SAÉ "Conception et implémentation d’une base de données"

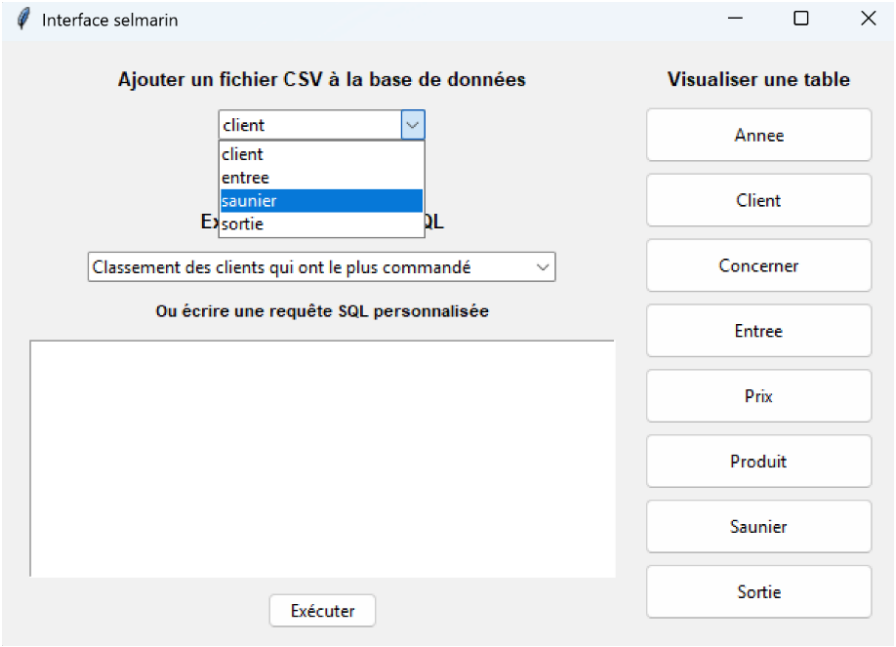
Dans le cadre de la Saé, nous avons travaillé sur le projet intitulé **"Les sauniers de l'île de Ré"**. L'objectif de ce projet était d'informatiser la gestion des approvisionnements et des ventes de la coopérative de sel marin, en concevant une base de données complète, en la remplissant de données réelles, et en créant une application graphique pour interagir avec celle-ci.

Après avoir conçu le modèle conceptuel de données (MCD) puis le schéma relationnel, nous avons créé et alimenté la base de données MySQL en utilisant des scripts SQL. En complément, nous avons développé une interface graphique en Python.

Au début du projet, nous avons travaillé en commun pour concevoir la base de données. Ensuite, la répartition des tâches s'est organisée naturellement : Nathaël s'est chargé de commencer l'interface Python, tandis que Paul s’est concentré sur l'écriture et le test des différentes requêtes SQL. Enfin, lors de la phase finale, nous avons travaillé ensemble pour corriger les bugs, ajuster les dernières fonctionnalités de l'interface, et vérifier que tout fonctionnait correctement avant la livraison du projet.

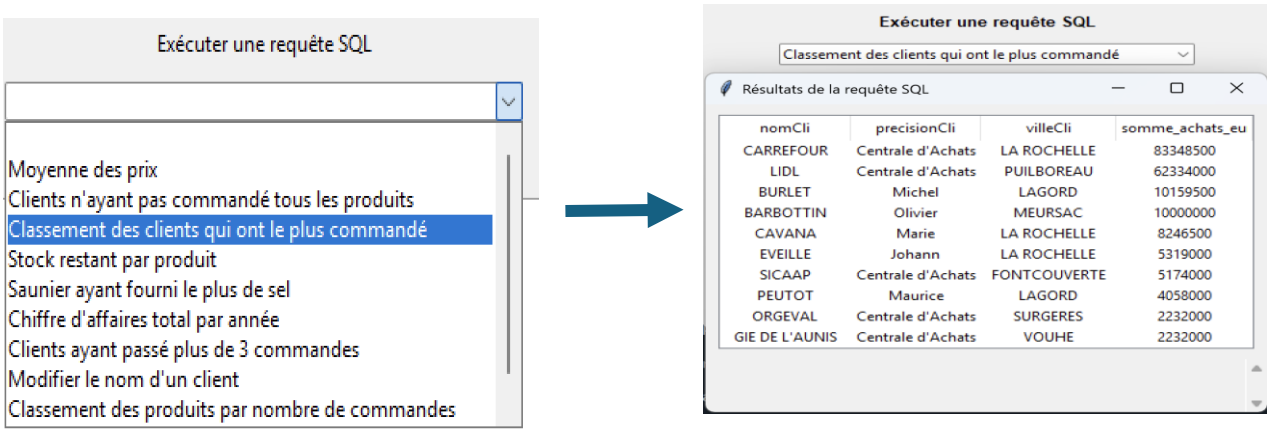
Ce document présente un bilan de la réalisation de cette application, accompagné d'une démonstration de ses fonctionnalités, d'un retour d'expérience sur ce qui fonctionne ou non, ainsi que des pistes d'amélioration envisagées.

DEMONSTRATION DE NOTRE TKINTER :

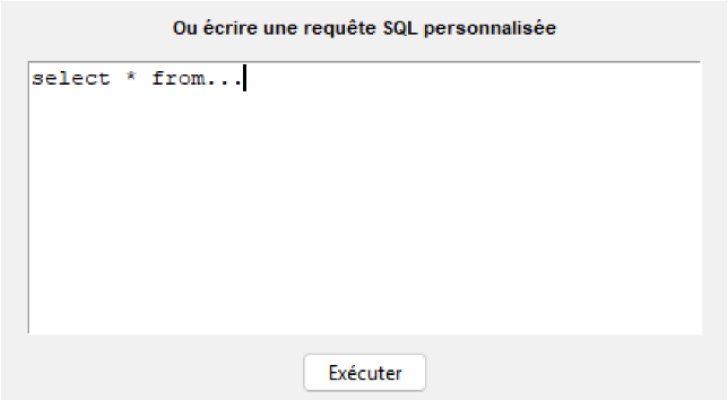


Voici notre interface tkinter qui est simple et fonctionnelle. Pour le style de l’interface, nous avons utilisé ttk (Tk themed widgets), une extension moderne de tkinter, qui permet d'obtenir des composants plus esthétiques, cohérents avec le système d'exploitation, et mieux adaptés à une interface professionnelle.

L’ajout de fichiers CSV fonctionne correctement, car la liste déroulante prend en compte les fichiers .csv présents dans le dossier. Si les fichiers ne sont pas les bons, des erreurs sont affichées car les tables ne peuvent pas être alimentées correctement.



La sélection d’une requête SQL prédéfinie fonctionne comme prévu : une fois la requête choisie dans la liste déroulante, son exécution affiche les résultats dans une nouvelle fenêtre. Les colonnes s’adaptent dynamiquement en fonction des données retournées par la requête, ce qui permet une visualisation claire et structurée des informations extraites de la base de données. Toutes les requêtes fonctionnent correctement.



Écriture manuelle d'une requête SQL : La zone de texte dédiée permet à l'utilisateur de saisir librement une requête SQL personnalisée. Une fois la requête rédigée et validée, elle est exécutée sur la base de données, et les résultats sont automatiquement affichés dans une nouvelle fenêtre. Nous avons testé la zone avec des requêtes fausses et cela affiche les erreurs correspondantes sans faire planter le programme. Les requêtes qui commencent par autre chose que ‘select’ sont programmées pour ne pas renvoyer de résultats hormis un message d’exécution. Nous avons fait en sorte dans notre programme que DROP database sur selmarin_final soit impossible. Pour les autres ‘drop database’, la base correspondante est bien supprimée.

A droite on a une visualisation possible d'une table complète via des boutons : chaque table peut être visualisée dynamiquement, ce qui permet de vérifier l'état de la base de données à tout moment.

Nous avons essayé d'intégrer la création de la base de données à partir de l'interface python mais nous n'avons pas réussi. La création de nouvelles bases de données est possible mais seulement si selmarin_final existe déjà. Notre programme est basé sur l'existence de selmarin_final, par conséquent, on peut créer et supprimer d'autres bases mais l'interaction avec ces dernières est impossible.

Nous aurions aimé pouvoir gérer plusieurs bases de données à partir de l'ihm sans être dépendant de selmarin_final.

Ce que nous avons le plus apprécié lors de ce projet est la création de la base en entier à partir de rien, ce qui nous était inconnu, et le fait de tout développer par nous-même. En revanche, les problèmes d'installation de 'mysql.connector' sont les seuls défauts du projet qui nous ont déplu.

Nathaël BENOIT Paul BABIN

SAÉ « Conception et implémentation d'une base de données »
Sujet « Les sauniers de l'île de Ré »

La coopérative de sel marin de l'île de Ré souhaite informatiser la gestion de ses approvisionnements et de ses ventes. Durant l'été, la plupart des sauniers[1] de l'île apportent leur récolte de sel à la coopérative. Ils fournissent à la coopérative 2 types de produit : du gros sel et de la fleur de sel.

La coopérative mémorise toutes les entrées réalisées par sauniers, par date et par produit. Une fois entré en stock, le sel subit quelques traitements (broyage, tamisage et conditionnement), puis il est mis en vente. Les principaux clients de la coopérative sont des centrales d'achat. La coopérative mémorise toutes les sorties réalisées par client, date et produit (*voir extrait en Annexe A, fichier complet de l'année 2023, dossier* `sel_marin` *sur Updago : fichiers csv*).

Les prix d'achat et les prix de vente hors taxes sont déterminés chaque année à la fin de la saison d'été lors d'une réunion de concertation entre professionnels. Un exemple de prix est joint en annexe B, fichier complet pour 2023 sur Updago.

Travail à faire

1.

➔ **Concevoir le modèle conceptuel de données (MCD) avec l'outil `looping_mcd`.** Vous nommerez votre fichier « `selmarin.loo` ».

Commencer par faire la rétro-conception à partir des informations proposées en Annexe A puis compléter le MCD obtenu afin d'intégrer la gestion des prix de l'annexe B.

Remarques :

- Penser aux contraintes à gérer selon le schéma relationnel défini en annexe A, selon la réalité observée ou les commentaires indiqués.
- Choisir le type de données `date-heure` plutôt que `date` dans `looping_mcd`.
- Attention, pas d'accents dans le nom des tables, champs ou associations.

2.

➔ Générer le MLD (schéma relationnel)

➔ Puis générer le **script SQL** et stocker le dans le fichier « `selmarin_create.sql` »

Remarque *: Attention aux clés étrangères, pensez à rajouter la ligne concernant le moteur de stockage des tables en **INNODB**.*

➔ Créer ensuite une nouvelle base de données dans MySQL nommée « `selmarin-tda` ou `selmarin-tdb` » sous EASYPHP puis importer le script SQL afin de créer les tables. Vérifier les liaisons avec la vue « concepteur » dans PHPMYADMIN sous EASYPHP (*idem TD effectué dans le cours de bdr2*).

[1] Sauniers : personnes qui récoltent le sel à l'île de Ré.

3.

➔ Alimenter vos tables en insérant, avec des requêtes SQL, les tuples proposés en exemples dans les annexes A et B.

Pour cela, **écrire les requêtes SQL d'insertion dans un script unique SQL «** `selmarin_insert.sql` **» que vous importerez ensuite dans MySQL**. Le script doit être fonctionnel ! Chaque requête se terminera par un point-virgule.

4.

➔ Importer dans MySQL les autres données qui apparaissent pour l'année 2023 au sein des feuilles de calcul du classeur `sel_marin_2023.xls` sur Updago. Plusieurs choix sont possibles à ce stade :

- Création de feuilles de calcul Excel correspondant aux colonnes des tables puis import direct dans MySQL. C'est plus simple et pas de Python mais pas automatisé => **Pénalisation !**
- Script d'automatisation Python « `selmarin.py` » qui fonctionnera pour les prochaines années : voir le td saé. Attention, il faudra ouvrir un fichier csv et le traiter en Python.

5.

➔ Concevoir, en langage SQL, **10 requêtes de votre choix (select, insert, update, delete) qui vous semble pertinentes pour un décideur de la coopérative de sela marin de l'île de Ré.**

Les requêtes suivantes sont à consigner dans un script SQL « `selmarin_requetes.sql` » et à tester dans PhpMyAdmin :

Dans l'ensemble des 10 requêtes à réaliser, vous devrez obligatoirement utiliser :

- Au moins un group by
- Au moins un group by having
- Au moins une différence algébrique
- Au moins un insert/update ou delete
- Au moins une vue créée et réutilisée (possibilité de créer un nouvel utilisateur).

6.

➔ Exporter votre base Mysql finale avec l'ensemble des tables et des tuples des questions précédentes dans un script nommé « `selmarin_final.sql` ».

7.

➔ Créer une interface graphique **TKinter** avec python qui servira de programme principal pour appeler les fonctionnalités précédentes. Vous êtes entièrement libre sur l'ergonomie et le choix des widgets.

L'interface devra proposer :

- Un menu de votre choix pour visualiser le contenu des tables principales de votre base de données : saunier, client, …
- L'insertion des tuples d'un fichier Excel de la question 4
- L'exécution des 10 requêtes dela question 5.
- D'autres fonctionnalités qui vous semblent pertinentes

Bonus : Tout automatiser à partir de cette interface graphique, de la création de la base de données, en passant par l'insertion des tuples et la réalisation des requêtes d'interrogation.

8. Faire un bilan de cette saé avec un vidéo contenant un discours cohérent et mentionnant les points suivants :

- Démonstration et tests de votre application

- ce qui fonctionne et ce qui ne fonctionne pas

- ce qui vous auriez aimé développer en plus, ce qui vous a plu/déplu

La qualité des tests sera évaluée et vous permettra de vérifier que votre travail est fonctionnel ou pas, votre honnêteté sera donc aussi évaluée !

<u>Pour récapituler, voici les fichiers à restituer sur Updago pour le jeudi 17 avril à 23h59 dernier délai</u> :

- Fichier « `selmarin.loo` » représentant le MCD issu de Looping_mcd => **Question 1**
- Script SQL « `selmarin_create.sql` » de création de la base de données => **Question 2**
- Script SQL « `selmarin_insert.sql` » => **Question 3**
- Programme Python d'automatisation « `selmarin.py` » => **Question 4**
- Script SQL « `selmarin_requetes.sql` » => **Question 5**
- Script SQL d'export de la base de données complète obtenue sous MySQL « `selmarin_final.sql` » => **Question 6**
- Programme Python avec interface graphique Tkinter « `selmarin_ihm.py` » pour exécuter les fonctionnalités du projet **=> Question 7**
- Vidéo « `selmarin_tuto_tests` » **=> Question 8**

Voici les apprentissages critiques pour cette SAÉ :

Correctement interpréter et prendre en compte le besoin du commanditaire ou du client
Respecter les formalismes de notation
Connaître la syntaxe des langages et savoir l'utiliser
Mesurer l'importance de maîtriser la structure de données à exploiter
Comprendre les structures algorithmiques de base et leur contexte d'usage
Prendre conscience de l'intérêt de la programmation

ANNEXE A : Schéma relationnel

ENTREE (numEnt, dateEnt, qteEnt, numSau, numPdt)
// qteEnt en tonnes obligatoire et supérieur à 0, dateEnt valeur par défaut à la date du jour
numEnt: clé primaire.
numSau: clé étrangère en référence à SAUNIER, obligatoire
numPdt: clé étrangère en référence à PRODUIT, obligatoire

SAUNIER (numSau, nomSau, prenomSau, villeSau)
// Tous les champs obligatoires.
numSau: clé primaire.

PRODUIT (numPdt, libPdt, stockPdt)
// libellé unique, tous les champs obligatoires..
numPdt: clé primaire.

CLIENT (numCli, nomCli, precisionCli, villeCli)
// Couple nom + villeCli => uniques. Tous les champs obligatoires.
numCli: clé primaire.

SORTIE (numSort, dateSort, numCli)
// dateSort valeur par défaut à la date du jour
numSort: clé primaire.
numCli: clé étrangère en référence à CLIENT, obligatoire

CONCERNER (numSort, numPdt, qteSort)
NumSort, numPdt : clé primaire.
NumSort : clé étrangère en référence à SORTIE.
NumPdt : clé étrangère en référence à PRODUIT.

Exemples de tuples de la base de données :

Table **ENTREE**.

NUMENT	**dateEnt**	**qteEnt (t)**	**numSau**	**numPdt**
20241	2024-06-16	1000	1	1
20242	2024-06-18	500	1	2
20243	2024-07-10	1500	2	2

Table **SAUNIER**.

NUMSAU	**nomSau**	**prenomSau**	**villeSau**
1	YVAN	Pierre	Ars-En-Ré
2	PETIT	Marc	Loix

Table **PRODUIT**.

NUMPDT	**libPdt**	**stockPdt (t)**
1	Gros sel	2000
2	Fleur de sel	1000

Table **CLIENT**.

NUMCLI	**nomCli**	**precisionCli**	**villeCli**
1	CAVANA	Marie	LA ROCHELLE
2	BURLET	Michel	LAGORD
3	PEUTOT	Maurice	LAGORD
4	ORGEVAL	Centrale d'Achats	SURGERES

Table **SORTIE**.

NUMSORT	**dateSort**	**numCli**
20241	2024-07-16	1
20242	2024-07-18	1
20243	2024-08-10	2

Table **CONCERNER**.

NUMSORT	*NUMPDT*	**qteSort (t)**
20241	1	300
20241	2	400
20242	1	200
20243	1	100
20243	2	500

ANNEXE B : Exemple de prix (toutes les colonnes sont obligatoires)

Année	Produit	Prix d'achat/Tonne en €	Prix de vente/Tonne en €
2023	Gros sel	270	280
2023	Fleur de sel	3 900	9 500
2024	Gros sel	270	290
2024	Fleur de sel	3 800	10 000
2025	Gros sel	240	300
2025	Fleur de sel	3 500	9 000