

Rapport de Projet - Mini-Projet DB2/COBOL

Thème : Développement d'un mini-projet COBOL-DB2 sous z/OS pour le suivi clientèle dans le secteur financier.

Candidat : Josué ROCHA (FORM1112) **Date :** 16-18 Décembre 2025 **Formation :** POEI Développeur Mainframe COBOL - M2i Formation, Strasbourg

Introduction

Ce projet a été réalisé dans le cadre de la formation POEI Développeur Mainframe COBOL. L'objectif était de mettre en pratique les compétences acquises en SQL/DB2 et en programmation COBOL avec accès aux bases de données.

Environnement de travail

- **Système :** z/OS sous emulateur Hercules (TK4-)
- **Interface :** ISPF / TSO
- **Base de données :** DB2
- **Outil SQL :** SPUFI
- **Libraries utilisées :**
 - FORM1112.FINANCE.SOURCE : Programmes COBOL
 - FORM1112.FINANCE.SCRIPT : Requêtes SQL (SPUFI)
 - FORM1112.FINANCE.RESULT : Fichiers de sortie SPUFI

Démarche suivie

1. **Analyse du sujet :** Étude des spécifications et du schéma de données
2. **Création des tables :** Définition des structures avec contraintes PK/FK
3. **Chargement des données :** Insertion de 20 clients et des référentiels
4. **Requêtes SQL :** Vues, index, jointures, agrégations
5. **Programmes COBOL-DB2 :** Développement avec Embedded SQL

Difficultés rencontrées et solutions

Problème	Solution
POSITION mot réservé COBOL	Renommé la colonne en POS
Erreur B37 (espace SPUFI)	Augmentation des tracks du fichier output
MOVE SPACES vers champ numérique edite	Utilisation de INITIALIZE
ABEND 4038 (SYSIN vide)	Validation du numéro compte apres ACCEPT
Alias SQL "C" mot réservé	Simplification de la requête sans alias

Compétences mises en œuvre

- Conception de base de données relationnelle (PK, FK, contraintes CHECK)
 - Requêtes SQL : SELECT, INSERT, UPDATE, CREATE VIEW, CREATE INDEX
 - Fonctions SQL : SUM, AVG, COUNT, CASE WHEN, COALESCE
 - Jointures : INNER JOIN, sous-requêtes corrélées
 - COBOL-DB2 : SELECT INTO, CURSOR, variables host, SQLCODE
 - Techniques COBOL : niveau 88, ruptures de contrôle, ACCEPT
-

Sommaire

1. [Partie 1 : Création et chargement des données](#)
 2. [Partie 2 : Exploitation et manipulation SQL](#)
 3. [Partie 3 : Programmation COBOL-DB2](#)
-

Partie 1 : Création et chargement des données

Exercice 1 : Création des tables

Énoncé

Exécuter les instructions CREATE TABLE de chacune des tables. Chaque table doit comporter sa clé primaire, et les clés étrangères doivent être correctement reliées (FK).

Mon travail

J'ai commencé par créer les 3 tables référentielles (REGION, NATCOMPT, PROFESSI) car elles n'ont pas de dépendances. Ensuite, j'ai créé la table CLIENT qui possède des clés étrangères vers ces 3 tables.

Points importants :

- L'ordre de création est crucial : les tables référencées doivent exister avant la table qui les référence
- J'ai renommé la colonne **POSITION** en **POS** car POSITION est un mot réservé en COBOL
- La contrainte CHECK sur POS garantit que seules les valeurs 'DB' et 'CR' sont acceptées

Résolution

```
-- Table REGION (référentiel des régions)
CREATE TABLE REGION (
    CODE_REGION CHAR(2) NOT NULL PRIMARY KEY,
    NOM_REGION VARCHAR(15)
);

-- Table NATCOMPT (référentiel des natures de compte)
CREATE TABLE NATCOMPT (
    CODE_NATCPT CHAR(2) NOT NULL PRIMARY KEY,
    LIB_NATCPT VARCHAR(30)
);
```

```
-- Table PROFESSI (référentiel des professions)
CREATE TABLE PROFESSI (
    CODE_PROF CHAR(2) NOT NULL PRIMARY KEY,
    LIB_PROF  VARCHAR(20)
);

-- Table CLIENT (table principale avec FK)
CREATE TABLE CLIENT (
    NUM_COMPTE  CHAR(3) NOT NULL PRIMARY KEY,
    CODE_REGION CHAR(2) REFERENCES REGION(CODE_REGION),
    CODE_NATCPT CHAR(2) REFERENCES NATCOMPT(CODE_NATCPT),
    NOM_CLIENT  VARCHAR(10),
    PREN_CLIENT  VARCHAR(10),
    DATE_NAIS   DATE,
    SEXE        CHAR(1),
    CODE_PROF   CHAR(2) REFERENCES PROFESSI(CODE_PROF),
    SIT_FAM     CHAR(1),
    ADRESSE     VARCHAR(20),
    SOLDE       DECIMAL(10,2),
    POS         CHAR(2),
    CHECK(POS IN ('DB','CR'))
);
```

Note : POS remplace POSITION (mot réservé COBOL)

Captures d'écran

```
***** Top of Data *****
-----+-----+-----+-----+-----+
-- EXERCICE 1 : Creation des tables          00010000
-- Note: POS remplace POSITION (mot reserve COBOL) 00020000
                                         00030000
-- DROP TABLE MOUVEMENT;                     00040004
-- DROP TABLE CLIENT;                       00050002
-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
  DROP TABLE REGION;                      00060002
-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
  DROP TABLE NATCOMPT;                   00070002
-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
  DROP TABLE PROFESSI;                  00080002
-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
CREATE TABLE REGION (
  CODE_REGION CHAR(2) NOT NULL PRIMARY KEY,      00100000
  NOM_REGION  VARCHAR(15)                         00110000
) ;                                              00120000
                                                00130000
-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
CREATE TABLE NATCOMPT (
  CODE_NATCPT CHAR(2) NOT NULL PRIMARY KEY,       00150000
  LIB_NATCPT  VARCHAR(30)                          00160000
) ;                                              00170000
-----+-----+-----+-----+-----+
                                                00180000
-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
CREATE TABLE PROFESSI (
  CODE_PROF CHAR(2) NOT NULL PRIMARY KEY,        00200000
  LIB_PROF  VARCHAR(20)                           00210000
) ;                                              00220000
                                                00230000
-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
CREATE TABLE CLIENT (
  NUM_COMPTE  CHAR(3) NOT NULL PRIMARY KEY,       00250001
  CODE_REGION CHAR(2) REFERENCES REGION(CODE_REGION), 00260000
  CODE_NATCPT CHAR(2) REFERENCES NATCOMPT(CODE_NATCPT), 00270000
) ;                                              00280000
```

```

NOM_CLIENT  VARCHAR(10),          00290000
PREN_CLIENT  VARCHAR(10),          00300000
DATE_NAIS    DATE,                00310000
SEXE         CHAR(1),              00320000
CODE_PROF    CHAR(2) REFERENCES PROFESSION(CODE_PROF), 00330000
SIT_FAM      CHAR(1),              00340000
ADRESSE      VARCHAR(20),          00350000
SOLDE        DECIMAL(10,2),        00360000
POS          CHAR(2),              00370000
CHECK(POS IN ('DB','CR'))        00380000
);
-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+
                                         00400000
-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 8
DSNE621I NUMBER OF INPUT RECORDS READ IS 40
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 73
***** Bottom of Data *****

```

Exercice 2 : Alimentation des tables

Énoncé

Insérer les données de base avec des commandes INSERT INTO. Le nombre de clients doit être au minimum de 20 avec des répartitions équilibrées.

Mon travail

J'ai d'abord inséré les données dans les tables référentielles, puis les 20 clients. J'ai veillé à respecter une répartition équilibrée des données :

- 5 clients par région (Paris, Marseille, Lyon, Lille)
- 10 hommes et 10 femmes
- Environ 40% débiteurs (8) et 60% créditeurs (12)
- Toutes les professions et types de comptes représentés

J'ai utilisé la fonction DATE() de DB2 pour formater les dates de naissance.

Résolution

Tables référentielles :

```

-- REGION (4 régions)
INSERT INTO REGION VALUES ('01', 'PARIS');
INSERT INTO REGION VALUES ('02', 'MARSEILLE');

```

```

INSERT INTO REGION VALUES ('03', 'LYON');
INSERT INTO REGION VALUES ('04', 'LILLE');

-- NATCOMPT (5 natures de compte)
INSERT INTO NATCOMPT VALUES ('20', 'COMPTE EPARGNE');
INSERT INTO NATCOMPT VALUES ('25', 'COMPTE CHEQUE');
INSERT INTO NATCOMPT VALUES ('30', 'COMPTE COMMERCIAL');
INSERT INTO NATCOMPT VALUES ('35', 'COMPTE CAMPAGNE AGRICOLE');
INSERT INTO NATCOMPT VALUES ('40', 'COMPTE CDI');

-- PROFESSI (6 professions)
INSERT INTO PROFESSI VALUES ('05', 'MEDECIN');
INSERT INTO PROFESSI VALUES ('10', 'INGENIEUR');
INSERT INTO PROFESSI VALUES ('15', 'COMPTABLE');
INSERT INTO PROFESSI VALUES ('20', 'COMMERCANT');
INSERT INTO PROFESSI VALUES ('25', 'FONCTIONNAIRE');
INSERT INTO PROFESSI VALUES ('30', 'PRIVEE');

```

Table CLIENT (20 enregistrements) :

```

INSERT INTO CLIENT VALUES ('001','01','20','DURAND','ALAIN',DATE('1980-11-02'),'M','05','C','12 RUE DE PARIS',1500.00,'CR');
INSERT INTO CLIENT VALUES ('002','02','25','MARTIN','JEAN',DATE('1985-05-10'),'M','10','M','5 AV JEAN JAURES',-200.00,'DB');
INSERT INTO CLIENT VALUES ('003','03','30','BERNARD','CLAUDE',DATE('1979-03-21'),'M','15','M','8 RUE DE LYON',800.00,'CR');
-- ... (20 clients au total)
INSERT INTO CLIENT VALUES ('020','04','40','GUYOT','PAULINE',DATE('1990-07-13'),'F','05','C','20 RUE DE LILLE',1100.00,'CR');

```

Répartition des données :

Critère	Répartition
Regions	5 clients par region
Sexe	10 M / 10 F
Position	8 DB / 12 CR
Professions	Toutes représentées

Captures d'écran

```
***** Top of Data *****
-----+-----+-----+-----+-----+
-- EXERCICE 2 : Alimentation des tables          00010001
                                                00020001
-- REGION                                     00030001
  INSERT INTO REGION VALUES ('01', 'PARIS');      00040003
-----+-----+-----+-----+-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
  INSERT INTO REGION VALUES ('02', 'MARSEILLE');   00050003
-----+-----+-----+-----+-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
  INSERT INTO REGION VALUES ('03', 'LYON');        00060003
-----+-----+-----+-----+-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
  INSERT INTO REGION VALUES ('04', 'LILLE');       00070003
-----+-----+-----+-----+-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+
-- NATCOMPT                                     00080001
                                                00090001
  INSERT INTO NATCOMPT VALUES ('20', 'COMPTE EPARGNE'); 00100003
-----+-----+-----+-----+-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
  INSERT INTO NATCOMPT VALUES ('25', 'COMPTE CHEQUE'); 00110003
-----+-----+-----+-----+-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
  INSERT INTO NATCOMPT VALUES ('30', 'COMPTE COMMERCIAL'); 00120003
-----+-----+-----+-----+-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
  INSERT INTO NATCOMPT VALUES ('35', 'COMPTE CAMPAGNE AGRICOLE'); 00130003
-----+-----+-----+-----+-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
  INSERT INTO NATCOMPT VALUES ('40', 'COMPTE CDI');     00140003
-----+-----+-----+-----+-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

```
--+-----+-----+-----+-----+-----+-----+
-- PROFESSI                                     00150001
  INSERT INTO PROFESSI VALUES ('05', 'MEDECIN');      00170001
--+-----+-----+-----+-----+-----+-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
--+-----+-----+-----+-----+-----+-----+
  INSERT INTO PROFESSI VALUES ('10', 'INGENIEUR');      00190003
--+-----+-----+-----+-----+-----+-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
--+-----+-----+-----+-----+-----+-----+
  INSERT INTO PROFESSI VALUES ('15', 'COMPTABLE');      00200003
--+-----+-----+-----+-----+-----+-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
--+-----+-----+-----+-----+-----+-----+
  INSERT INTO PROFESSI VALUES ('20', 'COMMERCANT');      00210003
--+-----+-----+-----+-----+-----+-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
--+-----+-----+-----+-----+-----+-----+
  INSERT INTO PROFESSI VALUES ('25', 'FONCTIONNAIRE');      00220003
--+-----+-----+-----+-----+-----+-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
--+-----+-----+-----+-----+-----+-----+
  INSERT INTO PROFESSI VALUES ('30', 'PRIVEE');      00230003
--+-----+-----+-----+-----+-----+-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
--+-----+-----+-----+-----+-----+-----+
```

```

-- CLIENT (20 enregistrements)          00240003
  INSERT INTO CLIENT VALUES           00250001
    ('001','01','20','DURAND','ALAIN',DATE('1980-11-02'),      00260003
     'M','05','C','12 RUE DE PARIS',1500.00,'CR');           00270003
-----+-----+-----+-----+-----+-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+
  INSERT INTO CLIENT VALUES           00290003
    ('002','02','25','MARTIN','JEAN',DATE('1985-05-10'),      00300003
     'M','10','M','5 AV JEAN JAURES',-200.00,'DB');        00310003
-----+-----+-----+-----+-----+-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+
  INSERT INTO CLIENT VALUES           00320003
-----+-----+-----+-----+-----+-----+
  INSERT INTO CLIENT VALUES           00860003
    ('020','04','40','GUYOT','PAULINE',DATE('1990-07-13'),      00870003
     'F','05','C','20 RUE DE LILLE',1100.00,'CR')           00880003
                                         -                   00890001
-----+-----+-----+-----+-----+-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 35
DSNE621I NUMBER OF INPUT RECORDS READ IS 85
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 233
***** Bottom of Data *****

```

Exercice 3 : Vérification de cohérence

Énoncé

Effectuer des requêtes pour valider le chargement.

Mon travail

J'ai exécuté des SELECT simples sur chaque table pour vérifier que toutes les données ont été correctement insérées. J'ai vérifié :

- 4 régions dans REGION
- 5 natures de compte dans NATCOMPT
- 6 professions dans PROFESSI
- 20 clients dans CLIENT avec les bonnes references (FK valides)

Résolution

```
SELECT * FROM REGION;
SELECT * FROM NATCOMPT;
SELECT * FROM PROFESSI;
SELECT * FROM CLIENT;
```

Captures d'écran

```
***** Top of Data *****
-----+-----+-----+-----+-----+
-- EXERCICE 3 : Verification de coherence          00010001
                                                00020000
                                                00030001
SELECT * FROM REGION;
-----+-----+-----+-----+-----+-----+
CODE_REGION  NOM_REGION
-----+-----+-----+-----+-----+-----+
01        PARIS
02        MARSEILLE
03        LYON
04        LILLE
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
-----+-----+-----+-----+-----+
                                                00040001
                                                00050001
SELECT * FROM NATCOMPT;
-----+-----+-----+-----+-----+
CODE_NATCPT  LIB_NATCPT
-----+-----+-----+-----+-----+
20        COMPTE EPARGNE
25        COMPTE CHEQUE
30        COMPTE COMMERCIAL
35        COMPTE CAMPAGNE AGRICOLE
40        COMPTE CDI
DSNE610I NUMBER OF ROWS DISPLAYED IS 5
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
-----+-----+-----+-----+-----+
                                                00060001
                                                00070001
SELECT * FROM PROFESSI;
-----+-----+-----+-----+
CODE_PROF   LIB_PROF
-----+-----+-----+-----+
05        MEDECIN
```

```

10      INGENIEUR
15      COMPTABLE
20      COMMERCANT
25      FONCTIONNAIRE
30      PRIVEE

```

DSNE610I NUMBER OF ROWS DISPLAYED IS 6

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100

NUM_COMPTE	CODE_REGION	CODE_NATCPT	NOM_CLIENT	PREN_CLIENT	DATE_NAIS	SEXE	
001	01	20	DURAND	ALAIN	1980-11-02	M	
002	02	25	MARTIN	JEAN	1985-05-10	M	
003	03	30	BERNARD	CLAUDE	1979-03-21	M	
004	04	35	ROBERT	PAUL	1990-08-15	M	
005	01	40	RICHARD	MICHEL	1976-12-12	M	
006	02	20	PETIT	MARIE	1992-07-04	F	
007	03	25	ROUX	NADIA	1988-09-27	F	
008	04	30	DAVID	JULIEN	1984-10-18	M	
009	01	35	MOREAU	LUCIE	1991-02-11	F	
010	02	40	SIMON	AMELIE	1987-06-24	F	
011	03	20	LAURENT	THOMAS	1993-01-05	M	
012	04	25	LEROY	SOPHIE	1989-04-09	F	
013	01	30	RENAUD	PIERRE	1982-12-08	M	
014	02	35	BLANC	INES	1994-03-02	F	
015	03	40	BONNET	JULIE	1995-05-25	F	
016	04	20	FRANCOIS	ARTHUR	1978-10-29	M	
017	01	25	LEFEBVRE	EMILIE	1992-01-14	F	
018	02	30	MERCIER	CAMILLE	1986-09-06	F	
019	03	35	DUVAL	NICOLAS	1983-02-18	M	
020	04	40	GUYOT	PAULINE	1990-07-13	F	

DSNE610I NUMBER OF ROWS DISPLAYED IS 20

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100

DSNE617I COMMIT PERFORMED, SQLCODE IS 0							
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0							
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72							
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 4							
DSNE621I NUMBER OF INPUT RECORDS READ IS 10							
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 78							
***** Bottom of Data *****							

Partie 2 : Exploitation et manipulation SQL

Exercice 1 : Extraction des clients par profession

Énoncé

Extraire la liste des clients exerçant les professions COMPTABLE, FONCTIONNAIRE et MEDECIN. Créer des vues permanentes pour chacune de ces catégories.

Mon travail

J'ai d'abord réalisé des SELECT simples pour vérifier les données, puis j'ai créé 3 vues permanentes. J'ai utilisé des jointures INNER JOIN avec la table PROFESSI pour inclure le libellé de la profession dans chaque vue.

Choix technique : Filtrer par LIB_PROF plutôt que CODE_PROF rend le code plus lisible et maintenable.

Résolution

```
-- Requêtes de sélection
SELECT * FROM CLIENT WHERE CODE_PROF = '15'; -- COMPTABLE
SELECT * FROM CLIENT WHERE CODE_PROF = '25'; -- FONCTIONNAIRE
SELECT * FROM CLIENT WHERE CODE_PROF = '05'; -- MEDECIN

-- Vues avec jointure (libellé profession)
CREATE VIEW V_CLIENT_COMPTABLE AS
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,
       P.CODE_PROF, P.LIB_PROF
  FROM CLIENT C
 INNER JOIN PROFESSI P ON C.CODE_PROF = P.CODE_PROF
 WHERE P.LIB_PROF = 'COMPTABLE';

CREATE VIEW V_CLIENT_FONCTION AS
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,
       P.CODE_PROF, P.LIB_PROF
  FROM CLIENT C
 INNER JOIN PROFESSI P ON C.CODE_PROF = P.CODE_PROF
 WHERE P.LIB_PROF = 'FONCTIONNAIRE';

CREATE VIEW V_CLIENT_MEDECIN AS
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,
       P.CODE_PROF, P.LIB_PROF
  FROM CLIENT C
 INNER JOIN PROFESSI P ON C.CODE_PROF = P.CODE_PROF
 WHERE P.LIB_PROF = 'MEDECIN';

-- Vérification
SELECT * FROM V_CLIENT_COMPTABLE;
SELECT * FROM V_CLIENT_FONCTION;
SELECT * FROM V_CLIENT_MEDECIN;
```

Captures d'écran

```
***** Top of Data *****
-----+-----+-----+-----+-----+
-- P2 EXERCICE 1 : Extraction des clients par profession          00010001
                                                               00020000
                                                               00030001
-- Clients COMPTABLE                                              00040001
SELECT * FROM CLIENT WHERE CODE_PROF = '15';
-----+-----+-----+-----+-----+
NUM_COMPTE  CODE_REGION  CODE_NATCPT  NOM_CLIENT  PREN_CLIENT  DATE_NAIS  SEXE
-----+-----+-----+-----+-----+
003        03           30          BERNARD     CLAUDE       1979-03-21  M
009        01           35          MOREAU      LUCIE        1991-02-11  F
015        03           40          BONNET      JULIE        1995-05-25  F
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
-----+-----+-----+-----+-----+
                                                               00050001
-- Clients FONCTIONNAIRE                                         00060001
-----+-----+-----+-----+-----+
SELECT * FROM CLIENT WHERE CODE_PROF = '25';                      00070001
-----+-----+-----+-----+-----+
NUM_COMPTE  CODE_REGION  CODE_NATCPT  NOM_CLIENT  PREN_CLIENT  DATE_NAIS  SEXE
-----+-----+-----+-----+-----+
005        01           40          RICHARD     MICHEL      1976-12-12  M
011        03           20          LAURENT    THOMAS      1993-01-05  M
017        01           25          LEFEBVRE   EMILIE      1992-01-14  F
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
-----+-----+-----+-----+-----+
                                                               00080000
-- Clients MEDECIN                                              00090001
-----+-----+-----+-----+-----+
SELECT * FROM CLIENT WHERE CODE_PROF = '05';                      00100001
-----+-----+-----+-----+-----+
NUM_COMPTE  CODE_REGION  CODE_NATCPT  NOM_CLIENT  PREN_CLIENT  DATE_NAIS  SEXE
-----+-----+-----+-----+-----+
001        01           20          DURAND      ALAIN       1980-11-02  M
-----+-----+-----+-----+-----+
WHERE P.LIB_PROF = 'COMPTABLE';                                    00220001
-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
                                                               00230001
CREATE VIEW V_CLIENT_FONCTION AS
-----+-----+-----+-----+-----+
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,
       P.CODE_PROF, P.LIB_PROF
-----+-----+-----+-----+-----+
FROM CLIENT C
-----+-----+-----+-----+-----+
INNER JOIN PROFESSI P ON C.CODE_PROF = P.CODE_PROF          00280001
-----+-----+-----+-----+-----+
WHERE P.LIB_PROF = 'FONCTIONNAIRE';                            00290001
-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
                                                               00300001
CREATE VIEW V_CLIENT_MEDECIN AS
-----+-----+-----+-----+-----+
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,             00310001
-----+-----+-----+-----+-----+

```

```

FROM CLIENT C
INNER JOIN PROFESSI P ON C.CODE_PROF = P.CODE_PROF
WHERE P.LIB_PROF = 'MEDECIN';
-----+-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+
00340001
00350001
00360001
-----+-----+-----+-----+-----+-----+
00370001
-- Verification des vues
00380001
SELECT * FROM V_CLIENT_COMPTABLE;
00390001
-----+-----+-----+-----+-----+-----+
NUM_COMPTE NOM_CLIENT PREN_CLIENT CODE_PROF LIB_PROF
-----+-----+-----+-----+-----+-----+
009      MOREAU      LUCIE       15      COMPTABLE
003      BERNARD     CLAUDE      15      COMPTABLE
015      BONNET      JULIE       15      COMPTABLE
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
-----+-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS_SUCCESSFUL, SQLCODE IS 100
-----+-----+-----+-----+-----+-----+
SELECT * FROM V_CLIENT_FONCTION;          00400001
-----+-----+-----+-----+-----+-----+
NUM_COMPTE NOM_CLIENT PREN_CLIENT CODE_PROF LIB_PROF
-----+-----+-----+-----+-----+-----+
011      LAURENT    THOMAS      25      FONCTIONNAIRE
017      LEFEBVRE   EMILIE      25      FONCTIONNAIRE
005      RICHARD    MICHEL     25      FONCTIONNAIRE
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
-----+-----+-----+-----+-----+-----+
SELECT * FROM V_CLIENT_MEDECIN;           00410001
-----+-----+-----+-----+-----+-----+
NUM_COMPTE NOM_CLIENT PREN_CLIENT CODE_PROF LIB_PROF
-----+-----+-----+-----+-----+-----+
007      ROUX        NADIA       05      MEDECIN
-----+-----+-----+-----+-----+-----+
001      DURAND      ALAIN       05      MEDECIN
013      RENAUD      PIERRE     05      MEDECIN
020      GUYOT        PAULINE     05      MEDECIN
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
-----+-----+-----+-----+-----+-----+
00420001
-----+-----+-----+-----+-----+-----+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 9
DSNE621I NUMBER OF INPUT RECORDS READ IS 42
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 116
***** Bottom of Data *****

```

Énoncé

Réaliser deux requêtes SQL permettant d'isoler les clients débiteurs (DB) et les clients créditeurs (CR).
Matérialiser ces sélections sous forme de vues.

Mon travail

J'ai créé deux vues distinctes pour séparer les clients selon leur position. Ces vues seront utiles pour les programmes COBOL-DB2 qui traiteront séparément les débiteurs et créditeurs.

Observation : Sur les 20 clients, 8 sont débiteurs (solde négatif) et 12 sont créditeurs (solde positif).

Résolution

```
-- Requêtes de sélection
SELECT * FROM CLIENT WHERE POS = 'DB';
SELECT * FROM CLIENT WHERE POS = 'CR';

-- Vues pour reutilisation COBOL-DB2
CREATE VIEW V_CLIENT_DEBITEUR AS
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,
       C.SOLDE, C.POS
  FROM CLIENT C
 WHERE C.POS = 'DB';

CREATE VIEW V_CLIENT_CREDITEUR AS
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,
       C.SOLDE, C.POS
  FROM CLIENT C
 WHERE C.POS = 'CR';

-- Vérification
SELECT * FROM V_CLIENT_DEBITEUR;
SELECT * FROM V_CLIENT_CREDITEUR;
```

Captures d'écran

***** Top of Data *****							
-- P2 EXERCICE 2 : Repartition selon la position du compte (DB/CR)							00010001
-- Clients debiteurs (DB)							00020000
SELECT * FROM CLIENT WHERE POS = 'DB';							00030001
							00040001
NUM_COMPTE	CODE_REGION	CODE_NATCPT	NOM_CLIENT	PREN_CLIENT	DATE_NAIS	SEXE	
002	02	25	MARTIN	JEAN	1985-05-10	M	
004	04	35	ROBERT	PAUL	1990-08-15	M	
007	03	25	ROUX	NADIA	1988-09-27	F	
009	01	35	MOREAU	LUCIE	1991-02-11	F	
012	04	25	LEROY	SOPHIE	1989-04-09	F	
014	02	35	BLANC	INES	1994-03-02	F	
016	04	20	FRANCOIS	ARTHUR	1978-10-29	M	
019	03	35	DUVAL	NICOLAS	1983-02-18	M	

DSNE610I NUMBER OF ROWS DISPLAYED IS 8

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100

-- Clients crediteurs (CR)							00050000
SELECT * FROM CLIENT WHERE POS = 'CR';							00060001
							00070001
NUM_COMPTE	CODE_REGION	CODE_NATCPT	NOM_CLIENT	PREN_CLIENT	DATE_NAIS	SEXE	
001	01	20	DURAND	ALAIN	1980-11-02	M	
003	03	30	BERNARD	CLAUDE	1979-03-21	M	
005	01	40	RICHARD	MICHEL	1976-12-12	M	
006	02	20	PETIT	MARIE	1992-07-04	F	
008	04	30	DAVID	JULIEN	1984-10-18	M	
010	02	40	SIMON	AMELIE	1987-06-24	F	
011	03	20	LAURENT	THOMAS	1993-01-05	M	
013	01	30	RENAUD	PIERRE	1982-12-08	M	
015	03	40	BONNET	JULIE	1995-05-25	F	
017	01	25	LEFEBVRE	EMILIE	1992-01-14	F	
018	02	30	MERCIER	CAMILLE	1986-09-06	F	
020	04	40	GUYOT	PAULINE	1990-07-13	F	

DSNE610I NUMBER OF ROWS DISPLAYED IS 12

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100

-- DROP VIEW V_CLIENT_DEBITEUR;							00080000
-- DROP VIEW V_CLIENT_CREDITEUR;							00090001
							00100001
							00110000
-- Vues pour reutilisation COBOL-DB2							00120001
CREATE VIEW V_CLIENT_DEBITEUR AS							00130001
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,							00140001
C.SOLDE, C.POS							00150001
FROM CLIENT C							00160001
WHERE C.POS = 'DB' ;							00170001

```
+-----+-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
+-----+-----+-----+-----+-----+-----+
CREATE VIEW V_CLIENT_CREDITEUR AS
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,
      C.SOLDE, C.POS
FROM CLIENT C
WHERE C.POS = 'CR';
+-----+-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
+-----+-----+-----+-----+-----+-----+
00180001
00190001
00200001
00210001
00220001
00230001
00240001
00250001
00260001
-- Verification des vues
SELECT * FROM V_CLIENT_DEBITEUR;
+-----+-----+-----+-----+-----+-----+
NUM_COMPTE NOM_CLIENT PREN_CLIENT      SOLDE   POS
+-----+-----+-----+-----+-----+-----+
002      MARTIN     JEAN      -200.00  DB
004      ROBERT     PAUL      -450.00  DB
007      ROUX       NADIA     -600.00  DB
009      MOREAU     LUCIE     -300.00  DB
012      LEROY       SOPHIE    -150.00  DB
014      BLANC       INES      -100.00  DB
016      FRANCOIS   ARTHUR    500.00   DB
019      DUVAL      NICOLAS   -250.00  DB
DSNE610I NUMBER OF ROWS DISPLAYED IS 8
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
+-----+-----+-----+-----+-----+-----+
SELECT * FROM V_CLIENT_CREDITEUR;          00270001
+-----+-----+-----+-----+-----+-----+
NUM_COMPTE NOM_CLIENT PREN_CLIENT      SOLDE   POS
+-----+-----+-----+-----+-----+-----+
001      DURAND     ALAIN     1500.00  CR
```

```

003      BERNARD    CLAUDE        800.00 CR
005      RICHARD    MICHEL       2500.00 CR
006      PETIT      MARIE        1200.00 CR
008      DAVID      JULIEN       950.00 CR
010      SIMON      AMELIE       2100.00 CR
011      LAURENT    THOMAS       600.00 CR
013      RENAUD     PIERRE       1750.00 CR
015      BONNET     JULIE        950.00 CR
017      LEFEBVRE   EMILIE       350.00 CR
018      MERCIER    CAMILLE      750.00 CR
020      GUYOT      PAULINE      1100.00 CR
DSNE610I NUMBER OF ROWS DISPLAYED IS 12
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
-----+-----+-----+-----+-----+-----+
                                         00280000
-----+-----+-----+-----+-----+-----+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 6
DSNE621I NUMBER OF INPUT RECORDS READ IS 28
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 107
***** Bottom of Data *****

```

Exercice 3 : Répartition des clients par region

Énoncé

Mettre en place une répartition des clients par region. Chaque region doit faire l'objet d'une extraction distincte.

Mon travail

J'ai créé 4 vues, une par region (Paris, Marseille, Lyon, Lille). Chaque vue inclut une jointure avec REGION pour afficher le nom de la region.

Vérification : Chaque region contient exactement 5 clients comme prévu dans la répartition initiale.

Résolution

```

-- Requêtes par region
SELECT * FROM CLIENT WHERE CODE_REGION = '01'; -- PARIS
SELECT * FROM CLIENT WHERE CODE_REGION = '02'; -- MARSEILLE
SELECT * FROM CLIENT WHERE CODE_REGION = '03'; -- LYON
SELECT * FROM CLIENT WHERE CODE_REGION = '04'; -- LILLE

-- Vues avec jointure (nom region)
CREATE VIEW V_CLIENT_PARIS AS
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,

```

```
R.CODE_REGION, R.NOM_REGION, C.SOLDE, C.POS  
FROM CLIENT C  
INNER JOIN REGION R ON C.CODE_REGION = R.CODE_REGION  
WHERE R.NOM_REGION = 'PARIS';  
  
CREATE VIEW V_CLIENT_MARSEILLE AS  
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,  
       R.CODE_REGION, R.NOM_REGION, C.SOLDE, C.POS  
FROM CLIENT C  
INNER JOIN REGION R ON C.CODE_REGION = R.CODE_REGION  
WHERE R.NOM_REGION = 'MARSEILLE';  
  
CREATE VIEW V_CLIENT_LYON AS  
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,  
       R.CODE_REGION, R.NOM_REGION, C.SOLDE, C.POS  
FROM CLIENT C  
INNER JOIN REGION R ON C.CODE_REGION = R.CODE_REGION  
WHERE R.NOM_REGION = 'LYON';  
  
CREATE VIEW V_CLIENT_LILLE AS  
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,  
       R.CODE_REGION, R.NOM_REGION, C.SOLDE, C.POS  
FROM CLIENT C  
INNER JOIN REGION R ON C.CODE_REGION = R.CODE_REGION  
WHERE R.NOM_REGION = 'LILLE';  
  
-- Vérification  
SELECT * FROM V_CLIENT_MARSEILLE;
```

Captures d'écran

***** Top of Data *****

-- P2 EXERCICE 3 : Repartition des clients par region 00010001

00020000

-- Clients region PARIS (01) 00030001

SELECT * FROM CLIENT WHERE CODE_REGION = '01'; 00040001

NUM_COMPTE	CODE_REGION	CODE_NATCPT	NOM_CLIENT	PREN_CLIENT	DATE_NAIS	SEXE
001	01	20	DURAND	ALAIN	1980-11-02	M
005	01	40	RICHARD	MICHEL	1976-12-12	M
009	01	35	MOREAU	LUCIE	1991-02-11	F
013	01	30	RENAUD	PIERRE	1982-12-08	M
017	01	25	LEFEBVRE	EMILIE	1992-01-14	F

DSNE610I NUMBER OF ROWS DISPLAYED IS 5

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100

00050000

-- Clients region MARSEILLE (02) 00060001

SELECT * FROM CLIENT WHERE CODE_REGION = '02'; 00070001

NUM_COMPTE	CODE_REGION	CODE_NATCPT	NOM_CLIENT	PREN_CLIENT	DATE_NAIS	SEXE
002	02	25	MARTIN	JEAN	1985-05-10	M
006	02	20	PETIT	MARIE	1992-07-04	F
010	02	40	SIMON	AMELIE	1987-06-24	F
014	02	35	BLANC	INES	1994-03-02	F
018	02	30	MERCIER	CAMILLE	1986-09-06	F

DSNE610I NUMBER OF ROWS DISPLAYED IS 5

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100

00080000

-- Clients region LYON (03) 00090001

SELECT * FROM CLIENT WHERE CODE_REGION = '03'; 00100001

NUM_COMPTE	CODE_REGION	CODE_NATCPT	NOM_CLIENT	PREN_CLIENT	DATE_NAIS	SEXE
003	03	30	BERNARD	CLAUDE	1979-03-21	M
007	03	25	ROUX	NADIA	1988-09-27	F
011	03	20	LAURENT	THOMAS	1993-01-05	M
015	03	40	BONNET	JULIE	1995-05-25	F
019	03	35	DUVAL	NICOLAS	1983-02-18	M

DSNE610I NUMBER OF ROWS DISPLAYED IS 5

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100

00110000

-- Clients region LILLE (04) 00120001

SELECT * FROM CLIENT WHERE CODE_REGION = '04'; 00130001

NUM_COMPTE	CODE_REGION	CODE_NATCPT	NOM_CLIENT	PREN_CLIENT	DATE_NAIS	SEXE
------------	-------------	-------------	------------	-------------	-----------	------

004	04	-	35	ROBERT	PAUL	1990-08-15	M
008	04	-	30	DAVID	JULIEN	1984-10-18	M
012	04	-	25	LEROY	SOPHIE	1989-04-09	F
016	04	-	20	FRANCOIS	ARTHUR	1978-10-29	M
020	04	-	40	GUYOT	PAULINE	1990-07-13	F

DSNE610I NUMBER OF ROWS DISPLAYED IS 5
 DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100

```
-- DROP VIEW V_CLIENT_PARIS;                                     00140001
-- DROP VIEW V_CLIENT_MARSEILLE;                                00150001
-- DROP VIEW V_CLIENT_LYON;                                    00160001
-- DROP VIEW V_CLIENT_LILLE;                                 00170001
-- DROP VIEW V_CLIENT_LILLE;                                 00180001
-- Vues avec jointure (nom region)                           00190001
CREATE VIEW V_CLIENT_PARIS AS                               00200001
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,          00210001
FROM CLIENT C                                           00220001
INNER JOIN REGION R ON C.CODE_REGION = R.CODE_REGION
WHERE R.NOM_REGION = 'PARIS';                            00230001
-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

```
CREATE VIEW V_CLIENT_MARSEILLE AS                         00240001
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,          00250001
       R.CODE_REGION, R.NOM_REGION, C.SOLDE, C.POS           00260001
FROM CLIENT C                                           00270001
INNER JOIN REGION R ON C.CODE_REGION = R.CODE_REGION
WHERE R.NOM_REGION = 'MARSEILLE';                        00280001
-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

```
CREATE VIEW V_CLIENT_LYON AS                            00290001
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,          00300001
       R.CODE_REGION, R.NOM_REGION, C.SOLDE, C.POS           00310001
FROM CLIENT C                                           00320001
INNER JOIN REGION R ON C.CODE_REGION = R.CODE_REGION
WHERE R.NOM_REGION = 'LYON';                            00330001
-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

```
CREATE VIEW V_CLIENT_LILLE AS                          00340001
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,          00350001
       R.CODE_REGION, R.NOM_REGION, C.SOLDE, C.POS           00360001
FROM CLIENT C                                           00370001
INNER JOIN REGION R ON C.CODE_REGION = R.CODE_REGION
WHERE R.NOM_REGION = 'LILLE';                            00380001
-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

```
CREATE VIEW V_CLIENT_LILLE AS                          00390001
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,          00400001
       R.CODE_REGION, R.NOM_REGION, C.SOLDE, C.POS           00410001
FROM CLIENT C                                           00420001
INNER JOIN REGION R ON C.CODE_REGION = R.CODE_REGION
WHERE R.NOM_REGION = 'LILLE';                            00430001
-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
```

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0

						00431301
-- Verification des vues						00431401
SELECT * FROM V_CLIENT_PARIS;						00431501
-----+-----+-----+-----+-----+-----+-----+						
NUM_COMPTE	NOM_CLIENT	PREN_CLIENT	CODE_REGION	NOM_REGION	SOLDE	
-----+-----+-----+-----+-----+-----+-----+						
009	MOREAU	LUCIE	01	PARIS	-300.00	
001	DURAND	ALAIN	01	PARIS	1500.00	
013	RENAUD	PIERRE	01	PARIS	1750.00	
017	LEFEBVRE	EMILIE	01	PARIS	350.00	
005	RICHARD	MICHEL	01	PARIS	2500.00	

DSNE610I NUMBER OF ROWS DISPLAYED IS 5

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100

						00431601
-----+-----+-----+-----+-----+-----+-----+						
SELECT * FROM V_CLIENT_MARSEILLE;						

-----+-----+-----+-----+-----+-----+-----+						
NUM_COMPTE	NOM_CLIENT	PREN_CLIENT	CODE_REGION	NOM_REGION	SOLDE	
-----+-----+-----+-----+-----+-----+-----+						
006	PETIT	MARIE	02	MARSEILLE	1200.00	
002	MARTIN	JEAN	02	MARSEILLE	-200.00	
010	SIMON	AMELIE	02	MARSEILLE	2100.00	
018	MERCIER	CAMILLE	02	MARSEILLE	750.00	
014	BLANC	INES	02	MARSEILLE	-100.00	

DSNE610I NUMBER OF ROWS DISPLAYED IS 5

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100

						00431701
-----+-----+-----+-----+-----+-----+-----+						
SELECT * FROM V_CLIENT_LYON;						

-----+-----+-----+-----+-----+-----+-----+						
NUM_COMPTE	NOM_CLIENT	PREN_CLIENT	CODE_REGION	NOM_REGION	SOLDE	
-----+-----+-----+-----+-----+-----+-----+						
015	BONNET	JULIE	03	LYON	950.00	
019	DUVAL	NICOLAS	03	LYON	-250.00	

```

007      ROUX      NADIA      03      LYON      -600.00
011      LAURENT   THOMAS     03      LYON      600.00
003      BERNARD   CLAUDE     03      LYON      800.00
DSNE610I NUMBER OF ROWS DISPLAYED IS 5
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
-----+-----+-----+-----+-----+-----+
SELECT * FROM V_CLIENT_LILLE;                      00431801
-----+-----+-----+-----+-----+-----+
NUM_COMPTE  NOM_CLIENT  PREN_CLIENT  CODE_REGION  NOM_REGION      SOLDE
-----+-----+-----+-----+-----+-----+
008      DAVID      JULIEN      04      LILLE      950.00
020      GUYOT       PAULINE     04      LILLE      1100.00
016      FRANCOIS   ARTHUR      04      LILLE      500.00
004      ROBERT      PAUL       04      LILLE      -450.00
012      LEROY       SOPHIE     04      LILLE      -150.00
DSNE610I NUMBER OF ROWS DISPLAYED IS 5
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
-----+-----+-----+-----+-----+-----+
                                         00431901
-----+-----+-----+-----+-----+-----+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 12
DSNE621I NUMBER OF INPUT RECORDS READ IS 53
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 162
***** Bottom of Data *****

```

Exercice 4 : Index secondaire sur la region

Énoncé

Créer un index secondaire sur la colonne CODE_REGION pour optimiser les recherches par region.

Mon travail

J'ai créé un index sur CODE_REGION pour accélérer les requêtes de recherche par region. Cet index est particulièrement utile car plusieurs de nos vues et requêtes filtrent par region.

Impact : Les SELECT avec WHERE CODE_REGION = ... bénéficient désormais d'un accès indexé au lieu d'un scan complet de la table.

Résolution

```

CREATE INDEX IDX_CLIENT_REGION ON CLIENT(CODE_REGION);

-- Vérification
SELECT * FROM CLIENT WHERE CODE_REGION = '01';

```

Captures d'écran

```
***** Top of Data *****
-----+-----+-----+-----+-----+
-- P2 EXERCICE 4 : Index secondaire sur CODE_REGION          00010001
                                                00020000
-- DROP INDEX IDX_CLIENT_REGION;                                00030001
-- COMMIT;                                         00040001
CREATE INDEX IDX_CLIENT_REGION ON CLIENT(CODE_REGION);        00050001
-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
                                                00060001
-- Verification                                         00070001
SELECT * FROM CLIENT WHERE CODE_REGION = '01';                00080001
-----+-----+-----+-----+-----+
NUM_COMPTE  CODE_REGION  CODE_NATCPT  NOM_CLIENT  PREN_CLIENT  DATE_NAIS  SEXE
-----+-----+-----+-----+-----+-----+-----+
001        01           20          DURAND      ALAIN        1980-11-02   M
005        01           40          RICHARD     MICHEL       1976-12-12   M
009        01           35          MOREAU      LUCIE        1991-02-11   F
013        01           30          RENAUD      PIERRE      1982-12-08   M
017        01           25          LEFEBVRE    EMILIE       1992-01-14   F
DSNE610I NUMBER OF ROWS DISPLAYED IS 5
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
-----+-----+-----+-----+-----+
                                                00090001
-----+-----+-----+-----+-----+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 2
DSNE621I NUMBER OF INPUT RECORDS READ IS 9
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 32
***** Bottom of Data *****
```

Exercice 5 : Index secondaire sur la profession

Énoncé

Créer un index secondaire sur la colonne CODE_PROF afin de faciliter les traitements regroupés par profession.

Mon travail

De même que pour la region, j'ai créé un index sur CODE_PROF. Les vues V_CLIENT_COMPTABLE, V_CLIENT_FONCTION et V_CLIENT_MEDECIN bénéficieront de cet index.

Remarque : Les deux index (region et profession) peuvent être utilisés simultanément par l'optimiseur DB2 pour les requêtes combinant ces deux critères.

Résolution

```
CREATE INDEX IDX_CLIENT_PROF ON CLIENT(CODE_PROF);

-- Vérification
SELECT * FROM CLIENT WHERE CODE_PROF = '15';
```

Captures d'écran

```
***** Top of Data *****
-----+-----+-----+-----+-----+
-- P2 EXERCICE 5 : Index secondaire sur CODE_PROF          00010001
                                         00020000
-- DROP INDEX IDX_CLIENT_PROF;                                00030001
-- COMMIT;                                                 00040000
CREATE INDEX IDX_CLIENT_PROF ON CLIENT(CODE_PROF);           00050001
-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
                                         00060000
-- Verification
SELECT * FROM CLIENT WHERE CODE_PROF = '15';                00080001
-----+-----+-----+-----+-----+
NUM_COMPTE  CODE_REGION  CODE_NATCPT  NOM_CLIENT  PREN_CLIENT  DATE_NAIS  SEXE
-----+-----+-----+-----+-----+-----+
003        03          30          BERNARD     CLAUDE       1979-03-21  M
009        01          35          MOREAU      LUCIE        1991-02-11  F
015        03          40          BONNET      JULIE        1995-05-25  F
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
-----+-----+-----+-----+-----+
                                         00090000
-----+-----+-----+-----+-----+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 2
DSNE621I NUMBER OF INPUT RECORDS READ IS 9
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 30
***** Bottom of Data *****
```

Exercice 6 : Édition des clients triés par région et profession

Énoncé

Concevoir une requête SQL permettant d'afficher les clients dans l'ordre suivant :

1. Par région (CODE_REGION)
2. Puis par profession (CODE_PROF)
3. Puis par numéro de compte (NUM_COMPTE)

Mon travail

J'ai écrit une requête avec double jointure (REGION et PROFESSION) pour afficher les libellés plutôt que les codes. Le tri multi-niveaux ORDER BY permet de regrouper visuellement les données.

Utilité : Cette requête servira de base pour le programme COBOL-DB2 avec ruptures (P3-Ex05).

Résolution

```
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,
       R.CODE_REGION, R.NOM_REGION,
       P.CODE_PROF, P.LIB_PROF,
       C.SOLDE, C.POS
  FROM CLIENT C
 INNER JOIN REGION R ON C.CODE_REGION = R.CODE_REGION
 INNER JOIN PROFESSION P ON C.CODE_PROF = P.CODE_PROF
 ORDER BY C.CODE_REGION, C.CODE_PROF, C.NUM_COMPTE;
```

Captures d'écran

```
***** Top of Data *****
-----+-----+-----+-----+-----+
-- P2 EXERCICE 6 : Edition des clients tries par region et profession 00010001
                                                               00020000
                                                               00030001
-- Tri : 1) CODE_REGION, 2) CODE_PROF, 3) NUM_COMPTE 00040001
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,
       R.CODE_REGION, R.NOM_REGION,
       P.CODE_PROF, P.LIB_PROF,
       C.SOLDE, C.POS 00050001
00060001
00070001
FROM CLIENT C 00080001
INNER JOIN REGION R ON C.CODE_REGION = R.CODE_REGION 00090001
INNER JOIN PROFESSI P ON C.CODE_PROF = P.CODE_PROF 00100001
ORDER BY C.CODE_REGION, C.CODE_PROF, C.NUM_COMPTE; 00110001
-----+-----+-----+-----+-----+
NUM_COMPTE NOM_CLIENT PREN_CLIENT CODE_REGION NOM_REGION           CODE_PROF LI
-----+-----+-----+-----+-----+
001 DURAND ALAIN 01 PARIS 05 ME
013 RENAUD PIERRE 01 PARIS 05 ME
009 MOREAU LUCIE 01 PARIS 15 CO
005 RICHARD MICHEL 01 PARIS 25 FO
017 LEFEBVRE EMILIE 01 PARIS 25 FO
002 MARTIN JEAN 02 MARSEILLE 10 IN
014 BLANC INES 02 MARSEILLE 10 IN
010 SIMON AMELIE 02 MARSEILLE 20 CO
006 PETIT MARIE 02 MARSEILLE 30 PR
018 MERCIER CAMILLE 02 MARSEILLE 30 PR
007 ROUX NADIA 03 LYON 05 ME
019 DUVAL NICOLAS 03 LYON 10 IN
003 BERNARD CLAUDE 03 LYON 15 CO
015 BONNET JULIE 03 LYON 15 CO
011 LAURENT THOMAS 03 LYON 25 FO
020 GUYOT PAULINE 04 LILLE 05 ME
008 DAVID JULIEN 04 LILLE 10 IN
004 ROBERT PAUL 04 LILLE 20 CO
016 FRANCOIS ARTHUR 04 LILLE 20 CO
012 LEROY SOPHIE 04 LILLE 30 PR
DSNE610I NUMBER OF ROWS DISPLAYED IS 20
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
-----+-----+-----+-----+-----+
00120001
-----+-----+-----+-----+-----+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 1
DSNE621I NUMBER OF INPUT RECORDS READ IS 12
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 47
***** Bottom of Data *****
```

Énoncé

Réaliser une requête SQL permettant de fusionner les listes de clients COMPTABLES et FONCTIONNAIRES dans un même résultat.

Mon travail

J'ai utilisé l'opérateur UNION pour fusionner deux SELECT indépendants. Chaque SELECT filtre sur une profession différente et inclut la jointure avec PROFESSI pour le libellé.

Note technique : UNION élimine automatiquement les doublons (si un client était dans les deux catégories). Pour conserver les doublons, on utiliseraient UNION ALL.

Résolution

```
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,
       P.CODE_PROF, P.LIB_PROF,
       C.SOLDE, C.POS
  FROM CLIENT C
 INNER JOIN PROFESSI P ON C.CODE_PROF = P.CODE_PROF
 WHERE P.LIB_PROF = 'COMPTABLE'
UNION
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,
       P.CODE_PROF, P.LIB_PROF,
       C.SOLDE, C.POS
  FROM CLIENT C
 INNER JOIN PROFESSI P ON C.CODE_PROF = P.CODE_PROF
 WHERE P.LIB_PROF = 'FONCTIONNAIRE';
```

Captures d'écran

```
***** Top of Data *****
-----+-----+-----+-----+-----+
-- P2 EXERCICE 7 : Fusion COMPTABLES + FONCTIONNAIRES          00010002
                                                               00020000
                                                               00030002
-- Fusion avec UNION                                              00040001
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,
       P.CODE_PROF, P.LIB_PROF,
       C.SOLDE, C.POS                                              00050002
                                                               00060002
FROM CLIENT C                                                 00070002
INNER JOIN PROFESSI P ON C.CODE_PROF = P.CODE_PROF          00080002
WHERE P.LIB_PROF = 'COMPTABLE'                                 00090002
UNION                                                       00100002
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,
       P.CODE_PROF, P.LIB_PROF,
       C.SOLDE, C.POS                                              00110002
                                                               00120002
                                                               00130002
FROM CLIENT C                                                 00140002
INNER JOIN PROFESSI P ON C.CODE_PROF = P.CODE_PROF          00150002
                                                               00160002
WHERE P.LIB_PROF = 'FONCTIONNAIRE';                           00170002
-----+-----+-----+-----+-----+-----+
NUM_COMPTE  NOM_CLIENT  PREN_CLIENT  CODE_PROF  LIB_PROF      SOL
-----+-----+-----+-----+-----+-----+
003        BERNARD     CLAUDE      15         COMPTABLE    800.
005        RICHARD     MICHEL     25         FONCTIONNAIRE 2500.
009        MOREAU      LUCIE      15         COMPTABLE   -300.
011        LAURENT     THOMAS     25         FONCTIONNAIRE 600.
015        BONNET      JULIE      15         COMPTABLE    950.
017        LEFEBVRE    EMILIE     25         FONCTIONNAIRE 350.
DSNE610I NUMBER OF ROWS DISPLAYED IS 6
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
-----+-----+-----+-----+-----+-----+
                                                               00170002
-----+-----+-----+-----+-----+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 1
DSNE621I NUMBER OF INPUT RECORDS READ IS 17
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 38
***** Bottom of Data *****
```

Exercice 8 : Vue CLIENT réduit

Énoncé

Créer une vue simplifiée de la table CLIENT contenant uniquement : numéro de compte, code région, nature de compte, nom/prenom, activité professionnelle, situation familiale, solde, position.

Mon travail

J'ai créé une vue qui exclut les colonnes DATE_NAIS, SEXE et ADRESSE. Cette vue allège les requêtes qui n'ont pas besoin de ces informations personnelles.

Avantage : La vue peut aussi servir à limiter l'accès aux données sensibles (date de naissance, adresse) pour certains utilisateurs.

Résolution

```
CREATE VIEW V_CLIENT_REDUIT AS
SELECT NUM_COMPTE,
       CODE_REGION,
       CODE_NATCPT,
       NOM_CLIENT,
       PREN_CLIENT,
       CODE_PROF,
       SIT_FAM,
       SOLDE,
       POS
FROM CLIENT;

-- Vérification
SELECT * FROM V_CLIENT_REDUIT;
```

Captures d'écran

```
***** Top of Data *****
-----+-----+-----+-----+-----+
-- P2 EXERCICE 8 : Vue CLIENT reduit          00010001
                                              00020000
                                              00030001
                                              00040001
                                              00050001
                                              00060001
                                              00070001
                                              00080001
                                              00090001
                                              00100001
                                              00110001
                                              00120001
                                              00130001
                                              00140001
                                              00150001
-- DROP VIEW V_CLIENT_REDUIT;
CREATE VIEW V_CLIENT_REDUIT AS
SELECT NUM_COMPTE,
       CODE_REGION,
       CODE_NATCPT,
       NOM_CLIENT,
       PREN_CLIENT,
       CODE_PROF,
       SIT_FAM,
       SOLDE,
       POS
FROM CLIENT;
-----+-----+-----+-----+-----+-----+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+
00160001
-- Verification
00170001
SELECT * FROM V_CLIENT_REDUIT;
00180001
-----+-----+-----+-----+-----+-----+-----+
NUM_COMPTE  CODE_REGION  CODE_NATCPT  NOM_CLIENT  PREN_CLIENT  CODE_PROF  SIT_FA
-----+-----+-----+-----+-----+-----+-----+
001        01          20          DURAND      ALAIN        05          C
002        02          25          MARTIN     JEAN         10          M
003        03          30          BERNARD    CLAUDE        15          M
004        04          35          ROBERT     PAUL         20          C
005        01          40          RICHARD    MICHEL        25          C
006        02          20          PETIT       MARIE        30          M
007        03          25          ROUX        NADIA        05          M
008        04          30          DAVID       JULIEN        10          C
```

009	01	35	MOREAU	LUCIE	15	M
010	02	40	SIMON	AMELIE	20	C
011	03	20	LAURENT	THOMAS	25	C
012	04	25	LEROY	SOPHIE	30	M
013	01	30	RENAUD	PIERRE	05	C
014	02	35	BLANC	INES	10	M
015	03	40	BONNET	JULIE	15	C
016	04	20	FRANCOIS	ARTHUR	20	M
017	01	25	LEFEBVRE	EMILIE	25	C
018	02	30	MERCIER	CAMILLE	30	C
019	03	35	DUVAL	NICOLAS	10	M
020	04	40	GUYOT	PAULINE	05	C
DSNE610I NUMBER OF ROWS DISPLAYED IS 20						
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100						
-----+-----+-----+-----+-----+-----+-----+						
00190001						
-----+-----+-----+-----+-----+-----+-----+						
DSNE617I COMMIT_PERFORMED, SQLCODE IS 0						
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0						
-----+-----+-----+-----+-----+-----+-----+						
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72						
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 2						
DSNE621I NUMBER OF INPUT RECORDS READ IS 19						
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 57						
***** Bottom of Data *****						

Exercice 9 : Analyse multi-critères avec agrégations conditionnelles

Énoncé

Afficher, pour chaque région :

- Nombre total de clients
- Nombre de clients débiteurs
- Nombre de clients créditeurs
- Solde total débiteur
- Solde total créditeur

Mon travail

J'ai utilisé la technique CASE WHEN dans les fonctions d'agrégation pour calculer des sous-totaux conditionnels en une seule requête. Cela évite de faire plusieurs requêtes séparées.

Version simplifiée : Sans jointure, directement sur CLIENT avec GROUP BY CODE_REGION. Les alias entre guillemets permettent des noms descriptifs avec espaces.

Résolution

```
SELECT
    CODE_REGION,
    COUNT(*)
        AS "NB CLIENTS",
    SUM(CASE WHEN POS = 'DB' THEN 1 ELSE 0 END)
        AS "NB DEBITEURS",
    SUM(CASE WHEN POS = 'CR' THEN 1 ELSE 0 END)
        AS "NB CREDITEURS",
    SUM(CASE WHEN POS = 'DB' THEN SOLDE ELSE 0 END)
        AS "SOLDE TOTAL DEBITEURS",
    SUM(CASE WHEN POS = 'CR' THEN SOLDE ELSE 0 END)
        AS "SOLDE TOTAL CREDITEURS"
FROM CLIENT
GROUP BY CODE_REGION;
```

Captures d'écran

```
***** Top of Data *****
-----+-----+-----+-----+-----+-----+
-- P2 EXERCICE 9 : Analyse multi-criteres par region          00010002
                                                               00020000
                                                               00030003
SELECT          00040003
  CODE_REGION,          00050003
  COUNT(*)          00060003
    AS "NO DES CLIENTS",          00061005
  SUM(CASE WHEN POS = 'DB' THEN 1 ELSE 0 END)          00070003
    AS "NO DES DEBITEURS",          00071005
  SUM(CASE WHEN POS = 'CR' THEN 1 ELSE 0 END)          00080003
    AS "NO DES CREDITEURS",          00081004
  SUM(CASE WHEN POS = 'DB' THEN SOLDE ELSE 0 END)          00090003
    AS "SOLDE TOTAL DEBITEURS",          00091004
  SUM(CASE WHEN POS = 'CR' THEN SOLDE ELSE 0 END)          00100003
    AS "SOLDE TOTAL CREDITEURS"
FROM CLIENT          00110003

-----+-----+-----+-----+-----+-----+
GROUP_BY CODE_REGION          00120003
;
-----+-----+-----+-----+-----+-----+
CODE_REGION  NO DES CLIENTS  NO DES DEBITEURS  NO DES CREDITEURS  SOLDE TOTA
-----+-----+-----+-----+-----+-----+
01          5              1              4
02          5              2              3
03          5              2              3
04          5              3              2

DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
-----+-----+-----+-----+-----+-----+
                                         00190001
-----+-----+-----+-----+-----+-----+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 1
DSNE621I NUMBER OF INPUT RECORDS READ IS 18
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 37
***** Bottom of Data *****
```

Exercice 10 : Clients anormalement débiteurs par profession

Énoncé

Lister les clients débiteurs dont le solde est supérieur à la moyenne des soldes débiteurs de leur profession.

Mon travail

Cet exercice m'a demandé une sous-requête corrélée. La sous-requête calcule la moyenne des soldes débiteurs pour chaque profession, et la requête principale compare chaque client à cette moyenne.

Subtilité : Les soldes débiteurs étant négatifs, un client "plus débiteur" a un solde inférieur (ex: -450 < -200). D'où l'utilisation de < au lieu de >.

Résolution

```
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,
       P.CODE_PROF, P.LIB_PROF,
       C.SOLDE
  FROM CLIENT C
 INNER JOIN PROFESSI P ON C.CODE_PROF = P.CODE_PROF
 WHERE C.POS = 'DB'
   AND C.SOLDE < (SELECT AVG(C2.SOLDE)
                  FROM CLIENT C2
                 WHERE C2.POS = 'DB'
                   AND C2.CODE_PROF = C.CODE_PROF)
 ORDER BY C.CODE_PROF, C.SOLDE;
```

Note : La comparaison utilisé < car les soldes débiteurs sont négatifs. Un solde de -450 est "plus débiteur" qu'un solde de -200.

Captures d'écran

```
***** Top of Data *****
-----+-----+-----+-----+-----+
-- P2 EXERCICE 10 : Clients anormalement débiteurs par profession      00010001
                                                               00020000
-- Clients débiteurs avec solde < moyenne des débiteurs de leur professi 00030001
SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,
       P.CODE_PROF, P.LIB_PROF,
       C.SOLDE
FROM CLIENT C
INNER JOIN PROFESSI P ON C.CODE_PROF = P.CODE_PROF
WHERE C.POS = 'DB'
   AND C.SOLDE < (SELECT AVG(C2.SOLDE)
                   FROM CLIENT C2
                   WHERE C2.POS = 'DB'
                     AND C2.CODE_PROF = C.CODE_PROF)
ORDER BY C.CODE_PROF, C.SOLDE;                                         00040001
                                                               00050001
                                                               00060001
                                                               00061001
                                                               00070001
                                                               00071001
                                                               00080001
                                                               00081001
                                                               00090001
                                                               00091001
                                                               00100001
-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+
NUM_COMPTE NOM_CLIENT PREN_CLIENT CODE_PROF LIB_PROF          SOL
-----+-----+-----+-----+-----+
019      DUVAL      NICOLAS     10      INGENIEUR      -250.
002      MARTIN    JEAN        10      INGENIEUR      -200.
004      ROBERT    PAUL        20      COMMERCANT    -450.
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
-----+-----+-----+-----+-----+
                                                               00110001
-----+-----+-----+-----+-----+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 1
DSNE621I NUMBER OF INPUT RECORDS READ IS 15
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 33
```

Partie 3 : Programmation COBOL-DB2

Exercice 1 : Afficher la région Marseille

Énoncé

Écrire un programme COBOL-DB2 permettant d'afficher la région Marseille (02).

Mon travail

Premier programme COBOL-DB2 : j'ai utilisé SELECT INTO pour lire une seule ligne. La variable SQLCODE me permet de vérifier si la requête a réussi (0) ou échoué.

Utilisation de DCLGEN : Au lieu de déclarer manuellement les variables host, j'utilisé **EXEC SQL INCLUDE REGION END-EXEC** pour inclure les variables générées par l'utilitaire DCLGEN de DB2.

Avantages DCLGEN :

- Variables host générées automatiquement depuis la structure de la table
- Correspondance exacte avec les types de colonnes DB2
- Maintenabilité : si la table change, on régénère le DCLGEN

Résolution

Programme : AFFREG.cbl

```

WORKING-STORAGE SECTION.
* SQLCA pour gestion erreurs DB2
  EXEC SQL INCLUDE SQLCA END-EXEC.
* DCLGEN pour la table REGION
  EXEC SQL INCLUDE REGION END-EXEC.

PROCEDURE DIVISION.
1000-SELECT-REGION.
  EXEC SQL
    SELECT CODE_REGION, NOM_REGION
    INTO :CODE-REGION, :NOM-REGION
    FROM REGION
    WHERE CODE_REGION = '02'
  END-EXEC

  IF SQLCODE = 0
    DISPLAY 'CODE      : ' CODE-REGION
    DISPLAY 'NOM      : ' NOM-REGION
  ELSE
    DISPLAY 'ERREUR SQL - SQLCODE : ' SQLCODE
  END-IF.

```

Techniques utilisées :

- SELECT INTO (lecture d'une seule ligne)
- DCLGEN (variables host générées par DB2)

Captures d'écran

```
***** **** Top of Data ****
000100      IDENTIFICATION DIVISION.
000200      PROGRAM-ID. AFFREG.
000300
000400      *
000500      * P3 EXERCICE 1 : AFFICHER LA REGION MARSEILLE (02)
000600      *
000700
000800      ENVIRONMENT DIVISION.
000900
001000      DATA DIVISION.
001100      WORKING-STORAGE SECTION.
001200
001300      * SQLCA POUR GESTION ERREURS DB2
001400          EXEC SQL
001500              INCLUDE SQLCA
001600          END-EXEC.

001700
001800      * DCLGEN POUR LA TABLE REGION
001900          EXEC SQL
002000              INCLUDE REGION
002100          END-EXEC.

002200
002300      PROCEDURE DIVISION.
002400      0000-PRINCIPAL.
002500          PERFORM 1000-SELECT-REGION
002600          PERFORM 9000-FIN
002700          STOP RUN.

002800
002900      1000-SELECT-REGION.
003000          EXEC SQL
003100              SELECT CODE_REGION, NOM_REGION
003200                  INTO :CODE-REGION, :NOM-REGION
003300              FROM REGION
```

```

003400      WHERE CODE_REGION = '02'
003500      END-EXEC
003600
003700      IF SQLCODE = 0
003800          DISPLAY '====='
003900          DISPLAY 'REGION MARSEILLE'
004000          DISPLAY '====='
004100          DISPLAY 'CODE    : ' CODE-REGION
004200          DISPLAY 'NOM     : ' NOM-REGION
004300          DISPLAY '====='
004400      ELSE
004500          DISPLAY 'ERREUR SQL - SQLCODE : ' SQLCODE
004600      END-IF.
004700
004800      9000-FIN.
004900      DISPLAY 'FIN DU PROGRAMME AFFREG'.
005000
=====
```

REGION MARSEILLE

=====
CODE : 02
NOM : MARSEILLE
=====

FIN DU PROGRAMME AFFREG

***** BOTTOM OF DATA *****

Exercice 2 : Insérer un nouveau client

Énoncé

Écrire un programme COBOL-DB2 permettant d'insérer un nouveau client dans la table CLIENT.

Mon travail

J'ai utilisé INSERT INTO avec des variables host. Les données du client sont lues depuis SYSIN via des ACCEPT (une ligne par champ). J'ai ajouté la gestion transactionnelle avec COMMIT en cas de succès et ROLLBACK en cas d'erreur.

JCL requis : Les données du client sont passées via SYSIN (12 lignes) :

```
//SYSIN DD *
021
01
25
DUPONT
MARC
1995-06-15
M
10
C
```

```
25 RUE NEUVE  
1800.00  
CR  
/*
```

Résolution

Programme : INSCLI.cbl

```
1000-LIRE-DONNEES.  
* Lecture des données depuis SYSIN (JCL In-Stream)  
    ACCEPT WS-NUM-COMPTE  
    ACCEPT WS-CODE-REGION  
    ACCEPT WS-CODE-NATCPT  
    ACCEPT WS-NOM-CLIENT  
    ACCEPT WS-PREN-CLIENT  
    ACCEPT WS-DATE-NAIS  
    ACCEPT WS-SEXE  
    ACCEPT WS-CODE-PROF  
    ACCEPT WS-SIT-FAM  
    ACCEPT WS-ADRESSE  
    ACCEPT WS-SOLDE-IN  
    ACCEPT WS-POS  
  
* Conversion du solde (texte -> numerique)  
    COMPUTE WS-SOLDE = FUNCTION NUMVAL(WS-SOLDE-IN).  
  
2000-INSERT-CLIENT.  
    EXEC SQL  
        INSERT INTO CLIENT (...)  
        VALUES (:WS-NUM-COMPTE, :WS-CODE-REGION, ...)  
    END-EXEC  
  
    IF SQLCODE = 0  
        EXEC SQL COMMIT END-EXEC  
    ELSE  
        EXEC SQL ROLLBACK END-EXEC  
    END-IF.
```

Techniques utilisées :

- ACCEPT pour lire les données depuis SYSIN
- FUNCTION NUMVAL pour convertir le solde texte en numérique
- COMMIT/ROLLBACK pour la gestion transactionnelle

Captures d'écran

```
***** **** Top of Data *****  
000100      IDENTIFICATION DIVISION.  
000200      PROGRAM-ID. INSCLI.  
000300      *-----  
000400      * P3 EXERCICE 2 : INSERER UN NOUVEAU CLIENT  
000500      *-----  
000600  
000700      DATA DIVISION.  
000800      WORKING-STORAGE SECTION.  
000900  
001000      * VARIABLES HOST POUR DB2  
001100      01 WS-NUM-COMPTE    PIC X(03).  
001200      01 WS-CODE-REGION   PIC X(02).  
001300      01 WS-CODE-NATCPT   PIC X(02).  
001400      01 WS-NOM-CLIENT    PIC X(10).  
001500      01 WS-PREN-CLIENT   PIC X(10).  
001600      01 WS-DATE-NAIS     PIC X(10).  
001700      01 WS-SEXE          PIC X(01).  
001800      01 WS-CODE-PROF     PIC X(02).  
001900      01 WS-SIT-FAM       PIC X(01).  
002000      01 WS-ADRESSE        PIC X(20).  
002100      01 WS-SOLDE         PIC S9(8)V99 COMP-3.  
002200      01 WS-POS           PIC X(02).  
002300  
002400      * VARIABLE POUR SAISIE DU SOLDE  
002500      01 WS-SOLDE-IN      PIC X(10).  
002600  
002700      * SQLCA POUR GESTION ERREURS DB2  
002800      EXEC SQL  
002900      INCLUDE SQLCA  
003000      END-EXEC.  
003100  
003200      PROCEDURE DIVISION.  
003300      0000-PRINCIPAL.
```

```

003400      PERFORM 1000-LIRE-DONNEES
003500      PERFORM 2000-INSERT-CLIENT
003600      PERFORM 9000-FIN
003700      STOP RUN.

003800
003900      1000-LIRE-DONNEES.
004000      * LECTURE DES DONNEES DEPUIS SYSIN (JCL IN-STREAM)
004100          ACCEPT WS-NUM-COMpte
004200          ACCEPT WS-CODE-REGION
004300          ACCEPT WS-CODE-NATCPT
004400          ACCEPT WS-NOM-CLIENT
004500          ACCEPT WS-PREN-CLIENT
004600          ACCEPT WS-DATE-NAIS
004700          ACCEPT WS-SEXE
004800          ACCEPT WS-CODE-PROF
004900          ACCEPT WS-SIT-FAM
005000          ACCEPT WS-ADRESSE
005100          ACCEPT WS-SOLDE-IN
005200          ACCEPT WS-POS
005300
005400      * CONVERSION DU SOLDE (TEXTE -> NUMERIQUE)
005500          COMPUTE WS-SOLDE = FUNCTION NUMVAL(WS-SOLDE-IN)
005600
005700          DISPLAY 'DONNEES LUES DEPUIS SYSIN'
005800          DISPLAY 'NUM COMPTE : ' WS-NUM-COMpte
005900          DISPLAY 'NOM           : ' WS-NOM-CLIENT.
006000
006100      2000-INSERT-CLIENT.
006200          EXEC SQL
006300              INSERT INTO CLIENT
006400                  (NUM_COMPTE, CODE_REGION, CODE_NATCPT,
006500                      NOM_CLIENT, PREN_CLIENT, DATE_NAIS,
006600                      SEXE, CODE_PROF, SIT_FAM,
006700                      ADRESSE, SOLDE, POS)

```

```

006800          VALUES
006900          (:WS-NUM-COMpte, :WS-CODE-REGION, :WS-CODE-NATCPT,
007000          :WS-NOM-CLIENT, :WS-PREN-CLIENT, :WS-DATE-NAIS,
007100          :WS-SEXE, :WS-CODE-PROF, :WS-SIT-FAM,
007200          :WS-ADRESSE, :WS-SOLDE, :WS-POS)
007300      END-EXEC
007400
007500      IF SQLCODE = 0
007600          DISPLAY 'CLIENT INSERE AVEC SUCCES'
007700          DISPLAY 'NUM COMPTE : ' WS-NUM-COMpte
007800          DISPLAY 'NOM           : ' WS-NOM-CLIENT
007900          EXEC SQL COMMIT END-EXEC
008000      ELSE
008100          DISPLAY 'ERREUR INSERTION - SQLCODE : ' SQLCODE
008200          EXEC SQL ROLLBACK END-EXEC
008300      END-IF.
008400

```

```

008500      9000-FIN.
008600      DISPLAY 'FIN DU PROGRAMME INSCLI'.
***** **** Bottom of Data ****

```

```

***** **** Top of Data ****
000100 //EXECPG02 JOB (2025), 'ROCHA', MSGCLASS=X, CLASS=A,
000200 //          MSGLEVEL=(1,1), NOTIFY=&SYSUID
000300 //JOBLIB    DD DSN=DSNA10.SDSNLOAD, DISP=SHR
000400 //          DD DSN=FORM1112.FINANCE.LOAD, DISP=SHR
000500 //          DD DSN=FORM1112.FINANCE.DBRM, DISP=SHR
000700 //OUT1      OUTPUT OUTDISP=(KEEP,KEEP)
000710 //STEP1     EXEC PGM=IKJEFT01, DYNAMNBR=20
000800 //SYSTSPRT  DD SYSOUT=*
000900 //SYSPRINT   DD SYSOUT=T, OUTPUT=*.OUT1
001000 //SYSPRINT   DD DUMMY
001100 //SYSTSIN   DD *
001200          DSN SYSTEM(DBAG)
001300          RUN PROG(INSCLI) PLAN(INSCLI)
001400          END
001500 /*
003991 //SYSIN    DD *

```

```

003992 021
003993 03
003994 25
003995 SILVA
003996 ROBERTO
003997 1999-11-22
003998 M
003999 05
004000 C
004001 STRASBOURG
004002 3000.00
004003 CR
004004 /*
004010 //
004100
***** ***** Bottom of Data *****

```

DONNEES LUES DEPUIS SYSIN

NUM COMPTE : 021

NOM : SILVA

CLIENT INSERE AVEC SUCCES

NUM COMPTE : 021

NOM : SILVA

FIN DU PROGRAMME INSCLI

```
***** BOTTOM OF DATA *****
```

Exercice 3 : Afficher tous les clients de Marseille

Énoncé

Écrire un programme COBOL-DB2 permettant d'afficher tous les clients de la région Marseille (02).

Mon travail

Pour lire plusieurs lignes, j'ai utilisé un CURSOR. Le cycle complet est : DECLARE (définition), OPEN (ouverture), FETCH en boucle (lecture), CLOSE (fermeture).

Gestion fin de fichier : SQLCODE = 100 indique qu'il n'y a plus de lignes à lire.

Résolution

Programme : AFFCLI.cbl

```

IDENTIFICATION DIVISION.
PROGRAM-ID. AFFCLI.

DATA DIVISION.
WORKING-STORAGE SECTION.
01 WS-NUM-COMPTE      PIC X(03).
01 WS-NOM-CLIENT       PIC X(10).

```

```
01 WS-FIN-CURSOR      PIC 9(01) VALUE 0.
EXEC SQL INCLUDE SQLCA END-EXEC.

EXEC SQL
    DECLARE C-CLIENTS CURSOR FOR
        SELECT NUM_COMPTE, NOM_CLIENT, ...
        FROM CLIENT
        WHERE CODE_REGION = '02'
        ORDER BY NUM_COMPTE
    END-EXEC.

PROCEDURE DIVISION.
EXEC SQL OPEN C-CLIENTS END-EXEC
PERFORM UNTIL WS-FIN-CURSOR = 1
    EXEC SQL
        FETCH C-CLIENTS INTO :WS-NUM-COMPTE, ...
    END-EXEC
    IF SQLCODE = 0
        DISPLAY WS-NUM-COMPTE ' ' WS-NOM-CLIENT
    ELSE
        MOVE 1 TO WS-FIN-CURSOR
    END-IF
END-PERFORM
EXEC SQL CLOSE C-CLIENTS END-EXEC
STOP RUN.
```

Technique utilisée : CURSOR (DECLARE, OPEN, FETCH, CLOSE) pour lecture multiple

Captures d'écran

```
***** **** Top of Data ****
000100      IDENTIFICATION DIVISION.
000200      PROGRAM-ID. AFFCLI.
000300      *
000400      * P3 EXERCICE 3 : AFFICHER TOUS LES CLIENTS DE MARSEILLE
000500      *
000600
000700      ENVIRONMENT DIVISION.
000800
000900      DATA DIVISION.
001000      WORKING-STORAGE SECTION.
001100
001200      * VARIABLES HOST POUR DB2
001300      01 WS-NUM-COMPTE      PIC X(03).
001400      01 WS-NOM-CLIENT      PIC X(10).
001500      01 WS-PREN-CLIENT      PIC X(10).
001600      01 WS-SOLDE          PIC S9(8)V99 COMP-3.
001700      01 WS-POS            PIC X(02).
001800
001900      * VARIABLE DE TRAVAIL
002000      01 WS-SOLDE-ED        PIC -ZZZ,ZZ9.99.
002100      01 WS-COMPTEUR        PIC 9(03) VALUE 0.
002200      01 WS-FIN-CURSOR       PIC 9(01) VALUE 0.
002300
002400      * SQLCA POUR GESTION ERREURS DB2
002500      EXEC SQL
002600          INCLUDE SQLCA
002700          END-EXEC.
002800      * DECLARATION DU CURSEUR
002900      EXEC SQL
003000          DECLARE C-CLIENTS CURSOR FOR
003100              SELECT NUM_COMPTE, NOM_CLIENT, PREN_CLIENT,
003200                  SOLDE, POS
003300              FROM CLIENT
003400          WHERE CODE_REGION = '02'
003500          ORDER BY NUM_COMPTE
003600          END-EXEC.
003700
003800      PROCEDURE DIVISION.
003900      0000-PRINCIPAL.
004000          PERFORM 1000-OUVRIR-CURSOR
004100          PERFORM 2000-AFFICHER-ENTETE
004200          PERFORM 3000-LIRE-CLIENTS
004300              UNTIL WS-FIN-CURSOR = 1
004400          PERFORM 4000-FERMER-CURSOR
004500          PERFORM 9000-FIN
004600          STOP RUN.
004700
004800      1000-OUVRIR-CURSOR.
004900      EXEC SQL
005000          OPEN C-CLIENTS
```

```
005100      END-EXEC
005200      IF SQLCODE NOT = 0
005300          DISPLAY 'ERREUR OUVERTURE CURSOR : ' SQLCODE
005400          MOVE 1 TO WS-FIN-CURSOR
005500      END-IF.
005600
005700      2000-AFFICHER-ENTETE.
005800          DISPLAY '=====
005900          DISPLAY 'CLIENTS DE LA REGION MARSEILLE (02)'
006000          DISPLAY '=====
006100          DISPLAY 'NUM      NOM          PRENOM      SOLDE      POS'.
006200
006300      3000-LIRE-CLIENTS.
006400          EXEC SQL
006500              FETCH C-CLIENTS
006600              INTO :WS-NUM-COMPTE, :WS-NOM-CLIENT,
006700                  :WS-PREN-CLIENT, :WS-SOLDE, :WS-POS
006800      END-EXEC
006900      EVALUATE SQLCODE
007000          WHEN 0
007100              ADD 1 TO WS-COMPTEUR
007200              MOVE WS-SOLDE TO WS-SOLDE-ED
007300              DISPLAY WS-NUM-COMPTE   '
007400                  WS-NOM-CLIENT   '
007500                  WS-PREN-CLIENT   '
007600                  WS-SOLDE-ED   '
007700                  WS-POS
007800          WHEN 100
007900              MOVE 1 TO WS-FIN-CURSOR
008000          WHEN OTHER
008100              DISPLAY 'ERREUR FETCH : ' SQLCODE
008200              MOVE 1 TO WS-FIN-CURSOR
008300      END-EVALUATE.
008400
```

```

008500      4000-FERMER-CURSOR.
008600      EXEC SQL
008700          CLOSE C-CLIENTS
008800      END-EXEC
008900      DISPLAY '=====
009000      DISPLAY 'TOTAL CLIENTS : ' WS-COMPTEUR.
009100
009200      9000-FIN.
009300      DISPLAY 'FIN DU PROGRAMME AFFCLI'.
009400
***** ***** Bottom of Data *****

```

```

=====
CLIENTS DE LA REGION MARSEILLE (02)
=====
NUM  NOM        PRENOM      SOLDE      POS
002 MARTIN    JEAN        - 200.00 DB
006 PETIT      MARIE       1,200.00 CR
010 SIMON      AMELIE     2,100.00 CR
014 BLANC      INES        - 100.00 DB
018 MERCIER    CAMILLE    750.00 CR
=====
```

```

TOTAL CLIENTS : 005
FIN DU PROGRAMME AFFCLI
***** BOTTOM OF DATA *****

```

Exercice 4 : Mise à jour d'un client

Énoncé

Écrire un programme COBOL-DB2 qui permet de mettre à jour l'adresse, le solde et la position d'un client existant.

Mon travail

J'ai utilisé UPDATE avec une clause WHERE pour cibler un client spécifique (005). Comme pour l'INSERT, j'applique COMMIT après vérification du SQLCODE.

Test : J'ai vérifié avec un SELECT avant et après la mise à jour pour confirmer les changements.

Résolution

Programme : MAJCLI.cbl

```

WORKING-STORAGE SECTION.
* Variables host pour DB2
 01 WS-NUM-COMPTE      PIC X(03).
 01 WS-ADRESSE         PIC X(20).
 01 WS-SOLDE           PIC S9(8)V99 COMP-3.
 01 WS-POS              PIC X(02).
* Variable pour saisie du solde

```

```

01 WS-SOLDE-IN      PIC X(10).

PROCEDURE DIVISION.
1000-LIRE-DONNEES.
* Lecture des données depuis SYSIN (JCL In-Stream)
  ACCEPT WS-NUM-COMPTE
  ACCEPT WS-ADRESSE
  ACCEPT WS-SOLDE-IN
  ACCEPT WS-POS
* Conversion du solde (texte -> numerique)
  COMPUTE WS-SOLDE = FUNCTION NUMVAL(WS-SOLDE-IN)

2000-UPDATE-CLIENT.
  EXEC SQL
    UPDATE CLIENT
    SET ADRESSE = :WS-ADRESSE,
        SOLDE = :WS-SOLDE,
        POS = :WS-POS
    WHERE NUM_COMPTE = :WS-NUM-COMPTE
  END-EXEC

  IF SQLCODE = 0
    DISPLAY 'CLIENT MIS A JOUR'
    EXEC SQL COMMIT END-EXEC
  END-IF.

```

JCL d'exécution avec SYSIN :

```

//SYSIN    DD *
005
20 AVENUE FOCH
2800.00
CR
/*

```

Technique utilisée : UPDATE avec clause WHERE + COMMIT + ACCEPT pour données dynamiques

Captures d'écran

```
***** **** Top of Data *****  
000100      IDENTIFICATION DIVISION.  
000200      PROGRAM-ID. MAJCLI.  
000300      *-----  
000400      * P3 EXERCICE 4 : MISE A JOUR CLIENT (ADRESSE, SOLDE, POS)  
000500      *-----  
000600  
000700      ENVIRONMENT DIVISION.  
000800  
000900      DATA DIVISION.  
001000      WORKING-STORAGE SECTION.  
001100  
001200      * VARIABLES HOST POUR DB2  
001300      01 WS-NUM-COMPTE    PIC X(03).  
001400      01 WS-ADRESSE       PIC X(20).  
001500      01 WS-SOLDE        PIC S9(8)V99 COMP-3.  
001600      01 WS-POS          PIC X(02).  
  
001700  
001800      * VARIABLE POUR SAISIE DU SOLDE  
001900      01 WS-SOLDE-IN     PIC X(10).  
002000  
002100      * SQLCA POUR GESTION ERREURS DB2  
002200      EXEC SQL  
002300      INCLUDE SQLCA  
002400      END-EXEC.  
002500  
002600      PROCEDURE DIVISION.  
002700      0000-PRINCIPAL.  
002800      PERFORM 1000-LIRE-DONNEES  
002900      PERFORM 2000-UPDATE-CLIENT  
003000      PERFORM 9000-FIN  
003100      STOP RUN.  
003200  
003300      1000-LIRE-DONNEES.
```

```

003400      * LECTURE DES DONNEES DEPUIS SYSIN (JCL IN-STREAM)
003500          ACCEPT WS-NUM-COMpte
003600          ACCEPT WS-ADRESSE
003700          ACCEPT WS-SOLDE-IN
003800          ACCEPT WS-POS
003810
003900      * CONVERSION DU SOLDE (TEXTE -> NUMERIQUE)
004000          COMPUTE WS-SOLDE = FUNCTION NUMVAL(WS-SOLDE-IN)
004100
004200          DISPLAY 'DONNEES LUES DEPUIS SYSIN'
004300          DISPLAY 'NUM COMPTE : ' WS-NUM-COMPTE
004400          DISPLAY 'NV ADRESSE : ' WS-ADRESSE
004500          DISPLAY 'NV SOLDE : ' WS-SOLDE
004600          DISPLAY 'NV POS : ' WS-POS.
004700
004800      2000-UPDATE-CLIENT.
004900          EXEC SQL

```

```

005000          UPDATE CLIENT
005100              SET ADRESSE = :WS-ADRESSE,
005200                  SOLDE = :WS-SOLDE,
005300                  POS = :WS-POS
005400          WHERE NUM_COMPTE = :WS-NUM-COMPTE
005500          END-EXEC
005600
005700          EVALUATE SQLCODE
005800          WHEN 0
005900              DISPLAY 'CLIENT MIS A JOUR AVEC SUCCES'
006000              DISPLAY 'NUM COMPTE : ' WS-NUM-COMPTE
006100              DISPLAY 'NV ADRESSE : ' WS-ADRESSE
006200              DISPLAY 'NV SOLDE : ' WS-SOLDE
006300              DISPLAY 'NV POS : ' WS-POS
006400          EXEC SQL COMMIT END-EXEC
006500          WHEN 100
006600              DISPLAY 'CLIENT NON TROUVE : ' WS-NUM-COMPTE

```

```

006610          WHEN OTHER
006620              DISPLAY 'ERREUR UPDATE - SQLCODE : ' SQLCODE
006630              EXEC SQL ROLLBACK END-EXEC
006640          END-EVALUATE.
006641
006650      9000-FIN.
006660          DISPLAY 'FIN DU PROGRAMME MAJCLI'.
006670
***** ***** Bottom of Data *****

```

```
***** **** Top of Data ****
000100 //DDINSTRM JOB (2025), 'ROCHA', MSGCLASS=X, CLASS=A,
000200 // MSGLEVEL=(1,1), NOTIFY=&SYSUID
000300 //JOBLIB DD DSN=DSNA10.SDSNLOAD,DISP=SHR
000400 // DD DSN=FORM1112.FINANCE.LOAD,DISP=SHR
000500 // DD DSN=FORM1112.FINANCE.DBRM,DISP=SHR
000600 //OUT1 OUTPUT OUTDISP=(KEEP,KEEP)
000700 //STEP1 EXEC PGM=IKJEFT01,DYNAMNBR=20
000800 //SYSTSPRT DD SYSOUT=*
000900 //SYSPRINT DD SYSOUT=T,OUTPUT=*.OUT1
001000 //SYSPRINT DD DUMMY
001100 //SYSTSIN DD *
001200      DSN SYSTEM(DBAG)
001300      RUN PROG(MAJCLI) PLAN(MAJCLI)
001400      END
001500 /*
001600 //SYSIN   DD *
001700 005
001800 20 AVENUE FOCH
001900 2800.00
002000 CR
002100 /*
002200 //
***** **** Bottom of Data ****
```

DONNEES LUES DEPUIS SYSIN
NUM COMPTE : 005
NV ADRESSE : 20 AVENUE FOCH
NV SOLDE : 0000280000
NV POS : CR
CLIENT MIS A JOUR AVEC SUCCES
NUM COMPTE : 005
NV ADRESSE : 20 AVENUE FOCH
NV SOLDE : 0000280000
NV POS : CR
FIN DU PROGRAMME MAJCLI

***** BOTTOM OF DATA *****

Exercice 5 : Liste avec ruptures

Énoncé

Écrire un programme COBOL-DB2 qui affiché la liste des clients triés par region puis par profession. Pour chaque changement, affichér un titre de rupture.

Mon travail

J'ai implémenté les ruptures de contrôle en conservant les valeurs précédentes de CODE_REGION et CODE_PROF. A chaque FETCH, je compare les nouvelles valeurs avec les précédentes pour détecter les changements.

Logique : Si la region change, j'affiché le titre region ET je reinitialise la rupture profession (car on change de groupe).

Résolution

Programme : LSTRUPT.cbl

```
WORKING-STORAGE SECTION.  
01 WS-PREC-REGION      PIC X(02) VALUE SPACES.  
01 WS-PREC-PROF        PIC X(02) VALUE SPACES.  
  
PROCEDURE DIVISION.  
2100-VERIFIER-RUPTURES.  
    IF WS-CODE-REGION NOT = WS-PREC-REGION  
        DISPLAY '==> REGION : ' WS-NOM-REGION ' ==<'  
        MOVE WS-CODE-REGION TO WS-PREC-REGION  
        MOVE SPACES TO WS-PREC-PROF  
    END-IF  
  
    IF WS-CODE-PROF NOT = WS-PREC-PROF  
        DISPLAY '<---- PROFESSION : ' WS-LIB-PROF ' ----'  
        MOVE WS-CODE-PROF TO WS-PREC-PROF  
    END-IF.
```

Technique utilisée : Variables de rupture pour détecter les changements de groupe

Captures d'écran

```
***** **** Top of Data ****
000100      IDENTIFICATION DIVISION.
000200      PROGRAM-ID. LSTRUPT.
000300      *
000400      * P3 EXERCICE 5 : LISTE CLIENTS TRIES AVEC RUPTURES
000500      *
000600
000700      ENVIRONMENT DIVISION.
000800
000900      DATA DIVISION.
001000      WORKING-STORAGE SECTION.
001100      * VARIABLES HOST POUR DB2
001200      01 WS-NUM-COMPTE      PIC X(03).
001300      01 WS-NOM-CLIENT     PIC X(10).
001400      01 WS-PREN-CLIENT    PIC X(10).
001500      01 WS-CODE-REGION    PIC X(02).

001600      01 WS-NOM-REGION    PIC X(15).
001700      01 WS-CODE-PROF     PIC X(02).
001800      01 WS-LIB-PROF      PIC X(20).
001900      01 WS-SOLDE        PIC S9(8)V99 COMP-3.
002000      01 WS-POS          PIC X(02).

002100      * VARIABLES DE RUPTURE
002210      01 WS-PREC-REGION   PIC X(02) VALUE SPACES.
002220      01 WS-PREC-PROF     PIC X(02) VALUE SPACES.

002300
002400      * VARIABLES DE TRAVAIL
002500      01 WS-SOLDE-ED     PIC -ZZZ,ZZ9.99.
002600      01 WS-FIN-CURSOR   PIC 9(01) VALUE 0.

002700
002800      * SQLCA POUR GESTION ERREURS DB2
002810      EXEC SQL
002820      INCLUDE SQLCA

002830      END-EXEC.

002840
002900      * DECLARATION DU CURSEUR
003000      EXEC SQL
003100      DECLARE C-CLIENTS CURSOR FOR
003200      SELECT C.NUM_COMPTE, C.NOM_CLIENT, C.PREN_CLIENT,
003300                  R.CODE_REGION, R.NOM_REGION,
003400                  P.CODE_PROF, P.LIB_PROF,
003500                  C.SOLDE, C.POS
003600      FROM CLIENT C
003700      INNER JOIN REGION R ON C.CODE_REGION = R.CODE_REGION
003710      INNER JOIN PROFESSI P ON C.CODE_PROF = P.CODE_PROF
003720      ORDER BY C.CODE_REGION, C.CODE_PROF, C.NUM_COMPTE
003730      END-EXEC.

003800
003900      PROCEDURE DIVISION.
004000      0000-PRINCIPAL.
```

```
004100      PERFORM 1000-OUVRIR-CURSOR
004200      PERFORM 2000-TRAITER-CLIENTS
004300          UNTIL WS-FIN-CURSOR = 1
004400      PERFORM 3000-FERMER-CURSOR
004500      PERFORM 9000-FIN
004600      STOP RUN.
004700
004800      1000-OUVRIR-CURSOR.
004900      EXEC SQL
005000          OPEN C-CLIENTS
005100      END-EXEC
005200      IF SQLCODE NOT = 0
005300          DISPLAY 'ERREUR OUVERTURE CURSOR : ' SQLCODE
005400          MOVE 1 TO WS-FIN-CURSOR
005500      END-IF.
006011
006020      2000-TRAITER-CLIENTS.

006200      EXEC SQL
006300          FETCH C-CLIENTS
006400              INTO :WS-NUM-COMPTE, :WS-NOM-CLIENT, :WS-PREN-CLIENT,
006500                  :WS-CODE-REGION, :WS-NOM-REGION,
006600                  :WS-CODE-PROF, :WS-LIB-PROF,
006700                  :WS-SOLDE, :WS-POS
006800      END-EXEC
006900
006910      EVALUATE SQLCODE
007000          WHEN 0
007100              PERFORM 2100-VERIFIER-RUPTURES
007200              PERFORM 2200-AFFICHER-CLIENT
007300          WHEN 100
007400              MOVE 1 TO WS-FIN-CURSOR
007500          WHEN OTHER
007900              DISPLAY 'ERREUR FETCH : ' SQLCODE
007910              MOVE 1 TO WS-FIN-CURSOR
```

```
007920           END-EVALUATE.  
007930  
007940           2100-VERIFIER-RUPTURES.  
007950           * RUPTURE SUR REGION  
007960               IF WS-CODE-REGION NOT = WS-PREC-REGION  
007970                   DISPLAY ''  
007980                   DISPLAY '==> REGION : ' WS-NOM-REGION ' =='  
007990               MOVE WS-CODE-REGION TO WS-PREC-REGION  
007991               MOVE SPACES TO WS-PREC-PROF  
007992           END-IF  
007993  
007994           * RUPTURE SUR PROFESSION  
007995               IF WS-CODE-PROF NOT = WS-PREC-PROF  
008000                   DISPLAY '---- PROFESSION : ' WS-LIB-PROF ' ----'  
008100                   MOVE WS-CODE-PROF TO WS-PREC-PROF  
008200           END-IF.  
008210  
  
008300           2200-AFFICHER-CLIENT.  
008400               MOVE WS-SOLDE TO WS-SOLDE-ED  
008410               DISPLAY '' WS-NUM-COMpte ''  
008420                   WS-NOM-CLIENT ''  
008430                   WS-PREN-CLIENT ''  
008440                   WS-SOLDE-ED ''  
008450                   WS-POS.  
008460  
008470           3000-FERMER-CURSOR.  
008480               EXEC SQL  
008490                   CLOSE C-CLIENTS  
008600           END-EXEC.  
008700  
008710           9000-FIN.  
008720               DISPLAY ''  
008730               DISPLAY 'FIN DU PROGRAMME LSTRUPT'.  
008740
```

```

==== REGION : PARIS =====
---- PROFESSION : MEDECIN -----
 001 DURAND    ALAIN      1,500.00 CR
 013 RENAUD    PIERRE     1,750.00 CR
---- PROFESSION : COMPTABLE -----
 009 MOREAU    LUCIE      - 300.00 DB
---- PROFESSION : FONCTIONNAIRE -----
 005 RICHARD   MICHEL     2,800.00 CR
 017 LEFEBVRE   EMILIE     350.00 CR

==== REGION : MARSEILLE =====
---- PROFESSION : INGENIEUR -----
 002 MARTIN   JEAN       - 200.00 DB
 014 BLANC    INES       - 100.00 DB
---- PROFESSION : COMMERCANT -----
 010 SIMON    AMELIE     2,100.00 CR
---- PROFESSION : PRIVEE -----
 006 PETIT    MARIE     1,200.00 CR

 018 MERCIER   CAMILLE    750.00 CR

==== REGION : LYON =====
---- PROFESSION : MEDECIN -----
 007 ROUX     NADIA      - 600.00 DB
 021 SILVA    ROBERTO    3,000.00 CR
---- PROFESSION : INGENIEUR -----
 019 DUVAL    NICOLAS    - 250.00 DB
---- PROFESSION : COMPTABLE -----
 003 BERNARD  CLAUDE     800.00 CR
 015 BONNET   JULIE      950.00 CR
---- PROFESSION : FONCTIONNAIRE -----
 011 LAURENT  THOMAS     600.00 CR

==== REGION : LILLE =====
---- PROFESSION : MEDECIN -----
 020 GUYOT    PAULINE     1,100.00 CR
---- PROFESSION : INGENIEUR -----
 008 DAVID    JULIEN     950.00 CR

---- PROFESSION : COMMERCANT -----
 004 ROBERT   PAUL       - 450.00 DB
 016 FRANCOIS ARTHUR     500.00 DB
---- PROFESSION : PRIVEE -----
 012 LEROY    SOPHIE     - 150.00 DB

```

FIN DU PROGRAMME LSTRUCT

***** BOTTOM OF DATA *****

Exercice 6 : Statistiques DB/CR

Énoncé

Écrire un programme COBOL-DB2 permettant de calculer le montant général et la moyenne des comptes débiteurs et créditeurs.

Mon travail

J'ai fait deux SELECT INTO avec les fonctions SUM, AVG et COUNT : un pour les débiteurs (POS = 'DB') et un pour les créditeurs (POS = 'CR').

Affichage : Les variables d'édition (PIC ZZZ,ZZ9.99) permettent de formater les montants avec séparateurs et décimales.

Résolution

Programme : STATCLI.cbl

```

PROCEDURE DIVISION.
  EXEC SQL
    SELECT SUM(SOLDE), AVG(SOLDE), COUNT(*)
    INTO :WS-TOTAL-DB, :WS-MOYENNE-DB, :WS-COUNT-DB
    FROM CLIENT WHERE POS = 'DB'
  END-EXEC

  EXEC SQL
    SELECT SUM(SOLDE), AVG(SOLDE), COUNT(*)
    INTO :WS-TOTAL-CR, :WS-MOYENNE-CR, :WS-COUNT-CR
    FROM CLIENT WHERE POS = 'CR'
  END-EXEC

  DISPLAY '==> STATISTIQUES DES CLIENTS ==>'
  DISPLAY '--- DEBITEURS ---'
  DISPLAY 'TOTAL : ' WS-TOTAL-DB
  DISPLAY 'MOYENNE : ' WS-MOYENNE-DB
  DISPLAY '--- CREDITEURS ---'
  DISPLAY 'TOTAL : ' WS-TOTAL-CR
  DISPLAY 'MOYENNE : ' WS-MOYENNE-CR.

```

Technique utilisée : Fonctions d'agrégation SQL (SUM, AVG, COUNT)

Captures d'écran

```
***** **** Top of Data *****  
000100      IDENTIFICATION DIVISION.  
000200      PROGRAM-ID. STATCLI.  
000300      *-----  
000400      * P3 EXERCICE 6 : STATISTIQUES DB/CR (TOTAL ET MOYENNE)  
000500      *-----  
000600  
000700      ENVIRONMENT DIVISION.  
000800  
000900      DATA DIVISION.  
001000      WORKING-STORAGE SECTION.  
001100  
001200      * VARIABLES HOST POUR DB2 - DEBITEURS  
001300      01 WS-TOTAL-DB      PIC S9(10)V99 COMP-3.  
001400      01 WS-MOYENNE-DB    PIC S9(10)V99 COMP-3.  
001500      01 WS-COUNT-DB     PIC S9(05) COMP.  
001600  
001700      * VARIABLES HOST POUR DB2 - CREDITEURS  
001800      01 WS-TOTAL-CR     PIC S9(10)V99 COMP-3.  
001900      01 WS-MOYENNE-CR   PIC S9(10)V99 COMP-3.  
002000      01 WS-COUNT-CR     PIC S9(05) COMP.  
002100  
002200      * VARIABLES D'EDITION  
002300      01 WS-TOTAL-ED     PIC -ZZZ,ZZZ,ZZ9.99.  
002400      01 WS-MOYENNE-ED   PIC -ZZZ,ZZZ,ZZ9.99.  
002500  
002600      * SQLCA POUR GESTION ERREURS DB2  
002700          EXEC SQL  
002800          INCLUDE SQLCA  
002900          END-EXEC.  
003000  
003100      PROCEDURE DIVISION.  
003200      0000-PRINCIPAL.  
003300          PERFORM 1000-STATS-DEBITEURS
```

```
003400      PERFORM 2000-STATS-CREDITEURS
003500      PERFORM 3000-AFFICHER-RESULTATS
003600      PERFORM 9000-FIN
003700      STOP RUN.

003800
003900      1000-STATS-DEBITEURS.
004000      EXEC SQL
004100          SELECT SUM(SOLDE), AVG(SOLDE), COUNT(*)
004200          INTO :WS-TOTAL-DB, :WS-MOYENNE-DB, :WS-COUNT-DB
004300          FROM CLIENT
004400          WHERE POS = 'DB'
004500      END-EXEC
004600
004700      IF SQLCODE NOT = 0
004800          DISPLAY 'ERREUR STATS DEBITEURS : ' SQLCODE
004900      END-IF.

005000
```

```
005100      2000-STATS-CREDITEURS.
005200      EXEC SQL
005300          SELECT SUM(SOLDE), AVG(SOLDE), COUNT(*)
005400          INTO :WS-TOTAL-CR, :WS-MOYENNE-CR, :WS-COUNT-CR
005500          FROM CLIENT
005600          WHERE POS = 'CR'
005700      END-EXEC
005800
005900      IF SQLCODE NOT = 0
006000          DISPLAY 'ERREUR STATS CREDITEURS : ' SQLCODE
006100      END-IF.

006200
006300      3000-AFFICHER-RESULTATS.
006400          DISPLAY '--- STATISTIQUES DES CLIENTS ---'
006500          DISPLAY ''
006600          DISPLAY '--- DEBITEURS ---'
006700          MOVE WS-TOTAL-DB TO WS-TOTAL-ED
```

```

006800      DISPLAY 'TOTAL : ' WS-TOTAL-ED
006900      MOVE WS-MOYENNE-DB TO WS-MOYENNE-ED
007000      DISPLAY 'MOYENNE : ' WS-MOYENNE-ED
007100      DISPLAY ''
007200      DISPLAY '--- CREDITEURS ---'
007300      MOVE WS-TOTAL-CR TO WS-TOTAL-ED
007400      DISPLAY 'TOTAL : ' WS-TOTAL-ED
007500      MOVE WS-MOYENNE-CR TO WS-MOYENNE-ED
007600      DISPLAY 'MOYENNE : ' WS-MOYENNE-ED.
007700
007800      9000-FIN.
007900      DISPLAY ''
008000      DISPLAY 'FIN DU PROGRAMME STATCLI'.
008100
***** ***** Bottom of Data *****


```

== STATISTIQUES DES CLIENTS ==

--- DEBITEURS ---

```

TOTAL : -    1,550.00
MOYENNE : -   193.75

```

--- CREDITEURS ---

```

TOTAL :     17,850.00
MOYENNE :    1,373.07

```

FIN DU PROGRAMME STATCLI

***** BOTTOM OF DATA *****

Exercice 7 : Totaux par region avec niveau 88

Énoncé

Écrire un programme COBOL-DB2 qui calcule, pour chaque région, la valeur totale des comptes débiteurs et créditeurs. Utiliser une variable conditionnelle (niveau 88).

Mon travail

J'ai déclaré des niveaux 88 pour représenter les 4 régions (REGION-PARIS, REGION-MARSEILLE, etc.). Une boucle PERFORM VARYING parcourt les régions et SET permet d'activer chaque condition.

Avantage du niveau 88 : Le code est plus lisible avec IF REGION-PARIS qu'avec IF WS-CODE-REGION = '01'.

Résolution

Programme : TOTREG.cbl

```

WORKING-STORAGE SECTION.
01 WS-CODE-REGION    PIC X(02).
     88 REGION-PARIS      VALUE '01'.

```

```
88 REGION-MARSEILLE  VALUE '02'.
88 REGION-LYON        VALUE '03'.
88 REGION-LILLE       VALUE '04'.
```

PROCEDURE DIVISION.

```
    PERFORM VARYING WS-IDX FROM 1 BY 1 UNTIL WS-IDX > 4
```

```
        EVALUATE WS-IDX
```

```
            WHEN 1 SET REGION-PARIS TO TRUE
```

```
            WHEN 2 SET REGION-MARSEILLE TO TRUE
```

```
            WHEN 3 SET REGION-LYON TO TRUE
```

```
            WHEN 4 SET REGION-LILLE TO TRUE
```

```
        END-EVALUATE
```

```
        PERFORM 2000-CALCULER-TOTAUX
```

```
    END-PERFORM.
```

Technique utilisée : Niveau 88 pour représenter les codes régions + boucle PERFORM VARYING

Captures d'écran

```
***** **** Top of Data *****  
000100      IDENTIFICATION DIVISION.  
000200      PROGRAM-ID. TOTREG.  
000300      *-----  
000400      * P3 EXERCICE 7 : TOTAUX PAR REGION AVEC NIVEAU 88  
000500      *-----  
000600  
000700      ENVIRONMENT DIVISION.  
000800  
000900      DATA DIVISION.  
001000      WORKING-STORAGE SECTION.  
001100  
001200      * VARIABLE AVEC NIVEAU 88 POUR LES REGIONS  
001300      01 WS-CODE-REGION    PIC X(02).  
001400          88 REGION-PARIS      VALUE '01'.  
001500          88 REGION-MARSEILLE   VALUE '02'.  
001600          88 REGION-LYON       VALUE '03'.  


---

001700          88 REGION-LILLE     VALUE '04'.  
001800      * VARIABLES HOST POUR DB2  
001900          01 WS-NOM-REGION   PIC X(15).  
002000          01 WS-TOTAL-DB      PIC S9(10)V99 COMP-3.  
002100          01 WS-TOTAL-CR      PIC S9(10)V99 COMP-3.  
002200  
002300      * VARIABLES D'EDITION  
002400          01 WS-TOTAL-ED      PIC -ZZZ,ZZZ,ZZ9.99.  
002500  
002600      * COMPTEUR DE BOUCLE  
002700          01 WS-IDX        PIC 9(01).  
002800  
002900      * SQLCA POUR GESTION ERREURS DB2  
003000          EXEC SQL  
003100          INCLUDE SQLCA  
003200          END-EXEC.  
003300
```

```
003300
003400      PROCEDURE DIVISION.
003500      0000-PRINCIPAL.
003600          DISPLAY '=====
003700          DISPLAY 'TOTALS PAR REGION (DEBITEURS/CREDITEURS )'
003800          DISPLAY '=====
003900          PERFORM 1000-TRAITER-REGIONS
004000          PERFORM 9000-FIN
004100          STOP RUN.
004200
004300      1000-TRAITER-REGIONS.
004400          PERFORM VARYING WS-IDX FROM 1 BY 1 UNTIL WS-IDX > 4
004500              EVALUATE WS-IDX
004600                  WHEN 1 SET REGION-PARIS TO TRUE
004700                  WHEN 2 SET REGION-MARSEILLE TO TRUE
004800                  WHEN 3 SET REGION-LYON TO TRUE
004900                  WHEN 4 SET REGION-LILLE TO TRUE
005000          END-EVALUATE
005100          PERFORM 2000-CALCULER-TOTALS
005200          PERFORM 3000-AFFICHER-REGION
005300          END-PERFORM.
005400
005500      2000-CALCULER-TOTALS.
005600      * RECUPERER NOM REGION ET TOTALS
005700          EXEC SQL
005800              SELECT R.NOM_REGION,
005900                  COALESCE(SUM(CASE WHEN C.POS = 'DB'
006000                      THEN C.SOLDE ELSE 0 END), 0),
006100                  COALESCE(SUM(CASE WHEN C.POS = 'CR'
006200                      THEN C.SOLDE ELSE 0 END), 0)
006300          INTO :WS-NOM-REGION, :WS-TOTAL-DB, :WS-TOTAL-CR
006400          FROM REGION R
006500          LEFT JOIN CLIENT C ON R.CODE_REGION = C.CODE_REGION
006600          WHERE R.CODE_REGION = :WS-CODE-REGION
```

```

006700          GROUP BY R.NOM_REGION
006800          END-EXEC
006900
007000      IF SQLCODE NOT = 0 AND SQLCODE NOT = 100
007100          DISPLAY 'ERREUR SQL REGION ' WS-CODE-REGION
007200          ' : ' SQLCODE
007300      END-IF.
007400
007500 3000-AFFICHER-REGION.
007600      DISPLAY ''
007700      DISPLAY '--- REGION : ' WS-NOM-REGION ' ---'
007800      MOVE WS-TOTAL-DB TO WS-TOTAL-ED
007900      DISPLAY 'TOTAL DEBITEURS : ' WS-TOTAL-ED
008000      MOVE WS-TOTAL-CR TO WS-TOTAL-ED
008100      DISPLAY 'TOTAL CREDITEURS : ' WS-TOTAL-ED.
008200
008300 9000-FIN.
=====
```

TOTAUX PAR REGION (DEBITEURS/CREDITEURS)

--- REGION : PARIS ---
 TOTAL DEBITEURS : - 300.00
 TOTAL CREDITEURS : 6,400.00

--- REGION : MARSEILLE ---
 TOTAL DEBITEURS : - 300.00
 TOTAL CREDITEURS : 4,050.00

--- REGION : LYON ---
 TOTAL DEBITEURS : - 850.00
 TOTAL CREDITEURS : 5,350.00

--- REGION : LILLE ---
 TOTAL DEBITEURS : - 100.00
 TOTAL CREDITEURS : 2,050.00

FIN DU PROGRAMME TOTREG

***** BOTTOM OF DATA *****

Exercice 8 : Création table MOUVEMENT

Énoncé

Créer une table DB2 permettant de gérer les mouvements des clients avec : numéro de compte, libellé, montant, sens (DB/CR), nature (CHQ/VER/VIR), date.

Mon travail

J'ai créé la table MOUVEMENT avec une clé étrangère vers CLIENT(NUM_COMPTE). Deux contraintes CHECK valident les valeurs de SENS (DB/CR) et NATURE (CHQ/VER/VIR).

Données de test : J'ai inséré plusieurs mouvements pour différents clients afin de tester les programmes suivants (relevé, total, etc.).

Résolution

```
CREATE TABLE MOUVEMENT (
    NUM_COMPTE  CHAR(3) REFERENCES CLIENT(NUM_COMPTE),
    LIB_MOUV    VARCHAR(15),
    MONTANT_MVT DECIMAL(8,2),
    SENS        CHAR(2),
    NATURE      CHAR(3),
    DATE_MVT   DATE,
    CHECK(SENS IN ('DB','CR')),
    CHECK(NATURE IN ('CHQ','VER','VIR'))
);

-- Données de test
INSERT INTO MOUVEMENT VALUES ('001','VIREMENT
SAL',1500.00,'CR','VIR',DATE('2024-01-15'));
INSERT INTO MOUVEMENT VALUES ('001','CHEQUE
001',200.00,'DB','CHQ',DATE('2024-01-20'));
-- ... autres mouvements de test ...
```

Captures d'écran

```
***** Top of Data *****
-----+-----+-----+-----+-----+
-- P3 EXERCICE 8 : Creation table MOUVEMENT          00001001
                                                00002000
-- DROP TABLE MOUVEMENT;                            00003001
                                                00004001
CREATE TABLE MOUVEMENT (                           00005001
    NUM_COMPTE  CHAR(3) REFERENCES CLIENT(NUM_COMPTE), 00006001
    LIB_MOUV    VARCHAR(15),                            00007001
    MONTANT_MVT DECIMAL(8,2),                          00008001
    SENS        CHAR(2),                             00009001
    NATURE      CHAR(3),                            00010001
    DATE_MVT    DATE,                                00011001
    CHECK(SENS IN ('DB','CR')),                      00012001
    CHECK(NATURE IN ('CHQ','VER','VIR'))           00013001
);
-----+-----+-----+-----+-----+-----+
```

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0

```
-----+-----+-----+-----+-----+
                                                00015000
INSERT INTO MOUVEMENT VALUES                  00016001
('001','VIREMENT SAL',1500.00,'CR','VIR',DATE('2024-01-15')); 00017001
-----+-----+-----+-----+-----+
```

DSNE615I NUMBER OF ROWS AFFECTED IS 1

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0

```
-----+-----+-----+-----+-----+
                                                00017101
INSERT INTO MOUVEMENT VALUES                  00017201
('001','CHEQUE 001',200.00,'DB','CHQ',DATE('2024-01-20'));
```

DSNE615I NUMBER OF ROWS AFFECTED IS 1

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0

```
-----+-----+-----+-----+-----+
                                                00017301
INSERT INTO MOUVEMENT VALUES                  00017401
('001','VERSEMENT',500.00,'CR','VER',DATE('2024-02-01'));
```

DSNE615I NUMBER OF ROWS AFFECTED IS 1

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0

```
-----+-----+-----+-----+-----+
                                                00017501
INSERT INTO MOUVEMENT VALUES                  00017601
('002','VIREMENT',800.00,'CR','VIR',DATE('2024-01-10'));
```

DSNE615I NUMBER OF ROWS AFFECTED IS 1

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0

```
-----+-----+-----+-----+-----+
                                                00017701
INSERT INTO MOUVEMENT VALUES                  00017801
('002','CHEQUE 002',1000.00,'DB','CHQ',DATE('2024-01-25'));
```

DSNE615I NUMBER OF ROWS AFFECTED IS 1

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0

```
-----+-----+-----+-----+-----+
                                                00017901
INSERT INTO MOUVEMENT VALUES
```

```

('003','VERSEMENT ESP',300.00,'CR','VER',DATE('2024-02-05'));          00018001
-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+
INSERT INTO MOUVEMENT VALUES                                         00018101
('003','VIREMENT',450.00,'CR','VIR',DATE('2024-02-10'));          00018201
-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+
INSERT INTO MOUVEMENT VALUES                                         00018301
('005','CHEQUE 005',150.00,'DB','CHQ',DATE('2024-01-18'));          00018401
-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+
INSERT INTO MOUVEMENT VALUES                                         00018501
('005','VIREMENT SAL',2500.00,'CR','VIR',DATE('2024-01-31'));          00018601
-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+
INSERT INTO MOUVEMENT VALUES                                         00018701
('010','VERSEMENT',1000.00,'CR','VER',DATE('2024-02-15'));          00018801
-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+
INSERT INTO MOUVEMENT VALUES                                         00018901
('012','CHEQUE 012',350.00,'DB','CHQ',DATE('2024-01-22'));          00019001
-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+
INSERT INTO MOUVEMENT VALUES                                         00019101
('012','VIREMENT',200.00,'CR','VIR',DATE('2024-02-20'));          00019201
-----+
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+
-- Verification                                         00020001
SELECT * FROM MOUVEMENT ORDER BY NUM_COMPTE, DATE_MVT;           00021001
-----+
NUM_COMPTE  LIB_MOUV      MONTANT_MVT  SENS  NATURE  DATE_MVT
-----+
001        VIREMENT SAL    1500.00    CR    VIR     2024-01-15
001        CHEQUE 001       200.00     DB    CHQ     2024-01-20
001        VERSEMENT        500.00    CR    VER     2024-02-01
002        VIREMENT         800.00    CR    VIR     2024-01-10
-----+

```

002	CHEQUE 002	1000.00	DB	CHQ	2024-01-25
003	VERSEMENT ESP	300.00	CR	VER	2024-02-05
003	VIREMENT	450.00	CR	VIR	2024-02-10
005	CHEQUE 005	150.00	DB	CHQ	2024-01-18
005	VIREMENT SAL	2500.00	CR	VIR	2024-01-31
010	VERSEMENT	1000.00	CR	VER	2024-02-15
012	CHEQUE 012	350.00	DB	CHQ	2024-01-22
012	VIREMENT	200.00	CR	VIR	2024-02-20

DSNE610I NUMBER OF ROWS DISPLAYED IS 12

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100

-----+-----+-----+-----+-----+-----+-----+
00022001

DSNE617I COMMIT PERFORMED, SQLCODE IS 0

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0

DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72

DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72

DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 14

DSNE621I NUMBER OF INPUT RECORDS READ IS 43

DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 121

***** Bottom of Data *****

Exercice 9 : Total mouvements d'un client

Énoncé

Écrire un programme COBOL-DB2 permettant de calculer le montant total des mouvements d'un client et le nombre total de mouvements. Recevoir le numéro via ACCEPT.

Mon travail

J'ai utilisé ACCEPT pour lire le numéro de compte depuis SYSIN. La requête SELECT utilisé SUM et COUNT avec COALESCE pour éviter les valeurs NULL si le client n'a pas de mouvements.

Paramétrage JCL : Le numéro de compte est passé via donnée In-Stream (//SYSIN DD *).

Résolution

Programme : TOTMVT.cbl

```

1000-LIRE-NUM-COMPTE.
ACCEPT WS-NUM-COMPTE
DISPLAY 'NUMERO COMPTE SAISI : [' WS-NUM-COMPTE ']'

IF WS-NUM-COMPTE = SPACES
  DISPLAY 'ERREUR : NUMERO COMPTE VIDE'
  DISPLAY 'VERIFIER SYSIN DANS LE JCL'
  STOP RUN
END-IF.

```

```

3000-CALCULER-TOTAUX.
  EXEC SQL
    SELECT COALESCE(SUM(MONTANT_MVT), 0), COUNT(*)
    INTO :WS-TOTAL-MVT, :WS-NB-MVT
    FROM MOUVEMENT
    WHERE NUM_COMPTE = :WS-NUM-COMPTE
  END-EXEC.

```

Technique utilisée : ACCEPT pour saisie + SUM/COUNT + COALESCE pour gérer les NULL

Validation importante : Le programme vérifié que le numéro de compte n'est pas vide avant d'exécuter les requêtes SQL. Cela évite l'erreur ABEND 4038 si le SYSIN est mal configuré.

Captures d'écran

```

***** **** ----- Top of Data -----
000100      IDENTIFICATION DIVISION.
000200      PROGRAM-ID. TOTMVT.
000300      *-----
000400      * P3 EXERCICE 9 : TOTAL MOUVEMENTS D'UN CLIENT
000500      * RECUPERE NUM_COMPTE VIA ACCEPT (IN-STREAM JCL)
000600      *-----
000700
000800      ENVIRONMENT DIVISION.
000900
001000      DATA DIVISION.
001100      WORKING-STORAGE SECTION.
001200
001300      * VARIABLE D'ENTREE (ACCEPT)
001400      01 WS-NUM-COMPTE      PIC X(03).
001500
001600      * VARIABLES HOST POUR DB2
001700      01 WS-TOTAL-MVT      PIC S9(10)V99 COMP-3.
001800      01 WS-NB-MVT        PIC S9(05) COMP.
001900      01 WS-NOM-CLIENT    PIC X(10).
002000
002100      * VARIABLES D'EDITION
002200      01 WS-TOTAL-ED      PIC -ZZZ,ZZZ,ZZ9.99.
002300      01 WS-NB-ED        PIC ZZ9.
002400
002500      * SQLCA POUR GESTION ERREURS DB2
002600      EXEC SQL
002700          INCLUDE SQLCA
002800      END-EXEC.
002900
003000      PROCEDURE DIVISION.
003100      0000-PRINCIPAL.
003200          PERFORM 1000-LIRE-NUM-COMPTE
003300          PERFORM 2000-VERIFIER-CLIENT

```

```
003400          PERFORM 3000-CALCULER-TOTAUX
003500          PERFORM 9000-FIN
003600          STOP RUN.
003700
003800          1000-LIRE-NUM-COMpte.
003900          ACCEPT WS-NUM-COMpte
004000          DISPLAY 'NUMERO COMPTE SAISI : ' WS-NUM-COMpte.
004100
004200          IF WS-NUM-COMpte = SPACES
004300              DISPLAY 'ERREUR : NUMERO COMPTE VIDE'
004400              DISPLAY 'VERIFIER SYSIN DANS LE JCL'
004500              STOP RUN
004600          END-IF.
004700
004800          2000-VERIFIER-CLIENT.
004900          EXEC SQL
005000              SELECT NOM_CLIENT
005100          INTO :WS-NOM-CLIENT
005200          FROM CLIENT
005300              WHERE NUM_COMPTE = :WS-NUM-COMpte
005400          END-EXEC
005500
005600          IF SQLCODE = 100
005700              DISPLAY 'CLIENT NON TROUVE : ' WS-NUM-COMpte
005800              STOP RUN
005900          END-IF
006000          IF SQLCODE NOT = 0
006100              DISPLAY 'ERREUR SQL : ' SQLCODE
006200              STOP RUN
006300          END-IF.
006400
006500          3000-CALCULER-TOTAUX.
006600          EXEC SQL
006700              SELECT COALESCE(SUM(MONTANT_MVT), 0),
```

```

006800          COUNT(*)  

006900          INTO :WS-TOTAL-MVT, :WS-NB-MVT  

007000          FROM MOUVEMENT  

007100          WHERE NUM_COMPTE = :WS-NUM-COMPTE  

007200          END-EXEC  

007300  

007400          IF SQLCODE = 0  

007500              DISPLAY '===== '  

007600              DISPLAY 'CLIENT : ' WS-NOM-CLIENT  

007700              DISPLAY 'COMPTE : ' WS-NUM-COMPTE  

007800              DISPLAY '===== '  

007900              MOVE WS-NB-MVT TO WS-NB-ED  

008000              DISPLAY 'NOMBRE MOUVEMENTS : ' WS-NB-ED  

008100              MOVE WS-TOTAL-MVT TO WS-TOTAL-ED  

008200              DISPLAY 'TOTAL MOUVEMENTS : ' WS-TOTAL-ED  

008300          ELSE  

008400              DISPLAY 'ERREUR CALCUL : ' SQLCODE

```

```
008500          END-IF.
```

```
008600  

008700      9000-FIN.  

008800      DISPLAY 'FIN DU PROGRAMME TOTMVT'.
```

```
***** ***** Bottom of Data *****
```

```
***** ***** Top of Data *****
```

```

000100 //EXECPG09 JOB (2025), 'ROCHA', MSGCLASS=X, CLASS=A,  

000200 //           MSGLEVEL=(1,1), NOTIFY=&SYUID  

000300 //JOBLIB     DD DSN=DSNA10.SDSNLOAD, DISP=SHR  

000400 //           DD DSN=FORM1112.FINANCE.LOAD, DISP=SHR  

000500 //           DD DSN=FORM1112.FINANCE.DBRM, DISP=SHR  

000600 //OUT1       OUTPUT OUTDISP=(KEEP,KEEP)  

000700 //STEP1      EXEC PGM=IKJEFT01, DYNAMNBR=20  

000800 //SYSTSPRT   DD SYSOUT=*
```

```
000900 //SYSPRINT   DD SYSOUT=T, OUTPUT=*.OUT1
```

```
001000 //SYSPRINT   DD DUMMY
```

```
001100 //SYSTSIN    DD *
```

```
001200     DSN SYSTEM(DBAG)
```

```
001300     RUN PROG(TOTMVT) PLAN(TOTMVT)
```

```
001400     END
```

```
001500 /*
```

```
001600 //SYSIN      DD *
```

```

001600 //SYSIN DD *
001700 005
001800 /*
001900 //

***** **** Bottom of Data *****

NUMERO COMPTE SAISI : 005
=====
CLIENT : RICHARD
COMPTE : 005
=====
NOMBRE MOUVEMENTS : 2
TOTAL MOUVEMENTS : 2,650.00
FIN DU PROGRAMME TOTMVT
***** BOTTOM OF DATA *****

```

Exercice 10 : Relevé de compte

Énoncé

Écrire un programme COBOL-DB2 permettant d'éditer un relevé de compte des mouvements d'un client avec colonnes Credit/Debit séparées.

Mon travail

J'ai créé un relevé bancaire avec en-tête client et liste des mouvements. Chaque mouvement s'affiche soit dans la colonne Credit (si SENS = 'CR') soit dans la colonne Debit.

Presentation : L'en-tête affiché le nom du client et son numéro de compte, suivi des colonnes Date/Libelle/Credit/Debit.

JCL requis : Le numéro de compte est lu via ACCEPT, donc le JCL doit inclure :

```

//SYSIN DD *
001
/*

```

Résolution

Programme : RELEVE.cbl

```

5100-AFFICHER-LIGNE.
    INITIALIZE WS-CREDIT-ED
    INITIALIZE WS-DEBIT-ED

    IF WS-SENS = 'CR'
        MOVE WS-MONTANT-MVT TO WS-CREDIT-ED
    ELSE
        MOVE WS-MONTANT-MVT TO WS-DEBIT-ED

```

```
END-IF  
  
DISPLAY WS-DATE-MVT :  
WS-LIB-MOUV :  
WS-CREDIT-ED :  
WS-DEBIT-ED.
```

Note : Utiliser **INITIALIZE** au lieu de **MOVE SPACES** pour les champs numeriques édits (PIC ZZZ,ZZ9.99). MOVE SPACES cause l'erreur IGYPA3005-S.

Sortie attendue :

```
=====  
Nom Client : DURAND      Numero de compte : 001  
=====  
Date operation  Libelle          Credit    Debit  
=====  
2024-01-15      VIREMENT SAL    1,500.00  
2024-01-20      CHEQUE 001       200.00  
=====
```

Captures d'écran

```
***** **** Top of Data *****  
000100      IDENTIFICATION DIVISION.  
000200      PROGRAM-ID. RELEVE.  
000300      *-----  
000400      * P3 EXERCICE 10 : RELEVE DE COMPTE D'UN CLIENT  
000500      *-----  
000600  
000700      ENVIRONMENT DIVISION.  
000800  
000900      DATA DIVISION.  
001000      WORKING-STORAGE SECTION.  
001100  
001200      * VARIABLE D'ENTREE (ACCEPT)  
001300      01 WS-NUM-COMPTE      PIC X(03).  
001400  
001500      * VARIABLES HOST POUR DB2  
001600      01 WS-NOM-CLIENT      PIC X(10).  
001700      01 WS-DATE-MVT      PIC X(10).  
001800      01 WS-LIB-MOUV      PIC X(15).  
001900      01 WS-MONTANT-MVT    PIC S9(6)V99 COMP-3.  
002000      01 WS-SENS          PIC X(02).  
002100  
002200      * VARIABLES D'EDITION  
002300      01 WS-CREDIT-ED      PIC ZZZ,ZZ9.99.  
002400      01 WS-DEBIT-ED      PIC ZZZ,ZZ9.99.  
002500      01 WS-FIN-CURSOR    PIC 9(01) VALUE 0.  
002600  
002700      * SQLCA POUR GESTION ERREURS DB2  
002800      EXEC SQL  
002900          INCLUDE SQLCA  
003000          END-EXEC.  
003100  
003200      * CURSEUR POUR MOUVEMENTS  
003300      EXEC SQL
```

```
003400      DECLARE C-MOUVEMENTS CURSOR FOR
003500          SELECT DATE_MVT, LIB_MOUV, MONTANT_MVT, SENS
003600          FROM MOUVEMENT
003700          WHERE NUM_COMPTE = :WS-NUM-COMPTE
003800          ORDER BY DATE_MVT
003900      END-EXEC.

004000
004100      PROCEDURE DIVISION.
004200      0000-PRINCIPAL.
004300          PERFORM 1000-LIRE-NUM-COMPTE
004400          PERFORM 2000-RECUPERER-CLIENT
004500          PERFORM 3000-AFFICHER-ENTETE
004600          PERFORM 4000-OUVRIR-CURSOR
004700          PERFORM 5000-LIRE-MOUVEMENTS
004800              UNTIL WS-FIN-CURSOR = 1
004900          PERFORM 6000-FERMER-CURSOR
005000          PERFORM 9000-FIN
005100      STOP RUN.

005200
005300      1000-LIRE-NUM-COMPTE.
005400          ACCEPT WS-NUM-COMPTE.

005500
005600      2000-RECUPERER-CLIENT.
005700          EXEC SQL
005800              SELECT NOM_CLIENT
005900                  INTO :WS-NOM-CLIENT
006000                  FROM CLIENT
006100                  WHERE NUM_COMPTE = :WS-NUM-COMPTE
006200          END-EXEC

006300
006400          IF SQLCODE NOT = 0
006500              DISPLAY 'CLIENT NON TROUVE'
006600              STOP RUN
006700          END-IF.
```

```
006800
006900      3000-AFFICHER-ENTETE.
007000          DISPLAY '=====
007100              NOM CLIENT : ' WS-NOM-CLIENT
007200                  '      NUMERO DE COMPTE : ' WS-NUM-COMPTE
007300          DISPLAY '=====
007400          DISPLAY 'DATE OPERATION LIBELLE           CREDIT     DEBIT'
007500          DISPLAY '=====
007600
007700      4000-OUVRIR-CURSOR.
007800          EXEC SQL OPEN C-MOUVEMENTS END-EXEC
007900          IF SQLCODE NOT = 0
00800              MOVE 1 TO WS-FIN-CURSOR
008100          END-IF.
008200
008300      5000-LIRE-MOUVEMENTS.
008400          EXEC SQL
008500              FETCH C-MOUVEMENTS
008600                  INTO :WS-DATE-MVT, :WS-LIB-MOUV,
008700                      :WS-MONTANT-MVT, :WS-SENS
008800          END-EXEC
008900
009000          EVALUATE SQLCODE
009100              WHEN 0
009200                  PERFORM 5100-AFFICHER-LIGNE
009300              WHEN 100
009400                  MOVE 1 TO WS-FIN-CURSOR
009500              WHEN OTHER
009600                  DISPLAY 'ERREUR : ' SQLCODE
009700                  MOVE 1 TO WS-FIN-CURSOR
009800          END-EVALUATE.
009900
010000      5100-AFFICHER-LIGNE.
010100          INITIALIZE WS-CREDIT-ED
```

```

010200      INITIALIZE WS-DEBIT-ED
010300
010400      IF WS-SENS = 'CR'
010500          MOVE WS-MONTANT-MVT TO WS-CREDIT-ED
010600      ELSE
010700          MOVE WS-MONTANT-MVT TO WS-DEBIT-ED
010800      END-IF
010900
011000      DISPLAY WS-DATE-MVT   '
011100          WS-LIB-MOUV   '
011200          WS-CREDIT-ED   '
011300          WS-DEBIT-ED.
011400
011500      6000-FERMER-CURSOR.
011600      EXEC SQL CLOSE C-MOUVEMENTS END-EXEC
011700      DISPLAY '=====
011800

```

```

011800
011900      9000-FIN.
012000      DISPLAY 'FIN DU PROGRAMME RELEVE'.
012100
***** ***** Bottom of Data *****

```

```

***** ***** Top of Data *****
000100 //DDINSTRM JOB (2025), 'ROCHA', MSGCLASS=X, CLASS=A,
000200 //           MSGLEVEL=(1,1), NOTIFY=&SYSUID
000300 //JOBLIB     DD DSN=DSNA10.SDSNLOAD, DISP=SHR
000400 //           DD DSN=FORM1112.FINANCE.LOAD, DISP=SHR
000500 //           DD DSN=FORM1112.FINANCE.DBRM, DISP=SHR
000600 //OUT1       OUTPUT OUTDISP=(KEEP,KEEP)
000700 //STEP1      EXEC PGM=IKJEFT01, DYNAMNBR=20
000800 //SYSTSPRT  DD SYSOUT=*
000900 //SYSPRINT  DD SYSOUT=T, OUTPUT=*.OUT1
001000 //SYSPRINT  DD DUMMY
001100 //SYSTSIN   DD *
001200      DSN SYSTEM(DBAG)
001300      RUN PROG(RELEVE) PLAN(RELEVE)
001400      END
001500 /*
001600 //SYSIN    DD *

```

```

001700 005
001800 /*
001900 //
***** **** Bottom of Data ****
=====
NOM CLIENT : RICHARD      NUMERO DE COMPTE : 005
=====
DATE OPERATION LIBELLE      CREDIT      DEBIT
=====
2024-01-18  CHEQUE 005      0.00      150.00
2024-01-31  VIREMENT SAL    2,500.00     0.00
=====
FIN DU PROGRAMME RELEVE
***** BOTTOM OF DATA ****

```

Exercice 11 : Mouvements de l'année 2024

Énoncé

Écrire un programme COBOL-DB2 permettant d'afficher les mouvements de l'année 2024 de tous les clients.

Mon travail

J'ai utilisé la fonction YEAR(DATE_MVT) dans le WHERE pour filtrer uniquement l'année 2024. Une jointure avec CLIENT permet d'afficher le nom du client à côté de chaque mouvement.

Compteur : Le programme compte et affiché le nombre total de mouvements 2024 à la fin.

Résolution

Programme : MVT2024.cbl

```

EXEC SQL
DECLARE C-MVT2024 CURSOR FOR
SELECT M.NUM_COMPTE, C.NOM_CLIENT,
       M.DATE_MVT, M.LIB_MOUV,
       M.MONTANT_MVT, M.SENS, M.NATURE
  FROM MOUVEMENT M
 INNER JOIN CLIENT C ON M.NUM_COMPTE = C.NUM_COMPTE
 WHERE YEAR(M.DATE_MVT) = 2024
 ORDER BY M.DATE_MVT, M.NUM_COMPTE
END-EXEC.

```

Technique utilisée : Fonction YEAR() pour filtrer par année + jointure CLIENT

Captures d'écran

```
***** **** Top of Data ****
000100      IDENTIFICATION DIVISION.
000200      PROGRAM-ID. MVT2024.
000300      *
000400      * P3 EXERCICE 11 : MOUVEMENTS DE L'ANNEE 2024
000500      *
000600
000700      ENVIRONMENT DIVISION.
000800
000900      DATA DIVISION.
001000      WORKING-STORAGE SECTION.
001100
001200      * VARIABLES HOST POUR DB2
001300      01 WS-NUM-COMPTE      PIC X(03).
001400      01 WS-NOM-CLIENT      PIC X(10).
001500      01 WS-DATE-MVT        PIC X(10).
001600      01 WS-LIB-MOUV        PIC X(15).

001700      01 WS-MONTANT-MVT    PIC S9(6)V99 COMP-3.
001800      01 WS-SENS          PIC X(02).
001900      01 WS-NATURE         PIC X(03).
002000
002100      * VARIABLES DE TRAVAIL
002200      01 WS-MONTANT-ED     PIC ZZZ,ZZ9.99.
002300      01 WS-FIN-CURSOR    PIC 9(01) VALUE 0.
002400      01 WS-COMPTEUR       PIC 9(03) VALUE 0.
002500
002600      * SQLCA POUR GESTION ERREURS DB2
002700          EXEC SQL
002800              INCLUDE SQLCA
002900          END-EXEC.
003000
003100      * CURSEUR POUR MOUVEMENTS 2024
003200          EXEC SQL
003300              DECLARE C-MVT2024 CURSOR FOR
```

```
003400      SELECT M.NUM_COMPTE, C.NOM_CLIENT,
003500          M.DATE_MVT, M.LIB_MOUV,
003600          M.MONTANT_MVT, M.SENS, M.NATURE
003700      FROM MOUVEMENT M
003800      INNER JOIN CLIENT C ON M.NUM_COMPTE = C.NUM_COMPTE
003900      WHERE YEAR(M.DATE_MVT) = 2024
004000      ORDER BY M.DATE_MVT, M.NUM_COMPTE
004100      END-EXEC.

004200
004300      PROCEDURE DIVISION.
004400      0000-PRINCIPAL.
004500          PERFORM 1000-AFFICHER-ENTETE
004600          PERFORM 2000-OUVRIR-CURSOR
004700          PERFORM 3000-LIRE-MOUVEMENTS
004800              UNTIL WS-FIN-CURSOR = 1
004900          PERFORM 4000-FERMER-CURSOR
005000          PERFORM 9000-FIN

005100      STOP RUN.

005200
005300      1000-AFFICHER-ENTETE.
005400          DISPLAY '=====
005500          DISPLAY '        MOUVEMENTS DE L ANNEE 2024'
005600          DISPLAY '=====
005700          DISPLAY 'NUM CLIENT      DATE      LIBELLE
005800              ' MONTANT    SENS NAT'.
005900
006000      2000-OUVRIR-CURSOR.
006100          EXEC SQL OPEN C-MVT2024 END-EXEC
006200          IF SQLCODE NOT = 0
006300              DISPLAY 'ERREUR OUVERTURE : ' SQLCODE
006400              MOVE 1 TO WS-FIN-CURSOR
006500          END-IF.

006600
006700      3000-LIRE-MOUVEMENTS.
```

```
006800      EXEC SQL
006900          FETCH C-MVT2024
007000          INTO :WS-NUM-COMPTE, :WS-NOM-CLIENT,
007100              :WS-DATE-MVT, :WS-LIB-MOUV,
007200              :WS-MONTANT-MVT, :WS-SENS, :WS-NATURE
007300      END-EXEC
007400
007500      EVALUATE SQLCODE
007600          WHEN 0
007700              ADD 1 TO WS-COMPTEUR
007800              MOVE WS-MONTANT-MVT TO WS-MONTANT-ED
007900              DISPLAY WS-NUM-COMPTE
008000                  WS-NOM-CLIENT
008100                  WS-DATE-MVT
008200                  WS-LIB-MOUV
008300                  WS-MONTANT-ED
008400                  WS-SENS
008500          WS-NATURE
008600          WHEN 100
008700              MOVE 1 TO WS-FIN-CURSOR
008800          WHEN OTHER
008900              DISPLAY 'ERREUR FETCH : ' SQLCODE
009000              MOVE 1 TO WS-FIN-CURSOR
009100      END-EVALUATE.
009200
009300      4000-FERMER-CURSOR.
009400      EXEC SQL CLOSE C-MVT2024 END-EXEC
009500      DISPLAY '=====
009600          DISPLAY 'TOTAL MOUVEMENTS 2024 : ' WS-COMPTEUR.
009700
009800      9000-FIN.
009900      DISPLAY 'FIN DU PROGRAMME MVT2024'.
010000
***** ***** Bottom of Data *****
```

MOUVEMENTS DE L'ANNEE 2024						
NUM	CLIENT	DATE	LIBELLE	MONTANT	SENS	NAT
002	MARTIN	2024-01-10	VIREMENT	800.00	CR	VIR
001	DURAND	2024-01-15	VIREMENT SAL	1,500.00	CR	VIR
005	RICHARD	2024-01-18	CHEQUE 005	150.00	DB	CHQ
001	DURAND	2024-01-20	CHEQUE 001	200.00	DB	CHQ
012	LEROY	2024-01-22	CHEQUE 012	350.00	DB	CHQ
002	MARTIN	2024-01-25	CHEQUE 002	1,000.00	DB	CHQ
005	RICHARD	2024-01-31	VIREMENT SAL	2,500.00	CR	VIR
001	DURAND	2024-02-01	VERSEMENT	500.00	CR	VER
003	BERNARD	2024-02-05	VERSEMENT ESP	300.00	CR	VER
003	BERNARD	2024-02-10	VIREMENT	450.00	CR	VIR
010	SIMON	2024-02-15	VERSEMENT	1,000.00	CR	VER
012	LEROY	2024-02-20	VIREMENT	200.00	CR	VIR
<hr/>						
TOTAL MOUVEMENTS 2024 : 012						
FIN DU PROGRAMME MVT2024						

Exercice 12 : Mouvements 2024 d'un client spécifique

Énoncé

Refaire l'exercice 11 en précisant le numéro de compte d'un client déterminé. Récupérer le numéro via donnée In-Stream.

Mon travail

Ce programme reprend la logique des mouvements 2024 (Ex11) mais pour un client spécifique. Le numéro de compte est lu dynamiquement via ACCEPT. Le curseur filtre sur **NUM_COMPTE** ET **YEAR(DATE_MVT) = 2024**.

Déférence avec Ex11 : Ex11 affiché tous les clients avec un JOIN, Ex12 filtre sur un seul client passé en SYSIN.

Résolution

Programme : RLV012.cbl

```

WORKING-STORAGE SECTION.
* Variable d'entrée (ACCEPT depuis SYSIN In-Stream)
 01 WS-NUM-COMPTE      PIC X(03).
* Variables host pour DB2
 01 WS-NOM-CLIENT      PIC X(10).
 01 WS-DATE-MVT        PIC X(10).
 01 WS-LIB-MOUV         PIC X(15).
 01 WS-MONTANT-MVT     PIC S9(6)V99 COMP-3.
 01 WS-SENS             PIC X(02).
 01 WS-NATURE           PIC X(03).

```

```
* Curseur pour mouvements 2024 d'un client spécifique
EXEC SQL
    DECLARE C-MVT2024-CLI CURSOR FOR
        SELECT DATE_MVT, LIB_MOUV, MONTANT_MVT, SENS, NATURE
        FROM MOUVEMENT
        WHERE NUM_COMPTE = :WS-NUM-COMPTE
            AND YEAR(DATE_MVT) = 2024
        ORDER BY DATE_MVT
    END-EXEC.

PROCEDURE DIVISION.
1000-LIRE-NUM-COMPTE.
ACCEPT WS-NUM-COMPTE
DISPLAY 'COMPTE DEMANDE : ' WS-NUM-COMPTE.
```

JCL avec donnée In-Stream :

```
//SYSIN DD *
001
/*
```

Technique utilisée : CURSOR avec double filtre (NUM_COMPTE + YEAR) + ACCEPT pour paramètre dynamique

Captures d'écran

```
***** **** Top of Data ****
000100      IDENTIFICATION DIVISION.
000200      PROGRAM-ID. RLV012.
000300      *
000400      * P3 EXERCICE 12 : MOUVEMENTS 2024 D'UN CLIENT (IN-STREAM)
000500      * REPRISE DE L'EXERCICE 11 AVEC NUMERO CLIENT EN SYSIN
000600      *
000700
000800      ENVIRONMENT DIVISION.
000900
001000      DATA DIVISION.
001100      WORKING-STORAGE SECTION.
001200
001300      * VARIABLE D'ENTREE (ACCEPT DEPUIS SYSIN IN-STREAM)
001400      01 WS-NUM-COMpte      PIC X(03).
001500
001600      * VARIABLES HOST POUR DB2
001700      01 WS-NOM-CLIENT      PIC X(10).
001800      01 WS-DATE-MVT        PIC X(10).
001900      01 WS-LIB-MOUV         PIC X(15).
002000      01 WS-MONTANT-MVT     PIC S9(6)V99 COMP-3.
002100      01 WS-SENS            PIC X(02).
002200      01 WS-NATURE          PIC X(03).
002300
002400      * VARIABLES DE TRAVAIL
002500      01 WS-MONTANT-ED      PIC ZZZ,ZZ9.99.
002600      01 WS-FIN-CURSOR       PIC 9(01) VALUE 0.
002700      01 WS-COMPTEUR         PIC 9(03) VALUE 0.
002800
002900      * SQLCA POUR GESTION ERREURS DB2
003000      EXEC SQL
003100      INCLUDE SQLCA
003200      END-EXEC.
003300
003400      * CURSEUR POUR MOUVEMENTS 2024 D'UN CLIENT SPECIFIQUE
003500      EXEC SQL
003600      DECLARE C-MVT2024-CLI CURSOR FOR
003700      SELECT DATE_MVT, LIB_MOUV, MONTANT_MVT, SENS, NATURE
003800      FROM MOUVEMENT
003900      WHERE NUM_COMPTE = :WS-NUM-COMpte
004000      AND YEAR(DATE_MVT) = 2024
004100      ORDER BY DATE_MVT
004200      END-EXEC.
004300
004400      PROCEDURE DIVISION.
004500      0000-PRINCIPAL.
004600      PERFORM 1000-LIRE-NUM-COMpte
004700      PERFORM 2000-RECUPERER-CLIENT
004800      PERFORM 3000-AFFICHER-ENTETE
004900      PERFORM 4000-OUVRIR-CURSOR
005000      PERFORM 5000-LIRE-MOUVEMENTS
```

```
005100          UNTIL WS-FIN-CURSOR = 1
005200          PERFORM 6000-FERMER-CURSOR
005300          PERFORM 9000-FIN
005400          STOP RUN.

005500
005600          1000-LIRE-NUM-COMpte.
005700          * LECTURE DEPUIS SYSIN (DONNEE IN-STREAM DU JCL)
005800          ACCEPT WS-NUM-COMpte
005900          DISPLAY 'COMpte DEMANDE : ' WS-NUM-COMpte.
006000
006100          2000-RECUPERER-CLIENT.
006200          EXEC SQL
006300          SELECT NOM_CLIENT
006400          INTO :WS-NOM-CLIENT
006500          FROM CLIENT
006600          WHERE NUM_COMpte = :WS-NUM-COMpte
006700          END-EXEC

006800
006900          IF SQLCODE NOT = 0
007000          DISPLAY 'CLIENT NON TROUVE : ' WS-NUM-COMpte
007100          STOP RUN
007200          END-IF.

007300
007400          3000-AFFICHER-ENTete.
007500          DISPLAY '=====
007600          DISPLAY ' MOUVEMENTS 2024 - CLIENT : ' WS-NOM-CLIENT
007700          DISPLAY ' NUMERO DE COMpte : ' WS-NUM-COMpte
007800          DISPLAY '=====
007900          DISPLAY 'DATE           LIBELLE           MONTANT   SENS NAT'.
008000
008100          4000-Ouvrir-CURSOR.
008200          EXEC SQL OPEN C-MVT2024-CLI END-EXEC
008300          IF SQLCODE NOT = 0
008400          DISPLAY 'ERREUR OUVERTURE : ' SQLCODE
```

```
008500      MOVE 1 TO WS-FIN-CURSOR
008600      END-IF.
008700
008800      5000-LIRE-MOUVEMENTS.
008900      EXEC SQL
009000          FETCH C-MVT2024-CLI
009100          INTO :WS-DATE-MVT, :WS-LIB-MOUV,
009200                  :WS-MONTANT-MVT, :WS-SENS, :WS-NATURE
009300      END-EXEC
009400
009500      EVALUATE SQLCODE
009600          WHEN 0
009700              ADD 1 TO WS-COMPTEUR
009800              MOVE WS-MONTANT-MVT TO WS-MONTANT-ED
009900              DISPLAY WS-DATE-MVT   :
010000                  WS-LIB-MOUV   :
010100                  WS-MONTANT-ED   :
010200                  WS-SENS   :
010300                  WS-NATURE
010400          WHEN 100
010500              MOVE 1 TO WS-FIN-CURSOR
010600          WHEN OTHER
010700              DISPLAY 'ERREUR FETCH : ' SQLCODE
010800              MOVE 1 TO WS-FIN-CURSOR
010900      END-EVALUATE.
011000
011100      6000-FERMER-CURSOR.
011200      EXEC SQL CLOSE C-MVT2024-CLI END-EXEC
011300      DISPLAY '=====
011400          DISPLAY 'TOTAL MOUVEMENTS 2024 : ' WS-COMPTEUR.
011500
011600      9000-FIN.
011700      DISPLAY 'FIN DU PROGRAMME RLV012'.
011800
```

```

001000 //SYSPRINT DD DUMMY
001100 //SYSTSIN DD *
001200      DSN SYSTEM(DBAG)
001300      RUN PROG(RLV012) PLAN(RLV012)
001400      END
001500 /*
001600 //SYSIN    DD *
001700 005
002100 /*
002200 //
***** ***** Bottom of Data *****

```

COMpte DEMANDE : 005

MOUVEMENTS 2024 - CLIENT : RICHARD
NUMERO DE COMPTE : 005

DATE	LIBELLE	MONTANT	SENS	NAT
2024-01-18	CHEQUE 005	150.00	DB	CHQ
2024-01-31	VIREMENT SAL	2,500.00	CR	VIR

TOTAL MOUVEMENTS 2024 : 002

FIN DU PROGRAMME RLV012

***** BOTTOM OF DATA *****

Annexes

Liste des programmes COBOL

Programme	Description
AFFREG	Afficher region
INSCLI	Insérer client
AFFCLI	Afficher clients region
MAJCLI	Mise à jour client
LSTRUPT	Liste avec ruptures
STATCLI	Statistiques DB/CR
TOTREG	Totaux par region
TOTMVT	Total mouvements
RELEVE	Relevé de compte
MVT2024	Mouvements 2024
RLV012	Mouvements 2024 client

Liste des scripts SQL

Script	Partie	Description
PT1EX01	P1	Création des tables (REGION, NATCPT, PROF, CLIENT)
PT1EX02	P1	Alimentation des tables de référence
PT1EX03	P1	Insertion des 20 clients
PT2EX01	P2	Extraction clients par profession
PT2EX02	P2	Répartition DB/CR
PT2EX03	P2	Répartition par région
PT2EX04	P2	Index sur CODE_REGION
PT2EX05	P2	Index sur CODE_PROF
PT2EX06	P2	Édition triés région/profession
PT2EX07	P2	Fusion populations (UNION)
PT2EX08	P2	Vue CLIENT_REDUIT
PT2EX09	P2	Analyse multi-critères
PT2EX10	P2	Clients anormalement débiteurs
PT3EX08	P3	Création table MOUVEMENT

Liste des vues créées

Vue	Description
V_CLIENT_COMPTABLE	Clients comptables
V_CLIENT_FONCTION	Clients fonctionnaires
V_CLIENT_MEDECIN	Clients médecins
V_CLIENT_DEBITEUR	Clients débiteurs
V_CLIENT_CREDITEUR	Clients créditeurs
V_CLIENT_PARIS	Clients région Paris
V_CLIENT_MARSEILLE	Clients région Marseille
V_CLIENT_LYON	Clients région Lyon
V_CLIENT_LILLE	Clients région Lille
V_CLIENT_REDUIT	Vue simplifiée CLIENT

Liste des index créés

Index	Colonne
IDX_CLIENT_REGION	CODE_REGION
IDX_CLIENT_PROF	CODE_PROF

Conclusion

Ce projet m'a permis de mettre en pratique l'ensemble des compétences acquises durant la formation POEI Mainframe COBOL. A travers les trois parties du projet, j'ai pu :

- **Maîtriser SQL/DB2** : Création de tables avec contraintes (PK, FK, CHECK), insertion de données, requêtes complexes avec jointures, sous-requêtes, fonctions d'agrégation et vues.
- **Développer en COBOL-DB2** : Intégration du SQL embarqué dans les programmes COBOL, utilisation des curseurs pour le traitement multi-lignes, gestion des erreurs via SQLCODE, et techniques avancées (ruptures de contrôle, niveau 88, ACCEPT/SYSIN).
- **Travailler dans l'environnement z/OS** : Navigation ISPF, utilisation de SPUFI pour les requêtes interactives, compilation et execution de programmes via JCL.

Le projet couvre un cas concret de gestion clientèle dans le secteur financier, avec 12 programmes COBOL et plus de 10 requêtes SQL. Les principales difficultés rencontrées (mots réservés, gestion des erreurs, formats de données) m'ont permis de développer une approche méthodique de résolution de problèmes.

Cette expérience constitue une base solide pour aborder des projets mainframe en entreprise.

Rapport réalisé par Josué ROCHA - Formation POEI Mainframe COBOL - M2i Formation, Strasbourg - Décembre 2025