TP – CHAPITRE 5 – Base de données

Sommaire

TP1 - Exemple de relation

- On considère la relation représentée par le tableau ci-dessous :

Commandes

noCommande	date	montant	noClient
50124897	20/09/2021	217.89	6987
48531682	21/09/2021	148.58	1073
26749138	21/09/2021	87.26	2397

- Le nom de cette relation est : Commandes .
- Les attributs de cette relation sont :

  noCommand, date, montant et noClient

- Un exemple d'enregistrement est le 4 -uplet :

  (50124897, 20/09/2021, 217.89, 6987)

- Un exemple de valeur de l'attribut noClient est 6987 .

Lever la main pour valider ce TP.

TP2 - Schéma de relation

- On considère la relation représentée par le tableau ci-dessous :

Commandes

noCommande	date	montant	noClient
50124897	20/09/2021	217.89	6987
48531682	21/09/2021	148.58	1073
26749138	21/09/2021	87.26	2397

- Écrire ci-dessous le schéma de relation, on repéra la clé primaire à l'aide d'une **étoile ***, placée après le nom de l'attribut, comme dans l'exemple suivant :

table1(attribut1 : INTEGER, attribut2 : REAL, attribut3 : TEXT)*

Commandes(noCommande* : INTEGER, date : TEXT, montant : REAL, noClient : INTEGEI

Lever la main pour valider ce TP.

TP3 - Schéma de relation

- On considère la relation représentée par le tableau ci-dessous :

Clients

nom	prénom	téléphone	noClient
Dupont	Lucie	0645962315	6987
Martin	Bertrand	0656348721	1073
Mangin	Chloé	0778436918	2397

- Écrire ci-dessous le schéma de relation :

Clients(nom : TEXT, prénom : TEXT, téléphone : INTEGER, noClient* : INTEGER)

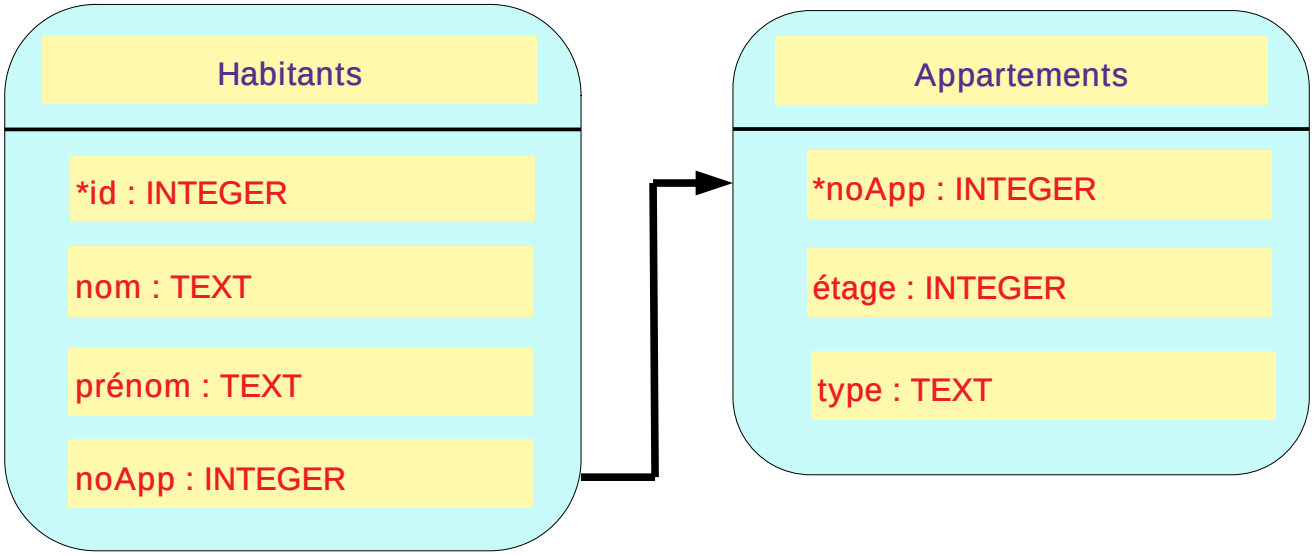
Lever la main pour valider ce TP.

TP4 - Schéma relationnel

- On considère une base de données concernant les habitants et les appartements d'un immeuble, dont le schéma relationnel est donné ci-dessous :

  Habitants(id : INTEGER, nom : TEXT, prénom : TEXT, #noApp : INTEGER)
  Appartements(noApp : INTEGER, étage : INTEGER, type : TEXT)

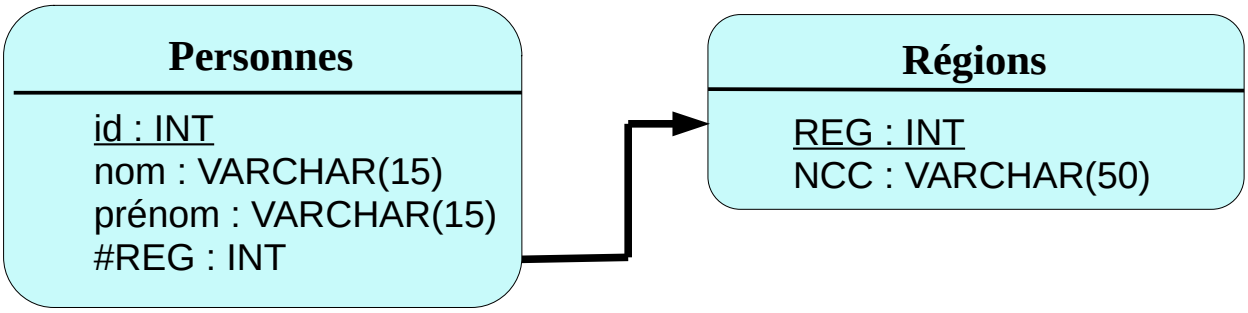
- Représenter graphiquement ce schéma relationnel ci-dessous :



Lever la main pour valider ce TP.

TP5 - Schéma relationnel

- On considère une base de données dont le schéma relationnel est représenté graphiquement ci-dessous :



- Compléter les tables suivantes, en insérant des exemples de personnes et de régions de France, REG est le code INSEE de la région.

Personnes

id	nom	prénom	REG
1	Dupont	Claude	84
2	Vivien	Louis	27
3	Martin	Julie	53

Régions

REG	NCC
84	Auvergnes-Rhônes-Alpes
27	Bourgogne-Franche-Compté
53	Bretagne

Lever la main pour valider ce TP.

TP6 - Exemple d'anomalie

- On considère la relation représentée par le tableau ci-dessous :

Commandes

noCommande	date	noProduit	produit
50124897	20/09/2021	356	galaxy S10
48531682	21/09/2021	548	iphone 7
26749138	21/09/2021	401	galaxy J3
50469187	22/09/2021	548	iphone7
669478123	23/09/2021	401	galaxy J3

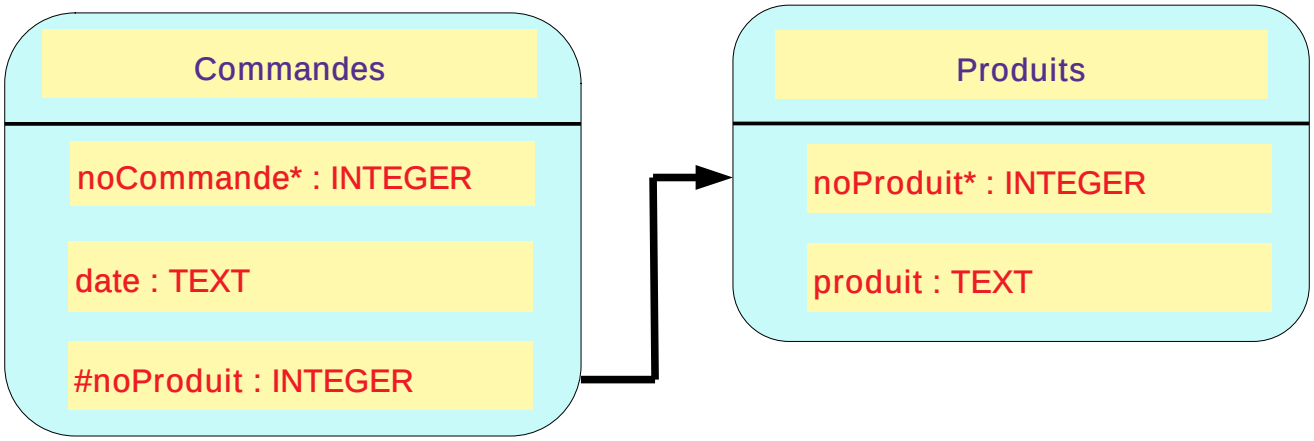
- Cette table présente des anomalies de redondances. Écrire un schéma relationnel ci-dessous pour corriger ces anomalies. Les clés primaires seront repérés par une étoile après le nom de l'attribut comme dans l'exemple suivant :

table1(attribut1 : INTEGER, attribut2 : TEXT, attribut3 : REAL)*

Commandes(noCommande* : INTEGER, date : TEXT, #noProduit : INTEGER)

Produits(noProduit* : INTEGER, produit : TEXT)

- Représenter graphiquement ce schéma relationnel :



Lever la main pour valider ce TP.

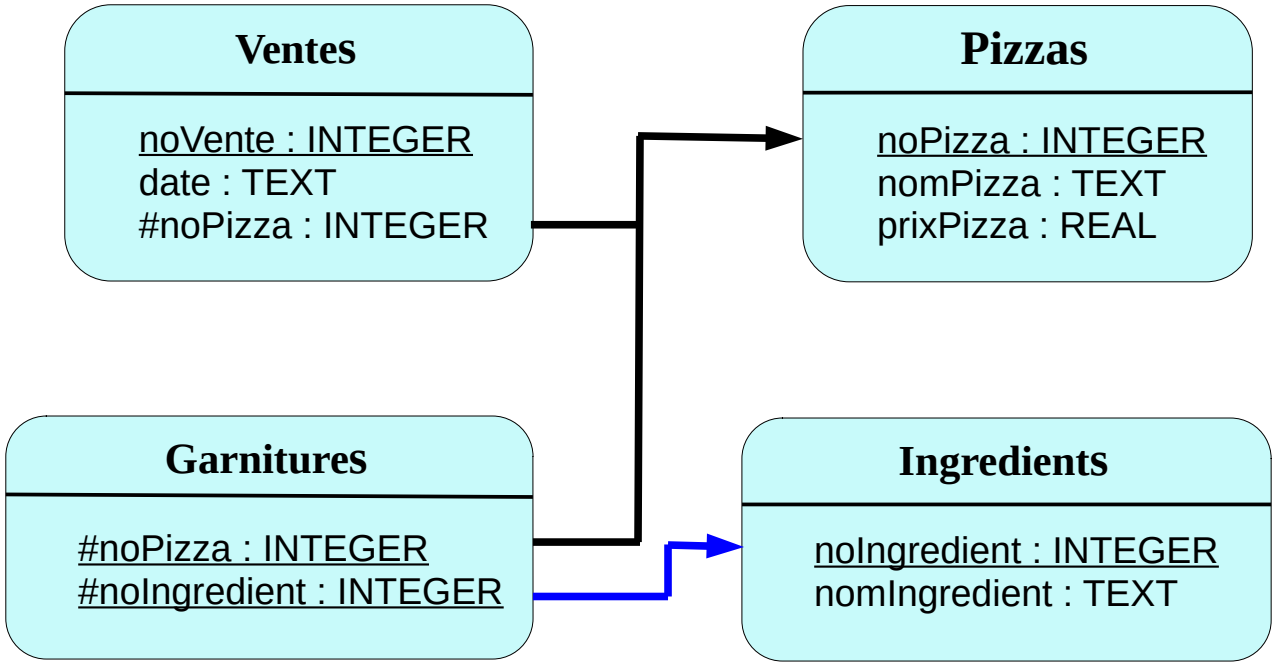
Base de données pizza

Pour les prochains TP on va utiliser une base de données nommée **pizza.db** dont le schéma relationnel est donné et représenté ci-dessous.

Schéma relationnel :

pizzas(noPizza : INTEGER, nomPizza : TEXT, prixPizza : REAL)
ventes(noVente : INTEGER, date : TEXT, #noPizza : INTEGER)
ingredients(noIngredient : INTEGER, nomIngredient : TEXT)
garnitures(#noPizza : INTEGER, #noIngredient : INTEGER)

Représentation graphique du schéma relationnel :



TP7 - Afficher tous les enregistrements d'une table

- On considère la base de données présentée précédemment : **pizza.db**.
- Lire le fichier **tuto sqlite.pdf** donné en ressource et suivre les instructions pour lancer la **console sqlite**.
- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour afficher toutes les **enregistrements** de la table **pizzas**.

```
sqlite> select * from pizzas;
```

- Le retour attendu dans la console est :

```
noPizza          nomPizza         prixPizza
---------------  ---------------  ---------------
1                Marguerite       9.0
2                Marinara         9.5
3                Napolitaine      10.5
4                4 Saisons        11.5
5                4 Fromages       12.0
6                Reine            11.0
7                Calzone          11.4
8                Chorizo          11.6
9                Montagnarde      11.5
10               Savoyarde        12.0
11               Carbonara        11.2
```

Lever la main pour valider ce TP.

TP8 - Afficher les différents prix de pizzas

- On considère toujours la base de données présentée précédemment : **pizza.db**.
- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour afficher les différents prix de pizzas.

```
sqlite> select DISTINCT prixPizza from pizzas;
```

- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour afficher les différents prix de pizzas, ordonné du plus petit eu plus grand.

```
sqlite> select DISTINCT prixPizza from pizzas ORDER BY prixPizza;
```

- Le retour attendu dans la console est :

```
prixPizza
---------------
9.0
9.5
10.5
11.0
11.2
11.4
11.5
11.6
12.0
```

Lever la main pour valider ce TP.

TP9 - Noms des pizzas dont le prix est inférieur à 10 euros

- On considère toujours la base de données présentée précédemment : **pizza.db**.
- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour afficher les noms des pizzas dont le prix est strictement inférieur à 10 euros.

```
sqlite> select nomPizza from pizzas WHERE prixPizza < 10;
```

- Le retour attendu dans la console est :

```
nomPizza
---------------
Marguerite
Marinara
```

Lever la main pour valider ce TP.

TP10 - Noms et prix des pizzas dont le prix est supérieur ou égal à 11,5 euros

- On considère toujours la base de données présentée précédemment : **pizza.db**.
- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour afficher les noms et le prix des pizzas dont le prix est supérieur ou égal à 11,5 euros.

```
sqlite> SELECT nomPizza, prixPizza FROM pizzas WHERE prixPizza >=
11.5;
```

- Le retour attendu dans la console est :

```
nomPizza         prixPizza
---------------  ---------------
4 Saisons        11.5
4 Fromages       12.0
Chorizo          11.6
Montagnarde      11.5
Savoyarde        12.0
```

Lever la main pour valider ce TP.

TP11 - Ingrédients pour les garnitures

- On considère toujours la base de données présentée précédemment : **pizza.db**.
- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour afficher les noms de tous les ingrédients utilisés pour les garnitures de pizzas.

```
sqlite> SELECT nomIngredient FROM ingredients;
```

- Le retour attendu dans la console est :

```
nomIngredient
---------------
Sauce tomate
Mozzarella
Basilic
Huile d'olive
Ail
Origan
Anchois
Jambon cuit
Artichauts
Champignons
Olives Noires
Gorgonzola
Fontina
Parmesan
oeuf
Chorizo
Poivrons
Reblochon
Lardons
jambon cru
Oignons
Crème
```

Lever la main pour valider ce TP.

TP12 - Première jointure

- On considère toujours la base de données présentée précédemment : **pizza.db**.
- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour afficher les noms de toutes les pizzas qui ont été vendues le 25/10/2021.

```
sqlite> SELECT nomPizza FROM pizzas INNER JOIN ventes ON
pizzas.noPizza=ventes.noPizza WHERE ventes.date='25/10/2021';
```

- Le retour attendu dans la console est :

```
nomPizza
---------------
Marguerite
4 Saisons
Montagnarde
Marguerite
Chorizo
Napolitaine
4 Saisons
```

Lever la main pour valider ce TP.

TP13 - Deuxième jointure

- On considère toujours la base de données présentée précédemment : **pizza.db**.
- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour afficher la date, le noms et le prix de toutes les pizzas qui ont été vendues à un prix strictement inférieur à 10 euros.

```
sqlite> SELECT ventes.date, pizzas.nomPizza, pizzas.noPizza
   ...> FROM ventes
   ...> LEFT JOIN pizzas ON ventes.noPizza = pizzas.noPizza
   ...> WHERE pizzas.prixPizza < 10;
```

- Le retour attendu dans la console est :

```
date            nomPizza        noPizza
--------------  --------------  --------------
25/10/2021      Marguerite      1
25/10/2021      Marguerite      1
26/10/2021      Marguerite      1
26/10/2021      Marinara        2
26/10/2021      Marinara        2
26/10/2021      Marguerite      1
26/10/2021      Marinara        2
28/10/2021      Marguerite      1
30/10/2021      Marguerite      1
30/10/2021      Marinara        2
30/10/2021      Marguerite      1
30/10/2021      Marguerite      1
31/10/2021      Marinara        2
31/10/2021      Marguerite      1
```

Lever la main pour valider ce TP.

TP14 - Double jointure

- On considère toujours la base de données présentée précédemment : **pizza.db**.
- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour afficher les noms de tous les ingrédients utilisés pour la garniture d'une pizza **4 Saisons**.

```
sqlite> SELECT ingredients.nomIngredient
   ...> FROM ingredients
   ...> LEFT JOIN garnitures ON garnitures.noIngredient =
ingredients.noIngredient
   ...> LEFT JOIN pizzas ON garnitures.noPizza = pizzas.noPizza
   ...> WHERE pizzas.nomPizza = '4 Saisons';
```

- Le retour attendu dans la console est :

```
nomIngredient
---------------
Sauce tomate
Mozzarella
Huile d'olive
Jambon cuit
Artichauts
Champignons
Olives Noires
```

Lever la main pour valider ce TP.

TP15 - * Double jointure

- On considère toujours la base de données présentée précédemment : **pizza.db**.
- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour afficher le nom et le prix de toutes les pizza qui ont du **Jambon cuit** dans leur garniture.

```
sqlite> SELECT pizzas.nomPizza, pizzas.noPizza
   ...> FROM pizzas
   ...> LEFT JOIN garnitures ON pizzas.noPizza = garnitures.noPizza
   ...> LEFT JOIN ingredients ON ingredients.noIngredient =
garnitures.noIngredient
   ...> WHERE nomIngredient = 'Jambon cuit';
```

- Le retour attendu dans la console est :

```
nomPizza         noPizza
---------------  ---------------
4 Saisons        4
Reine            6
Calzone          7
```

Lever la main pour valider ce TP.

TP16 - Fonction d'agrégation COUNT

- On considère toujours la base de données présentée précédemment : **pizza.db**.
- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour compter le nombre de pizzas présentes dans la relation **pizzas**.

```
sqlite> SELECT COUNT(*) FROM pizzas;
```

- Le nombre de pizzas dans la relation pizzas est de : 11 .
- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour compter le nombre de prix différents de pizzas dans la relation **pizzas**.

```
sqlite> SELECT COUNT(DISTINCT prixPizza) FROM pizzas;
```

- Le nombre de prix différents de pizzas dans la relation pizzas est de : 9 .

Lever la main pour valider ce TP.

TP17 - Autres fonctions d'agrégation

- On considère toujours la base de données présentée précédemment : **pizza.db**.
- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour obtenir le prix moyen d'une pizza dans la relation pizzas.

```
sqlite> SELECT AVG(prixPizza) FROM PIZZAS;
```

- Le prix moyen d'une pizza dans la relation pizzas est de : 11 .

- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour obtenir le prix maximum d'une pizza dans la relation **pizzas**.

```
sqlite> SELECT MAX(prixPizza) FROM pizzas;
```

- Le prix maximum d'une pizza dans la relation pizzas est de : 12 .

- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour obtenir le prix minimum d'une pizza dans la relation **pizzas**.

```
sqlite> SELECT MIN(prixPizza) FROM pizzas;
```

- Le prix minimum d'une pizza dans la relation pizzas est de : 9 .

- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour obtenir le prix minimum d'une pizza dont le prix est supérieur à 10 euros dans la relation **pizzas**.

```
sqlite> SELECT MIN(prixPizza) FROM pizzas WHERE prixPizza > 10;
```

- Le prix minimum d'une pizza dont le prix est supérieur à 10 euros dans la relation pizzas est de : 10.5 .

Lever la main pour valider ce TP.

TP18 - Insérer une nouvelle pizza

- On considère toujours la base de données présentée précédemment : **pizza.db**.
- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour insérer une **12ème pizza** dans la table **pizzas** sachant que cette pizza se nomme **Royale** et que son prix est de **12,30** euros.

```
sqlite> INSERT INTO pizzas VALUES (12, 'Royale', 12.3);
```

- Vérifier à l'aide d'une **requête sql** que la pizza **Royale** figure bien maintenant dans la table **pizzas**. Le retour attendu dans la console est suivant.

```
noPizza          nomPizza         prixPizza
---------------  ---------------  ---------------
1                Marguerite       9.0
2                Marinara         9.5
3                Napolitaine      10.5
4                4 Saisons        11.5
5                4 Fromages       12.0
6                Reine            11.0
7                Calzone          11.4
8                Chorizo          11.6
9                Montagnarde      11.5
10               Savoyarde        12.0
11               Carbonara        11.2
12               Royale           12.3
```

Lever la main pour valider ce TP.

TP19 - Insérer une nouvelle garniture

- On considère toujours la base de données présentée précédemment : **pizza.db**.
- La garniture de la pizza royale est composé des ingrédients suivant :

```
Sauce tomate
Mozzarella
Jambon cuit
Champignons
Lardons
Oignons
```

- A l'aide d'une **requête sql** compléter le tableau ci-dessous :

Ingrédients

noIngredient	nomIngredient
1	Sauce tomate
2	Mozzarella
8	Jambon cuit
10	Champignons
19	Lardons
21	Oignons

- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour insérer dans la table **garnitures** les six **2-uplets** correspondant à la garniture de la pizza **Royale**.

```
sqlite> INSERT INTO garnitures
   ...> VALUES
   ...> (12, 1),
   ...> (12, 2),
   ...> (12, 8),
   ...> (12, 10),
   ...> (12, 19),
```

- Vérifier à l'aide d'une **requête sql** que les six **2-uplets** correspondant à la garniture de la pizza **Royale** sont bien dans la table **garnitures** .

Lever la main pour valider ce TP.

TP20 - Ingrédients de la pizza Royale

- On considère toujours la base de données présentée précédemment : **pizza.db**.
- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour afficher les noms de tous les ingrédients utilisés pour la garniture d'une pizza **Royale**

```
sqlite> SELECT pizzas.nomPizza, ingredients.nomIngredient
   ...> FROM pizzas
   ...> LEFT JOIN garnitures ON garnitures.noPizza = pizzas.noPizza
   ...> LEFT JOIN ingredients ON ingredients.noIngredient =
garnitures.noIngredient
   ...> WHERE garnitures.noPizza = 12;
```

- Le retour attendu dans la console est :

```
nomPizza         nomIngredient
---------------  ---------------
Royale           Sauce tomate
Royale           Mozzarella
Royale           Jambon cuit
Royale           Champignons
Royale           Lardons
Royale           Oignons
```

Lever la main pour valider ce TP.

TP21 - Modifier le prix de la pizza Royale

- On considère toujours la base de données présentée précédemment : **pizza.db**.
- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour modifier le prix de la pizza **Royale** et le passer à **12.70** euros.

```
sqlite> UPDATE pizzas
   ...> SET prixPizza = 12.7
   ...> WHERE noPizza = 12;
```

- Écrire une **requête** sql dans la console pour vérifier que la modification a bien eu lieu. Le retour attendu dans la console est suivant.

```
noPizza           nomPizza          prixPizza
---------------   ---------------   ---------------
12                Royale            12.7
```

Lever la main pour valider ce TP.

TP22 - Insérer une nouvelle pizza Regina

- On considère toujours la base de données présentée précédemment : **pizza.db**.
- Écrire dans la console **sqlite** une **requête sql** pour insérer dans la table **pizzas** une **13ème pizza** nommée **Regina** dont le prix est **12.10** euros.
- Écrire dans la console **sqlite** une **requête sql** pour insérer les **cinq 2-uplets** dans la table **garnitures** correspondant à la garniture de la pizza Regina suivante :

  Sauce tomate
  Mozzarella
  Jambon cuit
  Champignons
  Olives noires

- Écrire dans la console **sqlite** une **requête sql** pour vérifier que la garniture de la 13ème pizza Regina a bien été insérée. Le retour attendu dans la console est suivant.

```
nomPizza         nomIngredient
---------------  ---------------
Regina           Sauce tomate
Regina           Mozzarella
Regina           Jambon cuit
Regina           Champignons
Regina           Olives noires
```

- Écrire dans la console **sqlite** une **requête sql** pour vérifier que la 13ème pizza Regina a bien été insérée dans la table pizzas. Le retour attendu dans la console est suivant.

```
noPizza          nomPizza         prixPizza
---------------  ---------------  ---------------
1                Marguerite       9.0
2                Marinara         9.5
3                Napolitaine      10.5
4                4 Saisons        11.5
5                4 Fromages       12.0
6                Reine            11.0
7                Calzone          11.4
8                Chorizo          11.6
9                Montagnarde      11.5
10               Savoyarde        12.0
11               Carbonara        11.2
12               Royale           12.7
13               Regina           12.1
```

Lever la main pour valider ce TP.

TP23 - * Modifier la garniture et le prix de la pizza Regina

- On considère toujours la base de données présentée précédemment : **pizza.db**.
- Écrire dans la console **sqlite** une **requête sql** pour modifier la garniture de la pizza **Regina** en enlevant les olives noires.
- Écrire dans la console **sqlite** une **requête sql** pour baisser le prix de la pizza **Regina** de 20 centimes.
- Écrire dans la console **sqlite** une **requête sql** pour vérifier que le prix de la pizza **Regina** a bien été modifié. Le retour attendu dans la console est suivant.

```
noPizza          nomPizza         prixPizza
---------------  ---------------  ---------------
12               Regina           11.9
```

- Écrire dans la console **sqlite** une **requête sql** pour vérifier que la garniture de la 13ème pizza Regina a bien été modifiée. Le retour attendu dans la console est suivant.

```
nomPizza         nomIngredient
---------------  ---------------
Regina           Sauce tomate
Regina           Mozzarella
Regina           Jambon cuit
Regina           Champignons
```

Lever la main pour valider ce TP.

TP24 - Supprimer la pizza Regina de la base de données

- On considère toujours la base de données présentée précédemment : **pizza.db**.
- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour supprimer la pizza **Royale** de la base de données.

```
sqlite> DELETE FROM garnitures WHERE noPizza=13;
sqlite> DELETE FROM pizzas WHERE noPizza=13;
```

- Écrire dans la console **sqlite** une **requête sql** pour vérifier que la 13ème pizza **Regina** a bien été supprimée de la base de données. Le retour attendu dans la console est suivant.

```
noPizza          nomPizza         prixPizza
---------------  ---------------  ---------------
1                Marguerite       9.0
2                Marinara         9.5
3                Napolitaine      10.5
4                4 Saisons        11.5
5                4 Fromages       12.0
6                Reine            11.0
7                Calzone          11.4
8                Chorizo          11.6
9                Montagnarde      11.5
10               Savoyarde        12.0
11               Carbonara        11.2
12               Royale           12.7
```

Lever la main pour valider ce TP.

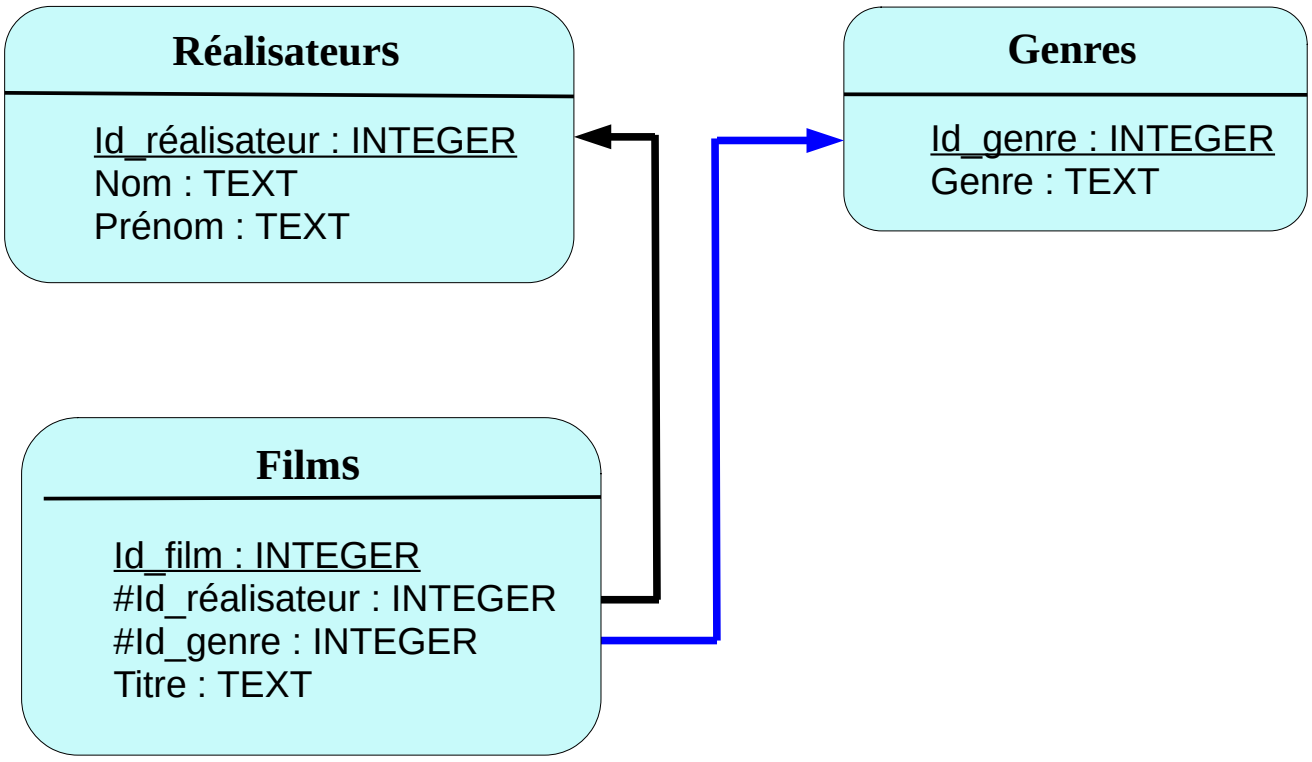
Base de données film

Pour les prochains TP on va utiliser une base de données nommée **film.db** dont le schéma relationnel est donné et représenté ci-dessous.

Schéma relationnel :

Réalisateurs(Id_réalisateur : INTEGER, Nom : TEXT, Prénom : TEXT)
Genres(Id_genre : INTEGER, Genre : TEXT)
Films(Id_film : INTEGER, #Id_réalisateur : INTEGER, #Id_genre : INTEGER, Titre : TEXT)

Représentation graphique du schéma relationnel :



TP25 - Création de la table Réalisateurs

- On va créer une nouvelle base nommée **film.db** comme présentée sur la page précédente. Pour cela, lancer **l'émulateur linux**, puis dans la console lancer **sqlite3** en créant la nouvelle base **film.db** en tapant la commande :

```
tc@boxc:~$ sqlite3 film.db
```

- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour créer la table **Réalisateurs** en respectant bien les contraintes du schéma relationnel présenté précédemment : **film.db** .

```
sqlite> CREATE TABLE Réalisateurs
   ...> (
   ...> id_réalisateur INTEGER PRIMARY KEY,
   ...> nom TEXT,
   ...> prénom TEXT
   ...> );
```

- Vérifier dans la console que le schéma de la relation **Réalisateurs** est correct. Le retour attendu dans la console est suivant.

```
sqlite> .schema Réalisateurs
CREATE TABLE Réalisateurs(
Id_réalisateur INTEGER PRIMARY KEY,
Nom TEXT,
Prénom TEXT);
```

Lever la main pour valider ce TP.

TP26 - Création de la table Genres

- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour créer la table **Genres** en respectant bien les contraintes du schéma relationnel présenté précédemment : **film.db** .

```
sqlite> CREATE TABLE Genres
   ...> (
   ...> id_genre INTEGER PRIMARY KEY,
   ...> genre TEXT
   ...> );
```

- Vérifier dans la console que le schéma de la relation **Genres** est correct, en tapant la commande :

```
sqlite> .schema Genres
```

Lever la main pour valider ce TP.

TP27 - Création de la table Films

- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour créer la table **FILMS** en respectant bien les contraintes du schéma relationnel présenté précédemment : **film.db** notamment les **deux clés étrangères**.

```
sqlite> CREATE TABLE Films(
   ...> id_film INTEGER PRIMARY KEY,
   ...> id_realisateur INTEGER,
   ...> id_genre INTEGER,
   ...> titre TEXT,
   ...> FOREIGN KEY(id_realisateur)
   ...> REFERENCES Réalisateurs(id_realisateur),
```

- Vérifier dans la console que le schéma de la relation **Genres** est correct, en tapant la commande :

```
sqlite> .schema Films
```

Lever la main pour valider ce TP.

TP28 - Création des enregistrements de la table Réalisateurs

- Écrire dans la console **sqlite** une **requête sql** pour insérer dans la table **Réalisateurs** tous les enregistrements correspondant au tableau ci-dessous.

Réalisateurs

Id_réalisateur	Nom	Prénom
1	Scorsese	Martin
2	Chaplin	Charlie
3	Hitchcock	Alfred
4	Cameron	James
5	Scott	Ridley

- Vérifier dans la console à l'aide d'une r**equête sql** que tous les enregistrements ont bien été insérés dans la table **Réalisateurs** . Le retour attendu dans la console est suivant.

```
Id_réalisateur   Nom              Prénom
--------------   ---------------  ---------------
1                Scorsese         Martin
2                Chaplin          Charlie
3                Hitchcock        Alfred
4                Cameron          James
5                Scott            Ridley
```

Lever la main pour valider ce TP.

TP29 - Création des enregistrements de la table Genres

- Écrire dans la console **sqlite** une **requête sql** pour insérer dans la table **Genres** tous les enregistrements correspondant au tableau ci-dessous.

Genres

Id_genre	Genre
1	Polar
2	Drame
3	Comédie
4	Thriller
5	Péplum
6	Science-fiction

- Vérifier dans la console à l'aide d'une **requête sql** que tous les enregistrements ont bien été insérés dans la table **Genres** . Le retour attendu dans la console est suivant.

```
Id_genre         genre
---------------  ---------------
1                Polar
2                Drame
3                Comédie
4                Thriller
5                Péplum
6                Science-fiction
```

Lever la main pour valider ce TP.

TP30 - Création des enregistrements de la table Films

- Écrire dans la console **sqlite** une **requête sql** pour insérer dans la table **Genres** tous les enregistrements correspondant au tableau ci-dessous.

Films

Id_film	Id_réalisateur	Id_genre	Titre
1	1	1	Taxi Driver
2	3	4	Psychose
3	4	2	Titanic
4	2	3	Le Dictateur
5	3	4	Les Oiseaux
6	5	5	Gladiator
7	4	6	Avatar

- Vérifier dans la console à l'aide d'une **requête sql** que tous les enregistrements ont bien été insérés dans la table **Films** . Le retour attendu dans la console est suivant.

```
Id_films        Id_réalisateur  Id_genre        Titre
--------------  --------------  --------------  ---------------
1               1               1               Taxi Driver
2               3               4               Psychose
3               4               2               Titanic
4               2               3               Le Dictateur
5               3               4               Les Oiseaux
6               5               5               Gladiator
7               4               6               Avatar
```

Lever la main pour valider ce TP.

TP31 - Requête dans la base de données film.db

- On considère la base de données que l'on vient de construire : **film.db** .
- Écrire ci-dessous et dans la console **sqlite** (on peut copier coller depuis la console) une **requête sql** pour afficher le **titre**, le **Prénom** et le **Nom** du **réalisateur** et le **genre** du film **Gladiator**.

```
sqlite> SELECT titre, prénom, nom, genre
   ...> FROM Films
   ...> JOIN Réalisateurs ON
Réalisateurs.id_réalisateur=Films.id_réalisateur
   ...> JOIN Genres ON Genres.id_genre=Films.id_genre
   ...> WHERE titre="Gladiator";
```

- Le retour attendu dans la console est :

```
Titre           Prénom          Nom             genre
--------------  --------------  --------------  --------------
Gladiator       Ridley          Scott           Péplum
```

Lever la main pour valider ce TP.

TP32 - * Nouveaux enregistrements dans la base de données film.db

- On considère toujours la base de données que l'on vient de construire : **film.db** .
- Insérer de nouveau films que vous aimez bien dans la base de données **film.db** . Il faut bien sûr insérer pour chaque film, un nouvel enregistrement dans chacune des trois tables : **Réalisateurs**, **Genres** puis **Films**.
- Écrire dans la console **sqlite** des **requêtes sql** pour vérifier que les nouveaux enregistrements se sont bien faits correctement.

Lever la main pour valider ce TP.

TP33 - * Concevoir une base de données cinema.db

- Proposer une schéma relationnel pour une base de donnée **cinema.db** qui contiendra quatre tables : **Films**, **Salles**, **Horaires** et **Séances**.
  Chacune des trois premières tables doit avoir un identifiant qui sert de clé primaire et d'autre identifiants permettant de bien les définir. La quatrième table **Séances** ne contiendra que des clés étrangères et servira à définir une séance : c'est à dire un film dans telle salle selon tel horaire. Les clés primaires seront repérées par des étoiles.

  Schéma relationnel :

  Films(id_film* : INTEGER, titre : TEXT, durée : TIME)
  Salles(id_salle* : INTEGER, nom_salle : TEXT)
  Horaires(id_horaire* : INTEGER, heure_debut : TIME)
  Séances(id_séance* : INTEGER, #id_film : INTEGER, #id_salle : INTEGER, #id_horaire : INTEGER)

- Sur une feuille de papier, dessiner une représentation graphique de ce schéma relationnel.

Lever la main pour valider ce TP.

TP34 - * Concevoir une base de données cinema.db

- En utilisant la console **sqlite**, créer le contenu d'une telle base, puis rédiger quelques **requêtes sql** pour décrire la base de donnée ainsi créée.

Lever la main pour valider ce TP.