



INFORMATIQUE

Laboratoires de bases de données

Laboratoire n°1 DDL et gestion des données

par Danièle BAYERS et Louis SWINNEN
Révision 2024 : Vincent Reip

Ce document est disponible sous licence Creative Commons indiquant qu'il peut être reproduit, distribué et communiqué pour autant que le nom des auteurs reste présent, qu'aucune utilisation commerciale ne soit faite à partir de celui-ci et que le document ne soit ni modifié, ni transformé, ni adapté.



<http://creativecommons.org/licenses/by-nc-nd/2.0/be/>

La Haute Ecole Libre Mosane (HELMo) attache une grande importance au respect des droits d'auteur. C'est la raison pour laquelle nous invitons les auteurs dont une oeuvre aurait été, malgré tous nos efforts, reproduite sans autorisation suffisante, à contacter immédiatement le service juridique de la Haute Ecole afin de pouvoir régulariser la situation au mieux.

Octobre 2024

1. Introduction

L'objectif de ce laboratoire est de montrer, par quelques exemples pratiques, la **création**, la **modification**, la **suppression** de la structure d'une table. Ensuite, nous aborderons les contraintes qui peuvent être définies sur une table. Nous continuerons en détaillant **l'insertion**, **la modification et la suppression d'un enregistrement**.

Nous allons commencer par quelques définitions simples afin de (re-)préciser certaines notions :

- **Base de données** : définitions
 - Grande collection de données exploitées par un ensemble d'applications
 - « *Lot d'informations stockées dans un dispositif informatique. Les technologies existantes permettent d'organiser et de structurer la base de données de manière à pouvoir facilement manipuler le contenu et stocker efficacement de très grandes quantités d'informations.* ». Source : Wikipedia
 - Système d'organisation de l'information, conçu pour une localisation et une mise à jour rapide et facile des données. Une base de données organise l'information qu'elle contient en tables, en champs (les colonnes) et en enregistrements (les lignes).
 - « *Chaque enregistrement correspond à un item stocké dans la base de données.* » Source : dicodunet
- **SGBD** (Système de gestion de base de données) : définitions
 - Intermédiaire entre les applications et la base de données offrant (entre autres) les fonctionnalités de lecture, d'insertion, de modification et de suppression de données.
 - « *Un système de gestion de base de données est un ensemble de logiciels qui sert à la manipulation des bases de données. Il sert à effectuer des opérations ordinaires telles que consulter, modifier, construire, organiser, transformer, copier, sauvegarder ou restaurer des bases de données. Il est souvent utilisé par d'autres logiciels ainsi que les administrateurs ou les développeurs.* » Source : Wikipedia

AVERTISSEMENT ! Ce laboratoire aborde les éléments importants des bases de données de manière empirique. Il convient de se référer au cours théorique pour trouver des définitions plus précises des notions présentées ici.

2. La base de données

Supposons que nous souhaitions créer la table correspondant à la relation suivante :

Client
<u>NCLI</u>
Nom
Adresse
Localite
cat
compte
id: NCLI

Il faut commencer par définir **la structure** des enregistrements avant de pouvoir remplir cette table. Pour créer la structure, nous utiliserons l'instruction CREATE TABLE. Il est possible de modifier une structure créée grâce à ALTER TABLE. Enfin, supprimer une table peut se faire à l'aide de DROP.

2.1 CREATE TABLE

Format général de cette commande :

```
CREATE TABLE nom_table (
    nom_attribut_1      type  [contraintes],
    nom_attribut_2      type  [contraintes],
    ...
    [contrainte1]
    [contrainte2]
    ...
);
```

Ce qu'il faut respecter :

- Pas de nom d'attributs répétés
- Pas d'espace dans les noms d'attributs et de table
- Le type de l'attribut est obligatoire
- Des contraintes peuvent être définies sur un attribut. Elles sont alors placées le plus souvent en regard de cet attribut. Si des contraintes concernent plusieurs attributs, elles sont regroupées à la fin de la commande CREATE TABLE (c'est ce qu'on appelle les contraintes de table).

Les différents SGBD imposent des limites quant aux nombres de caractères pour les noms de tables et d'attributs. Il convient de se reporter à la documentation du SGBD pour vérifier si ces contraintes sont bien respectées.

2.1.1 Quelques types standard définis en SQL-2

La norme SQL-2 (ou SQL-92) définit plusieurs types de données. Nous allons voir ici quelques types standards que nous utiliserons dans le cadre de ces laboratoires.

Type	Explication
INTEGER	Représente un entier. La taille de cet entier dépend du type de SGBD utilisé
NUMERIC(p,s) DECIMAL(p,s)	Définit une donnée numérique avec fraction décimale fixe. <i>p</i> définit la précision alors que <i>s</i> définit l' échelle . Sous Oracle, la précision est <i>par défaut</i> fixée à 39 ou 40 et l'échelle à 0. Sous SQL Server, la précision est fixée par défaut à 18 et l'échelle à 0.
REAL	Représente un réel. La précision de ce dernier dépend de l'implémentation.
FLOAT(p)	Représente un réel. Il est possible de définir la précision dans ce type de données.
CHAR(t)	Définit une zone de <i>t</i> caractères. Les caractères non-utilisés sont placés à blanc.
VARCHAR(t)	Définit une zone de <i>t</i> caractères maximum. Seul les caractères présents occupent de l'espace. Il n'y a donc pas de « remplissage ».
DATE	Champ pouvant contenir une date. Il faut remarquer que malheureusement, les dates en SQL-2 posent beaucoup de problèmes car les fournisseurs de SGBD proposent très souvent des implémentations diverses et parfois incompatible.

	<p>Sous SQL Server, nous utiliserons le plus souvent le type DATE (\geq v.2005) qui stocke la date sous la forme AAAA-MM-JJ ou SMALLDATETIME (\geq v. 2000) qui stocke la date et l'heure sous la forme AAAA-MM-JJ hh:mm:ss.</p> <p>Sous Oracle, le type DATE est défini et il permet de stocker la date et l'heure.</p>
--	--

Il existe encore bien d'autres types (TIME, TIMESTAMP, ...) utilisés en fonction des usages particuliers. Nous aborderons lors d'une prochaine séance les différentes fonctions qui permettent de gérer ces types de données, de réaliser des conversions, ...

Certains SGBD autorisent également les utilisateurs (i.e. les programmeurs) à spécifier des types particuliers (USER DOMAIN).

2.1.2 Les contraintes

Lors de la définition d'un attribut, en plus de spécifier un type (qui est déjà une contrainte sur le domaine des valeurs permises), il est possible de définir d'autres contraintes.

Quelques contraintes standard :

NOT NULL	<p>Indiquée directement à la suite de l'attribut, elle rend l'attribut obligatoire. Ainsi, lors d'une insertion ou modification d'un enregistrement, une valeur doit absolument être indiquée pour cet attribut.</p> <p>Sans cette contrainte, l'attribut peut, en plus de l'ensemble des valeurs autorisées par son type (i. e. contrainte de domaine), prendre la valeur particulière <i>NULL</i> qui signale l'absence de valeur pour cet attribut.</p>
PRIMARY KEY	<p>Cette contrainte permet de spécifier la clé primaire (identifiante) d'une table. En conséquence, 2 enregistrements ayant une même valeur de clé ne peuvent exister. Si la clé est constituée d'un seul attribut, la contrainte peut alors être mentionnée directement à la suite de cet attribut. Par contre, si la clé primaire est composée de plusieurs attributs, il est nécessaire de définir celle-ci en tant que contrainte de table (à la fin de l'instruction CREATE TABLE).</p> <p>Les attributs composant la clé primaire sont obligatoires (contrainte NOT NULL).</p> <p>Il ne peut y avoir qu'une seule clé primaire par table.</p>
UNIQUE	<p>Cette contrainte permet de définir une clé secondaire ou candidate. Lorsque cette contrainte est positionnée pour un ou un ensemble d'attributs, elle exprime qu'il ne peut exister dans la table 2 enregistrements ayant même valeur pour ce(s) attribut(s).</p> <p>Si la clé candidate n'est pas composée de plusieurs attributs, elle peut être exprimée directement après la définition de celui-</p>

	ci. Dans le cas contraire, la clé candidate doit faire l'objet d'une définition séparée.
CHECK(expression)	Cette contrainte oblige le SGBD à vérifier que tout ajout ou modification dans une table respecte bien l'expression indiquée.
DEFAULT	Cette contrainte particulière permet d'indiquer une valeur par défaut pour un attribut. Cette contrainte permet très souvent d'éviter des valeurs <i>NULL</i> qui sont souvent particulières à gérer.
REFERENCES table (attribut)	<p>Cette contrainte indique la présence d'une clé étrangère. Une clé étrangère est un <i>lien</i> entre 2 tables (une sorte de pointeur). Les tables sont liées entre-elles simplement en important une clé (le plus souvent la clé primaire) comme attribut d'une autre table. Ainsi, comme on peut le lire dans [1] : « <i>une clé étrangère est un attribut ou un groupe d'attributs dans une table T1, dont les valeurs doivent exister comme valeurs de la clé candidate dans une table T2. T1 et T2 ne sont pas nécessairement distinctes.</i> »</p> <p>En résumant, on peut dire qu'une clé étrangère est un attribut ou un groupe d'attributs qui est (sont) clé candidate dans une autre table (la clé candidate peut être la clé primaire).</p> <p>Il faut également remarquer que lors de l'insertion d'un enregistrement, il est primordial que la valeur mentionnée dans cet attribut existe dans la table qui lui est liée.</p> <p>Si la clé étrangère est constituée d'un seul attribut, la contrainte peut être mentionnée juste après celui-ci. Dans le cas contraire, il est nécessaire de l'exprimer séparément.</p> <p>Cette contrainte très importante est nommée contrainte d'intégrité référentielle.</p>

2.1.3 Exemples

```
CREATE TABLE client (
  ncli CHAR(4) PRIMARY KEY,
  nom VARCHAR(20) NOT NULL,
  adresse VARCHAR(30),
  localite VARCHAR(20),
  cat CHAR(2),
  compte DECIMAL(9,2) ) ;
```

Dans cet exemple, on crée une table *client* dont l'attribut *ncli* est la clé primaire. Son type est une suite de 4 caractères. Cet attribut est obligatoire. Le *nom* est le second attribut, il est également obligatoire et son type est une suite variable de caractères de taille maximale égale à 20. L'*adresse*, la *localité*, la *catégorie* et le *compte* sont des attributs facultatifs. En particulier, le *compte* est une donnée numérique de 9 chiffres au total dont 2 chiffres se trouvent après la virgule.

Variante :

```
CREATE TABLE client (  
  ncli CHAR(4),  
  nom VARCHAR(20) NOT NULL,  
  adresse VARCHAR(30),  
  localite VARCHAR(20),  
  cat CHAR(2),  
  compte DECIMAL(9,2),  
  PRIMARY KEY(ncli)  
) ;
```

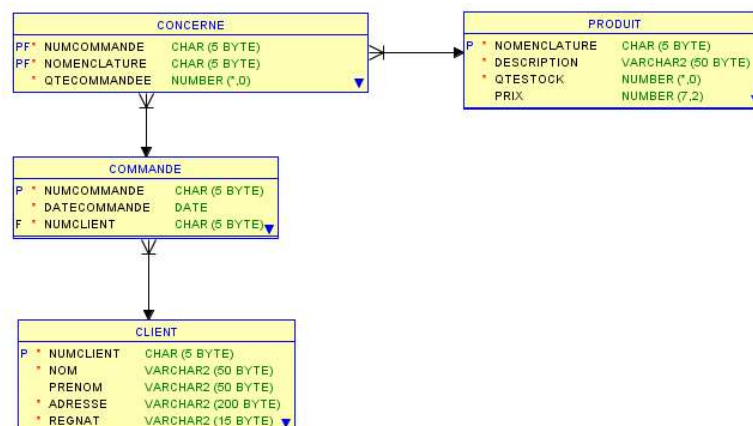
Cet exemple montre que la contrainte *primary key* peut être placée à la fin de la création de la table. Il faut noter que si la clé primaire est composée de plusieurs attributs, cette forme est obligatoire.

On peut également exprimer quelques contraintes particulières :

```
CREATE TABLE client (  
  ncli CHAR(4),  
  nom VARCHAR(20) NOT NULL,  
  adresse VARCHAR(30),  
  localite VARCHAR(20),  
  cat CHAR(2),  
  compte DECIMAL(9,2),  
  PRIMARY KEY(ncli),  
  CHECK(cat IN('C1','C2','C3','C4')),  
  CONSTRAINT compte_positif CHECK(compte >= 0)  
) ;
```

La première contrainte exprime que l'attribut *catégorie* doit être compris parmi les valeurs *C1*, *C2*, *C3* ou *C4*. Le mot clé *IN* sera détaillé ultérieurement lors de l'étude des requêtes. La seconde contrainte exprime le fait que la valeur du compte d'un client ne peut être négative. Cette seconde contrainte sera associée à un nom (compte_positif) qui apparaîtra dans le message d'erreur généré par le SGBD lorsque la contrainte est violée.

Enfin, en guise d'exemple, voici le script de création de la base de données dont le schéma est donné ci-dessous.



```

CREATE TABLE client (
  NumClient CHAR(5) PRIMARY KEY,
  Nom VARCHAR(50) NOT NULL,
  Prenom VARCHAR(50),
  Adresse VARCHAR(200) NOT NULL,
  RegNat VARCHAR(15) NOT NULL
);

CREATE TABLE produit (
  Nomenclature CHAR(5) PRIMARY KEY,
  Description VARCHAR(50) NOT NULL,
  QteStock INTEGER NOT NULL,
  Prix DECIMAL(7,2)
);

CREATE TABLE commande (
  NumCommande CHAR(5) PRIMARY KEY,
  DateCommande DATE NOT NULL,
  NumClient CHAR(5) NOT NULL REFERENCES Client(NumClient)
);

CREATE TABLE concerne (
  NumCommande CHAR(5) REFERENCES Commande(NumCommande),
  Nomenclature CHAR(5) REFERENCES Produit(Nomenclature),
  QteCommandee INTEGER NOT NULL,
  PRIMARY KEY(NumCommande, Nomenclature)
);

```

2.2 ALTER TABLE

Cette commande permet de modifier la structure d'une table. Modifier la structure d'une table n'est pas nécessairement une opération anodine. Par exemple, lors de l'ajout d'un attribut dans une table, que faire avec les enregistrements existants ?

Très souvent, la modification de la structure concerne soit une table vide, soit introduit des astuces pour les enregistrements existants. Par exemple, l'ajout d'un champ avec mention d'une clause DEFAULT ou encore l'ajout d'un champ pouvant prendre la valeur NULL.

Il faut, par contre, faire très attention à la suppression d'un attribut dans une table car cela peut conduire à des problèmes d'intégrité (suppression d'une clé candidate utilisée comme clé étrangère, ...).

Opérations possibles :

- **ADD** qui permet d'ajouter un attribut, une contrainte, ...
- **MODIFY** qui permet de modifier le type d'un attribut ou changer sa précision
- **DROP** qui permet de supprimer un attribut, une contrainte, ...

Format :

```

ALTER TABLE nom_table
  [ADD|MODIFY] attribut type [contraintes]
  [DROP COLUMN|CONSTRAINT] nom
  [ADD CONSTRAINT] nom_contrainte type [contraintes];

```

2.2.1 Exemples

```
ALTER TABLE client
  ADD reg_nat DECIMAL(11) NOT NULL UNIQUE ;
```

Dans cet exemple, nous ajoutons à la table `client` l'attribut `reg_nat` devant contenir le numéro d'inscription au registre national. Cet attribut est une clé candidate et doit être définie comme unique. De par ce fait, il serait impossible que 2 clients différents disposent du même numéro d'inscription au registre national.

```
ALTER TABLE client
  ADD CONSTRAINT FK_CatClient
  FOREIGN KEY (CatClient) REFERENCES categorieclient(NumCategorie)
```

Dans cet exemple, nous ajoutons à la table `client` une contrainte d'intégrité référentielle appelée `FK_CatClient`. L'attribut `CatClient` de la table `client` devient ainsi une clé étrangère référençant l'attribut `NumCategorie` de la table `categorieclient`.

2.3 DROP TABLE

Cette commande permet de modifier sensiblement la structure de la base de données en supprimant une table. Elle doit toujours être utilisée avec la plus grande précaution car les éventuelles données contenues dans la table seront perdues.

```
DROP TABLE nom_table ;
```

2.3.1 Exemple

```
DROP TABLE client ;
```

Dans cet exemple, la table *client* est supprimée par le SGBD.

2.4 INSERT INTO

La commande `INSERT INTO` permet d'ordonner l'insertion d'un enregistrement dans la table. Il est nécessaire de fournir une valeur pour tous les attributs obligatoires (déclaré `NOT NULL`) ne disposant pas de valeur par défaut (option `DEFAULT`).

La forme de la commande est la suivante :

```
INSERT INTO nom_table (attr1, attr2, ..., attrn)
  VALUES (val1, val2, ..., valn)
```

Cette commande provoque la création d'un enregistrement dans la table en fixant respectivement, pour les attributs `attr1, attr2, ..., attrn` les valeurs `val1, val2, ..., valn`.

Il faut remarquer que la liste d'attributs n'est obligatoire que si la commande ne fixe pas une valeur pour tous les attributs de la table. En effet, si des attributs sont facultatifs ou acceptent une valeur par défaut, il faut lister les attributs pour lesquels une valeur est fournie.

2.4.1 Exemples

```
INSERT INTO client VALUES ('C001', 'Tintin', 'Rue du château 3',
'Moulinsart', 'C1',0) ;
```


Dans cet exemple, nous ajoutons dans la table client l'enregistrement identifié comme C001 reprenant l'ensemble des valeurs. Remarquons que, comme nous donnons une valeur à chaque attribut, il n'est pas nécessaire de lister les champs de la table.

```
INSERT INTO client(ncli,nom) VALUES ('C002', 'Dupont');
```

Voici un exemple dans lequel une valeur n'est pas donnée pour chaque champ. Il est donc nécessaire de lister les attributs qui seront initialisés. Dans cet exemple et au vu de la définition de la table, les autres champs ont, comme valeur, *NULL*.

2.5 UPDATE

La commande UPDATE permet de modifier des enregistrements. Grâce à cette commande, la valeur d'un attribut ou plusieurs attributs peut être mise à jour. Il faut toujours être vigilant à bien identifier le ou les enregistrements à mettre à jour.

Format de la commande :

```
UPDATE nom_table  
  SET nom_attribut1 = valeur1[, ..., nom_attributn = valeurn]  
WHERE attribut = valeur
```

Cette commande mettra à jour la table mentionnée en remplaçant les valeurs des attributs listés par les nouvelles valeurs mentionnées en paramètres pour tous les enregistrements répondant à la condition énoncée dans le WHERE.

Nous verrons plus loin dans les séances de laboratoires qu'il est possible de construire des requêtes retournant des valeurs. Il faut savoir que de telles requêtes peuvent être utilisées comme « valeur » dans l'argument SET. Ce champ peut également mentionner une expression (arithmétique par exemple si la donnée le permet, ...).

2.5.1 Exemples

```
UPDATE client  
  SET cat = 'C2', adresse='rue de la police, 4'  
WHERE nom = 'Dupont';
```

Dans cet exemple, tous les enregistrements pour lesquels le nom est *Dupont* sont modifiés. Dans cette modification, la *catégorie* est fixée à la valeur *C2* et l'attribut *adresse* est initialisé.

2.6 DELETE

La commande DELETE permet d'ordonner la suppression d'un, de plusieurs ou même de la totalité des enregistrements présents dans une table. Il faut toujours identifier très clairement les enregistrements à supprimer et utiliser cette commande avec une extrême précaution.

Format de la commande :

```
DELETE FROM nom_table  
WHERE attribut = valeur
```

Cette commande ordonne dans la table mentionnée la suppression de tous les enregistrements répondant à la condition exprimée dans le WHERE.

Comme mentionné ci-dessus, il est également possible d'utiliser dans le champ « valeur » (énoncé dans le WHERE de la commande) une requête plutôt qu'une valeur précise.

Attention ! Ecrire une requête DELETE FROM sans la clause WHERE revient à vider la table de son contenu.


3. Accès à Oracle

Dans le cadre de ce laboratoire, nous utiliserons 2 SGBD : *Microsoft SQL Server* et *Oracle*. *Oracle* est installé sur le serveur dont l'adresse IP est 192.168.132.200 (port 11521). Il s'agit d'un serveur Linux qui est accessible aux étudiants pour réaliser leurs laboratoires.

La version d'Oracle installée est la version gratuite Oracle Express Edition 11g. Cette version est disponible pour les postes Windows et Linux. L'interaction avec Oracle peut se faire au moyen du client Oracle SQL Developer. Ce logiciel est installé sur les machines des laboratoires et vous pouvez le télécharger gratuitement sur le site de Oracle (<https://www.oracle.com/database/sqldeveloper/technologies/download/>) ou sur l'espace '[Ressources informatiques](#)' si vous désirez l'installer sur votre ordinateur personnel.

3.1 Etablir la connexion et exécuter des requêtes

Pour accéder à la base de données, vous devrez tout d'abord configurer la connexion. La capture d'écran ci-dessous vous montre la marche à suivre :

- Cliquez sur le sigle  en haut à gauche
- Complétez le formulaire avec les données de connexion et cliquez sur le bouton Enregistrer

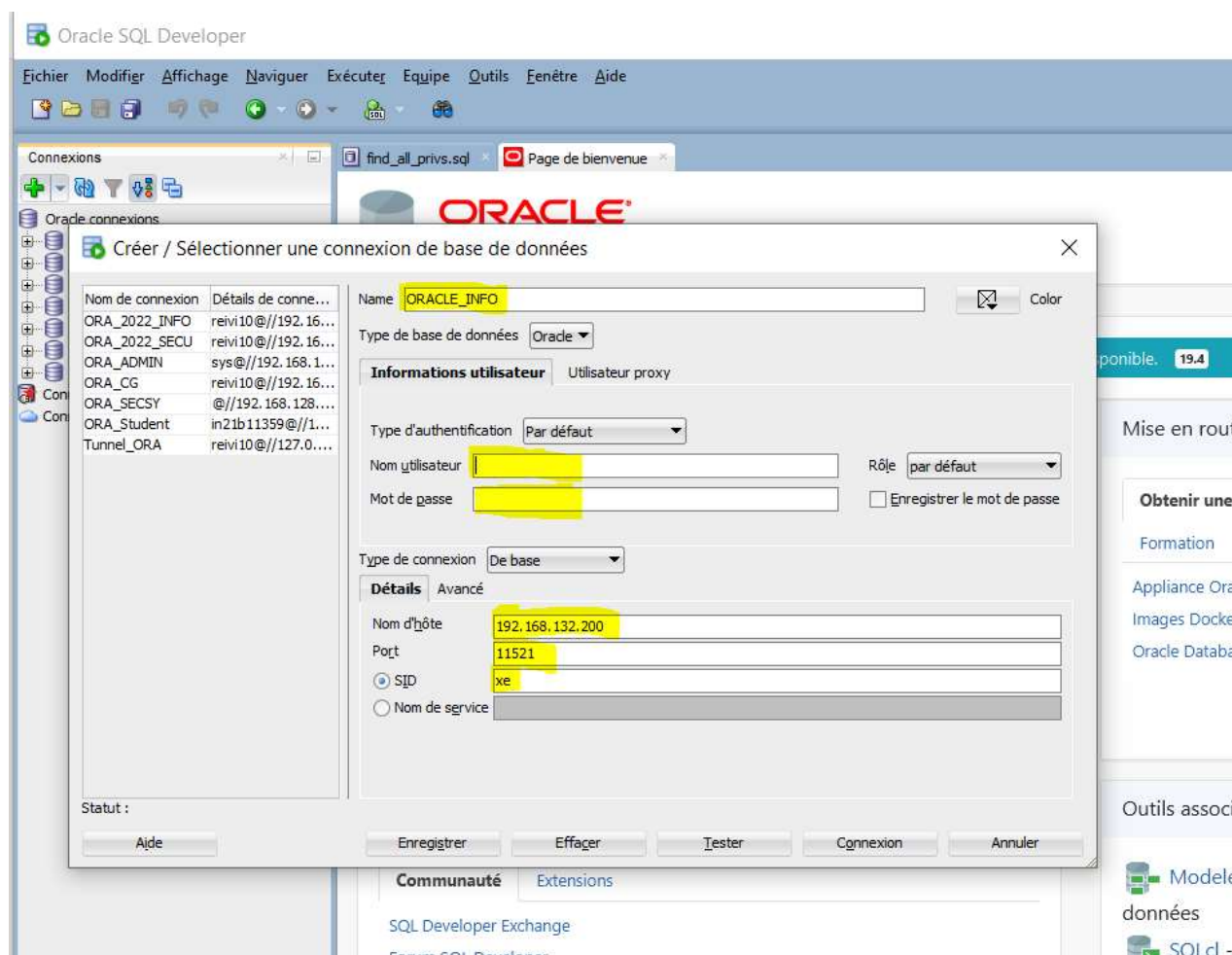



Figure 1 : configuration de la connexion à la BD

Une fois cette manipulation effectuée, vous verrez apparaître votre connexion dans la colonne de gauche. Un 'click-droit' sur celle-ci vous permet de vous connecter (en précisant votre mot de passe).

Vous verrez alors apparaître toute une série d'item dans la colonne de gauche et une nouvelle 'Feuille de calcul SQL' apparaîtra.

Vous pouvez alors entrer des commandes SQL de manière interactive et les exécuter en cliquant sur le bouton 'Exécuter l'instruction'  (raccourci clavier : CTRL – Enter).

Le résultat de votre requête apparaîtra dans la partie basse de votre fenêtre.

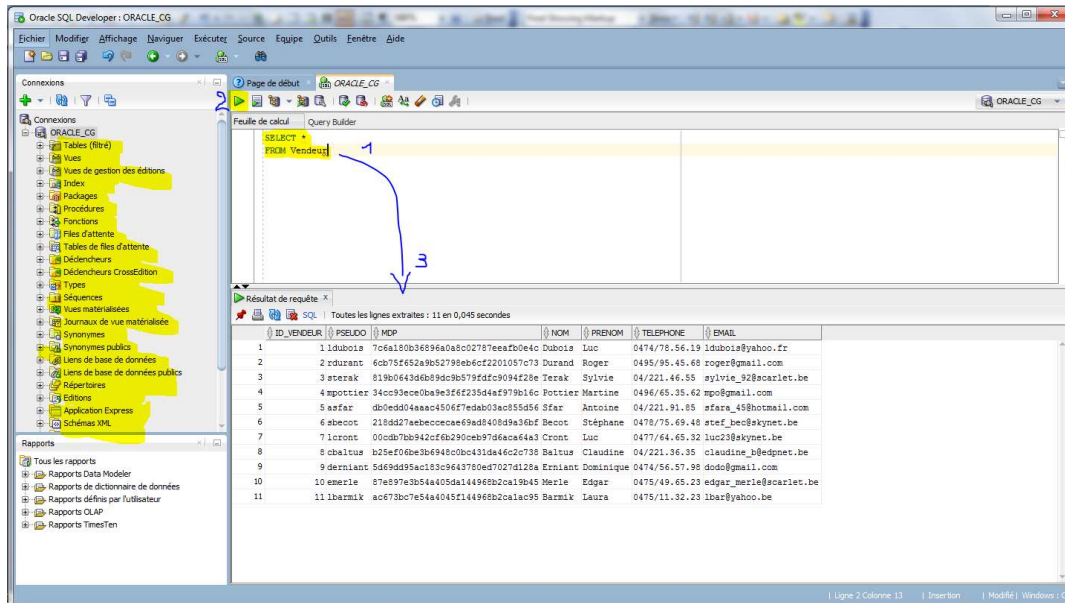



Figure 2 : Exécution d'une requête SQL

3.2 Les scripts

Il est aussi possible de d'exécuter des scripts SQL (un script est un ensemble de requêtes SQL séparées par un point-virgule). Cette option est très utile pour les scripts de création des bases de données qui consistent en une liste d'instructions CREATE TABLE... Pour exécuter un tel script, il faut écrire le script dans la 'Feuille de calcul SQL' (ou faire un copier/coller à partir d'un fichier, ou encore ouvrir le fichier contenant le script via le menu Fichier>Ouvrir) et ensuite cliquer sur le bouton 'Exécuter un script'  (raccourci clavier : F5).

Après exécution, il faut impérativement vérifier le résultat qui est visible dans la partie basse de la fenêtre ('Sortie de script'). Si des erreurs se sont produites, il faut analyser le message d'erreur et, le cas échéant, faire des corrections au niveau du script.

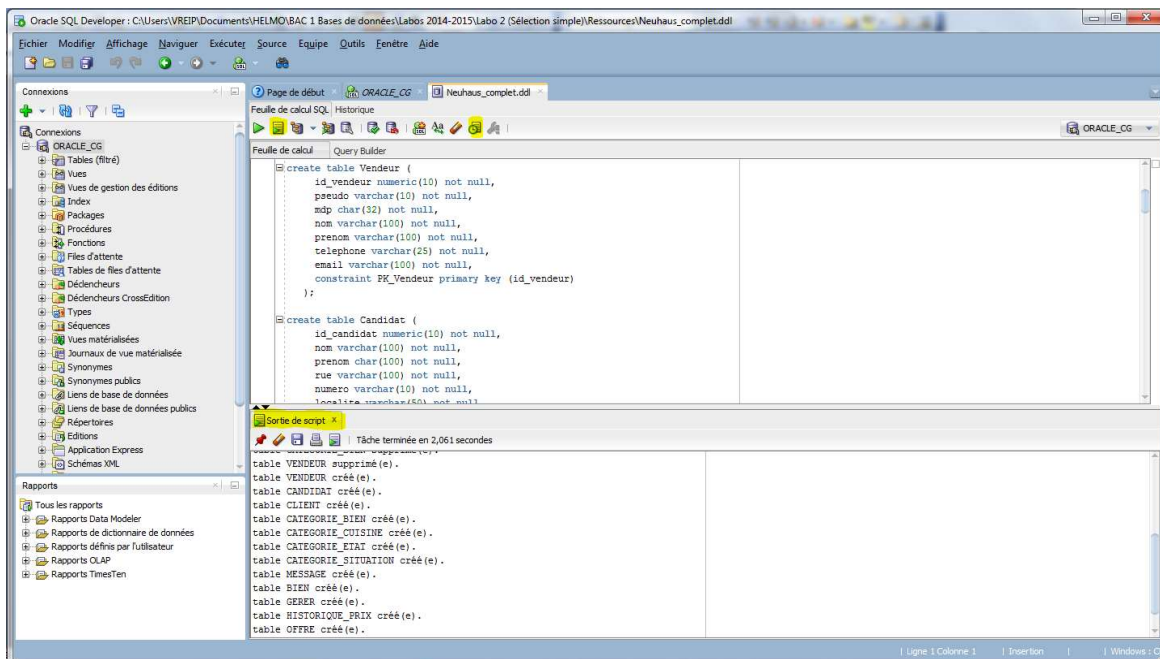



Figure 3 : Exécution d'un script

Par ailleurs, l'historique des requêtes et scripts exécutés est disponible en cliquant sur le bouton 'Historique SQL'  (raccourci clavier : F8).

3.3 Importer des données

Il est fréquent de devoir importer des données dans des tables à partir de fichiers (CSV, XML, ...). Pour importer un fichier de données dans une table, il faut faire un 'clic-droit' sur le nom de cette table (colonne de gauche) ; un menu contextuel apparaît alors et propose une option « Importer des données... ». Vous devez ensuite sélectionner le fichier de données sur votre disque dur. Il faudra alors préciser toute une série de paramètres indiquant la nature du fichier à importer. Les captures d'écran ci-dessous vous montrent la marche à suivre pour des fichiers dont les données sont séparées par de point-virgule (;) comme ceux qui vous seront fournis pour les laboratoires BD.

ETAPE 1 : sélectionnez le format 'Delimited' et précisez que les données sont séparées par des points-virgules. Les données doivent alors être présentées en colonnes.

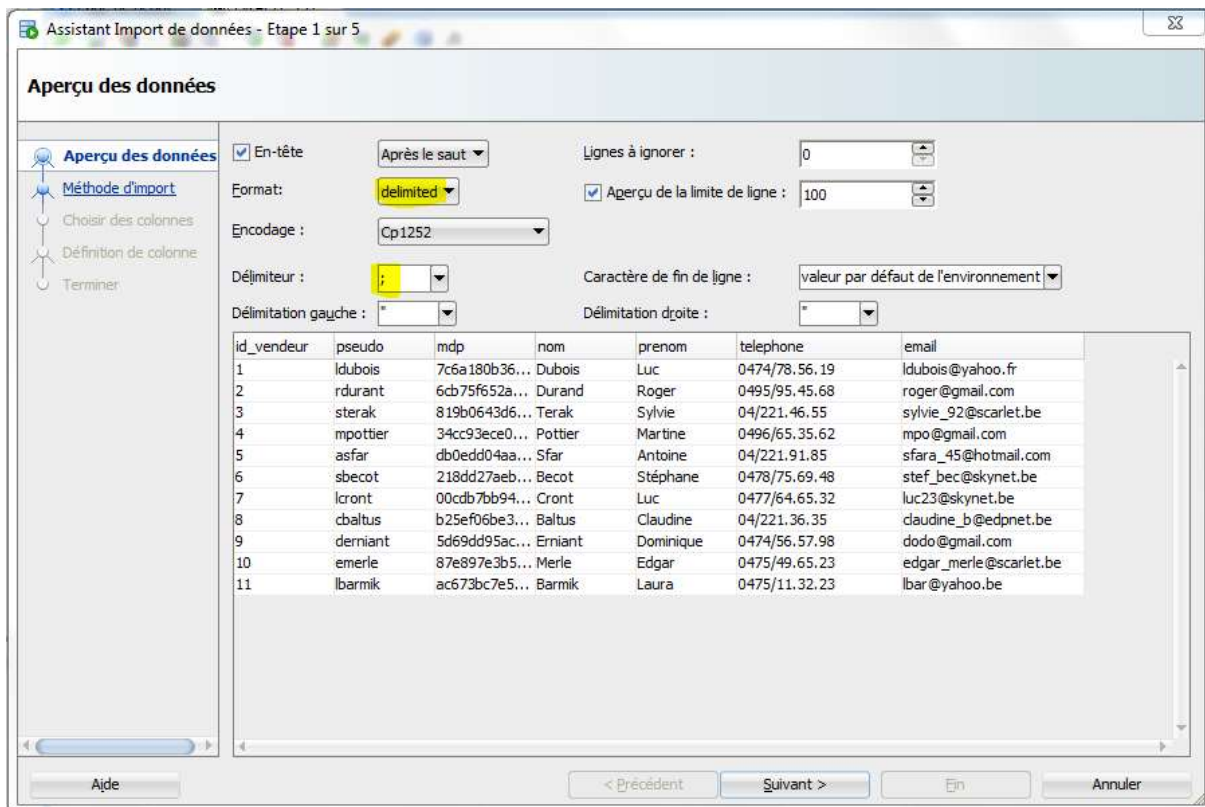


Figure 4 : Import (1) : aperçu des données

ETAPE 2 : la méthode d'import est 'Insérer' (sélectionnée par défaut)

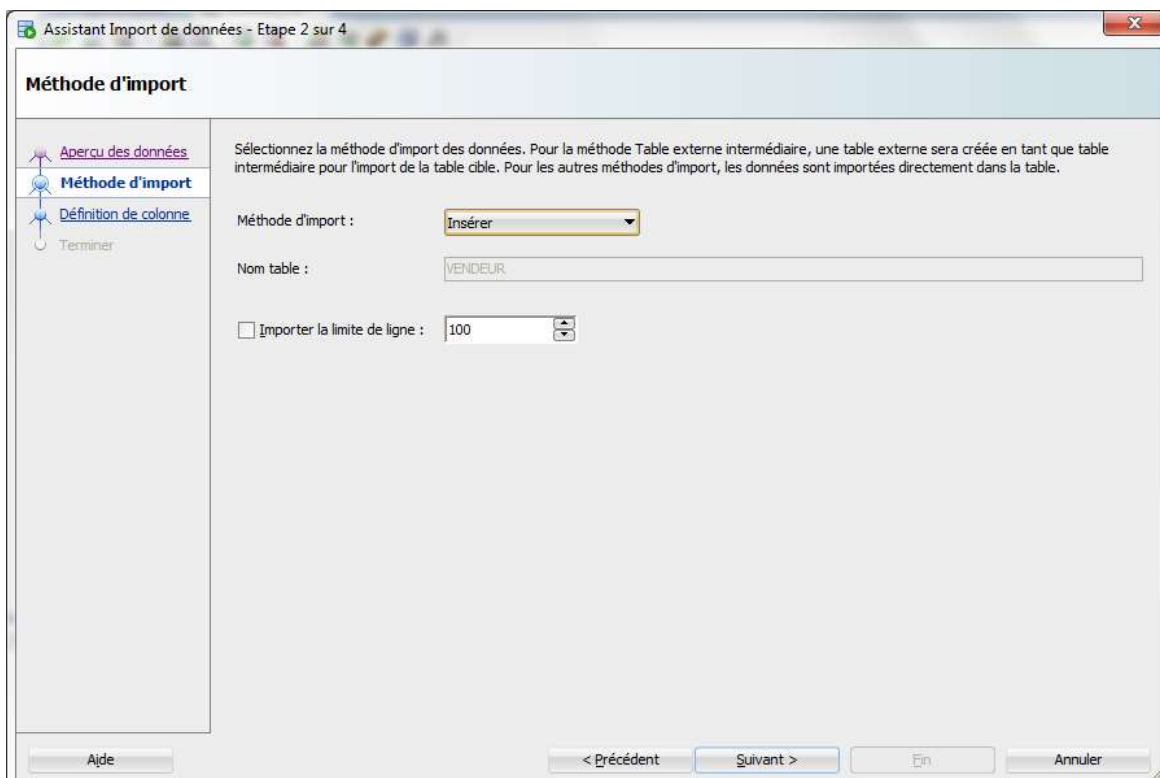


Figure 5 : Import (2) Méthode d'import

ETAPE 3 : on laisse le choix par défaut qui est de sélectionner toutes les colonnes présentes dans le fichier.

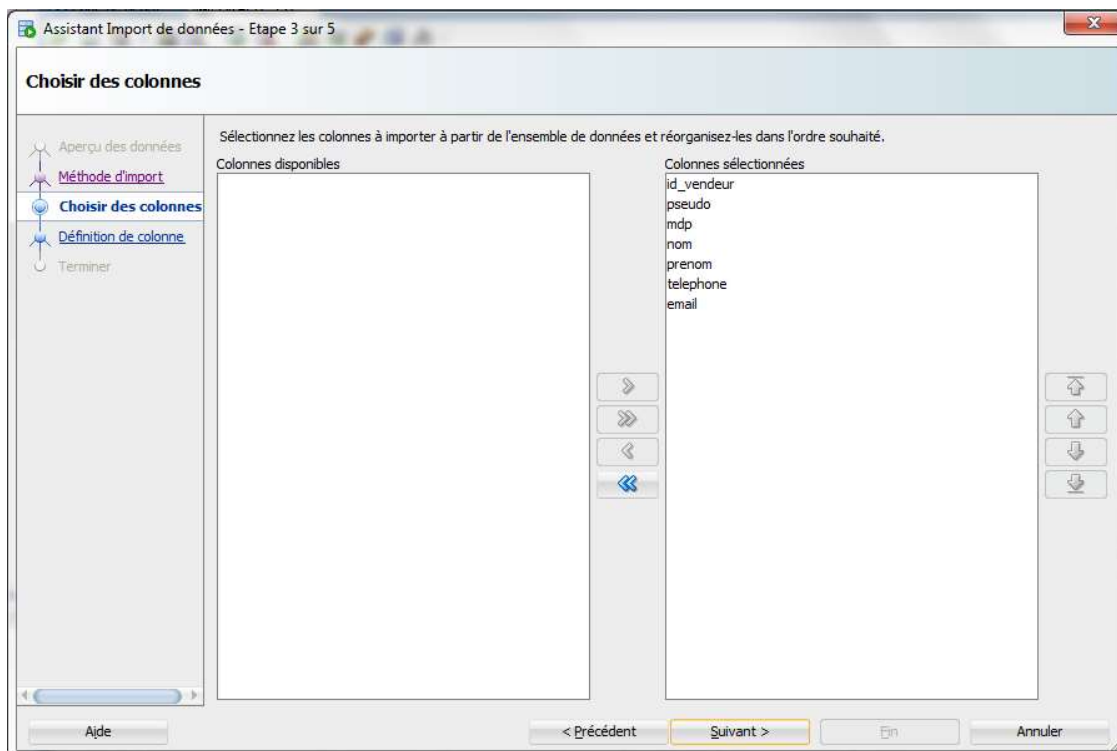


Figure 6 : Import (3) Choix des colonnes à importer

ETAPE 4 : on met en correspondance les données présentes dans le fichier et les différentes colonnes de la table. La correspondance pourra s'opérer sur le nom (car nos fichiers de données ont des entêtes en première ligne qui correspondent aux noms des colonnes. Certaines tables contiennent des données au format 'Date'. Il faut alors explicitement préciser le format qui est 'mm/dd/yy' dans nos fichiers).

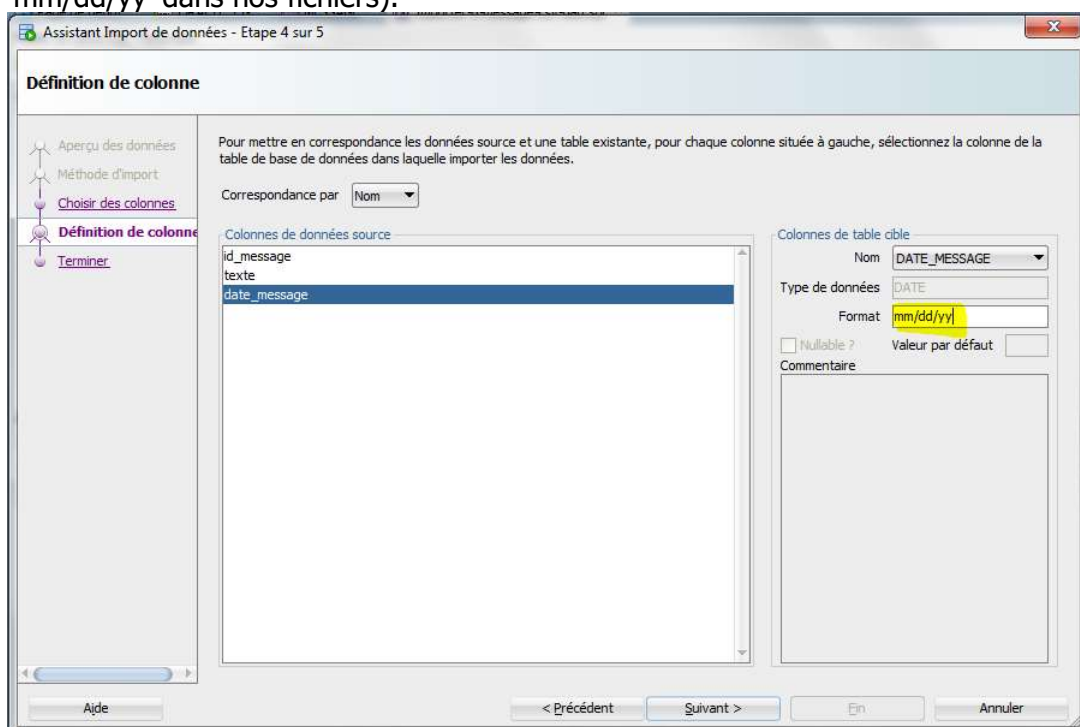


Figure 7 : Import (4) Correspondance données/colonnes

ETAPE 5 : on peut procéder à une vérification (optionnelle) et terminer l'import en cliquant sur le bouton 'Fin'

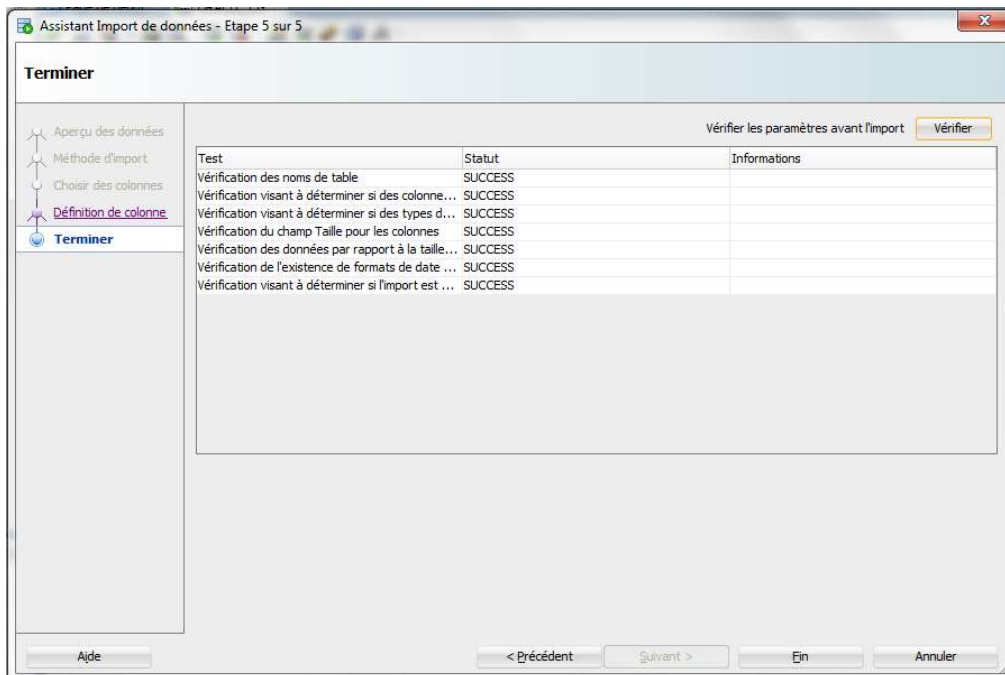


Figure 8 : Import (5) Vérification (optionnelle)

Bibliographie

- [1] C. MAREE et G. LEDANT, *SQL-2 : Initiation, Programmation*, 2^{ème} édition, Armand Colin, 1994, Paris
- [2] P. DELMAL, *SQL2 – SQL3 : application à Oracle*, 3^{ème} édition, De Boeck Université, 2001, Bruxelles
- [3] JL. HAINAUT, *Bases de données : concepts, utilisation et développement*, Dunod, 2009, Paris