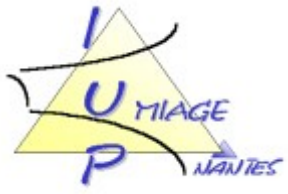
Module M4
Base de données

Chapitre 2

DDL
(Data Definition Language)

Auteurs : Laura Monceaux / Véronique Laime



Le LDD ou DDL (Data Definition Language) est la partie du SQL qui permet de ***créer une base de données.*** Il permet de créer une table, mais aussi de la modifier ou de la supprimer. Ce langage est utilisé par l'administrateur de la base de données.

2.1 Création des tables d'une base de données

Pour créer une table, la syntaxe est la suivante :

CREATE TABLE Nom_de_la_table (***colonne1*** ***Type_champ1*****,**
colonne2 ***Type_champ2*****,**
colonne3 ***Type_champ3,***
...);

Exemple : Si nous souhaitons gérer une base de données des étudiants de L3 info-gestion, nous avons besoin d'une table Etudiant permettant de stocker les informations concernant chaque étudiant : son numéro d'étudiant, son nom, son prénom, sa date de naissance, etc (les champs de la table). Il nous faut ainsi créer la table Etudiant comme suit :

create table Etudiant (noetu ***number,***
nom ***varchar(20),***
prenom ***varchar(20),***
...);

Mais lors de la création d'une table, d'autres informations sont nécessaires, dont :

- les contraintes d'intégrité
- les clés primaires et étrangères

A. Contraintes d'intégrité

Une contrainte d'intégrité doit être définie dès la création de la table afin que les données saisies par l'utilisateur soient conformes aux valeurs attendues.

A.1. Définir une valeur par défaut

La clause **DEFAUT** permet de donner par défaut une valeur à une donnée avant que celle-ci ne soit valuée (c'est-à-dire avant que l'utilisateur n'est saisie une valeur pour cette donnée).

La clause DEFAUT doit être suivie de la valeur donnée par défaut. Cette valeur peut être de type:

- Numérique
- Alphanumérique
- USER (nom de l'utilisateur)
- NULL (vide)
- CURRENT_DATE (date de saisie = date du système)
- CURRENT_TIME (heure de saisie = heure du système)
- CURRENT_TIMESTAMP (date et heure de saisie = date et heure du système)

A.2. Obliger l'utilisateur à saisir une valeur dans un champ

Le mot clé **NOT NULL** permet de spécifier qu'une valeur doit être saisie dans un champ. Le SGBDR refusera d'insérer dans une table un enregistrement (aussi appelé tuples) dont un champ comportant la clause NOT NULL n'est pas valué.

A.3. Définir une condition sur un champ

La clause **CHECK()** permet de tester un champ. La clause CHECK() comporte une condition logique portant sur une valeur indiquée entre les parenthèses. Si la valeur saisie est différente de NULL, le SGBDR va effectuer un test grâce à la condition logique. Celui-ci peut éventuellement être une condition avec des ordres SELECT...

A.4. Tester l'unicité d'une valeur

La clause **UNIQUE** permet de vérifier que la valeur saisie pour un champ n'existe pas déjà dans la table. Cela permet de garantir que toutes les valeurs d'une colonne d'unetable seront différentes.

Exemple : Reprenons l'exemple de notre table Etudiant – nous souhaitons que le nom, le prénom et le numéro d'un étudiant soient obligatoirement renseignés lors de l'ajout d'un tuple dans la table et que la date de naissance soit inférieure ou égale à la date du jour.

```
create table Etudiant (
        noetu number NOT NULL,
        nom varchar(20) NOT NULL,
        prenom varchar(20) NOT NULL,
        dateN number check (DateN <= Date()));
```

B. Définition des clés

Il existe deux types de clés dans une table : clé primaire et clé étrangère.

B.1. La clé primaire

Il est toujours nécessaire de définir la clé primaire d'une table dès sa création. La clé primaire peut être composée d'un ou de plusieurs champs. Toutes les valeurs de la clé primaire doivent être uniques. Une clé primaire permet d'***identifier sans ambiguïté chaque enregistrement (tuples ou ligne) de la table***.

La clause ***PRIMARY KEY (champ1, champ2...)*** permet ainsi de définir le ou les champs qui composent la clé primaire. Les champs composant la clé primaire ne peuvent prendre la valeur NULL et doivent être telles que deux enregistrements ne puissent avoir simultanément la même combinaison de valeurs pour ces champs.

Exemple : lors de son inscription, chaque étudiant se voit attribuer un numéro **noetu** qui va permettre de le distinguer de tous les autres étudiants. En effet, plusieurs étudiants peuvent avoir le même nom, le même prénom voire la même date de naissance…

```
create table Etudiant (noetu number NOT NULL,
                       nom varchar(20) NOT NULL,
                       prenom varchar(20) NOT NULL,
```

dateN ***number check (DateN <= Date())***
primary key (noetu));

B.2. La clé étrangère

Lors de la création des relations entre les tables de la base de données relationnelle, celles-ci sont reliées deux à deux par le champ qu'elles ont en commun. Ce champ est la clé primaire dans une table et la clé étrangère ou élément de la clé primaire dans l'autre table. Ce champ doit être de même type, même taille et même format. Lors de la définition de la table contenant une clé étrangère, il faut utiliser la clause ***FOREIGN KEY (champ1, champ2,…)*** suivie de la clause ***REFERENCES Nom_table_reliée (clé primaire)***.

Exemple : Si l'on souhaite stocker les notes d'un étudiant de L3 pour une matière donnée, on devra créer une nouvelle table Note stockant la note de CC et la note d'examen des étudiants pour une matière donnée :

create table NOTES (noetu ***number NOT NULL***,
codemat ***number NOT NULL***,
notecc ***real***,
noteex ***real***,
primary ***key(noetu, codemat)***,
foreign key (noetu) references Etudiant (noetu);

Ici un tuple est identifié par le numéro de l'étudiant et le code de la matière et le numéro de l'étudiant est une clé étrangère puisque qu'elle correspond à la clé de la table Etudiant.

B.3. L'intégrité référentielle (trigger)

Lorsque l'on relie les tables deux à deux on peut choisir ou non d'appliquer l'intégrité référentielle. Dans le cas où l'intégrité référentielle est appliquée il faut que le champ commun soit de même taille, de même type et de même format mais il faut également que les valeurs qui sont saisies dans le champ commun de la table liée (table où le champ commun est clé étrangère ou élément de la clé primaire) existent dans la table maître (table où le champ commun est la clé primaire).

Pour ceux qui connaissent Access, lorsque l'on applique l'intégrité référentielle, le SGBDR propose deux options qui permettent de répercuter ou non, dans la ou les tables liées, les modifications ou suppressions faites, sur les valeurs du champ commun, dans la table maître.

Les **triggers** (ou gâchettes) pour chaque relation établie entre deux tables (une maître et l'autre liée) sont:

- Pour la **mise à jour** :
  - *ON UPDATE* est suivi d'arguments entre accolades permettant de spécifier l'action à réaliser dans la table liée en cas de modification d'une valeur de la clé primaire d'un enregistrement de la table maître.
  - *CASCADE* indique la répercussion de la modification dans la table liée en cas de modification d'une valeur de la clé primaire dans la table maître. Par exemple, si, dans la table maître des étudiants le numéro de l'étudiant 200814 est modifié en 200029 les enregistrements de la table liée qui concernait cet étudiant 200814 doivent ils

automatiquement être modifiés en 200029 ?

- *RESTRICT* indique une erreur en cas de modification d'une valeur de la clé primaire dans la table maître.
- *SET NULL* place la valeur NULL dans la clé étrangère de l'enregistrement de la table liée en cas de modification d'une valeur de la clé primaire dans la table maitre.
- *SET DEFAULT* place la valeur par défaut (qui suit ce paramètre) dans la clé étrangère de l'enregistrement de la table liée en cas de modification d'une valeur de la clé primaire dans la table maitre.

➢ Pour l'**effacement**:

- *ON DELETE* est suivi d'arguments entre accolades permettant de spécifier l'action à réaliser en cas d'effacement d'une valeur de la clé primaire d'un enregistrement de la table maître.
- *CASCADE* indique la répercussion de la suppression dans la table liée en cas de suppression d'un enregistrement dans la table maître. Par exemple, si, dans la table maître des étudiants l'étudiant numéro 200814 est supprimé, les enregistrements de la table liée qui concernait cet étudiant 200814 doivent ils automatiquement être supprimés, dans la table liée?
- *RESTRICT* indique une erreur en cas de suppression d'une valeur de la clé primaire dans la table maître.
- *SET NULL* place la valeur NULL dans la clé étrangère de l'enregistrement de la table liée en cas de suppression d'une valeur de la clé primaire dans la table maitre.
- *SET DEFAULT* place la valeur par défaut (qui suit ce paramètre) dans la clé étrangère de l'enregistrement de la table liée en cas de modification d'une valeur de la clé primaire dans la table maitre.

Cette notion de trigger sera vue en L3 dans le cours 'Base de données 2'. Nous ne verrons pas dans les TD de ce module M4 du DVD MIAGE son utilisation.

C. Exemple de création d'une base de données

Si nous reprenons ainsi notre exemple de gestion des étudiants. Dans un premier, nous allons déclarer les 2 tables Matiere (stockant les différentes informations sur les matières que peuvent suivre les étudiants) et Etudiant (stockant les informations des étudiants inscrits) puis la table Notes qui fait référence aux deux autres (stockant les matières suivies par les étudiants ainsi que les notes obtenues dans ces matières).

```
create table Matiere ( codemat number NOT NULL primary key,
                       titre varchar(40) NOT NULL,
                       responsable varchar(20) NOT NULL);

create table Etudiant ( noe number NOT NULL primary key,
                        nom varchar(20) NOT NULL,
                        prenom varchar(20) NOT NULL,
                        dateN number check (DateN <= Date()));

create table Notes (    noetu number NOT NULL,
                        codemat number NOT NULL,
                        notecc real check ((notecc <= 20) and (notecc >= 0)),
                        noteex real check ((noteex <= 20) and (noteex >= 0)),
                        primary key (noetu, codemat),
                        foreign key (noetu) references Etudiant (noe));
```

2.2 Insertion de données dans la base de données

L'insertion de nouvelles données dans une table se fait grâce à l'ordre **INSERT**, qui permet d'insérer de nouveaux enregistrements dans la table. L'ordre INSERT attend la clause **INTO**, suivie du nom de la table, ainsi que du nom de chacun des champs entre parenthèses (les champs omis prendront la valeur NULL par défaut).

Les valeurs à insérer peuvent être précisées de deux façons :

- avec la clause **VALUES** : un seul enregistrement est inséré, il contient comme valeurs, l'ensemble des valeurs passées en paramètre de la clause VALUES. Les données sont affectées aux champs dans l'ordre dans lequel les champs ont été déclarés dans la clause INTO

```
INSERT INTO Nom_de_la_table(champ1,champ2,champ3,...)
VALUES (Valeur1,Valeur2,Valeur3,...)
```

  Champ1 prend la valeur Valeur1, le champ2 prend la valeur Valeur2…
  Lorsque chaque champ de la table est modifié, l'énumération de l'ensemble des champs est facultative. Si les valeurs sont des chaînes de caractères, il faut les encadrer de guillemets.

- avec la clause **SELECT**: plusieurs enregistrements peuvent être insérés, il contient comme valeurs, l'ensemble des valeurs découlant de la sélection. Les données sont affectées aux champs dans l'ordre dans lequel les champs ont été déclarés dans la clause INTO

```
INSERT INTO Nom_de_la_table1(champ1,champ2,...)
SELECT champ1,champ2,... FROM Nom_de_la_table2
WHERE condition;
```

  Lorsque l'on remplace un nom de champ suivant la clause SELECT par une constante, sa valeur est affectée par défaut aux enregistrements. Attention, le Nom_de_la_table1 doit être différent de Nom_de_la_table2.

  Si l'on reprend l'exemple de notre base de la gestion des étudiants, il faut insérer d'abord des tuples dans les tables Matiere et Etudiant pour pouvoir en insérer dans Notes à cause des références faites aux attributs 'noe' et 'codemat' des 2 tables précédentes :

```
insert into Etudiant values ('E9658C','Dupont','Pierre',1992);
insert into Matiere values ('M4','Base de données','Laura Monceaux');
insert into Notes values ('E9658C','M4',14,12);
```

2.3. Modification des données de la base

La modification ou mise à jour de données consiste à modifier des enregistrements dans une table grâce à l'ordre UPDATE. La modification à effectuer est précisée après la clause SET. Il s'agit d'une affectation d'une valeur à un champ grâce à l'opérateur = suivi d'une expression algébrique, d'une constante ou du résultat provenant d'une clause SELECT. La clause WHERE permet de préciser les enregistrements sur lesquels la mise à jour aura lieu

```
UPDATE Nom_de_la_table
SET Champ = Valeur_Ou_Expression
WHERE condition;
```

Attention: cet ordre est irréversible. Il faut donc s'assurer que les enregistrements sélectionnés sont bien ceux sur lesquels on souhaite agir.

R6 : Augmenter de 10% le prix des produits dont le prix est actuellement inférieurà 3 euros.

```
UPDATE Produit SET Produit.PUProd = [puprod]*(1+0.1)
WHERE Produit.PUProd<3;
```

R7 : Supprimer le produit "Ixprim 37,5 mg".

```
DELETE Produit.CodePod
FROM Produit
WHERE Produit.CodePod Like "Ixprim375";
```

2.4. Suppression de données

La suppression de données dans une table se fait grâce à l'ordre *DELETE*. Celui-ci est suivi de la clause *FROM*, précisant la table sur laquelle la suppression s'effectue, puis d'une clause *WHERE* qui décrit la qualification, c'est-à-dire l'ensemble des lignes qui seront supprimées.

```
DELETE FROM Nom_de_la_table
WHERE qualification
```

Attention: cet ordre est irréversible. Il faut donc s'assurer que les enregistrements sélectionnés sont ceux que l'on souhaite supprimer. Il est donc recommandé de commencer par sauvegarder la table dans laquelle on désire supprimer le ou les enregistrements.

2.5 Types de données

Les différents types de données que manipule SQL ne sont pas différents des types manipulés par d'autres langages.

A. Type numérique

FORMATS	VALEUR CONTENUE
NUMERIC (ou **DECIMAL** ou **DEC**)	nombre décimal à représentation exacte à échelle et précision facultatives
INTEGER (ou **INT**)	entier long
SMALLINT	entier court
FLOAT	réel à virgule flottante dont la représentation est binaire à échelle et précision obligatoire
REAL	réel à virgule flottante dont la représentation est binaire, de faible précision

DOUBLE PRECISION	réel à virgule flottante dont la représentation est binaire, de grande précision
BIT	chaîne de bit de longueur fixe
BIT VARYING	chaîne de bit de longueur maximale

Pour les types réels NUMERIC, DECIMAL, DEC et FLOAT, on doit spécifier le nombre de chiffres significatifs et la précision des décimales après la virgule.

Exemple : NUMERIC (15,2) signifie que le nombre comportera au plus 15 chiffres significatifs dont deux décimales.

ATTENTION :
Pour des calculs comptables il est indispensable d'utiliser le type DECIMAL car ce type se comporte comme un entier dans lequel la virgule n'est qu'une représentation positionnelle. Pour des calculs scientifiques il vaut mieux utiliser le type FLOAT, plus rapide dans les calculs.

B. Type alphanumérique

FORMATS	VALEUR CONTENUE
CHARACTER (ou **CHAR**)	valeurs alphabétique de longueur fixe
CHARACTER VARYING (ou **VARCHAR** ou **CHAR VARYING**)	valeur alphabétique de longueur maximale fixée
NATIONAL CHARACTER (ou **NCHAR** ou **NATIONAL CHAR**)	valeurs alphabétique de longueur fixe
NATIONAL CHARACTER VARYING (ou **NCHAR VARYING** ou **NATIONAL CHAR VARYING**)	valeur alphabétique de longueur maximale fixée sur le jeu de caractère du pays

Character et Character varying sont des types de données sont codés sur 2 octets (EBCDIC ou ASCII) et on doit spécifier la longueur de la chaîne. National character et National character varying sont des types de données sont codés sur 4 octets (UNICODE) et on doit spécifier la longueur de la chaîne.

Le nombre maximum de caractères contenus dans un de ces types de données dépend du SGBDR.

Exemple :
```
NOM_CLIENT CHAR(32)
OBSERVATIONS VARCHAR(32000)
```
Exemple :
```
NOM_CLIENT NCHAR(32)
OBSERVATIONS NCHAR VARYING(32000)
```

C. Type Date et Heure

FORMATS	VALEUR CONTENUE

DATE	date du calendrier grégorien
TIME	temps sur 24 heures
TIMESTAMP	combiné date et temps
INTERVAL	intervalle de date / temps qui est un entier

Le format standard de la date est: AAAA-MM-JJ. Où JJ représente les 2 chiffres du jour, MM les 2 chiffres du mois et AAAA les 4 chiffres de l'année.
Le format standard de l'heure est hh:mm:ss.nnn. Où hh représente les 2 chiffres des heures, mm les 2 chiffres des minutes, ss les 2 chiffres des secondes et nn les 3 chiffres des millisecondes.

Le type INTERVAL est rarement présent dans les SGBDR. Sa définition se fait à l'aide de la syntaxe suivante : INTERVAL Max [TO Min]
Où Min est optionnel et ne peut prendre qu'une mesure du temps plus petite que Max. Min et Max peuvent prendre les valeurs : YEAR, MONTH, DAY, HOUR,MINUTE, SECOND

Exemples :

JOUR	INTERVAL DAY
TRIMESTRE	INTERVAL MONTH TO DAY

D. Autres types courants, hors norme SQL 92

FORMATS	VALEUR CONTENUE
BOOLEAN (ou **LOGICAL**)	La norme laisse à chaque éditeur de SGBDR le soin de concevoir ou non un booléen " à sa manière ".
MONEY	est un sous type du type NUMERIC avec une échelle maximale et une précision de deux chiffres après la virgule.
BYTES (ou **BINARY**)	Type binaire (octets) de longueur devant être précisée. Permet par exemple le stockage d'un code barre.
AUTOINC	entier à incrément automatique par trigger.
TEXT	suite longue de caractères de longueur indéterminé
IMAGE	stockage d'image dans un format déterminé
OLE	stockage d'objet OLE (Windows)