UNIVERSITÉ PARIS 13

Eppur si muove !
Institut Galilée

LIPN

RAPPORT DE BASES DE DONNÉES AVANCÉES

Master EID²

Gestion Commande Intelligente : « *GestCom2* »

Groupe 4 :
Mohamed BEN SAAD 11400535
Elias ABDELLI 11501114

Enseignant :
Faouzi BOUFARES

Table des matières

Nous avons commencé par compiler le fichier pour relater l'étendu de notre futur travail. La résolution de plusieurs bugs dans le code comme les fautes de frappes, les créations de tables en amont et les tests des fonctions déjà présentent ont été nécessaires. Le résultat des requêtes sont commentés dans le fichier **B4_G2.sql.**

1.Historique des Mouvements Clients

Notre première création est la table **Historique_Mvts_Clients**. Pour ce faire, nous avons utilisé comme modèles la table **Historique_Mvts_Articles** ainsi que son Triggers pour générer les insertions de cette table.

```
DROP TABLE HISTORIQUE_MVTS_CLIENTS;


CREATE TABLE HISTORIQUE_MVTS_CLIENTS
(
        NOMUSER         VARCHAR2(15),
        TYPEMVT         VARCHAR2(15),
        CODCLI          VARCHAR2(10),
        CIVCLI          VARCHAR2(12),
        NOMCLI          VARCHAR2(20),
        PRENCLI         VARCHAR2(20),
        CATCLI          NUMBER(1),
        ADNCLI          VARCHAR2(10),
        ADRCLI          VARCHAR2(50),
        CPCLI           VARCHAR2(10),
        VILCLI          VARCHAR2(20),
        PAYSCLI         VARCHAR2(30),
        MAILCLI         VARCHAR2(30),
        TELCLI          VARCHAR2(20),
        DATMVT          DATE,
        CONSTRAINT NN_HIST_CLIENTS_CODCLI       CHECK(CODCLI IS NOT NULL),
        CONSTRAINT NN_HIST_CLIENTS_NOMUSER      CHECK(NOMUSER IS NOT NULL),
        CONSTRAINT NN_HIST_CLIENTS_NOMTYPEMVT   CHECK(TYPEMVT IS NOT NULL)
);
```

```
CREATE OR REPLACE TRIGGER MOUVEMENTS_CLIENTS
        AFTER UPDATE OR INSERT OR DELETE ON CLIENTS
        FOR EACH ROW
DECLARE

        V_CODCLI                VARCHAR2(10)    := :OLD.CODCLI;
        V_CIVCLI                VARCHAR2(12)    := :OLD.CIVCLI;
        V_NOMCLI                VARCHAR2(20)    := :OLD.NOMCLI;
        V_PRENCLI               VARCHAR2(20)    := :OLD.PRENCLI;
        V_CATCLI                NUMBER(1)       := :OLD.CATCLI;
        V_ADNCLI                VARCHAR2(10)    := :OLD.ADNCLI;
        V_ADRCLI                VARCHAR2(50)    := :OLD.ADRCLI;
        V_CPCLI                 VARCHAR2(10)    := :OLD.CPCLI;
        V_VILCLI                VARCHAR2(20)    := :OLD.VILCLI;
        V_PAYSCLI               VARCHAR2(30)    := :OLD.PAYSCLI;
        V_MAILCLI               VARCHAR2(30)    := :OLD.MAILCLI;
        V_TELCLI                VARCHAR2(20)    := :OLD.TELCLI;

        VI_CODCLI               VARCHAR2(10)    := :NEW.CODCLI;
        VI_CIVCLI               VARCHAR2(12)    := :NEW.CIVCLI;
        VI_NOMCLI               VARCHAR2(20)    := :NEW.NOMCLI;
        VI_PRENCLI              VARCHAR2(20)    := :NEW.PRENCLI;
        VI_CATCLI               NUMBER(1)       := :NEW.CATCLI;
        VI_ADNCLI               VARCHAR2(10)    := :NEW.ADNCLI;
        VI_ADRCLI               VARCHAR2(50)    := :NEW.ADRCLI;
        VI_CPCLI                VARCHAR2(10)    := :NEW.CPCLI;
        VI_VILCLI               VARCHAR2(20)    := :NEW.VILCLI;
        VI_PAYSCLI              VARCHAR2(30)    := :NEW.PAYSCLI;
        VI_MAILCLI              VARCHAR2(30)    := :NEW.MAILCLI;
        VI_TELCLI               VARCHAR2(20)    := :NEW.TELCLI;


BEGIN
        IF UPDATING THEN
                INSERT INTO HISTORIQUE_MVTS_CLIENTS (NOMUSER, TYPEMVT, CODCLI, CIVCLI, NOMCLI, PRENCLI, CATCLI, ADNCLI, ADRCLI, CPCLI, VILCLI, PAYSCLI, MAILCLI, TELCLI, DATMVT)
                VALUES (USER, 'UPDATE', V_CODCLI, V_CIVCLI, V_NOMCLI, V_PRENCLI, V_CATCLI, V_ADNCLI, V_ADRCLI, V_CPCLI, V_VILCLI, V_PAYSCLI, V_MAILCLI, V_TELCLI, SYSDATE);
        END IF;

        IF DELETING THEN
                INSERT INTO HISTORIQUE_MVTS_CLIENTS (NOMUSER, TYPEMVT, CODCLI, CIVCLI, NOMCLI, PRENCLI, CATCLI, ADNCLI, ADRCLI, CPCLI, VILCLI, PAYSCLI, MAILCLI, TELCLI, DATMVT)
                VALUES (USER, 'DELETE', V_CODCLI, V_CIVCLI, V_NOMCLI, V_PRENCLI, V_CATCLI, V_ADNCLI, V_ADRCLI, V_CPCLI, V_VILCLI, V_PAYSCLI, V_MAILCLI, V_TELCLI, SYSDATE);
        END IF;

        IF INSERTING THEN
                INSERT INTO HISTORIQUE_MVTS_CLIENTS (NOMUSER, TYPEMVT, CODCLI, CIVCLI, NOMCLI, PRENCLI, CATCLI, ADNCLI, ADRCLI, CPCLI, VILCLI, PAYSCLI, MAILCLI, TELCLI, DATMVT)
                VALUES (USER, 'INSERT', VI_CODCLI, VI_CIVCLI, VI_NOMCLI, VI_PRENCLI, VI_CATCLI, VI_ADNCLI, VI_ADRCLI, VI_CPCLI, VI_VILCLI, VI_PAYSCLI, VI_MAILCLI, VI_TELCLI, SYSDATE);
        END IF;
END;
/
```

2. Articles toujours commandés ensemble

Pour sélectionner les articles qui sont toujours commandé ensemble nous avons utilisé plusieurs étapes :

- Création de la table **COUPLE** comprenant toutes les paires distinctes d'articles d'une même commande
- Création de la table **NB_COUPLE** comprenant le nombre de fois ou apparaît un couple
- Création de la table **NB_COM** comprenant le nombre commande par articles

```
PROMPT ========================================================================
PROMPT >> Requete  : Articles toujours commandés ensemble
PROMPT ========================================================================
PROMPT
TTITLE CENTER 'Requete: les articles qui sont toujours commandés ensemble' skip 1 -
       LEFT   '========================================================================' skip 0

-- >>>>>>>>>>>>>>>>>>>>>>>>> -- Résultat généré:
-- Le(s) les articles qui sont toujours commandés ensemble sont :

DROP TABLE COUPLE;
CREATE TABLE COUPLE(
        NUMCOM          VARCHAR2(20),
        REFART1         VARCHAR2(20),
        REFART2         VARCHAR2(20)
);

CREATE OR REPLACE PROCEDURE MEME_COMMANDE AS
        CURSOR C1 IS SELECT * FROM DETAILCOM ORDER BY REFART;
        CURSOR C2 IS SELECT * FROM DETAILCOM ORDER BY REFART;
        BEGIN
                FOR REC1 IN C1
                LOOP

                        FOR REC2 IN C2
                        LOOP

                                IF C2%ROWCOUNT > C1%ROWCOUNT AND REC1.REFART != REC2.REFART AND REC1.NUMCOM = REC2.NUMCOM THEN

                                        EXECUTE IMMEDIATE 'INSERT INTO COUPLE VALUES('||chr(39)||REC1.NUMCOM||chr(39)||','||chr(39)||
                                                          REC1.REFART||chr(39)||','||chr(39)||REC2.REFART||chr(39)||')';
                                END IF;
                        END LOOP;
                END LOOP;
        END;
/
EXEC MEME_COMMANDE;

DROP TABLE NBCOUPLE;
CREATE TABLE NBCOUPLE AS SELECT COUNT(*) AS NB,REFART1,REFART2 FROM COUPLE GROUP BY REFART1,REFART2 ORDER BY REFART1;
DROP TABLE NB_COM;
CREATE TABLE NB_COM AS SELECT COUNT(NUMCOM) as NBC,REFART FROM DETAILCOM GROUP BY REFART ORDER BY REFART;

SELECT * FROM NBCOUPLE;

SELECT * FROM NB_COM;
```

Une fois toutes nos tables créer, nous avons juste à comparer si le nombre de fois ou la paire apparaît est égale au nombre de commande de ceux-ci à l'aide notre procédure **TOUJOURS_COMMANDE**. Si ce n'est pas le cas on les supprimes de la table NBCOUPLE.

```
CREATE OR REPLACE PROCEDURE TOUJOURS_COMMANDER AS
        CURSOR C2 IS SELECT * FROM NB_COM;
        MAXI            NUMBER(5);
        REFART11        VARCHAR2(20);
        REFART22        VARCHAR2(20);

        BEGIN
                FOR REC2 IN C2 LOOP
                        EXECUTE IMMEDIATE 'DELETE FROM NBCOUPLE WHERE NB != '||REC2.NBC||' AND REFART1='||chr(39)||REC2.REFART||chr(39);
                        EXECUTE IMMEDIATE 'DELETE FROM NBCOUPLE WHERE NB != '||REC2.NBC||' AND REFART2='||chr(39)||REC2.REFART||chr(39);
                END LOOP;
        END;
/
EXEC TOUJOURS_COMMANDER;

SELECT * FROM NBCOUPLE;
TTITLE '';
```

Voici le seul couple qui est toujours commander ensemble :

```
Requete: les articles qui sont toujours commandés ensemble
========================================================================
        NB REFART1              REFART2
---------- -------------------- --------------------
         2 F1.003               F1.013
```

3.Cohérence de la BD

Pour vérifier si toutes les commandes portent au moins sur un article et affichez celles qui sont donc vides, nous créons le procédure **COHERENCE_BD**.
Pour cette procédure, nous avons simplement afficher toujours les commandes en sortie de notre **SELECT … MINUS**.

```
--Requêtes Ji (PL/SQL Procédures-Fonctions ; pour vérifier la Cohérence globale des données ;
-->>>>>>>>>>>>>>> Qualité des données
CREATE OR REPLACE PROCEDURE COHERENCE_BD IS
        CURSOR curseur IS
        SELECT NUMCOM FROM COMMANDES
        MINUS (SELECT NUMCOM FROM DETAILCOM);

BEGIN
        DBMS_OUTPUT.PUT_LINE('Le(s) numéro(s) de commande vide(s) sont :');
        FOR i IN curseur LOOP
                DBMS_OUTPUT.PUT_LINE('-   ' || i.NUMCOM);
        END LOOP;
END;
/

EXECUTE COHERENCE_BD;
```

```
Le(s) numéro(s) de commande vide(s) sont :
-   20011RB
-   20184FB
```

Nous pouvons conclure que les commandes 20011RB et 2018AFB sont vides.

4.Dépendances fonctionnelles dans la table des clients

Pour les Dépendances fonctionnelles de la table CLIENTS, la méthode utiliser pour les *PaysVille* nous a été très utile car nous sommes partis sur le même principe. Elle renvoi False s'il n'y a pas de dépendances sinon True.

```
CREATE OR REPLACE VIEW LISTAVERIFIER_C (VILLE, PAYS) AS SELECT DISTINCT VILCLI, PAYSCLI FROM CLIENTS;

SELECT * FROM LISTAVERIFIER_C ;

CREATE OR REPLACE VIEW VERIFDF_C (VILLE, NBR) AS
SELECT VILLE, COUNT(*) AS NBR FROM LISTAVERIFIER_C GROUP BY VILLE ORDER BY VILLE;
SELECT * FROM VERIFDF_C ;

-- REM Si MAXOCCUR > 1 Alors la DF n'est pas vérifiée !

CREATE OR REPLACE FUNCTION VerifDFC( p_COLONNE1 IN VARCHAR2, p_COLONNE2 IN VARCHAR2 )
RETURN VARCHAR2
IS
    RESULT VARCHAR2(50);
    MAXOC number(2);
    BEGIN

        SELECT MAX(NBR) AS MAXOCCUR into MAXOC FROM VERIFDF_C;
        SELECT CASE WHEN MAXOC> 1 THEN 'FALSE' ELSE 'TRUE' END INTO RESULT FROM DUAL ;

    RETURN(RESULT);
    END;
/

SELECT VerifDFC(VILCLI, PAYSCLI) as verificationDFC FROM CLIENTS;
```

5.Élimination des doubles-similaires

Pour l'élimination des doubles nous avons décidé d'utiliser notre fonction créée lors de notre devoir **DEDUPLICATION.sql**. Nous avons pu l'adapter facilement car elle était proche de notre cas.

```
CREATE OR REPLACE PROCEDURE GENKEY(ETAPE IN NUMBER) AS
    CURSOR C1 IS SELECT * FROM T;
    CURSOR C2 IS SELECT * FROM T;
    KEYWORD     VARCHAR2(200):='';
    KEYWORD2    VARCHAR2(200):='';
    BEGIN
        IF ETAPE = 1 THEN
            FOR RES IN C1
            LOOP
                KEYWORD := RES.NOMCLI||RES.PRENCLI||RES.MAILCLI||RES.VILCLI||RES.PAYSCLI;
                KEYWORD := REPLACE(KEYWORD,' ','');
                KEYWORD := REPLACE(KEYWORD,'-','');
                KEYWORD := REPLACE(KEYWORD,'/','');
                EXECUTE IMMEDIATE 'UPDATE T SET KeyWordsCli= '||chr(39)||UPPER(KEYWORD)||chr(39)||
                                  ' WHERE CODCLI ='||chr(39)||RES.CODCLI||chr(39);
            END LOOP;
        END IF;
        IF ETAPE = 2 THEN
            FOR RES IN C1
            LOOP
                FOR RES2 IN C2
                LOOP
                    IF C2%ROWCOUNT > C1%ROWCOUNT THEN
                        IF UTL_MATCH.EDIT_DISTANCE_SIMILARITY(RES.KeyWordsCli,RES2.KeyWordsCli) > 70  THEN
                            KEYWORD := RES.NOMCLI||RES.PRENCLI||RES.MAILCLI;
                            KEYWORD2 := RES2.NOMCLI||RES2.PRENCLI||RES2.MAILCLI;
                            IF UTL_MATCH.EDIT_DISTANCE_SIMILARITY(RES.VILCLI,RES2.VILCLI) < 60 THEN
                                KEYWORD := KEYWORD||RES.VILCLI;
                                KEYWORD2 := KEYWORD2||RES2.VILCLI;
                            END IF;
                            IF UTL_MATCH.EDIT_DISTANCE_SIMILARITY(RES.PAYSCLI,RES2.PAYSCLI) < 60 THEN
                                KEYWORD := KEYWORD||RES.PAYSCLI;
                                KEYWORD2 := KEYWORD2||RES2.PAYSCLI;
                            END IF;
                            KEYWORD := REPLACE(KEYWORD,' ','');
                            KEYWORD := REPLACE(KEYWORD,'-','');
                            KEYWORD := REPLACE(KEYWORD,'/','');
                            KEYWORD2 := REPLACE(KEYWORD2,' ','');
                            KEYWORD2 := REPLACE(KEYWORD2,'-','');
                            KEYWORD2 := REPLACE(KEYWORD2,'/','');
                            EXECUTE IMMEDIATE 'UPDATE T SET KeyWordsCli2= '||chr(39)||UPPER(KEYWORD)||chr(39)||
                                              ' WHERE CODCLI ='||chr(39)||RES.CODCLI||chr(39);
                            EXECUTE IMMEDIATE 'UPDATE T SET KeyWordsCli2= '||chr(39)||UPPER(KEYWORD2)||chr(39)||
                                              ' WHERE CODCLI ='||chr(39)||RES2.CODCLI||chr(39);
                        END IF;
                    END IF;
                END LOOP;
            END LOOP;
        END IF;
    END;
/
```

La procédure **GENKEY** permet de générer une *KeyWord* à partir d'attribut choisit par nos soins qui nous permettra de décider si deux clients sont les mêmes. En fonction de taux de similarité, nous choisissons nous attribut. Par exemple pour le plus haut taux de similarité nous avons choisi *Nom, Prénom, Mail, Pays et Ville*. Notre taux de similarité est défini à l'aide de la fonction **EDIT_DISTANCE**.

La Procédure **DEDUP** ci-dessous décidera qui est doublons-similaire ou non.

```
CREATE OR REPLACE FUNCTION DEDUP(ETAPE IN NUMBER, SEUIL IN NUMBER)
RETURN NUMBER
AS
    CURSOR C1 IS SELECT * FROM T;
    CURSOR C2 IS SELECT * FROM T;
    NOM         VARCHAR2(30);
    PRENOM      VARCHAR2(30);
    MAIL        VARCHAR2(30);
    VILLEN      VARCHAR2(30);
    PAYSN       VARCHAR2(30);
    SCORE       NUMBER(5);
    SCORE2      NUMBER(5);
    TEST        NUMBER(5):=0;
    RESULT      NUMBER(5) := 0;
    BEGIN
        IF ETAPE = 1 THEN
            FOR REC1 IN C1 LOOP
                FOR REC2 IN C2 LOOP
                    IF C2%ROWCOUNT > C1%ROWCOUNT THEN
                        EXECUTE IMMEDIATE 'SELECT UTL_MATCH.EDIT_DISTANCE_SIMILARITY('||chr(39)||REC1.KeyWordsCli||chr(39)||
                                                ','||chr(39)||REC2.KeyWordsCli||chr(39)||
                                                ') FROM DUAL' INTO SCORE;
                        IF SCORE >= SEUIL THEN
                            IF LENGTH(REC1.NOMCLI) >  LENGTH(REC2.NOMCLI) OR REC2.NOMCLI IS NULL THEN
                                NOM := REC1.NOMCLI;
                            ELSE
                                NOM := REC2.NOMCLI;
                            END IF;
                            IF LENGTH(REC1.PRENCLI) >  LENGTH(REC2.PRENCLI) OR REC2.PRENCLI IS NULL THEN
                                PRENOM := REC1.PRENCLI;
                            ELSE
                                PRENOM := REC2.PRENCLI;
                            END IF;
                            IF LENGTH(REC1.MAILCLI) >  LENGTH(REC2.MAILCLI) OR REC2.MAILCLI IS NULL THEN
                                MAIL := REC1.MAILCLI;
                            ELSE
                                MAIL := REC2.MAILCLI;
                            END IF;
                            IF LENGTH(REC1.VILCLI) >  LENGTH(REC2.VILCLI) OR REC2.VILCLI IS NULL THEN
                                VILLEN := REC1.VILCLI;
                            ELSE
                                VILLEN := REC2.VILCLI;

                            END IF;
                            IF LENGTH(REC1.PAYSCLI) >  LENGTH(REC2.PAYSCLI) OR REC2.PAYSCLI IS NULL THEN
                                PAYSN := REC1.PAYSCLI;
                            ELSE
                                PAYSN := REC2.PAYSCLI;
                            END IF;
                            EXECUTE IMMEDIATE 'UPDATE T SET NOMCLI='    ||chr(39)||NOM||chr(39)||
                                        ',PRENCLI='  ||chr(39)||PRENOM||chr(39)||
                                        ', MAILCLI=' ||chr(39)||MAIL||chr(39)||
                                        ', VILCLI='  ||chr(39)||VILLEN||chr(39)||
                                        ', PAYSCLI=' ||chr(39)||PAYSN||chr(39)||
                                    ' WHERE CODCLI='    ||chr(39)||REC1.CODCLI ||chr(39);
```

Comme pour la *Keyword*, nous avons procédé à l'élimination des doublons en 2 étapes (Taux Fort puis Medium) à partir d'**EDIT_DISTANCE**.

```
                        EXECUTE IMMEDIATE 'DELETE FROM T WHERE CODCLI='||chr(39)||REC2.CODCLI ||chr(39);
                        RESULT := 1;

                        dbms_output.put_line(REC1.KeyWordsCli||'    =?    '||REC2.KeyWordsCli|| '   =   ' || SCORE);
                    END IF;
                END IF;
            END LOOP;
        END LOOP;
    END IF;
    IF ETAPE = 2 THEN
        FOR REC1 IN C1 LOOP
            FOR REC2 IN C2 LOOP
                IF C2%ROWCOUNT > C1%ROWCOUNT THEN

                    EXECUTE IMMEDIATE 'SELECT UTL_MATCH.EDIT_DISTANCE_SIMILARITY('||chr(39)||REC1.KeyWordsCli2||chr(39)||
                                            ','||chr(39)||REC2.KeyWordsCli2||chr(39)||
                                            ') FROM DUAL' INTO SCORE;
                    IF SCORE >= SEUIL THEN
                        IF (TO_CHAR(REC1.MAILCLI) = TO_CHAR(REC2.MAILCLI)) AND REC2.MAILCLI IS NOT NULL THEN

                            EXECUTE IMMEDIATE 'SELECT UTL_MATCH.EDIT_DISTANCE_SIMILARITY('||chr(39)||REC2.MAILCLI||chr(39)||
                                                    ','||chr(39)||REC1.MAILCLI||chr(39)||
                                                    ') FROM DUAL' INTO SCORE2;
                            DBMS_OUTPUT.PUT_LINE(REC1.MAILCLI || '   ?=   ' || REC2.MAILCLI || '  =  ' || SCORE2);
                            IF (REGEXP_LIKE (SUBSTR(REC2.NOMCLI,1,2),'^[A-Za-z][.]+$')
                                OR REGEXP_LIKE (SUBSTR(REC1.NOMCLI,1,2),'^[A-Za-z][.]+$'))
                                AND  SUBSTR(REC1.NOMCLI,1,1) = SUBSTR(REC2.NOMCLI,1,1)
                             THEN
                                DBMS_OUTPUT.PUT_LINE(REC1.NOMCLI || '   ?=    ' || REC2.NOMCLI);
                                IF LENGTH(REC1.NOMCLI) >  LENGTH(REC2.NOMCLI) OR REC2.NOMCLI IS NULL THEN
                                    NOM := REC1.NOMCLI;
                                ELSE
                                    NOM := REC2.NOMCLI;
                                END IF;
                                TEST := TEST+1;
                            END IF;
                            IF (REGEXP_LIKE (SUBSTR(REC2.PRENCLI,1,2),'^[A-Za-z][.]+$')
                            OR REGEXP_LIKE (SUBSTR(REC1.PRENCLI,1,2),'^[A-Za-z][.]+$'))
                                AND  SUBSTR(REC1.PRENCLI,1,1) = SUBSTR(REC2.PRENCLI,1,1)
                             THEN
                                DBMS_OUTPUT.PUT_LINE(REC1.PRENCLI || '   ?=    ' || REC2.PRENCLI);
                                IF LENGTH(REC1.PRENCLI) >  LENGTH(REC2.PRENCLI) OR REC2.PRENCLI IS NULL THEN
                                    PRENOM := REC1.PRENCLI;
                                ELSE
                                    PRENOM := REC2.PRENCLI;
                                END IF;
                                TEST := TEST+1;
                            END IF;
```

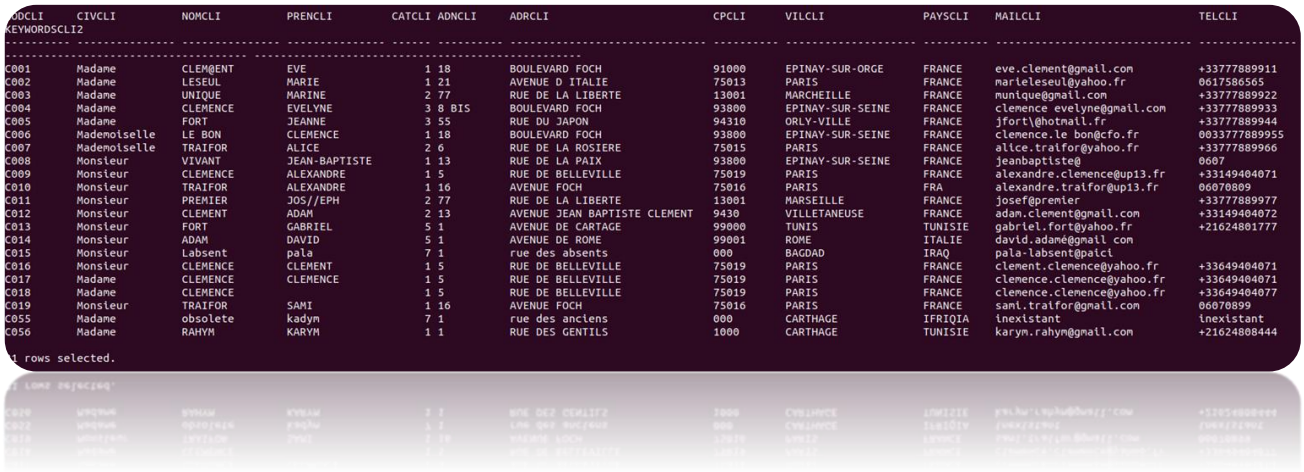
```
                    IF UTL_MATCH.EDIT_DISTANCE_SIMILARITY(REC1.NOMCLI,REC2.NOMCLI) >= 70 THEN
                        EXECUTE IMMEDIATE 'SELECT UTL_MATCH.EDIT_DISTANCE_SIMILARITY('||chr(39)||REC2.NOMCLI||chr(39)||
                                                ','||chr(39)||REC1.NOMCLI||chr(39)||
                                                ') FROM DUAL' INTO SCORE2;
                        DBMS_OUTPUT.PUT_LINE(REC1.NOMCLI || '   ?=    ' || REC2.NOMCLI || '  =  ' || SCORE2);
                        IF LENGTH(REC1.NOMCLI) >  LENGTH(REC2.NOMCLI) OR REC2.NOMCLI IS NULL THEN
                            NOM := REC1.NOMCLI;
                        ELSE
                            NOM := REC2.NOMCLI;
                        END IF;
                        TEST := TEST+1;
                    END IF;
                    IF UTL_MATCH.EDIT_DISTANCE_SIMILARITY(REC1.PRENCLI,REC2.PRENCLI) >= 70 THEN
                        EXECUTE IMMEDIATE 'SELECT UTL_MATCH.EDIT_DISTANCE_SIMILARITY('||chr(39)||REC2.PRENCLI||chr(39)||
                                                ','||chr(39)||REC1.PRENCLI||chr(39)||
                                                ') FROM DUAL' INTO SCORE2;
                        DBMS_OUTPUT.PUT_LINE(REC1.PRENCLI || '   ?=    ' || REC2.PRENCLI || '  =  ' || SCORE2);
                        IF LENGTH(REC1.PRENCLI) >  LENGTH(REC2.PRENCLI) OR REC2.PRENCLI IS NULL THEN
                            PRENOM := REC1.PRENCLI;
                        ELSE
                            PRENOM := REC2.PRENCLI;
                        END IF;
                        TEST:=TEST+1;
                    END IF;
                    IF TEST >= 2 THEN
                        IF LENGTH(REC1.MAILCLI) >  LENGTH(REC2.MAILCLI) OR REC2.MAILCLI IS NULL THEN
                            MAIL := REC1.MAILCLI;
                        ELSE
                            MAIL := REC2.MAILCLI;
                        END IF;
                        IF LENGTH(REC1.VILCLI) >  LENGTH(REC2.VILCLI) OR REC2.VILCLI IS NULL THEN
                            VILLEN := REC1.VILCLI;
                        ELSE
                            VILLEN := REC2.VILCLI;

                        END IF;
                        IF LENGTH(REC1.PAYSCLI) >  LENGTH(REC2.PAYSCLI) OR REC2.PAYSCLI IS NULL THEN
                            PAYSN := REC1.PAYSCLI;
                        ELSE
                            PAYSN := REC2.PAYSCLI;
                        END IF;
                        EXECUTE IMMEDIATE 'UPDATE T SET NOMCLI='    ||chr(39)||NOM    ||chr(39)||
                                        ',PRENCLI='    ||chr(39)||PRENOM ||chr(39)||
                                        ', MAILCLI=' ||chr(39)||MAIL||chr(39)||
                                        ', VILCLI='    ||chr(39)||VILLEN ||chr(39)||
                                        ', PAYSCLI='     ||chr(39)||PAYSN  ||chr(39)||
                                ' WHERE CODCLI='||chr(39)||REC1.CODCLI ||chr(39);

                        EXECUTE IMMEDIATE 'DELETE FROM T WHERE CODCLI='||chr(39)||REC2.CODCLI||chr(39);
                        RESULT := 1;
                    END IF;
                END IF;
            END IF;
        END IF;
    END LOOP;
```

La Procédure **DEDUPLICATIONF** ci-dessus est le « **main** » de l'élimination des doubles. Pour la première étape nous choisissons 85% de similarité et 50 pour la deuxième.

```
            END LOOP;
        END IF;
    RETURN RESULT;
    END;
/

SELECT * FROM T ORDER BY CODCLI;
PROMPT
pause Tapez sur Enter...
PROMPT
CREATE OR REPLACE PROCEDURE DEDUPLICATIONF
IS
    x number;
    BEGIN
        GENKEY(1);
        x:= DEDUP(1,85);
        WHILE x = 1
        LOOP
            x:= DEDUP(1,85);
        END LOOP;

        GENKEY(2);
        x:= DEDUP(2,50);
        WHILE x = 1
        LOOP
            x:= DEDUP(2,50);
        END LOOP;


    END;
/

EXEC DEDUPLICATIONF;

ALTER TABLE T DROP (KeyWordsCli, KeyWordsCli2);
SELECT * FROM T ORDER BY NOMCLI;
```

Voici les résultats de notre déduplication :

Avant élimination :

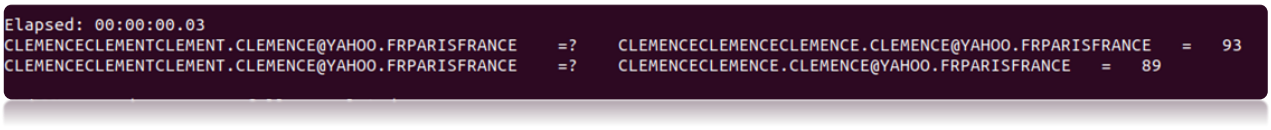


CODCLI	CIVCLI	NOMCLI	PRENCLI	CATCLI	ADNCLI	ADRCLI	CPCLI	VILCLI	PAYSCLI	MAILCLI	TELCLI	KEYWORDSCLI2
C001	Madame	CLEM@ENT	EVE	1	18	BOULEVARD FOCH	91000	EPINAY-SUR-ORGE	FRANCE	eve.clement@gmail.com	+33777889911	
C002	Madame	LESEUL	MARIE	1	21	AVENUE D ITALIE	75013	PARIS	FRANCE	marieleseul@yahoo.fr	0617586565	
C003	Madame	UNIQUE	MARINE	2	77	RUE DE LA LIBERTE	13001	MARCHEILLE	FRANCE	munique@gmail.com	+33777889922	
C004	Madame	CLEMENCE	EVELYNE	3	8 BIS	BOULEVARD FOCH	93800	EPINAY-SUR-SEINE	FRANCE	clemence evelyne@gmail.com	+33777889933	
C005	Madame	FORT	JEANNE	3	55	RUE DU JAPON	94310	ORLY-VILLE	FRANCE	jfort\@hotmail.fr	+33777889944	
C006	Mademoiselle	LE BON	CLEMENCE	1	18	BOULEVARD FOCH	93800	EPINAY-SUR-SEINE	FRANCE	clemence.le bon@cfo.fr	0033777889955	
C007	Mademoiselle	TRAIFOR	ALICE	2	6	RUE DE LA ROSIERE	75015	PARIS	FRANCE	alice.traifor@yahoo.fr	+33777889966	
C008	Monsieur	VIVANT	JEAN-BAPTISTE	1	13	RUE DE LA PAIX	93800	EPINAY-SUR-SEINE	FRANCE	jeanbaptiste@	0607	
C009	Monsieur	CLEMENCE	ALEXANDRE	1	5	RUE DE BELLEVILLE	75019	PARIS	FRANCE	alexandre.clemence@up13.fr	+33149404071	
C010	Monsieur	TRAIFOR	ALEXANDRE	1	16	AVENUE FOCH	75016	PARIS	FRA	alexandre.traifor@up13.fr	06070809	
C011	Monsieur	PREMIER	JOS//EPH	2	77	RUE DE LA LIBERTE	13001	MARSEILLE	FRANCE	josef@premier	+33777889977	
C012	Monsieur	CLEMENT	ADAM	2	13	AVENUE JEAN BAPTISTE CLEMENT	9430	VILLETANEUSE	FRANCE	adam.clement@gmail.com	+33149404072	
C013	Monsieur	FORT	GABRIEL	5	1	AVENUE DE CARTAGE	99000	TUNIS	TUNISIE	gabriel.fort@yahoo.fr	+21624801777	
C014	Monsieur	ADAM	DAVID	5	1	AVENUE DE ROME	99001	ROME	ITALIE	david.adam@gmail com		
C015	Monsieur	Labsent	pala	7	1	rue des absents	000	BAGDAD	IRAQ	pala-labsent@paici		
C016	Monsieur	CLEMENCE	CLEMENT	1	5	RUE DE BELLEVILLE	75019	PARIS	FRANCE	clement.clemence@yahoo.fr	+33649404071	
C017	Madame	CLEMENCE	CLEMENCE	1	5	RUE DE BELLEVILLE	75019	PARIS	FRANCE	clemence.clemence@yahoo.fr	+33649404071	
C018	Madame	CLEMENCE		1	5	RUE DE BELLEVILLE	75019	PARIS	FRANCE	clemence.clemence@yahoo.fr	+33649404077	
C019	Monsieur	TRAIFOR	SAMI	1	16	AVENUE FOCH	75016	PARIS	FRANCE	sami.traifor@gmail.com	06070899	
C055	Madame	obsolete	kadym	7	1	rue des anciens	000	CARTHAGE	IFRIQIA	inexistant	inexistant	
C056	Madame	RAHYM	KARYM	1	1	RUE DES GENTILS	1000	CARTHAGE	TUNISIE	karym.rahym@gmail.com	+21624808444	

21 rows selected.

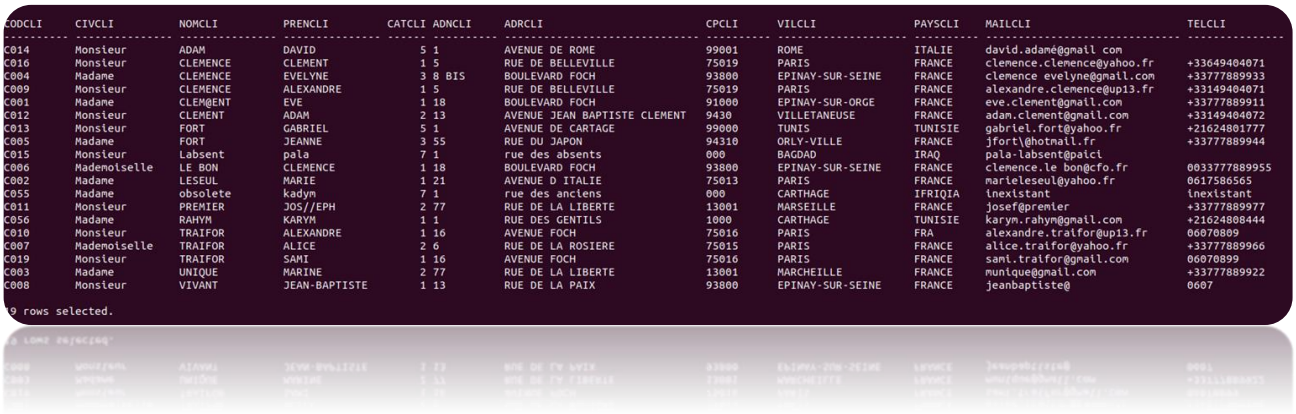
21 Lignes

Pendant élimination :



```
Elapsed: 00:00:00.03
CLEMENCECLEMENTCLEMENT.CLEMENCE@YAHOO.FRPARISFRANCE    =?    CLEMENCECLEMENCECLEMENCE.CLEMENCE@YAHOO.FRPARISFRANCE   =   93
CLEMENCECLEMENTCLEMENT.CLEMENCE@YAHOO.FRPARISFRANCE    =?    CLEMENCECLEMENCE.CLEMENCE@YAHOO.FRPARISFRANCE   =   89
```

Après élimination :



CODCLI	CIVCLI	NOMCLI	PRENCLI	CATCLI	ADNCLI	ADRCLI	CPCLI	VILCLI	PAYSCLI	MAILCLI	TELCLI
C014	Monsieur	ADAM	DAVID	5	1	AVENUE DE ROME	99001	ROME	ITALIE	david.adam@gmail com	
C016	Monsieur	CLEMENCE	CLEMENT	1	5	RUE DE BELLEVILLE	75019	PARIS	FRANCE	clemence.clemence@yahoo.fr	+33649404071
C004	Madame	CLEMENCE	EVELYNE	3	8 BIS	BOULEVARD FOCH	93800	EPINAY-SUR-SEINE	FRANCE	clemence evelyne@gmail.com	+33777889933
C009	Monsieur	CLEMENCE	ALEXANDRE	1	5	RUE DE BELLEVILLE	75019	PARIS	FRANCE	alexandre.clemence@up13.fr	+33149404071
C001	Madame	CLEM@ENT	EVE	1	18	BOULEVARD FOCH	91000	EPINAY-SUR-ORGE	FRANCE	eve.clement@gmail.com	+33777889911
C012	Monsieur	CLEMENT	ADAM	2	13	AVENUE JEAN BAPTISTE CLEMENT	9430	VILLETANEUSE	FRANCE	adam.clement@gmail.com	+33149404072
C013	Monsieur	FORT	GABRIEL	5	1	AVENUE DE CARTAGE	99000	TUNIS	TUNISIE	gabriel.fort@yahoo.fr	+21624801777
C005	Madame	FORT	JEANNE	3	55	RUE DU JAPON	94310	ORLY-VILLE	FRANCE	jfort\@hotmail.fr	+33777889944
C015	Monsieur	Labsent	pala	7	1	rue des absents	000	BAGDAD	IRAQ	pala-labsent@paici	
C006	Mademoiselle	LE BON	CLEMENCE	1	18	BOULEVARD FOCH	93800	EPINAY-SUR-SEINE	FRANCE	clemence.le bon@cfo.fr	0033777889955
C002	Madame	LESEUL	MARIE	1	21	AVENUE D ITALIE	75013	PARIS	FRANCE	marieleseul@yahoo.fr	0617586565
C055	Madame	obsolete	kadym	7	1	rue des anciens	000	CARTHAGE	IFRIQIA	inexistant	inexistant
C011	Monsieur	PREMIER	JOS//EPH	2	77	RUE DE LA LIBERTE	13001	MARSEILLE	FRANCE	josef@premier	+33777889977
C056	Madame	RAHYM	KARYM	1	1	RUE DES GENTILS	1000	CARTHAGE	TUNISIE	karym.rahym@gmail.com	+21624808444
C010	Monsieur	TRAIFOR	ALEXANDRE	1	16	AVENUE FOCH	75016	PARIS	FRA	alexandre.traifor@up13.fr	06070809
C007	Mademoiselle	TRAIFOR	ALICE	2	6	RUE DE LA ROSIERE	75015	PARIS	FRANCE	alice.traifor@yahoo.fr	+33777889966
C019	Monsieur	TRAIFOR	SAMI	1	16	AVENUE FOCH	75016	PARIS	FRANCE	sami.traifor@gmail.com	06070899
C003	Madame	UNIQUE	MARINE	2	77	RUE DE LA LIBERTE	13001	MARCHEILLE	FRANCE	munique@gmail.com	+33777889922
C008	Monsieur	VIVANT	JEAN-BAPTISTE	1	13	RUE DE LA PAIX	93800	EPINAY-SUR-SEINE	FRANCE	jeanbaptiste@	0607

19 rows selected.

19 Lignes

6. Système de recommandation

Dans cette partie nous allons créer un début de système de recommandation.

6.1 Avant-Première

```
CREATE OR REPLACE VIEW V_CLIARTQTE (CLIENT, ARTICLE, QUANTITE) AS
SELECT K.CODCLI, D.REFART, SUM(TO_NUMBER(D.QTCOM)) FROM COMMANDES K, DETAILCOM D
WHERE K.NUMCOM = D.NUMCOM
AND K.DATCOM >= '01-SEPTEMBER-2018'
AND K.DATCOM <= '30-SEPTEMBER-2018'
GROUP BY K.CODCLI, D.REFART
ORDER BY 1, 2;

SELECT * FROM V_CLIARTQTE ;
```

Cette procédure crée une vue qui contient le nombre de chaque article par client commandé du 01/09/2018 au 30/09/2018.

Voici le contenue final après exécution de la procédure :

SELECT * FROM V_CLIARTQTE

CLIENT	ARTICLE	QUANTITE
C001	FB.001	8
C001	FB.002	3
C001	FB.003	1
C001	F1.004	1
C001	F1.005	1
C002	FB.001	1
C002	FB.002	1
C002	FB.003	1
C003	FB.001	5
C003	FB.002	2
C003	FB.003	1
C003	F1.005	1
C004	FB.002	3
C004	FB.003	3
C004	F1.004	3
C004	F1.005	4
C006	F1.006	1
C006	F1.007	1
C006	F1.008	1
C007	F1.006	1
C007	F1.007	1
C009	FB.001	5
C009	F1.009	2
C009	F1.010	2
C010	FB.001	1
C010	F1.009	2
C011	FB.001	2
C011	F1.009	2
C011	F1.010	1
C012	FB.001	7
C013	FB.001	3
C013	FB.002	1
C013	F1.006	4
C014	FB.001	2
C014	FB.002	1
C014	F1.006	4
C014	F1.007	5
C016	F1.004	3
C016	F1.006	1
C016	F1.009	1
C017	F1.004	1
C017	F1.009	1
C019	F1.011	1

43 rows selected.

A l'aide de la procédure créée précédemment, nous avons créé une vue par client pour pouvoir les comparer par la suite.

```
-- Pour tout client Ci AYANT SATISFAIT certains critères !
-- Les articles commandés par le client C001.
CREATE OR REPLACE VIEW V_C001 AS
SELECT * FROM V_CLIARTQTE
WHERE CLIENT = 'C001';

-- Les articles commandés par le client C002.
CREATE OR REPLACE VIEW V_C002 AS
SELECT * FROM V_CLIARTQTE
WHERE CLIENT = 'C002';
-- Les articles commandés par le client C003.
CREATE OR REPLACE VIEW V_C003 AS
SELECT * FROM V_CLIARTQTE
WHERE CLIENT = 'C003';
-- Etc...
-- Les articles commandés par le client C006.
CREATE OR REPLACE VIEW V_C006 AS
SELECT * FROM V_CLIARTQTE
WHERE CLIENT = 'C006';
```

Nous pouvons maintenant faire l'intersection des différents articles commandés par nos clients.

Par exemple entre les Clients C001 et C002 :

```
CREATE OR REPLACE VIEW V_C001_C002_A (ARTICLE) AS
SELECT ARTICLE FROM V_C001
INTERSECT
SELECT ARTICLE FROM V_C002;
SELECT * FROM V_C001_C002_A;

CREATE OR REPLACE VIEW V_C001_C002_A (ARTICLE) AS
SELECT ARTICLE FROM V_C001
WHERE ARTICLE IN (SELECT ARTICLE FROM V_C002);
SELECT * FROM V_C001_C002_A;

-- Les articles commandés, à la fois, par les clients C001 et C002.
-- 2ème Solution : Les articles commandés, à la fois, par les clients C001 et C002.
CREATE OR REPLACE VIEW V_C001_C002_B (CLIG, ARTG, QTEG, CLID, ARTD, QTED) AS
SELECT X.CLIENT, X.ARTICLE, X.QUANTITE, Y.CLIENT, Y.ARTICLE, Y.QUANTITE
FROM V_C001 X, V_C002 Y
WHERE X.ARTICLE = Y.ARTICLE ;
-- Remarque : Que faut-il faire pour les articles qui sont commandés par l'un mais pas par l'autre?

-- Les articles commandés, à la fois, par les clients C001 et C003.
-- 2ème Solution : Les articles commandés, à la fois, par les clients C001 et C003.
CREATE OR REPLACE VIEW V_C001_C003_B (CLIG, ARTG, QTEG, CLID, ARTD, QTED) AS
SELECT X.CLIENT, X.ARTICLE, X.QUANTITE, Y.CLIENT, Y.ARTICLE, Y.QUANTITE
FROM V_C001 X, V_C002 Y
WHERE X.ARTICLE = Y.ARTICLE ;
```

Grace à cela, on pourra proposer au client C002 des articles que le client C001 a acheter en plus et cela réciproquement.

```
SELECT ARTICLE FROM V_C001
MINUS
SELECT ARTICLE FROM V_C002;

-- On recommande (on propose) a
SELECT ARTICLE FROM V_C002
MINUS
SELECT ARTICLE FROM V_C001;
```

Par exemple :

Acheté par C001 et C002	Acheté par C001	Proposé a C002
FB.001 FB.002 FB.003	FB.001 FB.002 FB.003 F1.004 F1.005	F1.004 F1.005

6.2. Première proposition/solution

Les procédures suivantes sont une pseudo généralisation des précédentes méthodes, elle créé une vue par client. Il ne s'agit plus de fixer une date d'échantillon mais plutôt de manière générale avec tous les client cette fois-ci.

```
CREATE OR REPLACE VIEW V_CLIARTQTE (CLIENT, ARTICLE, QUANTITE) AS
SELECT K.CODCLI, D.REFART, SUM(D.QTCOM) FROM COMMANDES K, DETAILCOM D
WHERE K.NUMCOM = D.NUMCOM
GROUP BY K.CODCLI, D.REFART
ORDER BY 1, 2;

CREATE OR REPLACE PROCEDURE VueCLIART AS
    CURSOR curseur IS SELECT CLIENT FROM V_CLIARTQTE;
    BEGIN
        FOR i IN curseur LOOP
            EXECUTE IMMEDIATE 'CREATE OR REPLACE VIEW V_'||i.CLIENT||'
            AS SELECT * FROM V_CLIARTQTE WHERE CLIENT = '''||i.CLIENT||'''' ;
        END LOOP;
    END;
/
EXEC VueCLIART;
```

```
CREATE OR REPLACE PROCEDURE VueCOCLI AS
    BEGIN
        FOR i IN (SELECT DISTINCT CLIENT FROM V_CLIARTQTE ORDER BY CLIENT) LOOP
            For j In (SELECT DISTINCT CLIENT FROM V_CLIARTQTE WHERE client>i.client ORDER BY CLIENT)
            LOOP
                EXECUTE IMMEDIATE 'CREATE OR REPLACE VIEW V_'||i.CLIENT||'_'||j.CLIENT||'_A
                                   AS SELECT ARTICLE FROM V_'||i.CLIENT||'
                                       WHERE ARTICLE IN (SELECT ARTICLE FROM V_'||j.CLIENT||')';
            END LOOP;
        END LOOP;
    END;
/
EXEC VueCOCLI;
```

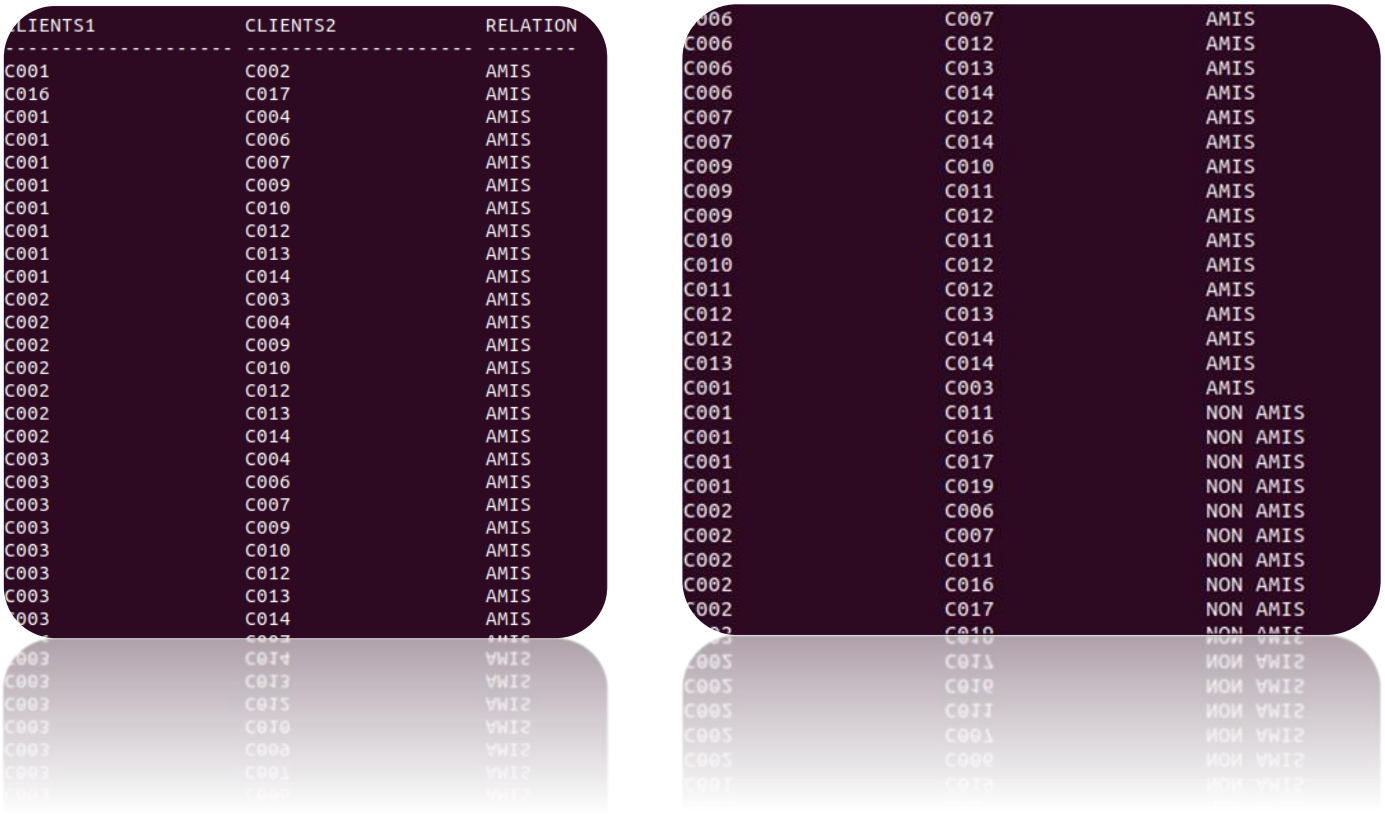
Ainsi, on peut avoir les clients ayant des articles en commun avec le nombre de ceux-ci.

```
DROP TABLE ARTCOMM;
CREATE table ARTCOMM(CLIENTS1 VARCHAR2(20), CLIENTS2 VARCHAR2(20), NOMBRE NUMERIC(3));

CREATE OR REPLACE PROCEDURE VueARTCOM AS
    BEGIN

        FOR i IN (SELECT DISTINCT CLIENT FROM V_CLIARTQTE ORDER BY CLIENT ) LOOP
            For j In (SELECT DISTINCT CLIENT FROM V_CLIARTQTE WHERE client>i.client ORDER BY
client)LOOP
            EXECUTE IMMEDIATE 'INSERT INTO ARTCOMM(CLIENTS1,CLIENTS2,NOMBRE)
            VALUES ('''||i.CLIENT||''','''||j.CLIENT||''',(SELECT count(*) FROM V_'||i.CLIENT||'_'||
j.CLIENT||'_A))';
            END LOOP;
        END LOOP;
    END;
/
EXEC VueARTCOM;
```

Si le nombre d'article en commun dépasse un certain seuil, on pourra définir s'ils sont amis (dépendances de lien/besoin/achat) ou non amis.

```
SELECT CLIENTS1,CLIENTS2,(CASE WHEN NOMBRE>1 THEN 'AMIS' ELSE 'NON AMIS' END) AS RELATION FROM
artcoMM
ORDER BY (CASE WHEN NOMBRE>1 THEN 'AMIS' ELSE 'NON AMIS' END);
```

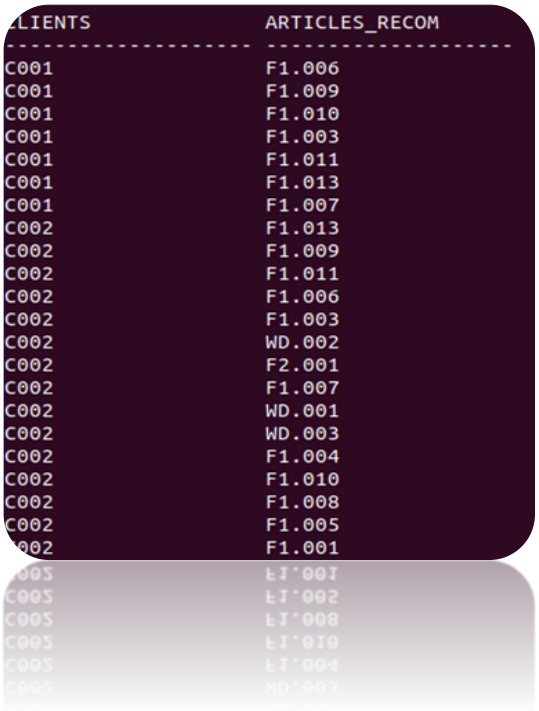
Voici une partie de la vue :



CLIENTS1	CLIENTS2	RELATION
C001	C002	AMIS
C016	C017	AMIS
C001	C004	AMIS
C001	C006	AMIS
C001	C007	AMIS
C001	C009	AMIS
C001	C010	AMIS
C001	C012	AMIS
C001	C013	AMIS
C001	C014	AMIS
C002	C003	AMIS
C002	C004	AMIS
C002	C009	AMIS
C002	C010	AMIS
C002	C012	AMIS
C002	C013	AMIS
C002	C014	AMIS
C003	C004	AMIS
C003	C006	AMIS
C003	C007	AMIS
C003	C009	AMIS
C003	C010	AMIS
C003	C012	AMIS
C003	C013	AMIS
C003	C014	AMIS
C006	C007	AMIS
C006	C012	AMIS
C006	C013	AMIS
C006	C014	AMIS
C007	C012	AMIS
C007	C014	AMIS
C009	C010	AMIS
C009	C011	AMIS
C009	C012	AMIS
C010	C011	AMIS
C010	C012	AMIS
C011	C012	AMIS
C012	C013	AMIS
C012	C014	AMIS
C013	C014	AMIS
C001	C003	AMIS
C001	C011	NON AMIS
C001	C016	NON AMIS
C001	C017	NON AMIS
C001	C019	NON AMIS
C002	C006	NON AMIS
C002	C007	NON AMIS
C002	C011	NON AMIS
C002	C016	NON AMIS
C002	C017	NON AMIS

Donc on a la procédure qui recommande les articles aux clients de la même façon expliquer précédemment.

```
DROP TABLE RECOMMANDATION;
CREATE table RECOMMANDATION(CLIENTS VARCHAR2(20), ARTICLES_recom VARCHAR2(20));

CREATE OR REPLACE PROCEDURE Recom_aRT AS
    BEGIN

        FOR i IN (SELECT DISTINCT CLIENT FROM V_CLIARTQTE ORDER BY CLIENT ) LOOP
            For j In (SELECT DISTINCT CLIENT FROM V_CLIARTQTE WHERE client<>i.client ORDER BY
client) LOOP

            EXECUTE IMMEDIATE 'INSERT INTO RECOMMandation
                (SELECT '''||i.CLIENT||''',ARTICLE FROM V_'||j.CLIENT||'
                 MINUS SELECT '''||i.CLIENT||''',ARTICLE FROM V_'||i.CLIENT||')';

            END LOOP;
        END LOOP;
    END;
/
EXEC Recom_aRT;

SELECT DISTINCT * FROM Recommandation ORDER BY CLIENTS;
```



LIENTS	ARTICLES_RECOM
C001	F1.006
C001	F1.009
C001	F1.010
C001	F1.003
C001	F1.011
C001	F1.013
C001	F1.007
C002	F1.013
C002	F1.009
C002	F1.011
C002	F1.006
C002	F1.003
C002	WD.002
C002	F2.001
C002	F1.007
C002	WD.001
C002	WD.003
C002	F1.004
C002	F1.010
C002	F1.008
C002	F1.005
002	F1.001

Voici les articles recommander pour C001 et C002 par rapport à tous les autres clients qui sont considéré comme leur « Amis »

On remarque qu'il y a beaucoup plus d'article recommander et que cela varie en fonction du seuil définissant leur « Amitié ». Plus celui-ci est bas plus ils auront d'article recommandé inversement plus celui-ci est haut moins ils en auront.

6.3. Deuxième proposition/solution

```
CREATE OR REPLACE VIEW CombinaisonClient(Client1, Client2) AS SELECT A.codCli, B.codCli FROM clients
A, clients B WHERE A.codCli < B.codCli;

CREATE OR REPLACE PROCEDURE vueArticleDuClient(codeClient IN VARCHAR, dateDebut IN VARCHAR, dateFin
IN VARCHAR)
as
    req VARCHAR(2000) := '';
    BEGIN
        req := 'CREATE OR REPLACE VIEW V_' || codeClient || '(CLIENT, ARTICLE, QUANTITE) AS SELECT
K.CODCLI, D.REFART, SUM(D.QTCOM)
        FROM COMMANDES K, DETAILCOM D WHERE K.NUMCOM = D.NUMCOM
        AND K.DATCOM >= ''' || dateDebut || ''' AND K.DATCOM <= ''' || dateFin || '''
        AND K.CODCLI = ''' || codeClient || ''' GROUP BY K.CODCLI, D.REFART ORDER BY 1, 2';
        EXECUTE IMMEDIATE req;
    END;
/
```

Après avoir créé une vue avec la combinaison de tous les clients, on a une procédure (ci-dessus) créant une vue pour un client donné en paramètre contenant la quantité de chaque article commandé par celui-ci entre deux dates données en paramètre également.

```
CREATE OR REPLACE PROCEDURE vueArticleDesClients(codeClient1 IN VARCHAR, dateDebut1 IN VARCHAR,
dateFin1 IN VARCHAR, codeClient2 IN VARCHAR, dateDebut2 IN VARCHAR, dateFin2 IN VARCHAR)
as
    req VARCHAR(2000) := '';
    BEGIN
        vueArticleDuClient(codeClient1,dateDebut1,dateFin1);
        vueArticleDuClient(codeClient2,dateDebut2,dateFin2);
        req := 'CREATE OR REPLACE VIEW V_'||codeClient1||'_'||codeClient2||'_A (ARTICLE)
        AS SELECT ARTICLE FROM V_'||codeClient1||' INTERSECT SELECT ARTICLE FROM V_'||codeClient2;
        EXECUTE IMMEDIATE req;
    END;
/
```

La procédure ci-dessus permet de la création d'une vue sur deux clients en paramètre sur l'intersection de leurs articles commander sur le même intervalle ou non entré en paramètre.

Grâce à toutes nos procédures précédentes, nous pouvons désormais connaître les articles acheter par un A et non B sur une/des périodes données.

```
CREATE OR REPLACE PROCEDURE vueArticleNonDesClients(codeClient1 IN VARCHAR, dateDebut1 IN VARCHAR,
dateFin1 IN VARCHAR, codeClient2 IN VARCHAR, dateDebut2 IN VARCHAR, dateFin2 IN VARCHAR)
as
    req VARCHAR(2000) := '';
BEGIN
    vueArticleDuClient(codeClient1,dateDebut1,dateFin1);
    vueArticleDuClient(codeClient2,dateDebut2,dateFin2);
    req := 'CREATE OR REPLACE VIEW V_Non_'||codeClient1||'_'||codeClient2||'_A (ARTICLE)
    AS SELECT ARTICLE FROM V_'||codeClient1||' MINUS SELECT ARTICLE FROM V_'||codeClient2;
    EXECUTE IMMEDIATE req;
    req := 'CREATE OR REPLACE VIEW V_Non_'||codeClient2||'_'||codeClient1||'_A (ARTICLE)
    AS SELECT ARTICLE FROM V_'||codeClient2||' MINUS SELECT ARTICLE FROM V_'||codeClient1;
    EXECUTE IMMEDIATE req;
END;
/

PROMPT
pause Tapez sur Enter...
PROMPT

EXEC vueArticleNonDesClients('C001', '01-SEPTEMBER-2018' ,'30-SEPTEMBER-2018', 'C003', '01-
SEPTEMBER-2018' ,'30-SEPTEMBER-2018');
SELECT * FROM V_NON_C001_C003_A;
SELECT * FROM V_NON_C003_C001_A;
```

Exemple NON_C001_C003 :

```
SELECT * FROM V_NON_C001_C003_A

ARTICLE
----------
F1.004
```

Pour finir, il nous reste à
généraliser nos procédures à tous les clients et stocker la liste des articles à leurs recommandées

```
CREATE OR REPLACE PROCEDURE propositionArticle(codeClient1 IN VARCHAR, codeClient2 IN
VARCHAR,dateDebut IN VARCHAR, dateFin IN VARCHAR)
is
    Type curseurType IS REF CURSOR;
    monCurseur curseurType;
    requete VARCHAR(2000);
    v_Article VARCHAR(55);
    BEGIN
        vueArticleNonDesClients(codeClient1, dateDebut, dateFin, codeClient2, dateDebut, dateFin);
        requete := 'SELECT article FROM V_Non_' || codeClient1 || '_' || codeClient2 || '_A';
        OPEN monCurseur FOR requete;
            LOOP
                FETCH monCurseur INTO v_Article;
                EXECUTE IMMEDIATE 'INSERT INTO articleARecomder values ('''||codeClient1||''','''||
codeClient2||''','''' ||v_Article|| ''')';
                EXIT WHEN monCurseur%NOTFOUND;
            END LOOP;
        CLOSE monCurseur;

        requete := 'SELECT article FROM V_Non_' || codeClient2 || '_' || codeClient1 || '_A';
        OPEN monCurseur FOR requete;
            LOOP
                FETCH monCurseur INTO v_Article;
                EXECUTE IMMEDIATE 'INSERT INTO articleARecomder values ('''||codeClient2||''','''||
codeClient1||''','''' ||v_Article|| ''')';
                EXIT WHEN monCurseur%NOTFOUND;
            END LOOP;
        CLOSE monCurseur;
    END;
/
```

```
CREATE OR REPLACE PROCEDURE systemeRecommandation(pourcentageAmi IN NUMBER, dateDebut IN VARCHAR, dateFin IN VARCHAR)
as
    req VARCHAR(2000) := '';
    reqIn VARCHAR(2000) := '';
    nbArticle NUMBER := 0;
    BEGIN
        -- Suppresion et Création de la table qui va contenir les articles qu'ont les clients en commun
        EXECUTE IMMEDIATE 'DROP TABLE totalArticleCommun';
        EXECUTE IMMEDIATE 'CREATE TABLE totalArticleCommun(client1 VARCHAR(30), client2 VARCHAR(30), nbArtCm NUMBER)';
        -- Parcours sur tous les clients pour creer une vue Vij avec i et j des clients différents
        FOR mesClients in (SELECT * FROM combinaisonClient)
        LOOP
        vueArticleDesClients(''||mesClients.client1||'', dateDebut, dateFin, ''||mesClients.client2||'', dateDebut, dateFin);
        req := 'SELECT COUNT(*) FROM V_' || mesClients.client1 || '_' || mesClients.client2 || '_A';
        EXECUTE IMMEDIATE req INTO nbArticle;
        reqIn := 'INSERT INTO totalArticleCommun values (''' || mesClients.client1 || ''', ''' || mesClients.client2 || ''', ' || nbArticle || ')';
        EXECUTE IMMEDIATE reqIn;
        END LOOP;
        -- On va creer une vue avec les clients et le nombre d'articles en commun
        EXECUTE IMMEDIATE 'CREATE OR REPLACE VIEW V_NBArtCm (client1, client2, nbArtCm) as
                           (SELECT client1, client2, (SELECT getNbArtCm(client1, client2) FROM dual) FROM CombinaisonClient)';
        -- Et pour finir on va creer une vue qui est n'composée que des clients qui possèdent un nombre d'articles supérieur au seuil défini.
        -- Ces clients sont considérés comme Amis
        EXECUTE IMMEDIATE 'CREATE OR REPLACE VIEW V_RcmdAmi (client1, client2) as
                        SELECT client1, client2
                        FROM (SELECT client1, client2, nbArtCm
                                FROM V_NBArtCm group by client1, client2, nbArtCm
                                having nbArtCm >=   (SELECT FLOOR(max(nbArtCm) * ' || pourcentageAmi || ' / 100)
                                FROM V_NBArtCm))
                      ORDER BY client1, client2';
        -- Maintenant qu'on a une table contenant tous les amis il faut recommander tous les articles du client C1 au client C2 et inversement
        -- On utilise la procedure vueArticleNonDesClients
        -- Suppresion et creation de la table qui va contenir le client 1 qui va recommander au client 2 l'article X
        EXECUTE IMMEDIATE 'DROP TABLE articleARecomder';
        EXECUTE IMMEDIATE 'CREATE TABLE articleARecomder(client1 VARCHAR(30), client2 VARCHAR(30), article VARCHAR(30))';
        -- Pour tous les clients considérés comme ami nous allons recommander les articles de sont client ami
        FOR mesClients IN (SELECT * FROM V_RcmdAmi)
        LOOP
            propositionArticle(mesClients.client1, mesClients.client2, dateDebut, dateFin);
        END LOOP;
        -- Pour finir on cré la vue qui ne contient que les clients et les articles qu'on souhaite leur recommander
        -- Cette étape n'est pas obligatiore mais elle nous permet de supprimer les doublons au niveau des recommandations
        -- De ce fait, un client aura une seule fois un article recommandé
        EXECUTE IMMEDIATE 'CREATE OR REPLACE VIEW V_ListArtARecom(client, article)  as
                        SELECT DISTINCT client2,article
                        FROM articleARecomder WHERE ARTICLE IS NOT NULL ORDER BY CLIENT2, ARTICLE';
    END;
/

show error;

EXEC systemeRecommandation(80, '01-SEPTEMBER-2018' ,'30-SEPTEMBER-2018');
SELECT * FROM V_ListArtARecom;
```

On se retrouve avec la table final V_ListArtARecom :

CLIENT	ARTICLE
C001	FB.001
C001	F1.004
C001	F1.005
C002	F1.004
C002	F1.005
C003	F1.004
C004	FB.001
C013	F1.007