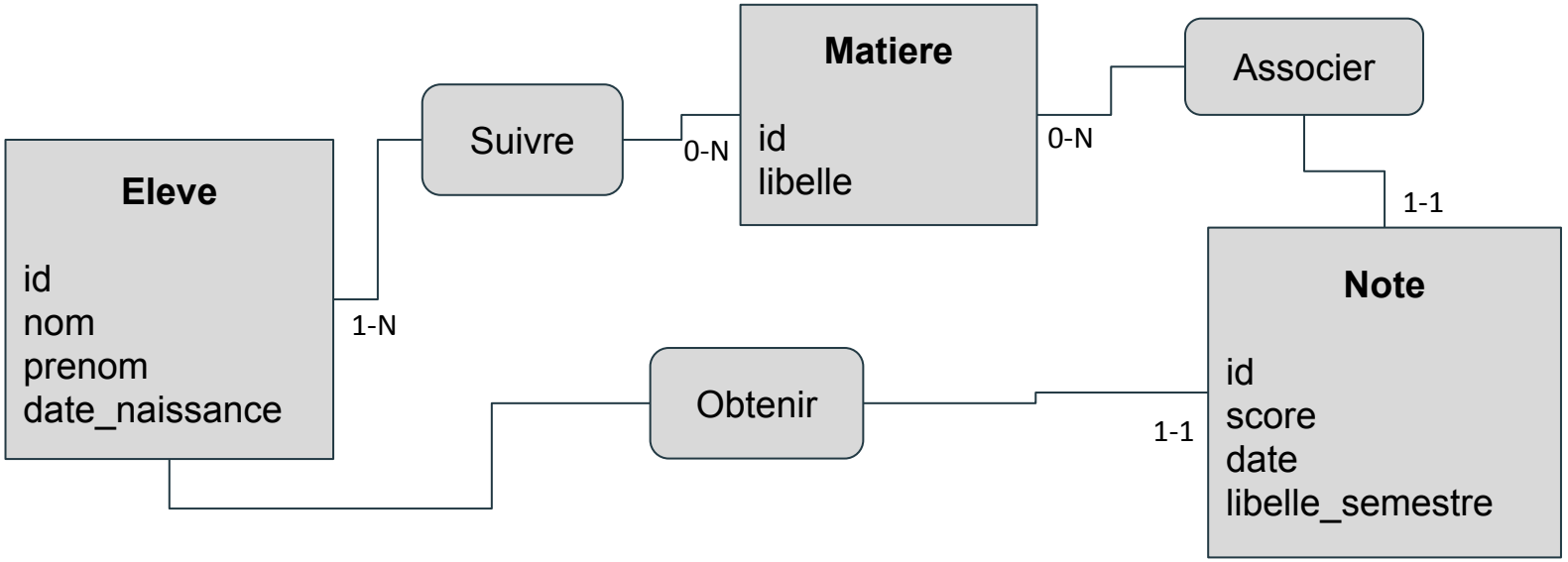
1. libellé: 'AA', prix: 12.3
2. Libélle: 'BB', prix: 5

Pour Jane

1. Libellé: 'CC', prix: 17

Exercice 2.1: création de tables

Dans cet exercice, on souhaite créer des tables pour des élèves inscrits à des matières scolaires. Chaque note enregistrée est associée à un élève et à une matière.



Exercice 2.1: création de tables

Créez les tables suivantes. Les clés étrangères sont préfixées par #

Eleve

id: identifiant
nom: varchar(20)
prenom: varchar(20)
date_naissance: date

Matiere

id: identifiant
libelle: varchar(20)

Note

id: identifiant
score: decimal(4,2)
date: date
libelle_semestre: varchar(20)
#id_matiere: int
#id_eleve: int

Une table `inscription` est également nécessaire pour associer les élèves aux matières.

inscription

#id_matiere: int
#id_eleve: int

Exercices 2.2: Insertion de données

Reprendre les tables de l'exercice 2.1 et insérer les données suivantes.

Eléve 1
Nom: Pourquier
prenom: Gaston
date: 22/02/1943

Eléve 2
nom: Potter
prenom: Harry
date: 15/10/1991

Matière 1
libellé: Français

Matière 2
libellé: Magie

Matière 3
libellé: Math

Matière 4
libellé: Histoire

Insérer les informations suivantes

Harry Potter suit les cours Magie et Histoire.
Gaston Pourquier suit les cours Français, Histoire et Math.

Harry Potter a obtenu 18 en Magie et 10 en Histoire.
Gaston Pourquier a obtenu 15 en Français, 17 en Histoire et 16 en Math.
Utilisez le libellé 'Terminale Sem1' et la date du jour pour toutes les notes.

Exercice 2.3: Requêtes SQL (Data Query Language)

En reprenant les tables de l'exercice 3, réalisez les requêtes SQL suivantes

1 - Quelle est la date de naissance de Gaston Pourquier ?

2 - Récupérer la note de Gaston Pourquier en Histoire.

3.1- Récupérer la meilleurs note obtenue en histoire.
3.2- Qui a obtenu cette note ?

4 - Quelle est la différence d'âge (en années) entre le plus jeune et le plus âgé des élèves ?

5 - Donner les moyennes des notes pour chaque élève. Ordonner les résultats par moyenne décroissante.

Exercice 3: Requêtes SQL (Data Query Language)

Récupérer les fichiers `departements.sql` et `villes_france.sql` puis exécutez les fichiers pour la création et la population des tables `ville` et des tables `departements`.

Les fichiers sont disponibles ici
https://github.com/ZorroLeVrai/MySQL.Exercices/tree/main/Data/Departements_villes

Ecrire les requêtes SQL pour répondre aux demandes suivantes

1. Obtenir la liste des 10 villes les plus peuplées en 2012
2. Obtenir la liste des 50 villes ayant la plus faible superficie
3. Obtenir la liste des départements d'outres-mer, c'est-à-dire ceux dont le numéro de département commencent par "97"
4. Obtenir le nom des 10 villes les plus peuplées en 2012, ainsi que le nom du département associé

Exercice 3: Requêtes SQL (Data Query Language)

5. Obtenir la liste du nom de chaque département, associé à son code et du nombre de commune au sein de ces départements, en triant afin d'obtenir en priorité les départements qui possèdent le plus de communes.
6. Obtenir la liste des 10 plus grands départements, en terme de superficie
7. Compter le nombre de villes dont le nom commence par "Saint"
8. Obtenir la liste des villes qui ont un nom existants plusieurs fois, et trier afin d'obtenir en premier celles dont le nom est le plus souvent utilisé par plusieurs communes
9. Obtenir en une seule requête SQL la liste des villes dont la superficie est supérieur à la superficie moyenne.
10. Obtenir la liste des départements qui possèdent plus de 2 millions d'habitants
11. Remplacez les tirets par un espace vide, pour toutes les villes commençant par "SAINT-" (dans la colonne qui contient les noms en majuscule)

Exercice 4: Requêtes SQL (Data Query Language)

Créer les tables `titanic_passenger` et `titanic_survival` en exécutant le fichier titanic/create_tables.sql
Les tables seront peuplés à partir des fichiers `passengers.csv` et `survival.csv`

Créez des requêtes SQL afin de répondre aux questions suivantes

1. Combien y a t-il de passagers dans notre base de données ?
2. Quelle est la moyenne du prix du trajet ?
3. Quel passager a payé le billet le plus cher ?
4. Quelles personnes n'ont pas payé leurs billets de 1ère classe ?
5. Quelle personne a obtenu le billet de première classe le moins cher ?
6. Quelle est le passager le plus jeune ? et le passager le plus âgé ?
7. Quel est le prix moyen des billets de 1ère, 2ème ou 3ème classe ?
8. Quel est le pourcentage de femmes dans la liste des passagers ?

Exercice 4: Requêtes SQL (Data Query Language)

9. Quel est le pourcentage de passagers par tranche d'âge.
   On se propose de prendre en considération les tranches d'âge suivantes
   a. 0 à 10 ans
   b. 10 à 25 ans
   c. 25 à 60 ans
   d. Plus de 60 ans
10. Quelle est la proportion des survivants ?
11. Quelle est la proportion des survivants par classe de transport ?
12. Quelle est la proportion des survivants par tranche d'âge ?
13. Quelle est la proportion des survivants par sexe ?

Exercices 5: Requêtes SQL (DQL, DDL, DML)

Pour cet exercice, vous pouvez récupérer les données `Notes`.
Pour créer les différentes tables, exécutez les commandes du fichier `Creation_tables.sql`.
Puis remplissez les tables à l'aide des fichiers CSV.

Réalisez les tâches ci-dessous

1. Créez une vue `veleve_niveau` qui associe la table `eleve` et la table `niveau` afin d'avoir les informations de l'élève et le niveau d'étude dans une seule vue.
2. Récupérez toutes les notes de Léa Benoit.
3. Ajoutez une note à Léa Benoit supérieur à 20 en Allemand (par exemple 25).
4. Ecrivez une requête pour supprimer la note précédemment ajoutée. Avant de supprimer la note on se propose de regarder le plan d'exécution de cette suppression. Quelles opération coûteuses est effectuée lors de la suppression ?
   Supprimer la note de 25.
5. Dans la requête n°3 nous avons vu que l'on pouvait entrer des notes supérieures à 20. Ajoutez une contrainte à la colonne `note` pour qu'elle soit toujours entre 0 et 20.

Exercices 5: Requêtes SQL (DQL, DDL, DML)

6. En écrivant la requête n°3 nous avons vu qu'il fallait gérer l'id de la clé primaire manuellement ce qui n'est pas idéal. Modifiez la table `notes` pour que le champ `id` soit généré automatiquement.
7. Actuellement un élève peut avoir plusieurs notes pour une même matière.
   Faire en sorte qu'un élève ne puisse avoir qu'une seule note par matière.
8. On se propose d'améliorer les performances de la requête n°2. Car les `Seq Scan` des tables `eleves` et `notes` peuvent être coûteux.
   Ajoutez les indexes nécessaires pour éviter ces `Seq Scan`.
9. Vérifiez maintenant le plan d'exécution de la requête n°4. Que faut-il en déduire ?