Rapport
TP 5

Création d'un magasin de données

Réalisé en binôme par :
BENSALAH Kawthar / ABBACI Khaled
Numero du binôme : 22
Master 2 IL - Groupe 1
USTHB 2019/2020

TP 5

Objectif du TP

Création d'un magasin de données

Introduction

À travers ce TP, nous allons apprendre à créer un magasin de données en se basant sur le modèle R-OLAP dénormalisé.

Création d'un nouvel utilisateur

- Création d'un nouvel utilisateur Master2 en lui attribuant tous les privilèges :

```
SQL> CREATE TABLESPACE DefaultTBS2
  2       DATAFILE 'C:\DefaultTBSFile2.dat'
  3       SIZE 100M AUTOEXTEND ON ONLINE;

Tablespace créé.

SQL> CREATE TEMPORARY TABLESPACE TempTBS2
  2       TEMPFILE 'C:\TempTBSFile2.dat'
  3       SIZE 100M AUTOEXTEND ON;

Tablespace créé.

SQL> CREATE USER Master2
  2       IDENTIFIED BY psw
  3       DEFAULT TABLESPACE DefaultTBS2
  4       TEMPORARY TABLESPACE TempTBS2;

Utilisateur créé.

SQL> GRANT ALL PRIVILEGES TO Master2;

Autorisation de privilèges (GRANT) acceptée.

SQL>
```

- Se connecter en tant que Master2 :

```
SQL> disconnect;
Déconnecté de Oracle Database 11g Enterprise Edition Release 11.2.0.2.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL> connect Master2/psw
Connecté.
SQL>
```

Création des tables

- **Création de la table DClient :**

```
SQL> CREATE TABLE DClient (
  2       NumClient Number(10) PRIMARY KEY,
  3       NomClient VARCHAR2(50),
  4       DNClient Date );

Table créée.

SQL>
```

- **Vérifier la création de la table DClient :**

```
SQL> desc dclient
 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ----------------------------
 NUMCLIENT                                 NOT NULL NUMBER(10)
 NOMCLIENT                                          VARCHAR2(50)
 DNCLIENT                                           DATE

SQL>
```

- **Création de la table DAgence :**

```
SQL> CREATE TABLE DAgence (
  2       NumAgence Number(10) PRIMARY KEY,
  3       NomAgence VARCHAR2(50),
  4       CodeBanque Number(10),
  5       NomBanque VARCHAR2(50),
  6       CodeVille Number(10),
  7       NomVille VARCHAR2(50),
  8       CodeWilaya Number(10),
  9       NomWilaya VARCHAR2(50)
 10  );

Table créée.

SQL>
```

- **Vérifier la création de la table DAgence :**

```
SQL> desc dagence
 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ----------------------------
 NUMAGENCE                                 NOT NULL NUMBER(10)
 NOMAGENCE                                          VARCHAR2(50)
 CODEBANQUE                                         NUMBER(10)
 NOMBANQUE                                          VARCHAR2(50)
 CODEVILLE                                          NUMBER(10)
 NOMVILLE                                           VARCHAR2(50)
 CODEWILAYA                                         NUMBER(10)
 NOMWILAYA                                          VARCHAR2(50)

SQL>
```

- **Création de la table DTypeCompte :**

```
SQL> CREATE TABLE DTypeCompte (
  2       CodeType Number(1) PRIMARY KEY,
  3       LibType VARCHAR2(50)
  4   );

Table créée.

SQL>
```

- **Vérifier la création de la table DTypeCompte :**

```
SQL> desc dtypecompte
 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ----------------------------
 CODETYPE                                  NOT NULL NUMBER(1)
 LIBTYPE                                            VARCHAR2(50)

SQL>
```

- **Création de la table DTemps :**

```
SQL> CREATE TABLE DTemps (
  2       CodeTemps Number(10) PRIMARY KEY,
  3       Jour VARCHAR2(15),
  4       LibJour VARCHAR2(15),
  5       Mois VARCHAR2(15),
  6       LibMois VARCHAR2(15),
  7       Année VARCHAR2(15)
  8   );

Table créée.
```

- **Vérifier la création de la table DTemps :**

```
SQL> desc dtemps
 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ----------------------------
 CODETEMPS                                 NOT NULL NUMBER(10)
 JOUR                                               VARCHAR2(15)
 LIBJOUR                                            VARCHAR2(15)
 MOIS                                               VARCHAR2(15)
 LIBMOIS                                            VARCHAR2(15)
 ANNÉE                                              VARCHAR2(15)

SQL>
```

- **Création de la table FOperation :**

```
SQL> CREATE TABLE FOperation (
  2      NumClient Number(10),
  3      NumAgence Number(10),
  4      CodeTypeCompte Number(10),
  5      CodeTemps Number(10),
  6      NbOperationR Number(10),
  7      NbOperationV Number(10),
  8      MontantR number(10,2),
  9      MontantV number(10,2),
 10      CONSTRAINT FK_O1
 11          FOREIGN KEY (NumClient)
 12          REFERENCES DClient(NumClient),
 13      CONSTRAINT FK_O2
 14          FOREIGN KEY (NumAgence)
 15          REFERENCES DAgence(NumAgence),
 16      CONSTRAINT FK_O3
 17          FOREIGN KEY (CodeTypeCompte)
 18          REFERENCES DTypeCompte(CodeType),
 19      CONSTRAINT FK_O4
 20          FOREIGN KEY (CodeTemps)
 21          REFERENCES DTemps(CodeTemps),
 22      CONSTRAINT PK_O
 23          PRIMARY KEY (NumClient, NumAgence,CodeTypeCompte, CodeTemps)
 24  );

Table créée.

SQL>
```

- **Vérifier la création de la table FOperation :**

```
SQL> desc foperation
 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ----------------------------
 NUMCLIENT                                 NOT NULL NUMBER(10)
 NUMAGENCE                                 NOT NULL NUMBER(10)
 CODETYPECOMPTE                            NOT NULL NUMBER(10)
 CODETEMPS                                 NOT NULL NUMBER(10)
 NBOPERATIONR                                       NUMBER(10)
 NBOPERATIONV                                       NUMBER(10)
 MONTANTR                                           NUMBER(10,2)
 MONTANTV                                           NUMBER(10,2)

SQL>
```

Remplissage des tables

- **Remplissage de la table DClient :**

```
SQL> begin
  2    for i in (SELECT NumClient, NomClient, DNClient
  3                   FROM Master.Client)
  4      loop
  5        insert into DClient
  6        values (i.NumClient, i.NomClient, i.DNClient);
  7      end loop;
  8  commit;
  9  end;
 10  /

Procédure PL/SQL terminée avec succès.

SQL>
```

- **Vérifier que la table DClient est bien remplie :**

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

  COUNT(*)
----------
    100000

SQL>
```

- **Remplissage de la table DAgence :**

```
SQL> begin
  2    for i in (
  3      select a.NumAgence, a.NomAgence, b.CodeBanque,
  4             b.NomBanque, v.CodeVille, v.NomVille,
  5             w.CodeWilaya, w.NomWilaya
  6             from Master.Agence a, Master.Banque b,
  7                  Master.Ville v, Master.Wilaya w
  8             where (a.AppartientBanque=b.CodeBanque)
  9                 and (a.SeTrouveVille=v.CodeVille)
 10                 and (v.DependWilaya=w.CodeWilaya)
 11      )
 12    loop
 13    insert into DAgence values
 14      ( i.NumAgence, i.NomAgence, i.CodeBanque, i.NomBanque, i.CodeVille,
 15        i.NomVille, i.CodeWilaya, i.NomWilaya);
 16    end loop;
 17  COMMIT;
 18  end;
 19  /

Procédure PL/SQL terminée avec succès.

SQL>
```

- **Vérifier que la table DAgence est bien remplie :**

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

  COUNT(*)
----------
     12300

SQL>
```

- **Remplissage de la table DTypeCompte :**

```
SQL> begin
  2    for i in (SELECT CodeType, LibType
  3                FROM Master.Type_Compte)
  4      loop
  5        insert into DTypeCompte
  6        values (i.CodeType, i.LibType);
  7      end loop;
  8  commit;
  9  end;
 10  /

Procédure PL/SQL terminée avec succès.

SQL>
```

- **Vérifier que la table DTypeCompte est bien remplie :**

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

  COUNT(*)
----------
         2

SQL>
```

- **Création d'une séquence pour l'utiliser dans le remplissage de la table DTemps :**

```
SQL> CREATE SEQUENCE seq MINVALUE 1 MAXVALUE 1000000 START WITH 1 INCREMENT BY 1;

Séquence créée.

SQL>
```

- **Remplissage de la table DTemps :**

```
SQL> begin
  2   for i in (SELECT distinct TO_CHAR(DateOp,'DD/MM/YYYY') as Jour, TO_CHAR(DateOp,'DAY') as LibJour,
  3             TO_CHAR(DateOp,'MM/YYYY') as Mois, TO_CHAR(DateOp,'MONTH') as LibMois,
  4             TO_CHAR(DateOp,'YYYY') as Année
  5             FROM Master.Operation)
  6      loop
  7          insert into DTemps values (seq.NEXTVAL, i.Jour, i.LibJour, i.Mois, i.LibMois, i.Année);
  8      end loop;
  9  commit;
 10  end;
 11  /

Procédure PL/SQL terminée avec succès.

SQL>
```

- **Vérifier que la table DTemps est bien remplie :**

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

  COUNT(*)
----------
      1460

SQL>
```

- **Remplissage de la table FOperation :**

```
SQL> begin
  2    for i in (
  3      SELECT C.PossedeClient, C.est_domicileAg, C.AppartientType_Compte, T.CodeTemps,
  4      sum(decode(O.TypeOp,2,1,0)) as NbOperationR,
  5      sum(decode(O.TypeOp,1,1,0)) as NbOperationV,
  6      sum(decode(O.TypeOp,2,O.Montant,0)) as MontantR,
  7      sum(decode(O.TypeOp,1,O.Montant,0)) as MontantV
  8          FROM Master.Compte C, DTemps T, Master.Operation O
  9          WHERE (C.NumCompte = O.VersementCompte or C.NumCompte = O.RetraitCompte) and
 10              TO_CHAR(O.DateOp,'DD/MM/YYYY') = T.Jour
 11              GROUP BY C.PossedeClient, C.est_domicileAg, C.AppartientType_Compte, T.CodeTemps)
 12      loop
 13          insert into FOperation values (i.PossedeClient, i.est_domicileAg,
 14                                         i.AppartientType_Compte, i.CodeTemps,
 15                                         i.NbOperationR, i.NbOperationV,
 16                                         i.MontantR, i.MontantV);
 17    end loop;
 18  COMMIT;
 19  end;
 20  /

Procédure PL/SQL terminée avec succès.

SQL>
```

- **Vérifier que la table FOperation est bien remplie :**

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

  COUNT(*)
----------
    609672

SQL>
```

Conclusion

Après la réalisation de ce TP, nous avons conçu un magasin de données en se basant sur les modèle R-OLAP dénormalisé. Dans un tel modèle, le fait et les dimensions sont représenté par des tables Sql.
La dénormalisation consiste à regrouper plusieurs tables liées par des références en une seule table en se basant sur la jointure (dans la table DAgence par exemple), la troisième forme normale n'est pas respectée.
L'avantage d'un tel processus est sa simplicité mais il est considéré comme facteur de redondance et d'incohérence. Des dépendances entre des attributs non clés peuvent apparaître (par exemple, la dépendance entre les attributs non clés NumBanque et NomBanque dans la table DAgence).