

UEL315 Base de données (Groupe D)

Membres du groupe

Etudiant.e	Alias
Mathilde C.	Cloudy23
Kamo G.	Spaghette5
Mathieu L.	mathleys
Filippos K.	filkat34

Objectifs

- Concevoir un diagramme de cas d'utilisation
- Concevoir un diagramme de classes
- Concevoir un modèle conceptuel de données
- Créer la base de données SQLite
- Créer les tables de la base de données SQLite
- Rédiger un script sql avec les requêtes à la base de données
- Exécuter le script SQL pour créer et peupler la base de données
- Tester les requêtes de SELECT, UPDATE, DELETE

Préparation de l'environnement de travail

- Lire la documentation sur le [site officiel](#) concernant les diagrammes de cas d'utilisation et de classes.
- Installer l'extension [PlantUML](#) sur VSCode.
- Regarder [un tutoriel](#) pour prendre en main l'extension VSCode de PlantUML
- Télécharger et installer [DBBrowser](#) pour pouvoir gérer une base de données SQLite
- Se renseigner sur les spécificités de [SQLite](#)

Processus à suivre

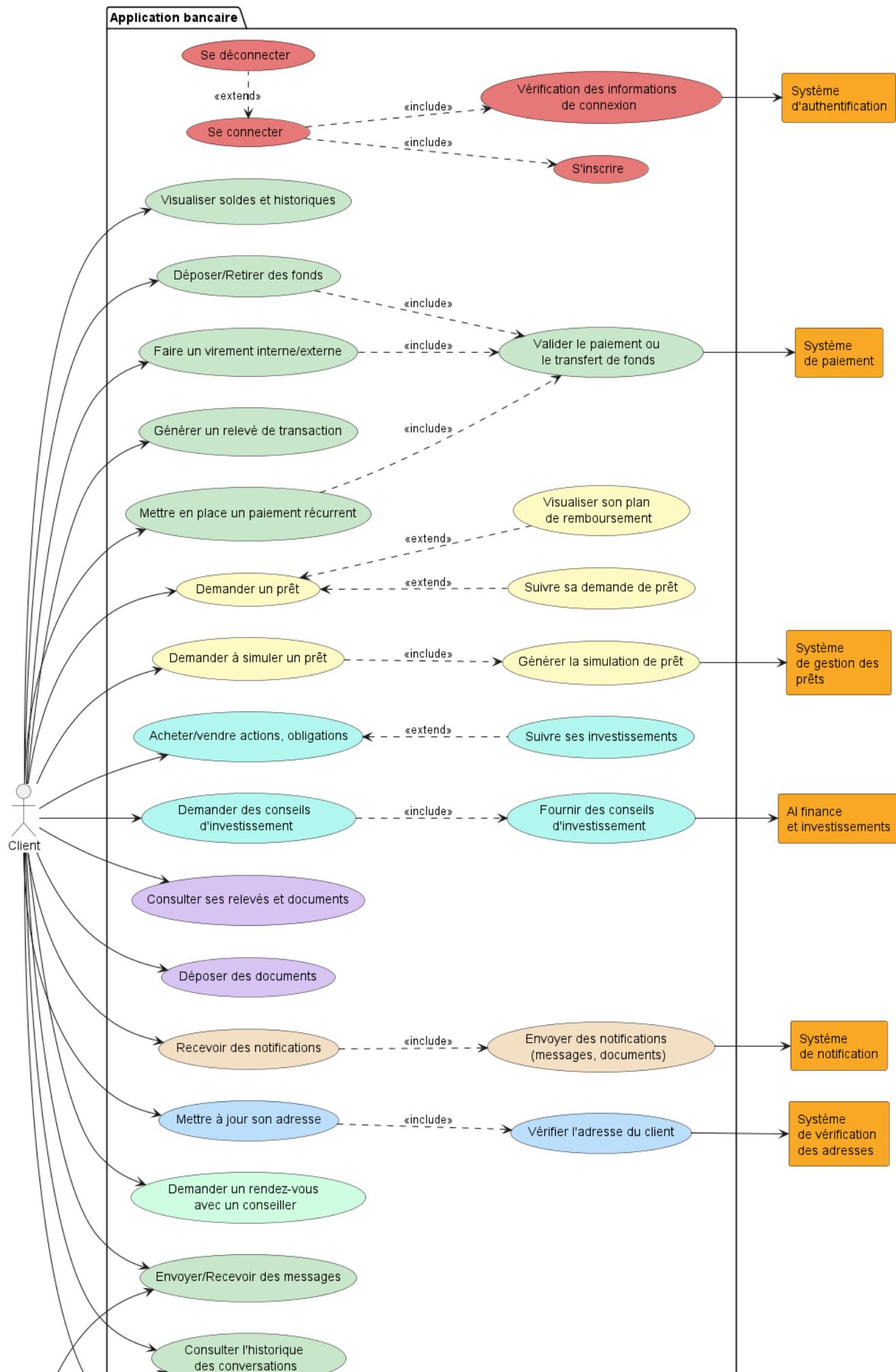
1. Créer une branche du dépôt et travailler sur ses requêtes dans le fichier [bankapp_script.sql](#)
2. Tester ses requêtes en les insérant à la base (fichier [baknapp.db](#)) grâce à DB Browser
3. Prendre des captures d'écran de ce que vous faites sur DB Browser et les déposer dans le dossier [docs](#) du dépôt
4. Faire sa demande de tirage

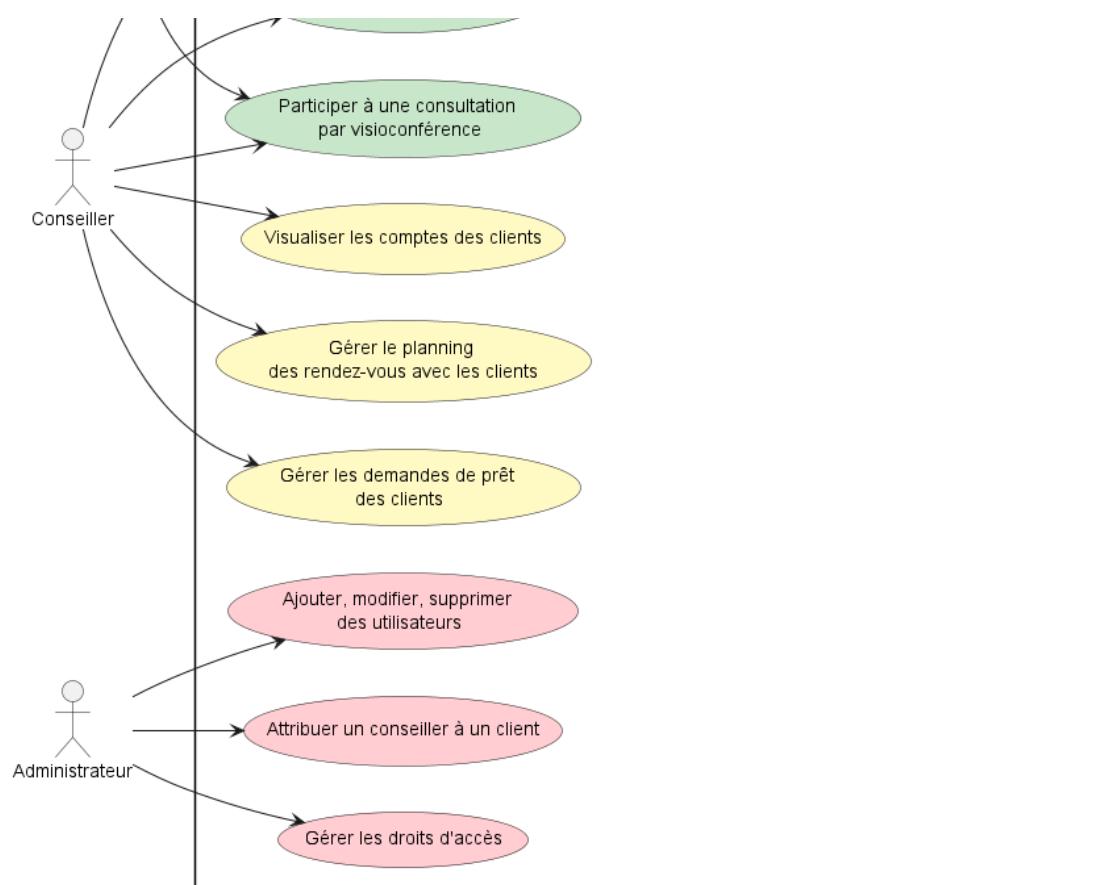
Tâches

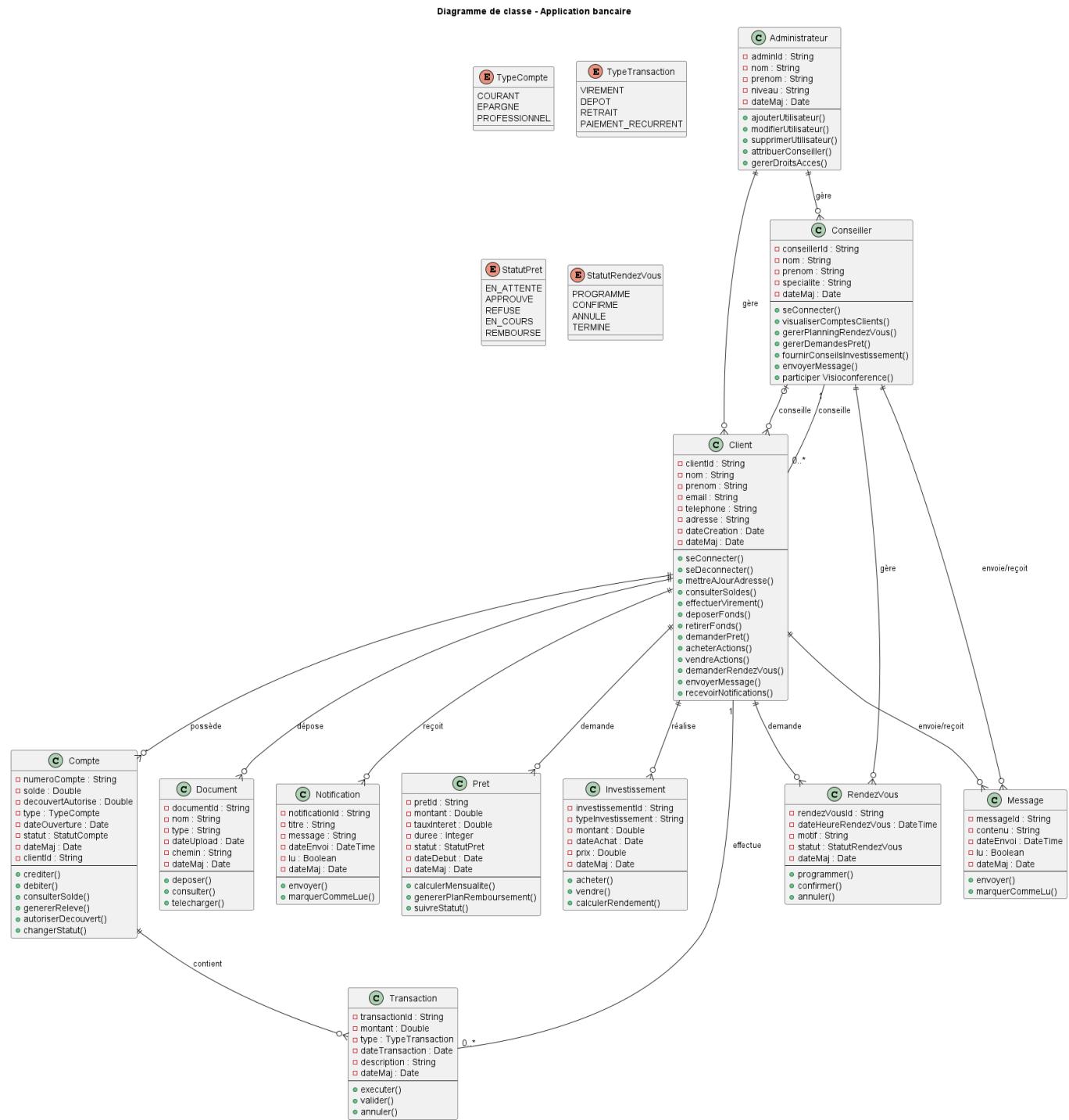
Modelisation de la base des données

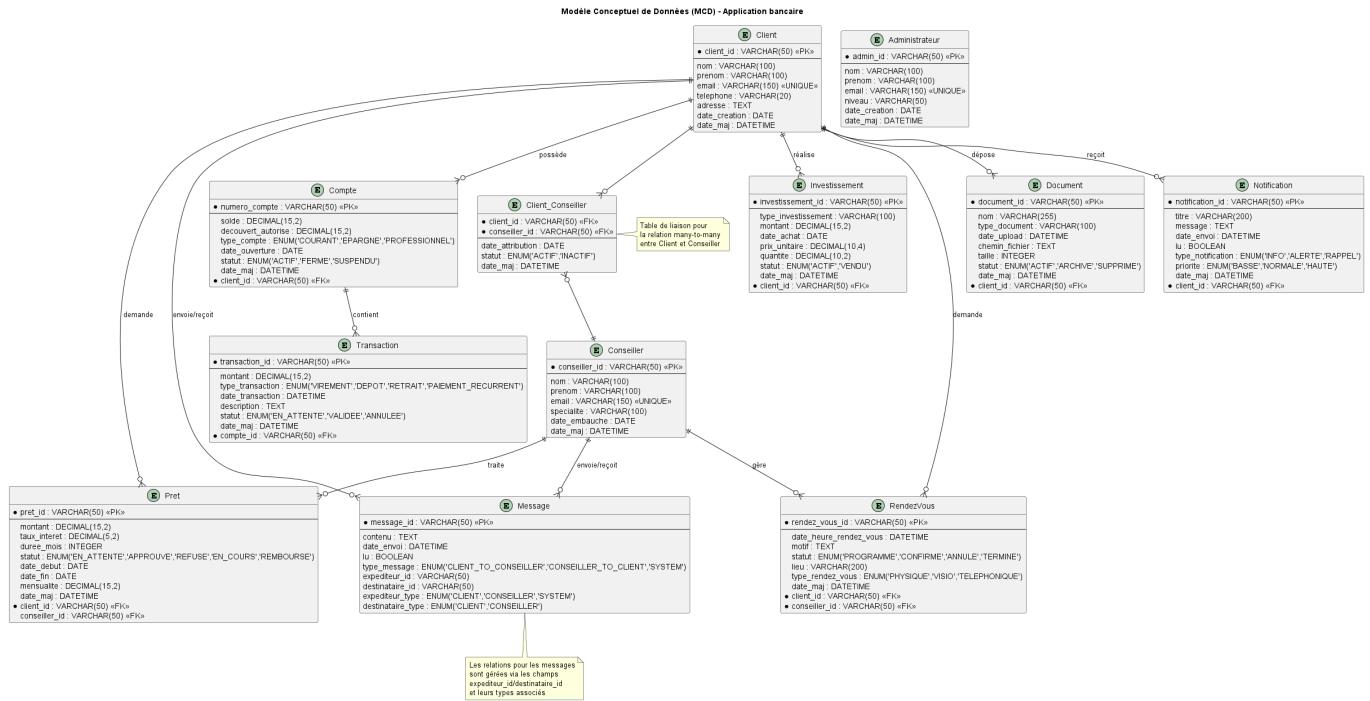
- Concevoir un diagramme de cas d'utilisation
- Concevoir un diagramme de classes
- Concevoir un modèle conceptuel de données

Diagramme des cas d'utilisation - Application bancaire



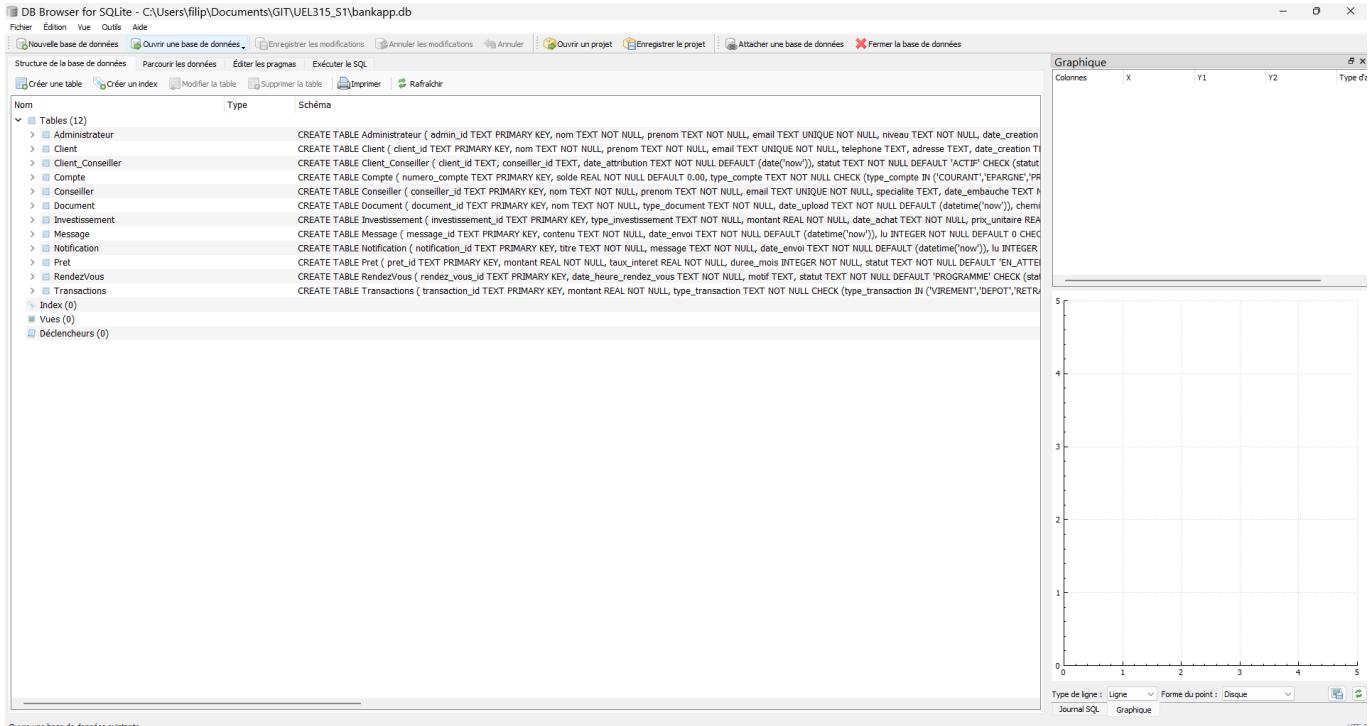






Création de la base de données

- Création d'une base de données SQLite **bankapp.db** grâce au logiciel *DB Browser*
- Ecriture des requêtes SQL de création des différentes tables à partir du MCD
- Exécution du script de création des tables sur DB Browser



Requêtes SQL du TD

1. Insérer des Données

- Ajouter 5 nouveaux clients dans la table Clients.

The screenshot shows the SQLite Manager interface with the following details:

- SQL Editor:** Contains the following SQL query:


```
1 SELECT client_id, nom, prenom, email
2 FROM Client
3 WHERE client_id IN ('CL101','CL102','CL103','CL104','CL105')
4 ORDER BY client_id;
5
6
7
8
```
- Results Table:** Shows the initial 5 rows of the Client table.
- SQL Log:** Displays the execution of the insert statement, showing the insertion of 5 new rows (CL106-CL110) with names like 'Dupont', 'Martin', etc., and their respective emails.
- Message:** "Execution finished without errors. Result: 5 rows returned in 16ms".

- Créer 3 nouveaux comptes pour chaque client ajouté.

The screenshot shows the SQLite Manager interface with the following details:

- SQL Editor:** Contains the following SQL query:


```
1 SELECT numero_compte, client_id, type_compte, solde, statut, date_ouverture
2 FROM Compte
3 WHERE numero_compte LIKE 'ACC10%'*
4 ORDER BY numero_compte;
5
6
7
8
9
10
11
12
13
14
15
```
- Results Table:** Shows the initial 15 rows of the Compte table.
- SQL Log:** Displays the execution of the insert statements, showing the insertion of 15 new rows (ACC106-ACC120) with various account types like 'COURANT', 'EPARGNE', and 'PROFESSIONNEL' across different client IDs.
- Message:** "Execution finished without errors. Result: 15 rows returned in 17ms".

- Insérer 10 transactions pour différents comptes.

```

1 SELECT transaction_id, compte_id, type_transaction, montant, statut, date_transaction
2 FROM Transactions
3 WHERE transaction_id LIKE 'TRX%'
4 ORDER BY datetime(date_transaction) DESC;
5
6
7
8

```

transaction_id	compte_id	type_transaction	montant	statut	date_transaction
TRX008	ACC103C	DEPOT	200.0	VALIDEE	2026-01-12 17:43:04
TRX007	ACC103C	RETRAIT	60.0	EN_ATTENTE	2026-01-09 17:43:04
TRX006	ACC102E	VIREMENT	600.0	VALIDEE	2026-01-06 17:43:04
TRX005	ACC102E	DEPOT	1000.0	VALIDEE	2026-01-04 17:43:04
TRX003	ACC101C	VIREMENT	300.0	VALIDEE	2026-01-02 17:43:04
TRX010	ACC105P	VIREMENT	900.0	VALIDEE	2026-12-30 17:43:04
TRX002	ACC101C	RETRAIT	120.0	VALIDEE	2026-12-27 17:43:04
TRX001	ACC101C	DEPOT	2600.0	VALIDEE	2026-12-26 17:43:04
TRX004	ACC101C	PAIEMENT_RECURRENT	80.0	VALIDEE	2026-12-10 17:43:04
TRX009	ACC104C	PAIEMENT_RECURRENT	180.0	ANNULEE	2026-12-06 17:43:04

Execution finished without errors.
Result: 10 rows returned in 16ms
At line 1:
SELECT transaction_id, compte_id, type_transaction, montant, statut, date_transaction
FROM Transactions
WHERE transaction_id LIKE 'TRX%'
ORDER BY datetime(date_transaction) DESC;

2. Lire des Données

- ✓ Sélectionner les clients ayant un solde supérieur à 10 000 €.

```

2 c.client_id, c.nom, c.prenom,
3 ROUND(SUM(cp.solde), 2) AS solde_total
4 FROM Client c
5 JOIN Compte cp ON cp.client_id = c.client_id
6 GROUP BY c.client_id
7 HAVING solde_total > 10000
8 ORDER BY solde_total DESC;
9
10
11

```

client_id	nom	prenom	solde_total
CL108	Moreau	Yanis	31000.0
CL101	Dupont	Marie	24700.0
CL102	Martin	Lucas	18400.0
CL104	Cassie	Nora	10200.0

Execution finished without errors.
Result: 4 rows returned in 16ms
At line 1:
SELECT
c.client_id, c.nom, c.prenom,
ROUND(SUM(cp.solde), 2) AS solde_total
FROM Client c
JOIN Compte cp ON cp.client_id = c.client_id
GROUP BY c.client_id
HAVING solde_total > 10000
ORDER BY solde_total DESC;

- ✓ Afficher toutes les transactions effectuées le mois dernier.

```

1 SELECT
2     t.transaction_id, t.type_transaction, t.montant, t.date_transaction, t.statut,
3     t.compte_id
4 FROM Transactions t
5 WHERE datetime(t.date_transaction) >= datetime('now','1 month')
6 ORDER BY datetime(t.date_transaction) DESC;
7
8
9
10

```

transaction_id	type_transaction	montant	date_transaction	statut	compte_id
TRX008	DEPOT	200.0	2026-01-12 17:43:04	VALIDEE	ACC103C
TRX007	RETRAIT	60.0	2026-01-09 17:43:04	EN ATTENTE	ACC103C
TRX006	VIREMENT	600.0	2026-01-06 17:43:04	VALIDEE	ACC102E
TRX005	DEPOT	1000.0	2026-01-04 17:43:04	VALIDEE	ACC102E
TRX003	VIREMENT	300.0	2026-01-02 17:43:04	VALIDEE	ACC101C
TRX010	VIREMENT	900.0	2026-12-30 17:43:04	VALIDEE	ACC106P
TRX002	RETRAIT	120.0	2026-12-27 17:43:04	VALIDEE	ACC101C
TRX001	DEPOT	2500.0	2026-12-26 17:43:04	VALIDEE	ACC101C

Execution finished without errors.
Result: 8 rows returned in 12ms
At line 1:
SELECT
t.transaction_id, t.type_transaction, t.montant, t.date_transaction, t.statut,
t.compte_id
FROM Transactions t
WHERE datetime(t.date_transaction) >= datetime('now','1 month')
ORDER BY datetime(t.date_transaction) DESC;

SQL Log

```

187 -- Result: 12 rows returned in 6ms
188 - EXECUTING ALL IN 'SQL 1'
189 -
190 - At line 1:
191 SELECT
192     c.client_id, c.nom, c.prenom,
193     ROUND(SUM(cp.solde), 2) AS solde_total
194     FROM Client c
195     JOIN Compte cp ON cp.client_id = c.client_id
196     GROUP BY c.client_id
197     ORDER BY solde_total DESC;
198 -- Result: 5 rows returned in 14ms
199 - EXECUTING ALL IN 'SQL 1'
200 -
201 - At line 1:
202 SELECT
203     t.transaction_id, t.type_transaction, t.montant, t.date_transaction, t.statut,
204     t.compte_id
205     FROM Transactions t
206     WHERE datetime(t.date_transaction) >= datetime('now','1 month')
207     ORDER BY datetime(t.date_transaction) DESC;
208 - Result: 8 rows returned in 12ms
209

```

- ✓ Lister tous les comptes avec un découvert autorisé.

```

1 SELECT numero_compte, client_id, type_compte, solde
2 FROM Compte
3 WHERE solde < 0;
4

```

numero_compte	client_id	type_compte	solde
ACC103C	CL103	OUEURANT	-180.0

Execution finished without errors.
Result: 1 rows returned in 16ms
At line 1:
SELECT numero_compte, client_id, type_compte, solde
FROM Compte
WHERE solde < 0;

SQL Log

```

213     FROM Transactions
214     WHERE transaction_id LIKE TRX%
215     ORDER BY datetime(date_transaction) DESC;
216 - Result: 10 rows returned in 14ms
217 - EXECUTING ALL IN 'SQL 1'
218 -
219 - At line 1:
220     SELECT
221         SUM(IIF(datetime(date_transaction) >= datetime('now','1 month'), 1, 0)) AS total
222         FROM Transactions
223         WHERE transaction_id LIKE TRX%;
224 - Result: 1 rows returned in 14ms
225 - EXECUTING ALL IN 'SQL 1'
226 -
227 - At line 1:
228     SELECT numero_compte, client_id, type_compte, solde
229     FROM Compte
230     WHERE solde < 0;
231 - Result: 1 rows returned in 16ms
232 - EXECUTING ALL IN 'SQL 1'
233 -
234

```

3. Mettre à Jour des Données

- ✓ Mettre à jour le numéro de téléphone d'un client spécifique.
- ✓ Augmenter le découvert autorisé pour certains comptes.

- Modifier le statut des transactions en attente.

4. Supprimer des Données

- Supprimer les comptes inactifs depuis plus de 2 ans.
- Effacer les transactions refusées ou annulées.
- Retirer les clients sans transactions actives.

5. Requêtes Complexes

- Compter le nombre total de transactions par type de compte.

```

1 -- Compter le nombre total de transactions par type de compte
2 SELECT c.type_compte, COUNT(t.transaction_id) as nombre_transactions
3 FROM Compte c
4 LEFT JOIN Transactions t ON c.numero_compte = t.compte_id
5 GROUP BY c.type_compte;

```

	type_compte	nombre_transactions
1	COURANT	5
2	EPARGNE	3
3	PROFESSIONNEL	0

Execution finished without errors.

Result: 3 rows returned in 8ms

At line 1:

```
-- Compter le nombre total de transactions par type de compte
SELECT c.type_compte, COUNT(t.transaction_id) as nombre_transactions
FROM Compte c
LEFT JOIN Transactions t ON c.numero_compte = t.compte_id
GROUP BY c.type_compte;
```

- Calculer la moyenne des soldes de tous les comptes épargne.

```
1 -- Calculer la moyenne des soldes de tous les comptes épargne
2 SELECT AVG(solde) as moyenne_solde_epargne
3 FROM Compte
4 WHERE type_compte = 'EPARGNE';
```

moyenne_solde_epargne
1 5450.0

Execution finished without errors.

Result: 1 rows returned in 5ms

At line 1:

```
-- Calculer la moyenne des soldes de tous les comptes épargne
SELECT AVG(solde) as moyenne_solde_epargne
FROM Compte
WHERE type_compte = 'EPARGNE';
```

- Trouver les 5 clients les plus actifs en termes de transactions.

```

1 -- Trouver les 5 clients les plus actifs en termes de transactions
2 SELECT cl.nom, cl.prenom, COUNT(t.transaction_id) as total_transactions
3 FROM Client cl
4 JOIN Compte c ON cl.client_id = c.client_id
5 JOIN Transactions t ON c.numero_compte = t.compte_id
6 GROUP BY cl.client_id
7 ORDER BY total_transactions DESC
8 LIMIT 5;

```

	nom	prenom	total_transactions
1	Dupont	Jean	3
2	Petit	Thomas	2
3	Durand	Marie	2
4	Martin	Luc	1

Execution finished without errors.

Result: 4 rows returned in 5ms

At line 1:

```

-- Trouver les 5 clients les plus actifs en termes de transactions
SELECT cl.nom, cl.prenom, COUNT(t.transaction_id) as total_transactions
FROM Client cl
JOIN Compte c ON cl.client_id = c.client_id
JOIN Transactions t ON c.numero_compte = t.compte_id
GROUP BY cl.client_id
ORDER BY total_transactions DESC
LIMIT 5;

```

- Lister les prêts dont la durée restante est inférieure à un an.

```

1 -- Lister les prêts dont la durée restante est inférieure à un an
2 SELECT *
3 FROM Pret
4 WHERE date(date_debut, '+ || duree_mois || 'months') < date('now', '+1 year')
5 AND statut = 'EN_COURS';

```

	pret_id	montant	taux_interet	duree_mois	statut	date_debut	date_fin	mensualite	date_maj	client_id	conseiller_id
1	PRET002	10000.0	3.0	12	EN_COURS	2025-02-14	NULL	850.0	2026-01-14 15:52:48	CLI002	CONSO02

Execution finished without errors.

Result: 1 rows returned in 6ms

At line 1:

```

-- Lister les prêts dont la durée restante est inférieure à un an
SELECT *
FROM Pret
WHERE date(date_debut, '+ || duree_mois || 'months') < date('now', '+1 year')
AND statut = 'EN_COURS';

```

- Afficher le total des prêts accordés par conseiller.

```
1 -- Afficher le total des prêts accordés par conseiller
2 SELECT co.nom, co.prenom, COUNT(p.pret_id) as nombre_prets, SUM(p.montant) as montant_total
3 FROM Conseiller co
4 JOIN Pret p ON co.conseiller_id = p.conseiller_id
5 WHERE p.statut IN ('APPROUVE', 'EN_COURS', 'REMBOURSE')
6 GROUP BY co.conseiller_id;
```

	nom	prenom	nombre_prets	montant_total
1	Grand	Paul	1	200000.0
2	Petit	Sarah	1	10000.0

Execution finished without errors.

Result: 2 rows returned in 15ms

At line 1:

```
-- Afficher le total des prêts accordés par conseiller
SELECT co.nom, co.prenom, COUNT(p.pret_id) as nombre_prets, SUM(p.montant) as montant_total
FROM Conseiller co
JOIN Pret p ON co.conseiller_id = p.conseiller_id
WHERE p.statut IN ('APPROUVE', 'EN_COURS', 'REMBOURSE')
GROUP BY co.conseiller_id;
```

6. Requêtes Avancées

- Identifier les clients avec un total d'investissements supérieur à leur solde total.

```
1 -- Identifier les clients avec un total d'investissements supérieur à leur solde total
2 SELECT c.client_id, c.nom, c.prenom,
3     SUM(i.montant) as total_investissements,
4     (SELECT SUM(solde) FROM Compte WHERE client_id = c.client_id) as total_solde
5 FROM Client c
6 JOIN Investissement i ON c.client_id = i.client_id
7 GROUP BY c.client_id
8 HAVING total_investissements > total_solde;
```

	client_id	nom	prenom	total_investissements	total_solde
1	CLIO02	Durand	Marie	15000.0	13000.0

Execution finished without errors.

Result: 1 rows returned in 5ms

At line 1:

```
-- Identifier les clients avec un total d'investissements supérieur à leur solde total
SELECT c.client_id, c.nom, c.prenom,
    SUM(i.montant) as total_investissements,
    (SELECT SUM(solde) FROM Compte WHERE client_id = c.client_id) as total_solde
FROM Client c
JOIN Investissement i ON c.client_id = i.client_id
GROUP BY c.client_id
HAVING total_investissements > total_solde;
```

- Trouver les comptes ayant le plus haut taux de transactions réussies.

```

1 -- Trouver les comptes ayant le plus haut taux de transactions réussies
2 SELECT c.numero_compte,
3   (CAST(SUM(CASE WHEN t.statut = 'VALIDEE' THEN 1 ELSE 0 END) AS REAL) / COUNT(t.transaction_id)) * 100 as taux_reussite
4 FROM Compte c
5 JOIN Transactions t ON c.numero_compte = t.compte_id
6 GROUP BY c.numero_compte
7 ORDER BY taux_reussite DESC;

```

	numero_compte	taux_reussite
1	CPT006_2	100.0
2	CPT005_1	100.0
3	CPT003_1	100.0
4	CPT002_2	100.0
5	CPT002_1	100.0
6	CPT001_2	100.0
7	CPT001_1	100.0

Execution finished without errors.

Result: 7 rows returned in 6ms

At line 1:

```

-- Trouver les comptes ayant le plus haut taux de transactions réussies
SELECT c.numero_compte,
       (CAST(SUM(CASE WHEN t.statut = 'VALIDEE' THEN 1 ELSE 0 END) AS REAL) / COUNT(t.transaction_id)) * 100 as taux_reussite
FROM Compte c
JOIN Transactions t ON c.numero_compte = t.compte_id
GROUP BY c.numero_compte
ORDER BY taux_reussite DESC;

```

- Lister les clients qui n'ont pas utilisé de services de prêt ou d'investissement.

```
1 -- Lister les clients qui n'ont pas utilisé de services de prêt ou d'investissement
2 SELECT client_id, nom, prenom
3 FROM Client
4 WHERE client_id NOT IN (SELECT client_id FROM Pret)
5 AND client_id NOT IN (SELECT client_id FROM Investissement);
```

	client_id	nom	prenom
1	CLI003	Martin	Luc
2	CLI005	Petit	Thomas

Execution finished without errors.

Result: 2 rows returned in 14ms

At line 1:

```
-- Lister les clients qui n'ont pas utilisé de services de prêt ou d'investissement
SELECT client_id, nom, prenom
FROM Client
WHERE client_id NOT IN (SELECT client_id FROM Pret)
AND client_id NOT IN (SELECT client_id FROM Investissement);
```

- Déterminer le montant total des intérêts générés par les prêts.

```
1 -- Déterminer le montant total des intérêts générés par les prêts
2 SELECT SUM((mensualite * duree_mois) - montant) as total_interets
3 FROM Pret
4 WHERE statut IN ('EN_COURS', 'REMBOURSE') AND mensualite IS NOT NULL;
```

total_interets
1 31800.0

Execution finished without errors.

Result: 1 rows returned in 15ms

At line 1:

```
-- Déterminer le montant total des intérêts générés par les prêts
SELECT SUM((mensualite * duree_mois) - montant) as total_interets
FROM Pret
WHERE statut IN ('EN_COURS', 'REMBOURSE') AND mensualite IS NOT NULL;
```

- Calculer la variation mensuelle du nombre de transactions.

```
1 -- Calculer la variation mensuelle du nombre de transactions
2 WITH MonthlyStats AS (
3     SELECT strftime('%Y-%m', date_transaction) as mois,
4         COUNT(*) as nb_transactions
5     FROM Transactions
6     GROUP BY mois
7 )
8     SELECT mois,
9         nb_transactions,
10        LAG(nb_transactions, 1, 0) OVER (ORDER BY mois) as mois_precedent,
11        (nb_transactions - LAG(nb_transactions, 1, 0) OVER (ORDER BY mois)) as variation
12 FROM MonthlyStats;
```

	mois	nb_transactions	mois_precedent	variation
1	2026-01	8	0	8

Webographie

Autres logiciels de modélisation open source

- [Looping](#)
- [Umbrello](#)

Pour s'entraîner au SQL

- [SQLBolt](#)