



# Rapport de fin de module, Beta<sup>1</sup> SGBD1

Date de Création : 28/10/2013

Etablissement : ISMONTIC de Tanger

Filière : TDI

Niveau : 2<sup>ème</sup> année

Réalisé par

- TDI 2<sup>ème</sup> année G4 et G5

Encadré Par :

- ES-SARRAJ Fouad

---

<sup>1</sup> Attention : cette version du rapport est imprimée être corrigé, il est strictement interdit de le prendre comme un support de formation ou le partager sur internet.

## **Remerciement**

Nous présentons nos sincères remerciements à notre formateur et encadrant Mr ES-SERRAJ Fouad pour ses conseils et ses efforts pendant l'enseignement du module « SGBD1 ».

Nous présentons un grand merci pour toute l'équipe administrative de l'institut ISMONTIC et tous les autres formateurs pour avoir assuré les parties théoriques et pratiques de notre formation. Ainsi que pour leurs soutiens et compréhension tout au long de notre formation.

## Table des matières

Remerciement .....	2
Table des figures .....	6
Table des requêtes .....	7
Introduction .....	10
Partie 1 : Notions de base .....	11
Chapitre 1 : Avant de commencer .....	12
Création de la table des stagiaires .....	12
Insérer des groupes et leurs stagiaires : .....	13
Modifier un stagiaire : .....	14
Supprimer un stagiaire .....	14
Rechercher un stagiaire .....	15
Afficher tous les stagiaires avec le nom de leurs groupes.....	15
Produit cartésien .....	16
Jointure .....	17
Chapitre 2 : Langage de définition de données (LDD) .....	19
Création de la base de données .....	19
Création des tables .....	20
Contraintes d'intégrité : .....	22
Contraintes Unique .....	22
Contraintes Check .....	22
Contraintes Primary Key .....	23
Contraintes Foreign Key .....	23
Supprimer une table : .....	24
Modifier une table .....	24
Ajouter une colonne : .....	24
Modifier une colonne : .....	25
Ajouter une contrainte d'intégrité : .....	26
Supprimer une colonne : .....	28
Supprimer une contrainte : .....	28
Chapitre 3 : Langage de manipulation de données (LMD) .....	29
Insertion d'enregistrement : « Insert » .....	29
Modification d'enregistrement : « Update » .....	30
Suppression d'enregistrement : « Delete » .....	31

Manipulation des données avec le type : « DateTime » .....	31
Chapitre 4 : Interrogation des données .....	35
Extraction des données : « Select » .....	35
Duplicate : « distinct » .....	36
Select avec expression : .....	36
Ordonnancement : « Order By ».....	37
Concaténation : « + » .....	38
Opérateurs intégrés : .....	38
L'opérateur « IN » :.....	38
L'opérateur « LIKE » : .....	38
Statistiques sur les données (Sum, Min, Max, Avg, Count) .....	39
Regroupement : « GROUP BY ».....	39
L'élimination de certains groupes par « HAVING »: .....	40
Chapitre 5 : Sous interrogations .....	41
Sous-interrogations mono-lignes .....	41
Sous-interrogations multi-lignes : (IN, ANY, ALL) .....	41
Utilisation de « IN » : .....	41
Utilisation de « ANY » : .....	42
Utilisation de « ALL » : .....	43
Chapitre 6 : Jointure .....	44
Produit Cartésien avec SQL .....	44
Jointure relationnelle : .....	45
Équijointure : .....	45
InÉquijointure : .....	45
Chapitre 7 : Séquences .....	47
Chapitre 8 : Création des vues .....	48
Simplification : .....	48
Sécurité d'accès : .....	48
Chapitre 9 : Index .....	49
Avant la création d'index : .....	49
Création des index : .....	49
Après la réaction de l'index.....	50
Chapitre 10 : Transaction.....	51
Chapitre 11 : Création des utilisateurs et des rôles .....	52
Création d'une connexion .....	52

Création utilisateur :.....	52
Droit utilisateur : LMD .....	52
Accorder des autorisations : grant .....	52
Retirer des autorisations : revoke :.....	52
Droit d'utilisateur : LDD.....	53
Accorder: grant.....	53
Retirer : revoke.....	53
Partie 2 : Travaux dirigés.....	54
TD : Gestion des Stagiaire.....	55
TD : Gestion Approvisionnement.....	57
TD : Société « Tanger Soft ».....	61
Partie 2 : Travaux Pratiques.....	63
TP : Gestion des Vols.....	63
Conclusion .....	65

## Table des figures

Figure 1 : Ensemble A .....	16
Figure 2 : Ensemble B .....	16
Figure 3 : Produit cartésien de l'ensemble A et B.....	17
Figure 4 : Résultat de requête de jointure entre Stagiaires et Groupes .....	18
Figure 5 : Base de données des exemples utilisés dans le chapitre 2, Gestion des stagiaires .....	19
Figure 6 : Création de la base de données - Gestion des stagiaires.....	19
Figure 7 : Création de la table « Groupes » .....	20
Figure 8 : Création de la table « Stagiaires » .....	21
Figure 9 : Création de la table « Formateurs ».....	22
Figure 10 : Ajouter la colonne "id_groupe" à la table « Stagiaires » .....	25
Figure 11 : Ajouter la contrainte « not null » au colonne Id de la table « Stagiaires » .....	25
Figure 12 : Ajouter la contrainte « not null » au colonne « id » de la table « Groupes » .....	26
Figure 13 : Ajouter la contrainte Check au champ "Ville" de la table « Stagiaires » .....	26
Figure 14 : Ajouter une contrainte Clé primaire à la table « Stagiaires » .....	27
Figure 15 : Ajouter une contrainte Clé primaire à la table « Groupes » .....	27
Figure 16 : Ajouter une contrainte « Foreign key » entre la table « Stagiaires » et « Groupes ».....	27
Figure 17 : Base de données de gestion des stagiaires, version 3 .....	28
Figure 18 : Base de données d'exemples de chapitre 4 .....	35
Figure 19 : Données de la table « Stagiaires », Chapitre 4.....	35
Figure 20 : Données de la table « Groupes », Chapitre 4 .....	35
Figure 21 : Résultat de sous-interrogations ALL .....	43
Figure 22 : Base de données des exemples de chapitre 6 .....	44
Figure 23 : Données de la table « Stagiaires », chapitre 6.....	44
Figure 24 : Données de la table « Mensions » .....	44
Figure 25 : Exécution sous SQL des vues .....	48
Figure 26 : Temps d'exécution de la requête.....	49
Figure 27 : Création d'index .....	50
Figure 28 : Après la réaction de l'index .....	50
Figure 29 : Base de données de gestion des stagiaires, TD .....	55
Figure 30 : Base de données de gestion approvisionnement .....	57
Figure 31 : Base de données de société « Tanger Soft » .....	61
Figure 32 : Base de données de gestion des vols .....	63

## Table des requêtes

Requête 1 : Création de la table « Stagiaires » .....	12
Requête 2 : Création de la table « Groupes ».....	12
Requête 3 : Exemple de requête pour insérer le groupe TDI1 .....	13
Requête 4 : Ajouter le stagiaire « Karim » à la table « Stagiaires ».....	13
Requête 5 : Changement du nom du stagiaire "Ali" à "Moad" .....	14
Requête 6 : Suppression du stagiaire "Mouna" du numéro 1 .....	14
Requête 7 : Affichage de tous les stagiaires. ....	15
Requête 8 : Affichage de tous les stagiaires de groupe numéro 1 .....	15
Requête 9 : Produit entre la table Stagiaires et la table Groupes .....	15
Requête 10 : Création de la base de données « GestionStagiaires » .....	19
Requête 11 : Création de la table « Groupes ».....	20
Requête 12 : Création de la table « Stagiaires ».....	20
Requête 13 : Création de la table « Formateurs » .....	21
Requête 14 : Création de la table « Formateurs » avec la contrainte « Unique » .....	22
Requête 15 : Création de la table « Formateurs » avec la contrainte « Check ».....	22
Requête 16 : Création de la table « Groupe » avec la contrainte « Primary Key ».....	23
Requête 17 : Création de la table « Stagiaires » avec la contrainte « Foreign Key » .....	23
Requête 18 : Création de la table « Stagiaires » avec plusieurs contraintes d'intégrités. ....	23
Requête 19 : Suppression de la table « formateurs » .....	24
Requête 20 : Suppression de la table « groupes ».....	24
Requête 21 : Ajoute de la colonne ville à la table stagiaire en utilisation de « Alter table » .....	24
Requête 22 : Insertion de la colonne « id_groupe » à la table stagiaire .....	24
Requête 23 : Modification de la colonne ville .....	25
Requête 24 : Ajouter la contrainte « not null » au colonne « id » de la table « stagiaires ».....	25
Requête 25 : Ajouter la contrainte « not null » au colonne « id » de la table « groupes ».....	25
Requête 26 : Ajouter le groupe TD4 .....	29
Requête 27 : Ajouter le stagiaire « Ali » au groupe TDI4.....	29
Requête 28 : Ajoute de la colonne « DateNaissance » à la table Stagiaires .....	31
Requête 29 : Mise à jour de la date de naissance avec date et heure : minute : second.....	32
Requête 30 : Mise à jour de la date de naissance avec date et heure : minute .....	32
Requête 31 : Mise à jour de la date de naissance avec seulement la date .....	32

Requête 32 : Recherche par date .....	32
Requête 33 : Mise à jour la date de naissance par date d'aujourd'hui .....	33
Requête 34 : Recherche par jour .....	33
Requête 35 : Recherche par Mois .....	33
Requête 36 : Recherche par Année .....	33
Requête 37 : afficher la liste de tous les stagiaires.....	35
Requête 38 : afficher les noms de tous les stagiaires .....	35
Requête 39 : Utilisation de « distinct » pour éliminer les informations redoublantes. ....	36
Requête 40 : Afficher la note augmentée par 10 %.....	36
Requête 41 : Affichage de tous stagiaires avec ordre par défaut, croissant .....	37
Requête 42 : Affichage de tous stagiaires avec ordre croissant des notes .....	37
Requête 43 : Affichage de tous stagiaires avec ordre décroissante des notes.....	37
Requête 44 : Affichage de Nom Complet (Nom + Prénom) de tous les stagiaires. ....	38
Requête 45 : Afficher les stagiaires dont la ville = Tanger ou Casa .....	38
Requête 46 : Utilisation de l'opérateur «like» pour chercher des stagiaires .....	38
Requête 47 : Afficher la note moyenne de tous les stagiaires .....	39
Requête 48 : Afficher la note maximale de tous les stagiaires .....	39
Requête 49 : Afficher les notes maximums de toutes les villes .....	39
Requête 50 : Afficher les notes maximums supérieures à 10 de toutes les villes en utilisation de « Having ».....	40
Requête 51 : Afficher la liste des employés ayant le salaire maximal. ....	41
Requête 52 : Utilisation sous-interrogation avec l'opérateur « IN ».....	42
Requête 53 : Utilisation sous-interrogation avec l'opérateur « Any » .....	42
Requête 54 : Utilisation sous-interrogation avec l'opérateur « All ».....	43
Requête 55 : Exemple de « équijointure ».....	45
Requête 56 : Exemple de « In Equijointure » .....	46
Requête 57 : Création d'une table avec Id « auto-incrémenté », Syntax de « Identity » .....	47
Requête 58 : Exemple de création d'un view .....	48
Requête 59 : Exemple de création d'un index .....	49
Requête 60 : Création d'une connexion : login1 .....	52
Requête 61 : Création de l'utilisateur « user1 » de login « login1 ».....	52
Requête 62 : autoriser l'utilisateur « user1 » à faire de select sur la table stagiaire .....	52
Requête 63 : supprimer l'autorisation de l'utilisateur « user1 » à faire de delete sur la table stagiaire.....	52
Requête 64 : autoriser l'utilisateur « user1 » à créer les view. ....	53



## Rapport de fin de module SGBD1, ISMONTIC, Tanger

Requête 65 : retirer l'autorisation à création des views de l'utilisateur « user1 ».	53
Requête 66 : retirer tous les autorisations	53

## Introduction

Ce module de compétence permet au stagiaire de manipuler une base de données en utilisant le langage SQL.

SQL (Structured Query Language), en français langage de requête structurée, est un langage informatique normalisé servant à exploiter des bases de données relationnelles dans le SGBD. Par exemple, la partie langage de manipulation des données (LMD) de SQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données relationnelles.

Le SGBD (Système de gestion des bases de données) est un logiciel destiné à stocker et à partager les informations dans une base de données, en garantissant la qualité, la pérennité et la confidentialité de ces informations, tout en cachant la complexité des opérations. Dans ce module, nous utilisons le SGBD « Microsoft SQL Server 2008 » développé et commercialisé par la société Microsoft.

Le présent document est un rapport de fin de module. Il regroupe les rapports de TD, et TP ainsi qu'un résumé détaillé des notions de base de langage SQL.

Dans la première partie, intitulée « Notion de base », nous présentons un rappel détaillé des notions de base de langage SQL en se basant sur le support de cours de notre formateur « ES-SARRAJ Fouad ». Cette partie comprend 11 chapitres. Le premier chapitre, intitulé « Avant de commencer », représente notre première confrontation avec le langage SQL. L'objectif est d'apprendre les connaissances initiales pour se plonger dans les 65 requêtes de ce rapport. Les autres chapitres traitent les différentes parties du langage SQL : langage de définition de données (LDD), langage de manipulation des données (LMD) et le langage de contrôle de données (LCD).

La deuxième partie, intitulée « Travaux dirigés », présente un rapport des différents travaux dirigés que nous avons réalisés. Il comprend trois TD. Dans chaque un nous montrons le MPD (Modèle physique de données), le travail réalisé et une proposition de solution.

La dernière partie, travaux pratiques, présente les différents TP que nous avons réalisés. Dans chaque TP nous montrons le MPD, le travail réalisé, une proposition de solution et un rapport de différentes erreurs et exceptions rencontrées.

Nous terminons notre rapport par une conclusion dans laquelle nous présentons les compétences que nous avons développées durant ce module.

## Partie 1 : Notions de base

L'objectif de cette partie est de résumer les notions de base du langage SQL. Nous résumons les 11 chapitres que nous avons réalisés dans ce module. Les chapitres sont organisés selon la composition du langage SQL.

Ce langage est composé de trois parties majeures et distinctes :

- LMD (langage de manipulation des données) : il permet de consulter ou de modifier le contenu de la base de données.
- LDD (langage de définition des données) : il permet de modifier la structure de la base de données.
- LCD (langage de contrôle des données) : il permet de gérer les privilèges, ou les différents droits des utilisateurs sur la base de données.

Voici les principaux ordres employés :

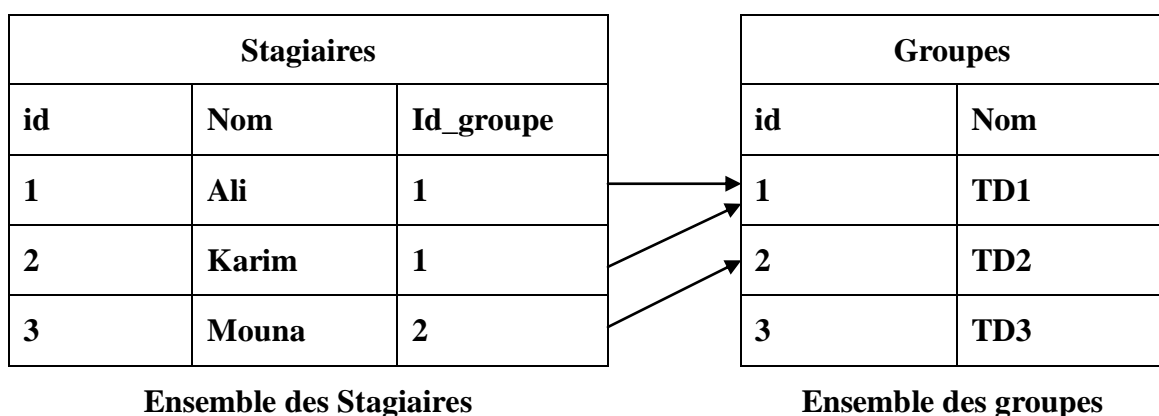
LMD	LDD	LCD
SELECT	CREATE	GRANT
INSERT	ALTER	REVOKE
DELETE	RENAME	
UPDATE	DROP	

## Chapitre 1 : Avant de commencer

L'objectif de ce chapitre est d'apprendre les connaissances initiales du langage SQL pour simplifier la rentrée dans le détail des différentes parties du langage SQL.

Nous allons travailler sur un ensemble des stagiaires et leurs groupes. Par exemple, nous allons apprendre comment déclarer l'existence des tables, ajouter, supprimer ou modifier un stagiaire et rechercher toutes les informations d'un stagiaire qui sont réparties sur plusieurs ensembles.

Dans chaque chapitre, nous travaillons sur un exemple unique pour simplifier la réalisation pratique de différents exemples. Dans ce chapitre, nous prenons deux ensembles : l'ensemble des stagiaires et l'ensemble de leurs groupes.



### Création de la table des stagiaires

Pour déclarer l'existence de l'ensemble des stagiaires nous devons déclarer les tables stagiaires avec la requête « Create table ». Il permet de créer une table en déclarant ses attributs :

*Requête 1 : Création de la table « Stagiaires »*

```
Create table Stagiaires (  
  Id Number,  
  Nom Varchar,  
  Ville Varchar,  
  Id_groupe Number  
);
```

Par même raisonnement nous créons la table « Groupes » :

*Requête 2 : Création de la table « Groupes »*

```
Create table Groupes (  
  Id Number,  
  Nom Varchar  
);
```

Résultat d'exécution des deux tables : deux tables vide

Stagiaires		
id	Nom	Id_groupe

Ensemble des Stagiaires

Groupes	
id	Nom

Ensemble des groupes

### Insérer des groupes et leurs stagiaires :

Pour ajouter un stagiaire à notre table nous utilisons la requête « Insert ». Cette requête permet d'insérer une ligne ou un enregistrement dans la table.

Mais avant d'insérer un stagiaire nous devons d'abord insérer les groupes.

*Requête 3 : Exemple de requête pour insérer le groupe TDI1*

```
Insert into Groupes values (1, 'TDI1') ;
```

Maintenant, nous pouvons ajouter le stagiaire « Karim » au groupe TDI1 avec la requête suivante :

*Requête 4 : Ajouter le stagiaire « Karim » à la table « Stagiaires »*

```
Insert into Stagiaires values (2, 'Karim',1) ;
```

Résultat d'exécution des deux tables :

Stagiaires		
id	Nom	Id_groupe
2	Karim	1

Ensemble des Stagiaires

Groupes	
id	Nom
1	TD1

Ensemble des groupes

### Modifier un stagiaire :

La requête « UPDATE », permet de mettre à jour un ou plusieurs enregistrements. Par exemple pour changer le nom de stagiaire numéro 1 à « Mouna » nous utilisons la requête suivante :

*Requête 5 : Changement du nom du stagiaire "Ali" à "Moad"*

```
Update Stagiaires set Nom= 'Moad' where id=1 ;
```

Résultat d'exécution des deux tables :

Stagiaires			Groupes	
id	Nom	Id_groupe	id	Nom
1	<u>Moad</u>	1	1	TD1
2	Karim	1	2	TD2
3	Mouna	2	3	TD3

Ensemble des groupes

Ensemble des groupes

### Supprimer un stagiaire

La requête « DELETE » permet de supprimer des enregistrements. Pour supprimer le stagiaire numéro 1 nous devons utiliser la requête suivant :

*Requête 6 : Suppression du stagiaire "Mouna" du numéro 1*

```
Delete from Stagiaires where id=1;
```

Résultat d'exécution des deux tables :

Stagiaires			Groupes	
id	Nom	Id_groupe	id	Nom
2	Karim	1	1	TD1
3	Mouna	2	2	TD2
			3	TD3

Ensemble des groupes

Ensemble des groupes

## Rechercher un stagiaire

La requête « SELECT » permet de chercher des données d'une ou plusieurs tables.

Par exemple, pour chercher la liste de tous les stagiaires, nous utilisons la requête « Select » sur la table stagiaire. Le symbole « \* » désigne que nous voudrions afficher tous les informations de la table stagiaire.

*Requête 7 : Affichage de tous les stagiaires.*

```
Select * from Stagiaires;
```

Maintenant, nous voudrions afficher que les stagiaires qui appartiennent au groupe numéro 1. Pour cela nous devons ajouter la condition « id\_groupe = 1 » avec la clause « Where ».

*Requête 8 : Affichage de tous les stagiaires de groupe numéro 1*

```
Select Nom from Stagiaires where Id_groupe = 1;
```

## Afficher tous les stagiaires avec le nom de leurs groupes.

Pour afficher par exemple la liste des stagiaires avec les noms de leurs groupes, nous devons utiliser la table stagiaire et la table groupe en même temps dans la clause « Form ».

*Requête 9 : Produit entre la table Stagiaires et la table Groupes*

```
Select * FROM Stagiaire, Groupe;
```

Résultat d'exécution :

id	Nom	Id_groupe	id	Nom
2	Karim	1	3	TD2
2	Karim	1	2	TD3
2	Karim	1	1	TD1
3	Mouna	2	1	TD1
3	Mouna	2	2	TD2
3	Mouna	2	3	TD3

Mais dans la logique du modèle relationnelle, cette requête ne donne pas la bonne solution. Il nous donne un tableau avec deux types d'information, le premier type concerne la solution cherchée,

c'est-à-dire la liste des stagiaires avec les noms de leurs groupes. Le deuxième type concerne des informations incorrectes. La question, ici, est d'où viennent ces informations incorrectes ?

Pour répondre à cette question, nous allons rappeler d'abord la notion de produit cartésien entre deux ensembles. Ensuite nous allons appliquer cette notion sur le produit entre la table « Stagiaires » et la table « Groupes ».

### Produit cartésien

Le but du produit cartésien est de croiser les données d'un ou plusieurs ensembles, de manière à obtenir toutes les combinaisons possibles.

#### Exemple 1 : Produit cartésien entre deux ensembles A et B

Soit deux ensembles A et B. l'ensemble A comprend quatre bulles (trois rouges et un bleu) (voir Figure 1 : Ensemble A).

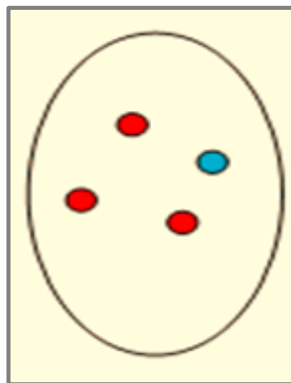


Figure 1 : Ensemble A

L'ensemble B comprend trois triangles (deux bleus et l'autre rouge) (voir Figure 2 : Ensemble B)

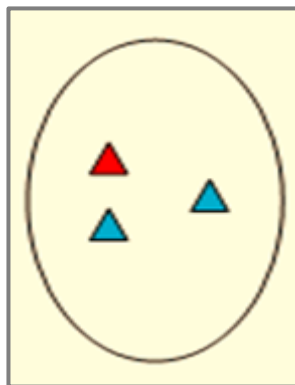


Figure 2 : Ensemble B

Le produit cartésien de l'ensemble A et B donne un ensemble de douze éléments (voir Figure 3 : Produit cartésien de l'ensemble A et B).



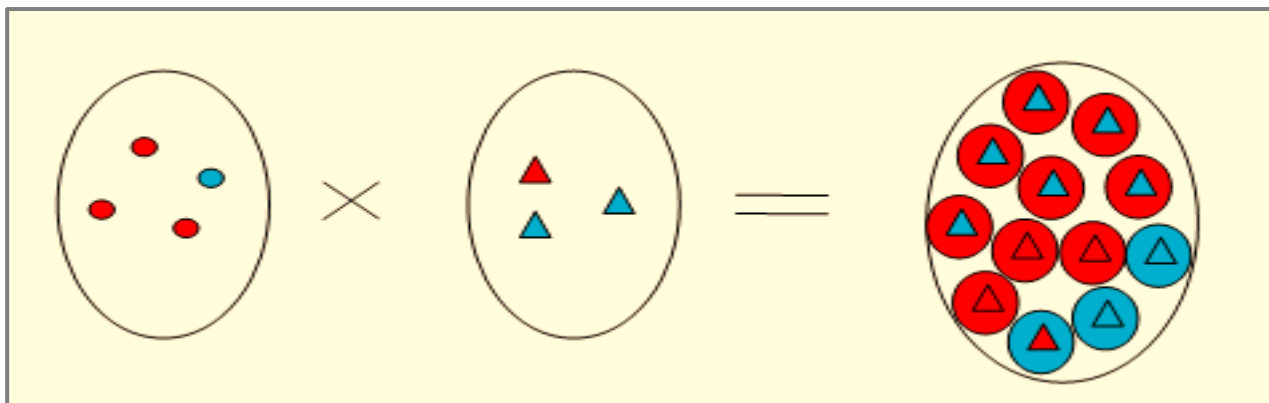


Figure 3 : Produit cartésien de l'ensemble A et B

### Remarque

Si nous voulons garder dans le résultat de produit que les éléments qui ont une couleur homogène (deux éléments de même couleur). Nous aurons seulement **deux bleus** et **trois rouges**.

### Jointure

Pour chercher nos informations dans la base de données nous devons parfois d'utiliser plusieurs tables. Parce qu'au départ dans la phase de la conception nous avons groupé les données dans des entités suivant le modèle relationnel.

Pour trouver les listes des stagiaires et leurs groupes nous devons chercher l'information dans les : Stagiaires et Groupes.

Dans SQL nous utilise la requête SQL suivante :

```
Select * FROM Stagiaires , Groupes ;
```

La clause « From », ici, réalise le produit cartésien de l'ensemble A (Stagiaires) et l'ensemble B (Groupes).

La **jointure** est aussi un **produit cartésien**. C'est une opération permettant de combiner des informations venant de plusieurs tables.

### Exemple 2 : Exemple de jointure entre la table Stagiaires et la Tables Groupes

La jointure est formulée par la requête suivante :

```
Select * FROM Stagiaires , Groupes ;
```

Le résultat de cette requête est aussi un tableau avec 5 colonnes et 6 lignes ( voir Figure 4 : Résultat de requête de jointure).

id	Nom	Id_groupe	id	Nom
2	Karim	1	3	TD2
2	Karim	1	2	TD3

2	Karim	1	1	TD1
3	Mouna	2	1	TD1
3	Mouna	2	2	TD2
3	Mouna	2	3	TD3

Figure 4 : Résultat de requête de jointure entre Stagiaires et Groupes

### Remarque

La multiplication des 2 tableaux nous donne toutes les combinaisons possibles mais sauf les 3 résultats en bleu qui sont corrects d'où vient la nécessité de **la condition de jointure**.

Si vous remarquez bien, la condition qui vérifie les lignes bleues est : **id\_groupe de la table stagiaire = id groupe de la table groupe**

Dans SQL, pour ajouter une condition de jointure nous utilisons la clause « Where » :

```
Select *FROM Stagiaire, Groupe where Stagiaire.id_groupe = Groupe.id ;
```

Autre façon d'écrire cette requête avec les **Alias** :

```
Select *from Stagiaire s,Groupe g where s.id_groupe=g.id ;
```

Le résultat de cette requête est comme suivant, qui est maintenant un résultat correct.

id	Nom	Id_groupe	id	Nom
2	Karim	1	1	TD1
3	Mouna	2	2	TD2

## Chapitre 2 : Langage de définition de données (LDD)

Nous allons aborder dans ce chapitre les ordres du langage de définition de données. Nous allons commencer par l'ordre « Create DataBase ».

Pour les exemples, nous allons utiliser la base de données « Gestion des Stagiaires ».



Figure 5 : Base de données des exemples utilisés dans le chapitre 2, Gestion des stagiaires

### Création de la base de données

Pour créer une base de données nous utilisons la requête « Create DataBase ».

*Requête 10 : Création de la base de données « GestionStagiaires »*

```
CREATE DATABASE GestionStagiaires;
```

Résultat d'exécution sous SQL Server 2008 :

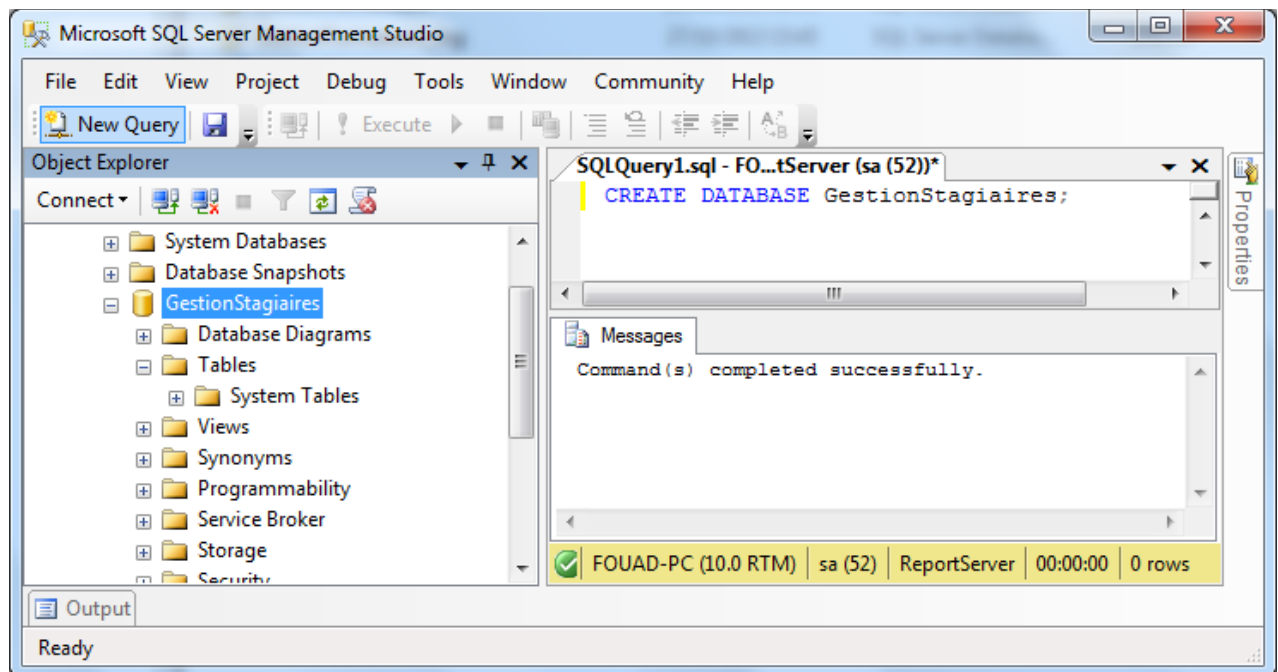


Figure 6 : Création de la base de données - Gestion des stagiaires

Remarque

Pour créer une base de données, il faut être connecté en tant qu'administrateur système, ou avoir la permission d'utiliser CREATE DATABASE, et être dans la base de données système master.

#### Quelques messages d'erreur :

Msg 5170, Level 16, State 1, Line 1 Cannot create file 'C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\GestionStagiaires.mdf' because it already exists. Change the file path or the file name, and retry the operation.

## Création des tables

Pour créer une table nous utilisons la requête « Create Table ».

#### Requête 11 : Création de la table « Groupes »

```
create table Groupes(  
id int,  
nom varchar(50)  
);
```

#### Résultat d'exécution sous SQL Server 2008 :

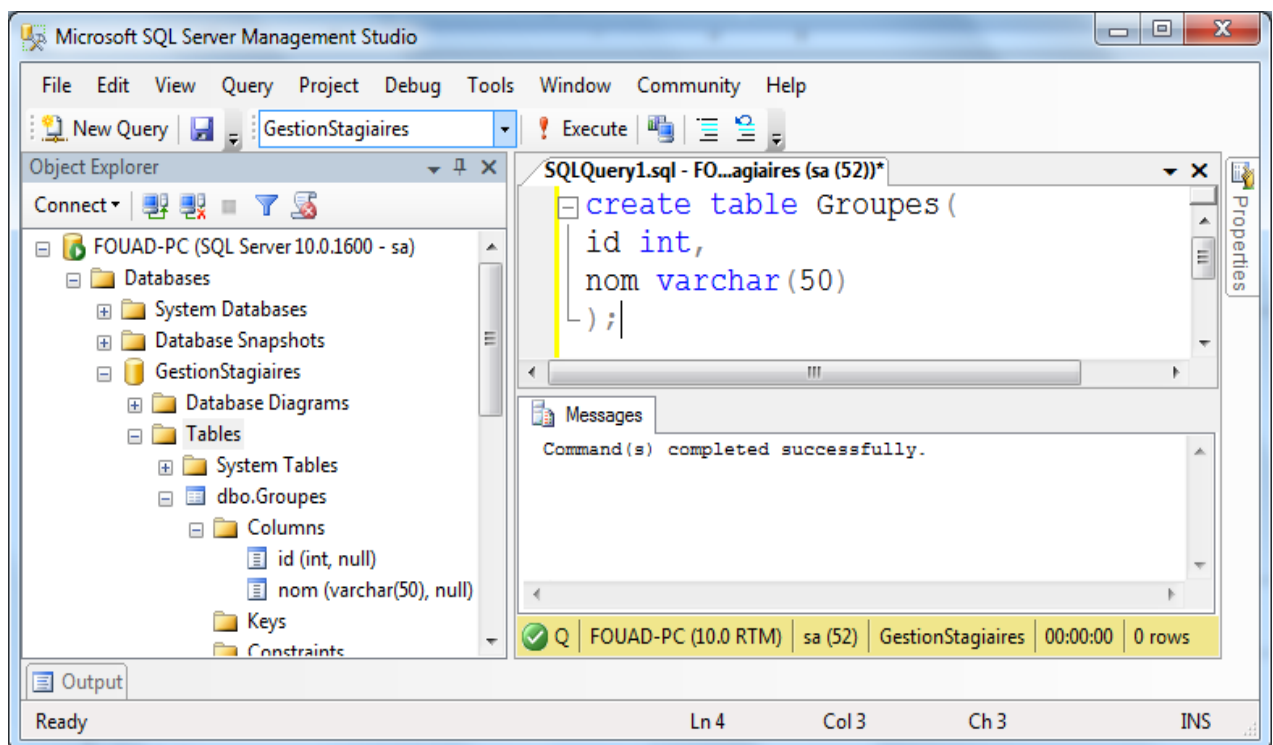


Figure 7 : Création de la table « Groupes »

#### Requête 12 : Création de la table « Stagiaires »

```
create table Stagiaires(  
id int ,  
nom varchar(50) not null,  
ville varchar(50)  
);
```

### Résultat d'exécution sous SQL Server 2008

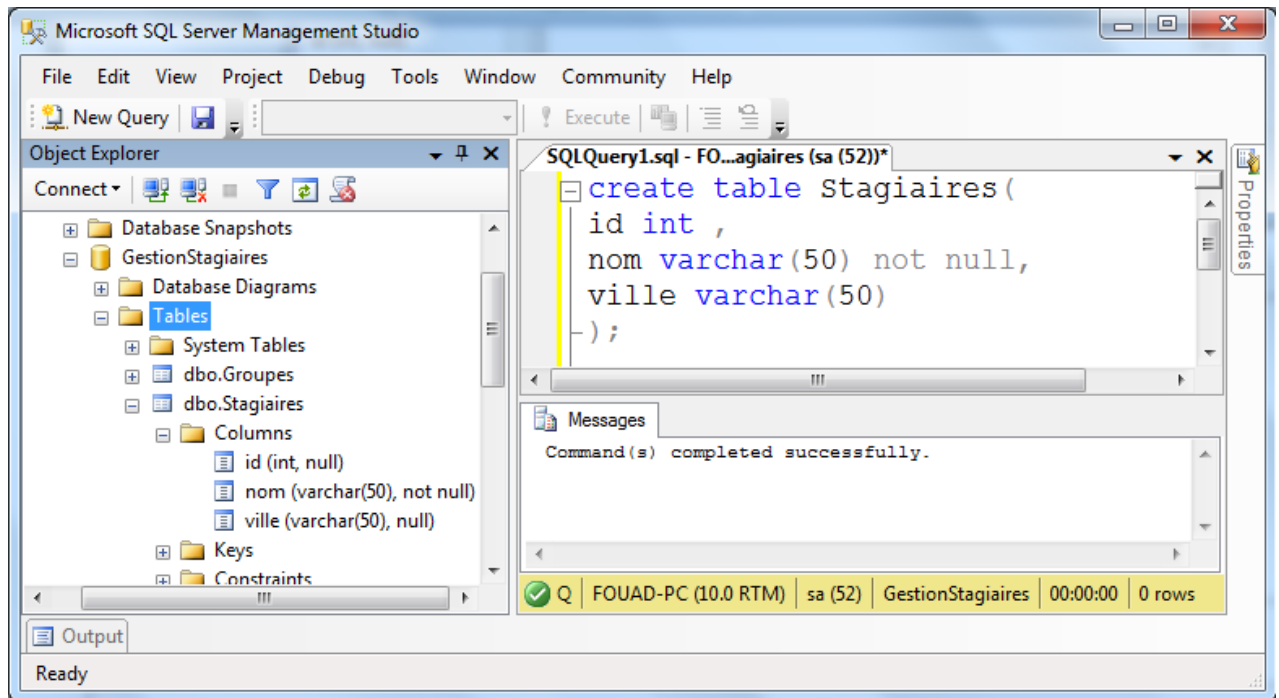


Figure 8 : Création de la table « Stagiaires »

### Requête 13 : Création de la table « Formateurs »

```
create table Formateurs(  
id int,  
nom varchar(50),  
ville varchar(50) default 'tanger'  
);
```

### Résultat d'exécution sous SQL Server 2008

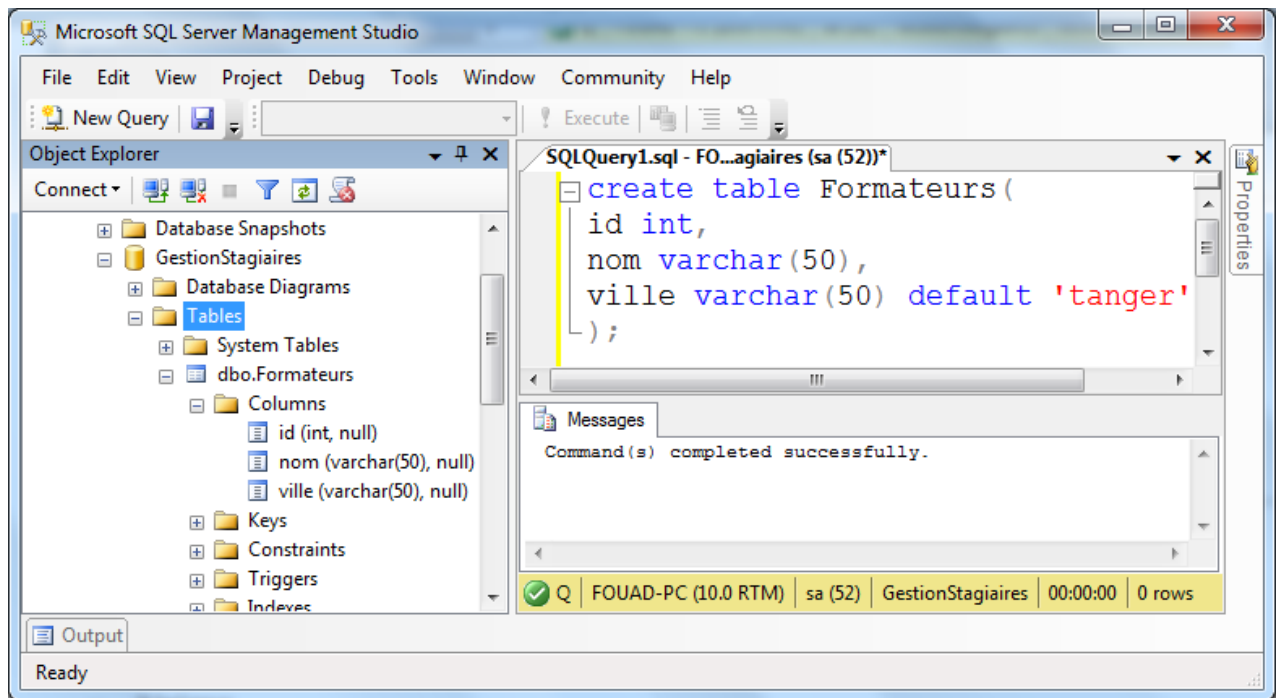


Figure 9 : Création de la table « Formateurs »

## Contraintes d'intégrité :

Il y a quatre types des contraintes d'intégrité :

- Unique.
- Check.
- Primary Key.
- Foreign Key.

### Contraintes Unique

La contrainte **Unique** : impose une valeur distincte au niveau de la table.

*Requête 14 : Création de la table « Formateurs » avec la contrainte « Unique »*

```
Create table Formateurs(  
id int,  
nom varchar(50),  
constraint un_nom Unique(nom)  
);
```

### Contraintes Check

La contrainte **Check** : impose un domaine de valeurs ou une condition entre les colonnes.

*Requête 15 : Création de la table « Formateurs » avec la contrainte « Check »*

```
Create table Formateurs(  
id int,  
nom varchar(50),  
Constraint nn_nom Check(nom is not null),  
);
```

### Contraintes Primary Key

La contrainte « Primary Key » déclare la clé primaire de la table, les colonnes ne peuvent être nul ni identique.

*Requête 16 : Création de la table « Groupe » avec la contrainte « Primary Key »*

```
Create table Groupes(  
id int,  
nom varchar(50),  
Constraint pk_groupe Primary key(id)  
);
```

### Contraintes Foreign Key

La contrainte « Foreign Key » déclare une clé étrangère entre une table enfant et une table père.

*Requête 17 : Création de la table « Stagiaires » avec la contrainte « Foreign Key »*

```
Create table Stagiaires(  
id int,  
nom varchar(50),  
note float,  
filiere varchar(50),  
id_groupe int,  
Constraint fk_stagiaire_groupe Foreign key (id_groupe) References Groupe(id)  
);
```

### Remarque :

Tous les 4 constraint peuvent être regroupés dans une même requête SQL

*Requête 18 : Création de la table « Stagiaires » avec plusieurs contraintes d'intégrités.*

```
Create table Stagiaires(  
id int,  
nom varchar(50),  
note float,  
filiere varchar(50),  
id_groupe int,  
Constraint n_note Check(note between 0 and 20),  
Check(filiere='tdi'or filiere='tri'),  
Constraint pk_stagiaire Primary key(id),  
Constraint fk_stagiaire_groupe Foreign key(id_groupe) References Groupes(id)  
);
```

### Supprimer une table :

*Requête 19 : Suppression de la table « formateurs »*

```
Drop table formateurs ;
```

*Requête 20 : Suppression de la table « groupes »*

```
Drop table Groupes ;
```

On ne peut pas supprimer cette table car elle est référenciée par une contrainte foreign key dans la table « Stagiaires ».

### Modifier une table

#### Ajouter une colonne :

*Requête 21 : Ajoute de la colonne ville à la table stagiaire en utilisation de « Alter table »*

```
Alter table Stagiaire Add ville varchar(50);
```

*Requête 22 : Insertion de la colonne « id\_groupe » à la table stagiaire*

```
Alter table Stagiaires Add id_groupe int;
```

Résultat d'exécution sous SQL Server 2008



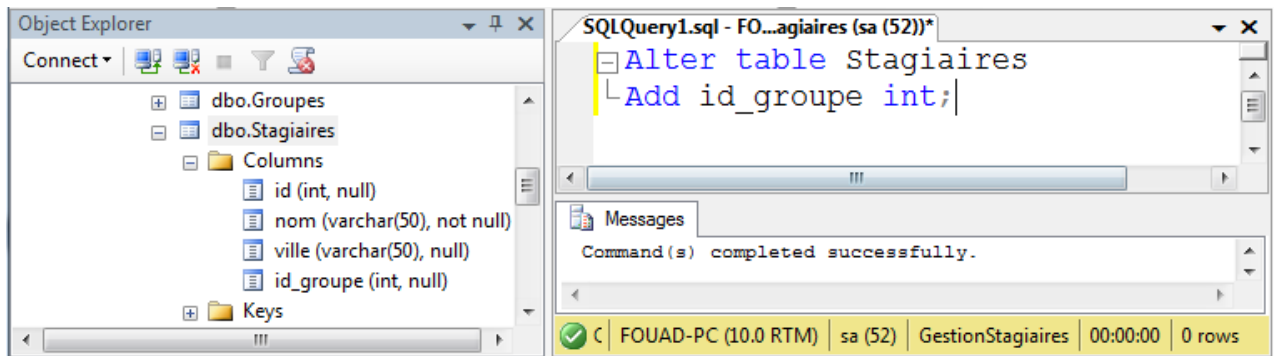


Figure 10 : Ajouter la colonne "id\_groupe" à la table « Stagiaires »

### Modifier une colonne :

Nous changeons dans cette requête le nombre des caractères et nous ajoutons la contrainte « not null ».

*Requête 23 : Modification de la colonne ville*

```
Alter table Stagiaires Alter column ville varchar(100) not null;
```

*Requête 24 : Ajouter la contrainte « not null » au colonne « id » de la table « stagiaires »*

```
Alter table Stagiaires Alter column id int not null;
```

Résultat d'exécution sous SQL Server 2008

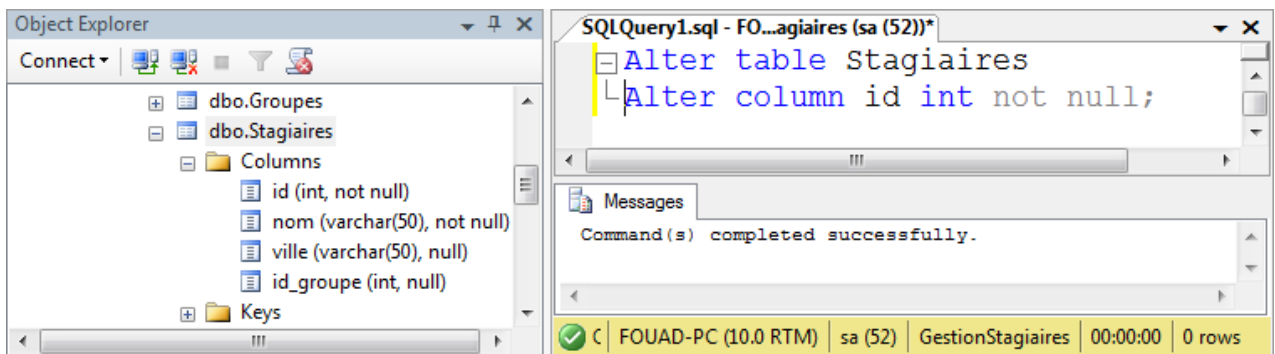


Figure 11 : Ajouter la contrainte « not null » au colonne Id de la table « Stagiaires »

*Requête 25 : Ajouter la contrainte « not null » au colonne « id » de la table « groupes »*

```
Alter table Stagiaires Alter column id int not null;
```

Résultat d'exécution sous SQL Server 2008

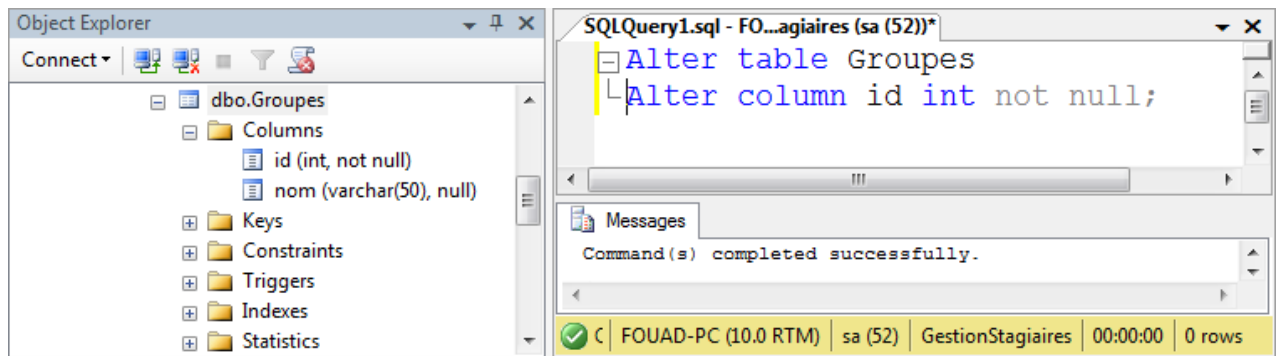


Figure 12 : Ajouter la contrainte « not null » au colonne « id » de la table « Groupe »

### Ajouter une contrainte d'intégrité :

Exemple : Ajouter une contrainte « Check » dans la table « Stagiaires »

```
Alter table Stagiaires Add Constraint cc_ville Check(ville='tanger' or ville='rabat');
```

### Résultat d'exécution sous SQL Server 2008

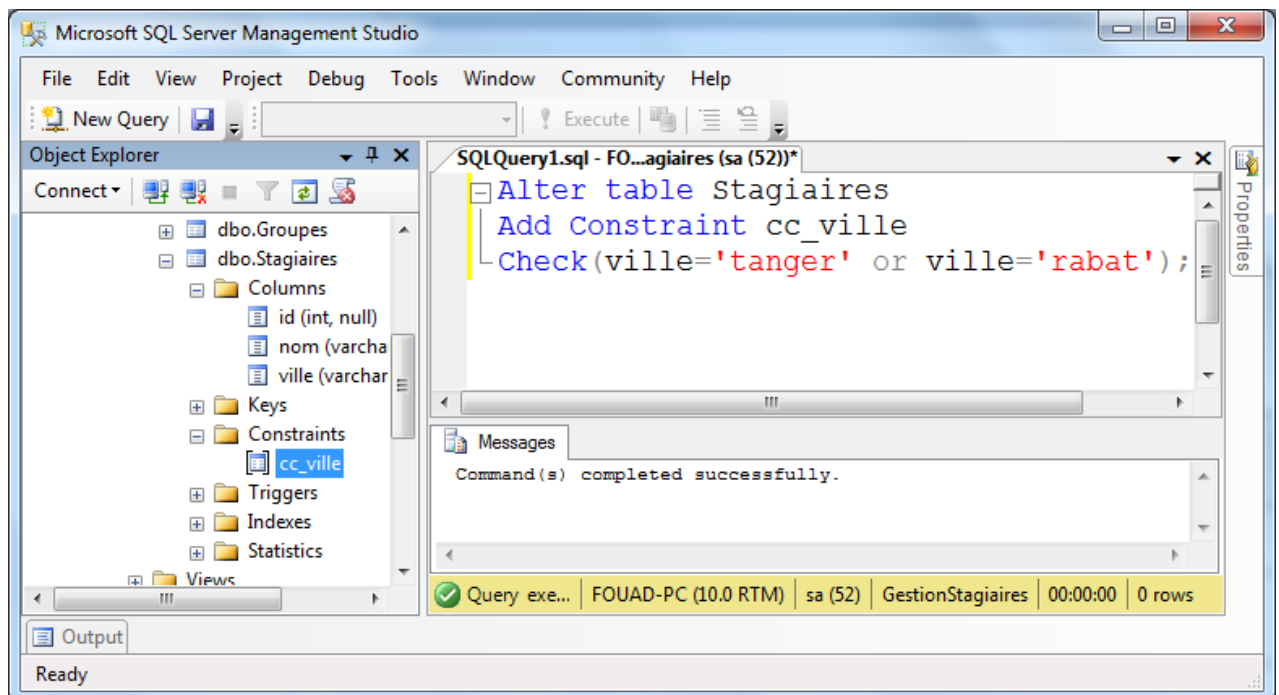


Figure 13 : Ajouter la contrainte Check au champ "Ville" de la table « Stagiaires »

Exemple : Ajouter une contrainte Clé primaire à la table Stagiaires

```
Alter table Stagiaires Add Constraint pk_stagiaire Primary key(id);
```

### Résultat d'exécution sous SQL Server 2008

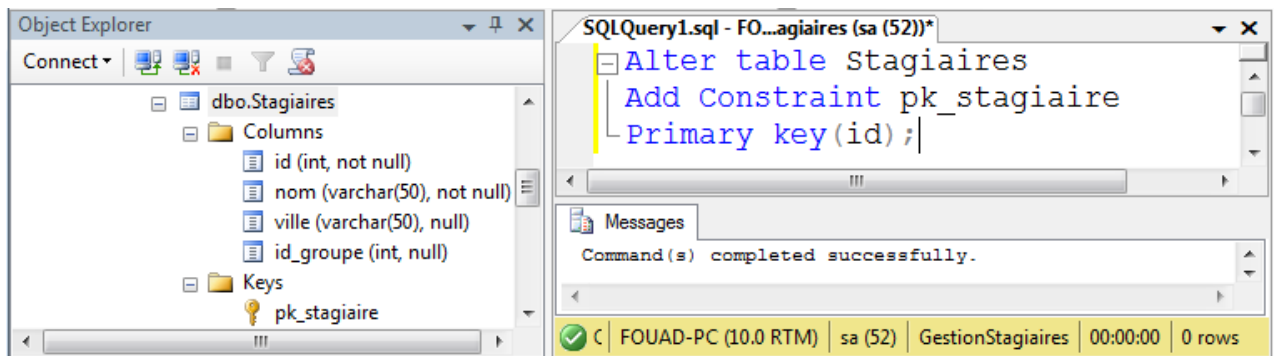


Figure 14 : Ajouter une contrainte Clé primaire à la table « Stagiaires »

*Exemple : Ajouter une contrainte Clé primaire à la table Groupes*

```
Alter table Groupes Add Constraint pk_groupes Primary key(id);
```

Résultat d'exécution sous SQL Server 2008

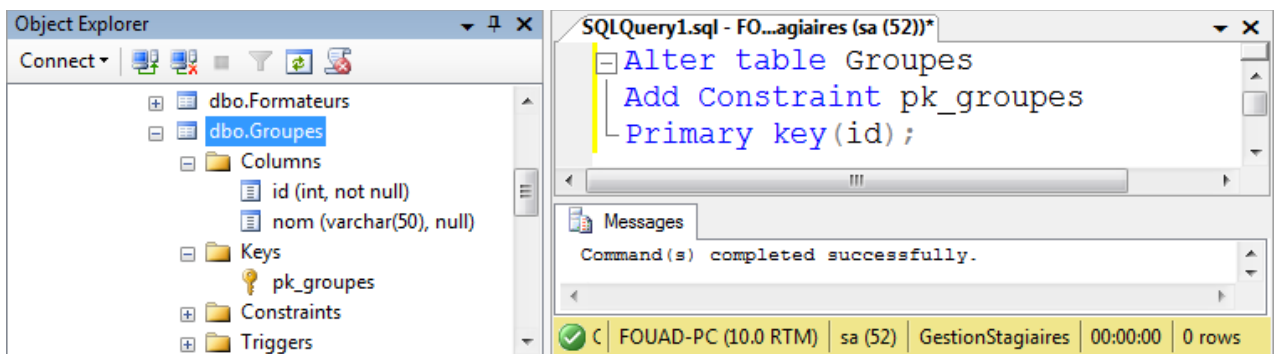


Figure 15 : Ajouter une contrainte Clé primaire à la table « Groupes »

*Exemple : Ajouter une contrainte « Foreign key » entre la table stagiaires et groupes*

```
Alter table Stagiaires  
Add Constraint fk_stagiaires_groupes  
Foreign key(id_groupe) References Groupes(id);
```

Résultat d'exécution sous SQL Server 2008

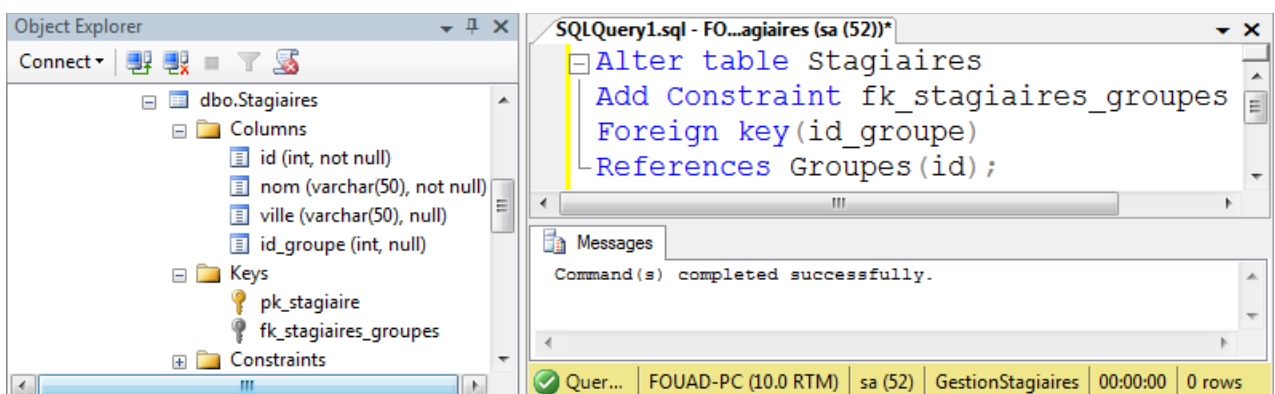


Figure 16 : Ajouter une contrainte « Foreign key » entre la table « Stagiaires » et « Groupes »

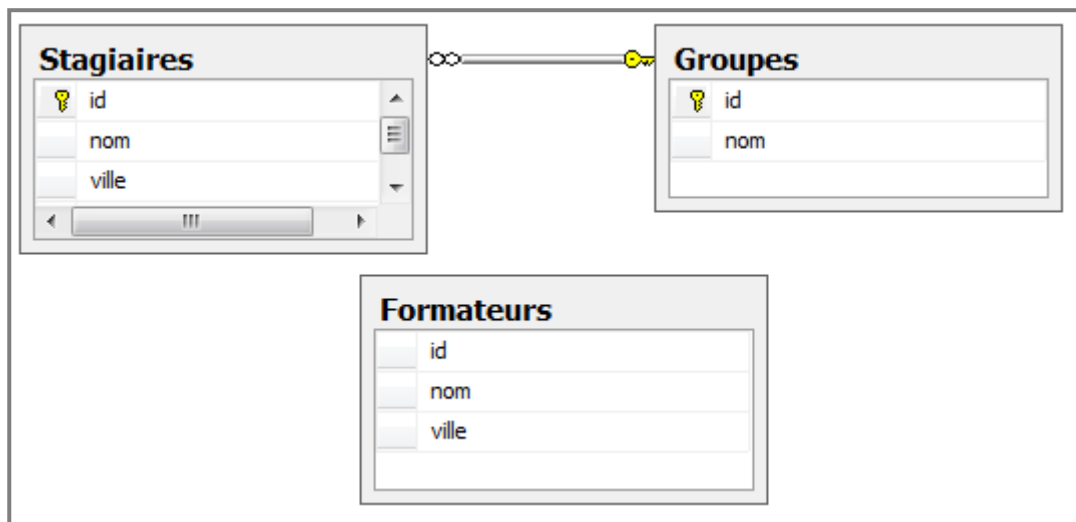


Figure 17 : Base de données de gestion des stagiaires, version 3

### Supprimer une colonne :

*Exemple : Supprimer la colonne « prenom ».*

```
Alter table Stagiaires Drop column prenom;
```

*Remarque : Supprimer la colonne « Ville ».*

```
Alter table Stagiaire Drop column ville;
```

On ne peut pas supprimer cette colonne car elle liée à la contrainte « cc\_ville ».

### Supprimer une contrainte :

*Exemple : Supprimer la contrainte « cc\_ville ».*

```
Alter table Stagiaire Drop Constraint cc_ville;
```

## Chapitre 3 : Langage de manipulation de données (LMD)

Nous allons aborder dans ce chapitre les ordres du langage de manipulation de données. Nous allons commencer par l'ordre Insert.

### Insertion d'enregistrement : « Insert »

Trois formes d'insertion :

- ⇒ Insérer tous les attributs
- ⇒ Insérer avec le choix d'attributs à utiliser
- ⇒ Insérer plusieurs enregistrements

Requête d'insertion avec l'utilisation de toutes les colonnes de la table :

Pour ajouter le stagiaire 'Ali' de numéro 1 et de groupe numéro 1 nous utilisons la requête suivant :

*Requête 26 : Ajouter le groupe TD4*

```
Insert into groupes values(1,'TDI4');
```

Résultat d'exécution

	id	nom
1	1	TDI4

*Requête 27 : Ajouter le stagiaire « Ali » au groupe TDI4*

```
Insert into stagiaires values(1,'Ali','Tanger',1);
```

Résultat d'exécution

	id	nom	ville	id_groupe
1	1	Ali	Tanger	1

Requête d'insertion avec l'utilisation quelques colonnes de la table :

Insertion du stagiaire 'Karim' de numéro 1 sans spécifier son groupe.

```
Insert into Stagiaires(Id, Nom) values (2,'Karim');
```

Résultat d'exécution

	id	nom	ville	id_groupe
1	1	Ali	Tanger	1
2	2	Karim	NULL	NULL

*Remarque :*

SQL server va affecter, ici, la valeur « Null » au colonne « Id\_groupe ».

Nous disons que la valeur **Null**, **ici**, est **implicite**. par ce que c'est le SQL Server qui a affecté cette valeur.

**Requête d'insertion de plusieurs stagiaires :**

```
Insert into Stagiaires (id,nom) values (3,'Mouna'),(4,'Karima') ;
```

**Résultat d'exécution**

	id	nom	ville	id_groupe
1	1	Ali	Tanger	1
2	2	Karim	NULL	NULL
3	3	Mouna	NULL	NULL
4	4	Karima	NULL	NULL

*Remarque :*

Les valeurs peuvent être explicitement **Null** ou **Default** c'est-à-dire que l'utilisateur peut choisir de donner des valeurs comme null ou bien par défaut ( ici nous avons inséré le 1<sup>er</sup> stagiaire avec un **nom Null** et le 2<sup>ème</sup> avec un **nom par défaut**).

```
Insert into Stagiaires (id,nom,ville)  
values (5,'Mouhamed',Null),(6,'Ali',Default);
```

**Résultat d'exécution**

	id	nom	ville	id_groupe
1	1	Ali	Tanger	1
2	2	Karim	NULL	NULL
3	3	Mouna	NULL	NULL
4	4	Karima	NULL	NULL
5	5	Mouhamed	NULL	NULL
6	6	Ali	NULL	NULL

**Modification d'enregistrement : « Update »**

*Exemple : Mise à jour d'un stagiaire*

Nous voudrions modifier le nom et le prénom de stagiaire numéro 1 ainsi que son groupe.

```
Update Stagiaires  
Set nom = 'Moad', ville = 'Rabat'  
where id=2 ;
```

### Résultat d'exécution

	id	nom	ville	id_groupe
1	1	Ali	Tanger	1
2	2	Moad	Rabat	NULL
3	3	Mouna	NULL	NULL
4	4	Karima	NULL	NULL
5	5	Mouhamed	NULL	NULL
6	6	Ali	NULL	NULL

### Remarque

Si nous éliminons la condition « Where » la requête va modifier le nom et le prénom de tous les stagiaires.

## Suppression d'enregistrement : « Delete »

*Exemple : Supprimer un stagiaire*

La requête consiste à supprimer le stagiaire numéro 3.

```
Delete from Stagiaires where id = 3;
```

### Résultat d'exécution

	id	nom	ville	id_groupe
1	1	Ali	Tanger	1
2	2	Moad	Rabat	NULL
3	4	Karima	NULL	NULL
4	5	Mouhamed	NULL	NULL
5	6	Ali	NULL	NULL

## Manipulation des données avec le type : « DateTime »

Pour traiter ce type de données, nous ajoutons la colonne « DateNaissance » de type « DateTime » à la table « Stagiaires » avec l'utilisation de requête « Alter table ».

**Exemple 3 : Utilisation de type DateTime dans SQL Server 2008"**

*Requête 28 : Ajoute de la colonne « DateNaissance » à la table Stagiaires*

```
Alter table Stagiaires add DateNaissance Datetime
```

### Résultat d'exécution

	id	nom	ville	id_groupe	DateNaissance
1	1	Ali	Tanger	1	NULL
2	2	Moad	Rabat	NULL	NULL
3	4	Karima	NULL	NULL	NULL
4	5	Mouhamed	NULL	NULL	NULL
5	6	Ali	NULL	NULL	NULL

*Requête 29 : Mise à jour de la date de naissance avec date et heur : minute : second*

Update Stagiaires Set DateNaissance = '20130918 13:22:06' where id =1 ;

Résultat d'exécution

	id	nom	ville	id_groupe	DateNaissance
1	1	Ali	Tanger	1	2013-09-18 13:22:06.000
2	2	Moad	Rabat	NULL	NULL
3	4	Karima	NULL	NULL	NULL
4	5	Mouhamed	NULL	NULL	NULL
5	6	Ali	NULL	NULL	NULL

*Requête 30 : Mise à jour de la date de naissance avec date et heur : minute*

Update Stagiaires Set DateNaissance = ' 20130918 13 :20' where id = 2 ;

Résultat d'exécution

	id	nom	ville	id_groupe	DateNaissance
1	1	Ali	Tanger	1	2013-09-18 13:22:06.000
2	2	Moad	Rabat	NULL	2013-09-18 13:20:00.000
3	4	Karima	NULL	NULL	NULL
4	5	Mouhamed	NULL	NULL	NULL
5	6	Ali	NULL	NULL	NULL

*Requête 31 : Mise à jour de la date de naissance avec seulement la date*

Update Stagiaires Set DateNaissance = '2013/09/14' where id =4 ;

Résultat d'exécution

	id	nom	ville	id_groupe	DateNaissance
1	1	Ali	Tanger	1	2013-09-18 13:22:06.000
2	2	Moad	Rabat	NULL	2013-09-18 13:20:00.000
3	4	Karima	NULL	NULL	2013-09-14 00:00:00.000
4	5	Mouhamed	NULL	NULL	NULL
5	6	Ali	NULL	NULL	NULL

*Requête 32 : Recherche par date*



```
Select * from Stagiaires
where DateNaissance between '20130918 12 :00' and '20130918 14 :00' ;
```

Résultat d'exécution

	id	nom	ville	id_groupe	DateNaissance
1	1	Ali	Tanger	1	2013-09-18 13:22:06.000
2	2	Moad	Rabat	NULL	2013-09-18 13:20:00.000

Requête 33 : Mise à jour la date de naissance par date d'aujourd'hui

```
Update Stagiaires Set DateNaissance = CURRENT_TIMESTAMP where id =5
;
```

Résultat d'exécution

	id	nom	ville	id_groupe	DateNaissance
1	1	Ali	Tanger	1	2013-09-18 13:22:06.000
2	2	Moad	Rabat	NULL	2013-09-18 13:20:00.000
3	4	Karima	NULL	NULL	2013-09-14 00:00:00.000
4	5	Mouhamed	NULL	NULL	2013-10-27 18:36:09.090
5	6	Ali	NULL	NULL	NULL

Requête 34 : Recherche par jour

```
Select *from Stagiaires where Day(DateNaissance)=14 ;
```

Résultat d'exécution

	id	nom	ville	id_groupe	DateNaissance
1	4	Karima	NULL	NULL	2013-09-14 00:00:00.000

Requête 35 : Recherche par Mois

```
Select *from Stagiaires where Month (DateNaissance)=14 ;
```

Résultat d'exécution

	id	nom	ville	id_groupe	DateNaissance
1	1	Ali	Tanger	1	2013-09-18 13:22:06.000
2	2	Moad	Rabat	NULL	2013-09-18 13:20:00.000
3	4	Karima	NULL	NULL	2013-09-14 00:00:00.000

Requête 36 : Recherche par Année

```
Select * from Stagiaires where Year (DateNaissance)=14 ;
```

**Résultat d'exécution**

	id	nom	ville	id_groupe	DateNaissance
1	1	Ali	Tanger	1	2013-09-18 13:22:06.000
2	2	Moad	Rabat	NULL	2013-09-18 13:20:00.000
3	4	Karima	NULL	NULL	2013-09-14 00:00:00.000

## Chapitre 4 : Interrogation des données

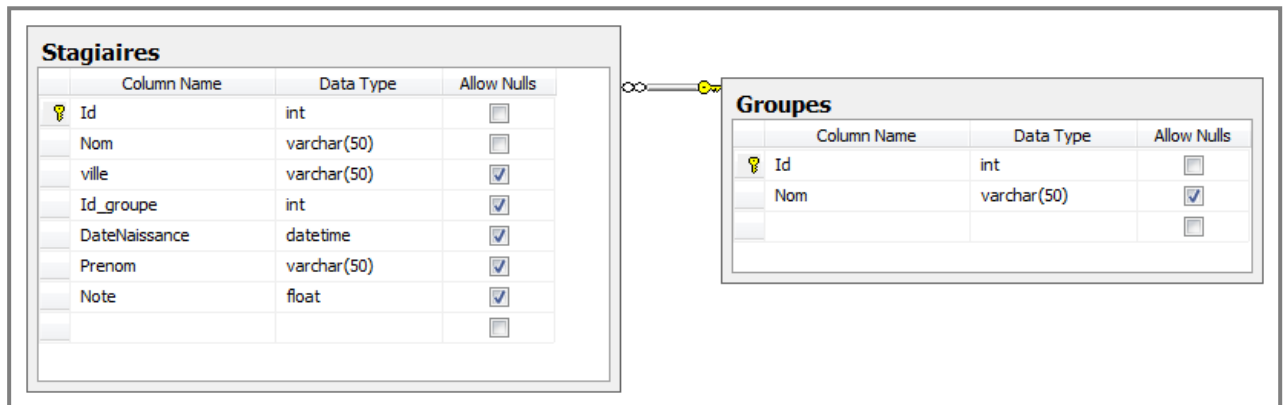


Figure 18 : Base de données d'exemples de chapitre 4

	Id	Nom	Prenom	ville	DateNaissance	Note	Id_groupe
1	1	Madani	Ismail	Tanger	1995-09-18 13:22:06.000	9,5	1
2	2	Madani	Mouhssine	Rabat	1995-09-18 13:20:00.000	17	1
3	4	Wahabi	Imaran	Tanger	1995-09-14 00:00:00.000	14	1
4	5	Wahabi	Youssef	Tanger	1995-10-27 18:36:09.090	11,5	2
5	6	Andaloussi	Yasimina	Rabat	1995-10-27 18:36:09.090	13	2

Figure 19 : Données de la table « Stagiaires », Chapitre 4

	Id	Nom
1	1	TDI4
2	2	TDI5

Figure 20 : Données de la table « Groupes », Chapitre 4

### Extraction des données : « Select »

Pour extraire ou chercher des informations dans la base de données, nous utilisons la requête « Select ».

*Requête 37 : afficher la liste de tous les stagiaires*

```
Select * from stagiaires ;
```

#### Résultat d'exécution

	Id	Nom	ville	Id_groupe	DateNaissance	Prenom	Note
1	1	Madani	Tanger	1	1995-09-18 13:22:06.000	Ismail	9,5
2	2	Madani	Rabat	1	1995-09-18 13:20:00.000	Mouhssine	17
3	4	Wahabi	Tanger	1	1995-09-14 00:00:00.000	Imaran	14
4	5	Wahabi	Tanger	2	1995-10-27 18:36:09.090	Youssef	11,5
5	6	Andaloussi	Rabat	2	1995-10-27 18:36:09.090	Yasimina	13

*Requête 38 : afficher les noms de tous les stagiaires*

```
Select nom from stagiaires ;
```

Résultat d'exécution

	nom
1	Madani
2	Madani
3	Wahabi
4	Wahabi
5	Andaloussi

### Duplicate : « distinct »

Si on a plusieurs données similaires à l'instant de l'affichage du résultat, distinct nous permet d'éliminer les doublant.

*Requête 39 : Utilisation de « distinct » pour éliminer les informations redoublantes.*

```
Select distinct(nom) from stagiaires;
```

Résultat d'exécution

	nom
1	Andaloussi
2	Madani
3	Wahabi

### Select avec expression :

Cet exemple nous montre qu'on peut modifier les données d'une colonne et les mettre dans une nouvelle colonne qui prend un nouveau nom qu'on appelle, ici « NotePlus »

*Requête 40 : Afficher la note augmentée par 10 %.*

```
Select Nom,Prenom,note,(note * 0.1 + note) as NotePlus from Stagiaires;
```

Résultat d'exécution

	Nom	Prenom	note	NotePlus
1	Madani	Ismail	9,5	10,45
2	Madani	Mouhssine	17	18,7
3	Wahabi	Imaran	14	15,4
4	Wahabi	Youssef	11,5	12,65
5	Andaloussi	Yasimina	13	14,3

## Ordonnement : « Order By »

*Requête 41 : Affichage de toutes stagiaires avec ordre par défaut, croissant*

```
Select Nom,Prenom, Note from stagiaires ORDER BY note;
```

Résultat d'exécution

	Nom	Prenom	Note
1	Madani	Ismail	9,5
2	Wahabi	Youssef	11,5
3	Andaloussi	Yasimina	13
4	Wahabi	Imaran	14
5	Madani	Mouhssine	17

La clause **ORDER BY** est utilisée dans une instruction **SELECT** pour trier les données d'une table (ou plusieurs tables) en fonction d'une ou plusieurs colonnes.

Par défaut, le rangement se fera par ordre croissant ou par ordre alphabétique.

Avec le mot clé **ASC**, le rangement se fera dans l'ordre ascendant ↗. Avec le mot clé **DESC**, le rangement se fera dans l'ordre descendant ↘.

*Requête 42 : Affichage de toutes stagiaires avec ordre croissant des notes*

```
Select Nom,Prenom, Note from stagiaires ORDER BY note ASC;
```

Résultat d'exécution

	Nom	Prenom	Note
1	Madani	Ismail	9,5
2	Wahabi	Youssef	11,5
3	Andaloussi	Yasimina	13
4	Wahabi	Imaran	14
5	Madani	Mouhssine	17

*Requête 43 : Affichage de toutes stagiaires avec ordre décroissante des notes*

```
Select Nom,Prenom, Note from stagiaires ORDER BY note DESC;
```

Résultat d'exécution

	Nom	Prenom	Note
1	Madani	Mouhssine	17
2	Wahabi	Imaran	14
3	Andaloussi	Yasimina	13
4	Wahabi	Youssef	11,5
5	Madani	Ismail	9,5

## Concaténation : « + »

C'est associer des informations venant de plusieurs colonnes de type chaîne de caractère dans une nouvelle colonne qui porte un nouveau nom qu'on appelle **Alias**.

*Requête 44 : Affichage de Nom Complet (Nom + Prénom) de tous les stagiaires.*

```
Select nom+' '+prenom as NomComplet , ville from Stagiaires;
```

Résultat d'exécution

	NomComplet	ville
1	Madani Ismail	Tanger
2	Madani Mouhssine	Rabat
3	Wahabi Imaran	Tanger
4	Wahabi Youssef	Tanger
5	Andaloussi Yasimina	Rabat

## Opérateurs intégrés :

### L'opérateur « IN » :

Permet de rechercher une valeur qui se trouve dans un ensemble de données.

*Requête 45 : Afficher les stagiaires dont la ville = Tanger ou Casa*

```
Select * from Stagiaires where Ville IN ('Tanger','Casa');
```

Résultat d'exécution

	Id	Nom	ville	Id_groupe	DateNaissance	Prenom	Note
1	1	Madani	Tanger	1	1995-09-18 13:22:06.000	Ismail	9,5
2	4	Wahabi	Tanger	1	1995-09-14 00:00:00.000	Imaran	14
3	5	Wahabi	Tanger	2	1995-10-27 18:36:09.090	Youssef	11,5

### L'opérateur « LIKE » :

Permet de rechercher des chaînes de caractères avec des conditions selon le besoin, on utilise le symbole « % » pour remplacer un alphabet ; par exemple si on veut rechercher les stagiaires dont le nom contient 'A' au milieu, la syntaxe sera comme suit :

*Requête 46 : Utilisation de l'opérateur «like» pour chercher des stagiaires*

```
select * from Stagiaires where nom LIKE ('%m%') ;
```

Résultat d'exécution

	Id	Nom	ville	Id_groupe	DateNaissance	Prenom	Note
1	1	Madani	Tanger	1	1995-09-18 13:22:06.000	Ismail	9,5
2	2	Madani	Rabat	1	1995-09-18 13:20:00.000	Mouhssine	17

## Statistiques sur les données (Sum, Min, Max, Avg, Count)

Les requêtes utilisant les fonctions **AVG**, **COUNT**, **SUM**, **MAX** et **MIN** ne renvoient qu'une seule valeur.

- **SUM ()** : somme des valeurs d'une colonne
- **AVG ()** : moyenne des valeurs d'une colonne
- **MAX ()** : maximum des valeurs d'une colonne
- **MIN ()** : minimum des valeurs d'une colonne.
- **COUNT ()** : compte le nombre des lignes.

*Requête 47 : Afficher la note moyenne de tous les stagiaires*

```
select AVG (note) as NoteMoyenne from Stagiaires;
```

Résultat d'exécution

	NoteMoyenne
1	13

*Requête 48 : Afficher la note maximale de tous les stagiaires*

```
select MAX(note) as NoteMaximumfrom Stagiaires;
```

Résultat d'exécution

	NoteMaximum
1	17

## Regroupement : « GROUP BY »

*Requête 49 : Afficher les notes maximums de toutes les villes*

```
Select Ville , max(note) as NoteMax From Stagiaires GROUP BY ville
```

Résultat d'exécution

	Ville	NoteMax
1	Rabat	17
2	Tanger	14

Remarque

Tous les attributs placés derrière la clause **SELECT** doivent être présents derrière la clause **GROUP BY**.

### L'élimination de certains groupes par « **HAVING** »:

Lorsqu'on obtient le résultat d'une requête issue d'un regroupement, on peut souhaiter ne sélectionner que certains groupes parmi d'autres. Pour cela, il faut ajouter à **GROUP BY** la clause **HAVING** qui permet la sélection de groupes parmi d'autres.

*Requête 50 : Afficher les notes maximums supérieures à 10 de toutes les villes en utilisation de « **Having** »*

```
Select Ville , max(note) as NoteMax From Stagiaires GROUP BY ville  
HAVING max(note) > 15 ;
```

Résultat d'exécution

	Ville	NoteMax
1	Rabat	17

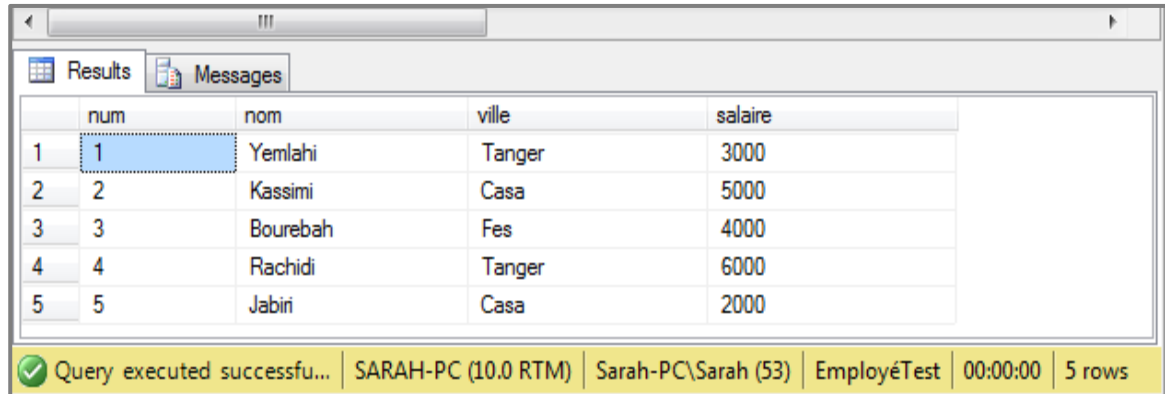
### Remarque

Il ne pas confondre **HAVING** et **WHERE** : **HAVING** permet la sélection de groupes à la suite d'une requête avec regroupement alors que **WHERE** permet de sélectionner des lignes pour construire la requête.



## Chapitre 5 : Sous interrogations

Prenant l'exemple de la table **Employé** pour bien comprendre :



	num	nom	ville	salaire
1	1	Yemlahi	Tanger	3000
2	2	Kassimi	Casa	5000
3	3	Bourebah	Fes	4000
4	4	Rachidi	Tanger	6000
5	5	Jabiri	Casa	2000

Query executed successfully... SARAH-PC (10.0 RTM) Sarah-PC\Sarah (53) EmployéTest 00:00:00 5 rows

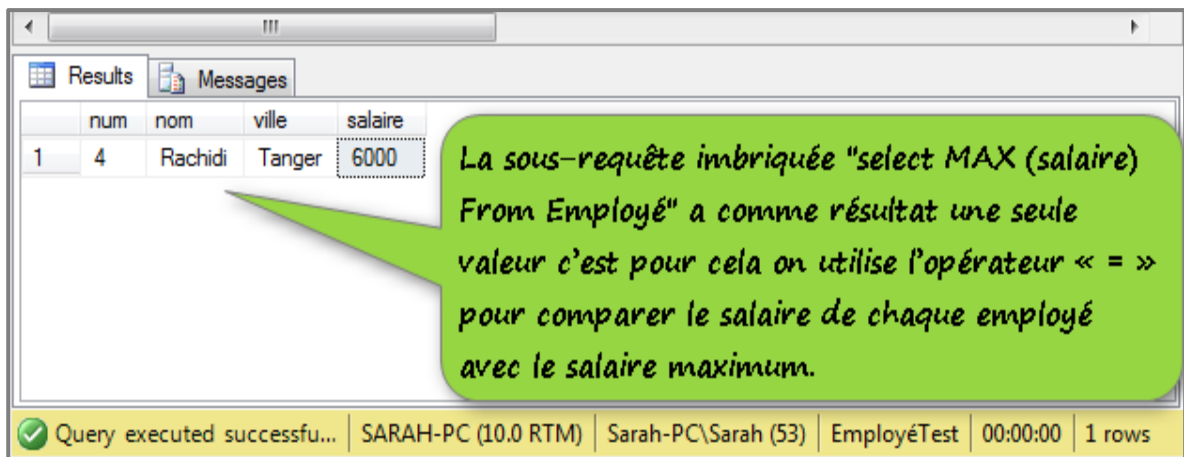
Figure 1 : Table Employé

### Sous-interrogations mono-lignes

*Requête 51 : Afficher la liste des employés ayant le salaire maximal.*

```
Select * From Employé where salaire =(select MAX(salaire) From Employé) ;
```

Résultat :



	num	nom	ville	salaire
1	4	Rachidi	Tanger	6000

La sous-requête imbriquée "select MAX (salaire) From Employé" a comme résultat une seule valeur c'est pour cela on utilise l'opérateur « = » pour comparer le salaire de chaque employé avec le salaire maximum.

Query executed successfully... SARAH-PC (10.0 RTM) Sarah-PC\Sarah (53) EmployéTest 00:00:00 1 rows

Figure 2 : Résultat sous-interrogations mono-lignes.

### Sous-interrogations multi-lignes : (IN, ANY, ALL)

#### Utilisation de « IN » :

Le principe consiste à construire une sous-requête qui donne un résultat équivalent à une liste.

Ensuite la requête principale permet de sélectionner des lignes dans la liste précédemment construite.

*Requête 52 : Utilisation sous-interrogation avec l'opérateur « IN »*

```
Select * from Employé where salaire IN (Select MAX(salaire) as SalaireMax from  
Employé GROUP BY ville) ;
```

Résultat d'exécution sous SQL Server 2008 :

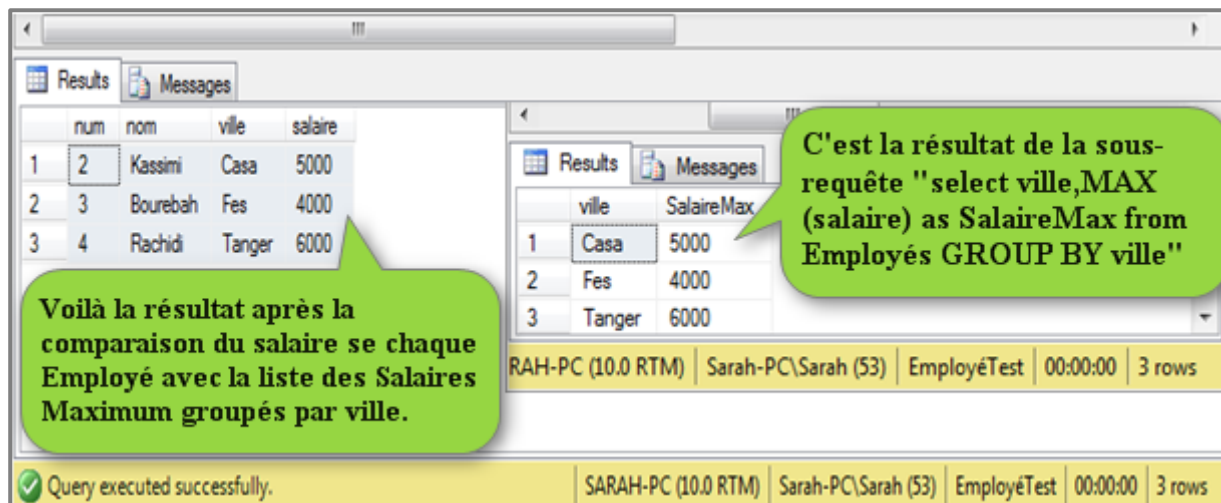


Figure 3 : Résultat de sous-interrogations IN

**Utilisation de « ANY » :**

Nous voudrions afficher la liste des employés ayant le salaire supérieur au moins d'un salaire maximale d'une ville.

*Requête 53 : Utilisation sous-interrogation avec l'opérateur « Any »*

```
Select * From Employé where salaire > ANY (Select MAX (salaire) from  
Employés GROUP BY ville) ;
```

Résultat d'exécution sous SQL Server 2008 :

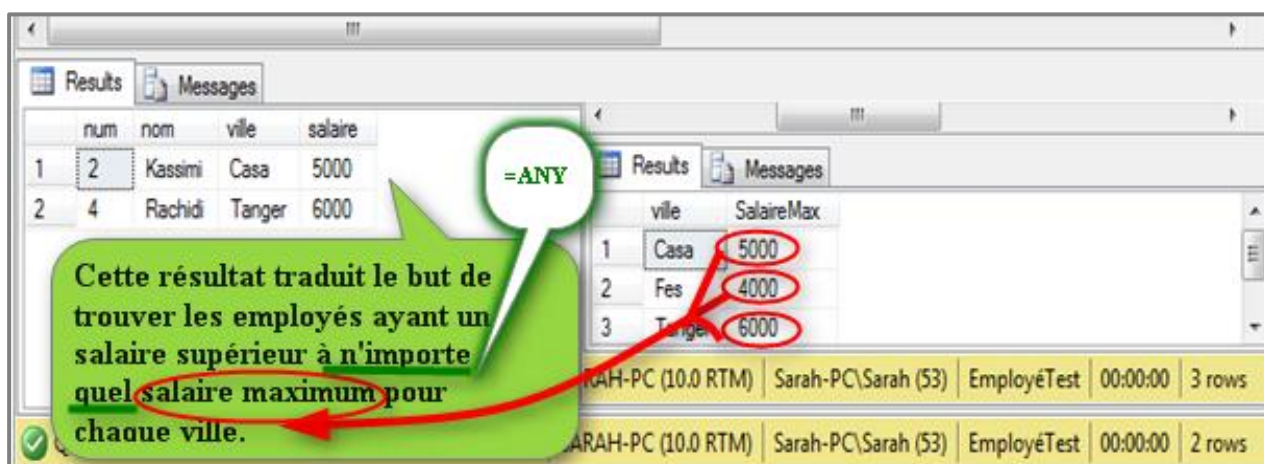


Figure 4 : Résultat de sous-interrogations ANY

### Utilisation de « ALL » :

#### Exemple :

Nous voudrions afficher la liste des employés ayant le salaire inférieur de tous les salaires maximums de toutes les villes.

*Requête 54 : Utilisation sous-interrogation avec l'opérateur « All »*

```
Select * From Employés where salaire < All(select MAX (salaire) from Employés  
GROUP BY ville) ;
```

Résultat d'exécution sous SQL Server 2008 :

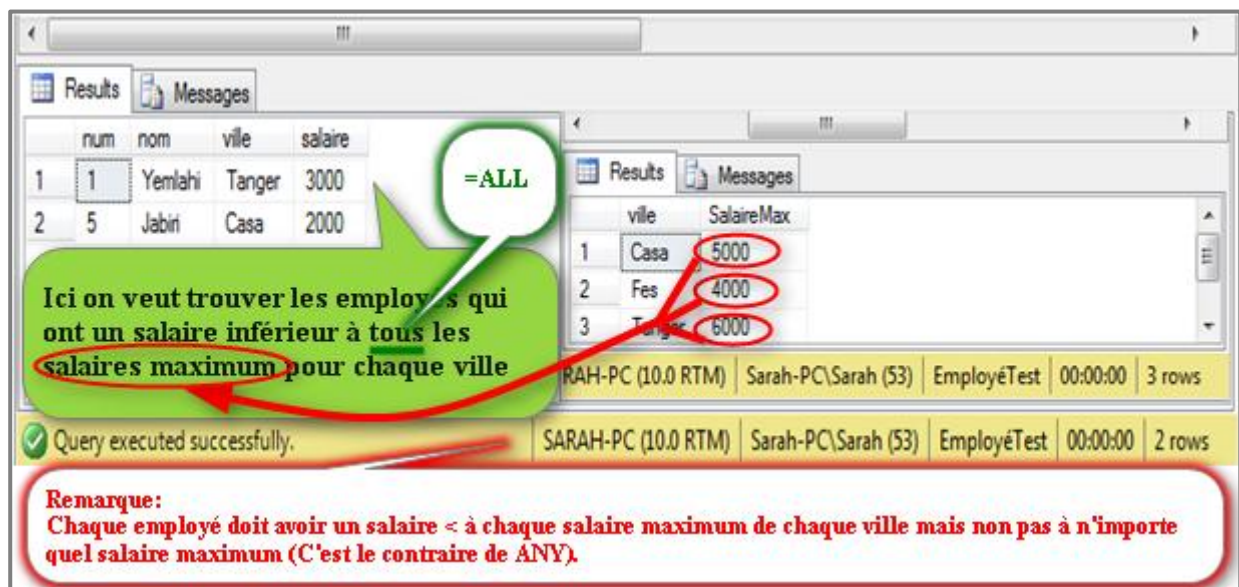


Figure 21 : Résultat de sous-interrogations ALL

## Chapitre 6 : Jointure

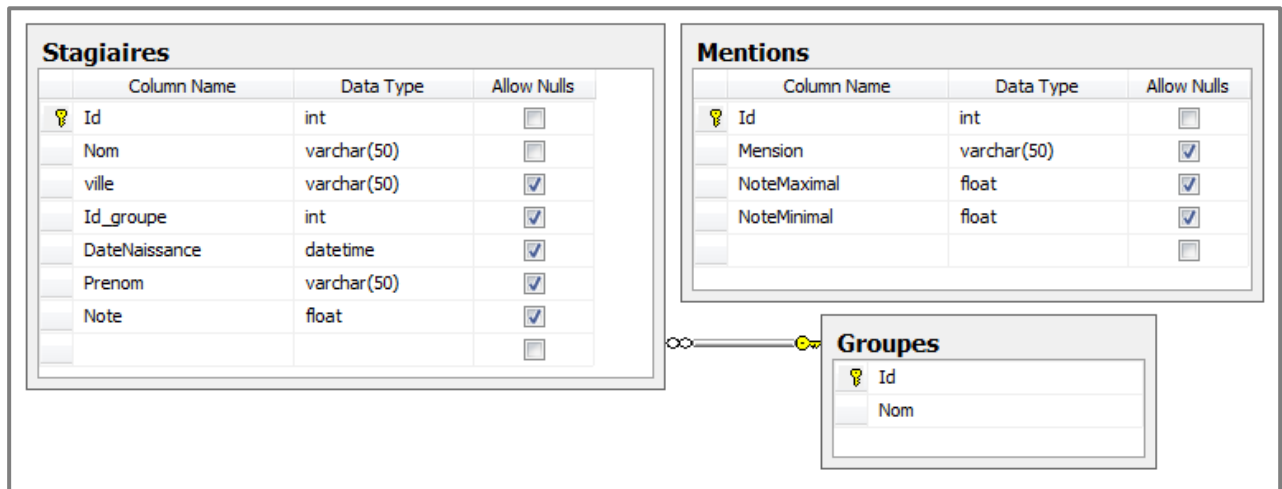


Figure 22 : Base de données des exemples de chapitre 6

	Id	Nom	ville	Id_groupe	DateNaissance	Prenom	Note
1	1	Madani	Tanger	1	1995-09-18 13:22:06.000	Ismail	9,5
2	2	Madani	Rabat	1	1995-09-18 13:20:00.000	Mouhssine	17
3	4	Wahabi	Tanger	1	1995-09-14 00:00:00.000	Imaran	14
4	5	Wahabi	Tanger	2	1995-10-27 18:36:09.090	Youssef	11,5
5	6	Andaloussi	Rabat	2	1995-10-27 18:36:09.090	Yasimina	13

Figure 23 : Données de la table « Stagiaires », chapitre 6

	Id	Mention	NoteMaximal	NoteMinimal
1	1	Non validé	0	10
2	2	Passable	10	12
3	3	Assez bien	12	14
4	4	Bien	14	17
5	5	Très bien	17	20

Figure 24 : Données de la table « Mentions »

### Produit Cartésien avec SQL

Exemple du produit cartésien entre deux tables Stagiaire et Groupe :

Requête SQL :

```
Select s.Prenom,s.Id_groupe,g.Id, g.Nom from Stagiaires s,Groupes g
```

Résultat d'exécution

	Prenom	Id_groupe	Id	Nom
1	Ismail	1	1	TDI4
2	Mouhssine	1	1	TDI4
3	Imaran	1	1	TDI4
4	Youssef	2	1	TDI4
5	Yasimina	2	1	TDI4
6	Ismail	1	2	TDI5
7	Mouhssine	1	2	TDI5
8	Imaran	1	2	TDI5
9	Youssef	2	2	TDI5
10	Yasimina	2	2	TDI5

#### Remarque :

La multiplication des deux tableaux nous donnent toutes les combinaisons possibles mais sauf les trois résultats où on a **s.id=g.id\_groupe** qui sont correctes d'où vient la notion de **la condition de jointure**.

#### Jointure relationnelle :

Il est caractérisé par une seule clause « **From** » contenant les tables et alias .

#### Équijointure :

Elle utilise l'opérateur **égalité** dans la clause de jointure.

#### Requête 55 : Exemple de « équijointure »

```
Select s.Prenom,s.Id_groupe,g.Id, g.Nom from Stagiaires s,Groupes g
where s.id_groupe = g.id;
```

#### Résultat d'exécution

	Prenom	Id_groupe	Id	Nom
1	Ismail	1	1	TDI4
2	Mouhssine	1	1	TDI4
3	Imaran	1	1	TDI4
4	Youssef	2	2	TDI5
5	Yasimina	2	2	TDI5

#### InÉquijointure :

Il fait intervenir tout type d'opérateur <, <=, >, >=, like,in,Between...

#### Exemple :

Nous voudrions afficher la liste des stagiaires avec leur mention.

*Requête 56 : Exemple de « In Equijointure »*

```
Select Nom,Prenom,Mension from Stagiaires ,Mentions where note >=
NoteMaximal and note <= NoteMinimal;
```

**Résultat d'exécution**

	Nom	Prenom	Mension
1	Madani	Ismail	Non validé
2	Wahabi	Youssef	Passable
3	Wahabi	Imaran	Assez bien
4	Andaloussi	Yasimina	Assez bien
5	Madani	Mouhssine	Bien
6	Wahabi	Imaran	Bien
7	Madani	Mouhssine	Trés bien

## Chapitre 7 : Séquences

La syntax de création de séquence ne diponible que dans SQL Server 2012. Pour réaliser ce principe nous allons utiliser la syntax 'Identity'.

✗ 2005

✗ 2008

✓ 2012

Création de la table « Mention » avec l'auto-incrémentation du champ Id.

*Requête 57 : Création d'une table avec Id « auto-incrémenté », Syntax de « Identity »*

```
Create table Mention (  
id int Identity( 1 ,2 ),  
nom varchar(50),  
nmaxfloat,  
nmin float  
);
```

## Chapitre 8 : Création des vues

Une **vue** « VIEW » est un SELECT qui est transformé en une table virtuelle. C'est tout simplement une requête SELECT que nous stockons dans notre base de donnée sous forme d'une table virtuelle et que nous allons pouvoir appeler comme si c'était une table.

Donc, c'est un outil très intéressant pour encapsuler un SELECT qui pourrait être un peu complexe avec beaucoup de jointure ou bien un SELECT qu'on exécute très régulièrement.

*Requête 58 : Exemple de création d'un view*

```
Create VIEW ListeStagiaires as Select Nom, id from Stagiaires ;
```

Résultat d'exécution

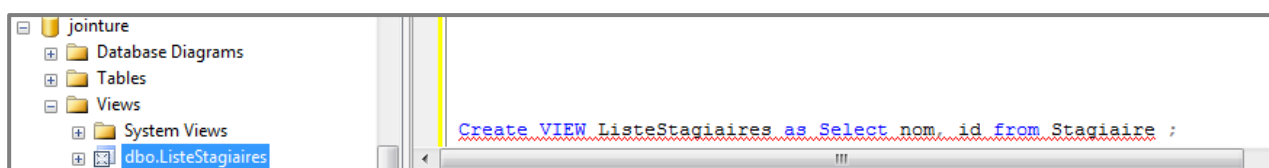


Figure 25 : Exécution sous SQL des vues

Les vues servent aussi à **simplifier** et à **sécuriser**:

### Simplification :

Les structures des tables deviennent peu pratiques à manipuler.

### Sécurité d'accès :

Il est possible de cacher des lignes et des colonnes.



## Chapitre 9 : Index

L'objectif des index est de permettre un accès plus rapide à l'information (Select, Update, Insert).

### Avant la création d'index :

Par exemple, le temps d'exécution de la requête suivante :

```
Select * from Stagiaire where nom= 'Ali' ;
```

Est : 196 ms

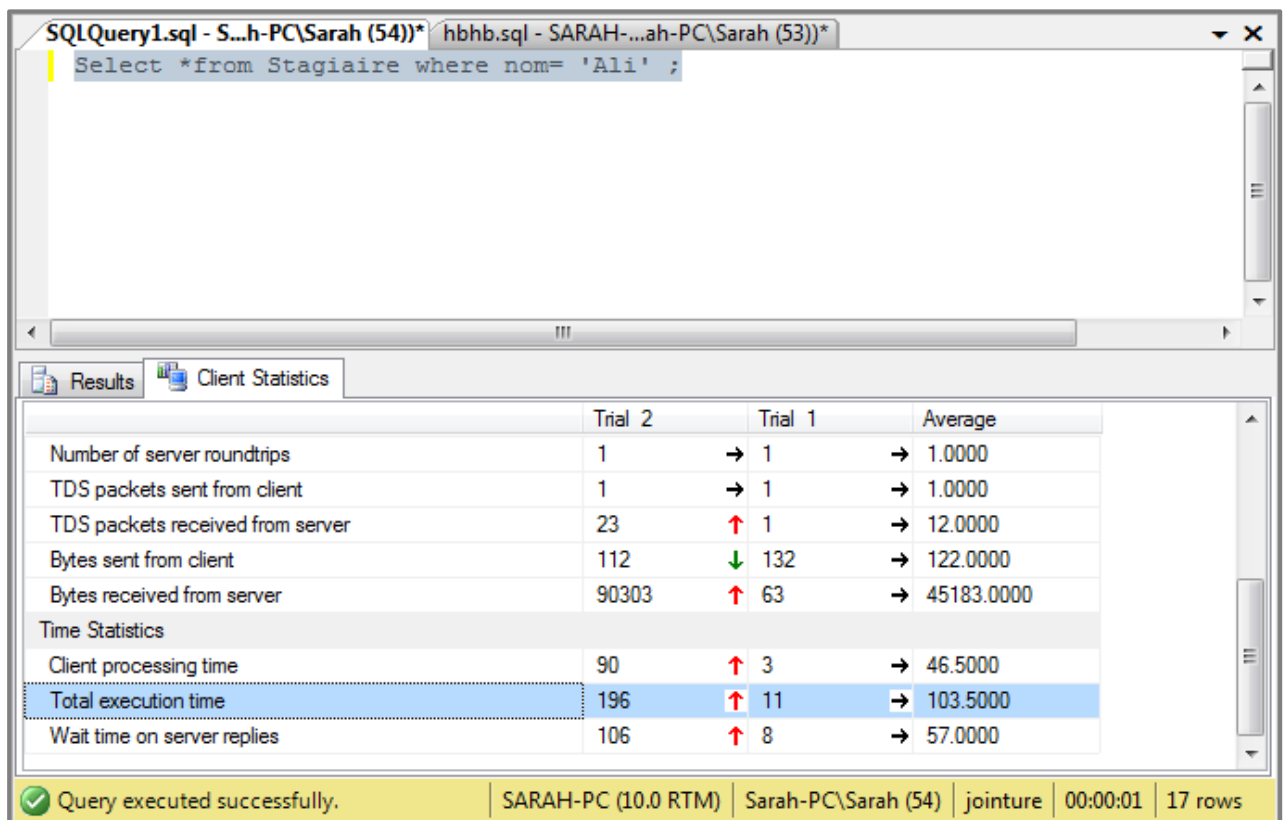


Figure 26 : Temps d'exécution de la requête

### Création des index :

*Requête 59 : Exemple de création d'un index*

Create index index\_nom on Stagiaire(nom);

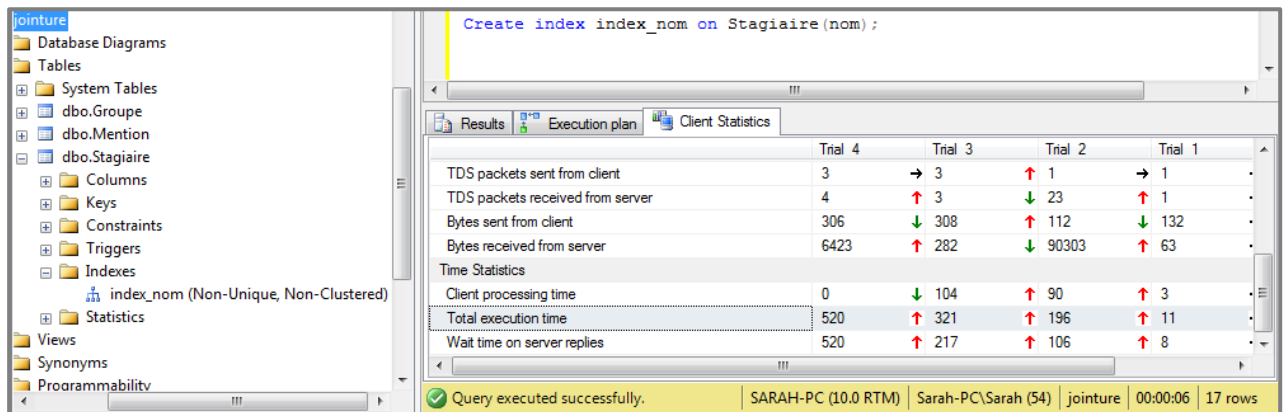


Figure 27 : Création d'index

## Après la réaction de l'index

Le temps d'exécution est passe de 520 ms à 121 ms.

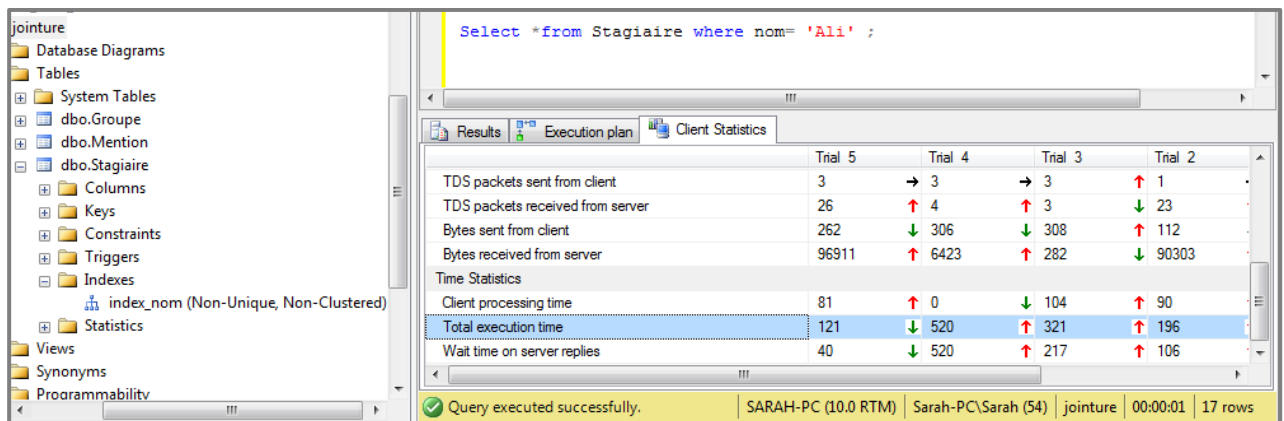


Figure 28 : Après la réaction de l'index

## Chapitre 10 : Transaction

Le modèle le plus simple d'une transaction est celui du transfert d'un compte vers un autre.

```
Update Client Set Compte1=Compte1-500 where id=1 ;
```

```
Update Client Set Compte2=Compte2+500 where id=1 ;
```

Le mécanisme transactionnel empêche une erreur en validant les opérations faites depuis le début de la transaction si une panne survient au cours de cette même transaction.

**Exemple :**

```
Begin Transaction Modification  
save Transaction p1  
Update stg set Nom='A' where id=1;  
save Transaction p2  
Update stg set Nom='B' where id=1;  
Roll back Transaction p2  
Commit Transaction Modification
```

**Remarque :**

Seules les instructions LMD (Select, Insert, Update, Delete) sont prises en compte dans une transaction mais (Create, Alter, Drop, Grant) ne sont pas prises en compte.

## Chapitre 11 : Création des utilisateurs et des rôles

### Création d'une connexion

*Requête 60 : Création d'une connexion : login1*

```
Create login login1 withpassword ='admin' ;
```

### Création utilisateur :

*Requête 61 : Création de l'utilisateur « user1 » de login « login1 »*

```
Create user user1 for login login1
```

### Droit utilisateur : LMD

fixer les autorisations sur les opérations SELECT ,INSERT ,UPDATE ,DELETE

### Accorder des autorisations : grant

Le super administrateur peut donner à l'utilisateur user1 le droit d'utiliser le SELECT dans la table dbo.stagiaires par la requête suivant :

*Requête 62 : autoriser l'utilisateur « user1 » à faire de select sur la table stagiaire*

```
Grant select on dbo.stagiaires to user1 ;
```

### Retirer des autorisations : revoke :

Le super administrateur peut enlever le droit DELETE dans la table dbo.stagiaires à l'utilisateur « User1 » par la requête suivant :

*Requête 63 : supprimer l'autorisation de l'utilisateur « user1 » à faire de delete sur la table stagiaire*

```
Revoke delete on dbo.stagiaires from user1 ;
```

## Droit d'utilisateur : LDD

Ces droits permettent à des utilisateurs de créer leurs propres BD table ,vues,procédures,fonctions

### Accorder: grant

Le super administrateur peut donner à l'utilisateur **user1** le droit de créer des vues.

*Requête 64 : autoriser l'utilisateur « user1 » à créer les view.*

```
Grant create view to user1 ;
```

### Retirer : revoke

Le super administrateur peut enlever le droit de créer des vues à l'utilisateur par la requête suivant.

*Requête 65 : retirer l'autorisation à création des views de l'utilisateur « user1 ».*

```
Revoke create view from user1 ;
```

Ici le super administrateur a enlevé tous les droits à l'utilisateur.

*Requête 66 : retirer tous les autorisations*

```
Revoke all from user1 ;
```

## Partie 2 : Travaux dirigés

« Les **Travaux dirigés** ou **TD** sont une forme d'enseignement qui permet d'appliquer les connaissances apprises pendant les cours théoriques ou d'introduire des notions nouvelles. Les élèves ou les étudiants travaillent individuellement sur des exercices d'application ou de découverte, en présence du professeur, qui intervient pour aider et pour corriger les exercices. Les *travaux dirigés* se font dans un groupe d'effectif réduit, pour que le professeur puisse aider plus facilement les élèves ou étudiants et adapter ses interventions à leurs difficultés », Wikipédia<sup>2</sup>.

Afin d'appliquer les connaissances apprises pendant les séances de cours, quatre travaux pratiques sont proposés par le formateur (Gestion des salles, Gestion des notes, Gestion d'inscription, et Gestion des réservations). Chaque TD comprend les neuf tâches :

- Identification des acteurs
- Détermination de diagramme de contexte
- Détermination de diagramme de cas d'utilisation
- Proposition des scénarios des cas d'utilisations principales.
- Proposition d'un diagramme de séquences dynamiques sur les activités principales
- Détermination des diagrammes de collaboration
- Identification des objets
- Détermination de diagramme de classes
- Détermination des diagrammes d'états

Parfois, l'objectif de TD n'est pas d'appliquer des nouvelles connaissances mais d'introduire des nouvelles notions. Par exemple, la première séance de ce module était une séance de travaux pratiques avec l'activité d'identification des objets existants dans la salle de TP4. L'objectif de ce TD était de développer la notion d'objet et faire la différence entre l'objet et l'attribut.

Le formateur nous a proposé de d'organiser en groupe de six stagiaires. Le même groupe est aussi divisé en trois binômes. Chaque tâche doit être réalisée en binôme. Ensuite, le groupe de six stagiaires doit choisir ou proposer la solution qui représente la groupe. Finalement on aura cinq solutions pour chaque tâche qu'est le nombre des groupes de la classe.

---

<sup>2</sup> [fr.wikipedia.org/wiki/Travaux\\_dirigés](https://fr.wikipedia.org/wiki/Travaux_dirigés)

## TD : Gestion des Stagiaire

### Modèle physique de données

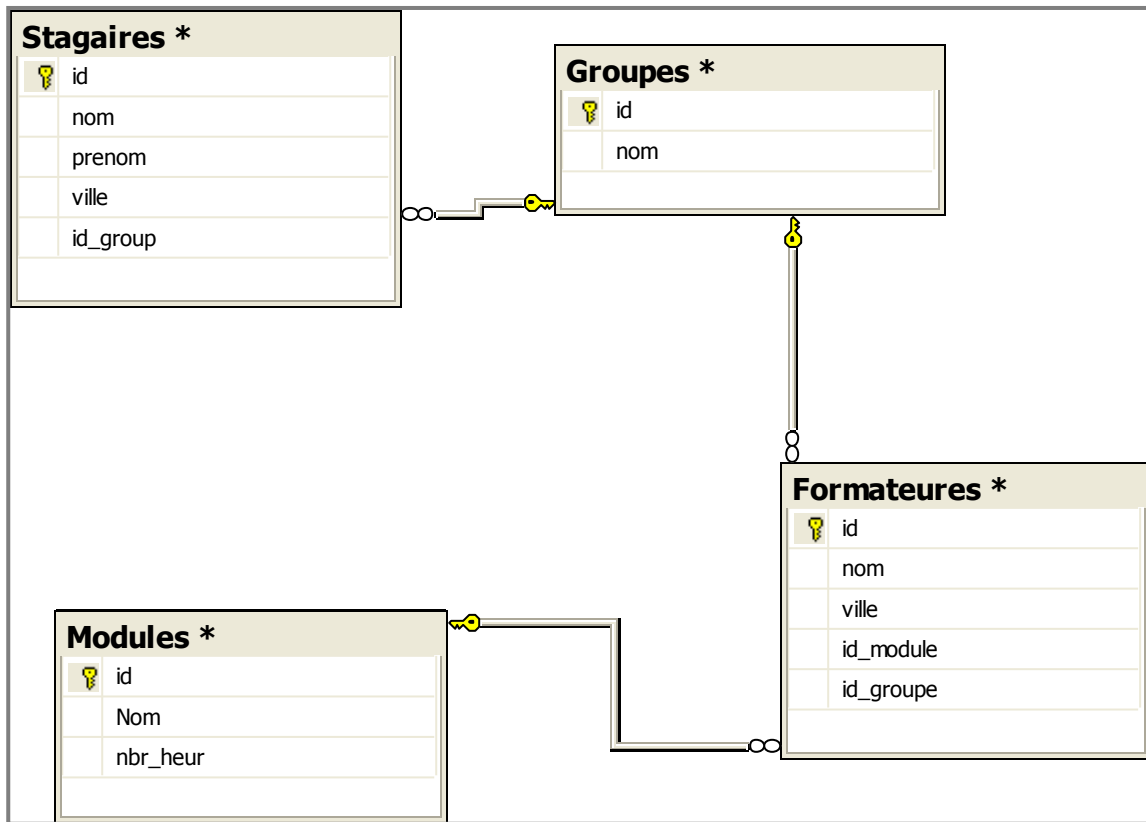


Figure 29 : Base de données de gestion des stagiaires, TD

### Travaux réalisés

- Q1 : Donnez la liste de stagiaires de groupe id=2
- Q2 : Donnez la liste de stagiaire de groupe TDI4
- Q3 : Donnez la liste de stagiaire de Formateur "Ali"
- Q4 : Donnez la liste des stagiaires qui étudient le module "SGBD1".
- Q5 : Donnez le formateur qui enseigne le stagiaire " Ali".

Question 1 : Donnez la liste de stagiaires de groupe id=2

```
Select * from stagiaire where id_groupe=2;
```

Question 2 : Donnez la liste de stagiaire de groupe TDI7

```
Select * from stagiaire s , Groupe g where g.nom='TDI7' and s.id_g=d.id
```

Question 3 : Donnez la liste de stagiaire de Formateur "Ali"

```
Select * from Stagiaire S, Formateur F where S.id_groupe=F.id_groupe and F.nom='Ali';
```

**Question 4 : Donnez la liste des stagiaires qui étudient le module "SGBD1".**

```
Select * from Stagiaire S, Formateur F, Module M where  
S.id_groupe=F.id_groupe and F.id_module=M.id and M.nom='SGBD1';
```

**Question 5 : Donnez le formateur qui enseigne le stagiaire " Ali"**

```
Select distinct * from stagiaire S, Formateur F, Groupe G where S.nom='Ali' and  
S.id_groupe = G.id and F.id_groupe= G.id;
```



## TD : Gestion Approvisionnement

### Modèle physique de données

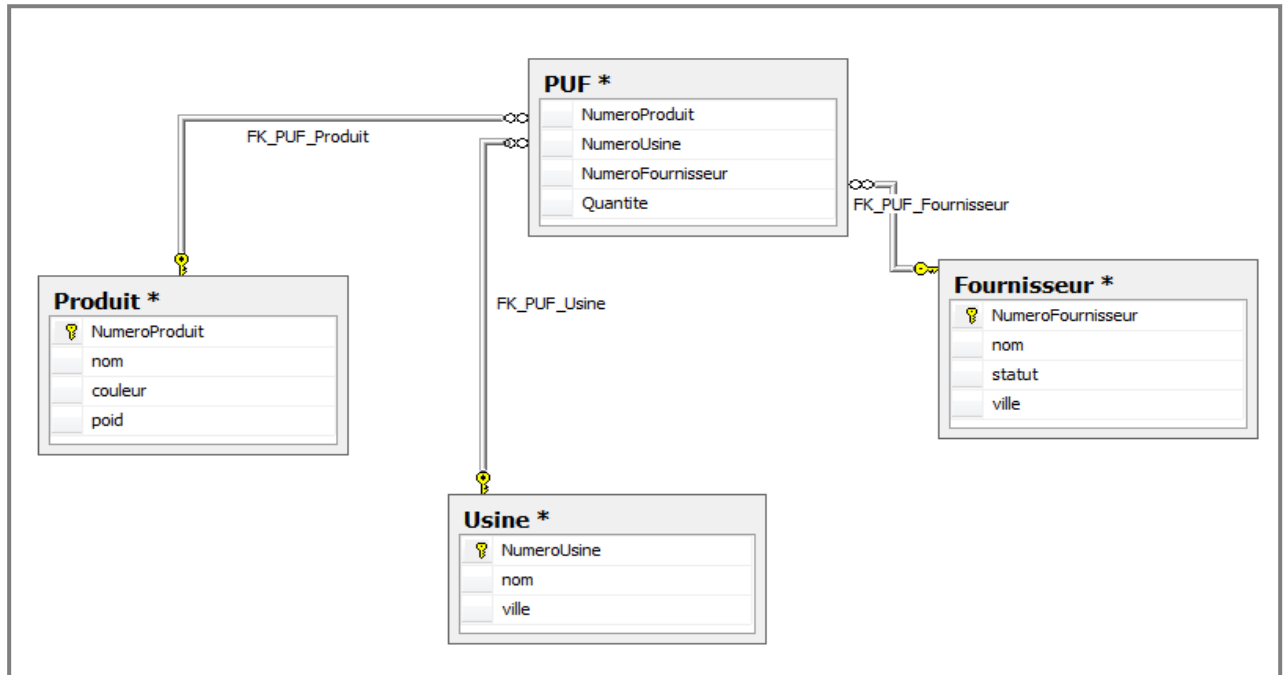


Figure 30 : Base de données de gestion approvisionnement

### Travaux réalisés

- Q1 : Ajouter un nouveau fournisseur avec les attributs de votre choix
- Q2 : Supprimer tout les produits de couleur noir et de numéro compris entre 10 et 99
- Q3 : Changer la ville de fournisseur 3 par Tanger
- Q4 : Donnez le numéro, le nom, la ville de toutes les usines de Tanger
- Q5 : Donnez les numéros des fournisseurs qui approvisionnent l'usine de numéro 2 en produit de numéro 100
- Q6 : Donnez les noms et les couleurs des produit livrés par le fournisseur de numéro=2
- Q7 : Donnez les numéros des fournisseurs qui approvisionnement l'usine de numéro 2 en produit rouge
- Q8 : Donnez les noms des fournisseurs qui approvisionnement une usine de Tanger ou de Rabat en produit rouge
- Q9 : Donnez les numéros des produit livrés a une usine par une fournisseur de la même ville
- Q10 : Donnez les numéros des produit livrés a une usine par une fournisseur de la même ville
- Q11 : Donnez les numéros des produits livrés à une usine de Paris par un fournisseur de Paris

- Q12 : Donnez les numéros des fournisseurs qui approvisionnent à la fois des usines de numéros 2 et 3
- Q13 : Donnez les numéros des usines qui utilisent au moins un produit disponible chez le fournisseur de numéro 3 (c'est-à-dire un produit que le fournisseur livre mais pas nécessairement à cette usine)
- Q14 : Donnez le numéro du produit le plus léger (les numéros si plusieurs produits ont ce même poids)
- Q15 : Donnez le numéro des usines qui ne reçoivent aucun produit rouge d'un fournisseur parisien
- Q16 : Donnez les numéros des fournisseurs qui fournissent au moins un produit fourni par au moins un fournisseur qui fournit au moins un produit rouge

**Question 1 : Ajouter un nouveau fournisseur avec les attributs de votre choix**

```
Insert into Fournisseur values (2, 'nezha ', 'SARL ', 'Rabat') ;
```

**Question 2 : Supprimer tout les produits de couleur noir et de numéro compris entre 10 et 99**

```
Delete from Produit where couleur ='noir ' and NumeroProduit between 10 and 99 ) ;
```

**Question 3 : Changer la ville de fournisseur 3 par Tanger**

```
Update Fournisseur set ville = 'Tanger ' where NumeroFournisseur=3 ;
```

**Question 4 : Donnez le numéro, le nom, la ville de toutes les usines de Tanger**

```
Select * from Usine where ville ='Tanger ' ;
```

**Question 5 : Donnez les numéros des fournisseurs qui approvisionnent l'usine de numéro 2 en produit de numéro 100**

```
Select NumeroFournisseur from PUF where NumeroUsine =2 and NumeroProduit=100 ;
```

**Question 6 : Donnez les noms et les couleurs des produit livrés par le fournisseur de numéro=2**

```
Select nom, couleur from produit, PUF where PUF.NumeroProduit =P.NumeroProduit and NumeroFournisseur =2;
```

**Question 7 : Donnez les numéros des fournisseurs qui approvisionnement l'usine de numéro 2 en produit rouge**

```
Select NumeroFournisseur from PUF pu , Produit p where pu.NumeroProduit  
=p.NumeroProduit and p.couleur='rouge'and NumeroUsine=2;
```

**Question 8 : Donnez les noms des fournisseurs qui approvisionnement une usine de Tanger ou de Rabat en produit rouge**

```
Select f.nom from Fournisseur f , Produit p , PUF pu , Usine u , where  
p.NumeroProduit= pu.NumeroProduit and u.NumeroUsine=pu.NumeroUsine  
and f.NumeroFournisseur= pu.NumeroFournisseur and p.couleur = 'rouge' and  
(u.ville='Rabat' or u.ville='Tanger');
```

**Question 9 : Donnez les numéros des produit livrés a une usine par une fournisseur de la même ville**

```
Select NumeroProduit from PUF pu , Usine u , Fournisseur f , where  
pu.NumeroUsine =u.NumeroUsine and f.NumeroFournisseur =  
pu.NumeroFournisseur and u.ville=f.ville;
```

**Question 10 : Donnez les numéros des produit livrés a une usine par une fournisseur de la même ville**

```
Select u.NumeroUsine from Usine u , Fournisseur f , PUF pu where  
pu.NumeroUsine = u.NumeroUsine and pu.NumeroFournisseur =  
f.NumeroFournisseur and u.ville!=f.ville ;
```

**Question 11 : Donnez les numéros des produits livrés à une usine de Paris par un fournisseur de Paris**

```
Select NumeroProduit from PUF pu , Fournisseur f , Usine u where  
pu.NumeroFournisseur =f.NumeroFournisseur and pu.NumeroUsine =  
u.NumeroUsine and u.ville=f.ville and u.ville='Paris';
```

**Question 12 : Donnez les numéros des fournisseurs qui approvisionnement à la fois des usines de numéros 2 et 3**

```
Select NumeroFournisseur from PUF where NumeroFournisseur in( select  
NumeroFournisseur from PUF where NumeroUsine = 1) and NumeroUsine  
=2;
```

**Question 13 : Donnez les numéros des usines qui utilisent au moins un produit disponible chez le fournisseur de numéro 3 (c'est-à-dire un produit que le fournisseur livre mais pas nécessairement à cette usine)**

```
Select pu.NumeroUsine from PUF where pu.NumeroProduit in ( select  
pu.NumeroProduit from PUF pu where pu.NumeroFournisseur = 3);
```

**Question 14:** Donnez le numéro du produit le plus léger (les numéros si plusieurs produits ont ce même poids)

```
Select NumeroProduit from Produit where poid = ( select min (poid) from  
Produit);
```

**Question 15:** Donnez le numéro des usines qui ne reçoivent aucun produit rouge d'un fournisseur parisien

```
Select NumeroUsine from PUF where NumeroProduit Not in ( Select  
pu.NumeroProduit from PUF pu , Fournisseur F , Produit P where  
pu.NumeroProduit = p.NumeroProduit and pu.NumeroFournisseur  
=f.NumeroFournisseur and f.ville='Paris' and p.couleur =' Rouge');
```

**Question 16:** Donnez les numéros des fournisseurs qui fournissent au moins un produit fourni par au moins un fournisseur qui fournit au moins un produit rouge

```
Select distinct (NumeroFournisseur) from PUF pu where pu.NumeroProduit in (  
select NumeroProduit from PUF where PUF.NumeroFournisseur in ( select  
NumeroFournisseur from PUF pu ,Produit p where pu.NumeroProduit  
=p.NumeroProduit and p.couleur ='Rouge'));
```

## TD : Société « Tanger Soft »

### Modèle physique de données

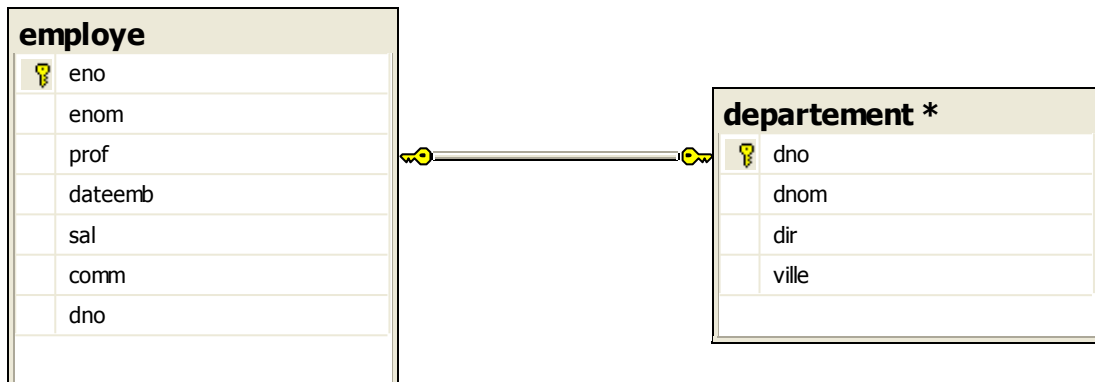


Figure 31 : Base de données de société « Tanger Soft »

### Travaux réalisés

- Q1 : Donnez la liste des employés ayant une commission (non NULL) classé par commission décroissante
- Q2 : Donnez les noms des personnes embauchées depuis le 01-09-2006
- Q3 : Donnez la liste des employés travaillant à « Tanger »
- Q4 : Donnez la liste des subordonnés de "Ali"
- Q5 : Donnez la moyenne des salaires
- Q6 : Donnez le nombre de commissions non NULL
- Q7 : Donnez la liste des employés gagnant plus que la moyenne des salaires de l'entreprise

**Question 1 : Donnez la liste des employés ayant une commission (non NULL) classé par commission décroissante**

```
Select Nom, Comm "Commission" from Emp where Comm IS NOT NULL AND Comm!=0 ORDER BY Comm DESC
```

**Question 2 : Donnez les noms des personnes embauchées depuis le 01-09-2006**

```
Select Nom, Embauche, N_Dept from Emp where Embauche > '01/10/2006'
```

**Question 3 : Donnez la liste des employés travaillant à Créteil**

```
Select Nom, Embauche, N_Dept from Emp, Dept where Emp.N_Dept=Dept.N_Dept and Lieu="Tanger"
```

**Question 4 : Donnez la liste des subordonnés de "Ali"**

```
Select a.Nom "Nom", Lieu from Emp a, Emp b where a.NumSup=b.NumSup  
and b.NumSup="Ali"
```

**Question 5 : Donnez la moyenne des salaires**

```
Select AVG(Salaire) « Moyenne des salaires » from Emp
```

**Question 6 : Donnez le nombre de commissions non NULL**

```
Select COUNT(Comm) « Nb. Commissions non-Null » from Emp where Comm  
IS NOT NULL
```

**Question 7 : Donnez la liste des employés gagnant plus que la moyenne des salaires de l'entreprise**

```
Select Nom, Fonction, Salaire from Emp where Salaire>(SELECT AVG(Salaire)  
from Emp)
```

## Partie 2 : Travaux Pratiques

### TP : Gestion des Vols

#### Modèle physique de données

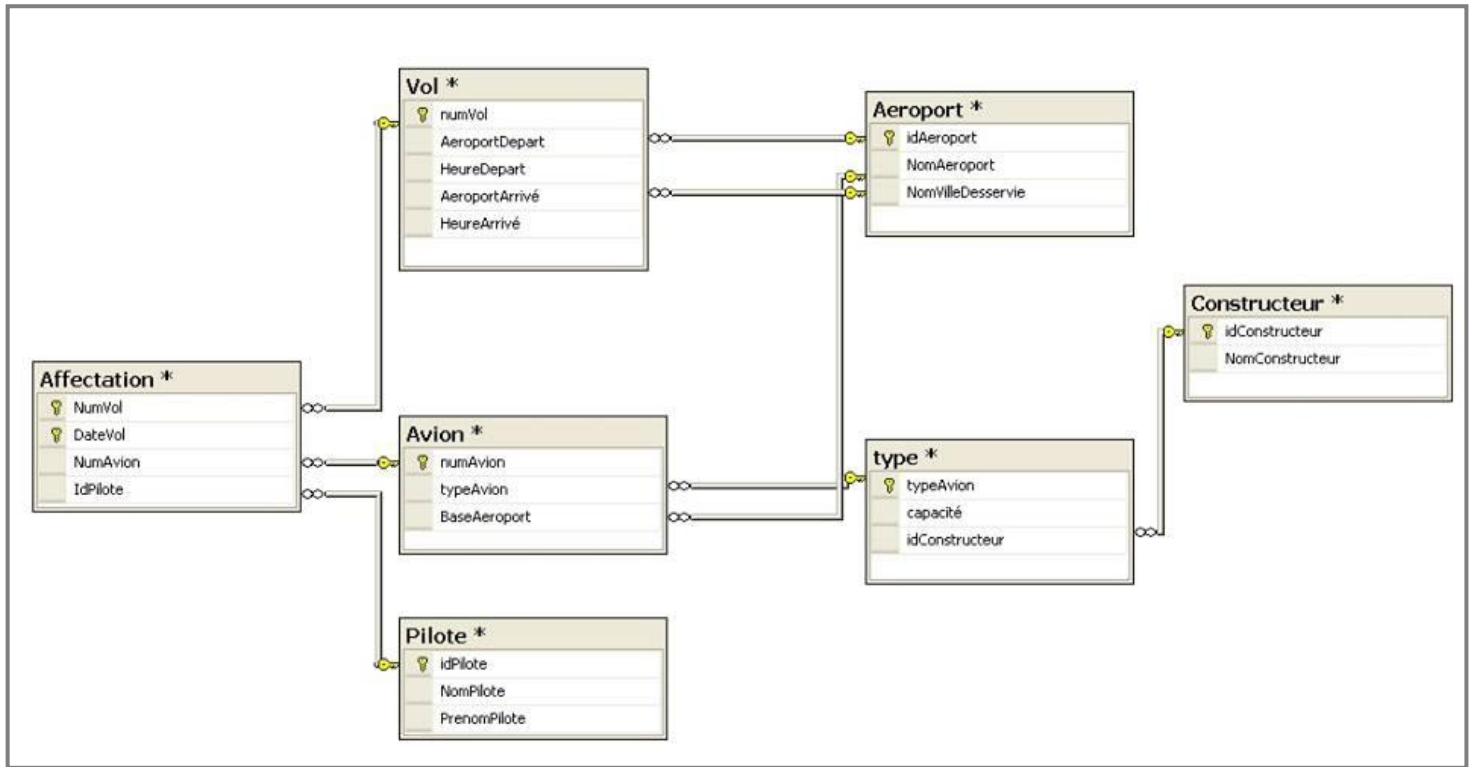


Figure 32 : Base de données de gestion des vols

#### Travaux réalisés

- Q1 : Créer les tables.
- Q2 : Ajouter le champ ville à la table pilote.
- Q3 : Donner le nom des pilotes planifiés pour des vols sur A320.
- Q4 : Donner le nom des pilotes planifiés pour des vols sur A320 qui habitent dans la ville de localisation d'un A320
- Q5 : Pour chaque ville desservie, donner la moyenne, le minimum, le maximum des capacités des avions qui sont localisés dans un aéroport Desservant cette ville

#### Question 1 : Créer les tables

Creation des tables

#### Question 2 : Ajouter le champ ville à la table pilote

```
Alter table pilote add ville varchar(20) ;
```

**Question 3 : Donner le nom des pilotes planifiés pour des vols sur A320**

```
select  nomPilote  from  pilote  p ,affectation  af ,avion  a  Where  
af.idPilote=p.idPilote  
And af.numAvion=a.numAvion And a.type='A320';
```

**Question 4 : Donner le nom des pilotes planifiés pour des vols sur A320 qui habitent dans la ville de localisation d'un A320**

```
Select distinct nomPilote from Pilote p, affectation af,avion av,aéroport a  
Where af.idPilote=p.idPilote And av.BaseAéroport=a.idAéroport  
And af.numAvion=av.numAvion And af.nomVilleDesservie=p.ville  
And av.typeAvion='A320';
```

**Question 5 : Pour chaque ville desservie, donner la moyenne, le minimum, le Maximum des capacités des avions qui sont localisés dans un aéroport Desservant cette ville**

```
select min(capacité) as minimum,max(capacité)as maximum,avg(capacité) as  
moyenne, a.nomVilleDesservie from aéroport a,avion av,type t  
Where a.idAéroport=av.baseAéroport  
And av.typeAvion=t.typeAvion  
Group by (nomVilleDesservie);
```



## Conclusion

L'objectif de filière TDI, technicien en développeur informatique, est de former des programmeurs capables de développer des applications de gestion simple et participer au développement d'applications de gestion interfacées à des bases de données plus complexes. Parmi les qualités requises dans le métier de développeur informatique nous citons : rigueur, méthode, autonomie, capacité à travailler en équipe, capacité d'adaptation, force de proposition

La réalisation de ce travail d'après notre encadrant avait comme objectif de développer chez nous la qualité à travailler en équipe. Ce rapport un fruit de synthèse de plusieurs rapports. Chaque groupe, TDI4 et TDI5, a été divisé en 5 sous-groupes de 6 stagiaires. D'abord chaque sous-groupe a réalisé son propre rapport. Ensuite les trois premiers groupes qui ont rendu leurs rapports sont chargés de synthétiser leurs rapports dans le présent document.

Ce document va nous permet de bien se préparer à l'examen de fin de module et l'examen de fin d'année. Parce qu'il résume dans la première partie : les notions de base du langage SQL avec les activités d'apprentissage des notions délicates pour les novices, comme : jointure, sous-interrogation, et « Groupe by » avec « Having », proposés par notre formateur. La deuxième et troisième parties regroupe les TD et TP avec une proposition de solution.

Comme perspectives de ce travail, nous voudrions que les stagiaires de l'année prochaine « incha allah » se concentrent sur la réalisation d'autres exemples explicatifs et Ajouter une autre partie sur les messages d'erreur rencontrée durant les séances de TP.