# Projet site web R3st0.fr

# Itération n°1 - Ticket n°1 : ré-ingénierie de la BD

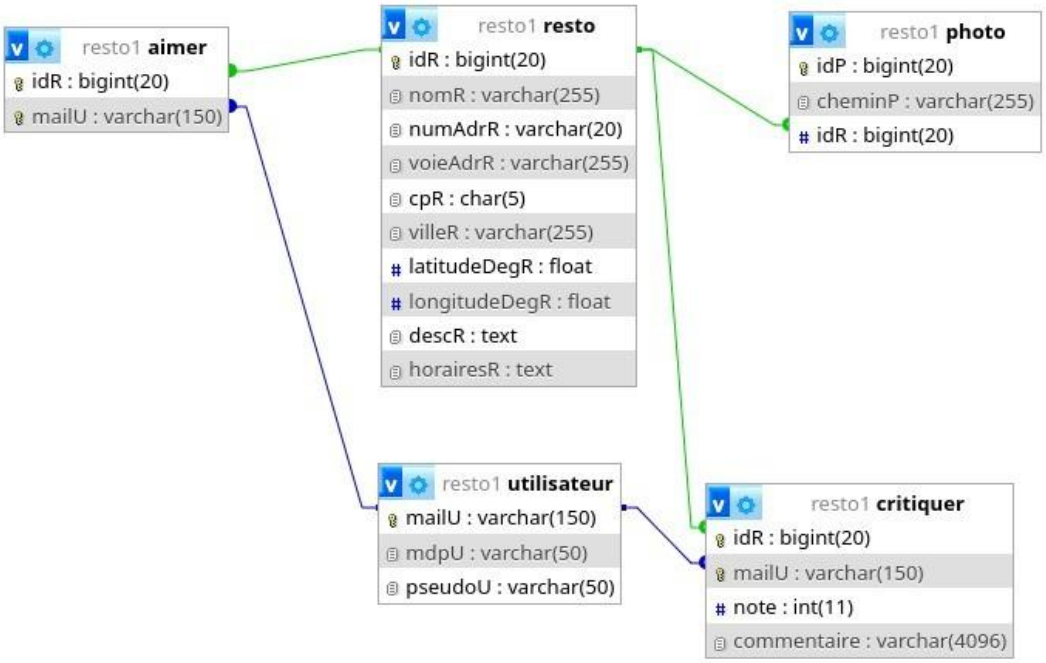
## Contexte

Une première base de données a été conçue par un précédent stagiaire.
Les scripts de création de la base de données initiale sont à télécharger depuis Moodle.

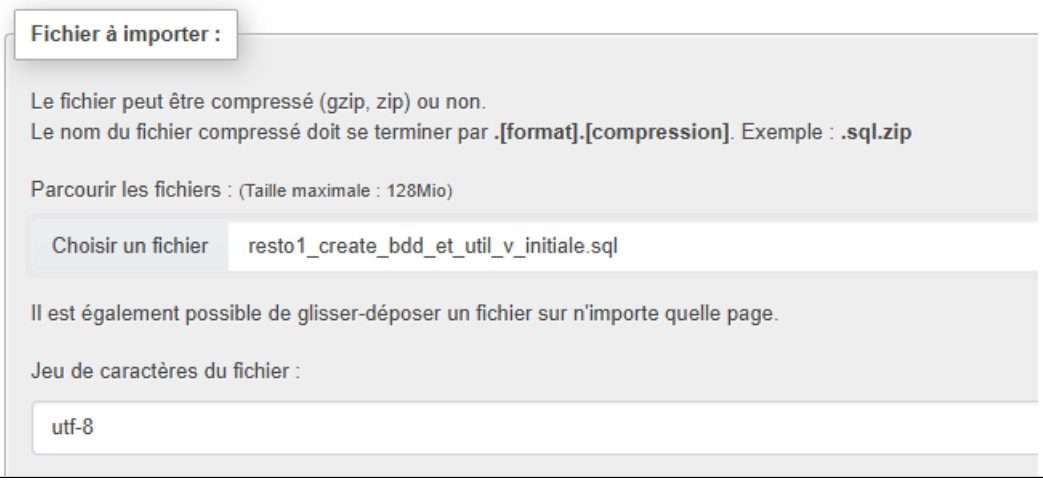
## Travail à faire

1. A l'aide des scripts SQL fournis sur Moodle, créez la base de données « resto1 » et l'utilisateur « resto_util » possédant des droits spécifiques sur cette base de données.

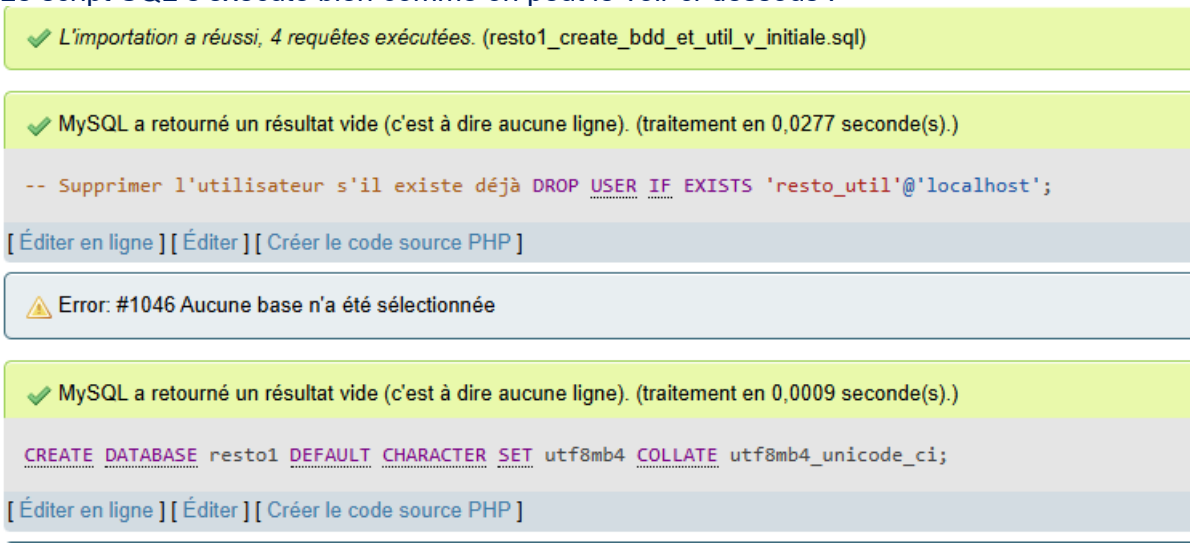
Voici le M.L.D. de la base actuelle :



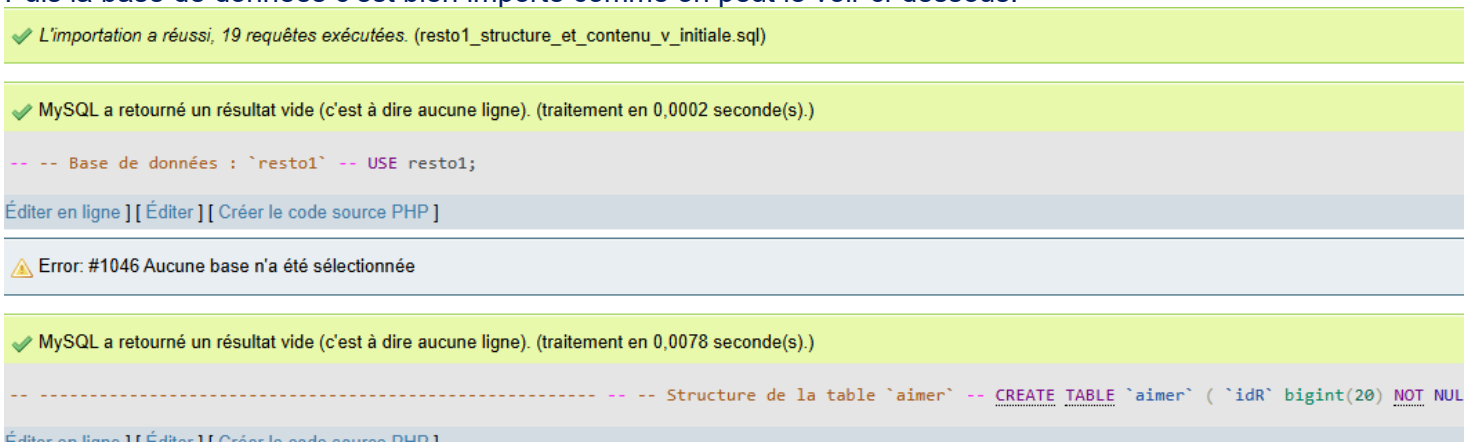

Bien importé la base de données :



Le script SQL s'exécute bien comme on peut le voir ci-dessous :




Puis la base de données c'est bien importé comme on peut le voir ci-dessous:




**On constate que la table Utilisateur a comme clef primaire l'e-mail de l'utilisateur.**

2. Formuler en SQL la requête suivante : « nom du restaurant et note attribuée lors des critique enregistrées par l'utilisateur jj.soueix@gmail.com » ; tester

   Voici la requête SQL qui permet d'afficher le nom du restaurant et la note attribuée à celui-ci lors des critiques :

```sql
SELECT resto.nomR, critiquer.note AS "Note atribuée"
FROM resto
INNER JOIN critiquer ON resto.idR = critiquer.idR;
```

   Puis on cherche pour l'adresse demandé ducoup on rajoute une restriction avec un WHERE et on cherche sur le champ demandé ducoup mail:

```sql
SELECT resto.nomR, critiquer.note AS "Note atribuée"
FROM resto
INNER JOIN critiquer ON resto.idR = critiquer.idR
INNER JOIN utilisateur ON critiquer.mailU = utilisateur.mailU
WHERE utilisateur.mailU = "jj.soueix@gmail.com";
```

Puis quand on lance la requête :


Cela affiche bien les noms des restaurants ainsi que les notes attribuées à celui-ci par cet utilisateur qui a cet adresse mail.

3. Expliquer pourquoi ce choix n'est pas judicieux ;

   Avoir comme clé primaire un e-mail n'est pas judicieux car elle peut changer, il est long à indexer donc moins performant, il peut créer des doublons à cause des variantes d'écriture et c'est une donnée sensible exposée dans toutes les jointures. Il vaut mieux utiliser un identifiant comme clé primaire

4. Mettre au point un script SQL qui modifie la base de données existante avec les actions suivantes :
   - créer une clef primaire idU plus adaptée pour la table Utilisateur et lui attribuer des valeurs ; le champ e-mail est conservé, avec une contrainte d'unicité ;
   - dans les tables où il est utilisé comme clef étrangère, remplacer le champ e-mail par un champ idU ; il faut bien entendu veiller à ce que les résultats des jointures ne soient pas altérés par cette modification.

Voici les explications pour la modification de la base de données :

Le script commence par **supprimer toutes les clés étrangères** sur **aimer** et **critiquer** pour permettre les modifications. Ensuite, il **supprime les index liés à mailU** qui empêcheraient de supprimer les clés primaires. Les c**lés primaires existantes sur les tables dépendantes** sont ensuite **supprimées**, puis la clé primaire actuelle sur utilisateur est retirée.

La **colonne idU** est ajoutée à utilisateur comme clé primaire AUTO_INCREMENT et un index UNIQUE est créé sur mailU. idU est ajouté aux tables dépendantes et rempli en fonction de mailU. Les colonnes mailU dans les tables dépendantes sont ensuite supprimées. Les nouvelles clés primaires sur (idR, idU) sont définies dans aimer et critiquer. Enfin, les clés étrangères sont recréées pour lier idU à utilisateur et idR à resto. Cela assure que toutes les relations et intégrités référentielles restent correctes après la migration.

resto1_structure_et_contenu_v_initiale.sql | modif.sql

C: > Users > anais > OneDrive > Bureau > modif.sql

Connect to MSSQL

```sql
USE resto1;

-- ========================================================
-- 1. Supprimer toutes les clés étrangères sur aimer et critiquer
-- ========================================================
ALTER TABLE aimer DROP FOREIGN KEY IF EXISTS aimer_ibfk_1;
ALTER TABLE aimer DROP FOREIGN KEY IF EXISTS aimer_ibfk_2;

ALTER TABLE critiquer DROP FOREIGN KEY IF EXISTS critiquer_ibfk_1;
ALTER TABLE critiquer DROP FOREIGN KEY IF EXISTS critiquer_ibfk_2;

-- ========================================================
-- 2. Supprimer les index liés à mailU dans les tables dépendantes
-- ========================================================
ALTER TABLE aimer DROP INDEX IF EXISTS mailU;
ALTER TABLE critiquer DROP INDEX IF EXISTS mailU;

-- ========================================================
-- 3. Supprimer les clés primaires existantes sur les tables dépendantes
-- ========================================================
ALTER TABLE aimer DROP PRIMARY KEY;
ALTER TABLE critiquer DROP PRIMARY KEY;

-- ========================================================
-- 4. Supprimer la clé primaire actuelle sur mailU dans utilisateur
-- ========================================================
ALTER TABLE utilisateur DROP PRIMARY KEY;

-- ========================================================
-- 5. Ajouter idU dans utilisateur comme clé primaire AUTO_INCREMENT
-- ========================================================
ALTER TABLE utilisateur
ADD COLUMN idU BIGINT NOT NULL AUTO_INCREMENT FIRST,
ADD PRIMARY KEY (idU),
ADD UNIQUE KEY unique_mail (mailU);

-- ========================================================
-- 6. Ajouter idU dans les tables dépendantes
-- ========================================================
ALTER TABLE aimer ADD COLUMN idU BIGINT DEFAULT NULL AFTER mailU;
ALTER TABLE critiquer ADD COLUMN idU BIGINT DEFAULT NULL AFTER mailU;

-- ========================================================
-- 7. Remplir idU à partir de mailU
-- ========================================================
UPDATE aimer a
JOIN utilisateur u ON a.mailU = u.mailU
SET a.idU = u.idU;

UPDATE critiquer c
JOIN utilisateur u ON c.mailU = u.mailU
SET c.idU = u.idU;

-- ========================================================
-- 8. Supprimer la colonne mailU maintenant que idU est renseigné
-- ========================================================
ALTER TABLE aimer DROP COLUMN mailU;
ALTER TABLE critiquer DROP COLUMN mailU;

```

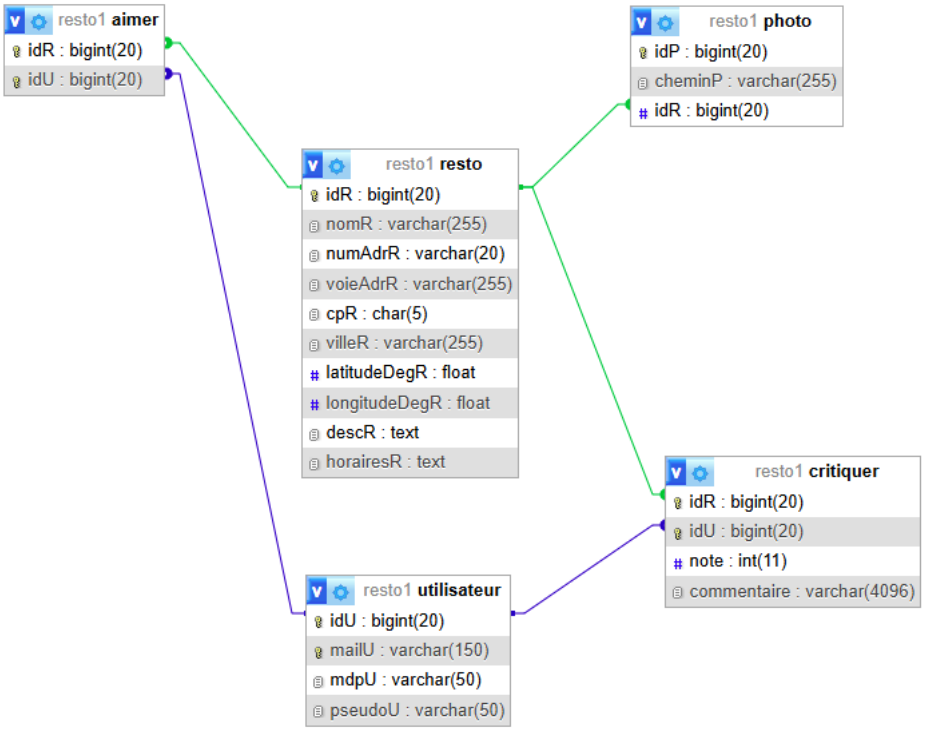
```sql
-- ============================================================
-- 8. Supprimer la colonne mailU maintenant que idU est renseigné
-- ============================================================
ALTER TABLE aimer DROP COLUMN mailU;
ALTER TABLE critiquer DROP COLUMN mailU;

-- ============================================================
-- 9. Définir les nouvelles clés primaires sur (idR, idU)
-- ============================================================
ALTER TABLE aimer ADD PRIMARY KEY (idR, idU);
ALTER TABLE critiquer ADD PRIMARY KEY (idR, idU);

-- ============================================================
-- 10. Ajouter les clés étrangères correctes
-- ============================================================
ALTER TABLE aimer
ADD CONSTRAINT aimer_ibfk_1 FOREIGN KEY (idR) REFERENCES resto(idR)
ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT aimer_ibfk_2 FOREIGN KEY (idU) REFERENCES utilisateur(idU)
ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE critiquer
ADD CONSTRAINT critiquer_ibfk_1 FOREIGN KEY (idR) REFERENCES resto(idR)
ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT critiquer_ibfk_2 FOREIGN KEY (idU) REFERENCES utilisateur(idU)
ON DELETE CASCADE ON UPDATE CASCADE;
```

Voici à quoi ressemble la base de données après la modification :



5. Afin de tester la base de données ainsi modifiée, formuler et tester une requête équivalente à celle de la question 2.

   Bien testé une requête équivalente à la question 2 et cela donne le même résultat comme on peut le voir ci-dessous :


| nomR | note |
|---|---|
| l'entrepote | 3 |
| le bar du charcutier | 2 |