SQL

Le langage SQL (*Structured Query Language*, langage de requête structuré) permet de créer des tables en spécifiant leur nom, leurs attributs, les types de ces derniers et les contraintes associées.

Médiathèque : relation `livre`

Reprenons l'exemple de la biblithèque et créons en SQL la relation `livre` de schéma :

$$livre \quad (titre : String, editeur : String, annee : Int, \underline{isbn} : String)$$

Pour cela :

1. Écrire les instructions `CREATE TABLE` suivie du nom de la relation.
2. Ensuite entres parenthèses indiquer les attributs de la relations séparés par des virgules.
3. Pour chaque attribut on indique son nom suivi de son type.
4. Pour finir, terminer la requête par un `;`

[ ]:
```
-- Commandes pour effacer les relations
-- utiles en cas d'erreurs pour recommencer calmement
--
-- DROP TABLE IF EXISTS auteur_de;
-- DROP TABLE IF EXISTS livre;
```

[ ]:
```
CREATE TABLE livre (titre TEXT,
                    editeur TEXT,
                    annee INTEGER,
                    isbn TEXT PRIMARY KEY);
```

Pour tester la requête nous allons :

- insérer quelques entités dans la relation
- puis afficher l'intégralité des entités de la relation.

Insérer On peut insérer des entités :

1. Utiliser les instructions `INSERT INTO`
2. suivis du nom de la relation
3. suivi de l'instruction VALUES
4. suivie des attributs de l'entités **dans l'ordre du schéma** entres parenthèses.
5. Pour la dernière insertion, finir par un ;

[ ]:
```
INSERT INTO livre VALUES ('Les Aventures de Huckleberry Finn', 'Flammarion', 2020, '978-2081509511');
```

Il est possible de saisir les attributs dans un ordre différents. Pour cela, il faut indiquer après le nom de la relation les noms des attributs entres parenthèses.

Par exemple pour saisir une entité en indiquant d'abord son `isbn` suivi de l'année d'édition, on écrira *(en allant à la ligne pour plus de lisibilité)* :

[ ]:
```
INSERT INTO livre (isbn, annee, titre, editeur)
VALUES ('978-2207249123', 1999, 'Fondation et Empire', 'Editions Denoël');
```

Il est d'ailleurs possible de saisir des données partielles.

Par exemple ici :

- pour `Akira` : l'année n'est pas saisie
- pour `Les Robots` : l'éditeur n'est pas saisi.

[ ]:
```
INSERT INTO livre(titre, editeur, isbn) VALUES ('Akira', 'Glénat', '978-2723428262');
INSERT INTO livre(titre, annee, isbn) VALUES ('Les Robots', 2017, '978-2745989857');
```

Insérer On peut aussi insérer plusieurs donnéees à la fois :

1. Utiliser les instructions `INSERT INTO`
2. suivis du nom de la relation
3. suivi de l'instruction VALUES.
4. Ensuite, ajouter les entitées entres parenthèses séparées par des virgules.
5. Pour la dernière insertion, finir par un ;

```
[ ]: INSERT INTO livre VALUES
        ('Astérix chez les Pictes', 'Editions Albert René', '2013', '978-2864972662'),
        ('Les Monades urbaines', 'Robert Laffont', '2016', '978-2221197691'),
        ('Les Voyages de Gulliver', 'Primento', '2015', '978-2335008586'),
        ('Lolita', 'Penguin UK', '2012', '978-0141391601'),
        ('La Nuit des temps', 'Presses de la Cité', '2014', '978-2258116429');
```

Si tout c'est passé sans erreur, vous pouvez afficher le contenu d'une relation.

Pour cela, il faut écrire la requête qui sélectionne tous les éléments contenus dans la relation `livre` et la terminer par `;`.

```
[ ]: SELECT * FROM livre;
```

Médiathèque : relation `livre`

À toi de jouer maintenant !

Pour te faire la main, tu vas (1) créer, (2) peupler et (3) afficher la relation `auteur` ayant le schéma suivant :

$$auteur \qquad (a\_id : Int, nom : String, prenom : String)$$

```
[ ]: -- Commandes pour effacer les relations
     -- utiles en cas d'erreurs pour recommencer calmement
     --
     -- DROP TABLE IF EXISTS auteur_de;
     -- DROP TABLE IF EXISTS auteur;
```

```
[ ]: CREATE TABLE auteur (a_id INTEGER PRIMARY KEY,
                         nom TEXT,
                         prenom TEXT);
```

(2) Une fois créée, tu vas insérer dans la relation `auteur` les données suivantes :

	a_id	nom	prenom
1		'Asimov'	'Isaac'
2		'Ōtomo'	'Katsuhiro'

a_id	nom	prenom
3	‘Martelle’	‘Myriam’
4	‘Touache’	‘Sébastien’
5	‘Goscinny’	‘René’
6	‘Ferri’	‘Jean-Yves’
7	‘Uderzo’	‘Albert’
8	‘Conrad’	‘Didier’
9	‘SILVERBERG’	‘Robert’
10	‘Swift’	‘Jonathan’
11	‘Ligaran’	’ ’
12	‘Nabokov’	‘Vladimir’
13	‘BARJAVEL’	‘René’

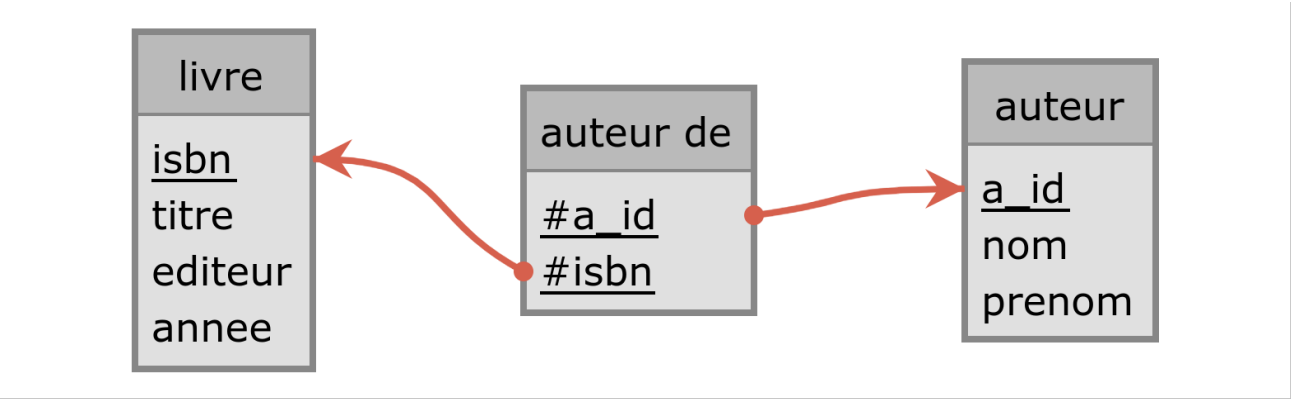
[ ]:
```
INSERT INTO auteur VALUES
    (0, 'Twain', 'Mark'),
    (1, 'Asimov', 'Isaac'),
    (2, 'Ōtomo', 'Katsuhiro'),
    (3, 'Martelle', 'Myriam'),
    (4, 'Touache', 'Sébastien'),
    (5, 'Goscinny', 'René'),
    (6, 'Ferri', 'Jean-Yves'),
    (7, 'Uderzo', 'Albert'),
    (8, 'Conrad', 'Didier'),
    (9, 'SILVERBERG', 'Robert'),
    (10, 'Swift', 'Jonathan'),
    (11, 'Ligaran', ''),
    (12, 'Nabokov', 'Vladimir'),
    (13, 'BARJAVEL', 'René');
```

(3) Enfin, tu vas afficher le contenu de la relation `auteur`.

[ ]:
```
SELECT * FROM auteur;
```

Médiathèque : relation `auteur_de`

Il va falloir maintenant créer la relation `auteur_de`.



Le schéma de la relation `auteur_de` est :

$$auteur\_de \qquad (\underline{a\_id} : Int, \underline{isbn} : String)$$

Comme on peut le voir ici, il faut mettre en place :

- une **clé primaire sur le couple (`a_id`, `isnb`)** (car l'`isbn` tout seul ou l'`a_id` tout seul ne rendent pas une entité de cette relation unique : un même isbn peut être associé à deux auteurs différents (et donc apparaître dans deux entités différentes) et un auteur peut écrire plusieurs livres (et donc peut apparaître dans de multiples entités). Ce qui est unique, c'est le couple (`a_id`, 'isbn).
- deux **contrainte de référence**.

Pour créer cette relation, on procède comme d'habitude mais après la définitions des attributs, on va ajouter trois lignes :

- `PRIMARY KEY` pour indiquer la clé primaire suivie du couple `(a_id, isbn)`
- `FOREIGN KEY(a_id)` pour indiquer que l'attribut `a_id` possède une contrainte de référence, suivi du mot-clé `REFERENCE auteur(a_id)` qui pointe vers l'attribut `a_id` de la relation `auteur`.
- `FOREIGN KEY(isbn) REFERENCE livre(isbn)` pour indiquer que l'attribut `isbn` possède une contrainte de référence en lien avec l'attribut `isbn` de la relation `livre`.

[ ]:
```
DROP TABLE IF EXISTS auteur_de;
```

[ ]:
```
CREATE TABLE auteur_de (
    a_id    INTEGER,
    isbn    TEXT,
    PRIMARY KEY (a_id, isbn),                    -- clé primaire pour le couple
    FOREIGN KEY(a_id) REFERENCES auteur(a_id),   -- clés étrangères et contraintes de référence
    FOREIGN KEY(isbn) REFERENCES livre(isbn)     --      vers les relations 'auteur' et 'livre'
);
```

Maintenant que la relation et ses contraintes sont définies, on peut insérer des données.

Par exemple `Twain Mark` (a_id : 0) est l'auteur de `Les Aventures de Huckleberry Finn` (`isbn` : 978-2081509511)

[ ]:
```
INSERT INTO auteur_de VALUES (0, '978-2081509511');
```

Pas de doublons En revanche, impossible de saisir la meme fiche en double :

[ ]:
```
INSERT INTO auteur_de VALUES(0, '978-2081509511');    -- erreur d'unicitié de la clé primaire
```

Plusieurs auteurs Mais un livre peut avoir plusieurs auteurs (car la clé primaire est le couple ! et non pas l'isbn tout seul).

Ici, c'est le livre `Astérix chez les Pictes` qui possède 4 auteurs différents :

[ ]:
```
INSERT INTO auteur_de VALUES
    (5, '978-2864972662'),
    (6, '978-2864972662'),
    (7, '978-2864972662'),
    (8, '978-2864972662');
```

Contrainte de référence Et impossible de créer une fiche si la référence n'existe pas.

Par exemple :

- Comme il n'y a pas d'auteur ayant pour `a_id` la valeur `1000`, cette insertion est impossible.

[ ]:
```
INSERT INTO auteur_de VALUES (1000, '978-2864972662')    -- erreur : contrainte de référence
```

Par exemple :

— Comme le livre d'isbn `123-1234567890` n'existe pas, l'insertion lève une erreur

[ ]:
```
INSERT INTO auteur_de VALUES (0, '123-1234567890')    -- erreur : contrainte de référence
```

On peut maintenant insérer plus d'entités dans la relation :

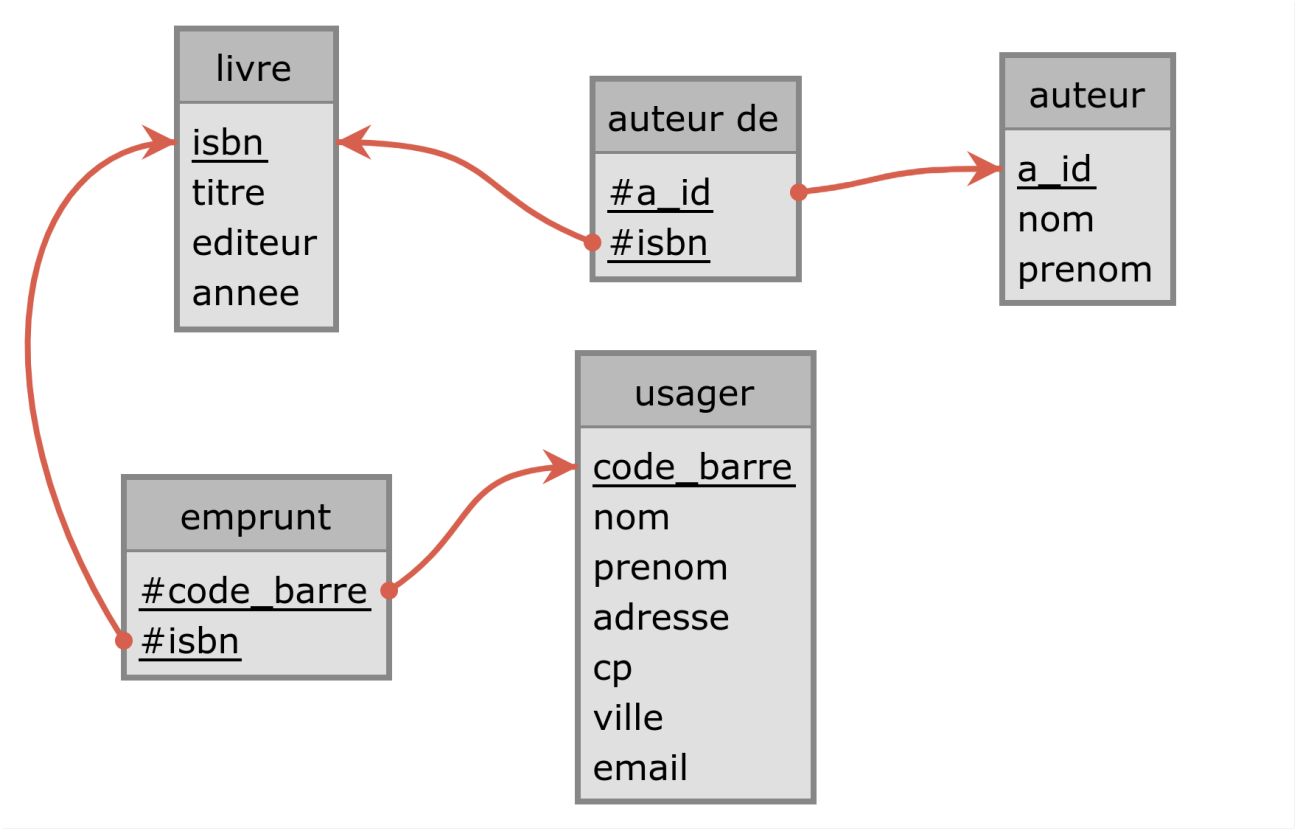
[ ]:
```
INSERT INTO auteur_de VALUES
    (1, '978-2207249123'),
    (2, '978-2723428262'),
    (3, '978-2745989857'),
    (4, '978-2745989857'),
    (9, '978-2221197691'),
    (10, '978-2335008586'),
    (11, '978-2335008586'),
    (12, '978-0141391601'),
    (13, '978-2258116429');
```

[ ]:
```
SELECT * FROM auteur_de;
```

Médiathèque : relations `usager` et `emrunt`

À toi de finir en ajoutant maintenant les relations `usager` puis `emprunt` en suivant le schéma suivant :

(tous les attributs à ajouter sont des String)


livre
isbn
titre
editeur
annee
auteur de
#a_id
#isbn
auteur
a_id
nom
prenom
usager
code_barre
nom
prenom
adresse
cp
ville
email
emprunt
#code_barre
#isbn

[ ]: