Introduction aux bases de données

Langage SQL Partie 3

Le langage SQL (SGBD ORACLE)

I. **Définition des données**
II. Les vues

Définition de données (schémas)

1. Création de schéma de relation **(CREATE TABLE)**
2. Création de schéma + instanciation **(CREATE TABLE … AS…)**
3. Modification de schéma de relation **(ALTER TABLE)**
4. Suppression de schéma de relation **(DROP TABLE)**
5. Index pour l'accélération des accès **(CREATE/DROP Index)**

1. Création de schéma de relation

```
CREATE TABLE Client(
        noClient      INTEGER,
        nom           VARCHAR(50),
        prénom        VARCHAR(15)
        )
```

- Transmise à l'interprète du LDD
  i. vérification
  ii. création de la table
    - schéma stocké dans le dictionnaire de données
    - allocation des structures physiques

Schéma et dictionnaire de données Oracle avec SQL*plus

```
SQL> CREATE TABLE Client
  2   (noCLIENT    INTEGER,
  3    nomClient    VARCHAR(15),
  4    noTéléphone   VARCHAR(15))
  5   /

Table créée.

SQL> SELECT Table_Name
  2   FROM    USER_TABLES
  3   /

TABLE_NAME
------------------------------
CLIENT

SQL> SELECT Column_Name, Data_Type
  2   FROM    USER_TAB_COLUMNS
  3   WHERE   Table_Name = 'CLIENT'
  4   /

COLUMN_NAME                    DATA_TYPE
------------------------------ ------------------
-----------
NOCLIENT                       NUMBER
NOMCLIENT                      VARCHAR2
NOTÉLÉPHONE                    VARCHAR2
```

Syntaxe générale du CREATE TABLE (1/2)

```
CREATE TABLE nomTable
        (définitionAttribut
        [,définitionAttribut]…
        [,définitionContrainte]…));
```

- Syntaxe de **définitionAttribut**

```
nomAttribut  typeDeDonnées
        [DEFAULT valeur][NULL|NOT NULL]
        [UNIQUE|PRIMARY KEY]
        [REFERENCES nomTableBis(nomAttributBis)]
        [[CONSTRAINT nomContrainte] CHECK (condition)]
```

Rappel : Les noms d'attributs ou de tables sont mémorisés en majuscules dans le catalogue

Syntaxe générale du CREATE TABLE (2/2)

- Syntaxe de **définitionContrainte**

```
[CONSTRAINT nomContrainte]
{PRIMARY KEY (listeAttributs)} |
 {UNIQUE (listeAttributs)} |
 {FOREIGN KEY (listeAttributs) REFERENCES
    nomTableBis[(listeAttributs)]
      [ON DELETE {NO ACTION}|CASCADE|{SET NULL}|
                                  {SET DEFAULT}]
      [ON UPDATE {NO ACTION}|CASCADE|{SET NULL}|
                                  {SET DEFAULT}]
  }|
 {CHECK (condition)}
```

Types de données dans ORACLE (1)

- *Numérique*

NUMBER ou NUMBER (p) ou NUMBER(p, c)

- **NUMBER(p)** : Nombre entier à ***p*** chiffres (***p*** ≤ 38)

  ex : NUMBER (3): entier à 3 chiffres maximum (de –999 à +999)

- **NUMBER(p,c)** : Nombre entier ou décimal à ***p*** chiffres (excluant la virgule) dont ***c*** chiffres après la virgule (si ***c*** > 0)

  ex: NUMBER (5,2) : réel à 5 chiffres au total dont 2 après la virgule; valeur maxi = 999.99

  - Si ***c*** < 0, les données réelles sont arrondies à ***c*** positions à gauche de la virgule

- **NUMBER** : nombres à virgule flottante (précision et échelle maximales); ex. $1.666\ e^{-30}$

Types de données dans ORACLE (2)

Donnée réelle	Type spécifié	Stocké en …
9645123.88	NUMBER	9645123.88
9645123.88	NUMBER(9)	9645123
9645123.88	NUMBER(9,2)	9645123.88
9645123.88	NUMBER(9,1)	9645123.9
9645123.88	NUMBER(6)	dépassement de précision
9645123.88	NUMBER (7,-2)	9645100
9645123.88	NUMBER (7,2)	dépassement de précision

Types de données dans ORACLE (3)

D'autres types **numériques** de la norme ANSI sont reconnus dans Oracle

- **INT(EGER)**, **SMALLINT**, **FLOAT**, **REAL**…

Ces types sont "convertis " dans le type NUMBER

Type SQL ANSI	**Type Oracle**
NUMERIC(p,c) DECIMAL (p,c)	NUMBER (p,c)
INT(EGER), SMALLINT	NUMBER (38)
FLOAT (b), REAL	NUMBER

Types de données dans ORACLE (4)

- *Chaîne de caractères*
  - **CHARACTER(n)** ou **CHAR(n)**
    - Chaîne de caractère de taille fixe égale à *n* ; avec *n* <=*2000*
    - Exemples : 'G. Lemoyne-Allaire', 'Paul L''Heureux '
  - **CHARACTER VARYING (n)** ou **VARCHAR(n)**
    - Taille variable (max de *n* caractères) ; *n*<=4000

Types de données dans ORACLE (5)

- *Un type pour représenter la date **et** l'heure*

DATE

- Une date comporte l'information sur l'année (2 ou 4 chiffres), le mois (2 chiffres), le jour (2 chiffres), l'heure (0-23), la minute(0-59), et la seconde (0-59)
- Format par défaut : DD-MON-YY ex: 01-FEB-11
- Utilisation des fonctions **`to_date`** et **`to_char`** pour entrer/afficher une date selon un format différent
  - Ex : to_date('1998/05/31:12:00:00AM', 'yyyy/mm/dd:hh:mi:ssam')
  - Ex : to_char(sysdate, 'Dy DD-Mon-YYYY HH24:MI:SS')
    - Résultat affiché : Tue 21-Apr-1998 21:18:27

Contraintes d'intégrité (CI)

- Plusieurs types de contraintes d'intégrité peuvent être déclarées avec **`CREATE TABLE`**

  - contraintes sur le domaine d'un attribut
  - contraintes de clé primaire
  - contraintes d'intégrité référentielle

- D'autres contraintes peuvent être programmées et porter sur des changements d'états de la BD, sur plusieurs tables…

  - `CREATE TRIGGER` (non traité ici)

CI sur le domaine d'un attribut (1/4)

- Type de données (`number`, `char`…)
- Contrainte NOT NULL
  - L'utilisateur doit fournir une valeur définie à l'attribut (si pas de DEFAULT)

```
CREATE TABLE Client
   (noClient          INTEGER      NOT NULL,
    nom               VARCHAR(50)  NOT NULL,
    ddn               VARCHAR(15)  NULL,
    téléphone         VARCHAR(15)  )
```

- Par défaut : NULL
  - Le SGBD insère la valeur NULL si absence de valeur et pas de DEFAULT

CI sur le domaine d'un attribut (2/4)

- Contrainte CHECK sur un attribut

- *Le numéro de client est positif et inférieur à 1000*

```
CREATE TABLE Client
  (noClient        INTEGER     NOT NULL
    CHECK (noClient>0 AND noClient<1000),
   nom       VARCHAR(50) NOT NULL,
   ddn       VARCHAR(15) NULL )

CREATE TABLE Client
  (noClient INTEGER     NOT NULL,
   nom       VARCHAR(50) NOT NULL,
   ddn       VARCHAR(15) NULL,
  CHECK (noClient>0 AND noClient<1000) )
```

CI sur le domaine d'un attribut (3/4)

- Contrainte CHECK sur plusieurs attributs du même tuple

➢ *Les produits dont le numéro est supérieur à 100 ont un prix supérieur à 15 €.*

```
CREATE TABLE Produit
   (noProduit       INTEGER     NOT NULL,
    libellé         VARCHAR(50),
    prixUnitaire    NUMBER(10,2) NOT NULL,
    CHECK (noProduit<=100 OR prixUnitaire>15) )
```

CI sur le domaine d'un attribut (4/4)

- Valeur d'un attribut par défaut (`DEFAULT`)

- *Le numéro de téléphone est 'confidentiel' par défaut*

```
CREATE TABLE employé
 (numEmp    INTEGER,
  numTel    VARCHAR(15) NOT NULL
      DEFAULT 'confidentiel' … )
```

Contrainte de clé primaire

- Deux tuples ne peuvent pas avoir la même valeur de la clé
- Le(s) attribut(s) clé ne peuvent pas être NULL
- *La clé primaire de la table Client est noClient* (clé atomique)

```
CREATE TABLE Client
 (noClient  INTEGER      PRIMARY KEY,
  nom       VARCHAR(50)  NOT NULL,
  ddn       VARCHAR(15)  NULL );

CREATE TABLE Client
 (noClient  INTEGER ,
  nom       VARCHAR(50)  NOT NULL,
  ddn       VARCHAR(15)  NULL,
  PRIMARY KEY (noClient))
```

Contrainte de clé primaire (clé composée)

- *La clé primaire de la table Commande est constituée des 3 attributs noClient , noProduit et dateCde.*

```
CREATE TABLE Commande
(noClient    INTEGER,
 noProduit   INTEGER,
 dateCommande DATE,
 quantité    INTEGER,
 PRIMARY KEY
        (noClient,noProduit,dateCommande)
)
```

- Lorsque la clé est composée de plusieurs attributs, définir la contrainte PRIMARY KEY au niveau de la table

Contrainte d'unicité (UNIQUE)

- Une seule clé primaire par table mais d'autres attributs peuvent avoir des valeurs uniques pour chaque tuple.

```
CREATE TABLE Citoyen
(numSécu     INTEGER PRIMARY KEY ,
 nom         VARCHAR(50),
 prénom      VARCHAR(50),
 ddn         DATE  NULL,     --date de naissance
 noPassport  INTEGER NULL  UNIQUE )
```

Attribut auto-incrémentés – Notion de séquence (1/2)

- Les champs *incrémentables automatiquement (auto increment)* n'existent pas dans le SGBD Oracle. Pour simuler ce comportement, il est possible d'utiliser la notion de séquence.

- -- Création d'une séquence

```
CREATE  SEQUENCE  ma_sequence
START  WITH  1
MAXVALUE  9999
MINVALUE  1;
```

Attribut auto-incrémentés _ Notion de séquence (2/2)

- `ma_sequence.nextval` retourne la valeur suivante de la séquence `ma_sequence`

- Exemple d'utilisation : insérer une ligne dans la table `ma_table`, avec la valeur suivante de `ma_sequence` comme valeur de l'attribut `colonne`

```
INSERT INTO ma_table (colonne, ...)
VALUES  (ma_sequence.NextVal, ...);
```

- On peut noter que la séquence n'est pas liée à un champ ou à une table et qu'il est possible de l'utiliser pour plusieurs attributs/tables

Contrainte d'intégrité référentielle (FOREIGN KEY / REFERENCES)

- *noClient dans Commande fait référence à la clé primaire dans Client.*

```
CREATE TABLE Commande
(noCommande INTEGER PRIMARY KEY,
 noClient   INTEGER,
 noProduit  INTEGER,
 dateCommande DATE,
 quantité   INTEGER,
 FOREIGN KEY (noClient) REFERENCES Client(noClient),
 FOREIGN KEY (noProduit)REFERENCES Produit(noProduit))
```

- Les tables *Client* et *Produit* doivent avoir été créées
- On ne peut ajouter des commandes que si le client et le produit existent déjà
- ***N.B. :*** **La cible d'une référence doit être PRIMARY KEY ou UNIQUE**

CI référentielle : conduite à tenir en cas de mise à jour

```
[CONSTRAINT nomContrainte]
FOREIGN KEY listeAttributs REFERENCES
      nomTableBis(listeAttributs)
[ON DELETE {NO ACTION|CASCADE|SET NULL| SET DEFAULT}]
[ON UPDATE {NO ACTION|CASCADE|SET NULL| SET DEFAULT}]
```

CI référentielle : conduite à tenir en cas de mise à jour (DELETE)

- Tentative de suppression de la clé primaire

```
CREATE TABLE Commande
(noCommande INTEGER PRIMARY KEY,
 noClient   INTEGER,  …
 FOREIGN KEY (noClient) REFERENCES Client(noClient))
```

```
DELETE FROM Client where noClient=45
```

Que deviennent les commandes du client numéro 45 ?

- Quatre options pour la conduite à tenir :
  - NO ACTION
  - CASCADE
  - SET NULL
  - SET DEFAULT

CI référentielle : conduite à tenir en cas de suppression de la clé primaire

- NO ACTION

s'il existe des commandes du client 45, la suppression est refusée (violation contrainte)

- CASCADE

s'il existe des commandes du client 45, supprimer ces commandes puis supprimer le client.

- SET NULL

s'il existe des commandes du client 45, y positionner noClient à NULL puis supprimer le client.

- SET DEFAULT

s'il existe des commandes du client 45, y positionner noClient à la valeur par défaut (lorsqu'elle existe) puis supprimer le client.

CI référentielle : conduite à tenir en cas de mise à jour (UPDATE)

- Tentative de modification de la clé primaire

```
CREATE TABLE Commande
(noCommande INTEGER PRIMARY KEY,
 noClient   INTEGER,  …
 FOREIGN KEY (noClient) REFERENCES Client(noClient))
```

```
UPDATE Client SET noClient=100 where noClient=45
```

- *Que deviennent les commandes du client numéro 45 ?*
- quatre options pour la conduite à tenir :
  - NO ACTION
  - CASCADE
  - SET NULL
  - SET DEFAULT

CI référentielle : conduite à tenir en cas de mise à jour (UPDATE)

- NO ACTION

s'il existe des commandes du client 45, la modification est refusée (violation contrainte)

- CASCADE

s'il existe des commandes du client 45, y modifier noClient puis modifier noClient dans Client

- SET NULL

s'il existe des commandes du client 45, y positionner noClient à NULL puis opérer la modification de noClient dans Client.

- SET DEFAULT

s'il existe des commandes du client 45, y positionner noClient à la valeur par défaut (lorsqu'elle existe) puis supprimer le client.

CI référentielle : conduite à tenir en cas de mise à jour

```
CREATE TABLE Commande
(noCommande INTEGER PRIMARY KEY,
 noClient   INTEGER,  …
 FOREIGN KEY (noClient) REFERENCES Client(noClient)
  ON DELETE CASCADE
  ON UPDATE CASCADE )
```

- *N.B.* : Sous Oracle
  - ✓ par défaut :
    - ON DELETE NO ACTION
    - ON UPDATE NO ACTION
  - ✓ mais seules possibilités dans CREATE TABLE
    - ON DELETE CASCADE
    - ON DELETE SET NULL

Création de schéma + instanciation (1/2)

- CREATE TABLE : ne permet **que** la définition d'un schéma de relation
- L'instanciation de la relation se fait par une suite d'insertions de tuples (commande INSERT)
- Il est possible de créer et d'instancier une table à partir d'une requête :

```
CREATE TABLE … AS SELECT …
```

Création de schéma + instanciation (2/2)

```
CREATE TABLE nomTable[(nomAtt1, nomAtt2 …)] AS SELECT …
```

a- créer un schéma de relation

b- remplir la relation par évaluation de la requête

```
CREATE TABLE Client
 (noClient integer not null,
  nom varchar(50) not null,
  CP  number(5) check CP>=1000 and CP<=99999);
INSERT INTO Client values (1, 'A', 54000);
INSERT INTO Client values (2, 'B', 75800);
…
CREATE TABLE Client54 AS
      (SELECT     noClient, nom
       from       Client
       where      CP between 54000 and 54999);
```

Modification de schéma de relation(1/3)

- Une fois un schéma de table créé et des données entrées, on peut :
  - ajouter un attribut (nom, type, défaut, NOT NULL si la table vide)
  - modifier ou supprimer le type ou la valeur par défaut d'un attribut
  - supprimer un attribut ou ajouter une contrainte
  - supprimer une contraintes nommée ou de clé primaire ou d'unicité

```
ALTER TABLE nomTable
  ADD (attribut type [DEFAULT valeur] [contrainte]) |
  MODIFY (attribut [type] [DEFAULT valeur] [contrainte]) |
  DROP COLUMN attribut |
  ADD [CONSTRAINT nom] contrainte |
  DROP {PRIMARY KEY} | UNIQUE (attributs)} |{CONSTRAINT nom}
```

Modification de schéma de relation(1/3)

ALTER TABLE effectue

- une copie temporaire de la table originale,
- les modifications sont faites sur cette copie,
- puis la table originale est effacée,
- enfin la copie est renommée pour remplacer l'originale.

Cette méthode permet de rediriger toutes les commandes automatiquement vers la nouvelle table sans pertes.

Durant l'exécution de ALTER TABLE, la table originale est lisible par d'autres utilisateurs.

Les modifications et insertions sont reportées jusqu'à ce que la nouvelle table soit prête.

Modification de schéma de relation(2/3)

```
ALTER TABLE client ADD (TEL_PORTABLE CHAR(10))

ALTER TABLE client MODIFY (ADR_CLIENT CHAR(70))

ALTER TABLE produit ADD CONSTRAINT PRIX_POSITIF
check (PRIXUNITAIRE >= 0)

ALTER TABLE produit DROP CONSTRAINT PRIX_POSITIF

ALTER TABLE client  DROP COLUMN TEL_PORTABLE

RENAME CLIENT to ACHETEUR
```

Modification de schéma de relation(3/3)

- La commande ALTER TABLE permet de **désactiver** une contrainte nommée

```
ALTER TABLE nom_table DISABLE CONSTRAINT
nom_contrainte;
```

- La commande ALTER TABLE permet **d'activer** une contrainte nommée

```
ALTER TABLE nom_table ENABLE CONSTRAINT
nom_contrainte;
```

4. Suppression d'un schéma de relation (DROP TABLE)

```
DROP TABLE  nom-relation [CASCADE CONSTRAINTS]
```

- ✓ suppression du **schéma** de la relation
- ✓ suppression des **tuples** de la relation
- ✓ Suppression des **index**, désactivation des **vues** liées

Si la clé primaire de la relation à supprimer est référencée dans d'autres tables, l'option CASCADE CONSTRAINTS permet de supprimer ces CI dans ces tables.

Le langage SQL (SGBD ORACLE)

I. Définition des données
II. **Les vues**

Les vues

- Une vue peut se définir comme une table virtuelle, construite à partir d'une requête
- Une vue est une façon de stocker une requête afin de pouvoir s'y référer simplement à tout moment. Un utilisateur peut donc lancer une requête directement sur la vue, comme s'il s'agissait d'une table.
- Lorsqu'une table sur laquelle une vue est construite est modifiée, alors la vue "prend en compte" cette modification

Les vues

- Création d'une vue : La syntaxe pour la création d'une vue est la suivante.

```
CREATE VIEW nom_de_la_vue [(liste_de_colonnes)]
AS expression_select
```

Ex.

```
CREATE VIEW produit_sans_prix AS
SELECT IdProd, nomProduit, marqProduit
FROM produit;
```

- Pour supprimer une vue on utilise la syntaxe

```
DROP VIEW nom_de_la_vue ;
```

Intérêt des vues

- Masquage des opérations de jointure
- Sauvegarde (indirecte) de requêtes complexes
- Indispensables pour les attributs calculés

  ex. âge, montantTTC, total de ventes par produit *etc.*
- Support de l'indépendance logique
  - si les tables sont modifiées, les vues doivent être réécrites mais les requêtes utilisant les vues ne subissent pas de changement
- Support de la confidentialité en combinaison avec des privilèges d'accès adéquats
  - Masquer les lignes ou colonnes sensibles pour les utilisateurs non autorisés
  - Créer une vue correspondant à ce qui peut être vu par un utilisateur U et donner à U le droit d'accès à cette vue et non aux tables

Exemples de vues

Création d'une vue VEtudiants qui donne le nom de la commune de naissance des étudiants

```
create view VEtudiant (nom, prenom, dateNaissance,
commune) as
        select E.nom, E.prenom, E.ddnaissance, C.nom
        from etudiant E, commune C,
        Where E.id_commune = C.id ;
```

Utilisation :

```
Select *
From   VEtudiant ;
```

Exemples de vues (2)

Exemple vue groupée :

Création d'une vue Stat de vente qui donne les statistiques sur un produit

```
CREATE VIEW stat_vente ( IdProduit, total_vente ) AS
        SELECT V.IdProduit, SUM(V.qte)
        FROM vente V
        GROUP BY V.IdProduit;
```

Utilisation :

```
        SELECT *
        FROM   stat_vente ;
```