Prof : Idrissi

S.Q.L

Au travers de

S.Q.L Server

SOMMAIRE

SOMMAIRE

Les bases d'un langage structuré d'interrogation de bases de données sont apparues à l'occasion de la publication d'un article de Mr CHAMBERLIN et Mr BOYCE en 1974. Ce premier langage s'appelait SEQUEL.

SQUARE qui utilisait les expressions mathématiques vît le jour en 1975.

SQL apparaît en 1980 et continue à couler des jours heureux sur gros systèmes et micro-ordinateurs. Eh, oui ! ce que vous découvrez aujourd'hui fonctionne aussi sur les 30XX d'IBM

De nombreux fournisseurs de logiciels proposent l'utilisation de ce langage.

Parmi les principaux S.G.B.D.R. possédant l'interface S.Q.L.; nous pouvons citer :

- ACCESS (sous Dos et Windows)
- SQL Server sous Windows)
- ORACLE (sous MS/DOS; MVS, ...)
- DATABASE MANAGER (sous OS/2)
- DB2/400 (sur AS/400)
- SQL/DS (sous DOS/VSE, VM/CMS)
- DB2 (sous MVS)
- ...

Le SQL standard a été défini par ANSI (American National Standard Institute) C'est un langage d'interrogation de bases de données relationnelles utilisant le concept de vues.
Que nous faut-il découvrir pour être un familier de SQL ?

Tout d'abord, la notion de base de données relationnelle et les concepts
qui lui sont accrochés,

Puis comment la générer ?

comment la mettre à jour ?.

et enfin comment en extraire les informations : à quoi sert une base de données si ce n'est de pouvoir sélectionner selon votre bon vouloir
(ou plutôt la volonté de l'utilisateur) ce qu'elle contient ?

1 Concepts base de données relationnelles

1.1 Domaine, attribut & relation

Tout au long de ce chapitre, nous imagerons notre discours par un cas dont le thème sera le cinéma.

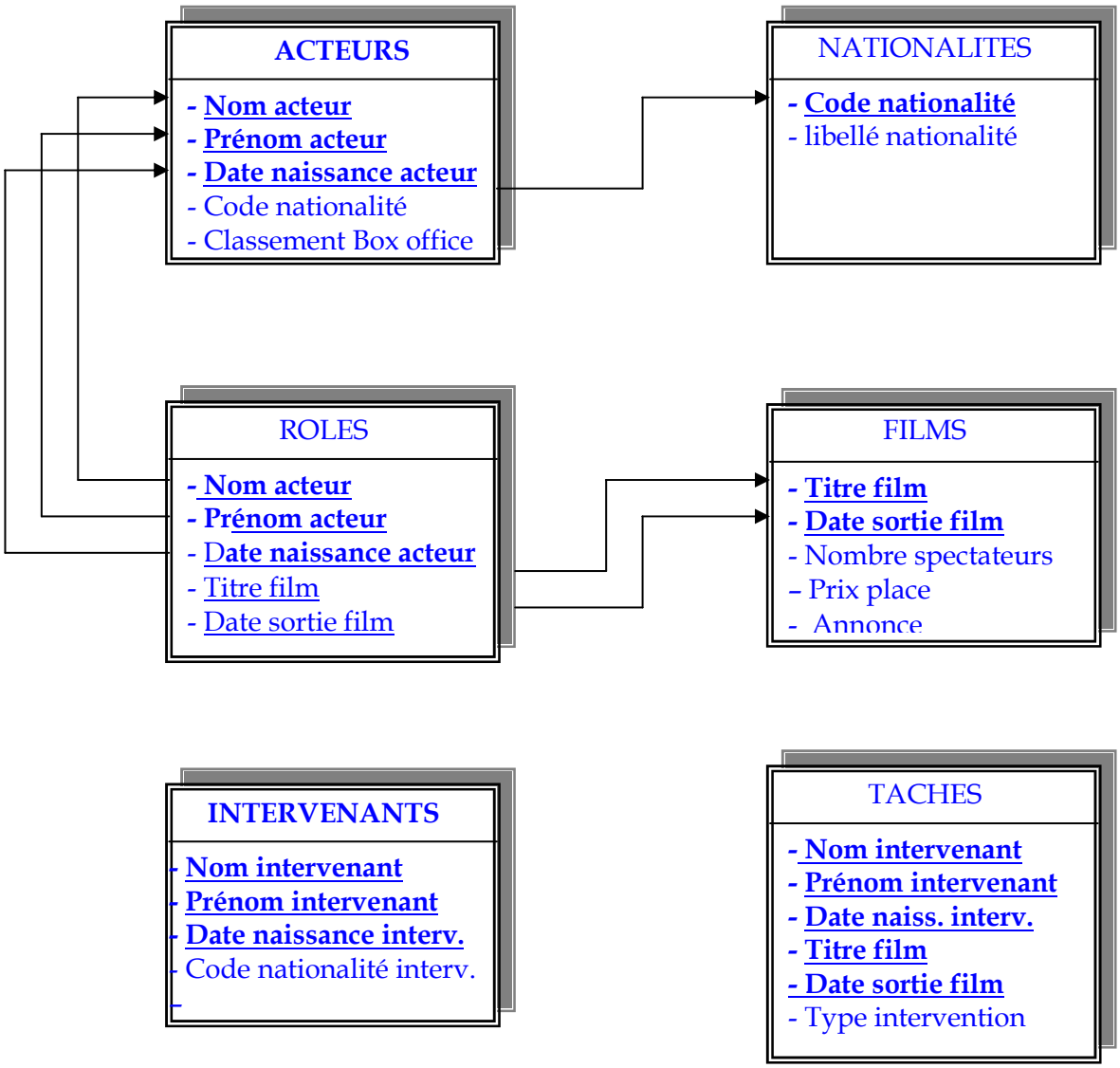
Nous allons créer et faire vivre une « base de données » que nous nommerons « CINEMA » et qui va contenir un certain nombre d'informations sur les acteurs, leurs nationalité, les films dans lesquels ils ont joué, et les personnes « non acteurs » qui ont participé à l'élaboration des films (type d'intervention : metteur en scène, habilleuse, script ; ....)

Certaines données qualifient l'acteur : son nom , son prénom, sa date de naissance (chut, faut pas le dire) et sa nationalité.
D'autres qualifient ou caractérisent le film : le titre, la date de sortie, le nombre de spectateurs l'ayant déjà vu.

Par contre, il faut indiquer la participation de l'acteur à un film : ceci caractérise un lien entre un acteur et un film.

Nous pouvons représenter ces informations avec la structure suivante

Les flèches mettent en évidence des informations que l'on retrouve d'une table à l'autre :



Quelles sont les flèches qui manquent ?

Exemple de requête que nous pourrons faire avec SQL : retrouver tous les films ayant eu des acteurs de nationalité «française».

Tout d'abord, on recherche le code de la nationalité dont le libellé est «française» soit "033".

Puis, on recherche dans la table acteurs, les acteurs ayant pour code nationalité «033»

Ensuite par la consultation de ROLES, on recherche tous les films de cet acteur
On répète ceci pour chaque acteur sélectionné

La base de données relationnelle CINEMAest constituée de tables:

- Acteurs
- Rôles
- Nationalités
- Films
- Intervenants
- Taches

Cette base de données est «relationnelle» car des éléments permettent de passer d'une table à l'autre: (on appelle souvent cela la navigation dans la base de données).

Dans la table ACTEURS, nous avons des colonnes: nom prénom date de naissance code nationalité.
Dans cette même table, on trouvera une ligne (enregistrement) par acteur:

Une table est donc un ensemble de lignes et de colonnes.

Il est donc possible de représenter une table sous forme de tableau avec un contenu :

ACTEURS	Nom	Prénom	Date Nais.	Cod Nat.é	Classement
	Béart	**Emmanuelle**	**15/01/1960**	**033**	**250**
	Blanc	**Michel**	**20/04/1952**	**033**	**560**
	Douglas	**Kirk**	**12/01/1928**	**100**	**50**
	Douglas	**Michael**	**16/05/1955**	**100**	**3**
	Durand	**Al**	**01/04/1915**	**033**	

Maintenant, nous allons introduire une notion nouvelle: le domaine

C'est l'ensemble des valeurs caractérisant un nom de donnée.
Dans notre exemple, si nous regardons le code nationalité, il prend des valeurs entières (symbolisées par n) comprises entre 001 et 999, 033 étant la valeur pour la France.

Traduit en modèle relationnel, nous écrirons :

Code nationalité = (n | 1<= n <= 999);

On parle aussi de schéma de la relation Acteurs ou extension de la relation

Acteurs(Nom, domaine de nom; Prénom, domaine de prénom, …….)

Proposition : M. Durand Al est un acteur de nationalité française, né le 01/04/1915 et n'a jamais été classé au box office.

Une proposition vraie est aussi appelée assertion.

nom prénom date de naissance code nationalité classement sont les 5 variables de cette phrase type.

On appelle prédicat la phrase type à laquelle se conforment les propositions qui en sont issues par la valorisation des variables.

Exemple : la variable Nom est valorisée par Durand
la variable date de naissance est valorisée par 01/04/1915.

Le prédicat écrit de manière concise devient :

Acteur(nom prénom date de naissance code nationalité classement)

Les variables du prédicat sont appelées : Attributs de la relation

Les éléments de la relation sont appelés U-PLETs ou TUPLE(in english).

Le degré d'une relation est le nombre de ses attributs.

ACTEURS est une relation de degré 5

Une relation unaire est une relation à 1 degré.
Une relation binaire est une relation à 2 degrés.
Une relation ternaire est une relation à 3 degrés.

On appelle cardinalité le nombre de N-UPLETs (tuple) d'une relation.

<-------- Degré = 5 -->

Attribut	**Attribut**	**Attribut**	**Attribut**	**Attribut**
Nom	**Prénom**	**Date Naissance**	**Code National.**	**CltBox Office**
Béart	**Emmanuelle**	**15/01/1960**	**033**	**250**
Blanc	**Michel**	**20/04/1952**	**033**	**560**
Douglas	**Kirk**	**12/01/1928**	**100**	**50**
Douglas	**Michael**	**16/05/1955**	**100**	**3**
Durand	**AL**	**01/04/1915**	**033**	

U-PLET -->

1.2 La notion de contraintes

Cohérence : toute valeur prise par une colonne doit appartenir à son domaine de définition.

Atomicité : une colonne ne doit pas pouvoir se décomposer.

Unicité : chaque ligne d'une table doit être unique afin de pouvoir l'isoler.

Clef primaire ou **identifiant** pour pouvoir identifier une ligne parmi toutes les lignes de la tables.
Quel est le critère qui identifie de façon unique une ligne de la table ACTEURS ? son nom ne suffit pas: par exemple, on trouvera DOUGLAS au moins deux fois papa KIRK et fiston MICHAEL donc le nom n'est pas suffisant pour pointer sur une ligne et une seule. Prenons maintenant comme critères d'identification nom et prénom: il existe des familles d'acteurs et rien n'empêche que le père et le fils n'ait le même prénom; il faut donc étendre le critère d'identification à nom, prénom, date de naissance
Ces trois colonnes constituent l'identifiant de la table ACTEURS**.**

Pour NATIONALITE, on a affaire à une **clé primaire simple** (car constituée d'une seule zone ou colonne). Pour ACTEUR, on a affaire à une **clé primaire composite** (car constituée de plusieurs zones ou colonnes).

Clé secondaire : ce sont éventuellement des colonnes ou groupe de colonnes qui permettent d'identifier aussi une ligne de la table.

Pour la nationalité, le code est suffisant pour identifier une nationalité. Le libellé aussi me direz-vous! Je vous l'accorde, mais en informatique, on préfère la zone la plus courte. donc l'identifiant sera le code et le libellé sera une clé secondaire.

Clé étrangère :
Le code nationalité qui se trouve dans ACTEUR est la clé primaire de NATIONALITE: le code nationalité dans ACTEUR sera appelé clé étrangère.

Lorsqu'on crée une table, il est possible d'indiquer ces contraintes au niveau des zones: c'est le moteur de la base de données[1] qui effectuera les contrôles à chaque action sur la table sans que nous, les développeurs, nous ayons à nous en préoccuper.

[1] C'est le noyau qui traite les ordres dans un SGBD/r système de gestion de base de données relationnelle

1.3 Quelques opérations relationnelles

1.3.1 Restriction ou Sélection :

Elle permet de sélectionner les lignes répondant à une condition de sélection

SeptNains	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	Prof	2	F	15000,00
	35	Atchoum	5	M	14000,00
	20	Timide	5	M	15500,00
	30	Simplet	7	M	13000,00
	25	Dormeur	8	F	14500,00

Sélection des lignes où numéro est compris entre 20 et 30

Résultat

⬇

SeptNains	Numéro	Nom	Classif.	Sexe	Salaire brut
	20	Timide	5	M	15500,00
	30	Simplet	7	m	13000,00
	25	Dormeur	8	F	14500,00

1.3.2 Projection

Elle permet de sélectionner les colonnes d'une table

SeptNains	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	Prof	2	F	15000,00
	35	Atchoum	5	M	14000,00
	20	Timide	5	M	15500,00
	30	Simplet	7	m	13000,00
	25	Dormeur	8	F	14500,00

Projection de numéro, nom, classif.

Résultat

SeptNains	Numéro	Nom	Classif.
	10	Prof	2
	35	Atchoum	5
	20	Timide	5
	30	Simplet	7
	25	Dormeur	8

1.3.3 Jointure

Elle permet de joindre 2 tables ou plus en indiquant des critères de jointure

A	N°	Nom	Classif.	Sexe	Salaire brut
	10	Prof	2	F	15000,00
	35	Atchoum	5	M	14000,00
	20	Timide	5	M	15500,00
	30	Simplet	7	M	13000,00
	25	Dormeur	8	F	14500,00

B

B	Classif	Libellé
	2	Comptable
	5	Secrétaire

Joindre les tables A et B sur critère classif

Résultat

⬇

C	N°	Nom	Classif.	Sexe	Salaire brut	Classif	Libellé
	10	Prof	2	F	15000,00	2	Comptable
	35	Atchoum	5	M	14000,00	5	Secrétaire
	20	Timide	5	M	15500,00	5	Secrétaire

Chaque ligne de la table A a été complétée par les données de la table B.

Prenons la première ligne de A : le code classification est 2, les valeurs ajoutées sont 2 et comptable qui correspondent à la ligne de B dont le code classif est également 2.

1.3.4 Produit cartésien

Il permet de joindre 2 tables en combinant les lignes de l'une avec les lignes de l'autre

A	N°	Nom	Classif.	Sexe	Salaire brut
	10	Prof	2	F	15000,00
	35	Atchoum	5	M	14000,00
	20	Timide	5	M	15500,00
	30	Simplet	7	M	13000,00
	25	Dormeur	8	F	14500,00

B	Classif	Libellé
	2	Comptable
	5	Secrétaire

Faire le produit cartésien des tables A et B

Résultat

⬇

P	N°	Nom	Classif.	Sexe	Salaire brut	Classif	Libellé
	10	Prof	2	F	15000,00	2	Comptable
	35	Atchoum	5	M	14000,00	2	Comptable
	20	Timide	5	M	15500,00	2	Comptable
	30	Simplet	7	M	13000,00	2	Comptable
	25	Dormeur	8	F	14500,00	2	Comptable
	10	Prof	2	F	15000,00	5	Secrétaire
	35	Atchoum	5	M	14000,00	5	Secrétaire
	20	Timide	5	M	15500,00	5	Secrétaire
	30	Simplet	7	M	13000,00	5	Secrétaire
	25	Dormeur	8	F	14500,00	5	Secrétaire

1.3.5 Union

Elle permet de mettre des lignes d'une table avec les lignes d'une autre table

A	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	Prof	2	F	15000,00
	35	Atchoum	5	M	14000,00

B	Numéro	Nom	Classif.	Sexe	Salaire brut
	20	Timide	5	M	15500,00
	30	Simplet	7	m	13000,00
	25	Dormeur	8	F	14500,00

Union des tables A et B

Résultat

	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	Prof	2	F	15000,00
	35	Atchoum	5	M	14000,00
	20	Timide	5	M	15500,00
	30	Simplet	7	m	13000,00
	25	Dormeur	8	F	14500,00

1.3.6 Différence

Elle permet d'extraire des lignes d'une table si elles n'ont pas de correspondance avec les lignes d'une autre table

A	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	Prof	2	F	15000,00
	20	Timide	5	M	15500,00
	30	Simplet	7	m	13000,00
	35	Atchoum	5	M	14000,00

B	Numéro	Nom	Classif.	Sexe	Salaire brut
	20	Timide	5	M	15500,00
	30	Simplet	7	m	13000,00
	25	Dormeur	8	F	14500,00

Différence entre les tables A et B

Résultat

⬇

D	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	Prof	2	F	15000,00
	35	Atchoum	5	M	14000,00

Un peu de mise en pratique:

À partir de la base "Cinéma" décrite dans le chapitre précédant, indiquer quelles sont les opérations à effectuer sur les tables pour répondre aux questions suivantes :

- ✓ Quels sont les acteurs de nationalité française.?
- ✓ Quels acteurs ont été aussi metteurs en scène ?
- ✓ Quels sont les films ayant eu un metteur en scène de nationalité "hongroise" ?

2 *Le langage S.Q.L & ses composantes*

Le langage S.Q.L & ses composantes

Pour le dictionnaire, le langage est la manière de s'exprimer au moyen de symboles.

Parmi les symboles de SQL, nous trouvons des verbes qui peuvent se classer en quatre chapitres :

- **DDL** Data **Définition** Language
  permet de définir une base de données et ses éléments.

- DML Data **Management** Language
  permet de gérer, manipuler les éléments d'une base de données.

- DCL Data **Control** language
  permet d'effectuer des contrôles sur les données.

- **QUERY** permet d'**interroger** une base de données en les sélectionnant.

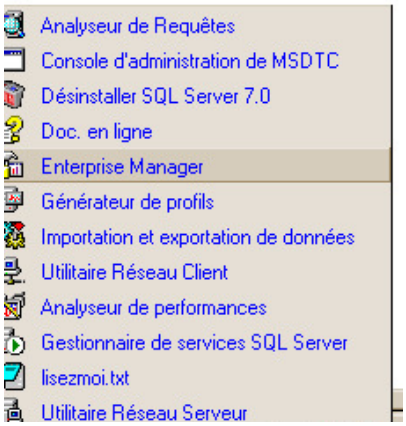
Autre particularité de ce langage : ses mots clés. Ils ne peuvent pas être utilisés comme nom de base de données, de tables, de données ou de variables. Mais à quoi correspondent tous ces termes, allez-vous me dire ? Nous allons vous l'expliquer dans les chapitres qui suivent. Mais tout d'abord, pour ceux qui n'ont pas ou peu d'anglais à leur actif, la page qui suit va décoder quelques mots revenant régulièrement chez ceux qui causent SQL dans le texte.

3 *Découverte de SqlServer*

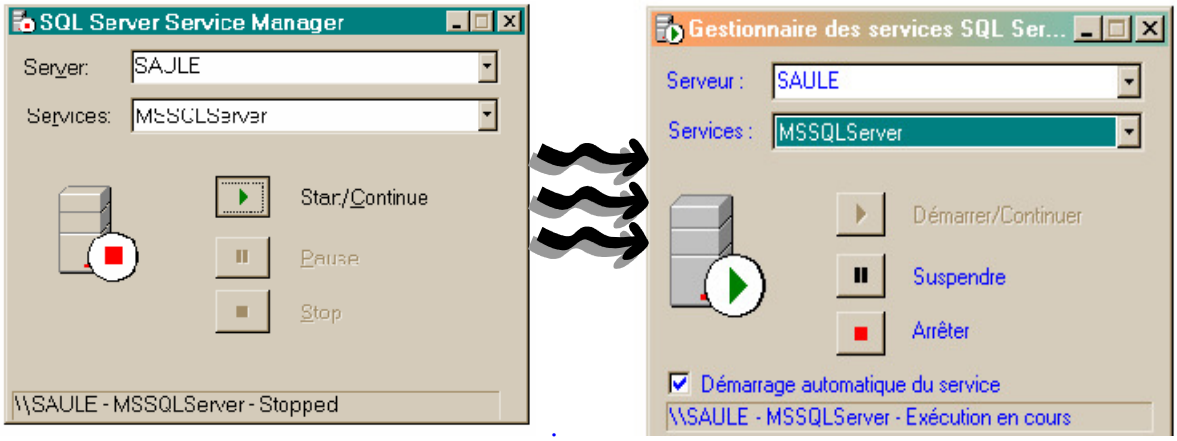
Dans un premier temps, nous allons vous présenter l'outil SqlServer. Une fois installé, le produit est accessible par

- Cliquer sur menu "Démarrer" de Windows
- Cliquer sur menu "Programmes"
- Cliquer sur menu Microsoft SQL Server

Le menu suivant apparaît :



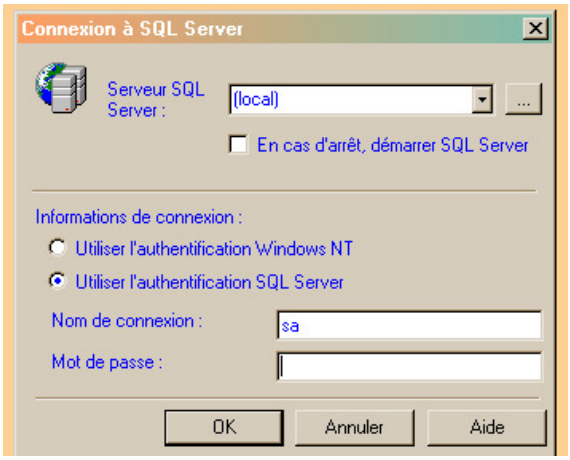
Assurez-vous tout d'abord que SQL Server est démarré : pour cela cliquez sur l'option "Gestionnaire de service SQL Server" Un carré rouge sur le symbole du serveur indique qu'il n'est pas démarré; un triangle vert indique au contraire qu'il est démarré.
Pour démarrer un serveur, cliquez son nom dans la liste des serveurs possibles : ici nous avons cliqué sur Saule. Choisissez MSSQLServer parmi les services possibles puis cliquez sur le symbole triangle vert pour démarrer ce service sur le serveur indiqué.



Pour démarrer Query analyser que nous allons utiliser tout au long des trois chapitres qui suivent :

- Cliquer sur menu "Démarrer" de Windows,
- Cliquer sur menu "Programmes",
- Cliquer sur menu Microsoft SQL Server,
- Cliquer sur "Analyseur de requêtes"(Query Analyser).

Si votre connexion est sécurisée, l'écran suivant apparaît :



Choisissez une authentification SQL Server et indiquez le mot de passe correspondant au profil de la connexion; Puis validez par OK.

Les commandes SQL sont à taper dans la partie haute de l'écran.
Une fois la commande complètement tapée, cliquer sur "Execute Query" (représenté par un triangle vert dans la barre d'outils
Soit des erreurs vous sont signalées ➔ il faut alors se déguiser en Sherlock Holmes et corriger l'erreur.
Ou un message de bon fonctionnement apparaît et il est possible de passer à la suite.

Nous reviendrons sur ces aspects lors du détail des instructions dans les chapitres suivants.

4 Définition d une base de données relationnelle

4.1 Création d'une base de données : CREATE DATABASE

On crée une base de données par la commande CREATE DATABASE.
Dans la base de données seront enregistés :

- le catalogue de la base de données,
- les objets utilisateurs (tables, vues, règles, triggers, procédures, …),
- les index, contraintes et types de données
- le journal des transactions

Les tables, vues, règles … sont créées dans un deuxième temps par d' autres commandes

Structure de la commande si vous la saisissez directement sur l'écran QUERY en ayant sélectionner Master comme database dans la barre d'outils.

```
CREATE DATABASE   <nom base de données> ON DEFAULT    = 1
```

Le nom de la base de données doit être valide c'est-à-dire commençant par une lettre et ne pas comporter de caractères spéciaux sauf '_'.

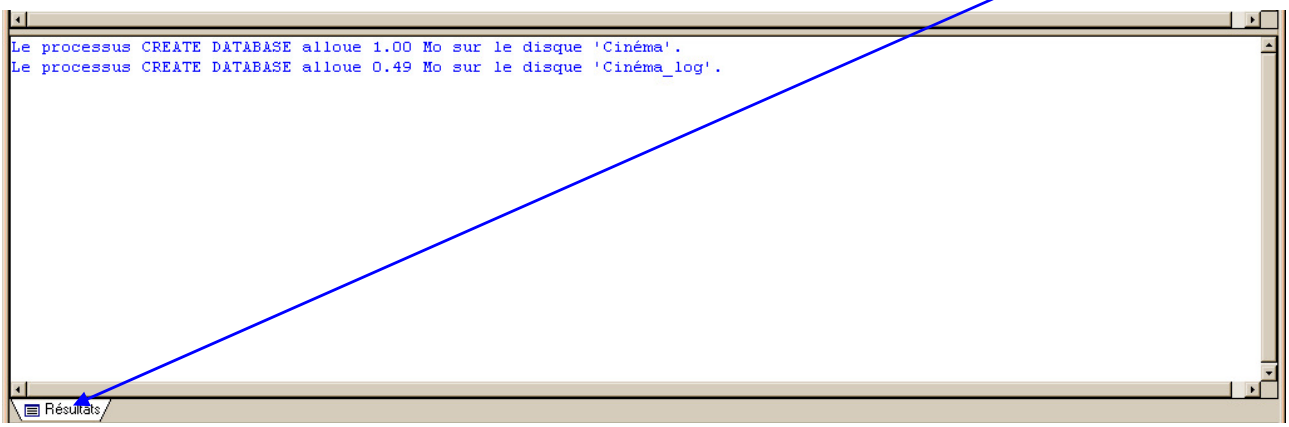
Attention le temps d'exécution de cette commande est relativement long.

Exemple de commande passée dans l'analyseur de requêtes

Une fois la commande tapée, vous avez accès à des outils jusqu'à alors inaccessibles :

- Le ✓ qui permet la vérification de la syntaxe des lignes d'instructions saisies
- Le ▶ qui permet la vérification de la syntaxe des lignes d'instructions saisies puis leur exécution s'il n'y a pas d'erreurs

Suite à une exécution sans problème, des messages apparaissent dans une fenêtre "résultats" en bas d'écran.



Un Create Database ne se fait qu'une fois; si vous le faites une deuxième fois, des erreurs vous seront affichées dans la fenêtre résultats.

4.2 Suppression d'une base de données DROP DATABASE

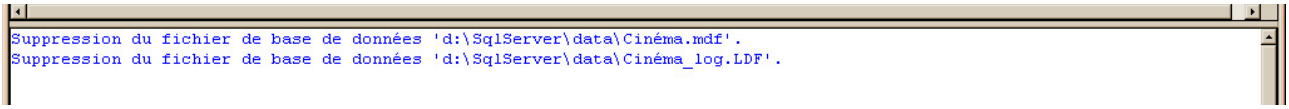
Pour supprimer une base de données, il faut impérativement qu'aucun utilisateur ne soit en train de l'utiliser.

Les données de chaque table de la base de données sont supprimées et irrécupérables.

Structure de la commande si vous la saisissez directement sur l'écran QUERY

```
DROP DATABASE   <nom base de données>
```

Suite à une exécution sans problème, des messages apparaissent dans une fenêtre "résultats" en bas d'écran.



4.3 Modification d'une base de données ALTER DATABASE

Pour modifier les attributs de définition et de taille d'une base de données, il faut impérativement qu'aucun utilisateur ne soit en train de l'utiliser.

```
ALTER DATABASE <nom base de données>
        ADD FILEGROUP <nomfichier1> …….
                ……
        REMOVE FILE <nomfichier2>
                ……
```

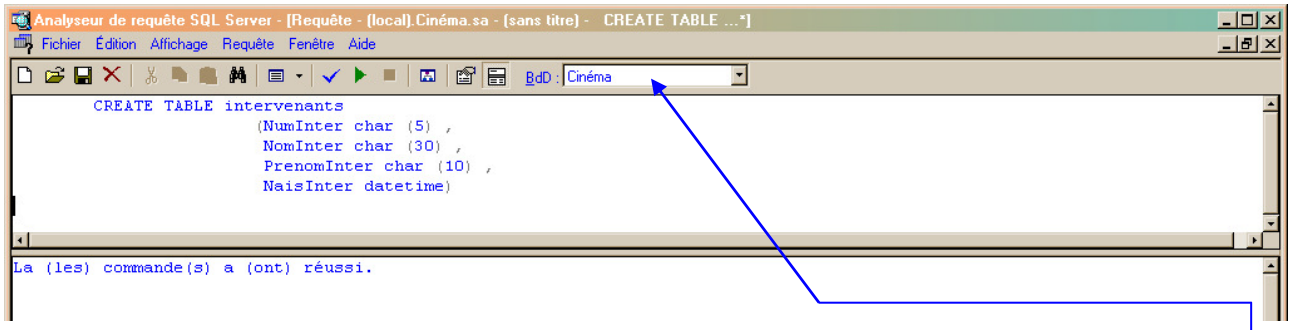
4.4 Création de table CREATE TABLE

La création d'une table s'effectue grâce à la commande CREATE TABLE.

Structure de la commande simple

```
CREATE TABLE nomtable
        (Nomcol1  typecol1 (Lg1 dec1) ,
        Nomcol2  typecol2 (Lg2 dec2) ,
              (       (       (       (
              )       )       )       )
        Nomcol3  typecol3 (Lg3 dec3) )
```

Exemple de commande simple



Nous n'avons pas encore examiné tous les éléments de la barre d'outils; il en est un très important : la base de données à utiliser. Nous voulons créer une table dans notre base de données Cinéma. Il faut que ce nom apparaisse comme base courante.

Pour faire apparaître ce nom, cliquez sur le symbole ▼ se trouvant derrière "Master" puis cliquez le nom souhaité dans la liste qui apparaît. Si celui-ci n'apparaît pas, cliquez sur"actualiser" puis sur le nom de base souhaitée.

Quelles sont les valeurs à indiquer ?

Nom de la table: tapez un nom valide en respectant les standards préalablement définis (maxi 116 caractères). Ce nom ne doit pas être celui d'un objet existant dans la base.

Colonne (Zone) indiquez les noms des colonnes (zones) de la table. Le nom de la colonne ne doit pas se répéter dans une même table. Vous pouvez définir jusqu'à 8000 colonnes par table.

Type Choisissez le type des données et indiquez éventuellement la longueur si le type le nécessite.

Quelles sont les types de données possibles avec SQL Server ?

	Type	Désignation	Lg	Dec	Exemple
	bit	Valeur entière à 0 ou 1			
	char	chaîne de caractères longueur fixe, de 1 à 8000			**char (30)**
	datetime	Donnée Date & heure du 1 janvier 1753 au 6 juin 9999			
	decimal	Nombre décimal avec position après la virgule			**decimal (11 2)**
	float	Nombre flottant			
	image	Zone en binaire de longueur variable jusqu'à 2 Giga			
	int	Nombre entier allant de -2^{31} (-2,147,483,648) à 2^{31} - (2,147,483,647).			
	money	Valeur monétaire allant de-2^63 (-922,337,203,685,477.5808) à 2^63			
	nchar	chaîne de caractères de longueur fixe, de 1 à 4000			
	ntext	Texte non unicode			
	numeric	Idem décimal			
	nvarchar	chaîne de caractères de longueur variable de 1 à 4000			
	real	Nombre flottant			
	smalldatetime	Donnée Date & heure du 1 janvier1900, au 6 juin 2079			
	smallint	Petit nombre entier allant de -2^{15} (-32,768) à 2^{15} - (32,768)			
	smallmoney	Valeur monétaire allant de- - 214,748.3647 à - +214,748.3647			
	tinyint	Entier de 0 à 255			
	text	chaîne de caractères de longueur variable et de très grande taille			
	timestamp	Désigne un nombre entier élevé.			
	uniqueidentifier	N° automatique			
	varbinary	Valeur binaire de longueur variable			
	varchar	chaîne de caractères de longueur variable			

Consulter l'aide de SQL Serveur pour plus de détails.

Valeurs indéfinies

L'attribut de valeur indéfinie associé à la colonne indiquée peut être :

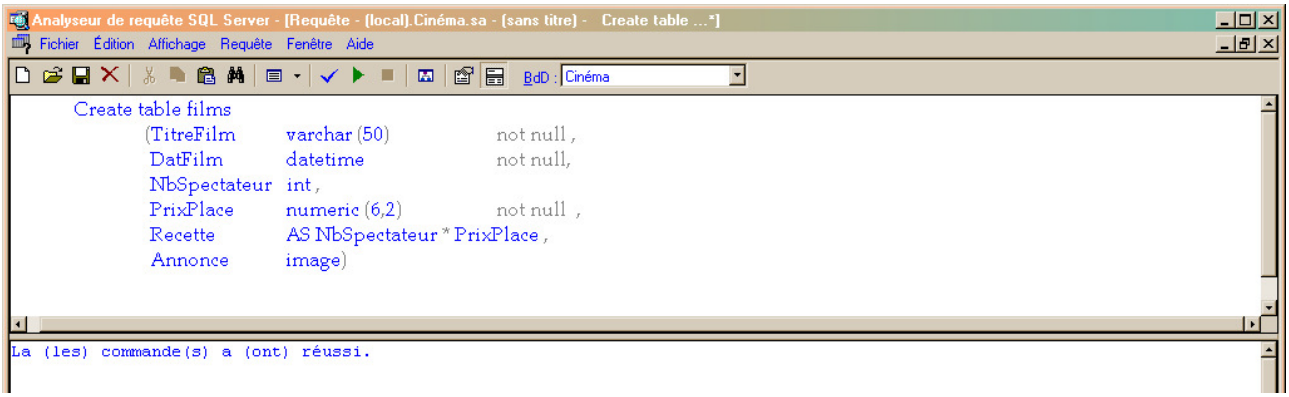
- NULL La colonne (zone) indiquée admet de ne pas avoir de valeurs.
- NOT NULL La colonne indiquée doit forcément contenir une valeur .

Valeur par défaut

Elle dépend du type de données

Type de données	Défaut
CHAR	Blancs
VARCHAR	Chaîne vide
DATE	Date en cours
TIME	Heure en cours
TIMESTAMP	Horodatage en cours
NUMERIC	0
DECIMAL	0

Exemple de commande avec colonne calculée



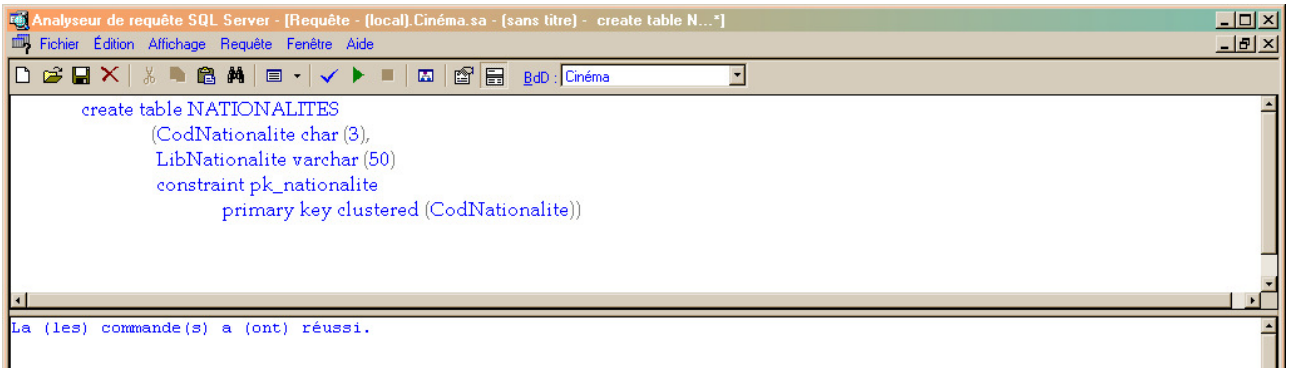
Nous avons utiliser la notion de NOT NULL sur les colonnes qui doivent nécessairement contenir une valeur définie;

De plus, nous avons une colonne virtuelle nommée "recette" qui est un résultat de calcul entre deux autres colonnes de la même table.

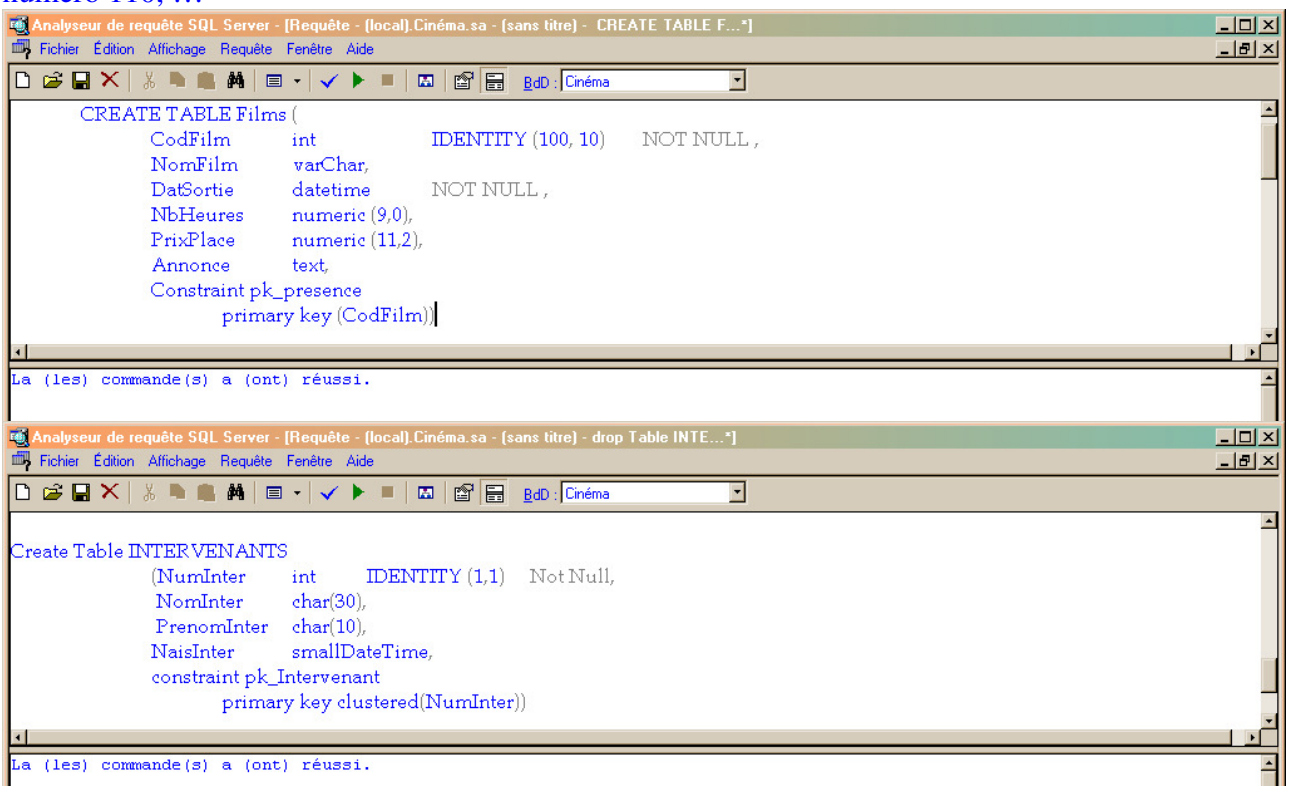
4.4.1 Contrainte Primary key

Elle permet d'indiquer dans une table une colonne (zone) ou un ensemble de colonnes qui doit contenir des valeurs identifiant de façon unique et certaine une ligne de la table.

Exemple : Dans une table SALARIES le numéro matricule est un identifiant permettant de repérer un et un seul salarié. Ce sera donc la "primary key" de la table SALARIES.
Cette colonne doit être définie impérativement **not null**.
Une table ne peut posséder qu'une seule clé primaire donc une seule contrainte primary par table.

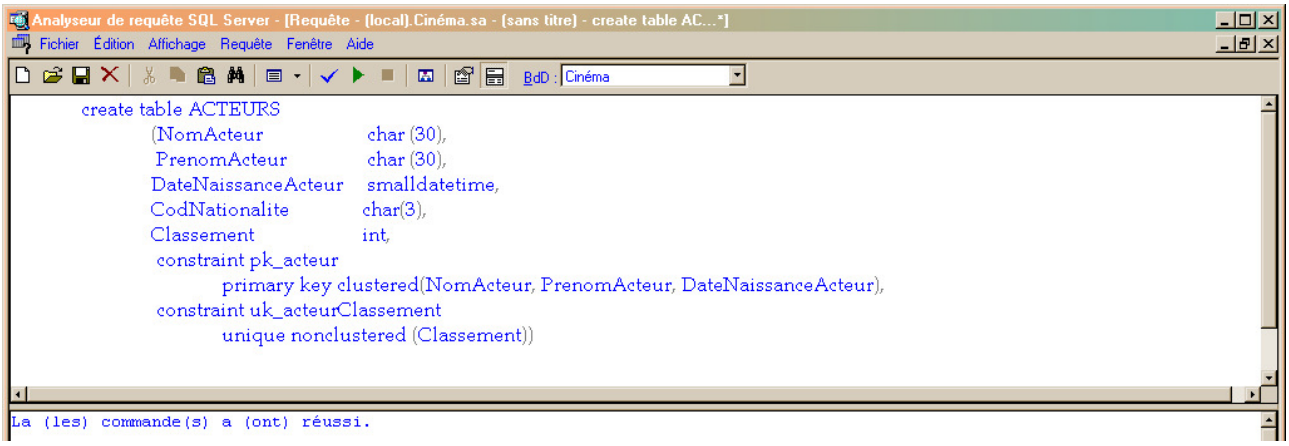


Autre exemple : la clé primaire est de type Identity avec une valeur initiale de 100 et un incrément de 10.Cela permet à chaque création de films dans la base de données d'attribuer automatiquement un numéro; le premier film créé se verra attribué le numéro 100, le second le numéro 110, …



4.4.2 Contrainte Unique

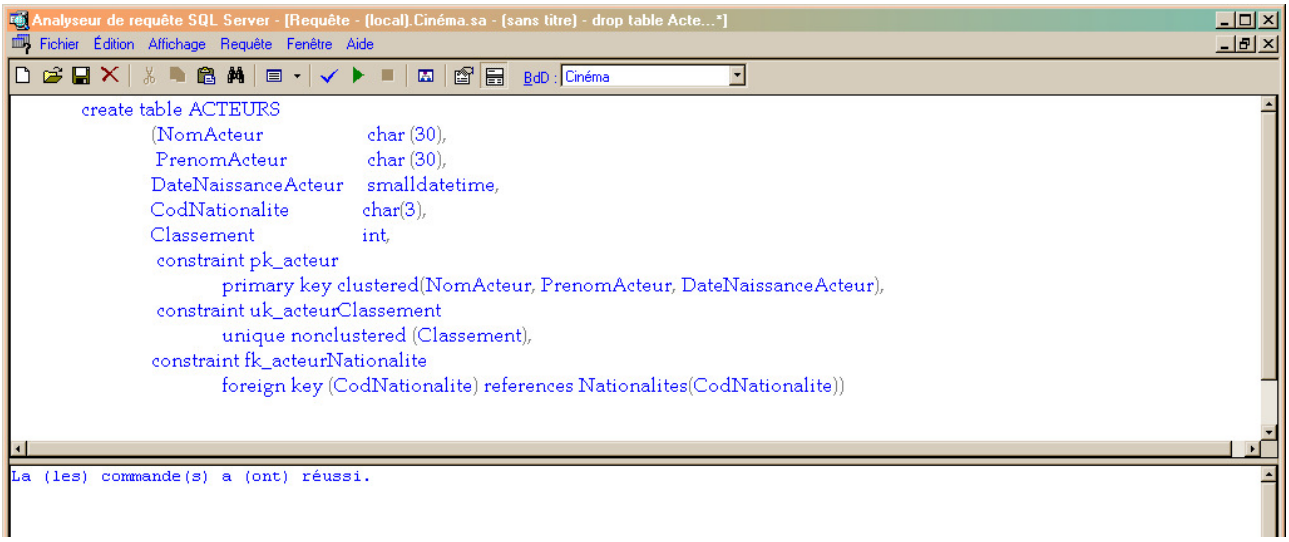
désigne dans une table une colonne (zone) ou un ensemble de colonnes (de zones) qui doit contenir des valeurs ne se retrouvant pas dans une autre ligne de la même table.
Plusieurs contraintes unique sont possibles sur une même table.



4.4.3 Contrainte Foreign key

désigne une clé étrangère qui se trouve dans une autre table en clé primaire; le nom de l'autre table est indiqué derrière références

Si les noms de colonne diffèrent d'une table à l'autre, il faut indiquer le nom de la colonne de la table suivi de références suivi entre parenthèses du nom de la colonne dans l'autre table (là où elle est primaire ou secondaire).
Plusieurs contraintes foreign key par table sont possibles.



Le nom des contraintes uk_acteur, pk_acteur, fk_acteur sont facultatives. SQL Server indique

lui-même des noms mais qui ne sont pas très clairs d' où des difficultés possibles lors de la maintenance de la base.
Un nom de contrainte doit être unique à l'intérieur d'une base de données.

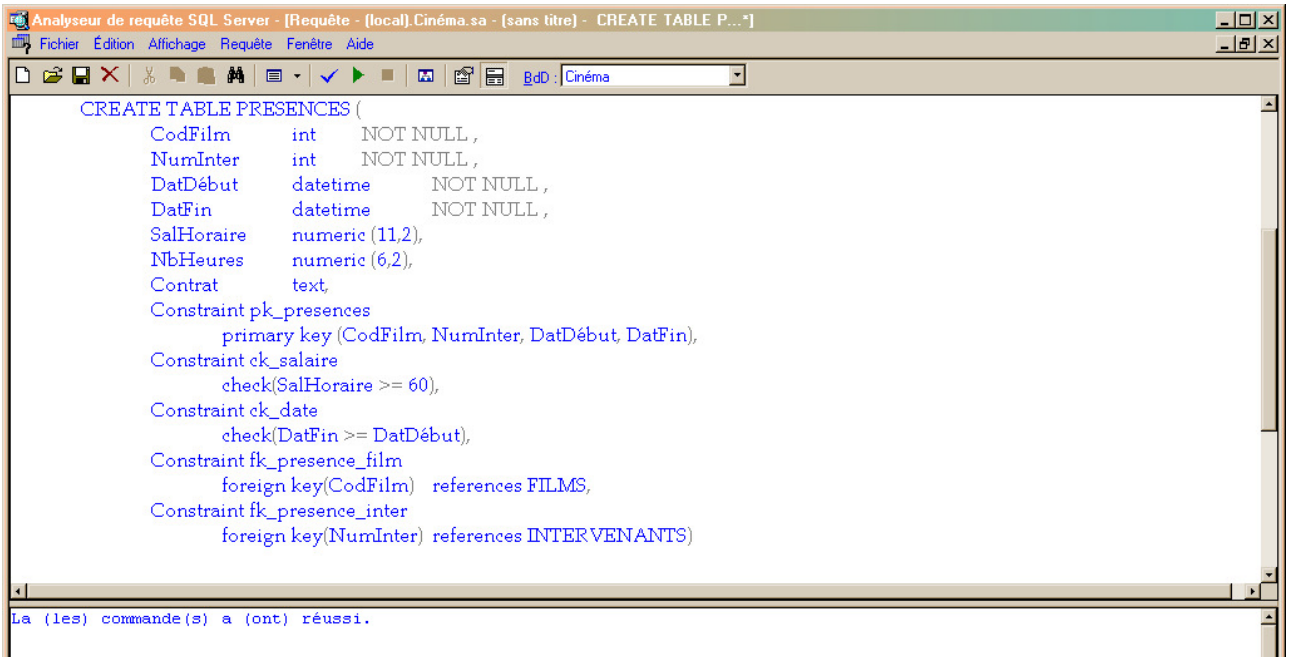
4.4.4 Contrainte Chech

Cette contrainte permet d'introduire des contrôles sur les zones de la table :

- contrôle de validité par rapport à des constantes ou des listes de constantes
- Contrôle de cohérence entre deux colonnes de la table

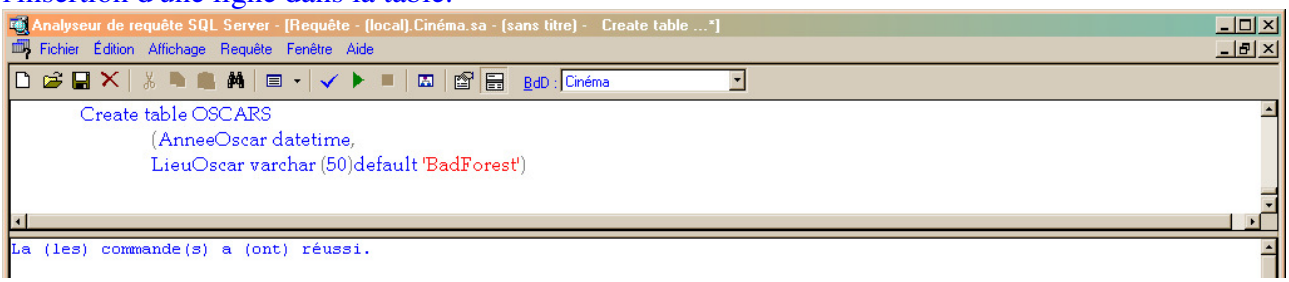
Dans l'exemple qui suit :

- Contrôle que salaire est plus grand ou égal à 60
- Contrôle que la date de début est inférieure ou égale à la date de fin.



4.4.5 Valeur par défaut : default

Cette notion peut être entrée par contrainte ou par mot clé derrière la définition de la colonne. Cette contrainte permet d'indiquer le contenu d'une zone quand elle n'est pas indiquée lors de l'insertion d'une ligne dans la table.



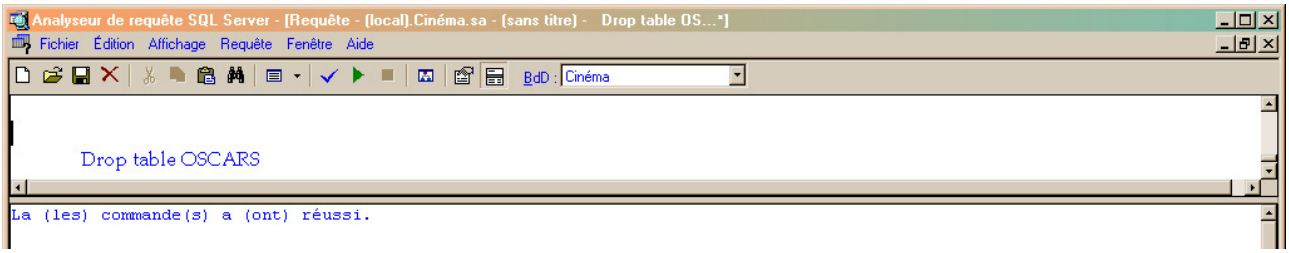
4.5 Suppression d'une table DROP TABLE

Pour supprimer une table et tous les objets en dépendant , il faut utiliser la commande DROP TABLE.

Structure de la commande

```
DROP  TABLE  <nom table>
```

Exemple de commande



Dès lors, les données de la table sont effacées et irrécupérables. (sinon par restauration depuis sauvegarde).

Bien sûr, nous ne pouvons que supprimer une table existante.

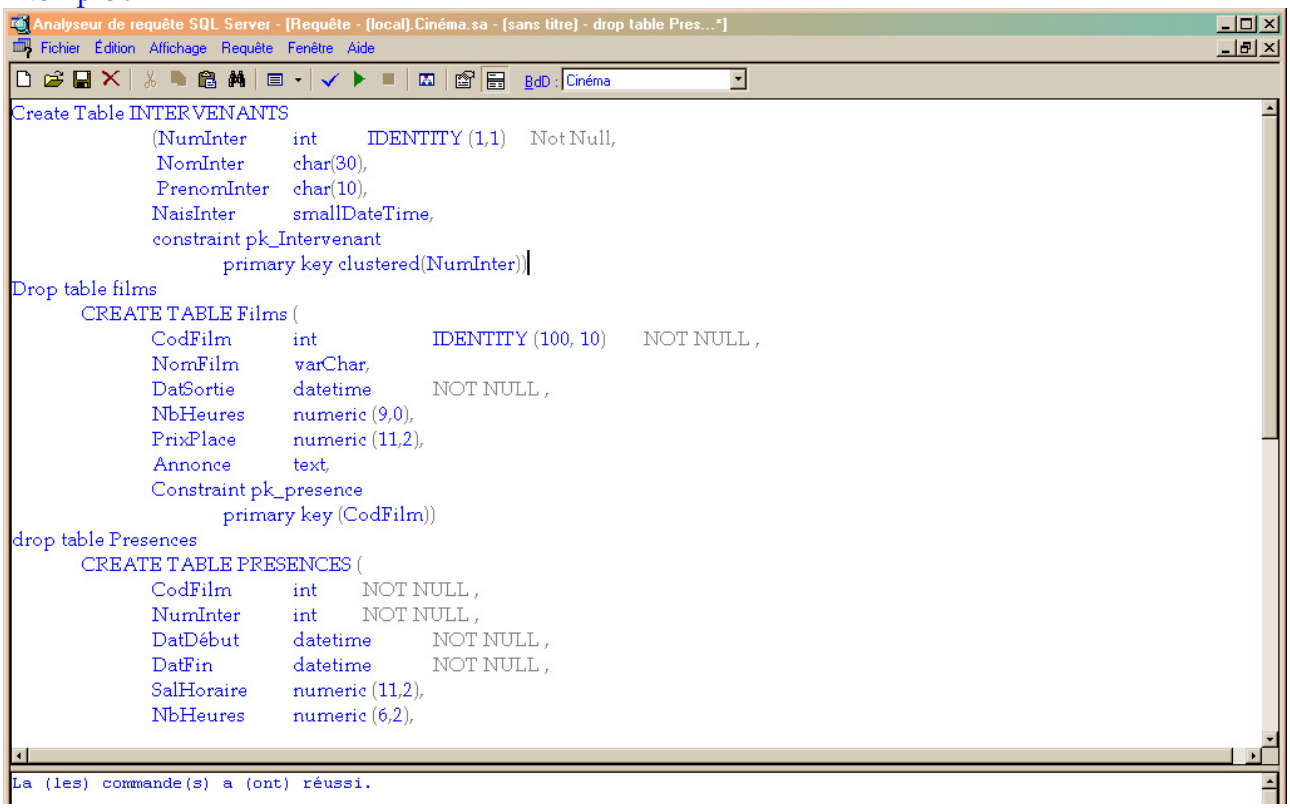
Pensez à la sauvegarde de votre database
Sous Enterprise manager :
Cliquez droit sur le nom de votre base de données et utiliser l'option de backup.

Si vous faites un Drop malencontreux :
Cliquez droit sur Databases et utiliser le restore en indiquant votre nom de base (il n'apparaît pas dans la liste **vous devez indiquez son nom**

Définition d une base de données relationnelle

Vous pouvez saisir plusieurs commandes sue l'écran de l'analyseur de requêtes.

Exemple :



Cela peut être pratique de conserver l'ensemble des commandes pour pouvoir les enregistrer dans un fichier qui sera un script SQL et que vous pourrez relancer autant de fois que nécessaire.

Comment sauver son travail :

Indiquez le nom de votre script (un nom qui n'existe pas) puis cliquez sur enregistrer.

Comme pour tout travail sur micro pensez à faire des sauvegardes régulières de vos scripts.

Pour récupérer un script existant, faire :

- Fichier
- Ouvrir
- Puis rechercher votre script.

Si vous appuyez sur "exécuter", l'ensemble des instructions s'exécute dans l'ordre où elles sont saisies. Si vous ne souhaitez exécuter une seule instruction, sélectionnez-la(mettre votre curseur, en début de la sélection souhaitée, puis appuyez sur la touche ⇧ (maj) et cliquez sur la fin de la zone à sélectionner. Votre sélection apparaît en inverse vidéo. Cliquez sur ▶ pour exécuter seulement la partie d'instructions sélectionnée.

4.6 Modification de la structure d'une table ALTER TABLE

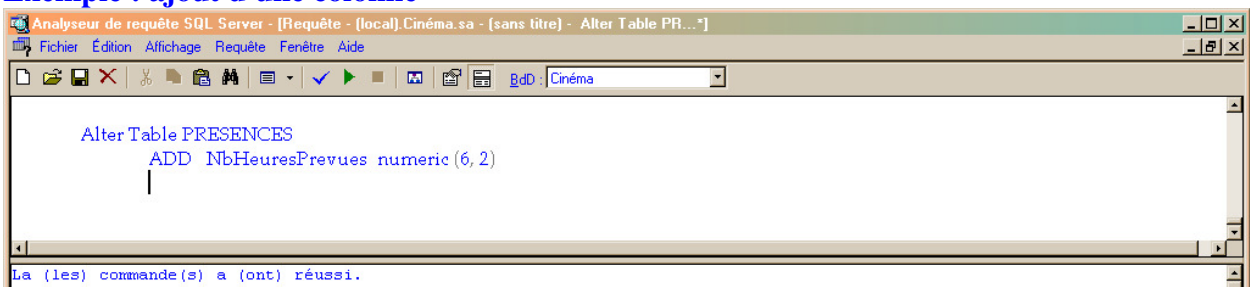
Il est parfois nécessaire

- d'ajouter des colonnes non prévues lors du create table
- de retirer des contraintes portées à tort sur la table.

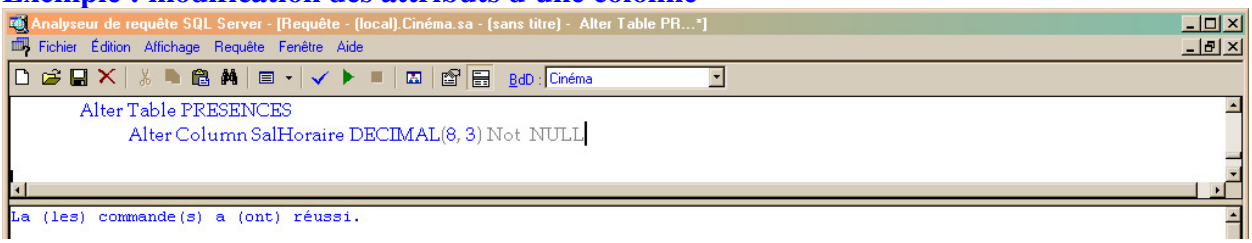
Structure

```
ALTER TABLE nom table
        ADD  nomNouvelleCol typeColonne,
        DROP nomColEn Trop,
        ALTER COLUMN nomcolAChanger nouveau_type_de_données
                                        NULL ou  NOT NULL
        DROP CONSTRAINT nom Contrainte
```

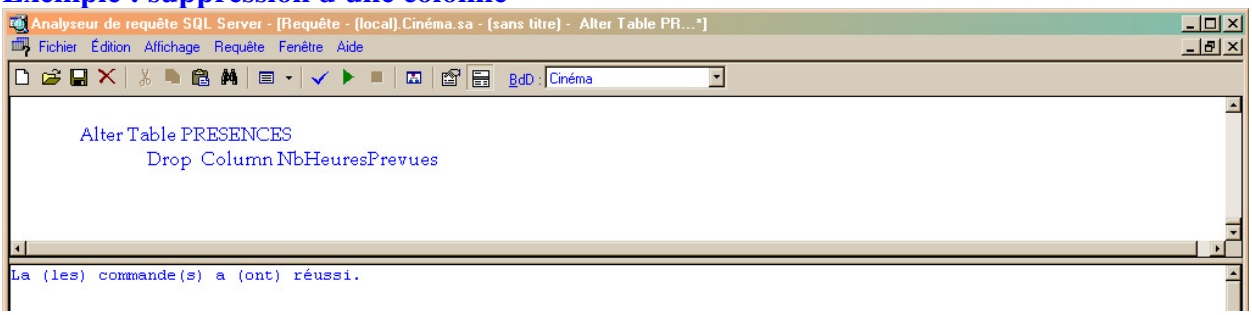
Exemple : ajout d'une colonne



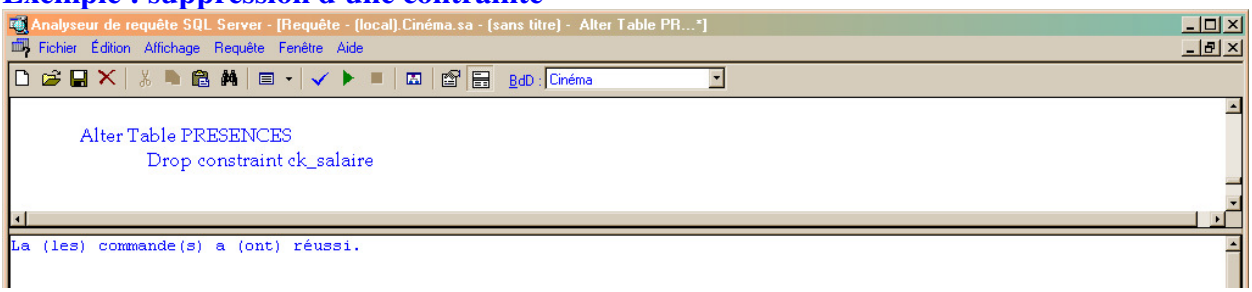
Exemple : modification des attributs d'une colonne



Exemple : suppression d'une colonne



Exemple : suppression d'une contrainte



Vous pouvez réaliser plusieurs opérations Add ou Drop avec un seul Alter Table.

Activité n° 1 : constitution d'une base

:

Réfléchissez sur papier avant de vous lancer sur écran
Passez les ordres SQL vous permettant de répondre aux demandes ci-dessous.
Conservez votre travail dans un script MYTRAVxx (xx étant vos initiales) se trouvant dans le dossier D:\ Données \ SQL

1) Création de la base de données ROYAUTxx (xx étant vos initiales) .

2) Création d'une première table dans cette base :

nom de la table	MONARQUE
Colonnes de la tables	nom du roi sur 15 caractères
	date de début de règne
	date de fin de règne

avec une contrainte de clé primaire sur nom du roi.

3) Ajout de deux nouvelles colonnes à cette table

dynastie sur 12 caractères
surnom sur 15 caractères

4) Toujours sur la même base de données, création de deux tables

table EPOUSES avec	nom époux sur 25 caractères
	nom épouse sur 25 caractères
	date de mariage

avec une contrainte clé primaire sur nom époux et nom épouse

table ENFANTS avec	nom père sur 25 caractères
	nom mère sur 25 caractères
	prénom enfant sur 30 caractères
	date naissance de l'enfant

Créer la contrainte clé primaire sur prénom et date de naissance et clé étrangère sur nom mére.

Définition d une base de données relationnelle

Nous venons de définir une structure de base de données. Dans ce chapitre, nous allons étudier comment alimenter cette base avec des données. Puis il nous faudra découvrir des notions fondamentales

- les conditions
- les clauses
- les prédicats.
- les vues

5 Alimentation d'une base de données relationnelle

5.1 Remplissage de valeurs de colonnes INSERT INTO

Les données sont saisies dans une table par la commande INSERT INTO.

Structure de la commande en remplissant avec des valeurs

Pour insérer des valeurs dans toutes les colonnes :

```
INSERT INTO <nom table>
                    values (val1, val2, val3, ....., valn)
```

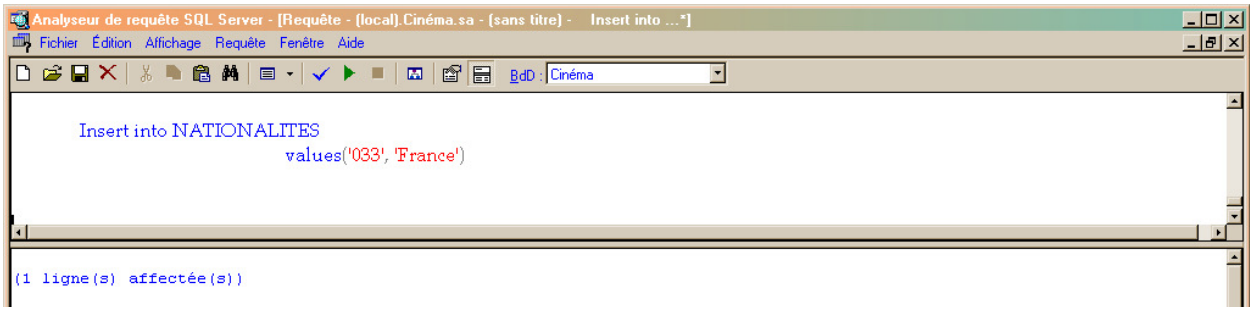
Ou pour insérer dans quelques colonnes

```
INSERT INTO <nom table> (col1, col2, col3,……., coln)
                    values (val1, val2, val3, ....., valn)
```

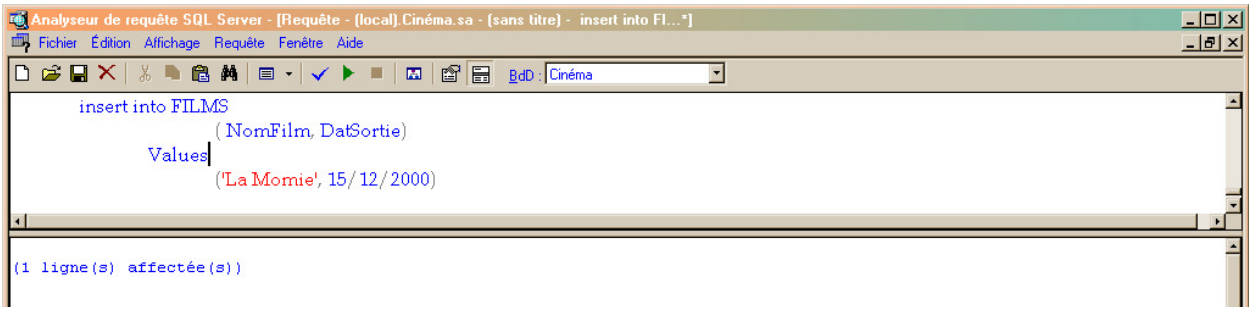
Il faut noter que :

- Toutes les colonnes ne sont pas nécessaires.
- L'indication des colonnes peut se faire dans n'importe quel ordre, il faut simplement respecter la correspondance avec VALUES.
- Pour les colonnes définies NOT NULL la valeur est obligatoire. À moins qu'une contrainte DEFAULT soit définie au quel cas c'est la valeur par défaut qui sera prise.

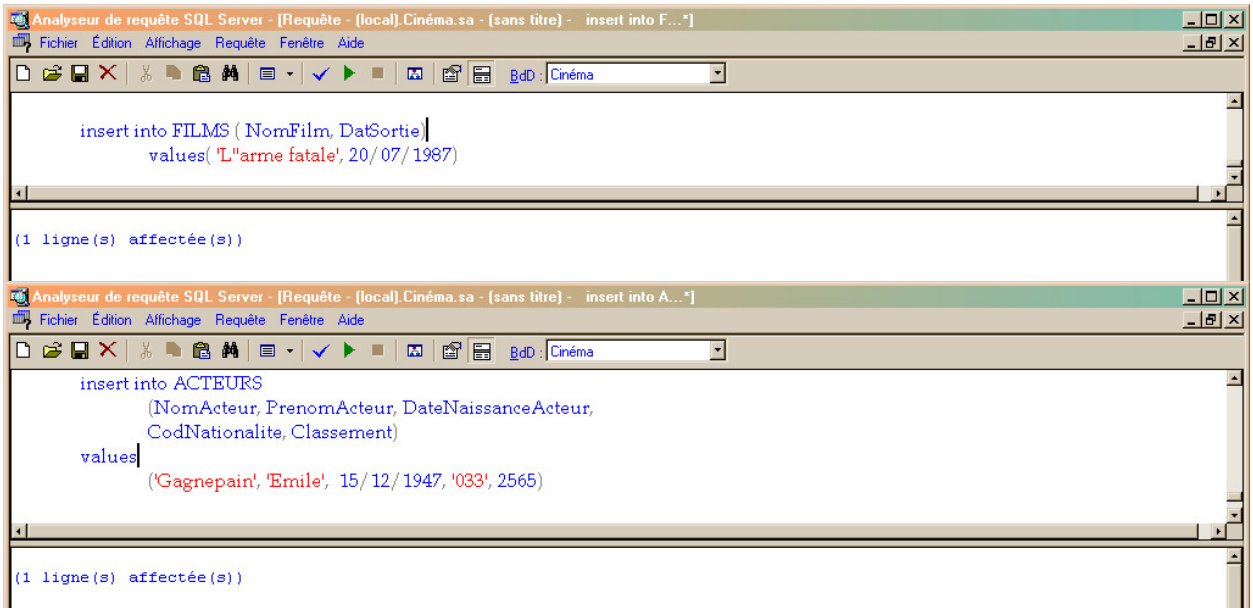
Exemple de commande



Par contre, si nous ne désirons alimenter que quelques colonnes de la table, nous pouvons utiliser cette structure de commande.



La clé primaire étant de type Identity, il ne faut pas l'indiquer et laisser la machine calculer sa valeur.
Vous remarquerez que les colonnes de type caractère sont entre ' ' ; les colonnes numériques ne le sont pas. Lorsqu'une valeur de type caractère doit contenir un ', il faut doubler le ' :



Alimentation d'une base de données relationnelle

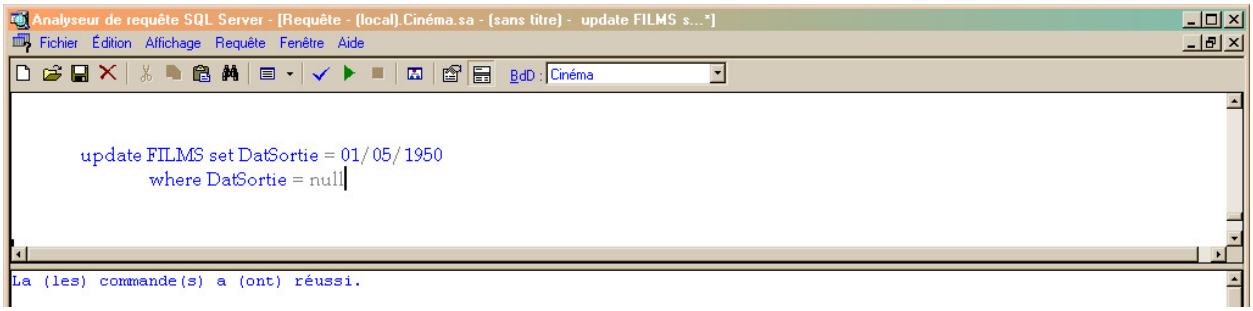
Un autre moyen de charger une table est d'utiliser une instruction SELECT qui prend des informations dans d'autres tables pour les insérer dans la table voulue. Nous verrons ce mode d'insertion ultérieurement.

5.2 Modification des valeurs dans les colonnes UPDATE

Les données peuvent être automatiquement modifiées par la commande UPDATE..

Structure de la commande

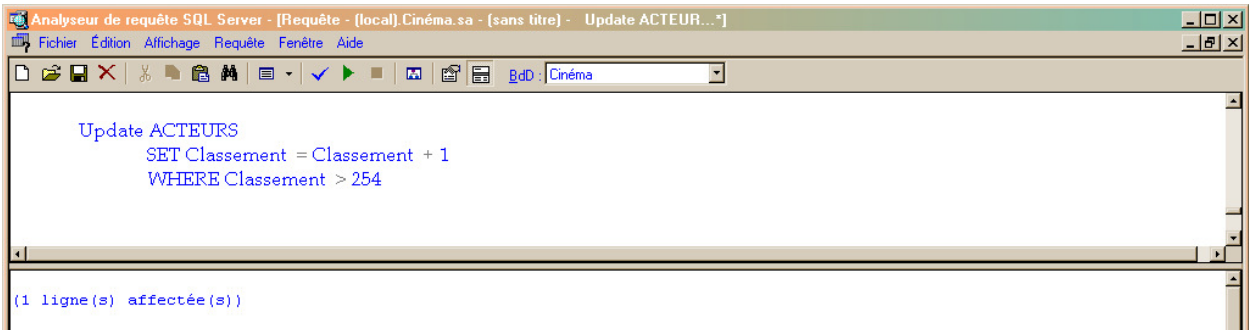
```
UPDATE <nom table>
        SET     <colonne1> = <expression>,
                <colonne2> = <expression2>
        WHERE  condition
```



Plusieurs lignes peuvent être modifiées par un seul ordre UPDATE..

Indiquez

- le nom de la table ou de la vue (notion étudiée ultérieurement
- le nom des zones à modifier avec dans la valeur :
  - soit la nouvelle valeur,
  - soit une expression permettant de déterminer la nouvelle valeur affectée à la zone.
- la ou les conditions de prise en compte: derrière la cluse WHERE



Ceci permet de décaler d'une place tous les acteurs classés 255éme ou plus.

5.3 Suppression dans une table DELETE

Les lignes sont supprimées d'une table par la commande DELETE

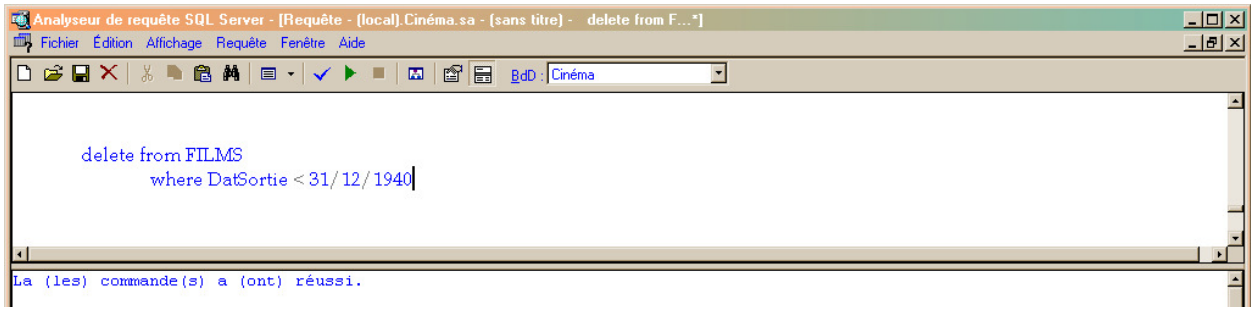
Structure de la commande

```
DELETE FROM   <nom table>   WHERE condition
```

Plusieurs lignes peuvent être supprimées par un ordre DELETE.

Vous pouvez même, si vous ni prenez garde, effacer toutes les lignes d'une table : il suffit que :

- votre condition soit toujours vraie quel que soit la ligne de la table,
- qu'il n'y ait pas de conditions.



La clause WHERE est identique à celle étudiée en détail avec la commande SELECT dans un chapitre qui suit

Attention : mettre les caractères en minuscules ou en majuscules a une importance :

* Mérovingien est # de mérovingien

Les accents aussi :

* mérovingien est # de merovingien

Activité n° 2 : gestion d'une base

Ajoutez la colonne trésor à la table monarque puis créez un script permettant de charger cette table avec les données ci-dessous

Nom du roi	Date début de règne	Date fin de règne	Dynastie	Surnom	Trésor
Chlodion	428	447	mérovingien	Le chevelu	28
Mérovée		458	mérovingien		25
Clovis 1er	465	511	mérovingien		120
Dagobert	604	639	Mérovingien		50
Pépin	714	768	mérovingien	Le Bref	0
Charlemagne	800	814	mérovingien		850
Philippe II	1165	1223	capétien		500
Louis 1er	781	850	mérovingien	le pieux	100
Louis II	877	879	mérovingien	le bègue	10
Louis III	879	882	mérovingien		0
Louis IV	936	954	mérovingien	d'outremer	20
Louis V	986	987	mérovingien		0
Louis VI	1108	1137	capétien	le gros	20
Louis VII	1137	1180	capétien	le jeune	100
Louis VIII	1223	1226	capétien	Le lion	0
Louis IX	1226	1270	capétien	Saint louis	30
Louis X	1314	1316	capétien	le hutin	50
Louis XI	1461	1483	capétien		100
Louis XII	1496	1515	capétien		20
Louis XIII	1610	1641	capétien	le juste	100
Louis XIV	1643	1715	capétien	le grand	50
Louis XV	1715	1774	capétien	le bien-aimé	-20
Louis XVI	1774	1791	capétien		-200

Créez un script permettant de charger la table épouse.

nom époux	nom épouse	Date mariage
Pépin	**Berthe aux grands pieds**	**749**
Louis 1er	**Inningarde**	**818**
Louis 1er	**Judith de bavière**	**819**
Louis VII	**Aliénor d'aquitaine**	
Louis VII	**Constance de Castille**	**1154**
Louis VII	**Adèle de champagne**	**1160**
Louis VIII	**Blanche de Castille**	
Louis X	**Marguerite de bourgogne**	
Dagobert	**Nanthilde**	**620**
Dagobert	**Ragnetrude**	**630**
Louis IX	**Marguerite de provence**	
Louis XII	**Anne de Bretagne**	
Charles VIII	**Anne de Bretagne**	
Louis XIII	**Anne d'Autriche**	
Louis XV	**Marie Leszczynska**	
Louis XVI	**Madame Capet**	

Notez que certaines dates de mariage sont inconnues.

Créez un script permettant de charger la table enfant.
N'hésitez pas à utiliser le copier – coller.

Nom père	Nom mère	Prénom	Naissance
Pépin	Berthe aux garns pieds	Carolan	751
Pépin	Berthe aux garns pieds	Charlemagne	742
Louis 1er	Inningarde	Lothaire	795
Louis 1er	Inningarde	Pépin II	
Louis 1er	Judith de Bavière	Charles	
Philippe III	Isabelle d'Aragon	Philippe IV	1268
Philippe III	Isabelle d'Aragon	Charles de valois	1270
Philippe IV	Jeanne de Navarre	Louis X	1289
Philippe IV	Jeanne de Navarre	Philippe V	1293
Philippe IV	Jeanne de Navarre	Charles IV	1294
Louis XI	Inconnu	Louis VII	1120
Philippe Auguste	Inconnu	Louis VIII	1187
Louis VIII	Blanche de castille	Louis IX	1214
Charles VII	Marie d'Anjou	Louis XI	1423
Charles d'Orléans	Marie de Clèves	Louis XII	1462
Henri IV	Marie de Médicis	Louis XIII	1601
Louis XIII	Anne d'Autriche	Louis XIV	1638
Louis de Bourgogne	Marie Adélaîde	Louis XV	1710
Louis XIV	Mme de Montespan	Louis de bourgogne	1682
Louis XV	Marie Leszczynska	Louis Dauphin de France	1729
Louis Dauphin		Louis XVI	1754

Note de l'auteur : toute ressemblance avec une certaine vérité historique serait un pur hasard
Certains enfants ont un père qui n'a pas été roi idem pour certaines mères non reines.

6 Extraction d'informations : SELECT

6.1 Structure de la commande

```
SELECT    *    FROM <nom de table>   clause 1,  .. ,  clause N
```

ou

```
SELECT <col 1> , <col 2> , <col N> FROM <nom de table>
        clause 1    .....          clause N
```

* est une notation qui permet de dire 'toutes les colonnes'.

Les clauses sont des mots réservés du langage SQL indiquant la disposition de la table (ou des tables) sur laquelle porte le verbe commande.

Derrière SELECT, tapez le nom des colonnes (zones) à visualiser
Les colonnes (zones) doivent être séparées par des virgules.

6.2 La clause FROM:

Cette clause indique le nom des tables ou des vues à utiliser pour créer la table résultante. Elle est obligatoire avec SELECT.

A	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	**Prof**	**2**	**F**	**15000,00**
	35	**Atchoum**	**5**	**M**	**14000,00**
	20	**Timide**	**5**	**M**	**15500,00**
	30	**Simplet**	**7**	**M**	**13000,00**
	25	**Dormeur**	**8**	**F**	**14500,00**

SELECT numéro, nom, classif, sexe, salaire FROM A

Résultat

⬇

	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	**Prof**	**2**	**F**	**15000,00**
	35	**Atchoum**	**5**	**M**	**14000,00**
	20	**Timide**	**5**	**M**	**15500,00**
	30	**Simplet**	**7**	**M**	**13000,00**
	25	**Dormeur**	**8**	**F**	**14500,00**

Structure de la commande SELECT :

```
SELECT      <nom colonne 1>, ...., <nom colonne n>     FROM <nom de table>
```

permet de sélectionner dans la table 'nom de table' les colonnes dont le nom est cité. Ou

```
SELECT  *  FROM  <nom de table>
```

permet de sélectionner toutes les colonnes de la table "nom de table".

Exemple : SELECT nom_roi,, surnom FROM monarque

6.3 La clause WHERE :

Cette clause est toujours suivie d'une condition de recherche pour chaque ligne (enregistrement) résultant de la clause FROM. La table résultante contient les lignes (enregistrements) répondant VRAI à la condition de recherche.

A	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	**Prof**	**2**	**F**	**15000,00**
	35	**Atchoum**	**5**	**M**	**14000,00**
	20	**Timide**	**5**	**M**	**15500,00**
	30	**Simplet**	**7**	**M**	**13000,00**
	25	**Dormeur**	**8**	**F**	**14500,00**

```
SELECT numéro, nom, classif, salaire FROM A
    WHERE classif = 5
```

Résultat

⬇

	Numéro	Nom	Classif.	Salaire brut
	35	**Atchoum**	**5**	**14000,00**
	20	**Timide**	**5**	**15500,00**

Structure de la commande SELECT :

```
SELECT      <nom colonne 1>, ...., <nom colonne n>
    FROM <nom de table>
    WHERE <condition>
```

permet de sélectionner dans la table "nom de table" les colonnes dont le nom est cité et les lignes pour qui la condition est vraie

ou

```
SELECT * FROM <nom de table>
    WHERE <condition>
```

permet de sélectionner toutes les colonnes de la table "nom de table" et les lignes pour qui la condition est vraie.

Exemple :
Pour mieux comprendre, passons les commandes suivantes :

```
SELECT nom_roi FROM  monarque ;
```

puis

```
SELECT  nom_roi FROM monarque WHERE datedebut > 1700
```

La première commande vous donne toutes les lignes de la table ; par contre, la seconde ne vous restitue que les rois dont la date de début de règne est plus grande que 1700. Vous obtenez ainsi une vue plus limitée.

Les données dans la clause WHERE :
Les données caractères doivent être comprises entre apostrophes.

```
SELECT numéro, nom, classif, salaire FROM A
                    WHERE sexe = 'M'
```

Les données numériques ne doivent pas être comprises entre apostrophes :

```
SELECT numéro, nom, classif, salaire FROM A
                    WHERE classif = 5
```

6.4 La clause ORDER BY :

Cette clause est facultative avec l'ordre SELECT. Si elle est indiquée, elle doit apparaître après les clauses FROM ET WHERE. Elle classe les lignes (enregistrements) de la table (fichier) résultante selon les noms des colonnes (zones) que vous avez identifiées. Si vous avez identifié plus d'une colonne (zone), les lignes (enregistrements) sont d'abord classées selon le nom des colonnes (zones) que vous avez identifié en premier, puis selon le nom des colonnes (zones) que vous avez identifié en second, etc. Elle permet de spécifier l'ordre d'affichage des lignes.

	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	**Prof**	**2**	**F**	**15000,00**
	35	**Atchoum**	**5**	**M**	**14000,00**
	20	**Timide**	**5**	**M**	**15500,00**
	30	**Simplet**	**7**	**M**	**13000,00**
	25	**Dormeur**	**8**	**F**	**14500,00**

```
SELECT numéro, nom, classif, salaire  FROM A
                ORDER BY numéro
```

Résultat

⬇

	Numéro	Nom	Classif.	Salaire brut
	10	**Prof**	**2**	**15000,00**
	20	**Timide**	**5**	**15500,00**
	25	**Dormeur**	**8**	**14500,00**
	30	**Simplet**	**7**	**13000,00**
	35	**Atchoum**	**5**	**14000,00**

Structure de la commande SELECT :

```
SELECT  <nom colonne 1>, ....,<nom colonne n>
                FROM <nom de table>
                ORDER BY <nom colonne x>
```

permet de sélectionner dans la table "nom de table" les colonnes dont le nom est cité et de les afficher par ordre croissant sur la valeur de <colonne x>

Exemple :
Passons la commande suivante :

```
SELECT nom_roi, datedebut, surnom
                FROM monarque
                ORDER BY surnom
```

Nous obtenons à l'écran les informations des trois colonnes nom du roi, sa date de début de règne et son surnom venant de la table monarque et affichées selon l'ordre des surnoms. Comme nous n'avons rien précisé de particulier sur l'ordre, les informations sortent dans l'ordre croissant.

Allons plus loin ! sortons les informations selon l'ordre suivant :

- par dynastie

- à l'intérieur de la dynastie par ordre décroissant de début de règne c'est-à-dire le règne le plus récent en tête :

```
SELECT nom_roi, datedebut, dynastie, surnom
                FROM monarque
                ORDER BY dynastie, datedebut DESC
```

Nous obtenons alors les informations demandées de la table monarque affichées par dynastie et pour tous les monarques d'une même dynastie, ils apparaissent par ordre décroissant de date de début de règne.

Si la clause ORDER BY n'est pas spécifiée, l'ordre de sortie est indéterminé.

Il peut changer d'une requête à l'autre.

Sélection sous condition :

	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	**Prof**	**2**	**F**	**15000,00**
	35	**Atchoum**	**5**	**M**	**14000,00**
	20	**Timide**	**5**	**M**	**15500,00**
	30	**Simplet**	**7**	**M**	**13000,00**
	25	**Dormeur**	**8**	**F**	**14500,00**

```
SELECT        * FROM A
  WHERE classif < 7
  ORDER BY numéro
```

Résultat

	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	**Prof**	**2**	**F**	**15000,00**
	20	**Timide**	**5**	**M**	**15500,00**
	35	**Atchoum**	**5**	**M**	**14000,00**

Sélection sous condition (suite) :

	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	**Prof**	**2**	**F**	**15000,00**
	35	**Atchoum**	**5**	**M**	**14000,00**
	20	**Timide**	**5**	**M**	**15500,00**
	30	**Simplet**	**7**	**M**	**13000,00**
	25	**Dormeur**	**8**	**F**	**14500,00**

```
SELECT    * FROM A
          WHERE salaire >= 14000
          ORDER BY numéro
```

Résultat

	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	**Prof**	**2**	**F**	**15000,00**
	20	**Timide**	**5**	**M**	**15500,00**
	25	**Dormeur**	**8**	**F**	**14500,00**
	35	**Atchoum**	**5**	**M**	**14000,00**

Sélection sous condition (suite) :

	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	**Prof**	**2**	**F**	**15000,00**
	35	**Atchoum**	**5**	**M**	**14000,00**
	20	**Timide**	**5**	**M**	**15500,00**
	30	**Simplet**	**7**	**M**	**13000,00**
	25	**Dormeur**	**8**	**F**	**14500,00**

```
SELECT   * FROM A
         WHERE      salaire >= 14000 AND sexe = 'M'
         ORDER BY numéro
```

Résultat

	Numéro	Nom	Classif.	Sexe	Salaire brut
	20	**Timide**	**5**	**M**	**15500,00**
	35	**Atchoum**	**5**	**M**	**14000,00**

Sélection sous condition (suite) :

	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	**Prof**	**2**	**F**	**15000,00**
	35	**Atchoum**	**5**	**M**	**14000,00**
	20	**Timide**	**5**	**M**	**15500,00**
	30	**Simplet**	**7**	**M**	**13000,00**
	25	**Dormeur**	**8**	**F**	**14500,00**

```
SELECT      * FROM A
        WHERE       salaire >= 14000 OR sexe = 'M'
        ORDER BY nom
```

Résultat

	Numéro	Nom	Classif.	Sexe	Salaire brut
	35	**Atchoum**	**5**	**M**	**14000,00**
	25	**Dormeur**	**8**	**F**	**14500,00**
	10	**Prof**	**2**	**F**	**15000,00**
	30	**Simplet**	**7**	**M**	**13000,00**
	20	**Timide**	**5**	**M**	**15500,00**

Sélection sous condition (suite) :

	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	**Prof**	**2**	**F**	**15000,00**
	35	**Atchoum**	**5**	**M**	**14000,00**
	20	**Timide**	**5**	**M**	**15500,00**
	30	**Simplet**	**7**	**M**	**13000,00**
	25	**Dormeur**	**8**	**F**	**14500,00**

```
SELECT      * FROM A
      WHERE classif <> 5
      ORDER BY nom
```

Résultat

	Numéro	Nom	Classif.	Sexe	Salaire brut
	25	**Dormeur**	**8**	**F**	**14500,00**
	10	**Prof**	**2**	**F**	**15000,00**
	30	**Simplet**	**7**	**M**	**13000,00**

Sélection sous condition (suite) :

	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	**Prof**	**2**	**F**	**15000,00**
	35	**Atchoum**	**5**	**M**	**14000,00**
	20	**Timide**	**5**	**M**	**15500,00**
	30	**Simplet**	**7**	**M**	**13000,00**
	25	**Dormeur**	**8**	**F**	**14500,00**

```
SELECT  * FROM A
        WHERE       salaire >= 14000 OR sexe = 'M'
        ORDER BY nom
```

Résultat

	Numéro	Nom	Classif.	Sexe	Salaire brut
	35	**Atchoum**	**5**	**M**	**14000,00**
	25	**Dormeur**	**8**	**F**	**14500,00**
	10	**Prof**	**2**	**F**	**15000,00**
	30	**Simplet**	**7**	**M**	**13000,00**
	20	**Timide**	**5**	**M**	**15500,00**

VI 6 Les fonctions scalaires

Avec la clause WHERE, il nous est possible d'utiliser les prédicats suivants :

ALL , ANY , BETWEEN , EXISTS , IN , LIKE
NOT BETWEEN, NOT EXISTS, NOT IN, NOT LIKE

VI 61 BETWEEN permet de définir des bornes de comparaison (bornes incluses dans la comparaison)Il peut se combiner avec NOT.

Exemple :sélectionnons les rois qui avaient un trésor de 10 à 100 inclus :

```
SELECT nom_roi, surnom FROM monarque
WHERE trésor BETWEEN 10 and 100
```

équivaut à

```
SELECT nom_roi, surnom FROM monarque
WHERE trésor >= 10  and  trésor <= 100
```

Exemple : sélectionnons les rois qui n'avaient pas un trésor de 10 à 100 inclus :

```
SELECT nom_roi, surnom FROM monarque
    WHERE trésor < 10   OR   trésor > 100
```

VI 62 IN permet de tester si le contenu d'une colonnes est dans une liste de valeurs données. Il peut se combiner avec NOT.

Exemples sélectionnons les rois dont le trésor était 10 ou 20 ou 100:

```
SELECT nom_roi, surnom FROM monarque
    WHERE trésor IN (10 20 100)
```

sélectionnons les rois dont le surnom est le bègue ou le saint:

```
SELECT nom_roi, surnom FROM monarque
    WHERE surnom IN ('le bègue' 'le saint')
```

les colonnes en caractères doivent être comparées à des valeurs entre ''.

VI 63 LIKE permet une comparaison entre une chaîne de caractères et une colonne Elle utilise deux caractères spéciaux % et - (équivalent de * et ? sous Ms-DOS)

Exemple :sélectionnons les rois dont le nom est LOUIS IV:

```
SELECT nom_roi, surnom FROM monarque
        WHERE nom_roi LIKE 'LOUIS IV'
```

sélectionnons les rois dont le nom commence par LOUIS :

```
SELECT nom_roi, surnom FROM monarque
        WHERE nom_roi LIKE 'LOUIS%'
```

Le % derrière LOUIS permet de dire 'avec n'importe quoi derrière'

sélectionnons les rois dont le nom fini par VIII :

```
SELECT nom_roi, surnom FROM monarque
        WHERE nom_roi LIKE '%VIII%'
```

Le % devant VIII permet de dire 'avec n'importe quoi devant'

sélectionnons les épouses dont le nom a quatre lettres et commence par A

```
SELECT nom_épouse  FROM épouses
        WHERE nom_épouse LIKE 'A___'
```

Le _ derrière A permet d'indiquer le nombre de caractères qui doivent suivrent.

VI 64 ANY permet une comparaison d'une colonne avec une liste de valeurs fournies par une sous-interrogation. Si une des valeurs de la liste rend la condition vraie, alors la ligne est sélectionnée.

Exemple sélectionnons les enfants des rois carolingiens :

```
SELECT nom_père enfant   FROM  enfants
        WHERE  nom_père = ANY
        (SELECT  nom_roi  FROM monarque
                WHERE dynastie = 'carolingien')
```

La sous interrogation donne comme résultat une liste de rois comparée ensuite avec nom_pére.

VI 65 ALL permet une comparaison d'une colonne avec l'ensemble des valeurs d'une liste de valeurs fournies par une sous-interrogation. Si toutes les valeurs de la liste rendent la condition vraie, alors la ligne est sélectionnée.

VI 66 EXISTSest un prédicat évalué à vrai si la sous-requête qui le suit donne au moins une ligne c'est à dire un résultat non vide.

Exemple : SELECT nom-roi , dynastie from monarque
WHERE exists (SELECT * from monarques , enfants
WHERE nom_roi = nom_pere)
AND not exists (SELECT * from monarque , épouses
WHERE nom_roi = nom_epoux)

La corrélation se fait par nom_roi qui se trouve dans la projection de la requête et dans les Where des sous_requêtes.
On obtient ici les rois ayant des enfants mais pas d'épouses.

Sélection sous condition (suite) :

A	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	Prof	2	F	15000,00
	35	Atchoum	5	M	14000,00
	20	Timide	5	M	15500,00
	30	Simplet	7	M	13000,00
	25	Dormeur	8	F	14500,00

```
SELECT * FROM A
WHERE numéro BETWEEN 20 AND 30 ORDER BY nom
```

Résultat

Résultat	Numéro	Nom	Classif.	Sexe	Salaire brut
	25	Dormeur	8	F	14500,00
	30	Simplet	7	M	13000,00
	20	Timide	5	M	15500,00

Remarques :

Les limites sont incluses.

La limite inférieure doit être définie la première.

Sélection sous condition (suite) :

A	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	Prof	2	F	15000,00
	35	Atchoum	5	M	14000,00
	20	Timide	5	M	15500,00
	30	Simplet	7	M	13000,00
	25	Dormeur	8	F	14500,00

```
SELECT   * FROM A
         WHERE      classif IN (2, 7, 8)
         ORDER BY nom
```

Résultat

Résultat	Numéro	Nom	Classif.	Sexe	Salaire brut
	25	Dormeur	8	F	14500,00
	10	Prof	2	F	15000,00
	30	Simplet	7	M	13000,00

Sélection sous condition (suite) :

A	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	**Prof**	**2**	**F**	**15000,00**
	35	**Atchoum**	**5**	**M**	**14000,00**
	20	**Timide**	**5**	**M**	**15500,00**
	30	**Simplet**	**7**	**M**	**13000,00**
	25	**Dormeur**	**8**	**F**	**14500,00**

```
SELECT    * FROM A
          WHERE nom LIKE 'Timide'
          ORDER BY nom
```

Résultat

Résultat	Numéro	Nom	Classif.	Sexe	Salaire brut
	20	**Timide**	**5**	**M**	**15500,00**

Sélection sous condition (suite) :

A	Numéro	Nom	Classif.	Sexe	Salaire brut
	5	George	3	F	10000,00
	10	Prof	2	F	15000,00
	35	Atchoum	5	M	14000,00
	20	Georges	5	M	15500,00
	30	Simplet	7	M	13000,00
	25	Dormeur	8	F	14500,00

```
SELECT    * FROM A
          WHERE nom LIKE 'George%'
          ORDER BY nom
```

Résultat

Resultat	Numéro	Nom	Classif.	Sexe	Salaire brut
	5	George	3	F	10000,00
	20	Georges	5	M	15500,00

Sélection sous condition (suite) :

A	Numéro	Nom	Classif.	Sexe	Salaire brut
	5	**Martine**	**3**	**F**	**10000,00**
	10	**Prof**	**2**	**F**	**15000,00**
	35	**Atchoum**	**5**	**M**	**14000,00**
	40	**Pauline**	**5**	**M**	**15500,00**
	30	**Simplet**	**7**	**M**	**13000,00**
	25	**Dormeur**	**8**	**F**	**14500,00**

```
SELECT      * FROM A
              WHERE nom LIKE '%ine'
              ORDER BY nom
```

Résultat

Resultat	Numéro	Nom	Classif.	Sexe	Salaire brut
	5	**Martine**	**3**	**F**	**10000,00**
	40	**Pauline**	**5**	**M**	**15500,00**

Sélection sous condition (suite) :

A	Numéro	Nom	Classif.	Sexe	Salaire brut
	5	Alain	3	F	10000,00
	10	Prof	2	F	15000,00
	35	Atchoum	5	M	14000,00
	20	Albin	5	M	15500,00
	30	Simplet	7	M	13000,00
	25	Dormeur	8	F	14500,00

```
SELECT   * FROM A
         WHERE nom LIKE 'AL%IN'
         ORDER BY nom
```

Résultat

Resultat	Numéro	Nom	Classif.	Sexe	Salaire brut
	5	Alain	3	F	10000,00
	20	Albin	5	M	15500,00

Sélection sous condition (suite) :

A	Numéro	Nom	Classif.	Sexe	Salaire brut
	5	Alain	3	F	10000,00
	10	Prof	2	F	15000,00
	35	Atchoum	5	M	14000,00
	20	Albin	5	M	15500,00
	30	Simplet	7	M	13000,00
	25	Dormeur	8	F	14500,00

```
SELECT    * FROM A
          WHERE nom NOT LIKE 'AL%IN'
          ORDER BY nom
```

Résultat

Resultat	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	Prof	2	F	15000,00
	35	Atchoum	5	M	14000,00
	30	Simplet	7	M	13000,00
	25	Dormeur	8	F	14500,00

6.5 Fonctions sur chaînes de caractères

Il est possible d'extraire des caractères depuis la valeur contenue dans une colonne quand celle-ci est de type caractère grâce à la fonction

SUBSTRING(<nom colonne>, posit, long)

Il faut indiquer sur quelle colonne faire l'extraction puis la position de départ de l'extraction et la longueur de la chaîne extraite.

Exemple :

	Numéro	Nom	Classif.	Sexe	Salaire brut
	5	**Dominique**	**3**	**F**	**10000,00**
	10	**Rémi**	**2**	**F**	**15000,00**
	35	**Sidonie**	**5**	**M**	**14000,00**
	40	**Dorian**	**5**	**M**	**15500,00**
	30	**Fadette**	**2**	**M**	**13000,00**
	25	**Michelle**	**2**	**F**	**14500,00**

```
SELECT    numéro, SUBSTRING(nom, 1, 2)
          FROM A
          WHERE classif <= 3
          ORDER BY numéro
```

Résultat

Numéro	
5	**Do**
10	**Ré**
25	**Mi**
30	**Fa**

6.6 SELECT DISTINCT

Cette clause permet de ne pas afficher les lignes qui ont même valeur. Si elle est indiquée, les enregistrements en double seront supprimés du fichier résultant final.:

Exemple :

```
SELECT DISTINCT dynastie FROM monarque
```

Vous obtenez ainsi la liste des dynasties de la table monarque.
Puissant ; isn't it ?

A	Numéro	Nom	Classif.	Sexe	Salaire brut
	10	**Prof**	**2**	**F**	**15000,00**
	35	**Atchoum**	**5**	**M**	**14000,00**
	20	**Timide**	**5**	**M**	**15500,00**
	30	**Simplet**	**7**	**M**	**13000,00**
	25	**Dormeur**	**8**	**F**	**14500,00**

```
SELECT DISTINCT classif
FROM A
```

Résultat

Classif.
2
5
7
8

6.7 Fonctions scalaires :

Il est possible de déterminer l'amplitude d'une colonne quand celle-ci est de type caractère ou numérique grâce à la fonction

LEN(<nom colonne>)

Il faut indiquer sur quelle colonne calculer l'amplitude.

Exemple :

	Numéro	Nom	Classif.	Sexe	Salaire brut
	5	**Dominique**	**3**	**F**	**10000,00**
	10	**Rémi**	**2**	**F**	**15000,00**
	35	**Sidonie**	**5**	**M**	**14000,00**
	40	**Dorian**	**5**	**M**	**15500,00**
	30	**Fadette**	**2**	**M**	**13000,00**
	25	**Michelle**	**2**	**F**	**14500,00**

```
SELECT    numéro, LEN(nom) as longueur
          FROM A
          WHERE classif <= 3
          ORDER BY numéro
```

Résultat

Numéro	longueur)
5	**9**
10	**4**
25	**8**
30	**7**

Autres possibilités parmi bien d'autres:

RTRIM(chaîne) permet de récupérer une chaîne de caractère sans espaces à la fin

LTRIM (chaîne) permet de récupérer une chaîne de caractère sans espaces au début

LEFT**(***chaîne*, *long***)** permet de récupérer le début de la chaîne sur une longueur de long caractères.

RIGHT**(***chaîne*, *début***)** permet de récupérer la fin de la chaîne à partir de la position début.

UPPER(chaîne) convertit une chaîne de minuscules en majuscules.

REVERSE(chaîne) permet de récupérer une chaîne de caractère inversée (1er car en dernier car).

REPLACE permet de remplacer une partie de chaîne par une autre

```
select replace('Emile', 'ile', 'manuelle') from acteurs
```

Conseil : allez fouiller dans l'aide de SQL Server pour les possibilités offertes et la syntaxe exacte.

6.8 Fonctions conversion :

Il est possible de restructurer la forme d'une colonne à des fonctions telles que CAST et CONVERT

Il est possible d'effectuer des calculs ce qui permet d'obtenir une colonne "résultat" en sortie.

Dans cet exemple, conversion du prix de money en décimal puis calcul d'un montant "mont.

```
select NumArticle,
        CatArticle,
        QuantiteStock * cast(PrixUnitaire as decimal (5,2)) as mont,
        from Produits
```

6.9 Expressions calculées:

Exemple : calcul d'une augmentation de 1 %

A	Numéro	Nom	Classif.	Sexe	Salaire brut
	5	Dominique	3	F	10000,00
	10	Rémi	2	F	15000,00
	35	Sidonie	5	M	14000,00
	40	Dorian	5	M	15500,00
	30	Fadette	2	M	13000,00
	25	Michelle	2	F	14500,00

```
SELECT    numéro, nom, salaire / 100  as MontKf
FROM A
```

Résultat

Numéro	Nom	MontKf
5	Dominique	10
10	Rémi	15
35	Sidonie	14
40	Dorian	15
30	Fadette	13
25	Michelle	14

Remarque Il est possible de spécifier des valeurs constantes ou des noms de colonne dans une expression calculée (exemple : calcul d'un pourcentage entre deux colonnes)

Expressions calculées :

Exemple : Expression calculée sur la clause WHERE

A	Numéro	Nom	Classif.	Sexe	Salaire brut
	5	**Dominique**	**3**	**F**	**10000,00**
	10	**Rémi**	**2**	**F**	**15000,00**
	35	**Sidonie**	**5**	**M**	**14000,00**
	40	**Dorian**	**5**	**M**	**15500,00**
	30	**Fadette**	**2**	**M**	**13000,00**
	25	**Michelle**	**2**	**F**	**14500,00**

```
SELECT *
        FROM A
        WHERE (salaire / 100) > 140
```

Résultat

Numéro	Nom	Classif.	Sexe	Salaire brut
10	**Rémi**	**2**	**F**	**15000,00**
40	**Dorian**	**5**	**M**	**15500,00**
25	**Michelle**	**2**	**F**	**14500,00**

Expressions calculées :

Exemple : Classement sur une colonne

A	Numéro	Nom	Classif.	Sexe	Salaire brut
	5	**Dominique**	**3**	**F**	**10000,00**
	10	**Rémi**	**2**	**F**	**15000,00**
	35	**Sidonie**	**5**	**M**	**14000,00**
	40	**Dorian**	**5**	**M**	**15500,00**
	30	**Fadette**	**2**	**M**	**13000,00**
	25	**Michelle**	**2**	**F**	**14500,00**

```
SELECT    numéro, nom, classif,  MontKF as salaire / 1000
          FROM A
          ORDER BY MontKF
```

Résultat

Numéro	Nom	Classif.	
5	**Dominique**	**3**	**100,00**
30	**Fadette**	**2**	**130,00**
35	**Sidonie**	**5**	**140,00**
25	**Michelle**	**2**	**145,00**
10	**Rémi**	**2**	**150,00**
40	**Dorian**	**5**	**155,00**

6.9.1 Expressions calculées : concaténation

Exemple :

A	Numéro	Nom	Classif.	Sexe	Salaire brut
	5	Dominique	3	F	10000,00
	10	Rémi	2	F	15000,00
	35	Sidonie	5	M	14000,00
	40	Dorian	5	M	15500,00
	30	Fadette	2	M	13000,00
	25	Michelle	2	F	14500,00

```
SELECT     numéro, 'Nom ='  + nom FROM A
```

Résultat

Numéro	
5	Nom = Dominique
10	Nom = Rémi
25	Nom = Michelle
30	Nom = Fadette
35	Nom = Sidonie
40	Nom = Dorian

6.10 Les fonctions récapitulatives sur les colonnes

SUM (<nom colonne>)

calcul du total des valeurs contenues dans la colonne.
(seulement sur valeurs numériques)

(<nom colonne>)

extraction la plus petite valeur contenue dans la colonne.

nom colonne>)

extraction de la plus grande valeur contenue dans la colonne.

(<nom colonne>)

calcule la moyenne des valeurs contenues dans la colonne.
(seulement sur valeurs numériques)

(*) nombre de lignes sélectionnées

A	Numéro	Nom	Classif.	Sexe	Salaire brut
	5	**Dominique**	**3**	**F**	**10000,00**
	10	**Rémi**	**2**	**F**	**15000,00**
	35	**Sidonie**	**5**	**M**	**14000,00**
	40	**Dorian**	**5**	**M**	**15500,00**
	30	**Fadette**	**2**	**M**	**13000,00**
	25	**Michelle**	**2**	**F**	**14500,00**

SELECT SUM(salaire), MIN(salaire), MAX(salaire), AVG(salaire), COUNT(*) FROM A

Résultat

SUM(salaire)	MIN(salaire)	MAX(salaire)	AVG(salaire)	COUNT(*)
82000,00	**10000,00**	**15500,00**	**13666,66**	**6**

COUNT(*) compte le nombre de lignes
COUNT(noncol) compte le nombre de valeurs non nulles sur cette colonne
COUNT(DISTINCT nomcol) compte le nombre de valeurs distinctes sur cette colonne

6.11 La clause GROUP BY

Cette clause permet de totaliser par groupe les lignes de la table sur le critère indiqué derrière GROUP BY:
Elle produit une table (fichier) résultant intermédiaire en groupant les lignes (enregistrements) par colonne. Les colonnes (zones) doivent être séparées par des virgules.
ORDER BY est inutile si vous souhaitez voir la table résultante triée, classée sur les mêmes critères que ceux du GROUP BY.

A	Numéro	Nom	Classif.	Sexe	Salaire brut
	5	**Dominique**	**3**	**F**	**10000,00**
	10	**Rémi**	**2**	**F**	**15000,00**
	35	**Sidonie**	**5**	**M**	**14000,00**
	40	**Dorian**	**5**	**M**	**15500,00**
	30	**Fadette**	**2**	**M**	**13000,00**
	25	**Michelle**	**2**	**F**	**14600,00**

```
SELECT      classif, AVG(salaire)
            FROM A
            GROUP BY classif
```

Résultat

Classif	AVG(salaire)
2	**14200,00**
3	**10000,00**
5	**14750,00**

Structure de la commande SELECT :

SELECT <nom colonne 1>, ..., <nom colonne n>
FROM <nom de table>
GROUP BY <nom colonne x>

permet de sélectionner dans la table "nom de table" les colonnes dont le nom est cité et de les regrouper sur la valeur de <colonne x>.

Remarque :
Sauf pour la colonne de groupage, **toutes** les autres colonnes **doivent** être spécifiées par **fonction**.

Exemple :
Pour mieux comprendre, nous allons passer quelques commandes :

```
SELECT dynastie, SUM(trésor)  FROM monarque
ORDER BY sum (trésor)
GROUP BY  dynastie
```

Vous avez une erreur de syntaxe.

```
SELECT dynastie, SUM(trésor)  FROM monarque
GROUP BY  dynastie
ORDER BY 2
```

Expliquez ce que vous voyez à l'écran .

6.12 La clause HAVING ;

Cette clause permet de totaliser sous condition par groupe les lignes de la table sur le critère indiqué derrière GROUP BY: elle produit une table (fichier) résultante intermédiaire en appliquant une condition de recherche à chaque groupe de la clause GROUP BY

A	Numéro	Nom	Classif.	Sexe	Salaire brut
	5	**Dominique**	**3**	**F**	**10000,00**
	10	**Rémi**	**2**	**F**	**15000,00**
	35	**Sidonie**	**5**	**M**	**14000,00**
	40	**Dorian**	**5**	**M**	**15500,00**
	30	**Fadette**	**2**	**M**	**13000,00**
	25	**Michelle**	**2**	**F**	**14600,00**

```
SELECT      classif, AVG(salaire) FROM A
        GROUP BY classif
        HAVING COUNT (*) > 1
        ORDER BY classif
```

Résultat

Classif	AVG(salaire)
2	**14200,00**
5	**14750,00**

Cette clause est toujours utilisée avec GROUP BY. C'est l'équivalent de la clause WHERE mais pour un ensemble de lignes regroupées par GROUP BY.

Il est possible d'utiliser des fonctions avec cette clause.

Exemple :

```
SELECT dynastie, SUM(trésor) FROM monarque
                GROUP BY dynastie
                HAVING SUM(trésor) > 150
```

Nous affichons deux colonnes (dynastie et trésor) en regroupant par dynastie avec le total des trésors par dynastie. Mais l'affichage n'est fait que si le total des trésors est supérieur à 150.

6.13 UNION

UNION avec une autre instruction SELECT
Ce mot permet d'indiquer que vous désirez combiner deux tables (fichiers) résultantes en une seule. Les valeurs possibles sont :

- UNION sans les lignes identiques entres les deux SELECT
- UNION All avec les lignes dupliquées

6.14 Jointure

La jointure permet de compléter les colonnes d'une table par des colonnes provenant d'autres tables avec lesquelles on a des critères de jointure c'est à dire des données de même nature qui permettent de faire une liaison, une relation.

Deux façons d'écrire une jointure

a) par la clause WHERE

```
USE cinéma
SELECT a.NumActeur, f.TitreFilm, a.Cachet
        FROM Acteurs  AS a, Films AS f, Participation as p
        WHERE a.NumActeur = p.NumActeur
          and       p.CodFilm      = f.CodFilm
```

b) par la clause FROM

```
USE cinéma
SELECT a.NumActeur, f.TitreFilm, a.Cachet
        FROM Acteurs  AS a
        LEFT OUTER JOIN Participation AS p
                ON a.NumActeur = p.NumActeur
        LEFT OUTER JOIN Films AS f
                ON p.CodFilm      = f.CodFilm
```

Il existe deux types de jointures

- INNER JOIN permet de compléter les colonnes d'une table par les colonnes d'une autre table s en utilisant un ou plusieurs critères de jointure
  Exemple : Nous souhaitons lister les acteurs et le libellé de leur nationalité

```
SELECT NumActeur, NomActeur, LibNationalité
    from Acteurs as a, Nationalités as n
    where a.CodNationalité = n.CodNationalité
```

La table résultante ne contiendra que les acteurs dont le code nationalité se trouve dans la table nationalité

- OUTER JOIN

LEFT OUTER JOIN or LEFT JOIN permet d'obtenir tous les acteurs même ceux pour lesquels on ne trouve pas de nationalité

```
SELECT NumActeur, NomActeur, LibNationalité
    from Acteurs as a LEFT OUTER JOIN, Nationalités as n
    ON a.CodNationalité = n.CodNationalité
```

La table résultante liste tous les acteurs avec le libellé de la nationalité quand une correspondance a été trouvée, ou la valeur NULL dans le cas de non correspondance.

RIGHT OUTER JOIN or RIGHT JOIN permet d'obtenir toutes les nationalités même celles pour lesquels on ne trouve pas d'acteurs, les acteurs sans nationalités sont ignorés.

FULL OUTER JOIN or FULL JOIN permet d'obtenir toutes les nationalités et tous les acteurs avec dans le cas de non correspondances de colonnes NULL.

-

Activité n° 3 : à la recherche du temps des rois

Réfléchissez sur papier avant de vous lancer sur écran
Passez les ordres SQL vous permettant de répondre aux demandes ci-dessous.
Conservez votre script dans un fichier se trouvant dans votre dossier de travail. Pensez à sauvegarder votre base.

1 Sélection de tous les rois capétiens

2 Sélection de tous les rois carolingiens affichés par montant de trésor

3 Sélection des épouses par ordre décroissant de date de mariage

4 Sélection de tous les rois ayant eu plus d'une épouse

5 Combien y-a-t-il en moyenne d'enfants par dynastie ? (vérifiez bien le résultat obtenu)

6 Quel est le montant total du trésor des rois ayant eu plus d'une épouse ?

7 Augmentez le trésor des carolingiens de 70 %

8 Remettez le trésor des rois capétiens à zéro.

En résumé

SELECT DISTINCT liste des attributs résultats

FROM liste des tables

WHERE conditions de prise en compte

GROUP BY critères de regroupement

HAVING conditions sur regroupements

ORDER BY critères de tri des résultats

UNION

SELECT .....

SELECT colonnes ⇗ *PROJECTION*

permet de définir les colonnes résultats

(colonnes de tables ou fonctions statistiques)

SELECT DISTINCT ➔ élimination des doublons

SELECT * ➔ sélectionne toutes les colonnes

Fonctions statistiques ➔ SUM(colonne) : cumul

➔ AVG(colonne) : moyenne

➔ COUNT(*) : comptage lignes

➔ MIN(colonne) : plus petite valeur

➔ MAX(colonne) : plus grande valeur

FROM tables ⇗ *PRODUIT CARTESIEN*

permet d'indiquer les tables

➔ où les colonnes suivant le SELECT se trouvent

➔ où les colonnes permettant les jointures se trouvent

WHERE conditions ⇗ *RESTRICTION*

permet d'indiquer des critères de prise en compte des lignes des tables se trouvant derrière FROM

Opérateurs ➔ = , < , > , <= , >= , <>

➔ AND , OR, NOT,

➔ IN , BETWEEN , LIKE

GROUP BY colonnes

permet d'effectuer des calculs par groupe de lignes sélectionnées

Les critères doivent être des colonnes de la projection

➔ on obtient une ligne résultat par critère

HAVING conditions ⇗ *RESTRICTION sur groupe*

équivalent à WHERE mais pour des lignes issues d'un GROUP BY

ORDER BY colonnes

Trie les lignes de la table resultat

➔ ASC : croissant ↗

➔ DESC : décroissant ↘

Comment arriver à une requête bien formulée ?

Enumération des données à utiliser pour arriver au résultat souhaité

- colonnes souhaitées en résultat : celles à afficher
- tables à l'origine de ces résultats

⇩

Indication des colonnes derrière la clause SELECT

Ajout de ces tables derrière la clause FROM

- fonctions statistiques souhaitées en résultat
- tables à l'origine de ces résultats

⇩

Indication des fonctions statistiques derrière la clause SELECT

Ajout de ces tables derrière la clause FROM

Ajout des critères de regroupement derrière la clause GROUP BY

- tables permettant le cheminement
- colonnes permettant le cheminement entre les différentes tables

⇩

Ajout de ces tables derrière la clause FROM

Indication des critères de jointure derrière la clause WHERE

(concordance des colonnes entre tables)

Indication des critères de restriction de prise en compte

- Quelles sont les lignes à prendre en compte pour obtenir le résultat ?

⇩

Ajout des critères de restriction derrière la clause WHERE

- Quels sont les critères sur les fonctions statistiques pour obtenir le résultat

⇩

Ajout des critères de restriction derrière la clause HAVING

Définition de la séquence de présentation des résultats

א Dans quel ordre veut-on voir apparaître les lignes résultat

⇩

Ajout des critères de tri derrière la clause ORDER BY

7 Compléments sur les chapitres précédents

7.1 Remplissage de colonnes par SELECT

Il est possible d'extraire des données d'une ou plusieurs tables et de les insérer dans une table préalablement créée par Create table.

Structure de la commande en remplissant avec des colonnes issues d'un SELECT
Pour insérer des valeurs dans toutes les colonnes :

```
INSERT INTO  <nom table>
        SELECT ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;
```

Il faut indiquer derrière le SELECT autant de noms de zones que de colonnes de la table réceptrice et les mettre dans le même ordre.

Exemple de commande

```
Insert into NATIONALITES
        SELECT CodPays, DesigFrançaise
                FROM REPERTPAYS
```

Par contre, si nous ne désirons alimenter que quelques colonnes de la table, nous pouvons utiliser cette structure de commande.

```
insert into  <nom table> (Col1, Col2, Coln)
        SELECT orig1, orig2, orig5 from ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;
```

Exemple de commande

```
create table tab1        (NomFilm char (20), NomActeur char (20),
                          PourcentSatisf smallint default 0)
Insert into tab1 (NomFilm, Nomacteur)
  Select TitreFilm, Nomacteur from ACTEURS,FILMS, PARTICIPATION
        where  ACTEURS.NumActeur = PARTICIPATION.NumActeur
               and FILMS.CodFilm     = PARTICIPATION.CodFilm
```

7.2 Création de vue CREATE VIEW

Une vue est une table virtuelle créée à partir d'autres tables. Elle peut permettre :

- une vision réduite sur des données de table :
  exemple : vue sur un e table salarié ne reprenant pas les éléments constituant le salaire.
- Une vision de plusieurs tables jointes.

Une vue s'utilise ensuite comme une table.

Structure de la commande :

```
CREATE VIEW nom-vue [(column [, ...n])]
        [WITH ENCRYPTION]
        AS
                select_……..
                [WITH CHECK OPTION]
```

Exemple de commande

```
Create view Vue1 (NomFilm, Nomacteur) AS
        Select TitreFilm, Nomacteur from ACTEURS,FILMS,
PARTICIPATION
                where ACTEURS.NumActeur =
PARTICIPATION.NumActeur
            and FILMS.CodFilm     = PARTICIPATION.CodFilm
```

Le SELECT se trouvant derrière le Create View ne peut contenir de ORDER BY mais peut être avec UNION.

7.3 Modification de vue ALTER VIEW

Une vue peut être modifiée sans passer par DROP VIEW suivi de CREATE VIEW ce qui obligerait à remettre en état les autorisations sur cette vue.
Alter View permet principalement de modifier le SELECT qui permet d'alimenter la vue

Structure de la commande :

```
ALTER  VIEW nom-vue [(column [, ...n])]
        [WITH ENCRYPTION]
        AS
                select_……..
                [WITH CHECK OPTION]
```

Exemple de commande

```
Alter view Vue1 (NomFilm, Nomacteur) AS
  Select TitreFilm, Nomacteur
        from ACTEURS a ,FILMS f , PARTICIPATION p
        where a.NumActeur = p.NumActeur
                and f.CodFilm   = p.CodFilm
         and a.Cachet <> null
```

7.4 Suppression de vue DROP VIEW

Une vue peut être supprimée par DROP VIEW : tout ce qui concerne la vue disparaît mais les tables concernées restent intactes.

Structure de la commande :

```
DROP  VIEW nom-vue
```

7.5 Création d'index CREATE INDEX

C'est un accélérateur pour les accès.
"Clustered" indique que physiquement les lignes de la table sont rangées selon l'ordre indiqué dans l'index.
Un index 'nonclustered' n'influe pas l'ordre physique de rangement : il s'agit d'un ordre logique.

Un index doit être créé qu'après étude sur les performances car plus il y a d'index et plus les temps de réponse s'effondrent.

Structure de la commande en remplissant avec des valeurs

```
CREATE UNIQUE CLUSTERED INDEX <nom index>
        On <nom table> (col1, col2, … coln)
```

Exemples de commande

```
CREATE INDEX Acteurs_pays
        ON ACTEURS (CodNationalite)
```

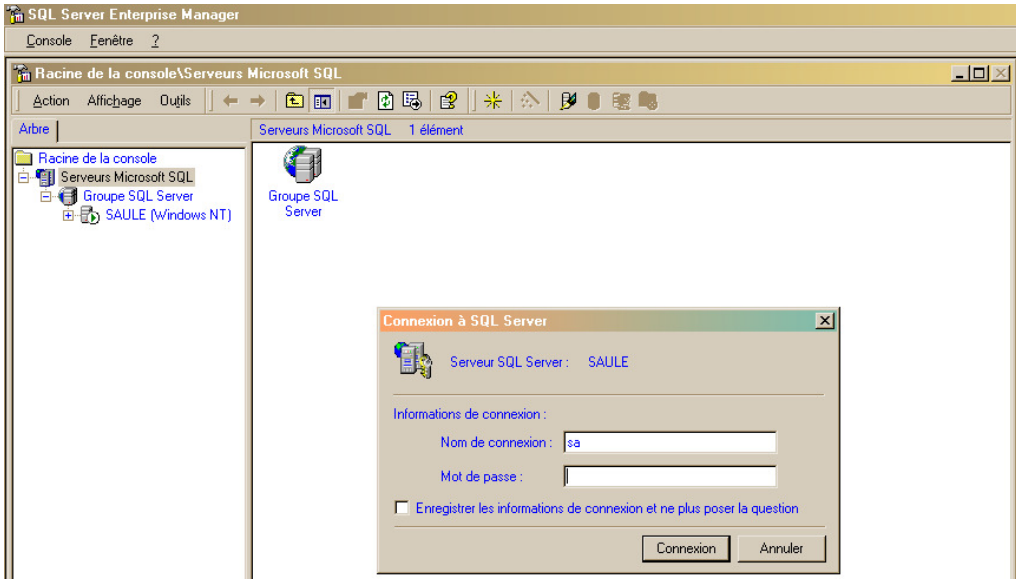
```
CREATE UNIQUE CLUSTERED INDEX Acteurs_nom
        ON ACTEURS (NomActeur, PrenomActeur, DatNaissActeur)
```

8 Enterprise Manager

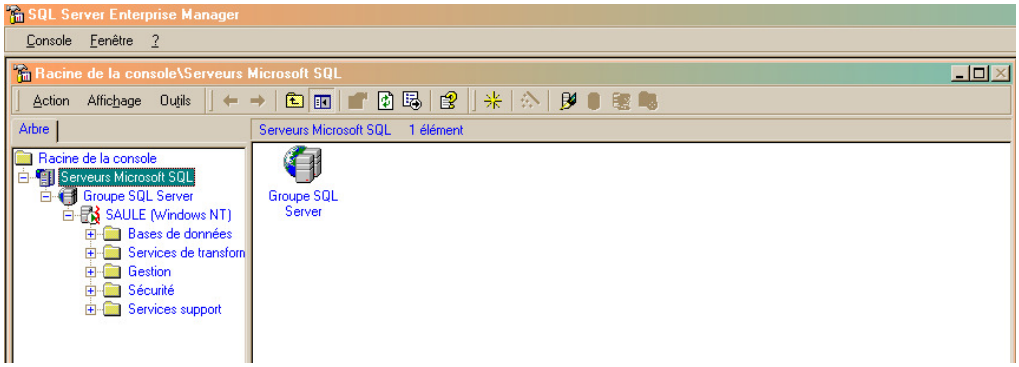
Jusqu'à maintenet nous vous avons fait utiliser l'analyseur de requêtes car lorsque vous programmez il vous est nécessaire d'écrire les requêtes. Il existe un outil vous permettant d'utiliser de nombreux assistants : c'est Enterprise Manager.

Pour y accéder, cliquez sur "Enterprise manager" : vous accédez à une fenêtre comportant dans sa partie gauche une liste à développer (cliquez sur +).

Un écran pour authentification peut apparaître : indiquez le mot de passe de la connexion.

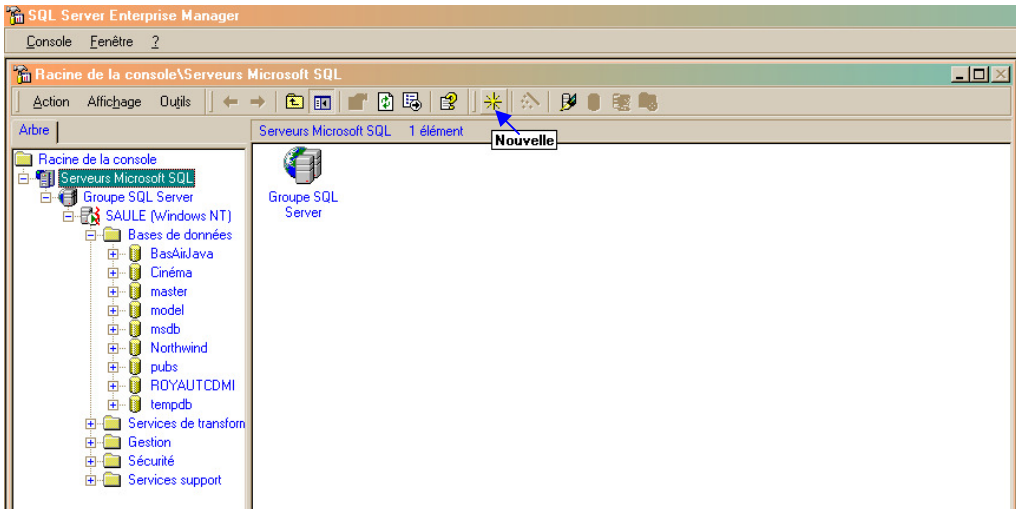


Vous obtiendrez un niveau supplémentaire dans l'arborescence.

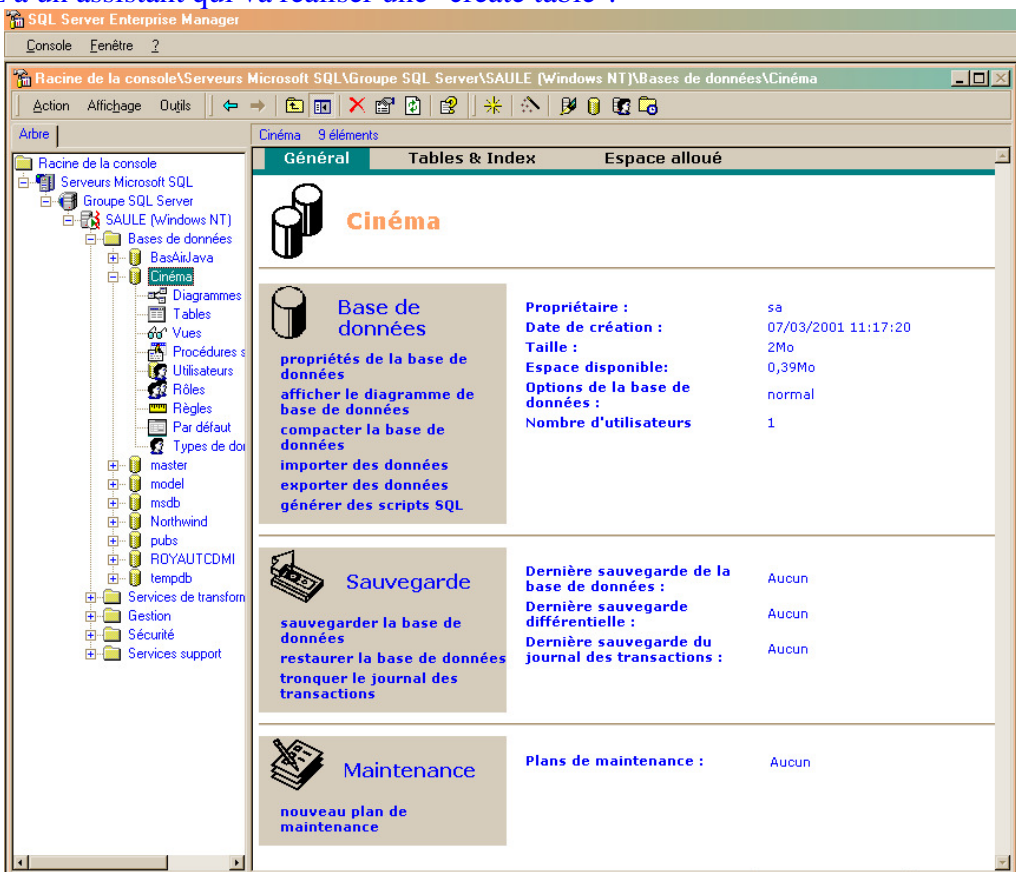


En cliquant sur "Bases de données" vous développerez un niveau supplémentaire faisant apparaître toutes les bases de données actuellement répertoriées.

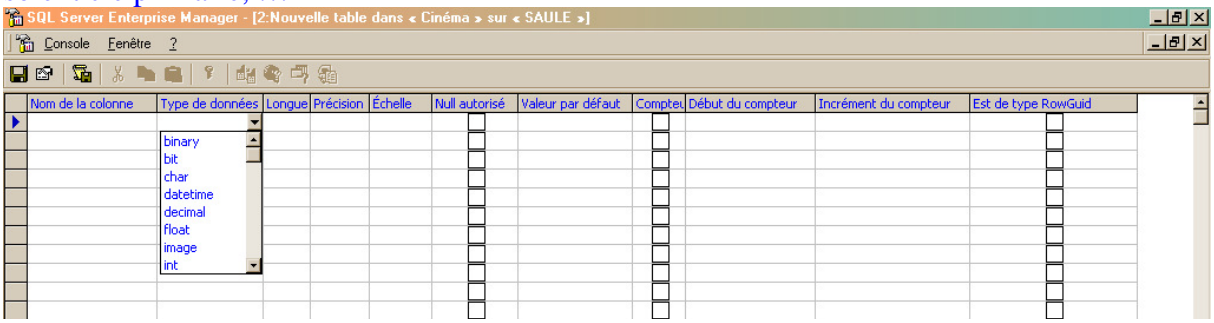
Lorsque vous êtes sur la liste des bases de données, en cliquant sur l'icône "Nouvelle", vous accédez à un assistant qui va réaliser une "create database".



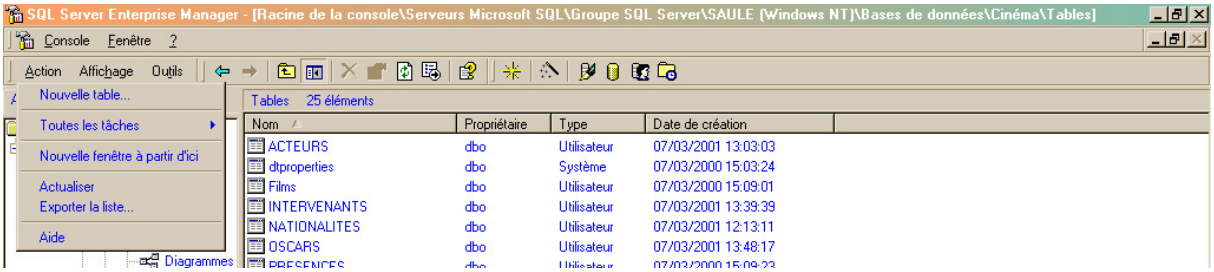
Lorsque vous êtes sur une base de données précise, , en cliquant sur l'icône "Nouvelle", vous accédez à un assistant qui va réaliser une "create table".



Un tableau apparaît où il est possible de saisir les noms de colonnes, leur type, le fait qu'elles soient cle primaire, …

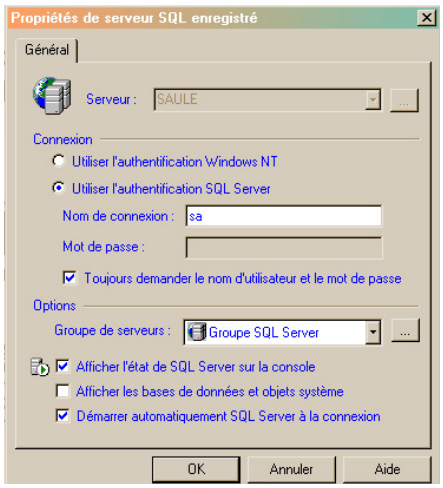


Vous pouvez passer par les options du memu action pour réaliser la même opération.

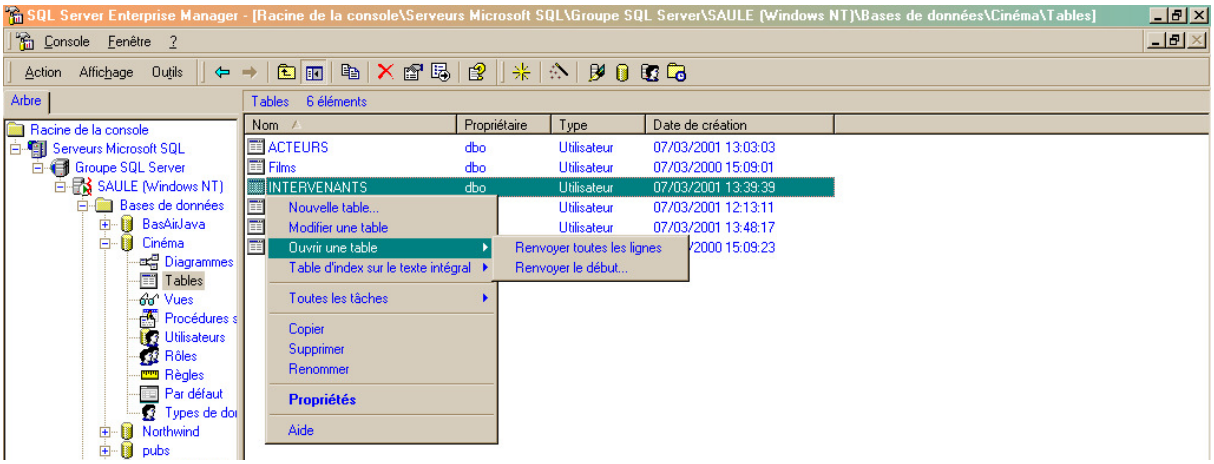


Il est génant de voir figurer les objets sytème dans les listes de tables, d'index, …
Pour y remédier modifier les propriétés de votre server SQL : cliquez droit sur le serveur (dans nos exemple c'est Saule) puis choisissez l'option "propriétés"

Désélectionnez " afficher les base de données et objets Systèmes".



Par un clic doit sur un nom de table, vous obtiendrez un menu permettant en autre d'entrer et de modifier le contenu des colonnes vous évitant ainsi les Insert into, Update et Delete.



9 Programmation en SQL

9.1 Création de type de données

Lorsque l'on crée une base de données, on utilise les types de données standards, mais il est fréquent que certaines données est la même structure (par exemple, toutes les données montant sont numériques de longueur 11 dont 2 décimales et peuvent ne pas être définies). Dans ce cas, il est pratique d'ajouter aux types de données standard des types de données "utilisateur" spécifiques à notre base. Ceci est possible par la procédure sp-addtype.

Structure de la commande

```
sp_addtype   <nomtype> ,  <typedonnée standard' ,    'NULLou Not Null'
```

.

Exemple de commande

```
Exec sp_addtype   Montant  , ' numeric (11 2)' ,    'NULL'

Create table STATIST (CodNationalite   char (3) NOT NULL,
                      TotalN1          Montant,
                      TotalN2          Montant)
```

9.2 Création de TRIGGER

Un Trigger est une procédure stockée dans les objets de la base et qui se déclenche automatiquement dès qu'une action de mise à jour est lancée sur la table (Insert, Update, Delete).

Ils sont très utiles pour renforcer la sécurité et l'intégrité des données.

On peut avoir plusieurs triggers par table.

Ils sont écrits en Transac-SQL.

Structure de la commande :

```
CREATE TRIGGER <nom-déclencheur>  ON  <nomtable>
        FOR (INSERT, UPDATE, DELETE)
        AS
                conditions de déclenchement
                instructions SQL
```

Exemple de commande

```
USE cinéma

IF EXISTS (SELECT name FROM sysobjects
                WHERE name = 'Pastouche' AND type = 'TR')
        DROP TRIGGER Pastouche
GO

CREATE TRIGGER pastouche ON ACTEURS
FOR DELETE AS
        declare         @detruit        int
        SELECT          @detruit = NumActeur        from deleted
        IF EXISTS (SELECT NumActeur FROM PARTICIPATION
                        WHERE NumActeur = @detruit)
        RAISERROR('Pas touche à %d : il participe encore', 10, 1, @detruit)
GO
```

Détaillons cette procédure :

- USE : permet de se mettre dans le contexte "cinéma"
- SELECT name FROM sysobjects WHERE name = 'Pastouche' AND type = 'TR')
  permet de chercher si le trigger (TR) de nom 'pastouche' existe
- If EXIST …… DROP TRIGGER Pastouche
  permet de supprimer le trigger Pastouche si le Select l'a trouvé

- CREATE TRIGGER pastouche ON ACTEURS FOR DELETE
  crée une procédure qui se déclenchera automatiquement lorsqu'un ordre DELETE surviendra sur la table ACTEURS
- Declare @detruit int
  permet de se réserver un emplacement mémoire de type entier qui se nommera pour la suite @detruit
- SELECT @detruit = NumActeur from deleted
  alimente notre variable avec le numéro d'acteur contenu par la ligne sur lequel DELETE est lancé.
- SELECT NumActeur FROM PARTICIPATION WHERE NumActeur = @detruit)
  recherche l'acteur à supprimer dans PARTICIPATION
- IF EXISTS ……
  RAISERROR('Pas touche à %d : il participe encore', 10, 1, @detruit
  permet d'envoyer un message signalant l'erreur détectée en remplaçant dans le texte du message %d par le contenu de @detruit

9.3 Suppression de TRIGGER

Un Trigger peut être supprimé par DROP TRIGGER.

Remarque : le fait de supprimer une table supprime les triggers qui lui sont attachés de façon automatique

Structure de la commande :

```
DROP TRIGGER  <nom-déclencheur
```

9.4 Modification de TRIGGER

Un Trigger est une procédure stockée dans les objets de la base et qui se déclenche automatiquement dès qu'une action de mise à jour est lancée sur la table (Insert, Update, Delete).

Ils sont très utiles pour renforcer la sécurité et l'intégrité des données.

On peut avoir plusieurs triggers par table.

Ils sont écrits en Transac-SQL.

Structure de la commande :

```
ALTER TRIGGER  <nom-déclencheur>   ON   <nomtable>
        FOR (INSERT, UPDATE, DELETE)
        AS
                conditions de déclenchement
                instructions SQL
```

9.5 Instructions de contrôle de flux :

9.5.1 /*...*/ (Comment)

Permet de commenter la procédure par un texte qui ne sera pas pris en compte lors de l'exécution.

9.5.2 BEGIN...END

Encadre un bloc d'instructions constituant un groupe au moment de l'exécution.

Syntaxe

```
BEGIN
   Groupe d'instructions SQL
END
```

9.5.3 DECLARE @local_variable

Permet de déclarer les variables dans le corps d'une procédure .c'est à dire de se réserver un emplacement mémoire nommé de façon unique. On utilise ensuite cet emplacement mémoire en utilisant le nom de la variable qui correspond pour la machine à une adresse précise. A la suite du DECLARE, la valeur contenue par la variable est NULL

Syntaxe

```
DECLARE @nomvar  type
```

Exemple :

```
DECLARE @msg varchar(100)
```

9.5.4 SET @local_variable

Permet d'alimenter le contenu d'une variable

Syntaxe

```
SET @nomvar  = expression
```

Exemple :

```
SET @msg = 'Coucou, c'est moi'
```

9.5.5 IF...ELSE

Permet de conditionner l'exécution de certaines instructions

Syntaxe

IF condition

Groupe d'instructions SQ L exécutées dans le cas de la condition vraie

ELSE

Groupe d'instructions SQ L exécutées dans le cas de la condition fausse

Exemple :

Si les dépense de salaire sont inférieures à 1000000, augmentation de 25 % du salaire horaire. Dans le cas contraire, l'augmentation sera de + 15 pour tout le monde.

```
IF (select sum(SalHoraire * NbHeures) from presences
      where CodFilm = 'Mon oeuvre')  < 1000000
    BEGIN
        update presences set SalHoraire = SalHoraire * 1.25
    END
ELSE
    BEGIN
        update presences set SalHoraire = SalHoraire * 15
END
```

9.5.6 WHILE

Permet d'exécuter un bloc d'instruction tant qu'une condition est réalisée.

Syntaxe

WHILE condition

Instruction ou bloc d'instructions SQL}

9.5.7 BREAK

Permet de sortir d'une boucle WHILE même si les conditions de fin ne sont pas réalisées

9.5.8 CONTINUE

Fait repartir l'exécution à l'instruction WHILE en ignorant les instructions qui se trouvent derrière CONTINUE

Exemple :

Tant que les dépense de salaire sont inférieures à 1000000, augmentation de 1 % du salaire horaire.

```
While (select sum(SalHoraire * NbHeures) from presences
where  CodFilm = 'Mon oeuvre')  < 1000000
        BEGIN
                update presences set SalHoraire = SalHoraire * 1.02
        END
```

9.5.9 CASE

Permet de sortir d'une procédure sans exécuter la suite des instructions

Structure de la commande

```
CASE expression sur une donnée connue
        WHEN condition1 THEN résultat1
        WHEN condition2 THEN résultat2
        WHEN conditionn THEN résultatn
[
        ELSE résultat si toute les autres conditions sont fausses
]
END
```

Exemple :
Au lieu de sortir les salaires, on souhaite avoir une liste avec la notion de petit salaire (si < 1000), de moyen salire si compris entre 1000,01 et 7000, de gros salaire jusqu'à 10000. On considère qu'au dessus de 10000, on se trouve hors budget.

```
Select 'TRanche salaire' =
        CASE
                When SalHoraire * NbHeures < 1000  then 'petit salaire'
                When SalHoraire * NbHeures < 7000  then 'moyen salaire'
                When SalHoraire * NbHeures < 10000 then 'gros salaire'
        else                                    'hors budget'
        END,
Codfilm, NumInter    from presences
```

Le résultat est une liste dont les colonnes sont :

```
                Tranche salaire     Codfilm     NumInter
--------------- ---------- --------
```

9.5.10 GOTO étiquette

Permet un branchement inconditionnel à une étiquette.

9.5.11 RETURN

Permet de sortir d'une procédure sans exécuter la suite des instructions

9.6 Qu'est-ce qu'un curseur

Dans tout ce qui a précédé, nous avons obtenu un résultat global, nous n'avons pas pu faire de traitement ligne par ligne. Il est évident que la machine traite ligne par ligne mais elle ne vous redonne la main que lorsqu'elle a été du début à la fin de sa démarche.

Prenons le SELECT … CASE que nous avons vu juste avant, on ne peut exécuter une autre commande que lorsque le SELECT est terminé.

Pour pouvoir exécuter d'autres commandes entre chaque ligne, nous utiliserons un curseur.
Il existe différentes étape s à suivre pour utiliser un curseur:

① le déclarer (**declare cursor**) afin de définir quel sera son contenu
Declare cursor contient un ordre SELECT qui défini les données auxquelles on souhaitent accéder

② l'ouvrir (**open**)comme vous le feriez pour un tiroir quand vous voulez aller voir ce qu'il contient.

③ le consulter (**fetch**) : cet ordre permet de passer à la suite de l'extraction demandée par le SELECT c'est à dire d'accéder ligne par ligne aux lignes issues du SELECT.

④ le fermer (**close**) normal, on l'avait ouvert avant.

9.6.1 DECLARE CURSOR

Dans l'exemple qui suit, nous déclarons un **curseur statique** qui permet d'accéder à toutes les colonnes des lignes de la table Acteurs qui ont un numéro > 100

```
DECLARE MonCurseur CURSOR FOR
        SELECT * from  FROM aActeursuthors WHERE NumActeur < 100;
```

Le second exemple est plus compliqué, car il met en jeu un **curseur dynamique** c'est à dire que toutes les informations constituant le SELECT ne sont pas connues une fois pour toutes, elles sont paramétrables : derrière le Where se trouve un point d'interrogation qui représente une inconnue lors de l'écriture de la procédure. L'endroit où se trouve le point d'interrogation sera complété au moment de l'exécution : c'est plus souple mais au combien plus coûteux donc à utiliser avec parcimonie (même si vous ne lui avez pas été présenté).

```
EXEC SQL BEGIN DECLARE SECTION;
    char szCommand[] = "SELECT au_fname FROM authors
        WHERE au_lname = ?";
    char szLastName[] = "White";
    char szFirstName[30];
EXEC SQL END DECLARE SECTION;

EXEC SQL
    DECLARE curseurd CURSOR FOR selection1

EXEC SQL
    PREPARE selection1 FROM :szCommand;
```

9.6.2 OPEN

Ouvre le curseur pour que l'on puisse accéder aux lignes issues du SELECT.

9.6.3 FETCH

Permet d'accéder à une ligne depuis le curseur.

- FETCH FIRST récupère la première ligne issue du SELECT déclaré dans le curseur.
- FETCH NEXT récupère la ligne suivant celle qui vient d'être lue via le curseur
- FETCH PRIOR récupère la ligne précédant celle qui vient d'être lue via le curseur
- FETCH LAST récupère la dernière ligne issue du SELECT déclaré dans le curseur.
- FETCH ABSOLUTE *n* récupère la n^{ième} ligne issue du SELECT via le curseur.
- FETCH RELATIVE *n* recupère la n^{ième} ligne suivant celle qui vient d'être lue via le curseur

9.6.4 CLOSE

Ferme le curseur.

9.6.5 DEALLOCATE

Libère les ressources qui étaient liées au curseur (déverrouillage et libération pour d'autres utilisations)

Exemple

Pour obtenir la liste des acteurs ayant un cachet < 500000

```
declare @CachetMaxi int
declare @NomActeur varchar(30)
set @CachetMaxi = 500000

  DECLARE Mon_Cursor CURSOR FOR
  SELECT NomActeur from Acteurs a, Participation p, Films f
       where a.NumActeur = p.NumActeur
        and P.CodFilm   = f.Codfilm
        and a.Cachet    < @cachetMaxi;
  OPEN  Mon_Cursor ;

  FETCH  Mon_Cursor  INTO @NomActeur;

  WHILE @@FETCH_STATUS = 0
   BEGIN
     PRINT 'le nom de l''acteur est : ' + @NomActeur
     FETCH NEXT FROM Mon_Cursor
   END

  CLOSE Mon_Cursor
  DEALLOCATE Mon_Cursor

GO
```

9.7 CREATE PROCEDURE

Plutôt que de re-saisir les instructions SQL, à chaque fois que nous avons à les exécuter, il est possible de les stocker dans une ***procédure stockée*** grâce à la commande CREATE procédure suivie d'un nom et de l'ensemble des commandes SQL constituant un ensemble d'actions à exécuter toujours dans un même processus.
Il suffit ensuite d'en lancer l'exécution par EXECUTE ou directement en donnant leur nom.

Drop Procedure permet de supprimer une procédure stockée si vous êtes autorisé à supprimer.
Alter Procédure permet de modifier le contenu d'une procédure stockée.

9.8 Quelques procédures utiles

9.8.1 sp_depends

Exemple
Pour obtenir la liste des objets qui dépendent de la table Acteurs de la base 'cinéma'.

```
USE cinéma
EXEC sp_depends 'Acteurs'
```

Résultat

```
In the current database, the specified object is referenced by the following:
name                                                    type
_____________________________________________________________
dbo.Vue1                                                view

(1 row(s) affected)
```

9.8.2 sp_addmessage

Ajoute un nouveau message d'erreur à la table "**sysmessages**".

Syntaxe :

```
sp_addmessage {Numéro, sévérité, 'texte msg'} [, 'language'] [, 'with_log'] [, 'replace']
```

Exemple

Pour obtenir la liste des objets qui dépendent de la table Acteurs de la base 'cinéma'.

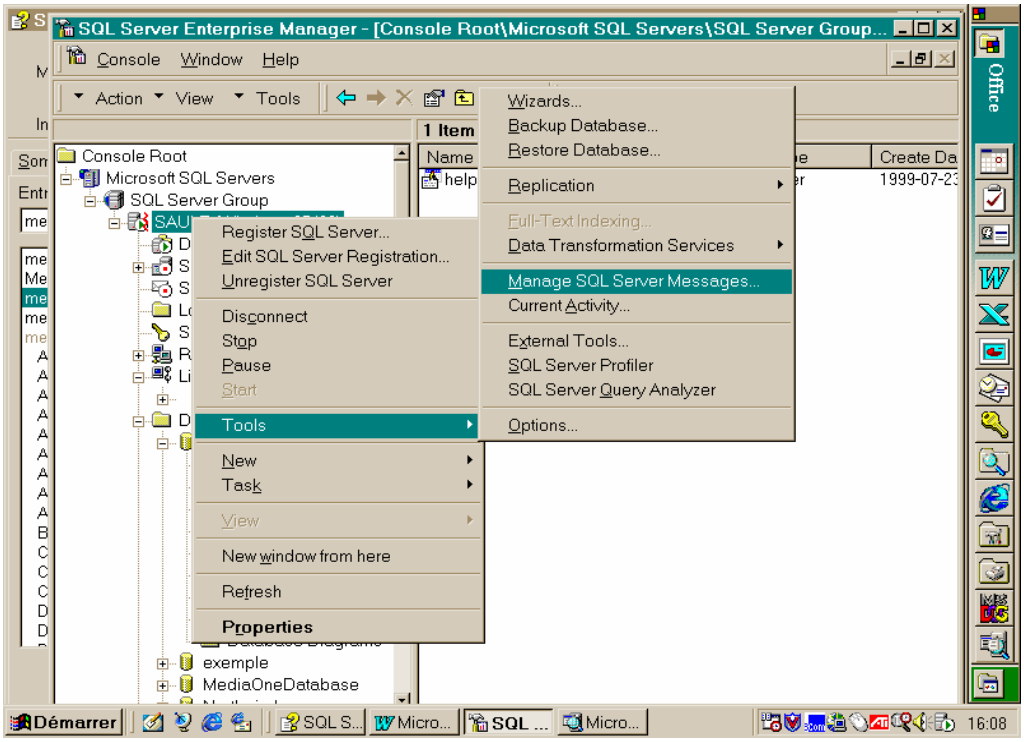
```
USE cinéma
declare @numerr int
declare @texterr nvarchar (255)
set @numerr = 52000
Set @texterr = 'Il vaut mieux retirer ses moufles avant de taper sur le clavier'
EXEC sp_addmessage @numerr, 16, @texterr
```

Résultat

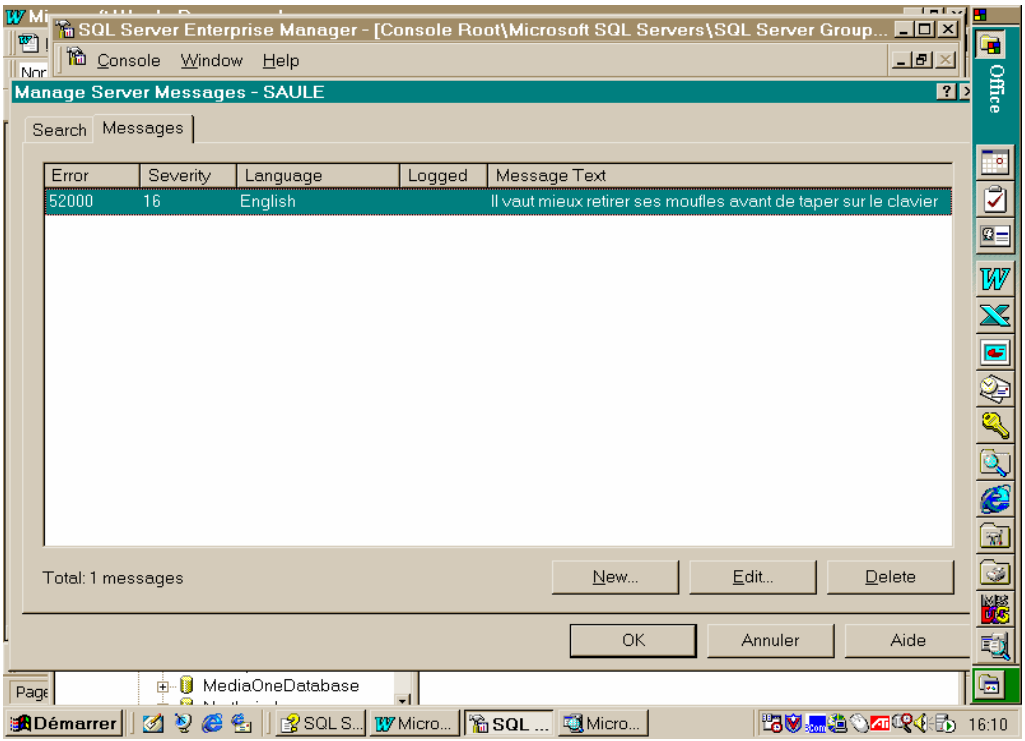
```
(1 row(s) affected)

New message added.
```

Pour obtenir les message, sous Entreprise
cliquer droit sur votre serveur
Puis choisir 'tools'



Cliquer sur "manager Server Messages pour obtenir l'écran de gestion des messages



10 Activités

Cas PAPYRUS
ou
la gestion des consommables

10.1 Cas PAPYRUS

Le souci majeur de M. PURCHASE, chef de la production informatique de la société BIDOUILLE EXPRESS, est d'assurer la gestion et le suivi des produits *consommables* tels que :

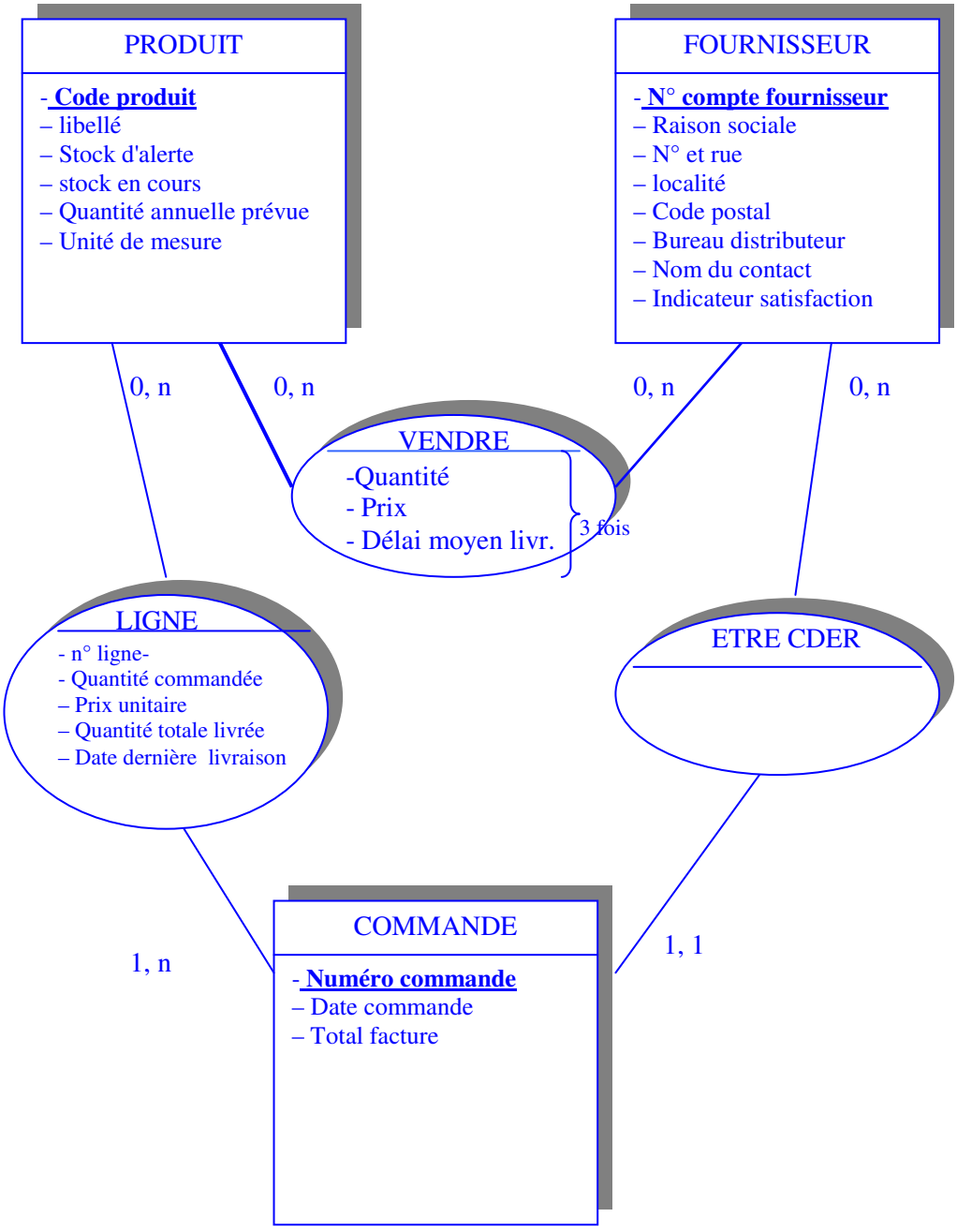
- papier listing en continu sous toutes ses formes,
- papier pré-imprmé (commandes, factures, bulletins paie, ...)
- rubans pour imprimantes
- bandes magnétiques,
- disquettes,
- .....

Pour chacun de ces produits, il existe plusieurs fournisseurs possibles ayant déjà livré la société ou avec lesquels M. PURCHASE est en contact De plus, de nombreux représentants passent régulièrement vanter leurs produits et leurs conditions de vente : ceci permet à M. PURCHASE de conserver leurs coordonnées pour d'éventuelles futures commandes ou futur appels d'offre.

Un appel d'offre se matérialise par un envoi de courrier précisant la nature de la demande (type de consommable, quantité prévisible de la commande, quantité annuelle, délai de livraison courant, délai de livraison en cas de rupture de stock, ...) aux fournisseurs susceptibles de faire une offre valable.
En retour, les fournisseurs intéressés par le marché renvoient leurs conditions de vente et M. PURCHASE peut faire son choix.

La commande est envoyée au fournisseur pour l'achat de un ou plusieurs produits pour une quantité donnée. Cette quantité peut être livrée en plusieurs fois. Les seules informations mémorisées sur la livraison sont la date de dernière livraison ainsi que la quantité livrée totale.

Modèle conceptuel des données



Les besoins :

a) Quelles sont les commandes du fournisseur "09120" ?

b) Quelles sont les commandes du fournisseur "GROBRIGAN" ?

c) Quelles sont les commandes des fournisseurs situés dans les départements 75 78 92 77 ?

d) Sortir les produits des commandes ayant "commande urgente' en observation et dont le fournisseur est situé dans l'un des départements suivants : 75 78 92 77.

e) Editer les produits ayant un stock inférieur ou égal au stock d'alerte (informations à fournir : n° produit libellé produit stock actuel stock d'alerte).

f) Editer la liste des fournisseurs susceptibles de livrer les produit dont le stock est inférieur ou égal à 150 % du stock d'alerte. La liste est triée par produit puis fournisseur.

g) Editer la liste des fournisseurs susceptibles de livrer les produit dont le stock est inférieur ou égal à 150 % du stock d'alerte et un délai de livraison d'au plus 30 jours . La liste est triée par fournisseur puis produit.

h) Avec le même type de sélection que ci-dessus, sortir un total des stocks par fournisseur trié par total décroissant.

i) En fin d'année, sortir la liste des produits dont la quantité annuelle prévue est inférieure de 10 % à la quantité réellement commandée. (les deux premières positions du numéro de commande représente l'année de la commande)

j) Calculer le chiffre d'affaire par fournisseur pour l'année 93 sachant que les prix indiqués sont hors taxes et que le taux de TVA est 20,60%.

k) Existe-t-il des lignes de commande non cohérentes avec les produits vendus par les fournisseurs. ?

Exemple base de données physique

PRODUIT	Nom colonne	Libellé	Type	longueur
	CODART	Code produit	caractère	4
	LIBART	Libellé produit	caractère	25
	STKALE	Stock d'alerte	numérique	4 0
	STKPHY	Stock physique	numérique	4 0
	QTEANN	Quantité annuelle	numérique	4 0
	UNIMES	Unité de mesure	caractère	5

Code	*Libellé*	*Stock* alerte	*Stock* en-cours	*Qté* annuelle	*Unité* mes.
I100	Papier 1 ex continu	100	557	3500	B1000
I105	Papier 2 ex continu	75	5	2300	B1000
I108	Papier 3 ex continu	200	557	3500	B500
I110	Papier 4 ex continu	10	12	63	B400
P220	Préimprimé commande	500	2500	24500	B500
P230	Préimprimé facture	500	250	12500	B500
P240	Préimprimé bulletin paie	500	3000	6250	B500
P250	Préimprimé bon livraison	500	2500	24500	B500
P270	Préimprimé bon fabrication	500	2500	24500	B500
R080	Ruban Epson 850	10	2	120	unité
R132	Ruban imp1200 lignes	25	200	182	unité
B002	Bande magnétique 6250	20	12	410	unité
B001	Bande magnétique 1200	20	87	240	unité
D035	Disquette 3/5	40	42	150	B010
D005	Disquette 5	50	4	0	B010

ENTCOM	Nom colonne	Libellé	Type	longueur
	NUMCOM	Numéro de commande	caractère	6
	OBSCOM	Observations	caractère	25
	DATCOM	Date de commande	date	
	NUMFOU	Numéro de compte fournisseur	Numérique	5 0

Numéro commande	*Observation commande*	*Date commande*	*N° compte* fournisseur
930010		10/02/93	00120
930011	Commande urgente	01/03/93	00540
930020		25/04/93	09180
930025	Commande urgente	30/04/93	09150
930210	Commande cadencée	05/05/93	00120
930300		06/06/93	09120
930250	Commande cadencée	02/10/93	08700
930620		02/10/93	00540
930625		09/10/93	00120
930629		12/10/93	09180

FOURNIS	Nom colonne	Libellé	Type	longueur
	NUMFOU	Numéro de compte fournisseur	Numérique	5 0
	NOMFOU	Nom fournisseur	Caractère	30
	RUEFOU	Adresse	Caractère	30
	POSFOU	Code postal fournisseur	Numérique	5 0
	VILFOU	Ville fournisseur	Caractère	25
	CONFOU	Contact chez fournisseur	Caractère	15
	SATISF	Indice satisfaction	Numérique	1 0

N° compte fournisseur	*Raison sociale*	*Adresse*	*Nom* Contact	*indice* satisfaction
00120	GROBRIGAN	20 rue du papier	Georges	08
		92200 papercity		
00540	ECLIPSE	53, rue laisse flotter les rubans	Nestor	07
		78250 Bugbugville		
08700	MEDICIS	120 rue des plantes	Lison	
		75014 Paris		
09120	DISCOBOL	11 rue des sports	Hercule	08
		85100 La Roche sur Yon		
09150	DEPANPAP	26, avenue des locomotives	Pollux	05
		59987 Coroncountry		
09180	HURRYTAPE	68, boulevard des octets	Track	
		04044 Dumpville		

VENTE	Nom colonne	Libellé	Type	longueur
	CODART	Code produit	caractère	4
	NUMFOU	Numéro de compte fournisseur	Numérique	5 0
	DELLIV	Délai de livraison	Numérique	2 0
	QTE1	Borne Quantité livraison 1	Numérique	2 0
	PRIX1	Prix unitaire 1	Numérique	3 0
	QTE2	Borne Quantité livraison 2	Numérique	2 0
	PRIX2	Prix unitaire 2	Numérique	3 0
	QTE3	Borne Quantité livraison 3	Numérique	2 0
	PRIX3	Prix unitaire 3	Numérique	3 0

Code produit	*N° cpt fourn.*	*Délai livr.*	*Oté 1*	*Prix 1*	*Oté 2*	*Prix 2*	*Oté 3*	*Prix 3*
I100	00120	90	0	700	50	600	120	500
I100	00540	70	0	710	60	630	100	600
I100	09120	60	0	800	70	600	90	500
I100	09150	90	0	650	90	600	200	590
I100	09180	30	0	720	50	670	100	490
I105	00120	90	10	705	50	630	120	500
I105	00540	70	0	810	60	645	100	600
I105	09120	60	0	920	70	800	90	700
I105	09150	90	0	685	90	600	200	590
I105	08700	30	0	720	50	670	100	510
I108	00120	90	5	795	30	720	100	680
I108	09120	60	0	920	70	820	100	780
I110	09180	90	0	900	70	870	90	835
I110	09120	60	0	950	70	850	90	790
D035	00120	0	0	40				
D035	09120	5	0	40	100	30		
I105	09120	8	0	37				
D035	00120	0	0	40				
D035	09120	5	0	40	100	30	5	0
I105	09120	8	0	37				
P220	00120	15	0	3700	100	3500		
P230	00120	30	0	5200	100	5000		
P240	00120	15	0	2200	100	2000		
P250	00120	30	0	1500	100	1400	500	1200
P250	09120	30	0	1500	100	1400	500	1200
P220	08700	20	50	3500	100	3350		
P230	08700	60	0	5000	50	4900		
R080	09120	10	0	120	100	100		
R132	09120	5	0	275				
B001	08700	15	0	150	50	145	100	140
B002	08700	15	0	210	50	200	100	185

LIGCOM	Nom colonne	Libellé	Type	longueur
	NUMCOM	Numéro de commande	caractère	6
	NUMLIG	Numéro de ligne commande	numérique	2 0
	CODART	Code produit	caractère	4
	QTECDE	Quantité commandée	Numérique	5 0
	PRIUNI	Prix unitaire de vente	Numérique	4 0
	QTELIV	Quantité livrée	Numérique	5 0
	DERLIV	Date dernière livraison	Date	

N° commande	*N° Lig*	*Produit*	*Quantité cdée*	*Prix Unitaire*	*Qté livrée*	*Dernière Livraison*
930010	01	I100	3000	470	3000	15/03/93
930010	02	I105	2000	485	2000	05/07/93
930010	03	I108	1000	680	1000	20/08/93
930010	04	D035	200	40	250	20/02/93
930010	05	P220	6000	3500	6000	31/03/93
930010	06	P240	6000	2000	2000	31/03/93
930011	01	I105	1000	600	1000	16/05/93
930020	01	B001	200	140		31/12/93
930020	02	B002	200	140		31/12/93
930025	01	I100	1000	590	1000	15/05/93
930025	02	I105	500	590	500	15/05/93
930210	01	I100	1000	470	1000	15/07/93
930010	02	P220	10000	3500	10000	31/08/93
930300	01	I110	50	790	50	31/10/93
930250	01	P230	15000	4900	12000	15/12/93
930250	02	P220	10000	3350	10000	10/11/93
930620	01	I105	200	600	200	01/02/94
930625	01	I100	1000	470	1000	15/10/93
930625	02	P220	10000	3500	10000	31/10/93
930629	01	B001	200	140		31/12/93
930629	02	B002	200	140		31/12/93

Les requêtes possibles :

a) Quelles sont les commandes du fournisseur "09120" ?

Question ?		Requête
Quelles sont les colonnes à afficher en résultat ?		**SELECT numcom**
De quelles tables sont issues ces colonnes		SELECT numcom **FROM entcom**
Y-a-t-il jointures ?	non	
N'y-a-t-il que certaines lignes à prendre en compte ?		SELECT numcom FROM entcom **WHERE numfou = 09120**
Veut-on un résultat par « paquets » de lignes ?	non	
Veut-on voir apparaître les résultats selon un ordre précis ?	non	
N'y-a-t-il que certaines lignes résultats à prendre en compte ?	non	

```
SELECT numcom FROM entcom
WHERE numfou = 09120
```

b) Quelles sont les commandes du fournisseur "GROBRIGAN" ?

Question ?		Requête
Quelles sont les colonnes à afficher en résultat ?		**SELECT numcom**
De quelles tables sont issues ces colonnes		SELECT numcom **FROM entcom**
Quelles sont les colonnes permettant la sélection des lignes ?		Choix de certains noms de fournisseur : nomfou
De quelles tables sont issues ces colonnes		SELECT numcom FROM entcom, **fournis**
Y-a-t-il jointures ?	oui	SELECT numcom FROM entcom, fournis WHERE entcom.numfou = fournis.numfou
N'y-a-t-il que certaines lignes à prendre en compte ?		SELECT numcom FROM entcom, fournis WHERE entcom.numfou = fournis.numfou **and nomfou = 'GROBRIGAN'**
Veut-on un résultat par « paquets » de lignes ?	non	
Veut-on voir apparaître les résultats selon un ordre précis ?	non	
N'y-a-t-il que certaines lignes résultats à prendre en compte ?	non	

```
SELECT numcom FROM entcom, fournis
    WHERE entcom.numfou = fournis.numfou
        and nomfou = 'GROBRIGAN'
```

c) Quelles sont les commandes des fournisseurs situés dans les départements 75 78 92 77 ?

Question ?		Requête
Quelles sont les colonnes à afficher en résultat ?		**SELECT numcom**
De quelles tables sont issues ces colonnes		SELECT numcom **FROM entcom**
Quelles sont les colonnes permettant la sélection des lignes ?		Choix de certains départements : 2 premières positions du code postal COSPOS du fournisseur
De quelles tables sont issues ces colonnes		SELECT numcom FROM entcom, **fournis**
Y-a-t-il jointures ?	oui	SELECT numcom FROM entcom, fournis WHERE entcom.numfou = fournis.numfou
N'y-a-t-il que certaines lignes à prendre en compte ?		SELECT numcom FROM entcom, fournis WHERE entcom.numfou = fournis.numfou **and ((fournis.posfou between 75000 and 75999)** **or (fournis.posfou between 78000 and 78999)** **or (fournis.posfou between 91000 and 91999)** **or (fournis.posfou between 92000 and 92999))**
Veut-on un résultat par « paquets » de lignes ?	non	
Veut-on voir apparaître les résultats selon un ordre précis ?	non	
N'y-a-t-il que certaines lignes résultats à prendre en compte ?	non	

```
SELECT  numcom   FROM entcom , fournis
            WHERE entcom.numfou = fournis.numfou
                and  ((fournis.posfou between 75000 and 75999)
                  or  (fournis.posfou between 78000 and 78999)
                  or  (fournis.posfou between 91000 and 91999)
                  or  (fournis.posfou between 92000 and 92999))
```

d) Sortir les produits des commandes ayant "commande urgente' en observation et dont le fournisseur est situé dans l'un des départements suivants : 75 78 92 77.

Question ?		Requête
Quelles sont les colonnes à afficher en résultat ?		**SELECT codart**
De quelles tables sont issues ces colonnes		SELECT codart **FROM ligcom**
Quelles sont les colonnes permettant la sélection des lignes ?		Choix de certains départements : 2 premières positions du code postal COSPOS du fournisseur Choix de certaines commandes en testant OBSCOM
De quelles tables sont issues ces colonnes		SELECT codart FROM ligcom, **entcom, fournis**
Y-a-t-il jointures ?	oui	SELECT codart FROM ligcom, entcom, fournis WHERE **entcom.numfou = fournis.numfou** **and entcom.numcom = ligcom.numcom**
N'y-a-t-il que certaines lignes à prendre en compte ?	oui	SELECT codart FROM ligcom, entcom, fournis WHERE entcom.numfou = fournis.numfou and entcom.numcom = ligcom.numcom **and ((fournis.posfou between 75000 and 75999)** **or (fournis.posfou between 78000 and 78999)** **or (fournis.posfou between 91000 and 91999)** **or (fournis.posfou between 92000 and 92999))** **and entcom.obscom = 'Commande urgente'**
Veut-on un résultat par « paquets » de lignes ?	non	
Veut-on voir apparaître les résultats selon un ordre précis ?	non	
N'y-a-t-il que certaines lignes résultats à prendre en compte ?	non	

```
SELECT codart FROM ligcom, entcom, fournis WHERE entcom.numfou = fournis.numfou
and entcom.numcom = ligcom.numcom and ((fournis.posfou between 75000 and 75999)
or (fournis.posfou between 78000 and 78999) or (fournis.posfou between 91000 and 91999)
or (fournis.posfou between 92000 and 92999)) and entcom.obscom = 'Commande urgente'
```

Autre forme pour la même question :

Question ?		Requête
Quelles sont les colonnes à afficher en résultat ?		**SELECT codart**
De quelles tables sont issues ces colonnes		SELECT codart **FROM ligcom**
Quelles sont les colonnes permettant la sélection des lignes ?		Choix de certains départements : 2 premières positions du code postal COSPOS du fournisseur Choix de certaines commandes en testant OBSCOM
De quelles tables sont issues ces colonnes		SELECT **numfou** FROM **fournis** SELECT **numcom** FROM **entcom**
Y-a-t-il jointures ?	oui	SELECT codart FROM ligcom WHERE **numcom IN** (SELECT numcom FROM entcom WHERE **numfou IN** (SELECT **numfou** FROM **fournis)**
N'y-a-t-il que certaines lignes à prendre en compte ?		SELECT codart FROM ligcom WHERE **numcom IN** (SELECT numcom FROM entcom WHERE **obscom = 'Commande urgente'** **and numfou IN** (SELECT **numfou** FROM **fournis)** **WHERE ((fournis.posfou between 75000 and 75999)** **or (fournis.posfou between 78000 and 78999)** **or (fournis.posfou between 91000 and 91999)** **or (fournis.posfou between 92000 and 92999))))**
Veut-on un résultat par « paquets » de lignes ?	non	
Veut-on voir apparaître les résultats selon un ordre précis ?	non	

```
SELECT  codart  FROM   ligcom   WHERE numcom IN (
          SELECT numcom FROM entcom   WHERE  obscom = 'Commande urgente'
                    and numfou IN (SELECT numfou  FROM fournis)
                    WHERE  ((fournis.posfou between 75000 and 75999)
                              or  (fournis.posfou between 78000 and 78999)
                             or  (fournis.posfou between 91000 and 91999)
                             or  (fournis.posfou between 92000 and 92999))))
```

e) Editer les produits ayant un stock inférieur ou égal au stock d'alerte (informations à fournir : n° produit libellé produit stock stock actuel d'alerte).

Question ?		Réponse
Quelles sont les colonnes à afficher en résultat ?		SELECT **codart, libart, skphy, stkale**
De quelles tables sont issues ces colonnes		SELECT codart, libart, stkphy, stkale FROM **produit**
Quelles sont les colonnes permettant la sélection des lignes ?		stkphy <= stkale
De quelles tables sont issues ces colonnes		
Y-a-t-il jointures ?	non	
N'y-a-t-il que certaines lignes à prendre en compte ?		SELECT codart, libart, stkphy, stkale FROM produit **WHERE stkphy <= stkale**
Veut-on un résultat par « paquets » de lignes ?	non	
Veut-on voir apparaître les résultats selon un ordre précis ?	non	

```
SELECT codart, libart, stkphy, stkale FROM produit
WHERE stkphy <= stkale
```

f) Editer la liste des fournisseurs susceptibles de livrer les produit dont le stock est inférieur ou égal à 150 % du stock d'alerte. La liste est triée par produit puis fournisseur.

En décomposant la question, on voit qu'il faut déterminer :

a) les fournisseurs qui vendent de produits donnés

b) puis trouver les produits dont le stock physique est inférieur à 150 % du stock d'alerte

Pour a)

Question ?		Réponse
Quelles sont les colonnes à afficher en résultat ?		SELECT **codart, numfou, nomfou**
De quelles tables sont issues ces colonnes		SELECT codart, fournis.numfou, nomfou FROM **fournis, vente**
Quelles sont les colonnes permettant la sélection des lignes ?		codart dans une liste d'articles donnés (voir § b)
Y-a-t-il jointures ?	oui	WHERE fournis.numfou = vente.numfou
N'y-a-t-il que certaines lignes à prendre en compte ?		SELECT codart, fournis.numfou, nomfou FROM **fournis, vente** WHERE **fournis.numfou = vente.numfou** **and vente.codart** IN (§ b)
Veut-on un résultat par « paquets » de lignes ?	non	
Veut-on voir apparaître les résultats selon un ordre précis ?	oui	ORDER BY **codart, numfou**

b)

Question ?		Réponse
Quelles sont les colonnes à afficher en résultat ?		SELECT **codart**
De quelles tables sont issues ces colonnes		SELECT codart FROM **produit**
Quelles sont les colonnes permettant la sélection des lignes ?		stkphy <= (stkale * 150 / 100)
Y-a-t-il jointures ?	non	
N'y-a-t-il que certaines lignes à prendre en compte ?		SELECT codart FROM **produit** **WHERE** stkphy <= (stkale * 150 / 100)

```
SELECT codart, fournis.numfou, nomfou FROM fournis, vente
        WHERE  fournis.numfou = vente.numfou
         and  vente.codart  IN  (SELECT codart  FROM produit
                                 WHERE stkphy  <= (stkale * 150 /
100))
ORDER BY codart, numfou
```

g) Editer la liste des fournisseurs susceptibles de livrer les produit dont le stock est inférieur ou égal à 150 % du stock d'alerte et un délai de livraison d'au plus 30 jours . La liste est triée par fournisseur puis produit.

Nous suivrons un raisonnement similaire au cas précédant : pour a)

Question ?		Réponse
Quelles sont les colonnes à afficher en résultat ?		SELECT **numfou, codart, nomfou**
De quelles tables sont issues ces colonnes		SELECT numfou, codart, nomfou FROM **fournis vente**
Quelles sont les colonnes permettant la sélection des lignes ?		codart dans une liste d'articles donnés (voir § b) et délai de livraison < 31
De quelles tables sont issues ces colonnes		
Y-a-t-il jointures ?	oui	WHERE fournis.numfou = vente.numfou
N'y-a-t-il que certaines lignes à prendre en compte ?		SELECT codart, fournis.numfou, nomfou FROM **fournis, vente** WHERE **fournis.numfou = vente.numfou** **and vente.delliv < 31** **and vente.codart** IN (§ b)
Veut-on voir apparaître les résultats selon un ordre précis ?	oui	ORDER BY fournis.**numfou, codart**

b)

Question ?		Réponse
Quelles sont les colonnes à afficher en résultat ?		SELECT **codart**
De quelles tables sont issues ces colonnes		SELECT codart FROM **produit**
Quelles sont les colonnes permettant la sélection des lignes ?		stkphy <= (stkale * 150 / 100)
De quelles tables sont issues ces colonnes		
Y-a-t-il jointures ?	non	
N'y-a-t-il que certaines lignes à prendre en compte ?		SELECT codart FROM **produit** **WHERE** stkphy <= (stkale * 150 / 100)

```
SELECT codart, fournis.numfou, nomfou FROM fournis, vente
        WHERE  fournis.numfou = vente.numfou
        and  vente.delliv  < 31
         and  vente.codart  IN  (SELECT codart  FROM produit
                                 WHERE stkphy  <= (stkale * 150 /
100))
ORDER BY fournis.numfou, codart
```

h) Avec le même type de sélection que ci-dessus, sortir un total de stock par fournisseur trié par total décroissant.

```
SELECT fournis.numfou, vente.codart, nomfou, sum(stkphy)
        FROM fournis, vente, produit
        WHERE  fournis.numfou = vente.numfou
              and vente.codart      = produit.codart
         and  vente.delliv  < 31
         and  stkphy  <= (stkale * 150 / 100))
GROUP BY fournis.numfou, vente.codart, nomfou
ORDER BY sum(stkphy)  desc
```

i) En fin d'année, sortir la liste des produits dont la quantité annuelle prévue est inférieure de 10 % à la quantité réellement commandée en 93.

i1) Détermination des quantités réellement commandées :

Question ?		Réponse
Quelles sont les colonnes à afficher en résultat ?		SELECT **codart, sum(qtecde)**
De quelles tables sont issues ces colonnes		SELECT codart, sum(qtecde)FROM **ligcom**
Quelles sont les colonnes permettant la sélection des lignes ?		substr(numcom, 1,2) = '93'
De quelles tables sont issues ces colonnes		
Y-a-t-il jointures ?	non	
N'y-a-t-il que certaines lignes à prendre en compte ?		SELECT codart, sum(qtecde) FROM **ligcom** WHERE substr(numcom, 1,2) = '93'
Veut-on un résultat par « paquets » de lignes ?	oui	GROUP BY codart
Veut-on voir apparaître les résultats selon un ordre précis ?		

```
SELECT codart, sum(qtecde)   FROM ligcom
          WHERE  substr(numcom, 1,2) = '93'
GROUP BY codart
```

i2)

Question ?		Réponse
Quelles sont les colonnes à afficher en résultat ?		SELECT codart, **qteann -**(sum(qtecde) * 0.9)
De quelles tables sont issues ces colonnes		SELECT produit.codart**, qteann,** sum(qtecde) FROM **produit,** ligcom
Quelles sont les colonnes permettant la sélection des lignes ?		substr(numcom, 1,2) = '93'
De quelles tables sont issues ces colonnes		
Y-a-t-il jointures ?	oui	produit.codart = ligcom.codart
N'y-a-t-il que certaines lignes à prendre en compte ?		SELECT produit.codart, **qteann ,** (sum(qtecde) * 90 / 100) FROM **produit,** ligcom WHERE substr(numcom, 1,2) = '93' and produit.codart = ligcom.codart
Veut-on un résultat par « paquets » de lignes ?	oui	GROUP BY produit.codart
Veut-on voir apparaître les résultats selon un ordre précis ?	non	car idem au group by
N'y-a-t-il que certaines lignes résultats à prendre en compte ?	oui	HAVING min(qteann) < -sum(qtecde) * 0.9

On utilise min (qteann) pour comparer deux données de niveau récap.

```
SELECT produit.codart, min(qteann) ,  (sum(qtecde) * 0.9)
        FROM produit, ligcom
        WHERE  substr(numcom, 1,2) = '93'
              and produit.codart = ligcom.codart
GROUP BY produit.codart, qteann
HAVING  min(qteann) -< sum(qtecde) * 0.9
```

j) Calculer le chiffre d'affaire par fournisseur pour l'année 93 sachant que les prix indiqués sont HT et que le taux de TVA est 20,60%.

Question ?		Réponse
Quelles sont les colonnes à afficher en résultat ?		SELECT **fournis.numfou, nomfou, sum(qtecde * priuni * 1,2060)**
De quelles tables sont issues ces colonnes		SELECT numfou, nomfou, sum(qtecde * priuni * 1,2060) FROM **fournis, entcom, ligcom**
Quelles sont les colonnes permettant la sélection des lignes ?		substr(numcom, 1,2) = '93'
De quelles tables sont issues ces colonnes		
Y-a-t-il jointures ?	oui	entcom.numcom = ligcom.numcom and fournis.numfou = entcom.numfou
N'y-a-t-il que certaines lignes à prendre en compte ?		SELECT fournis.numfou, nomfou, sum(qtecde * priuni * 1,2060) FROM fournis, entcom, ligcom WHERE substr(numcom, 1,2) = '93' and entcom.numcom = ligcom.numcom and fournis.numfou = entcom.numfou
Veut-on un résultat par « paquets » de lignes ?	oui	GROUP BY fournis.numfou, nomfou
Veut-on voir apparaître les résultats selon un ordre précis ?	non	car idem au group by
N'y-a-t-il que certaines lignes résultats à prendre en compte ?	non	

```
SELECT fournis.numfou, nomfou, sum(qtecde * priuni * 1,2060)
FROM  fournis, entcom, ligcom
 WHERE  substr(numcom, 1,2) = '93' and
                entcom.numcom = ligcom.numcom and
                fournis.numfou   = entcom.numfou
GROUP BY fournis.numfou, nomfou
```

k) Existe-t-il des lignes de commande non cohérentes avec les produits vendus par les
fournisseurs. ?

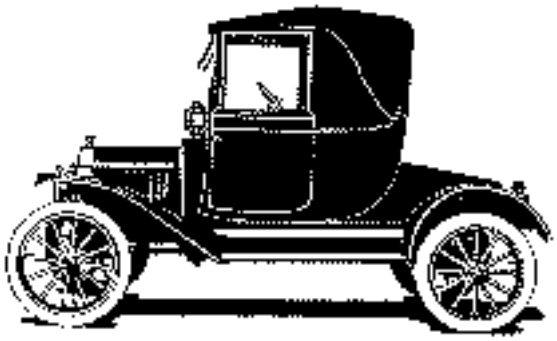
C'est à dire que nous voulons extraire les lignes de commandes d'aticle chez des fornisseurs qui ne les vendent pas.

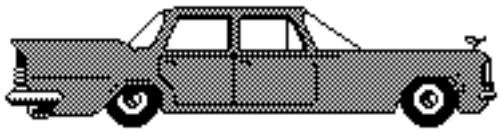
⇒ Création d'une table avec le numéro de fournisseur, le code article et un témoin numérique

```
CREATE TABLE mystere       (numfou numeric (5, 0) not null with
default,
                           codart character (4) not null with
default,
                           témoin numeric (2, 0) not null with
default)
```

```
INSERT INTO mystere   SELECT DISTINCT numfou, codart, 1
                           FROM vente
                      UNION ALL
                      SELECT DISTINCT entcom.numfou,
ligcom.codart, 10
                           FROM entcom, ligcom
                           WHERE entcom.numcom =
ligcom.numcom
```

```
SELECT numfou, codart, sum(temoin)  FROM mystere
        GROUP BY numfou, codart
        HAVING sum(témoin) = 10
```





Cas TACOT
ou
Gestion de parc de stationnement

10.2 Cas TACOT

Mr TURBO est gérant d'un parc de stationnement assez important (2500 places). Il est équipé d'un ordinateur où une base de données a été implantée. Au fil des jours, il se pose toujours les mêmes questions . Le travail qui vous est demandé est d'indiquer à Mr TURBO les requêtes lui permettant d'obtenir les réponses à ces questions.

Le contexte :

Le parking contient donc 2500 places réparties en

- parking en plein air : 600 places voitures et 100 places poids lourds
- parking couvert : 1900 places situées sur 3 niveaux.

Chaque place à un numéro l' identifiant de façon unique.

Les clients sont de deux types :

- les occasionnels laissant leur véhicule à l'heure ou à la journée.
- les abonnés réservant une place donnée pour le mois ou pour l' année.

Pour les abonnés le n° d'immatriculation de leur véhicule est noté comme attribué à une place donnée.
Pour rendre un meilleur service à ses abonnés, Mr TURBO permet que plusieurs véhicules puissent stationner sur une place donnée bien sûr pas au même moment mais cela permet aux membres d'une même famille de se parquer au même endroit (Monsieur dans la semaine et Madame en fin de semaine par exemple).

Les tarifs varient selon la place et le temps d'occupation : exemples

- véhicule stationné en plein air pendant 1 heure : 5 francs
- Véhicule stationné en couvert pendant 1 heure : 7 francs
- Véhicule stationné en plein air pendant plus de 3 heures : 12 francs
- Poids lourds stationné en plein air pendant 1 heure : 35 francs
- Poids lourds stationné en plein air pendant plus de 3 heures : 100 francs
- Véhicule stationné en plein air pendant 1 jour : 20 francs
- Véhicule stationné en plein air pendant plus de 1 mois : 240 francs
- Poids lourds stationné en plein air pendant 1 jour : 120 francs
- Poids lourds stationné en plein air pendant 1 mois : 1000 francs
- Véhicule stationné en plein air pendant de 1 an: 2500 francs

- Poids lourds stationné en plein air pendant 1 an : 10000 francs

Les véhicules appartiennent à un propriétaire qui est :

- soit une personne
- soit une société.

Mr TURBO à trois employés avec qui il assure l'ouverture 24h sur 24 du parking. Un local à l'entrée leur permet

- de délivrer au conducteur non abonné un carton avec l'heure d'entrée.
- d'encaisser les sommes dûes en espèce, en chèque ou en carte bleue à la sortie des véhicules

dans les cas des occasionnels ou de percevoir les abonnements sinon.

Ayant eu de nombreux incidents de paiement; Mr TURBO renseigne régulièrement son fichier abonné par un top "mauvais payeur" (O pour mauvais payeur N sinon).

Structure des tables

ABONNES

- N° abonné
- Nom ou raison sociale
- Prénom
- Code postal
- Ville
- Télephone en cas d'incident
- Type de personne (P personne physique ou S pour personne morale = société)

PLACE

- N° place
- Type place (plein air ou couvert)
- Catégorie véhicule (T pour tourisme P pour poids lourds)

ABONNEMENT

- N° abonné
- N° place
- Date début abonnement
- Date fin abonnement
- Montant payé
- Type paiement (E = espèces, B = chèque bancaire, C = carte bleue, P = chèque postal)
- N° chèque ou carte bleue
- Nom banque

VEHICULES

- N° véhicule
- Marque
- Modéle
- Couleur
- Catégorie véhicule (T pour tourisme P pour poids lourds)

OCCASIONNELS

N° véhicule

N° place

Date début stationnement

Date fin stationnement

Heure début stationnement

Heure fin stationnement

Montant payé

Type paiement (E = espèces, B = chèque bancaire, C = carte bleue, P = chèque postal)

N°chèque ou carte bleue

Nom banque

Les besoins :

a) Mr Turbo souhaite savoir quelles sont les places occupées par des abonnements au mois de mars de l'année en cours ?

b) Il souhaite aussi obtenir le total des montants de chèques par semaine pour contrôler sa remise en banque. Indiquez lui la ou les requêtes pour la semaine qui vient de s'écouler.

c) De plus, il souhaite connaitre la ventilation des sommes perçues par catégorie (véhicule ou Poids lourds), par type (personnes physiques ou sociétés ou véhicules occasionnels) en détaillant par plein air ou couvert.

- réaliser la maquette de l'état.
- par quelle(s) requête(s) pouvez-vous l'obtenir.

La fête au village

10.3 Cas FÊTE AU VILLAGE

Pour avoir plus de poids au niveau régional, trois communes se sont associées pour leur gestion (en particulier pour la gestion de leurs associations).

- Revienzy : 13800 habitants
- Oucesty : 28850 habitants
- Cesparlà : 11230 habitants.

Les différentes associations ont donc fusionnées par activité :

- Club "Barbouille" offre des activités artistiques;
- Club "Pieds nickelés" offre des activités foot, rugby,...
- Club "Musclor" offre des activités danse et gymnastique
- Club "Age d'or" offre des activités au troisième age (voyages, tricot,)
- Club "Petit futé" assure les activités enfantines.
- Club "Grosse Bouffe" organise des dîners dansants

Il faut savoir qu'avant la fusion, deux communes s'étaient équipées chacune d'un PC 486 avec WORD et EXCEL la troisième d'un AS/400 avec SQL.

Comme partout les problèmes budgétaires sont à l'ordre du jour et les communes souhaitent obtenir des résultats consolidés de manière à gérer leurs associations de la façon la plus efficace possible..

Chacun des clubs fonctionnent sensiblement pareil. Prenons l'exemple du club "Barbouille".

- l'achat des fournitures se fera en commun puis répartit sur chaque association.
- l'enregistrement et le suivi des cotisations des adhérents seront suivis par commune.

Les achats :

Ils sont de deux types :

- les achats de type investissement (local, véhicules, ...)

- les achats courants (toile, peinture, pinceaux, ...)

Les cotisations :

Elles sont également de deux types :

- les cotisations de base perçues en janvier pour les adhérents déjà inscrits et à l'adhésion lors de l'arrivée de nouvel adhérent.

- les cotisations exceptionnelles versées par les entreprises quand leurs salariés sont adhérents (leur cotisations de base est plus faible en fonction des accords entre l'entreprise et l'association).

Des rentrées exceptionnelles ont lieu quelques fois dans l'année : ce sont les bénéfices dégagés lors d'expositions des œuvres du club.

Le président de toutes les associations, Jean Voeu souhaite éditer mensuellement et annuellement :

- un bilan d'activité pour chaque village et ceci détaillé pour chaque association

- un bilan d'activité pour chaque association tous villages confondus.

- un bilan global pour toutes les associations.

Les adhérents de chacune des associations reçoivent périodiquement des bulletins d'informations sur l'association à laquelle ils adhèrent. Il faut donc pouvoir sortir des étiquettes pour les personnes concernées (1 par famille).

De plus, des invitations sont envoyées avec demande de réponse pour les dîners dansants, les expositions, ...Chaque "invité" doit payée une participation. Dans ce cas, ce qui est souhaitée, c'est l'édition d'une étiquette par personne ainsi qu'une liste des personnes ayant répondu oui, une liste des personnes ayant répondu non et une liste des personnes n'ayant pas répondu concernant une manifestation donnée.
La première de ces listes devra indiquer le montant de la participation par commune et au total

Pour les manifestations, chaque association souhaite un classement par montant de déficits ou profits (du plus gros déficit au meilleur profit), ce montant étant l'écart entre la recette prévue et le recette effective.

Mr Jean Voeu connaissant vos compétences vous demande de lui propose une solution pour automatiser tout çà (à titre bénévole, bien sûr).

Bon prince vous ne pouvez qu'accepter ! Retroussez vos neurones et sortez nous un dossier digne de Super-SQL. avec :

- les maquettes d'état

- le mode opératoire pour les obtenir.

- les commentaires que vous jugez utiles.

Proposition de structure des tables

ASSOCIATION

- Code association	A	10
- Dénomination	A	30
- Montant cotisation individuelle	N	5

ACHATS

- Numéro demande achat	N	7
- Quantité commandée	N	5
- Prix unitaire Hors taxe	N	7,2
- Date achat	D	
- Type achat (I investissement C courant)	A	1

REPACHATS

- Numéro demande achat	N	7
- Code association A	10	
- Montant Hors taxeN	7,2	

COTISATIONS

- Numéro adhérent	N	7
- Nom adhérent	A	20
- Prénom adhérent	A	12
- Adresse 1	A	30
- Adresse 2	A	30
- Code postal	N	5
- Commune	A	25
- Type adhérent (I individuel S société)	A	1
- Numéro société	N	7

SOCIETE

- Numéro société	N	7
- Raison sociale	A	20
- Prénom adhérent	A	12
- Adresse 1	A	30
- Adresse 2	A	30
- Code postal	N	5
- Commune	A	25

ADHERENT- ASSOCIATION

- Numéro adhérent	N	7
- Code association	A	10
- Montant cotisation individuelle	N	5
- Date de paiement cotisations	N	7,2

SOCIETE - ASSOCIATION

- Numéro société	N	7
- Code association	A	10
- Montant cotisation exceptionnelle	N	5
- Date de paiement cotisations	N	7,2

MANIFESTATION

- Code association	A	10
- Manifestation	A	20
- Montant participation individuelle	N	5
- Date manifestation	D	
- Date fin réservation	D	
- Code postal commune	N	5
- Nombre de participants prévus	N	5

PARTICIPATION- MANIFESTATION

- Code association		A	10
- Manifestation		A	20
- Numéro adhérent		N	7
- Code réponse	O = oui N = non ' ' = pas de réponse	A	1


FIN DOCUMENT