Sujet BAC 3 : Langage SQL

1. Métropole J1 : Base de données cinématographique

Métropole J1 : Base de données cinématographique

- 3 relations dans une base de données sur le cinéma
- 2 tables : **individu** et **realisation**

On pourra utiliser les mots clés SQL suivants : **SELECT**, **FROM**, **WHERE**, **JOIN**, **ON**, **INSERT**, **INTO**, **VALUES**, **UPDATE**, **SET**, **AND**.

Nous allons étudier une base de données traitant du cinéma dont voici le schéma relationnel qui comporte 3 relations :

- la relation **individu** (id_ind, nom, prenom, naissance)
- la relation **realisation** (id_rea, titre, annee, type)
- la relation **emploi** (id_emp, description, #id_ind, #id_rea)

Les clés primaires sont soulignées et les clés étrangères sont précédées d'un #.
Ainsi **emploi**.id_ind est une clé étrangère faisant référence à **individu**.id_ind.

Voici un extrait des tables **individu** et **realisation** :

Extrait de individu

id_ind	nom	prenom	naissance
105	'Hulka'	'Daniel'	'01-06-1968'
403	'Travis'	'Daniel'	'10-03-1968'
688	'Crog'	'Daniel'	'07-07-1968'
695	'Pollock'	'Daniel'	'24-08-1968'

Extrait de realisation

id_rea	titre	annee	type
105	'Casino Imperial'	2006	'action'
325	'Ciel tombant'	2012	'action'
655	'Fantôme'	2015	'action'
950	'Mourir pour attendre'	2021	'action'

1. On s'intéresse ici à la récupération de données dans une relation.

1.a. Décrire ce que renvoie la requête ci-dessous :

🗂️ **Requête SQL**

```sql
SELECT nom, prenom, naissance
FROM individu
WHERE nom = 'Crog';
```

Réponse

La requête renvoie les nom, prénom et date de naissance de tous les individus qui portent Crog comme nom de famille. Dans la mesure où l'on ne fournit que des extraits des tables, on ne peut pas fournir le résultat de cette requête de façon certaine.

1.b. Fournir une requête SQL permettant de récupérer le titre et la clé primaire de chaque film dont la date de sortie est strictement supérieure à 2020.

Réponse

🗂️ **Requête SQL**

```sql
SELECT titre, id_rea
FROM realisation
WHERE annee > 2020;
```

2. Cette question traite de la modification de relations.

2.a. Dire s'il faut utiliser la requête 1 ou la requête 2 proposées ci-dessous pour modifier la date de naissance de Daniel Crog. Justifier votre réponse en expliquant pourquoi la requête refusée ne pourra pas fonctionner.

🗂️ **Requête SQL 1**

```sql
UPDATE individu
SET naissance = '02-03-1968'
WHERE id_ind = 688 AND nom = 'Crog' AND prenom = 'Daniel';
```

🗂️ **Requête SQL 2**

```sql
INSERT INTO individu
VALUES (688, 'Crog', 'Daniel', '02-03-1968');
```

Réponse

Compte tenu de l'extrait fourni de la table `individu`, l'identifiant **`688`** est déjà utilisé pour un enregistrement et il ne peut pas y avoir de doublon pour les clés primaires, ainsi **la requête 2 provoquera une erreur**.

La requête 1 est correcte.

Bien que valide cette requête peut être simplifiée en n'utilisant que la clé primaire de la table :

🗂️ **Requête SQL 1**

```sql
UPDATE individu
SET naissance = '02-03-1968'
WHERE id_ind = 688;
```

2.b. Expliquer si la relation **`individu`** peut accepter (ou pas) deux individus portant le même nom, le même prénom et la même date de naissance.

Réponse

Aucun des champs correspondant ne possède la contrainte **`UNIQUE`** (*hypothèse réaliste*). Les deux individus n'auront donc pas le même identifiant ! Ainsi, **oui**, la relation **`individu`** peut accepter deux tels individus.

3. Cette question porte sur la notion de clés étrangères.

3.a. Recopier sur votre copie les demandes ci-dessous, dans leur intégralité, et les compléter correctement pour qu'elles ajoutent dans la relation emploi les rôles de Daniel Crog en tant que James Bond dans le film nommé `'Casino Impérial'` puis dans le film `'Ciel tombant'`.

🗂️ **Requête SQL**

```
INSERT INTO emploi
VALUES (5400, 'Acteur(James Bond)', ... , ... );

INSERT INTO emploi
VALUES (5401, 'Acteur(James Bond)', ... , ...);
```

Réponse

🗂️ **Requête SQL**

```
INSERT INTO emploi
VALUES (5400, 'Acteur(James Bond)', 688, 105);

INSERT INTO emploi
VALUES (5401, 'Acteur(James Bond)', 688, 325);
```

3.b. On désire rajouter un nouvel emploi de Daniel Crog en tant que James Bond dans le film `'Docteur Yes'`.
Expliquer si l'on doit d'abord créer l'enregistrement du film dans la relation **`realisation`** ou si l'on doit d'abord créer le rôle dans la relation **`emploi`**.

Réponse

Il faut d'abord créer l'enregistrement du film dans la relation **`realisation`**, car l'identifiant du film doit être connu afin d'être utilisé comme clé étrangère dans la relation **`emploi`**.

4. Cette question traite des jointures.

4.a. Recopier sur votre copie la requête SQL ci-dessous, dans son intégralité, et la compléter de façon à ce qu'elle renvoie le nom de l'acteur, le titre du film et l'année de sortie du film, à partir de tous les enregistrements de la relation **`emploi`** pour lesquels la description de l'emploi est `'Acteur(James Bond)'`.

🗂️ **Requête SQL**

```
SELECT ...
FROM emploi
JOIN individu ON ...
JOIN realisation ON ...
WHERE emploi.description = 'Acteur(James Bond)';
```

Réponse

🗂️ **Requête SQL**

```sql
SELECT nom, titre, annee
FROM emploi
JOIN individu ON emploi.id_ind = individu.id_ind
JOIN realisation ON emploi.id_rea = realisation.id_rea
WHERE emploi.description = 'Acteur(James Bond)';
```

4.b. Fournir une requête SQL permettant de trouver toutes les descriptions des emplois de Denis Johnson (Denis est son prénom et Johnson est son nom).
On veillera à n'afficher que la description des emplois et non les films associés à ces emplois.

Réponse

🗂️ **Requête SQL**

```sql
SELECT description
FROM emploi
JOIN individu ON emploi.id_ind = individu.id_ind
WHERE prenom = 'Denis' AND nom = 'Johnson';
```

D'après 2022, Métropole, J2

!!! exo D'après 2022, Métropole, J2"

📋 **Texte**

- 2 relations dans une base de données sur la musique
- 2 tables : **`morceaux`** et **`interpretes`**

On pourra utiliser les mots clés SQL suivants :
`SELECT`, `FROM`, `WHERE`, `JOIN`, `ON`, `INSERT`, `INTO`, `VALUES`, `UPDATE`, `SET`, `AND`.

La clause `ORDER BY` suivie d'un attribut permet de trier les résultats par ordre croissant de l'attribut.
L'instruction `COUNT(*)` renvoie le nombre de lignes d'une requête.

Un musicien souhaite créer une base de données relationnelle contenant ses morceaux et interprètes préférés. Pour cela il utilise le langage SQL.

Il crée une table **`morceaux`** qui contient entre autres attributs les titres des morceaux et leur année de sortie :

Table `morceaux`

id_morceau	titre	annee	id_interprete
1	Like a Rolling Stone	1965	1
2	Respect	1967	2
3	Imagine	1970	3
4	Hey Jude	1968	4
5	Smells Like Teen Spirit	1991	5
6	I Want To hold Your Hand	1963	4

Il crée la table **interpretes** qui contient les interprètes et leur pays d'origine :

Table `interpretes`

id_interprete	nom	pays
1	Bob Dylan	États-Unis
2	Aretha Franklin	États-Unis
3	John Lennon	Angleterre
4	The Beatles	Angleterre
5	Nirvana	États-Unis

`id_morceau` de la table **morceaux** et `id_interprete` de la table **interpretes** sont des clés primaires.

L'attribut `id_interprete` de la table **morceaux** fait directement référence à la clé primaire de la table **interpretes**.

1.a. Écrire le résultat de la requête suivante :

🗂️ **Requête SQL**

```sql
SELECT titre
FROM morceaux
WHERE id_interprete = 4;
```

Réponse

On obtient les titres `'Hey Jude'` et `'I Want To hold Your Hand'`.

1.b. Écrire une requête permettant d'afficher les noms des interprètes originaires d'Angleterre.

Réponse

🗂️ **Requête SQL**

```sql
SELECT nom
FROM interpretes
WHERE pays = 'Angleterre';
```

1.c. Écrire le résultat de la requête suivante :

🗂️ **Requête SQL**

```sql
SELECT titre, annee
FROM morceaux
ORDER BY annee;
```

Réponse

On obtient :

titre	annee
I Want To hold Your Hand	1963
Like a Rolling Stone	1965
Respect	1967
Hey Jude	1968
Imagine	1970
Smells Like Teen Spirit	1991

1.d. Écrire une requête permettant de calculer le nombre de morceaux dans la table `morceaux`.

Réponse

🗂️ **Requête SQL**

```
SELECT COUNT(*)
FROM morceaux;
```

1.e. Écrire une requête affichant les titres des morceaux par ordre alphabétique.

Réponse

🗂️ **Requête SQL**

```
SELECT titre
FROM morceaux
ORDER BY titre;
```

2.a. Citer, en justifiant, la clé étrangère de la table `morceaux`.

Réponse

La clé étrangère est `id_interprete` qui fait référence à un attribut de la table **`interpretes`**.

2.b. Écrire un schéma relationnel des tables **`interpretes`** et **`morceaux`**.

Réponse

On propose :

- **morceaux** (id_morceau, titre, annee, #id_interprete)
- **interpretes** (id_interprete, nom, pays)

Les clés primaires sont soulignées (`id_morceau` et `id_interprete`). Dans la table `morceaux`, l'attribut `id_interprete` est précédé d'un # : c'est une clé étrangère faisant référence à l'attribut `id_interprete` de la table **`interpretes`**.

2.c. Expliquer pourquoi la requête suivante produit une erreur :

🗂️ **Requête SQL**

```
INSERT INTO interpretes
VALUES (1, 'Trust', 'France');
```

Réponse

La table contient déjà une entrée dont l'attribut `id_interprete` vaut `1`. Comme il s'agit de la clé primaire cela provoque une erreur.

3.a. Une erreur de saisie a été faite. Écrire une requête SQL permettant de changer l'année du titre « Imagine » en 1971.

Réponse

On utilise la clé primaire du morceau afin d'éviter toute méprise :

🗂️ **Requête SQL**

```sql
UPDATE morceaux
SET annee = 1971
WHERE id_morceau = 3;
```

Si l'on considère que les tables fournies représentent l'ensemble des données (le sujet est ambigu à ce titre), on peut aussi se contenter de :

🗂️ **Requête SQL**

```sql
UPDATE morceaux
SET annee = 1971
WHERE titre = 'Imagine';
```

3.b. Écrire une requête SQL permettant d'ajouter l'interprète « The Who » venant d'Angleterre à la table **`interpretes`**. On lui donnera un `id_interprete` égal à 6.

Réponse

🗂️ **Requête SQL**

```sql
INSERT INTO interpretes
VALUES (6, 'The Who', 'Angleterre');
```

3.c. Écrire une requête SQL permettant d'ajouter le titre « My Generation » de « The Who » à la table **`morceaux`**. Ce titre est sorti en 1965 et on lui donnera un `id_morceau` de 7 ainsi que l'`id_interprete` qui conviendra.

Réponse

🗂️ **Requête SQL**

```sql
INSERT INTO morceaux
VALUES (7, 'My Generation', 1965, 6);
```

4. Écrire une requête permettant de lister les titres des interprètes venant des États-Unis.

Réponse

On utilise une jointure :

Requête SQL

```
SELECT titre
FROM morceaux
JOIN interpretes ON interpretes.id_interprete = morceaux.id_interprete
WHERE interpretes.pays = 'États-Unis';
```

2. Métropole, Candidats libres, J2 2021

Exercice n°3 : Métropole, Candidats libres, J2 2021

- 2 relations dans une base de données sur un CDI
- 3 tables : **Livres** , **Emprunts** et **Eleves**

L'énoncé de cet exercice utilise les mots du langage SQL suivants :
`SELECT FROM, WHERE, JOIN ON, INSERT INTO VALUES, UPDATE, SET, DELETE, COUNT, AND, OR` .
On considère dans cet exercice une gestion simplifiée des emprunts des ouvrages d'un CDI. La base de données utilisée sera constituée de trois relations (ou tables) nommées `Eleves` , `Livres` et `Emprunts` selon le schéma relationnel suivant :

- **Livres** (isbn (CHAR 13), titre (CHAR), auteur (CHAR))
- **Emprunts** (idEmprunt (INT), #idEleve (INT), #isbn (CHAR 13), dateEmprunt (DATE), dateRetour (Date))
- **Eleves** (idEleve (INT), nom (CHAR), prenom (CHAR), classe (CHAR))

Dans ce schéma relationnel, un attribut souligné indique qu'il s'agit d'une clé primaire.

Le symbole # devant un attribut indique qu'il s'agit d'une clé étrangère. Ainsi, l'attribut `idEleve` de la relation `Emprunts` est une clé étrangère qui fait référence à la clé primaire `idEleve` de la relation `Eleves` .
De même l'attribut `isbn` de la relation `Emprunts` est une clé étrangère qui fait référence à la clé primaire `isbn` de la relationcompléter `Livres` .

1. Expliquer pourquoi le code `SQL` ci-dessous provoque une erreur.

Requête SQL

```
INSERT INTO Eleves VALUES (128, 'Dupont', 'Jean', 'T1') ;
INSERT INTO Eleves VALUES (200, 'Dupont', 'Jean', 'T1') ;
INSERT INTO Eleves VALUES (128, 'Dubois', 'Jean', 'T2') ;
```

Réponse

On insère deux entrées dans lesquelles l'attribut `idEleve` est égal à `128`. Or cet attribut est la clé primaire de la table, il ne peut pas exister en doublon.

2. Dans la définition de la relation `Emprunts`, qu'est-ce qui assure qu'on ne peut pas enregistrer un emprunt pour un élève qui n'a pas encore été inscrit dans la relation `Eleves` ?

Réponse

Il s'agit de la clé étrangère `idEleve` qui doit respecter la contrainte d'intégrité référentielle.

3. Écrire une requête `SQL` qui renvoie les titres des ouvrages de Molière détenus par le CDI.

Réponse

🗂️ **Requête SQL**

```
SELECT titre
FROM Livres
WHERE auteur = 'Molière'
```

4. Décrire le résultat renvoyé par la requête ci-dessous.

🗂️ **Requête SQL**

```
SELECT COUNT(*)
FROM Eleves
WHERE classe = 'T2' ;
```

Réponse

On compte les élèves de la table `Eleves` dont la classe est la `'T2'`.

5. Camille a emprunté le livre « *Les misérables* ». Le code ci-dessous a permis d'enregistrer cet emprunt.

🗂️ **Requête SQL**

```sql
INSERT INTO Emprunts
VALUES (640, 192, '9782070409228', '2020-09-15', NULL);
```

Camille a restitué le livre le 30 septembre 2020. Recopier et compléter la requête ci-dessous de manière à mettre à jour la date de retour dans la base de données.

🗂️ **Requête SQL**

```sql
UPDATE Emprunts SET ..... WHERE ..... ;
```

Réponse

🗂️ **Requête SQL**

```sql
UPDATE Emprunts
SET dateRetour = '2020-09-30'
WHERE idEmprunt = 640
```

6. Décrire le résultat renvoyé par la requête ci-dessous.

🗂️ **Requête SQL**

```sql
SELECT DISTINCT nom, prenom
FROM Eleves, Emprunts
WHERE Eleves.idEleve = Emprunts.idEleve
AND Eleves.classe = 'T2' ;
```

Réponse

On récupère les noms et prénoms des élèves de la classe `'T2'` qui ont déjà emprunté un livre.

7. Écrire une requête SQL qui permet de lister les noms et prénoms des élèves qui ont emprunté le livre « *Les misérables* ».

Réponse

On propose (en utilisant l'ISBN cité dans la question 5):

🗂️ **Requête SQL**

```sql
SELECT nom, prenom
FROM Eleves
JOIN Emprunts ON Eleves.idEleves = Emprunts.idEleves
WHERE Emprunts.isbn = 192
```

Sans l'ISBN :

🗂️ **Requête SQL**

```sql
SELECT nom, prenom
FROM Eleves
JOIN Emprunts ON Eleves.idEleves = Emprunts.idEleves
JOIN Livres ON Livres.isbn = Emprunts.isbn
WHERE Livres.titre = 'Les Misérables'
```