

**RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE**  
**MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE**  
**SCIENTIFIQUE**



**UNIVERSITÉ DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMÉDIÈNE**

**FACULTÉ D'INFORMATIQUE**

**DÉPARTEMENT INFORMATIQUE**

**MASTER SYSTÈMES INFORMATIQUES INTELLIGENTS**

**MODULE : BASE DE DONNÉES AVANCÉS**

---

**PROJET BASE DE DONNÉES AVANCÉS**

---

**SI SABER Karim Mounir, 202031067146**  
**OULD ROUIS Zakaria, 202032035272**

**Année universitaire : 2023 / 2024**

## **Table des matières**

1	Introduction générale	<b>1</b>
2	Partie I : Relationnel-Objet	<b>2</b>
2.1	Introduction . . . . .	2
2.2	Tache A : Modélisation orientée objet . . . . .	2
2.3	Tache B : Création des TableSpaces et utilisateur . . . . .	3
2.4	Tache C : Langage de définition de données . . . . .	4
2.5	Tache D : Création des instances dans les tables . . . . .	10
2.6	Tache E : Langage d'interrogation de données . . . . .	14
3	Partie II : NoSQL – Modèle orienté « documents »	<b>18</b>
3.1	Tache A : Modélisation orientée document . . . . .	18
3.2	Tache B : Remplir la base de données . . . . .	20
3.3	Tache C : Répondre aux requêtes . . . . .	21
3.4	Tache D : Analyse de la conception sur les requêtes . . . . .	25
4	Conclusion générale	<b>26</b>

## Table des figures

1	Diagramme de classes UML pour la modélisation relationnelle-objet . . . . .	2
2	Création des TableSpaces . . . . .	3
3	Création de l'utilisateur SQL3 . . . . .	3
4	Attribution des TableSpaces à l'utilisateur . . . . .	3
5	Donner tout les privilèges à l'utilisateur . . . . .	4
6	Tableaux de référence 1 . . . . .	4
7	Tableaux de référence 2 . . . . .	5
8	Tableaux de référence 3 . . . . .	5
9	Ajout des Tab aux types . . . . .	6
10	Ajout des ref aux objets . . . . .	6
11	Méthode 1 . . . . .	7
12	Méthode 2 . . . . .	7
13	Méthode 3 . . . . .	8
14	Méthode 4 . . . . .	8
15	Création de tables p1 . . . . .	9
16	Création de tables p2 . . . . .	10
17	Insertion des succursales . . . . .	10
18	Insertion des Agences . . . . .	11
19	MAJ des tableaux Agence(pour chaque succursale) . . . . .	11
20	Insertion des comptes . . . . .	12
21	Insertions des comptes . . . . .	12
22	MAJ des comptes pour chaque client . . . . .	12
23	MAJ des comptes pour chaque agence . . . . .	13
24	Insertion des opérations . . . . .	13
25	Insertion des prêts . . . . .	13
26	Les comptes des entreprises . . . . .	14
27	Les prêts effectués aux agences . . . . .	14
28	Les comptes sans aucune opération . . . . .	15
29	Montant total des crédits effectués . . . . .	16
30	Les prêts non encore soldés . . . . .	16
31	Le compte le plus mouvementé en 2024 . . . . .	17
32	Exemple insertion . . . . .	19
33	insertion . . . . .	20
34	les prêts effectués de l'agence 102 . . . . .	21
35	Les prêts auprès des agences rattachés aux succursales . . . . .	21
36	Les prêts auprès des agences rattachés aux succursales . . . . .	22
37	Les prêts liés à des dossiers ANSEJ . . . . .	23
38	Les prêts effectués par des clients de type « Particulier » . . . . .	23
39	Les prêts effectués par des clients de type « Particulier » . . . . .	24
40	MapReduce . . . . .	25

# 1 Introduction générale

Dans le cadre de notre cursus à l'USTHB, année académique 2023/2024, nous avons entrepris un projet de base de données avancées visant à explorer les aspects relationnels et orientés documents. Ce projet, réalisé dans le cadre du cours de Bases de Données Avancées, consiste à concevoir et implémenter une base de données pour la gestion des opérations et des prêts bancaires.

L'objectif principal de ce projet est de mettre en pratique les concepts théoriques appris en cours en les appliquant à un cas concret. Nous allons donc transformer un schéma relationnel en un modèle orienté objet, puis explorer les possibilités offertes par un modèle orienté documents avec MongoDB.

Ce projet sera réalisé en binôme et comportera deux parties distinctes : la première partie se concentrera sur la modélisation relationnelle-objet et l'implémentation SQL3-Oracle, tandis que la seconde partie explorera le modèle orienté documents avec MongoDB. Chaque partie du projet sera accompagnée d'une documentation détaillée, comprenant les diagrammes de classes, les scripts de création de base de données, les requêtes SQL ou MongoDB correspondantes, ainsi que des analyses critiques des choix de conception.

À travers ce projet, nous chercherons à approfondir notre compréhension des bases de données avancées, à développer nos compétences en modélisation de données et en langages de requête, ainsi qu'à acquérir une expérience pratique dans la gestion et l'analyse de données complexes.

Dans ce rapport, nous commencerons par présenter la modélisation orientée objet du schéma relationnel fourni, puis nous détaillerons les étapes de création et de peuplement de la base de données relationnelle. Ensuite, nous explorerons la modélisation orientée documents avec MongoDB, en justifiant nos choix de conception et en répondant aux requêtes spécifiques. Enfin, nous conclurons en discutant des résultats obtenus, des limites de notre approche et des pistes d'amélioration pour de futurs travaux.

## 2 Partie I : Relationnel-Objet

### 2.1 Introduction

Dans cette première partie de notre projet de base de données avancées, nous aborderons la modélisation relationnelle-objet ainsi que l'implémentation SQL3-Oracle pour la gestion des opérations et des prêts bancaires. Cette partie est essentielle pour jeter les bases d'une structure de base de données solide et fonctionnelle, en alignement avec les besoins et les spécifications de notre système bancaire.

### 2.2 Tache A : Modélisation orientée objet

1- Transformer le schéma relationnel en un schéma Objet (diagramme de classes UML)

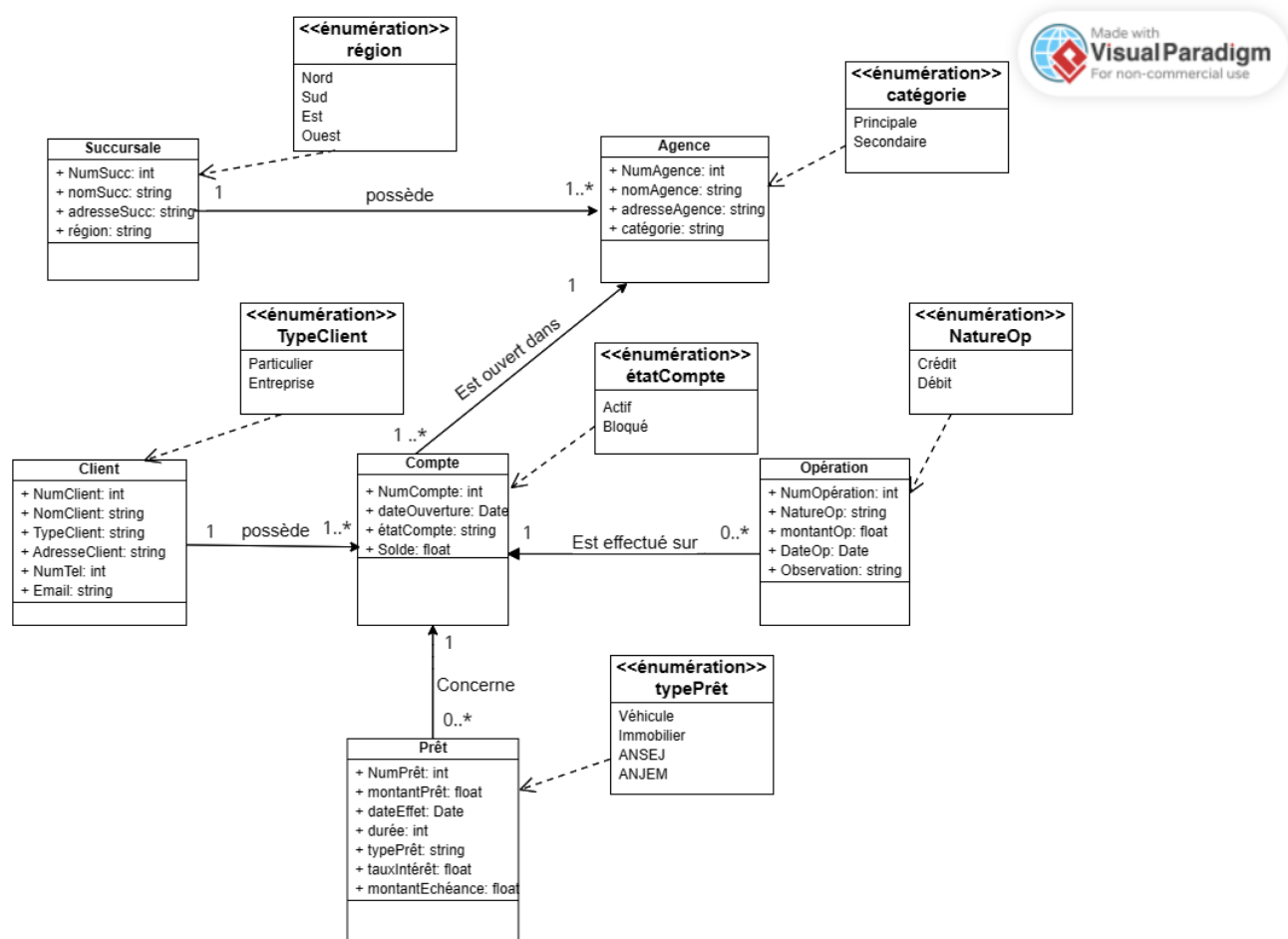


FIGURE 1 – Diagramme de classes UML pour la modélisation relationnelle-objet

## 2.3 Tache B : Création des TableSpaces et utilisateur

### 2. Création de deux TableSpaces SQL3 TBS et SQL3 TempTBS

```
SQL> CREATE TABLESPACE SQL3_TBS
 2  DATAFILE 'sql3_tbs.dbf'
 3  SIZE 100M
 4  AUTOEXTEND ON
 5  NEXT 10M
 6  MAXSIZE UNLIMITED;

Tablespace created.

SQL> CREATE TEMPORARY TABLESPACE SQL3_TempTBS
 2  TEMPFILE 'sql3_temp_tbs.dbf'
 3  SIZE 50M
 4  AUTOEXTEND ON
 5  NEXT 5M
 6  MAXSIZE UNLIMITED;

Tablespace created.
```

FIGURE 2 – Création des TableSpaces

### 3. Création de l'utilisateur SQL3 en lui attribuant les deux tableSpaces

#### \* Création de l'utilisateur

```
SQL> CREATE USER SQL3 IDENTIFIED BY 0666949364;

User created.
```

FIGURE 3 – Création de l'utilisateur SQL3

#### \* Attribution des TableSpaces

```
SQL> ALTER USER SQL3 DEFAULT TABLESPACE SQL3_TBS TEMPORARY TABLESPACE SQL3_TempTBS;

User altered.
```

FIGURE 4 – Attribution des TableSpaces à l'utilisateur

#### 4. Donner tous les privilèges à l'utilisateur

```
SQL> GRANT ALL PRIVILEGES TO SQL3;  
Grant succeeded.
```

FIGURE 5 – Donner tout les privilèges à l'utilisateur

## 2.4 Tache C : Langage de définition de données

#### 5. Définir tous les types abstraits nécessaires et toutes les associations qui existent

##### — Création des types

```
SQL> CREATE OR REPLACE TYPE T_Sucursalle AS OBJECT (  
2   NumSucc INTEGER,  
3   nomSucc VARCHAR(100),  
4   adresseSucc VARCHAR(100),  
5   region VARCHAR(15)  
6   );  
7   /  
  
Type created.  
  
SQL> CREATE OR REPLACE TYPE T_Agence AS OBJECT (  
2   NumAgence INTEGER,  
3   nomAgence VARCHAR(100),  
4   adresseAgence VARCHAR(100),  
5   categorie VARCHAR(15),  
6   succursale REF T_Sucursalle  
7   );  
8   /  
  
Type created.  
  
SQL> CREATE OR REPLACE TYPE T_CLIENT AS OBJECT (  
2   NumClient INTEGER,  
3   NomClient VARCHAR(100),  
4   Typeclient VARCHAR(10),  
5   AdresseClient VARCHAR(100),  
6   NumTel INTEGER,  
7   Email VARCHAR(50)  
8   );  
9   /  
  
Type created.  
  
SQL> CREATE OR REPLACE TYPE T_Compte AS OBJECT (  
2   NumCompte INTEGER,  
3   dateOuverture DATE,  
4   etatCompte VARCHAR(10),  
5   Solde REAL  
6   );  
7   /  
  
Type created.
```

FIGURE 6 – Tableaux de référence 1

```

SQL> CREATE OR REPLACE TYPE T_Pret AS OBJECT (
  2   numPret INTEGER,
  3   montantPret REAL,
  4   dateEffet DATE,
  5   duree INTEGER,
  6   typePret VARCHAR(10),
  7   tauxInteret REAL,
  8   montantEcheance REAL
  9 );
10 /

Type created.

SQL> CREATE OR REPLACE TYPE T_Operation AS OBJECT (
  2   numOperation INTEGER,
  3   natureOp VARCHAR(10),
  4   montantOp REAL,
  5   dateOp DATE,
  6   observation VARCHAR(150)
  7 );
  8 /

Type created.

```

FIGURE 7 – Tableaux de référence 2

## — Création des tableaux de référence

```

SQL> CREATE OR REPLACE TYPE T_ref_Compte_Client AS TABLE OF REF T_Compte ;
  2 /

Type created.

SQL> CREATE OR REPLACE TYPE T_ref_Pret_Compte AS TABLE OF REF T_Pret ;
  2 /

Type created.

SQL> CREATE OR REPLACE TYPE T_ref_Operation_Compte AS TABLE OF REF T_Operation;
  2 /

Type created.

SQL> CREATE OR REPLACE TYPE T_ref_Agence_Sucursalle AS TABLE OF REF T_Agence ;
  2 /

Type created.

SQL> CREATE OR REPLACE TYPE T_ref_Compte_Agence AS TABLE OF REF T_Compte ;
  2 /

Type created.

```

FIGURE 8 – Tableaux de référence 3



## — Ajout des tableaux de référence aux types

```
SQL> ALTER TYPE T_Agence ADD ATTRIBUTE Compte T_ref_Compte_Agence CASCADE ;
Type altered.

SQL> ALTER TYPE T_CLIENT ADD ATTRIBUTE CompteClient T_ref_Compte_Client CASCADE ;
Type altered.

SQL> ALTER TYPE T_Compte ADD ATTRIBUTE OperationCompte T_ref_Operation_Compte CASCADE ;
Type altered.

SQL> ALTER TYPE T_Sucursalle ADD ATTRIBUTE Agence T_ref_Agence_Sucursalle CASCADE;
Type altered.

SQL> ALTER TYPE T_Compte ADD ATTRIBUTE PretCompte T_ref_Pret_Compte CASCADE;
Type altered.
```

FIGURE 9 – Ajout des Tab aux types

## — Ajout des références aux objets

```
SQL> ALTER TYPE T_Compte ADD ATTRIBUTE agence REF T_Agence CASCADE ;
Type altered.

SQL> ALTER TYPE T_Operation ADD ATTRIBUTE compte ref T_Compte CASCADE ;
Type altered.

SQL> ALTER TYPE T_Pret ADD ATTRIBUTE compte REF T_Compte CASCADE ;
Type altered.

SQL> ALTER TYPE T_Compte ADD ATTRIBUTE client REF T_Client CASCADE ;
Type altered.
```

FIGURE 10 – Ajout des ref aux objets

## 6. Définir les méthodes

- Calculer pour chaque agence, le nombre de prêts effectués

```
SQL> ALTER TYPE T_Agence ADD MEMBER FUNCTION nombre_prets RETURN NUMBER CASCADE;  
Type altered.  
  
SQL> CREATE OR REPLACE TYPE BODY T_Agence AS  
2 MEMBER FUNCTION nombre_prets RETURN NUMBER IS  
3 nombre_prets NUMBER;  
4 BEGIN  
5 SELECT COUNT(*) INTO nombre_prets  
6 FROM Pret  
7 WHERE Deref(Deref(compte).agence).numAgence = SELF.numAgence;  
8  
9 RETURN nombre_prets;  
10 END;  
11 END;  
12 /  
Type body created.
```

FIGURE 11 – Méthode 1

- Calculer pour chaque succursale, le nombre d’agences principales qui lui sont rattachées

```
SQL> CREATE OR REPLACE TYPE BODY T_Succursale AS  
2 MEMBER FUNCTION nombre_agences_principales RETURN NUMBER IS  
3 nombre_agences NUMBER;  
4 BEGIN  
5 SELECT COUNT(*) INTO nombre_agences  
6 FROM Agence  
7 WHERE categorie = 'Principale' AND Deref(succursale).numSucc = SELF.numSucc;  
8  
9 RETURN nombre_agences;  
10 END;  
11 END;  
12 /  
Type body created.
```

FIGURE 12 – Méthode 2

- Calculer pour une agence (de numéro donné), le montant global des prêts effectués durant la période donné

```
1 ALTER TYPE ADD STATIC FUNCTION MONTANT_GLOBAL_PRETS_AGENCE(numAgence IN NUMBER)
  RETURN NUMBER CASCADE ;
2
3 CREATE OR REPLACE TYPE BODY T_Agence AS
4   STATIC FUNCTION MONTANT_GLOBAL_PRETS_AGENCE(numAgence IN NUMBER) RETURN
  NUMBER IS
5     montant_total NUMBER;
6   BEGIN
7     SELECT SUM(montantPret)
8     INTO montant_total
9     FROM Pret p
10    WHERE Deref(p.compte).agence.numAgence = numAgence
11    AND p.dateEffet BETWEEN TO_DATE('01-01-2020', 'DD-MM-YYYY') AND TO_DATE(
  '01-01-2024', 'DD-MM-YYYY');
12
13    RETURN montant_total;
14  END;
15 END;
16 /
```

FIGURE 13 – Méthode 3

- Lister toutes agences secondaires ayant au moins un pret «ansej»

```
1 CREATE OR REPLACE TYPE BODY T_Agence AS
2   STATIC FUNCTION agences_secondaires_avec_ANSEJ RETURN SYS_REFCURSOR IS
3     agences_cursor SYS_REFCURSOR;
4   BEGIN
5     OPEN agences_cursor FOR
6       SELECT Deref(A.succursale).numSucc AS numSuccursale, A.numAgence AS
  numAgenceSec
7       FROM Agence A
8       WHERE Deref(A.succursale).typeSuccursale = 'Secondaire'
9       AND EXISTS (
10         SELECT 1
11         FROM Pret p
12         WHERE Deref(Deref(p.compte).agence).numAgence = a.numAgence
13         AND p.typePret = 'ANSEJ'
14       );
15     RETURN agences_cursor;
16  END agences_secondaires_avec_ANSEJ;
17 END;
18 /
19
20 /
```

FIGURE 14 – Méthode 4

La fonction commence par déclarer un curseur nommé `agences_cursor`. Ensuite, elle ouvre ce curseur pour exécuter une requête SQL qui sélectionne les agences principales avec au moins un prêt de type "ANSEJ". Dans la requête, on sélectionne les attributs `numSuccursale` et `numAgencePrincipale` des agences principales.

La clause WHERE spécifie les conditions de sélection :

On vérifie d'abord que l'agence est une agence principale en consultant le type de la succursale associée à l'agence.

Ensuite, on vérifie l'existence d'au moins un prêt de type "ANSEJ" associé à l'agence.

Enfin, la fonction retourne le curseur contenant les résultats de la requête.

Ensuite, on pourra afficher les résultats avec un code PL/SQL

## 7. Création des tables

```
SQL> CREATE TABLE Succursale OF T_Sucursalle (  
2     PRIMARY KEY (NumSucc),  
3     CONSTRAINT CHK_Region CHECK (region IN ('Nord', 'Sud', 'Est', 'Ouest')))  
4     NESTED TABLE Agence Store AS SuccursaleAgence;
```

Table created.

```
SQL> CREATE TABLE Agence OF T_Agence (  
2     PRIMARY KEY (NumAgence),  
3     CONSTRAINT CHK_Categorie CHECK (categorie IN ('Principale', 'Secondaire')),  
4     FOREIGN KEY (succursale) REFERENCES Succursale)  
5     NESTED TABLE Compte STORE AS AgenceCompte;
```

Table created.

```
SQL> CREATE TABLE Client OF T_CLIENT (  
2     PRIMARY KEY (NumClient),  
3     CHECK (TypeClient IN ('Particulier', 'Entreprise'))  
4     NESTED TABLE CompteClient STORE AS ClientCompte ;
```

Table created.

FIGURE 15 – Création de tables p1

```

SQL> CREATE TABLE Compte OF T_Compte (
2     PRIMARY KEY (NumCompte),
3     CONSTRAINT CHK_EtatCompte CHECK (etatCompte IN ('Actif', 'Bloqué')),
4     FOREIGN KEY (client) REFERENCES Client ,
5     FOREIGN KEY (agence) REFERENCES Agence )
6     NESTED TABLE OperationCompte store AS CompteOper,
7     NESTED TABLE PretCompte store AS ComptePret;

Table created.

SQL> CREATE TABLE Pret OF T_Pret (
2     PRIMARY KEY (numPret),
3     CONSTRAINT CHK_TypePret CHECK (typePret IN ('Véhicule', 'Immobilier', 'ANSEJ', 'ANJEM')),
4     FOREIGN KEY (compte) REFERENCES Compte
5 );

Table created.

SQL> CREATE TABLE Operation OF T_Operation (
2     PRIMARY KEY (numOperation),
3     FOREIGN KEY (compte) REFERENCES Compte
4 );

Table created.

```

FIGURE 16 – Création de tables p2

## 2.5 Tache D : Création des instances dans les tables

### — Insertions des succursales

```

SQL> INSERT INTO Succursale VALUES (001, 'Succursale 1', 'Adresse 1', 'Nord', NULL);
1 row created.

SQL> INSERT INTO Succursale VALUES (002, 'Succursale 2', 'Adresse 2', 'Sud', NULL);
1 row created.

SQL> INSERT INTO Succursale VALUES (003, 'Succursale 3', 'Adresse 3', 'Est', NULL);
1 row created.

SQL> INSERT INTO Succursale VALUES (004, 'Succursale 4', 'Adresse 4', 'Ouest', NULL);
1 row created.

SQL> INSERT INTO Succursale VALUES (005, 'Succursale 5', 'Adresse 5', 'Sud', NULL);
1 row created.

SQL> INSERT INTO Succursale VALUES (006, 'Succursale 6', 'Adresse 6', 'Est', NULL);
1 row created.

```

FIGURE 17 – Insertion des succursales

## — Insertions des agences

```
SQL> INSERT INTO Agence VALUES (101, 'Agence 1', 'Adresse 1', 'Principale', (SELECT REF(S) FROM Succursale S WHERE S.NumSucc = 001), NULL);
1 row created.

SQL> INSERT INTO Agence VALUES (102, 'Agence 2', 'Adresse 2', 'Secondaire', (SELECT REF(S) FROM Succursale S WHERE S.NumSucc = 001), NULL);
1 row created.

SQL> INSERT INTO Agence VALUES (124, 'Agence 24', 'Adresse 18', 'Principale', (SELECT REF(S) FROM Succursale S WHERE S.NumSucc = 006), NULL);
1 row created.

SQL> INSERT INTO Agence VALUES (125, 'Agence 25', 'Adresse 18', 'Principale', (SELECT REF(S) FROM Succursale S WHERE S.NumSucc = 006), NULL);
1 row created.

SQL> select COUNT(*) FROM Agence ;

COUNT(*)
-----
25
```

FIGURE 18 – Insertion des Agences

## — Mise a jour des tableaux Agence(pour chaque succursale)

```
SQL> UPDATE Succursale
2 SET Agence = (CAST(MULTISET (SELECT REF(a) FROM Agence a WHERE Deref(a.succursale).numSucc = '001')
3 AS T_ref_Agence_Succursale))
4 WHERE NumSucc = '001';
1 row updated.

SQL> UPDATE Succursale
2 SET Agence = (CAST(MULTISET (SELECT REF(a) FROM Agence a WHERE Deref(a.succursale).numSucc = '002')
3 AS T_ref_Agence_Succursale))
4 WHERE NumSucc = '002';
1 row updated.

SQL> UPDATE Succursale
2 SET Agence = (CAST(MULTISET (SELECT REF(a) FROM Agence a WHERE Deref(a.succursale).numSucc = '003')
3 AS T_ref_Agence_Succursale))
4 WHERE NumSucc = '003';
1 row updated.

SQL> UPDATE Succursale
2 SET Agence = (CAST(MULTISET (SELECT REF(a) FROM Agence a WHERE Deref(a.succursale).numSucc = '004')
3 AS T_ref_Agence_Succursale))
4 WHERE NumSucc = '004';
1 row updated.

SQL> UPDATE Succursale
2 SET Agence = (CAST(MULTISET (SELECT REF(a) FROM Agence a WHERE Deref(a.succursale).numSucc = '005')
3 AS T_ref_Agence_Succursale))
4 WHERE NumSucc = '005';
1 row updated.

SQL> UPDATE Succursale
2 SET Agence = (CAST(MULTISET (SELECT REF(a) FROM Agence a WHERE Deref(a.succursale).numSucc = '006')
3 AS T_ref_Agence_Succursale))
4 WHERE NumSucc = '006';
1 row updated.
```

FIGURE 19 – MAJ des tableaux Agence(pour chaque succursale)

## — Insertions des clients

```
SQL> INSERT INTO Client VALUES (00001, 'Client 1', 'Entreprise', 'Adresse 1', 5678901234, 'client1@gmail.com', NULL);
1 row created.

SQL> INSERT INTO Client VALUES (00099, 'Client 99', 'Particulier', 'Adresse 99', 9012345678, 'client99@gmail.com', NULL);
1 row created.

SQL> INSERT INTO Client VALUES (00100, 'Client 100', 'Entreprise', 'Adresse 100', 0123456789, 'client100@gmail.com', NULL);
1 row created.

SQL> SELECT COUNT(*) FROM Client ;

COUNT(*)
-----
100
```

FIGURE 20 – Insertion des comptes

## — Insertions des comptes

```
SQL> INSERT INTO COMPTE(NumCompte, dateOuverture, etatCompte, solde, client, agence) VALUES
2  (1130005564, TO_DATE('2024-02-24','YYYY-MM-DD'),'Actif',50.00,
3  (SELECT REF(c) FROM Client c WHERE c.numClient = 35),
4  (SELECT REF(a) FROM Agence a WHERE a.numAgence = '113'));
1 row created.
```

```
SQL> select count(*) from Compte ;

COUNT(*)
-----
43
```

FIGURE 21 – Insertions des comptes

## — Mise à jour des Comptes pour chaque client

```
SQL> UPDATE CLIENT c SET CompteClient = (CAST(MULTISET(SELECT REF(cpt) FROM Compte cpt WHERE Deref(cpt.client).numClient
= c.numClient) AS t_ref_Compte_Client));
100 rows updated.
```

FIGURE 22 – MAJ des comptes pour chaque client

## — Mise à jour des comptes pour chaque agence

```
SQL> UPDATE Agence A SET Compte = (CAST(MULTISET(SELECT REF(cpt) FROM Compte cpt WHERE Deref(cpt.Agence).numAgence = A.numAgence) AS t_ref_Compte_Agence));  
25 rows updated.
```

FIGURE 23 – MAJ des comptes pour chaque agence

## — Insertions des opérations

```
SQL> INSERT INTO Operation (numOperation, dateOp, montantop, natureop, compte)  
2 VALUES (137, TO_DATE('2024-12-07', 'YYYY-MM-DD'), 8500.00, 'Crédit', (SELECT REF (c) FROM Compte c WHERE c.numCompte = 1200005583));  
1 row created.  
SQL> SELECT COUNT(*) FROM Operation ;  
  
COUNT(*)  
-----  
33
```

FIGURE 24 – Insertion des opérations

## — Insertion des prêts

```
SQL> INSERT INTO Pret(numPret, montantPret, dateEffet, duree, typePret, tauxInteret, MontantEcheance, compte)  
2 VALUES (36, 390000.00, TO_DATE('2024-06-27', 'YYYY-MM-DD'), 72, 'ANJEM', 0.49, 3900.54, (SELECT REF(c) FROM Compte C WHERE c.numCompte = 1200005583 ));  
1 row created.  
SQL> SELECT COUNT(*) FROM Pret ;  
  
COUNT(*)  
-----  
36
```

FIGURE 25 – Insertion des prêts



## 2.6 Tache E : Langage d'interrogation de données

9. Lister tous les comptes d'une agence donnée, dont les propriétaires sont des entreprises.

```
SELECT C.numCompte FROM Compte C
WHERE Deref(C.Agence).numAgence = 120
AND Deref(C.Client).TypeClient = 'Entreprise' ;
```

```
SQL> SELECT C.numCompte FROM Compte C WHERE Deref(C.agence).numAgence = 122 AND Deref(C.client).typeClient = 'Entreprise';

NUMCOMPTE
-----
1220005585

SQL> SELECT C.numCompte FROM Compte C WHERE Deref(C.agence).numAgence = 120 AND Deref(C.client).typeClient = 'Entreprise';

NUMCOMPTE
-----
1200005583
```

FIGURE 26 – Les comptes des entreprises

10. Lister les prêts effectués auprès des agences rattachées à une succursale

```
SELECT P.numPret, Deref(Deref(P.compte).agence).numAgence,
Deref(P.compte).numCompte, P.montantPret
FROM pret P
WHERE Deref(Deref(Deref(P.compte).agence).succursale).numSucc = 005 ;
```

```
SQL> SELECT P.numPret, Deref(Deref(P.compte).agence).numAgence, Deref(P.compte).numCompte, P.montantPret FROM pret P WHERE Deref(Deref
(Deref(P.compte).agence).succursale).numSucc = 005 ;

NUMPRET Deref(Deref(P.COMPTE).AGENCE).NUMAGENCE Deref(P.COMPTE).NUMCOMPTE
-----
MONTANTPRET
-----
36
390000 120 1200005583
35
390000 120 1200005583
1
50000 120 1200005583
```

FIGURE 27 – Les prêts effectuées aux agences

**11. les comptes sur lesquels aucune opération de débit n'a été effectuée entre 2000 et 2022**

```
SELECT C.numCompte
FROM Compte C
WHERE NOT EXISTS (
    SELECT 1
    FROM Operation O
    WHERE O.compte = REF(C)
    AND O.natureOp = 'Débit'
    AND O.dateOp BETWEEN TO_DATE('2000-01-01', 'YYYY-MM-DD')
    AND TO_DATE('2022-12-31', 'YYYY-MM-DD')
);
```

```
SQL> SELECT C.numCompte
2  FROM Compte C
3  WHERE NOT EXISTS (
4      SELECT 1
5      FROM Operation O
6      WHERE O.compte = REF(C)
7      AND O.natureOp = 'Débit'
8      AND O.dateOp BETWEEN TO_DATE('2000-01-01', 'YYYY-MM-DD') AND TO_DATE('2022-12-31', 'YYYY-MM-DD')
9  );

NUMCOMPTE
-----
1060005569
1300005593
1310005594
1210005584
1150005578
1180005581
1120005575
1340005597
1280005591
1080005571
1360005599
```

FIGURE 28 – Les comptes sans aucune opération

## 12. Le montant total des crédits effectués sur un compte

```
--Notre exemple selon notre BD
SELECT SUM(O.montantOp) AS montant_total_credit
FROM Operation O
WHERE Deref(O.compte).numCompte = 1200005583
AND O.natureOp = 'Crédit'
AND EXTRACT(YEAR FROM O.dateOp) = 2024;
```

```
SQL> SELECT SUM(O.montantOp) AS montant_total_credit
2 FROM Operation O
3 WHERE Deref(O.compte).numCompte = 1200005583
4 AND O.natureOp = 'Crédit'
5 AND EXTRACT(YEAR FROM O.dateOp) = 2024;

MONTANT_TOTAL_CREDIT
-----
117300
```

FIGURE 29 – Montant total des crédits effectués

## 13. Les prêts non encore soldés à ce jour

```
--Si montantEcheance =0 Alors Pret soldé
SELECT P.numPret, Deref(Deref(P.compte).agence).numAgence,
Deref(P.compte).numCompte ,
Deref(Deref(P.compte).client).numClient, P.montantPret
FROM Pret P
WHERE montantEcheance !=0 ;
```

```
SQL> SELECT P.numPret, Deref(Deref(P.compte).agence).numAgence, Deref(P.compte).numCompte , Deref(Deref(P.compte).client).numClient,
P.montantPret
2 FROM Pret P
3 WHERE montantEcheance !=0 ;

NUMPRET Deref(Deref(P.COMPTE).AGENCE).NUMAGENCE Deref(P.COMPTE).NUMCOMPTE
Deref(Deref(P.COMPTE).CLIENT).NUMCLIENT MONTANTPRET
-----
36 54 120 1200005583
390000
35 54 120 1200005583
390000
1 54 120 1200005583
50000
```

FIGURE 30 – Les prêts non encore soldés

#### 14. Le compte le plus mouvementé en 2024

```
SELECT Deref(o.compte).numCompte  
FROM operation o  
WHERE EXTRACT(YEAR FROM o.dateOp) = 2024  
GROUP BY Deref(o.compte).numCompte  
ORDER BY COUNT(o.numOperation) DESC  
FETCH FIRST 1 ROW ONLY;
```

```
SQL> SELECT Deref(o.compte).numCompte  
2 FROM operation o  
3 WHERE EXTRACT(YEAR FROM o.dateOp) = 2024  
4 GROUP BY Deref(o.compte).numCompte  
5 ORDER BY COUNT(o.numOperation) DESC  
6 FETCH FIRST 1 ROW ONLY;  
FETCH FIRST 1 ROW ONLY
```

```
Deref(O.COMPTE).NUMCOMPTE  
-----  
1180005564
```

FIGURE 31 – Le compte le plus mouvementé en 2024

## 3 Partie II : NoSQL – Modèle orienté « documents »

### 3.1 Tache A : Modélisation orientée document

```
"Pret":
{
  "NumPret": 1,
  "montantPret": 5000,
  "dateEffet": new Date("2023-03-24"),
  "duree": 3,
  "typePret": "ANJEM",
  "tauxInteret": 1.50,
  "montantEcheance": 500.50,
  "Compte": {
    "NumCompte": 1,
    "dateOuverture": new Date("2023-02-24"),
    "etatCompte": "Actif",
    "Solde": 100.00,
    "Client": {
      "NumClient": 00001,
      "NomClient": "Karim",
      "TypeClient": "Entreprise",
      "AdresseClient": "Adresse1",
      "NumTel": 0666949364,
      "Email": "ksisaber@gmail.com"
    },
    "Operations": [{
      "NumOperation": 1,
      "NatureOp": "Crédit",
      "montantOp": 8000.00,
      "DateOp": new Date("2023-08-25"),
      "Observation": "RAS"
    }]
  },
  "Agence": {
    "NumAgence": 1,
    "nomAgence": "Agence 1",
    "adresseAgence": "Adresse 1",
    "categorie": "Principale",
    "Succursale": {
      "NumSucc": 1,
      "nomSucc": "Sony",
      "adresseSucc": "Adresse1",
```

```

    "region": "Alger"
  }
}
}

```

## — Illustration avec un exemple

```

1 db.Pret.insertOne({
2   "NumPret": 1,
3   "montantPret": 5000,
4   "dateEffet": "2024-03-24",
5   "duree": 3,
6   "typePret": "ANDEM",
7   "tauxInteret": 1.50,
8   "montantEcheance": 500.50,
9   "Compte": {
10    "NumCompte": 1130005564,
11    "dateOuverture": "2024-02-24",
12    "etatCompte": "Actif",
13    "Solde": 50.00,
14    "Client": {
15     "NomClient": "Client 1",
16     "TypeClient": "Entreprise",
17     "AdresseClient": "Adresse 1"
18   },
19   "NumTel": "1234567890",
20   "Email": "client1@gmail.com"
21 },
22   "Agence": {
23     "NumAgence": 113,
24     "nomAgence": "Sony",
25     "adresseAgence":
26       "Adresse de l'agence",
27     "categorie":
28       "Catégorie de l'agence",
29     "Succursale": {
30       "NumSucc": 3,
31       "nomSucc":
32         "Master Trust Bank ",
33       "adresseSucc": "Tokyo",
34       "region": "Japan"
35     }
36   },
37   "Operations": [
38     {
39       "NumOperation": 1,
40       "NatureOp": "Crédit",
41       "montantOp": 5000.00,
42       "DateOp": "2024-03-24",
43       "Observation":
44         "Versement initial"
45     },
46     {
47       "NumOperation": 2,
48       "NatureOp": "Débit",
49       "montantOp": 500.00,
50       "DateOp": "2024-04-01",
51       "Observation":
52         "Débit pour frais"
53     },
54     {
55       "NumOperation": 3,
56       "NatureOp": "Crédit",
57       "montantOp": 2000.00,
58       "DateOp": "2024-04-15",
59       "Observation":
60         "Nouveau crédit"
61     }
62   ]
63 })

```

FIGURE 32 – Exemple insertion

### — Choix de conception

Dans notre conception, nous avons choisi une modélisation basé sur les prêts, et ce choix est dû au fait que les requêtes portent globalement sur les prêts. Donc, on aura une réduction dans le nombre de requêtes nécessaires pour obtenir toutes les informations associées à un prêt, et aussi moins de dépendance à l'égard des requêtes pour récupérer des informations supplémentaires.

### — Inconvénients de la conception

1. **Redondance des données** : Si les autres entités, telles que les comptes ou les clients, sont souvent référencées dans le contexte des prêts, cela peut entraîner une redondance des données. Par exemple, les mêmes informations sur un client ou un compte peuvent être stockées plusieurs fois dans différents documents de prêt.
2. **Difficulté de mise à jour** : Si une information associée à un compte ou à un client change, elle devra être mise à jour dans chaque document de prêt correspondant. Cela peut rendre la gestion des données plus complexe et potentiellement exposé à des erreurs.
3. **Consommation de stockage supplémentaire** : En stockant des informations sur les comptes et les clients dans chaque document de prêt, cela peut entraîner une utilisation supplémentaire de l'espace de stockage, surtout si les mêmes données sont répétées dans de nombreux documents.
4. **Performance des requêtes** : Bien que les requêtes sur les prêts soient simplifiées, les requêtes impliquant d'autres entités, comme les comptes ou les clients, peuvent devenir plus complexes et moins performantes en raison de la redondance des données.

## 3.2 Tache B : Remplir la base de données

On va utiliser un script pour l'insertion des données.

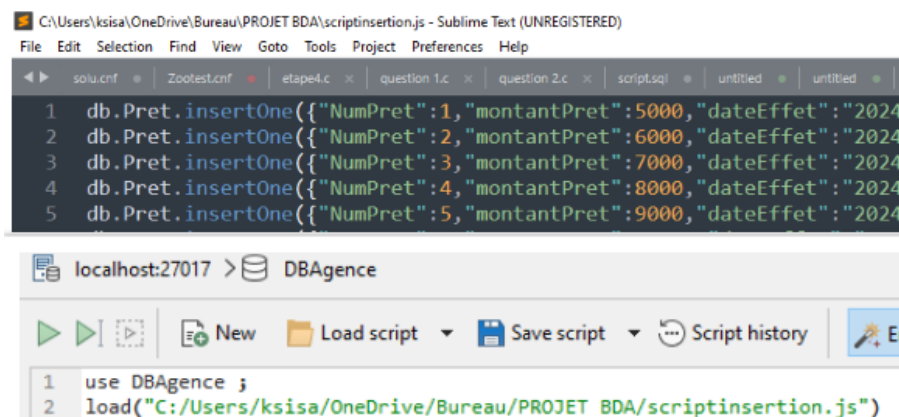
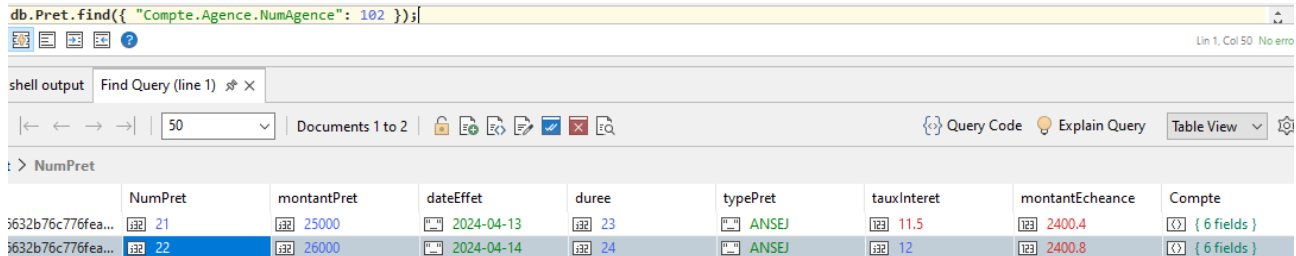


FIGURE 33 – insertion

### 3.3 Tache C : Répondre aux requêtes

#### 1. Afficher tous prêts effectués auprès de l'agence de numéro 102

```
db.Pret.find({ "Compte.Agence.NumAgence": 102 });
```



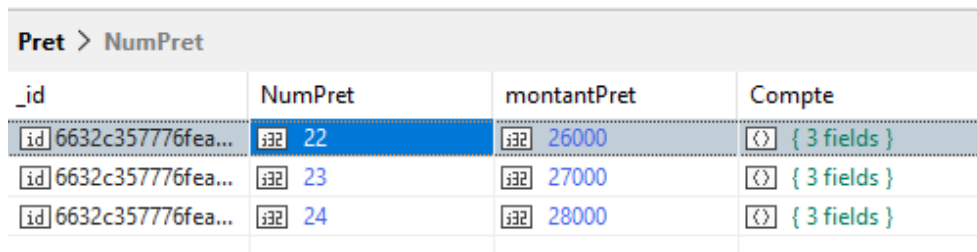
The screenshot shows a MongoDB query interface. The query entered is `db.Pret.find({ "Compte.Agence.NumAgence": 102 });`. The results are displayed in a table view with 9 columns: NumPret, montantPret, dateEffet, duree, typePret, tauxInteret, montantEcheance, and Compte. Two documents are returned, corresponding to NumPret 21 and 22.

	NumPret	montantPret	dateEffet	duree	typePret	tauxInteret	montantEcheance	Compte
632b76c776fea...	21	25000	2024-04-13	23	ANSEJ	11.5	2400.4	{ 6 fields }
632b76c776fea...	22	26000	2024-04-14	24	ANSEJ	12	2400.8	{ 6 fields }

FIGURE 34 – les prêts effectués de l'agence 102

#### 2. Afficher tous prêts effectués auprès des agences rattachées aux succursales de la région « Nord »

```
db.Pret.find({ "Compte.Agence.Succursale.region" : "Nord" },  
{ NumPret: 1, "Compte.Agence.NumAgence": 1, "Compte.NumCompte": 1,  
"Compte.Client.NumClient": 1, montantPret: 1 });
```



The screenshot shows a MongoDB query interface. The query entered is `db.Pret.find({ "Compte.Agence.Succursale.region" : "Nord" }, { NumPret: 1, "Compte.Agence.NumAgence": 1, "Compte.NumCompte": 1, "Compte.Client.NumClient": 1, montantPret: 1 });`. The results are displayed in a table view with 4 columns: \_id, NumPret, montantPret, and Compte. Three documents are returned, corresponding to NumPret 22, 23, and 24.

	NumPret	montantPret	Compte
632c357776fea...	22	26000	{ 3 fields }
632c357776fea...	23	27000	{ 3 fields }
632c357776fea...	24	28000	{ 3 fields }

FIGURE 35 – Les prêts auprès des agences rattachés aux succursales



### 3. Récupérer les numéros des agences et le nombre total de prêts, par agence

```
db.Pret.aggregate([
  { $group: { _id: "$Compte.Agence.NumAgence", totalPrets: { $sum: 1 } } },
  { $sort: { totalPrets: -1 } },
  { $out: "Agence-NbPrêts" }
])
```

Pret > totalPrets	
_id	totalPrets
126	1.0
102	2.0
128	1.0
134	1.0
135	1.0
117	1.0
121	1.0
124	1.0
115	1.0

FIGURE 36 – Les prêts auprès des agences rattachés aux succursales

#### 4. Récupérer tous les prêts liés à des dossiers ANSEJ

```
db.Pret.find(
  { typePret: "ANSEJ" },
  { NumPret: 1, "Compte.Client.NumClient": 1, montantPret: 1, dateEffet: 1 }
).forEach(function(prêt) {
  db.Pret_ANSEJ.insertOne(prêt);
});
```

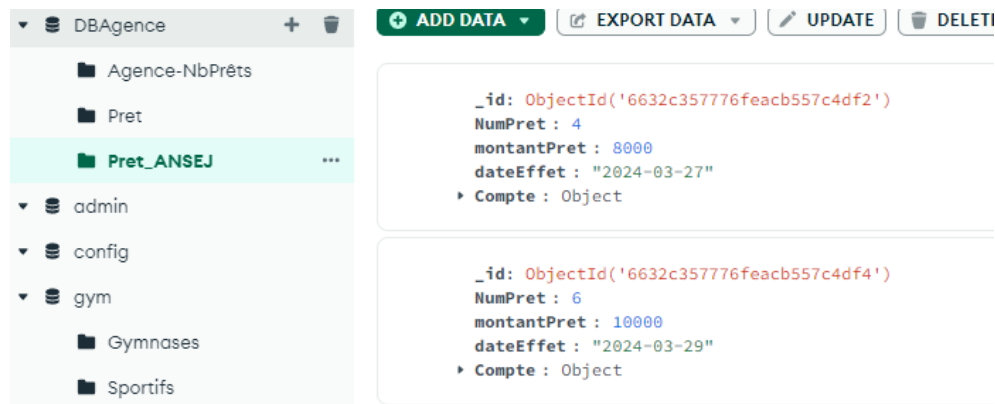


FIGURE 37 – Les prêts liés à des dossiers ANSEJ

#### 5. Afficher tous les prêts effectués par des clients de type « Particulier »

```
db.Pret.find({"Compte.Client.TypeClient" : "Particulier"},
  { NumPret: 1, "Compte.Client.NumClient": 1, montantPret: 1,
    "Compte.Client.NomClient": 1 })
```

Pret > NumPret			
_id	NumPret	montantPret	Compte
<code>id 6632c357776fea...</code>	<code>2</code>	<code>6000</code>	<code>{ 1 fields }</code>
<code>id 6632c357776fea...</code>	<code>4</code>	<code>8000</code>	<code>{ 1 fields }</code>
<code>id 6632c357776fea...</code>	<code>6</code>	<code>10000</code>	<code>{ 1 fields }</code>
<code>id 6632c357776fea...</code>	<code>8</code>	<code>12000</code>	<code>{ 1 fields }</code>
<code>id 6632c357776fea...</code>	<code>10</code>	<code>14000</code>	<code>{ 1 fields }</code>
<code>id 6632c357776fea...</code>	<code>12</code>	<code>16000</code>	<code>{ 1 fields }</code>
<code>id 6632c357776fea...</code>	<code>14</code>	<code>18000</code>	<code>{ 1 fields }</code>
<code>id 6632c357776fea...</code>	<code>16</code>	<code>20000</code>	<code>{ 1 fields }</code>
<code>id 6632c357776fea...</code>	<code>18</code>	<code>22000</code>	<code>{ 1 fields }</code>

FIGURE 38 – Les prêts effectués par des clients de type « Particulier »

## 6. Augmenter de 2000DA, le montant de l'échéance de tous les prêts non encore soldés,

```
db.Pret.updateMany(
  {
    montantEcheance: { $ne: 0 },
    dateEffet: { $lt: new Date("2021-01-01") }
  },
  { $inc: { montantEcheance: 2000 } }
);
```

Après requete

tauxInteret	montantEcheance
12.5	2700.7
12	2600.6
12	2400.8
11.5	2400.4
11	2400.4
10.5	2300.3
10	2200.2
9.5	2100.1

montantEcheance
4100.1
4200.2
4300.3
4400.4
4400.4
4400.8
4600.6
4700.7

FIGURE 39 – Les prêts effectués par des clients de type « Particulier »

## 7. Montant d'échéance des tout les prêts

```
// Fonction de map
var mapFunction = function() {
  emit(this.Agence.NumAgence, 1);
};

// Fonction de réduction :
var reduceFunction = function(key, values) {
  return Array.sum(values);
};

// Fonction finale
var finalizeFunction = function(key, reducedValue) {
  return reducedValue;
};

// Options pour la phase de Map-Reduce :
var options = {
  out: "Agence-NbPrêts",
  finalize: finalizeFunction
};
```

```

};

// Exécuter la phase de Map-Reduce avec les fonctions de map,
// de réduction et les options spécifiées.
db.Pret.mapReduce(
    mapFunction,
    reduceFunction,
    options
);

// Interroge la collection et trie les résultats par ordre décroissant :
db["Agence-NbPrêts"].find().sort({value: -1});

```

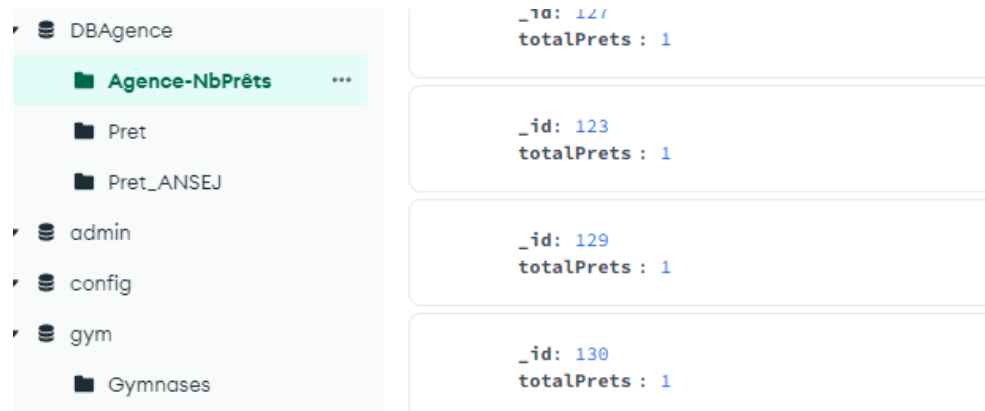


FIGURE 40 – MapReduce

#### 8. Peut-on répondre à la requête suivante : Afficher toutes les opérations de crédit effectuées sur les comptes des clients de type « Entreprise » pendant l'année 2023

Non, car pour obtenir les opérations seules nous devons disposer d'une collection opération distincte, et comme dans cette conception les opérations sont imbriquées et stockées dans l'objet Compte, donc pour arriver à répondre à cette requête il faut une modélisation distincte sur Opération est donc nécessaire

### 3.4 Tache D : Analyse de la conception sur les requêtes

Les requêtes sont efficaces et rapides grâce à une conception axée sur les prêts. Cela signifie que toutes les informations nécessaires sont facilement accessibles au premier niveau du document, évitant ainsi la nécessité de recherches imbriquées.

## 4 Conclusion générale

Le projet de gestion des opérations et des prêts bancaires représente une étape importante dans notre exploration des bases de données relationnelles et NoSQL. À travers ce projet, nous avons pu acquérir une expérience pratique dans la modélisation de données, la création de bases de données, l'écriture de requêtes SQL et MongoDB, ainsi que l'analyse des performances des différentes approches.

La modélisation orientée objet nous a permis de représenter efficacement les entités et les relations du domaine bancaire, offrant une vision claire de la structure de nos données. Nous avons également exploré les avantages de la modélisation orientée documents avec MongoDB, en mettant en évidence sa flexibilité et sa capacité à gérer des données non structurées.

En examinant les requêtes et les performances de nos systèmes relationnel et NoSQL, nous avons pu évaluer les forces et les faiblesses de chaque approche. Nous avons constaté que la conception centrée sur les prêts dans MongoDB permet une récupération rapide et efficace des données, tandis que le modèle relationnel offre une structure plus rigide mais bien définie pour les opérations complexes.

En conclusion, ce projet nous a permis d'approfondir nos connaissances en bases de données et de développer des compétences pratiques essentielles pour la conception et la gestion de systèmes d'information complexes. Il constitue une étape importante dans notre parcours académique et nous prépare à relever les défis futurs dans le domaine passionnant des bases de données.