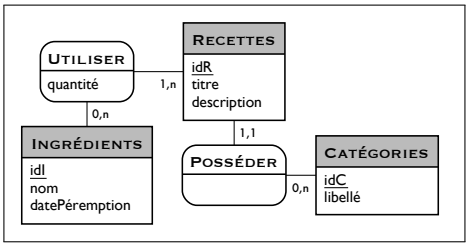
Tutoriel - création d'une base de données en SQL (PostgreSQL + `pgweb`)

Fabien Duchateau (Université Claude Bernard Lyon 1) - 2025

1. Une base de données contient un **ensemble de tables (ou relations)**. Ci-contre le schéma relationnel d'une base de données, et le diagramme entité/association correspondant. Ce schéma relationnel comporte 4 tables, que l'on va créer dans la base de données grâce au **langage SQL**.

CATÉGORIES (idC, *libellé*)
INGRÉDIENTS (idI, *nom*, *datePéremption*)
RECETTES (idR, *titre*, *description*, *#idC*)
UTILISER (*#idI*, *#idR*, *quantité*)



2. Pour interagir avec votre base de données PostgreSQL, nous allons **utiliser l'interface graphique `pgweb`**. Rendez-vous sur http://bdw.univ-lyon1.fr/ et remplissez le formulaire d'authentification pour vous connecter à votre BD.

Une fois connecté·e, vous voyez à gauche votre seul répertoire schéma (nommé *public*) . L'onglet `query` permet d'écrire et d'exécuter des requêtes SQL.

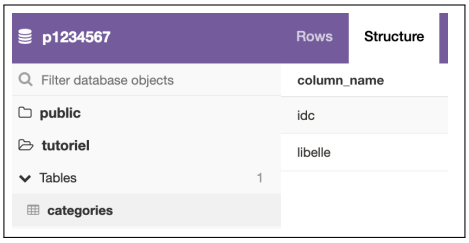
Host : `bd-pedago.univ-lyon1.fr`
Username : `[votre num. étu.]`
Password : sur Tomuss
(**pas** votre mot de passe UCBL !)
Database : `[votre num. étu.]`

3. Commençons par **créer un nouveau répertoire schéma** pour contenir nos ingrédients et recettes. Dans l'onglet `Query`, recopiez et exécutez le code SQL ci-contre.
L'instruction `DROP SCHEMA` supprime le schéma `tutoriel` (et tout son contenu) s'il existe. La suivante permet de créer le schéma `tutoriel`. Enfin, le `SET SEARCH_PATH` permet d'indiquer que l'on travaille désormais dans le répertoire schéma nommé `tutoriel`.

```
DROP SCHEMA IF EXISTS tutoriel
    CASCADE;
CREATE SCHEMA tutoriel;
SET SEARCH_PATH TO tutoriel;
```

4. C'est l'instruction `CREATE TABLE` qui permet de **créer une nouvelle table**. Pour chaque table, on définit ses attributs avec leur type, et éventuellement des contraintes. Exécutez le code de création de la table `Catégories`.
Sur `pgweb`, vous pouvez cliquer sur votre nouvelle table (à gauche) et l'onglet `Structure` décrit ses attributs.

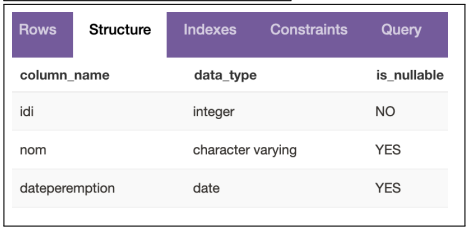
```
CREATE TABLE Categories (
  idC INTEGER NOT NULL,
  libelle VARCHAR(42),
  CONSTRAINT pk_categories PRIMARY
      KEY (idC)
);
```



5. Créons de la même manière les autres tables. Par exemple, `Recettes` contient un attribut `idR` de type `SERIAL`, c'est à dire un identifiant auto-incrémenté et utilisé comme clé primaire. Elle possède également deux attributs textuels (`titre` et `description`), ainsi qu'un entier `idC` qui permettra de référencer la catégorie de la recette à l'aide d'une clé étrangère.
Notez que les tables ne contiennent encore aucune instance/donnée.

```
CREATE TABLE Recettes (
  idR SERIAL NOT NULL,
  titre VARCHAR(200),
  description TEXT,
  idC INTEGER NOT NULL,
  PRIMARY KEY (idR)
);
```

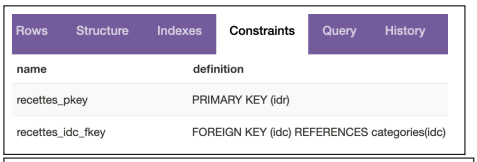
```
CREATE TABLE Ingredients (
  idI INTEGER NOT NULL,
  nom VARCHAR(200),
  datePeremption DATE
);
```



6. L'instruction `ALTER TABLE` permet de **modifier le schéma d'une table** (e.g., renommer ou changer le type d'un attribut).
Dans le code ci-contre, on modifie la table `Recettes` pour ajouter une contrainte de clé étrangère sur `idC`, qui référence désormais l'attribut `idC` de la table `Catégories`. L'instruction suivante ajoute une clé primaire dans la table `Ingrédients`. Après avoir cliqué sur une table (à gauche), on peut visualiser ces contraintes dans l'onglet `Constraints`.

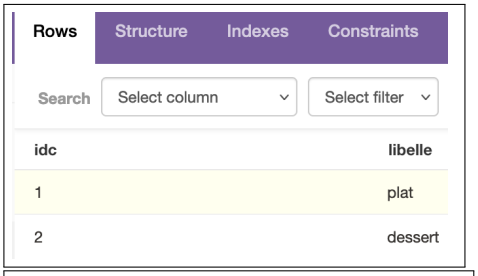
```
ALTER TABLE Recettes
ADD FOREIGN KEY (idC) REFERENCES
    Categories (idC);
```

```
ALTER TABLE Ingredients ADD PRIMARY
    KEY (idI);
```



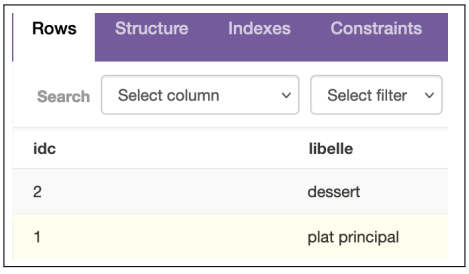
7. Avec l'instruction `INSERT`, on **crée des instances (ou tuples)** dans une table, en indiquant une valeur pour chaque attribut de la table. Pour la dernière insertion, on donne la valeur `DEFAULT` pour l'identifiant (que PostgreSQL remplace par une valeur unique générée par le `SERIAL`). Notez le double guillemet (`''`) pour mettre un guillemet simple dans la recette de cookies.
C'est l'onglet `Rows` qui permet de voir les données de la table sélectionnée.

```
INSERT INTO Categories VALUES(1,
    'plat');
INSERT INTO Categories VALUES(2,
    'dessert');
INSERT INTO Recettes
    VALUES(DEFAULT, 'Cookies
    vegan', 'cookies au sirop
    d''érable', 2);
```



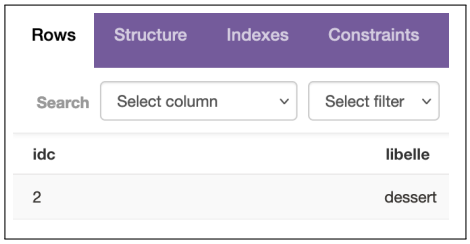
8. L'instruction `UPDATE` permet de **mettre à jour des instances**. La clause `SET` précise le ou les attributs pour lesquels la valeur sera modifiée. La clause `WHERE` permet de spécifier les instances concernées par la modification (sans un `WHERE`, toutes les instances de la table sont modifiées).
Ici, on modifie la valeur de l'attribut `libellé` en *plat principal* uniquement pour l'instance identifiée par 1.

```
UPDATE Categories
SET libelle='plat principal'
where idc=1;
```



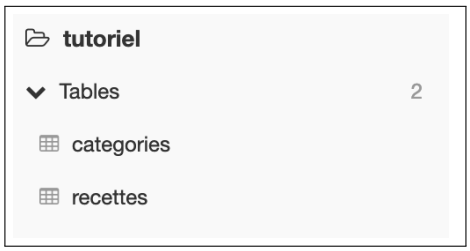
9. Pour **supprimer des instances**, on utilise l'instruction `DELETE FROM` en indiquant le nom de la table. La clause conditionnelle `WHERE` permet de sélectionner les instances à supprimer. **Attention** : sans clause `WHERE`, toutes les données de la table sont supprimées !
Dans l'exemple, on supprime la catégorie identifiée par 1 (*plat principal*).

```
DELETE FROM Categories
WHERE idC=1;
```



10. Enfin, on **supprime une table (schéma et instances)** avec l'instruction `DROP TABLE`. L'option `IF EXISTS` permet d'éviter une erreur si la table n'existe pas.
Dans l'exemple, la table `Ingrédients` est supprimée.

```
DROP TABLE IF EXISTS Ingredients;
```



Conclusion

Ces instructions SQL (pour PostgreSQL) permettent de définir le schéma d'une base de données (LDD) et de manipuler les données (LMD). Ci-dessous le code complet du tutoriel.
Ce tutoriel est volontairement limité pour en faciliter la prise en main. Des détails supplémentaires sont donnés dans les diapositives de cours (https://perso.liris.cnrs.fr/fabien.duchateau/) ou sur d'autres ressources comme sql.sh ou le livre ~~Not~~ Only SQL.

```
DROP SCHEMA IF EXISTS tutoriel CASCADE;
CREATE SCHEMA tutoriel;
SET SEARCH_PATH TO tutoriel;

CREATE TABLE Categories (
  idC INTEGER NOT NULL,
  libelle VARCHAR(42),
  CONSTRAINT pk_categories PRIMARY KEY (idC)
);

CREATE TABLE Recettes (
  idR SERIAL NOT NULL,
  titre VARCHAR(200),
  description TEXT,
  idC INTEGER NOT NULL,
  PRIMARY KEY (idR)
);

CREATE TABLE Ingredients (
  idI INTEGER NOT NULL,
  nom VARCHAR(200),
  datePeremption DATE
);

ALTER TABLE Recettes
ADD FOREIGN KEY (idC) REFERENCES Categories (idC);

ALTER TABLE Ingredients ADD PRIMARY KEY (idI);

INSERT INTO Categories VALUES(1, 'plat');
INSERT INTO Categories VALUES(2, 'dessert');
INSERT INTO Recettes VALUES(DEFAULT, 'Cookies vegan',
    'cookies au sirop d''érable', 2);
```

```
ALTER TABLE Utiliser ADD FOREIGN KEY (idR)
    REFERENCES Recettes (idR);
ALTER TABLE Utiliser ADD FOREIGN KEY (idI)
    REFERENCES Ingredients (idI);

UPDATE Categories
SET libelle='plat principal'
where idc=1;

DELETE FROM Categories
WHERE idC=1;

DROP TABLE IF EXISTS Ingredients;
```