ORACLE SQL – LDD

Langage de définition de données

Le langage SQL (Structured Query Language)

SQL a été normalisé par l'ANSI puis par l'ISO depuis 1986 sous ses différents aspects :

- LDD : définition des données
- LMD : manipulation des données
- LID : interrogation des données
- LCD : contrôle des données

Le langage SQL proposé par les SGBDR actuels est proche de la 2ème norme : SQL-2 ou SQL-92.

Du code SQL peut être intégré également dans un programme écrit dans un langage hôte : Cobol, Fortran, Pascal, C, ADA, PL/SQL, Java, ...

Le langage SQL (Structured Query Language)

La norme SQL comporte trois parties distinctes qui concernent :

(1) la manipulation de l'information, destinée au développeur, subdivisée en

langage d'interrogation de données (LID) : SELECT

langage de manipulation de données (LMD) : INSERT, UPDATE, DELETE

Le langage SQL (Structured Query Language)

La norme SQL comporte trois parties distinctes qui concernent :

(2) la définition des structures de données subdivisée en

langage de définition de données (LDD) : CREATE, ALTER, DROP

langage de contrôle de données (LCD) : GRANT, REVOKE

Le langage SQL (Structured Query Language)

La norme SQL comporte trois parties distinctes qui concernent :

(3) l'embedded SQL qui définit

l'accès aux informations stockées dans le SGBD depuis un langage de prog hôte,

l'utilisation des ordres SQL dans un langage hôte,

l'ensemble des codes d'erreur renvoyés par le SGBD au programme d'application.

MLDR → MPD

Le modèle logique de données (MLD) est exprimée dans un formalisme général qui doit être compatible avec l'état de l'art technique.

Il est alors transformé en modèle physique de données (MPD), script SQL tenant compte des spécificités du SGBD retenu.

Notre choix : MLDR (relationnel) et ORACLE.

Schéma ORACLE

Un schéma est un ensemble de structures logiques de données qui appartiennent à un utilisateur de la base de données et porte son nom. Un utilisateur ne peut donc avoir qu'un seul schéma

Un schéma peut-être composé de :

Table	Fonction	Procédure
Index	Déclencheur	Package
Vue	Synonyme	Séquence

Dans Oracle, la création d'un schéma est implicite : la création d'un utilisateur sous Oracle entraîne automatiquement la création du schéma (de même nom) correspondant.

Création de tables

Une table est composée de colonnes auxquelles on attribue obligatoirement un nom et un type de données.

CHAR(*n*) : chaîne de caractères de longueur fixe *n*. Le paramètre *n* est compris entre 1 (valeur par défaut) et 2000.

VARCHAR2(*n*) : chaîne de caractères de longueur variable *n*. Le paramètre *n* est compris entre 1 (valeur par défaut) et 4000.

DATE : date et heure, dont la forme standard d'affichage est DD/MM/YY. Dates possibles jusqu'au 31/12/9999.

Création de tables

NUMBER :
entier ou décimal jusqu'à 38 chiffres.
On pourra utiliser les chiffres de 0 à 9, les signes "+" et "-" et le point décimal (".") ou la notation scientifique (exemple : "1.85E3" pour 1850).

NUMBER(n)
Idem que NUMBER avec une taille spécifiée entre 1 et 38 et une précision de 0.
Donc 7,9 est enregistré comme étant 8.

NUMBER (n,m)
Nombre de précision n et d'échelle m.
n : nombre entier de chiffres significatifs, entre 1 et 38 (par défaut).
m : nombre de chiffres à droite de la marque décimale

Création de tables

LONG : chaîne de caractère de taille variable pouvant aller jusqu'à 2Go. Son utilisation est soumise à quelques restrictions : une seule colonne par table de type LONG

les colonnes de ce type ne peuvent pas être utilisées dans des sous- requêtes, des fonctions, des expressions, des clauses WHERE ou dans des index.

On peut lui préférer le type CLOB (Character Large Object -4 Go- Oracle 8i)

RAW(n)
Binaire de longueur fixe n.
« n » est compris entre 1 et 2000, doit toujours être spécifié (pas de valeur par défaut) en hexadécimal.

LONGRAW
Binaire de taille variable pouvant aller jusqu'à 2Go.
On peut lui préférer le type BLOB, (Binary Large Object - 4 Go- Oracle 8i)

Création simple d'une table

```
CREATE TABLE  [nomSchéma.] nomTable
            (
               nomColonne1  <typeColonne1>
            [,  nomColonne 2 <typeColonne2>] …
            ) ;
```

nomTable : nom unique de table pour un schéma donné.

Par défaut la table sera créée dans le schéma de l'utilisateur.

Si l'on veut créer une table dans un autre schéma, à condition d'en avoir les droits, il faut préfixer le nom de la table par le nom du schéma.

nomColonne : nom de colonne de la table. Entre 1 et 254.

typeColonne : un des types de donnée définis précédemment.

Règles générales pour les noms d'objets

Ne pas utiliser de mot réservé du langage SQL.

30 caractères au maximum

Une lettre obligatoirement au début et ensuite des lettres, des chiffres ou des caractères spéciaux : “_” (souligné ou underscore), “@”, etc.

Exemple : adresse_facturation

Règles générales pour les noms d'objets

Commencer chaque bout de nom par une lettre en majuscule (sauf la première éventuellement) :

Exemple : adresseFacturation

Ne pas être sensible à la case : par défaut Oracle enregistre tous les noms des objets en majuscule sauf s'ils ont été créés avec des doubles quotes

Exemple de création de table de base

Créez la table suivante :

Employe(id,nomEmploye,salaire,dateEmbauche) avec :

L’id est un nombre (entier ou décimal).

Le nomEmploye est une chaine sur 30 positions.

Le salaire est un nombre décimal à 2 chiffres avant la virgule et 4 après.

La date d’embauche est une date.

Exemple de création de table de base

```
CREATE TABLE Employe
 ( id                    Number
 , nomEmploye        Varchar2(30)
 , salaire               Number (4,2)
 , dateEmbauche Date
 );
```

```
DESC  Employe   -- sert à afficher la structure d'une table
```

```
Nom                                NULL ?   Type
 ---------------------------------- -------     ----
 ID                                          NUMBER
 NOMEMPLOYE                                VARCHAR2(30)
 SALAIRE                                    NUMBER
 DATEEMBAUCHE                             DATE
```

```
DESC  [nomSchéma.]nomTable
```

Valeur par défaut sur une colonne

nomColonne1 <typeColonne1> [DEFAULT <expr>]

Exemple : Rajoutez à la table précédente ce qu'il manque pour que le salaire par défaut soit 1000 euros.

Valeur par défaut sur une colonne

nomColonne1 <typeColonne1> [DEFAULT <expr>]

Exemple : Rajoutez à la table précédente ce qu'il manque pour que le salaire par défaut soit 1000 euros.

```
CREATE TABLE Employe
 ( id                    Number
 , nomEmploye            Varchar2(30)
 , salaire               Number(4,2)  DEFAULT 1000
 , dateEmbauche Date
 );
```

Création plus élaborée : contraintes

```
CREATE TABLE [nomSchéma.]nomTable
        (  nomColonne <typeColonne>  [DEFAULT <expr>]
                                     [<contrainteColonne>]...
        , nomColonne <typeColonne>   [DEFAULT <expr>]
                                     [<contrainteColonne>]...    ...

        [, <contrainteTable>]...
        );
```

Une contrainte d'intégrité est une règle qui permet de contrôler la valeur d'une colonne ou d'un ensemble de colonnes. C'est la traduction, au niveau physique de règle de gestion.

Une contrainte d'intégrité peut être spécifiée lors de la création d'une table : soit sous forme de contrainte de colonne, soit sous forme de contrainte de table.

Les contraintes de colonnes

[CONSTRAINT nomContrainte] <typeContrainte>

Si on ne nomme pas la contrainte Oracle lui donnera implicitement un nom.

Ce nom automatique sera de la forme 'SYS_Cn...n' où n est un chiffre particulièrement illisible notamment quand il apparaîtra dans les messages d'erreur (signalant, par exemple, que la contrainte n'a pas été respectée).

Il est donc fortement conseillé de nommer explicitement ses contraintes.

Les types de contraintes : <typeContrainte>

(1) : [NOT] NULL → Pour accepter ou non des valeurs nulles dans la colonne.

(2) : UNIQUE → Pour définir une contrainte d'unicité (de valeur) sur la colonne.

Attention : une colonne déclaré unique peut avoir la valeur NULL

(3) PRIMARY KEY → Pour une déclaration de clé primaire, et donc d'une contrainte implicite d'unicité + non nullité sur la colonne.

Exemple : rajoutez à la définition précédente de la table

Employe(id,nomEmploye,salaire,dateEmbauche)

du code afin que le salaire soit non null, le nomEmploye unique et l'id une clé primaire.

```
CREATE TABLE Employe

( idEmploye          Number
                     Constraint PK_Employe primary key

, nomEmploye           Varchar2(30)
                     Constraint UN_Employe_nomEmploye unique

, salaire            Number(4,2)  DEFAULT 1000
                     Constraint NN_Employe_salaire not null

, dateEmbauche    Date

);
```

Le fait de nommer explicitement les contraintes permet de comprendre le message d'erreur en cas de problème. Par exemple : un message contenant « NN_Employe_salaire » est assez explicite pour que tout le monde comprenne que c'est l'attribut salaire de la table Employe qui est non null.

Les types de contraintes : <typeContrainte>

(4) REFERENCES [nomSchema.] nomTable [(nomColonne)]
[ON DELETE CASCADE]

Pour définir une contrainte de référence, i.e. de définition de clé étrangère.

Attention sous ORACLE une clé étrangère peut avoir la valeur NULL !

[ON DELETE CASCADE] : En cas de suppression d'un élément X de la table référencée, toutes les lignes de la table que l'on est en train de définir et qui contiennent un élément qui fait référence à X seront supprimées.

Le nom de la colonne référencée doit être cité quand elle n'est pas clé primaire, et la contrainte de référence s'appuie toujours sur une autre contrainte (primaire ou unique).

Exemple : Soit la table Département (id_dep, nom_dep) et rajoutons une colonne id_dep à la table Employe pour définir l'id du département dans lequel travaille l'employé.

Sachant que idDep est une clé primaire dans la table Département, rajoutez dans le code de la table Employe la contrainte de colonne qui déclare idDep de la table Employe comme clé étrangère faisant référence à l'attribut idDep de la table Département.

```
CREATE TABLE Employe

( idEmploye          Number
                     Constraint PK_Employe primary key

, nomEmploye         Varchar2(30)
                     Constraint UN_Employe_nomEmploye unique

, salaire            Number(4,2)  DEFAULT 1000
                     Constraint NN_Employe_salaire not null

, dateEmbauche       Date

, idDep              Number
                     Constraint FK_Employe_Departement_idDep
                                  references Departement
);
```

```
CREATE TABLE Employe

(
 …..

, idDep                 Number
                        Constraint FK_Employe_Departement_idDep
                                references Departement
);
```

Que se passe t il si on supprime un département de la table Département : est ce que les employés de la table Employe qui travaillent dans ce département seront effacés ?

Comment faire pour qu'ils le soient ?

```
CREATE TABLE Employe

(
 …..
, idDep                 Number
                        Constraint FK_Employe_Departement_idDep
                                references Departement
);
```

Que se passe t il si on supprime un département de la table Département : est ce que les employés de la table Employe qui travaillent dans ce département seront effacés ?

- si le département apparaît dans une ligne de la table employé alors on aura une erreur sinon on aura pas d'erreur et il sera effacé uniquement dans la table departement.

Comment faire pour qu'ils soient effacé aussi dans la table employé sur toutes les lignes où un employé travaille dans ce département ?

Il faut rajouter : on delete cascade.

```
CREATE TABLE Employe

 ( id                  Number
                       Constraint PK_Employe primary key


  …..
  …..

 , idDep               Number
                       Constraint FK_Employe_Departement_idDep
                                  references Departement on delete cascade
 );
```

Ordre de création ?

Que se passe t il si l'on crée la table Employe avant la table Département ?

Exemple : que se passe t il si l'on crée la table Employe avant la table Département ?

Une erreur se produit car la table référencée n'existe pas encore !

Nous verrons plus tard comment créer un MLDR :

- soit en trouvant un ordre de création de tables,

- soit en utilisant la commande ALTER TABLE.

Ordre de suppression ?

Que se passe t il si l'on supprime la table Département ?

Ordre de suppression ?

Que se passe t il si l'on supprime la table Département avant Employe ?

Il y aura également une erreur car la table Employe a besoin de la table Département pour sa contrainte de clé étrangère.

Il faudra donc supprimer d'abord la table Employe puis supprimer la table Département ou utiliser une option de suppression particulière qu'on verra plus tard.

Les types de contraintes : <typeContrainte>

(5) CHECK (<condition>)

Pour préciser le type d'une colonne (énumération, intervalle...).

<condition> représente une expression booléenne (nous y reviendrons plus tard)

Exemple : check (code in (10,20,30)) où code est un NUMBER

check(grade in ('A', 'B','C')) où grade est un VARCHAR2

check(nbEnfant<>2) où nbEnfant est un NUMBER

check(salaire BETWEEN 1000 and 3000) où salaire est un NUMBER

```
check(dateCommande between to_date ('2003/01/01', 'yyyy/mm/dd')
                        AND to_date ('2003/12/31', 'yyyy/mm/dd'));
```

Ici dateCommande est de type DATE.

Les contraintes de tables

La contrainte d'intégrité, spécifiée sous forme de contrainte de table, concerne généralement un ensemble de colonnes de la table.

Les contraintes de table sont introduites après la description des colonnes.

On l'utilise, en général, pour définir une clé primaire ou étrangère composée.

[CONSTRAINT nomContrainte] UNIQUE (nomColonne [, nomColonne]...)

[CONSTRAINT nomContrainte] PRIMARY KEY (nomColonne [, nomColonne]...)

Exemple de création de clé primaire composée

Soit la table :

T1 (id1,id2)

avec id1 et id2 deux nombres.

Donnez le code qui permet de créer cette table puis rajouter la contrainte de table qui permet de déclarer le couple (id1,id2) comme clé primaire.

Exemple de création de clé primaire composée

```
CREATE TABLE  T1

    (id1 NUMBER

    ,id2 NUMBER

    ,CONSTRAINT PK_T1  PRIMARY KEY (id1,id2)

    );
```

Les contraintes de tables

[CONSTRAINT nomContrainte] FOREIGNKEY (nomColonne [, nomColonne]...)

REFERENCES [nomSchema.]nomTable[(nomColonne [, nomColonne]...)]

[ON DELETE CASCADE]

[CONSTRAINT nomContrainte] CHECK (<condition>)

Exemple de création de table avec contraintes de tables : attention à l'ordre de création

Rajoutons à la table T1 une nouvelle table T2 :

```
CREATE TABLE  T1
      (id1  NUMBER
      ,id2  NUMBER

      ,CONSTRAINT PK_T1  PRIMARY KEY (id1,id2)
      );


CREATE TABLE T2
    (att1  NUMBER
    ,att2  NUMBER
    ,att3  NUMBER
     );
```

Rajoutez du code qui déclare (att2,att3) clé étrangère qui fait référence au couple (id1,id2) de la table T1 !

Exemple de création de table avec contraintes de tables : attention à l'ordre de création

Rajoutons à la table T1 une nouvelle table T2 :

```
CREATE TABLE  T1
      (id1  NUMBER
      ,id2  NUMBER

      ,CONSTRAINT PK_T1  PRIMARY KEY (id1,id2)
      );


CREATE TABLE T2
    (att1  NUMBER
    ,att2  NUMBER
    ,att3  NUMBER



    ,CONSTRAINT FK_T2_T1_att2att3 FOREIGN KEY (att2,att3)
                REFERENCES T1

    ) ;
```

Modification de la structure d'une table : Suppression

- Suppression d'une colonne : ALTER TABLE nomTable DROP COLUMN nomColonne;

- Suppression d'une contrainte nommée :

ALTER TABLE [nomSchéma.] nomTable

DROP CONSTRAINT nomContrainte [CASCADE|RESTRICT];

Le mot clé CASCADE, permet de supprimer toutes les contraintes d'intégrité référentielle qui se réfèrent aux clés uniques ou primaires de la table.

Par défaut c'est le CASCADE qui est exécuté si on ne met rien.

Le mot RESTRICT génère une erreur sur une contrainte d'intégrité référentielle qui se réfèrent aux clés uniques ou primaires de la table.

Modification de la structure d'une table : Suppression

Supprimez de la table Département la contrainte nommée PK_departement qui stipule que IdDep est une clé primair.

Modification de la structure d'une table : Suppression


```
ALTER TABLE Departement

        DROP CONSTRAINT PK_Departement RESTRICT ;
```

Une erreur se produit !!!! Pourquoi ?

Modification de la structure d'une table : Suppression


```
ALTER TABLE Departement

                DROP CONSTRAINT PK_Departement  CASCADE;
```

Le mot cascade est obligatoire car il supprime aussi dans la table Employe la contrainte FK_Employe_Departement qui définit une clé étrangère qui fait référence à la clé primaire idDep de département.

Modification de la structure d'une table : Suppression

- Suppression d'une contrainte PRIMARY KEY ou UNIQUE nommé ou anonyme :

ALTER TABLE [nomSchéma.] nomTable DROP PRIMARY KEY [CASCADE| RESTRICT];

ALTER TABLE [nomSchéma.] nomTable

DROP UNIQUE (nomColonne[,nomColonne]...) [CASCADE | RESTRICT];

Le mot clé CASCADE, permet de supprimer toutes les contraintes d'intégrité référentielle qui se réfèrent aux clés uniques ou primaires de la table.

Modification de la structure d'une table : Suppression

Sur l'exemple précédent nous avons utilisé drop constraint pour supprimer la contrainte de clé primaire de la table Département :

```
ALTER TABLE Departement

                DROP CONSTRAINT PK_Departement  CASCADE;
```

Utilisez au lieu de ça la syntaxe avec : drop primary key

Modification de la structure d'une table : Suppression


```
ALTER TABLE Departement

        DROP    PRIMARY   KEY  CASCADE;
```

Suppression (suite)

Suppression d'une table :

DROP TABLE [nomSchéma.] nomTable [CASCADE CONSTRAINTS];

Avec cascade constraints : toutes les contraintes des autres tables qui utilisent les colonnes de la table à détruire seront supprimées. **On a donc plus besoin de chercher un ordre de suppression**

Sans cascade constraints : si des contraintes référencent la table à détruire, un message d'erreur apparaîtra et la table ne sera pas détruite. **On a donc besoin de chercher un ordre de suppression**

Modification de la structure d'une table : Suppression

Forcez la suppression des tables en commençant par la table Département et sans se soucier de l'ordre de suppression.

Modification de la structure d'une table : Suppression


```
DROP TABLE Departement CASCADE CONSTRAINTS;

DROP TABLE Employe CASCADE CONSTRAINTS;
```

Si on enlève le premier cascade constraints une erreur se produira car la table Employe fait référence à la table Département.

Ajout d'une colonne

ALTER TABLE [nomSchéma.]nomTable

ADD (< définitionColonne >

[, < définitionColonne >]...

[, <contrainteTable>]...);

Si l'on ajoute une colonne à une table dont des colonnes sont déjà remplies, des champs NULL sont insérés dans cette colonne. Dans ce cas, on ne pourra évidemment pas positionner en même temps une contrainte NOT NULL.

Exemple : rajoutez une colonne numTel dans la table Employe. C'est une chaine sur 10 positions.

Ajout d'une colonne

ALTER TABLE [nomSchéma.]nomTable

ADD (< définitionColonne >

[, < définitionColonne >]...

[, <contrainteTable>]...);

ALTER TABLE Employe ADD (numTel varchar2(10));

Les numéros de téléphone des Employés déjà saisis dans la table seront à NULL.

Modification du type d'une colonne

Les modifications suivantes sont possibles :

accroître la taille : toujours possible,

réduire la taille : seulement si la colonne ne contient que des valeurs nulles,

modifier le type : même condition que pour la réduction de taille,

Modification du type d'une colonne (suite)

```
ALTER TABLE [nomSchéma.]nomTable

    MODIFY
            (  nomColonne [<typeColonne>]  [DEFAULT <expr>]
                                           [<contrainteColonne>]...
            [, nomColonne [<typeColonne>]  [DEFAULT <expr>]
                                           [<contrainteColonne>]... ]  ...
            );
```

La définition d'une colonne a presque la même syntaxe que celle de la commande CREATE TABLE sauf que le type de colonne peut ne pas être spécifié, auquel cas le type actuel est inchangé.

Exemple : on souhaite que le numéro de tél soit sur 12 position au lieu de 10 dans la table Employe.

Modification du type d'une colonne (suite)

ALTER TABLE [nomSchéma.]nomTable

MODIFY

(nomColonne [<typeColonne>] [DEFAULT <expr>]
[<contrainteColonne>]...
[, nomColonne [<typeColonne>] [DEFAULT <expr>]
[<contrainteColonne>]...] ...
);

Exemple : on souhaite que le numéro de tél soit sur 12 position au lieu de 10 dans la table Employe.

ALTER TABLE Employe MODIFY (numTel varchar2(12));

Modification du type d'une colonne (suite)

ALTER TABLE [nomSchéma.]nomTable

MODIFY

(nomColonne [<typeColonne>] [DEFAULT <expr>]
[<contrainteColonne>]...
[, nomColonne [<typeColonne>] [DEFAULT <expr>]
[<contrainteColonne>]...] ...

);

C'est également avec cette syntaxe que l'on peut rajouter une contrainte NOT NULL sur une colonne (s'il n'y a pas encore de valeur NULL dans la table, dans ce cas un message d'erreur apparaîtra signalant que des valeurs nulles ont été trouvées).

Exemple : Create Table Connexion (nomUtilisateur VARCHAR2(8));
ALTER TABLE Connexion MODIFY (nomUtilisateur NOT NULL);

Fonctionne aussi pour NULL : ALTER TABLE Connexion MODIFY (nomUtilisateur NULL);

Ajout d'une contrainte sur une colonne

Mis à part le cas de la contrainte NOT NULL traitée précédemment par la clause MODIFY, on utilisera les syntaxes suivantes :

Clé primaire

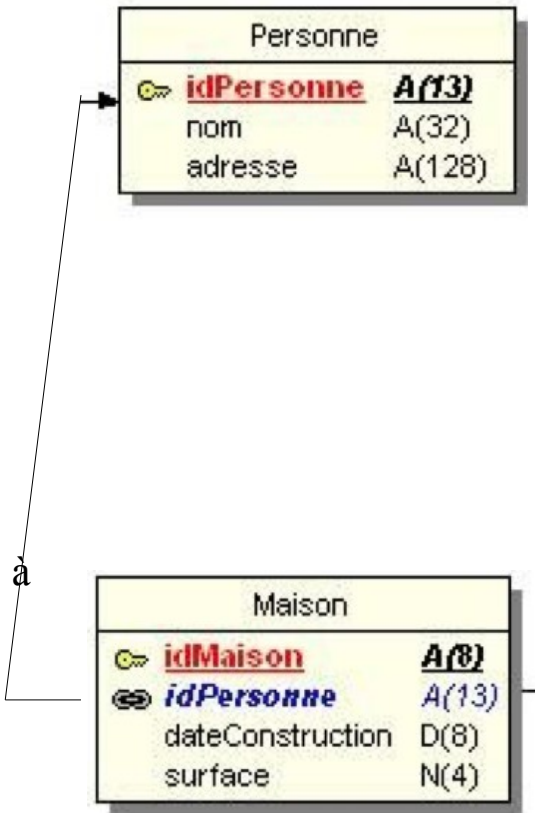
```
ALTER TABLE [nomSchéma.] nomTable
    ADD
        (CONSTRAINT nomContrainte
            PRIMARY KEY (nomColonne[,nomColonne]...));
```

Clé étrangère : très utilisé pour le passage MLDR → MPD

```
ALTER TABLE [nomSchéma.] nomTable
    ADD
        (CONSTRAINT nomContrainte
            FOREIGN KEY (nomColonne[,nomColonne]...)
                REFERENCES tableReference [(nomColonne[,nomColonne]...)]);
```

Script complet de traduction MLDR en syntaxe Oracle

Comment générer le MPD à partir d'un MLDR ?



Solution 1 : établir un ordre de création selon les contraintes de clés étrangères qui figurent dans le MLDR.

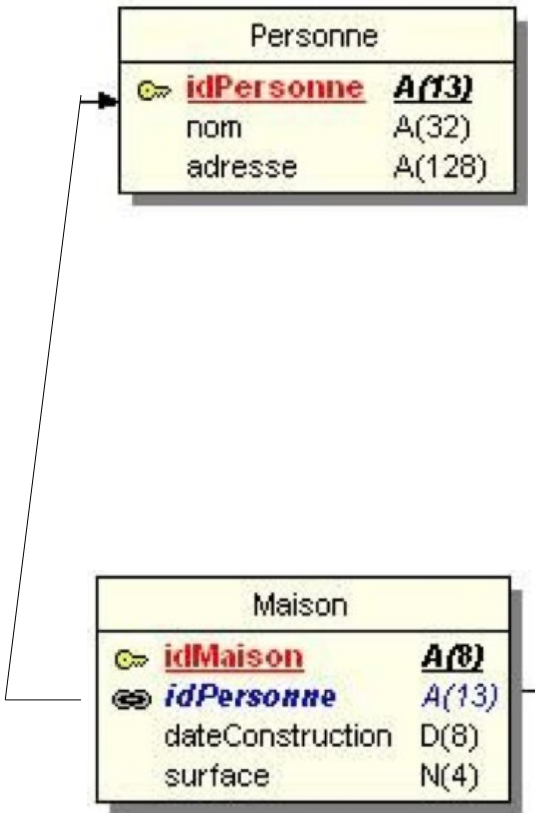
Seules les instructions de type CREATE TABLE seront alors utilisées à raison d'une par table.

Solution 2 : créer les tables dans n'importe quel ordre sans déclarer les contraintes de clés étrangères. Celles-ci seront ajoutées par la suite via l'instruction « ALTER TABLE ».

Solution 1

```
CREATE TABLE Personne
   ( idPersonne   VARCHAR2(13)
   , nom          VARCHAR2(32)
   , adresse      VARCHAR2(128)
   , CONSTRAINT PK_Personne PRIMARY KEY (idPersonne)
   );
```

```
CREATE TABLE Maison
   ( idMaison           VARCHAR2(8)
   , idPersonne         VARCHAR2(13)  NOT NULL
   , dateConstruction   DATE
   , surface            NUMBER(4)
   , CONSTRAINT PK_Maison PRIMARY KEY (idMaison)
   , CONSTRAINT FK_Maison_Personne_idPersonne
         FOREIGN KEY (idPersonne) REFERENCES Personne
   );
```

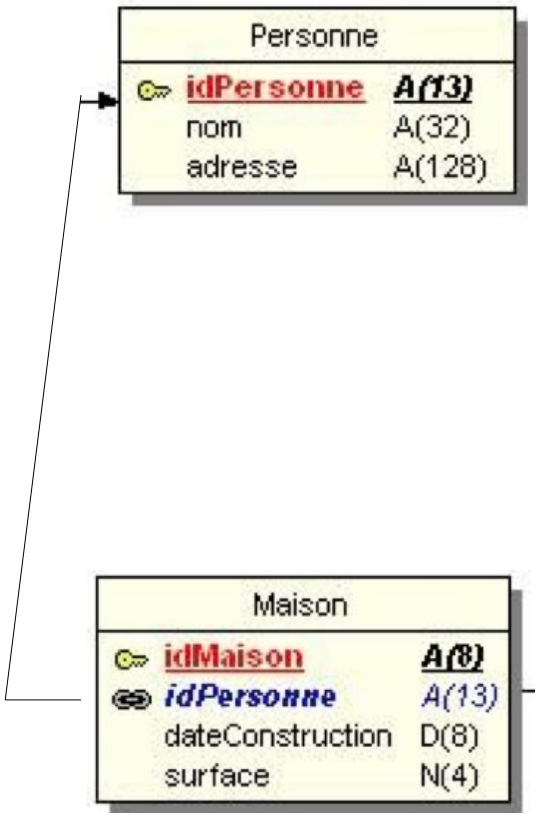


Solution 2

```
CREATE TABLE Personne
   ( idPersonne  VARCHAR2(13)
   , nom         VARCHAR2(32)
   , adresse     VARCHAR2(128)
   , CONSTRAINT PK_Personne PRIMARY KEY (idPersonne)
   );

CREATE TABLE Maison
   ( idMaison          VARCHAR2(8)
   , idPersonne        VARCHAR2(13)  NOT NULL
   , dateConstruction  DATE
   , surface           NUMBER(4)
   , CONSTRAINT PK_Maison PRIMARY KEY (idMaison)
   );

ALTER TABLE Maison
  ADD
   ( CONSTRAINT FK_Maison_Personne_idPersonne
        FOREIGN KEY (idPersonne)
            REFERENCES Personne
   );
```



Ajout d'une contrainte sur une colonne (suite)

Mis à part le cas de la contrainte NOT NULL traitée précédemment par la clause MODIFY, on utilisera les syntaxes suivantes :

Check

```
ALTER TABLE [nomSchéma.] nomTable
    ADD
        (CONSTRAINT nomContrainte
            CHECK (<condition>));
```

On peut éventuellement les combiner pour une même colonne en séparant les définitions de chaque contrainte par des virgules.

Modification d'une contrainte : il faudra d'abord supprimer la contrainte à modifier, puis la recréer avec les modifications. Seule exception : le passage de NOT NULL à NULL vu précédemment et éventuellement de NULL à NOT NULL.

ORACLE SQL – LMD

Langage de manipulation de données

Insertion

INSERT INTO [nomSchéma.] nomTable [(nomColonne[,nomColonne]...)]
VALUES (<exp> [,<expr>]...)

INSERT INTO [nomSchéma.] nomTable [(nomColonne[,nomColonne]...)]
<sous-requêtesSELECT>

L'insertion totale → toutes les colonnes en respectant l'ordre de création

Reprenons Employe(id,nomEmploye,salaire,dateEmbauche)

```
INSERT INTO Employe
        VALUES (1000,'Durand', 1000, to_date ('25/08/1987','DD/MM/YYYY'));
```

Notons la fonction to_date qui permet de convertir une chaine en date afin de l'insérer dans un champs DATE.

Insertion

INSERT INTO [nomSchéma.] nomTable [(nomColonne[,nomColonne]...)]
VALUES (<exp> [,<expr>]...)

INSERT INTO [nomSchéma.] nomTable [(nomColonne[,nomColonne]...)]
<sous-requêtesSELECT>

L'insertion partielle → seules les colonnes nommées sont valorisées. Attention, il faut valoriser toutes les colonnes définies en NOT NULL.

Reprenons Employe(id,nomEmploye,salaire,dateEmbauche)

INSERT INTO Employe (id,nomEmploye) VALUES (1000,'Durand') ;
Possible si la date de naissance et le salaire ne sont pas contraints par NOT NULL !

Suppression

DELETE [FROM] [nomSchéma.] nomTable [WHERE <condition>] ;

En l'absence de condition la table est vidée de toutes ses lignes.

```
DELETE FROM Employe
        WHERE salaire=1000 and to_char(dateEmbauche,'YYYY')='2000'
```

Notez bien les cotes autour de 2000 du fait de la conversion de la date en chaine.

Mise à jour

UPDATE [nomSchéma.] nomTable [(nomColonne[,nomColonne]...)]

SET nomColonne= <expr> [, nomColonne= <expr>] …

[WHERE <condition>] ;

En l'absence de condition les modifications sont appliquées à toutes les lignes.

Exemple : Affectez au salarié dont l'id est 1000, le salaire à 2000 et le tel à 0601012920.

Mise à jour

UPDATE [nomSchéma.] nomTable [(nomColonne[,nomColonne]...)]

SET nomColonne= <expr> [, nomColonne= <expr>] …

[WHERE <condition>] ;

En l'absence de condition les modifications sont appliquées à toutes les lignes.

UPDATE Employe SET salaire=2000, numTel='0601012920' WHERE id=1000;

Affecte le salaire à 2000 et le tel à 0601012920 lorsque l'id = 1000

Mise à jour avec date

```
UPDATE Client
        SET dateNaissance=to_date ('25/08/1987','DD/MM/YYYY')
        WHERE nom='Dupont' ;
```

Affecte la valeur 25/08/1987 à tous les client dont le nom est Dupont.

Du fait que l'attribut dateNaissance soit déclaré DATE nous oblige à utiliser la fonction to_date() qui convertit la chaine de caractères '25/08/1987' en une date sous format DD/MM/YYYY.