**Notes - Chapitre 3**

**Les Requêtes**

Cégep régional de Lanaudière à Joliette

Hiver 2020

## Introduction

### Qu’est-ce qu’une requête

* En Access, les requêtes serviront pour :
* Consulter et modifier les données
* Générer des données pour un formulaire ou un état
* On rencontrera des requêtes à différents endroits dans les interfaces d’Access. Il sera donc utile de sauvegarder une requête en tant qu’objet Access
* Un objet requête est essentiellement une instruction SQL.
* Access sauvegarde celle-ci :
  + sous un nom
  + stocke l’instruction SQL sous format de texte
  + stocke quelques valeurs de propriété de "présentation" de la requête

NORME 3

Un nom de requête sera préfixé par req.

### Langage SQL

* Il ne comporte que quelques instructions distinctes qui tiennent lieu de :
* DDL : Langage de Définition de Données
  + Create Table
  + Create Database
* DML : Langage de Manipulation de Données (le seul vu à cette session)
  + Select
  + Update
  + Delete
  + Insert
* DCL : Langage de Contrôle de Données
* Vous verrez les DDL et DCL à la prochaine session

|  |
| --- |
| Access |
| Interfaces utilisateurs |

|  |
| --- |
| Microsoft Jet Engine |

|  |
| --- |
| Stockage de données |

* Très intuitive pour construire (définir) des requêtes SQL, MS Jet Engine est le nom du moteur de BD qui constitue Access SQL qui interprète le langage SQL. Access SQL ne présente que quelques nuances par rapport à ANSI SQL ’92.

### Interface QBE Voir Apprentissage 3

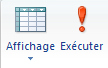
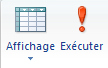
#### Query By Example :

* En français : Requête par l’Exemple
* Interface de la fenêtre de création de requêtes.
* Tout utilisateur d’Access peut se servir de QBE à profit!
* Excellent outil d’apprentissage du langage SQL pour nous.

#### Fenêtre de création de requête

* Dans le volet du haut :
* On place les données impliquées, c’est-à-dire les tables ou requêtes
* On identifie les jointures
* Ceci constitue une vue logique de nos données, ou encore un schéma de données pour la requête.
* Dans le volet du bas on identifie :
* Les champs (colonnes) à utiliser, ou à être affiché (projections)
* Les conditions à rencontrer (restrictions)
* Les tris à faire
* Les regroupements à faire
* Les fonctions à appliquer

#### Exécution d’une requête

* Par l’outil , ou bien par ou menu contextuel puis Mode feuille de données.
* On obtient alors une feuille de réponses dynamique (sous forme de feuille de données).
* Tous les enregistrements résultants sont générés temporairement en mémoire centrale.

Note

Microsoft Jet appelle cela un **Recordset**

Selon le type de requête faite, il est possible de modifier les données dans cette feuille de réponse. **Les enregistrements des tables impliquées sont alors modifiés.**

## Requêtes de « sélection » voir Document 1 pour syntaxe globale

* Mon approche sera d’amener, à travers des exemples d’instructions SQL et d’explications sur l’interface *création de requête*, la grande majorité des possibilités d’un SELECT en SQL.

### Syntaxe minimale

SELECT < liste de champs >  
FROM < liste de tables ou requêtes>;

* **ReqA** (table H, chapitre 2 document 1)

Seulement utilisé par Access, ne vient pas de SQL

SELECT ActNom  
FROM Acteur;

* Résultats :

|  |
| --- |
| ActNom |
| Birkins |
| Martins |
| Fonda |
| Pitt |
| Fonda |
| Martins |

NOTE

Dans une instruction SQL, il ya une boucle implicite sur tous les enregistrements de la seule table présenté, ou sur tous les enregistrements du produit cartésien des tables présentes dans la clause FROM

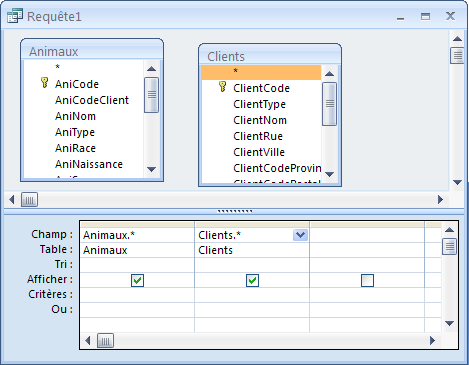
* Pour ReqA, on peut formuler :
* Pour tous les enregistrements de la table Acteurs faire  
   Afficher la valeur du champ ActNom  
  finPour
* **ReqB**

SELECT \*  
FROM Acteurs;

* L’astérisque représente tous les champs, donc équivalent à :

SELECT ActCode, ActNom, ActPrenom,…, ActNaissance  
FROM Acteurs;

* **ReqC**

****

* En plaçant les table côte à côte et qu’on sélectionne tous les champs de chaque table, ça donne un produit cartésien
* En SQL

SELECT Animaux.\*, Clients.\*  
FROM Animaux, Clients;

* Pour ReqC, on peut formuler :
* Pour tous les enregistrements de la table Animaux faire  
   Pour tous les enregistrements de la table Clients faire  
   Afficher la valeur de tous les champs de Animaux et Clients  
   finPour  
  finPour
* Cette requête représente un produit cartésien :  
  Animaux (131) X Clients (49) = 6419 enregistrements résultants

Remarques :

* Access qualifie toujours entièrement les identificateurs de champs : **[nom de table].[nom de champ]**
* Les crochets [ ] servent come délimiteur lorsqu’un caractère espace est présent dans un identificateur.

### Syntaxe avec prédicats

SELECT < prédicat > < liste de champs >  
FROM < liste de tables >;

* Où < prédicat > peut-être :
* ALL (par défaut)
* DISTINCT
* DISTINCTROW
* TOP n [PERCENT]
* Un prédicat spécifie les enregistrements sélectionnés par la requête SQL.

#### ALL

* Exemple ALL :

SELECT \*  
FROM Animaux;

Est donc équivalent à :

SELECT ALL \*  
FROM Animaux;

#### DISTINCT

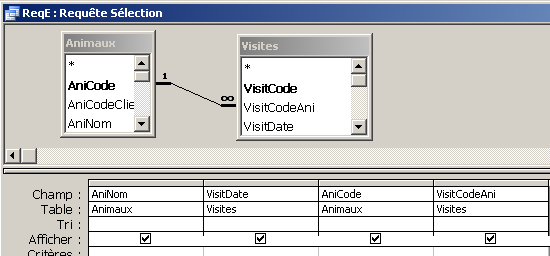
* **ReqD** - Exemple de projection :

SELECT DISTINCT ActNom  
FROM Acteurs;

* Résultats :

|  |
| --- |
| ActNom |
| Birkins |
| Fonda |
| Martins |
| Pitt |

* DISTINCT élimine les doublons, donc implémente une projection au sens formel de notre définition. Voir apprentissage 3
* La propriété de la requête *Valeurs distinctes = oui* (UniqueValues) provoque l’utilisation de DISTINCT en SQL.
* **ReqE**



* On veut la liste des visites faites à la clinique. Le nom des animaux ainsi que les dates de leurs visites seront suffisantes. On a ajouté *Anicode* et *VisiteCodeAni* seulement pour bien montrer l’équijointure dans les résultats.

SELECT Animaux.AniNom, Visites.VisitDate  
FROM Animaux INNER JOIN Visites   
 ON Animaux.AniCode = Visites.VisitCodeAni;

* Nombre de résultats : 86

| **ReqE** | | | |
| --- | --- | --- | --- |
| **AniNom** | **VisitDate** | **AniCode** | **VisitCodeAni** |
| Bobo | 21-02-1996 | AC001-01 | AC001-01 |
| Bobo | 01-04-1996 | AC001-01 | AC001-01 |
| Bobo | 01-11-1996 | AC001-01 | AC001-01 |
| Presto Chango | 03-09-1996 | AC001-02 | AC001-02 |
| Presto Chango | 11-10-1996 | AC001-02 | AC001-02 |
| Presto Chango | 26-11-1996 | AC001-02 | AC001-02 |
| …. |  |  |  |

* Interprétation complète : Obtenir la liste de toutes les visites faites à la clinique en spécifiant les dates et les des animaux impliqués.

Remarques :

* Le générateur d’Access qualifie toujours les noms de chaque champs *< nom de table >.< Nom de champ >*, mais c’est obligatoire uniquement si des noms identiques de tables différentes ou de champs différents sont présents dans la requête.
* Access utilise fréquemment [ ] pour tout identificateur, mais c’est obligatoire uniquement si des espaces ou tirets sont présents dans la dans l’identificateur.
* Dans le *ON* le nom des tables est obligatoire, mais pour Access seulement.
* Si la requête était :

SELECT Aninom  
FROM Animaux INNER JOIN Visites  
 ON Animaux.AniCode = Visites.VisitCodeAni;

* Nombre de résultats : 86
* **ReqF**

SELECT DISTINCT Aninom  
FROM Animaux INNER JOIN Visites  
 ON Animaux.Anicode = Visites.VisitCodeAni

* Nombre de résultats : 41
* Interprétation :

Obtenir la liste des noms différents d’animaux ayant fait au moins une visite à la clinique.

* Quelle est la différence entre les résultats de :

SELECT Aninom, ClientNom  
FROM Animaux, Clients  
WHERE ClientCode = AniCodeClient;

ET

SELECT Aninom, ClientNom  
FROM Clients INNER JOIN Animaux  
 ON Clients.ClientCode = Animaux.AniCodeClient;

AUCUNE

Norme 4

Toujours utiliser les mots réservés *INNER JOIN* et *ON* pour exprimer une équijointure. Standard ANSI SQL ’92.

* Ainsi *WHERE* servira uniquement à restreindre les enregistrements d’entrée d’une table (ou plusieurs), avant de procéder à la jointure s’il y a lieu, et se distinguera de la restriction de la jointure.

#### DICTINCTROW

* **ReqG**

SELECT DISTINCTROW Aninom  
FROM Animaux INNER JOIN Visites  
 ON Animaux.AniCode =Visites.VisitCodeAni;

* Nombre de résultats : 43
* Propriété : Enr. Unique (UniqueRecords)
* Interprétation :

Obtenir la liste des noms des animaux différents ayant fait au moins une visite à la clinique.

Note

La distinction entre *DISTINCT* et *DISTINCTROW* est parfois « subtile ». *DISTINCT* porte sur les résultats affichés, on y enlève les doublons à la toute fin. *DISTINCTROW* porte sur les enregistrements servant à la requête, s’assure que les résultats proviennent d’enregistrements différents.

* La différence serait plus évidente avec des requêtes comme :

SELECT DISTINCT Anitype  
FROM Animaux INNER JOIN Visites  
 ON AniCode =VisitCodeAni;

12 résultats

ET

SELECT DISTINCTROW Anitype  
FROM Animaux INNER JOIN Visites  
 ON AniCode =VisitCodeAni;

43 résultats

* La différence serait pas évidente avec des requêtes comme :

SELECT DISTINCT Anicode, Aninom  
FROM Animaux INNER JOIN Visites  
 ON AniCode =VisitCodeAni;

43 résultats

ET

SELECT DISTINCTROW Anicode, Aninom  
FROM Animaux INNER JOIN Visites  
 ON AniCode =VisitCodeAni;

43 résultats

Remarques :

* Pour effectuer un DISTINCT, Access doit trier les enregistrements résultants avant d’éliminer les doublons (pas pour DISTINCTROW), les résultats sont alors en ordre selon les champs exprimés.
* Ne pas se fier à *DISTINCT* dans le but de trier des enregistrements, il existe une clause *ORDER BY* pour cela.

#### TOP

* **ReqH**

SELECT TOP 10 Anitype  
FROM Animaux;

* Interprétation :

Quels sont les types des 10 premiers animaux?

* Résultats :

| **AniType** |
| --- |
| LAPIN |
| LEZARD |
| MOUFETTE |
| CHIEN |
| COCHON |
| CHEVAL |
| CHIEN |
| LOUP |
| DINOSAURE |
| CHIEN |

* Si on ajoute *PERCENT* le nombre devient un pourcentage des enregistrements (14 enregistrements) :

SELECT TOP 10 PERCENT Anitype  
FROM Animaux;

* Interprétation :

Quels sont les types des 1/10 premiers animaux.

* **Reqi**

SELECT DISTINCT TOP 10 Anitype  
FROM Animaux;

* Interprétation :

Par ordre alphabétique, quels sont les 10 premiers types différents d’animaux de la table animaux?

* Résultats :

| **AniType** |
| --- |
| AGNEAU |
| AUTRUCHE |
| BALEINE |
| CANARD |
| CHAT |
| CHEVAL |
| CHEVRE |
| CHIEN |
| COCHON |
| COCHON D'INDE |

### Clause WHERE (Restriction)

* **ReqJ**

SELECT Aninom  
FROM Animaux  
WHERE Anitype = "chien";

* Interprétation :

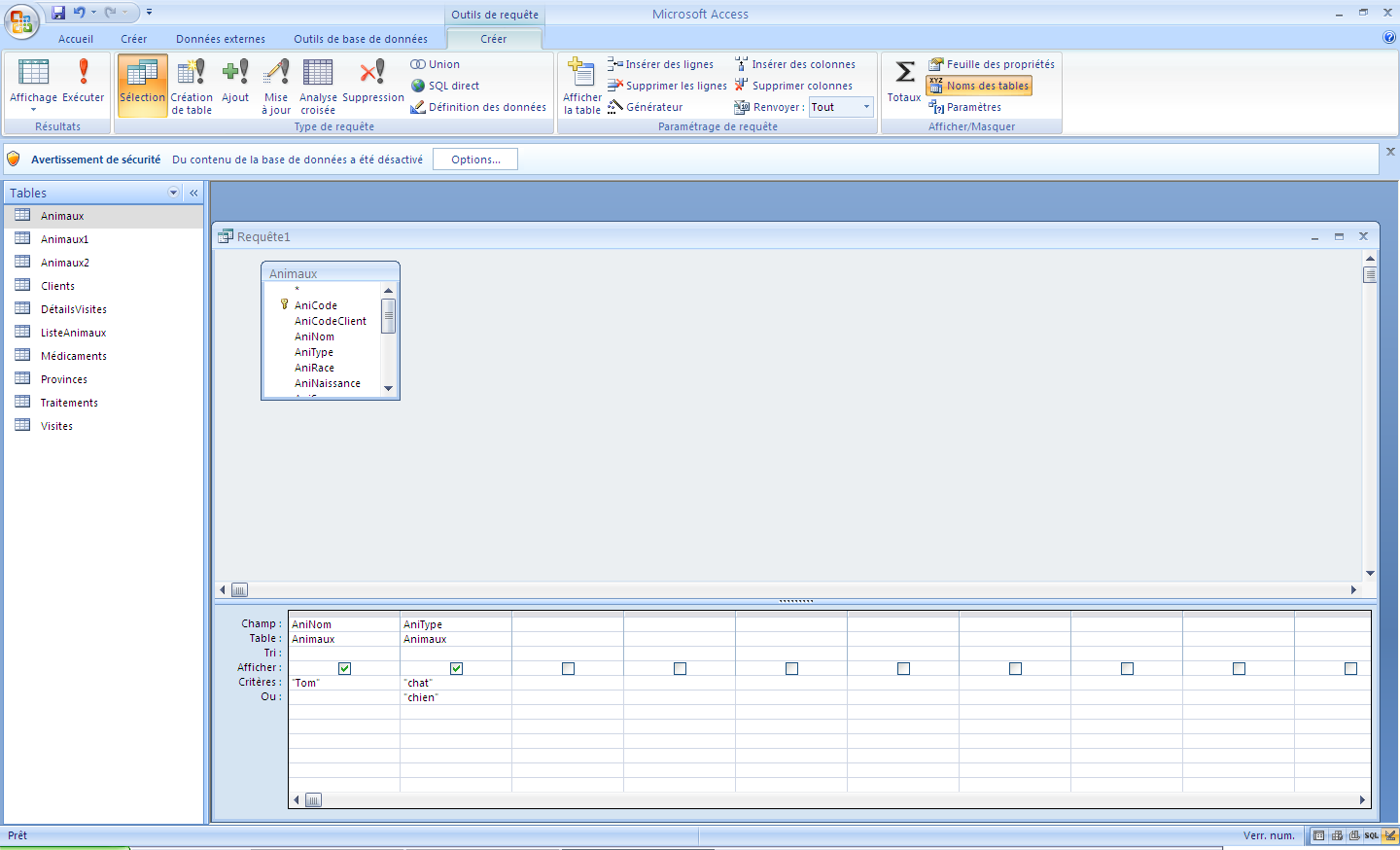
Quels sont les noms des animaux connus par la clinique la clinique qui sont des « chien »?

* Nombre de résultats : 27
* **ReqK (restriction complexe)**

SELECT AniNom, AniType  
FROM Animaux  
WHERE (AniNom = "Tom" AND AniType= "chat")  
 OR (Anitype = "chien");

* Interprétation :

On veut les noms de tous les chiens et les noms des chats qui s’appellent « Tom »

* En Access :

Des OU logiques

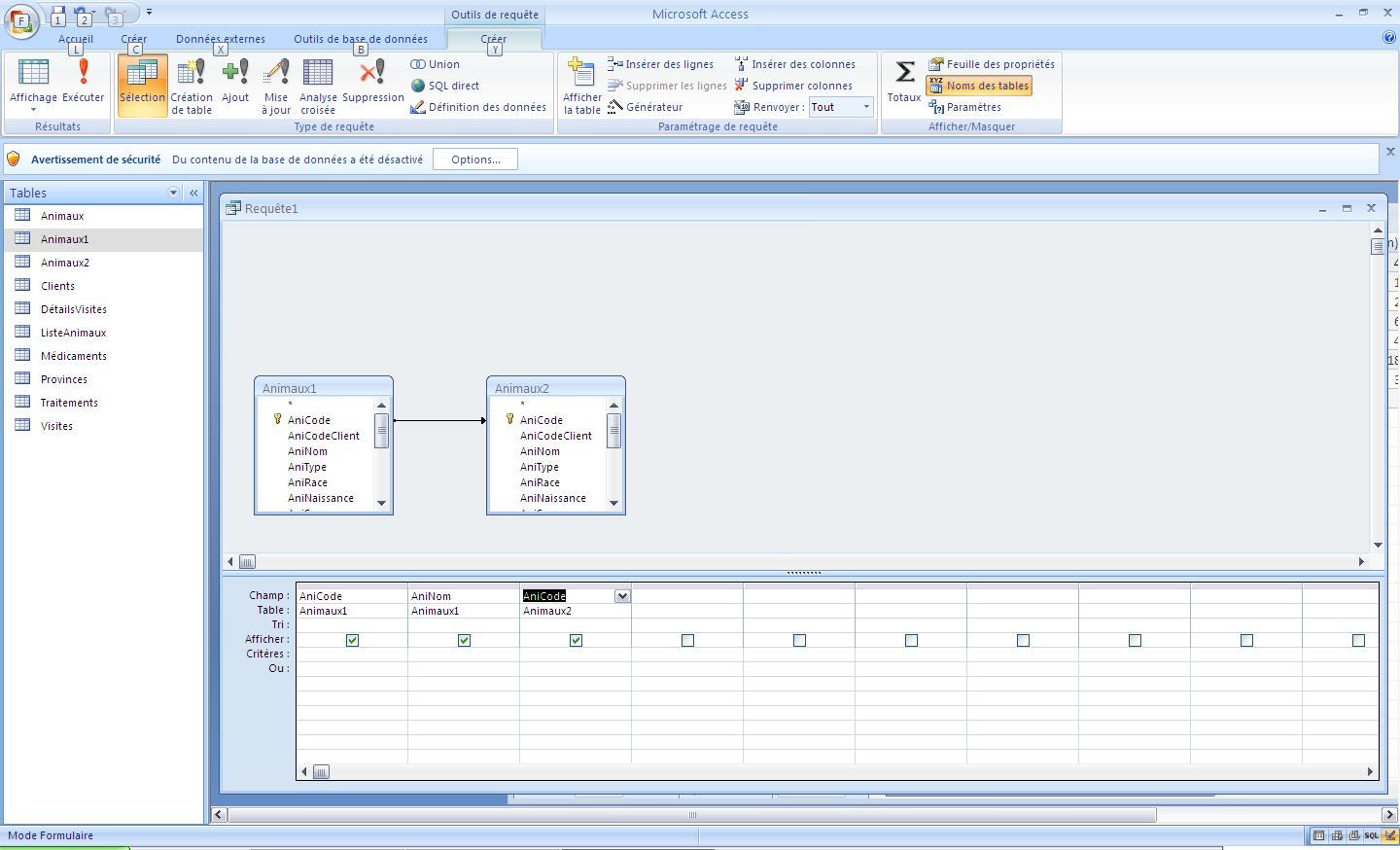
Des ET logiques

Remarques :

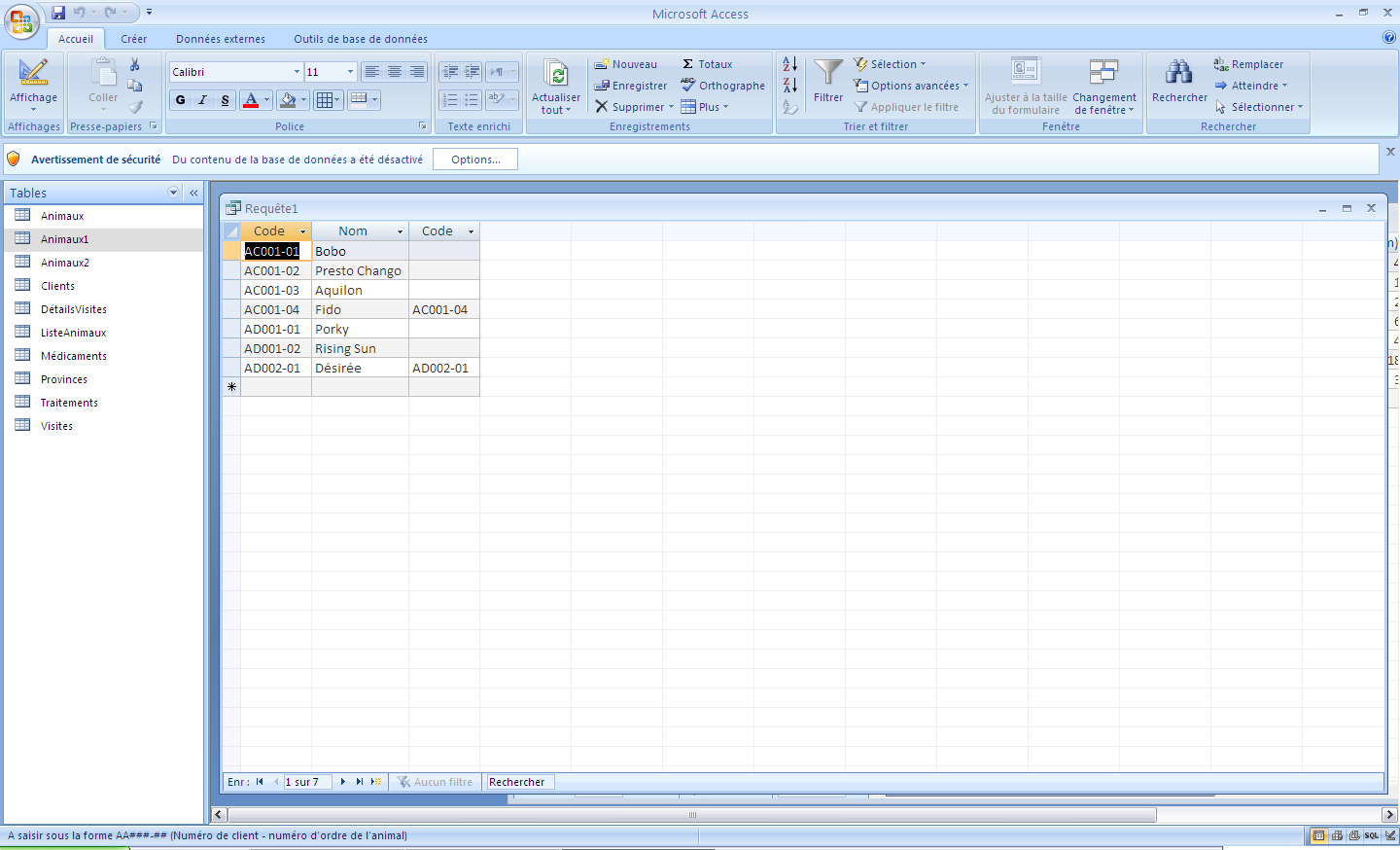
* On peut éditer ces conditions (restrictions) avec le générateur d’expression.
* Le générateur écrit les expressions en français : (*Aninom = "Tom » ET AniType = "chien") OU (Anitype = "chien")*
* Le langage SQL est strictement anglais!
* Attention aux dates, si dans la grille j’écris *#20-08-1995#* il est converti en *WHERE AniNaissance = #08-20-1995#* (format anglais)

### Implémentation de la différence algébrique

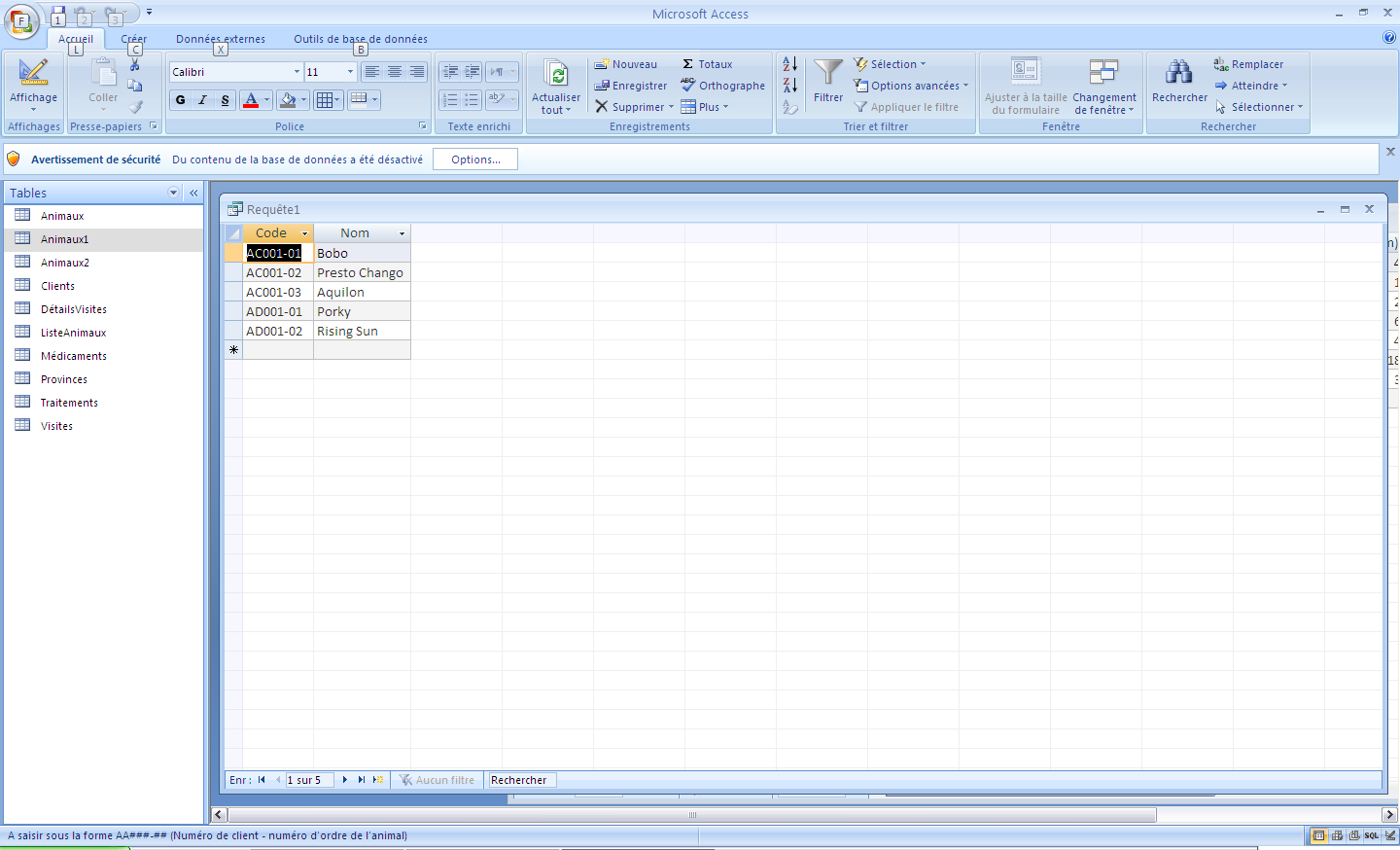
* Essayons un *LEFT JOIN*



SELECT Animaux1.AniCode, Animaux1.AniNom, Animaux2.AniCode  
FROM Animaux1 LEFT JOIN Animaux2  
 ON Animaux1.AniCode = Animaux2.AniCode;

* Les résultats ne sont pas ceux d’une différence:
* Les seuls résultats qu’on ne voudrait pas sont ceux où le deuxième Anicode à une valeur non-nulle. On va donc la modifier comme ceci :

SELECT Animaux1.AniCode, Animaux1.AniNom   
FROM Animaux1 LEFT JOIN Animaux2  
 ON Animaux1.AniCode = Animaux2.AniCode  
WHERE Animaux2.Anicode is null;



* Résultats :

Remarques :

* C’est donc une différence algébrique
* Access nous donne un assistant pour créer une telle requête : *Requête de non-correspondance.*
* Utilisations de *NULL*: *< champ > is null  
   < champ > is not null  
   not ( < champ > is null )*

ou

### Expressions

#### Syntaxe

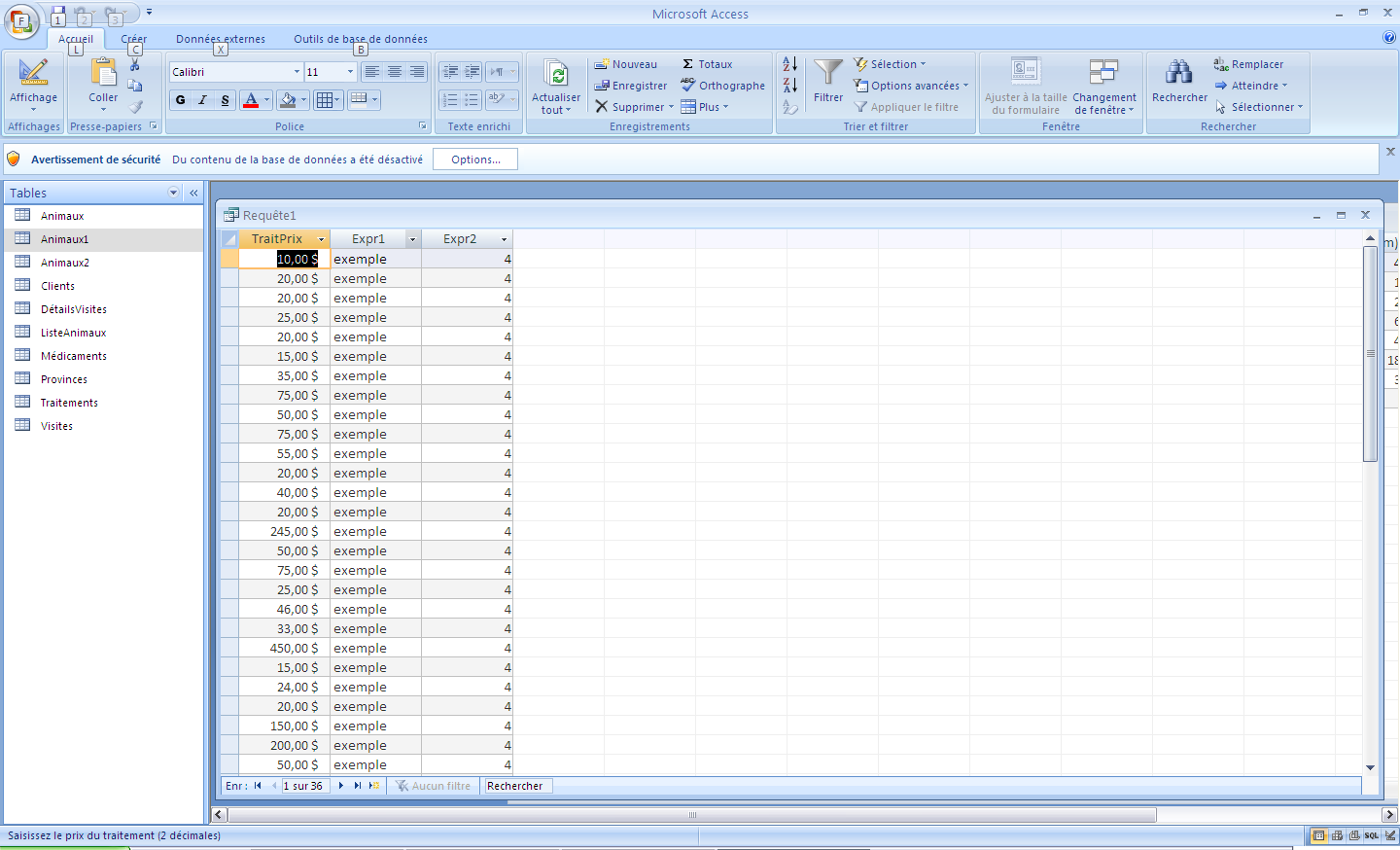
SELECT < liste\_d’expressions >  
FROM < liste\_table\_req >;

* Où *< liste\_d’expressions >* :
* Champ de table
* Constante numériques, date
* Expression impliquant des champs des enregistrements d’entrée
* Expression AS < alias >
* **ReqL** édité directement en SQL

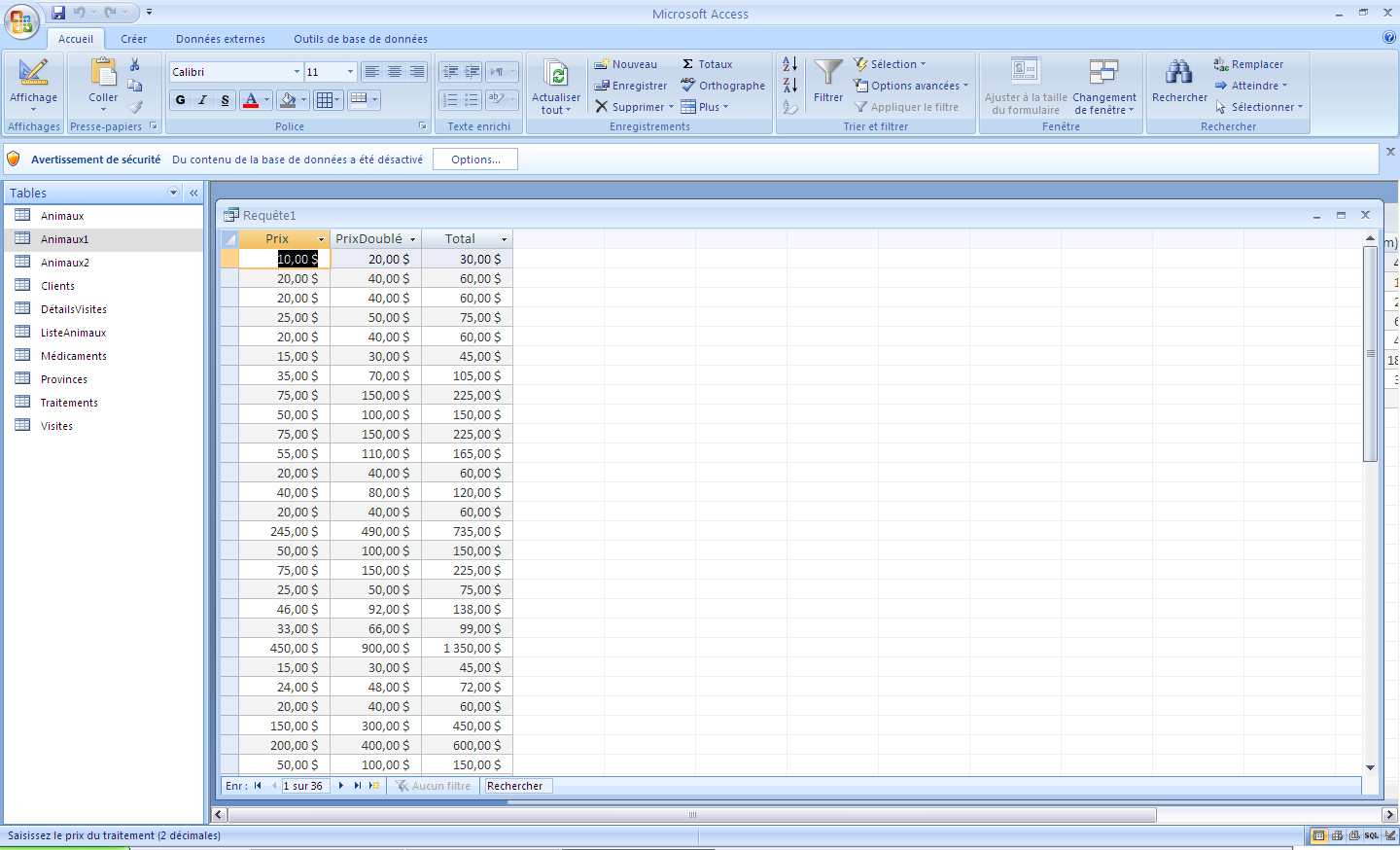
SELECT TraitPrix, "exemple", 4  
FROM Traitements;

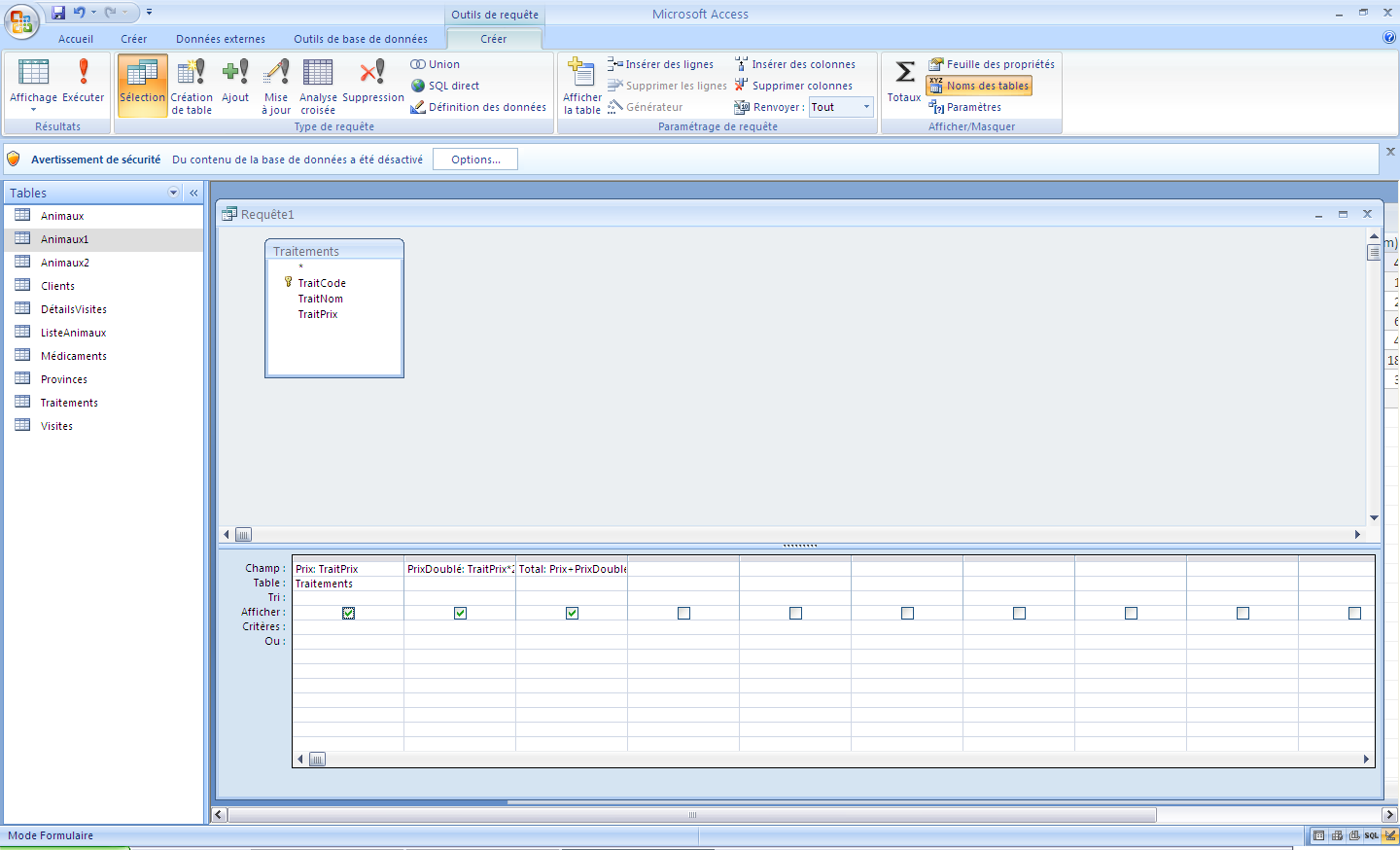
* Après une exécution Access réécrit :

SELECT Traitements.TraitPrix, "exemple" AS Expr1, 4 AS Expr2  
FROM Traitements;

* Résultats :
* On peut donc faire :

SELECT TraitPrix AS Prix, TraitPrix \* 2 AS PrixDoublé,  
 Prix + PrixDoublé AS Total  
FROM Traitements;

* Résultats :
* Donc < expression > AS < Alias >
* *Alias* devient l’en-tête de colonne de sortie
* *Alias* peut être réutilisé dans la liste de sélection en autant qu’il n’apporte pas d’ambiguïté.
* On peut écrire < Alias>
* En mode SQL
* Dans la grille de saisie en écrivant :



* Dans la grille avec le menu contextuel sur la colonne > Propriétés, dan légende.

À faire :

* Quel est l’ordre de priorité d’Access pour affichage en mode feuille de réponses d’une requête pour le nom de l’entête donné à une colonne?

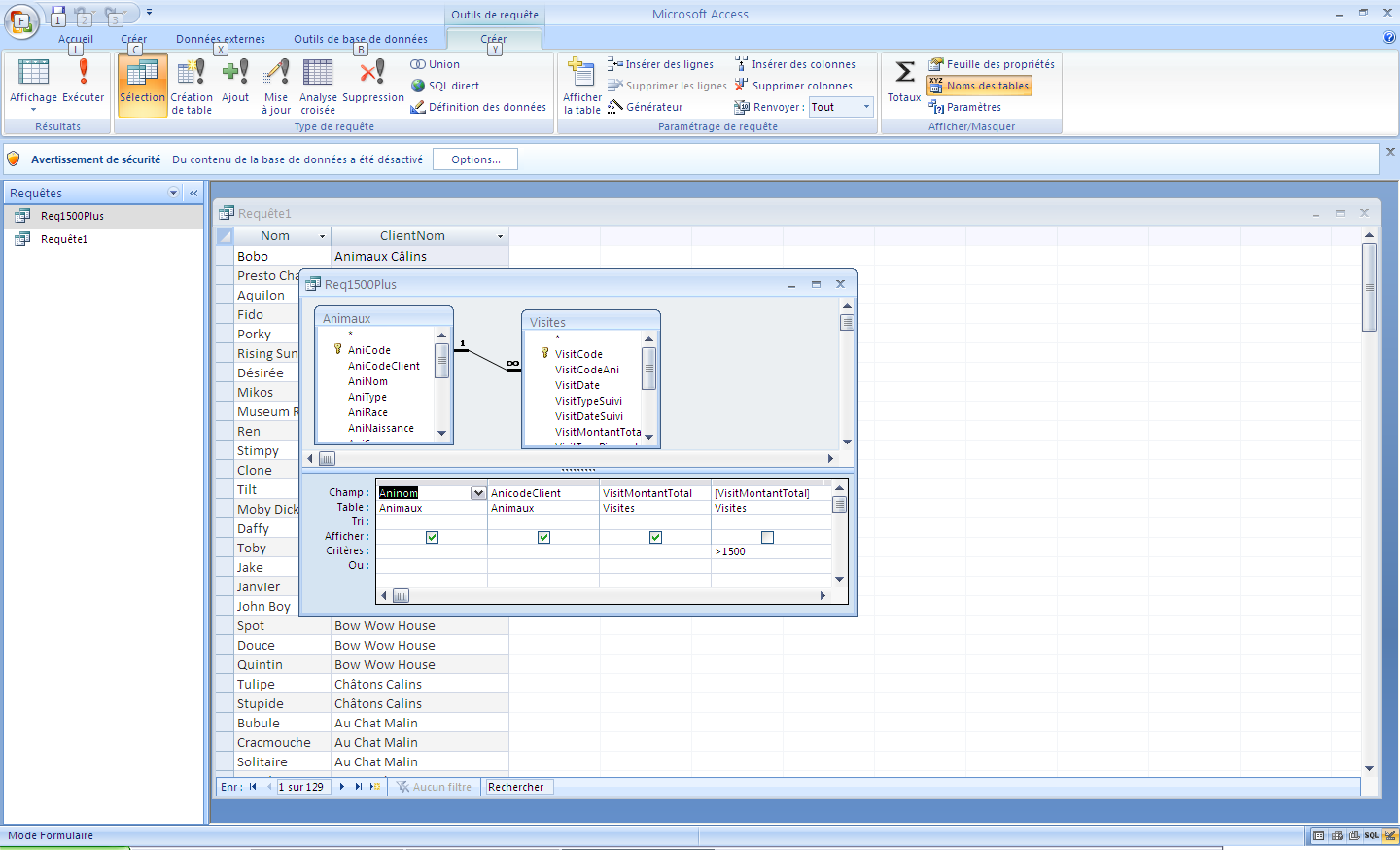
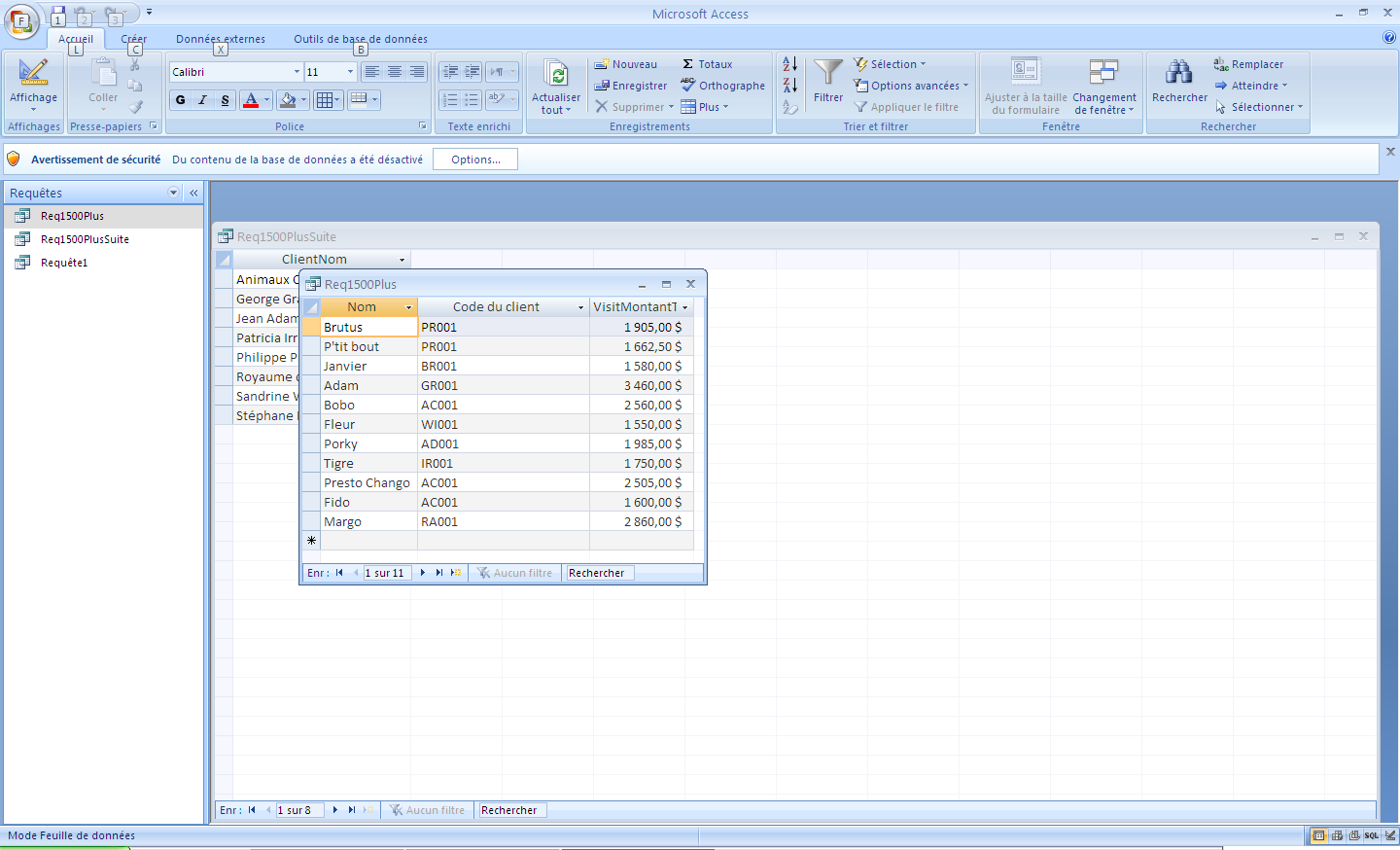
|  |  |
| --- | --- |
| Nom de champ de table | 4 |
| Propriété légende du champ en mode création de table | 2 |
| *Alias* donné dans une instruction SQL | 3 |
| Propriété légende de la colonne de requête | 1 |

#### < liste\_table\_ou\_requête >

* **Req1500Plus**
* Afficher la liste des visites dont le montant total est supérieur à 1500$ avec le nom de l’animal impliqué et le code de son propriétaire.

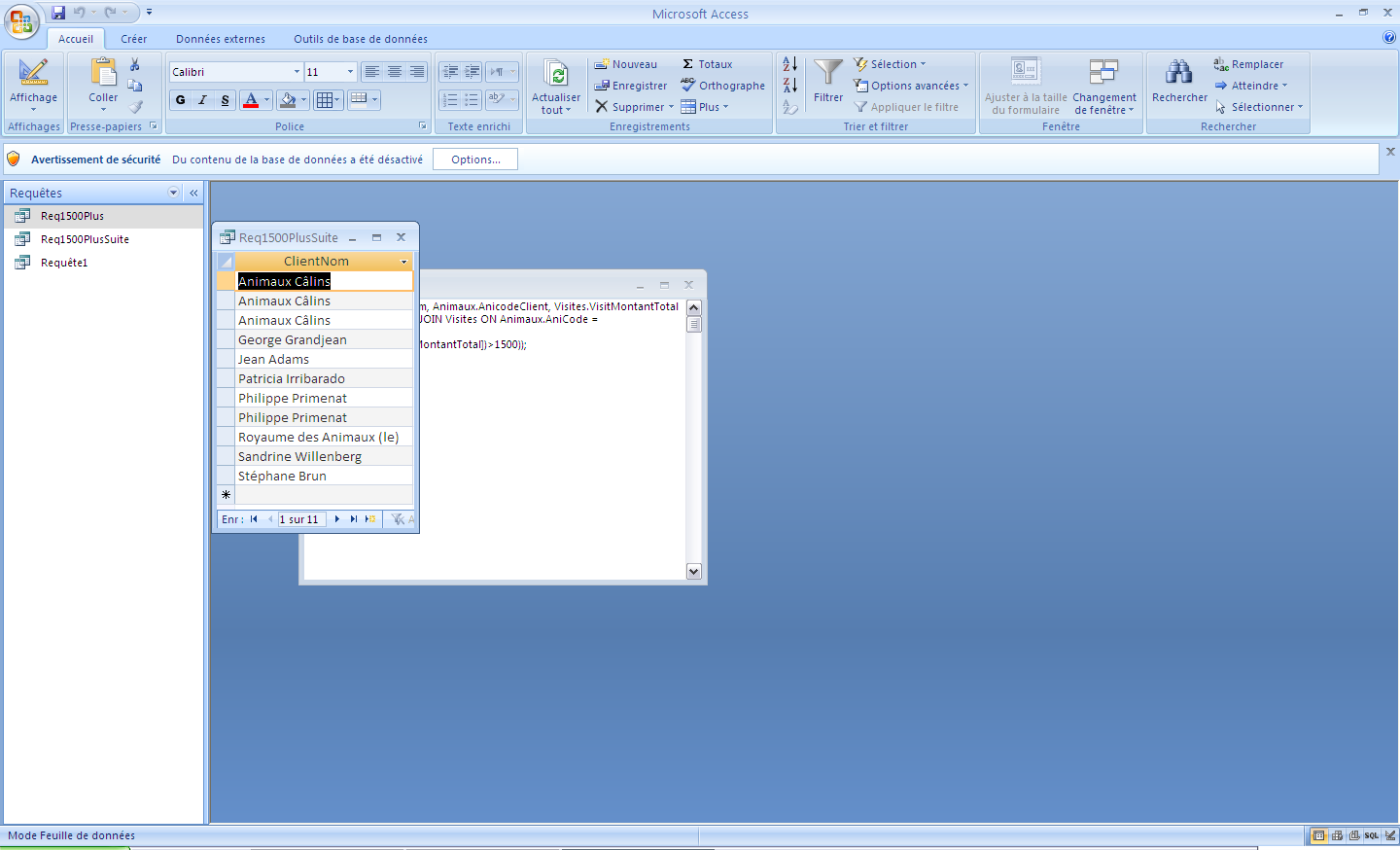
SELECT Aninom, AnicodeClient, VisitMontantTotal  
FROM Animaux INNER JOIN Visites  
 ON AniCode = VisitCodeAni  
WHERE VisitMontantTotal > 1500;

Rappel : Access ne le supporte pas. Il exige de le précéder du nom de la table.



* **Req1500PlusSuite**
* On veut connaître les noms des clients, en ordre, qui ont déboursé plus que 1500$ lors de visites à la clinique.

SELECT ClientNom  
FROM Clients INNER JOIN Req1500Plus  
 ON Clients.ClientCode = Req1500Plus.AniCodeClient  
ORDER BY ClientNom;



* 11 Résultats :
* Oups des doublons :

SELECT DISTINCT ClientNom  
FROM Clients INNER JOIN Req1500Plus  
 ON Clients.ClientCode = Req1500Plus.AniCodeClient  
ORDER BY ClientNom;

Table Autos

| **Autos** | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Code** | **Marque** | **Modèle** | **Année** | **Couleur** | **Prix** | **Cylindres** | **Puissance (CV)** |
| 1 | Toyota | Corolla | 2001 | bleu | 19 500,00 $ | 4 | 138 |
| 2 | Ford | Grand Marquis | 2000 | verte | 26 500,00 $ | 6 | 175 |
| 3 | Nissan | Sentra | 1999 | bleu | 13 565,00 $ | 4 | 103 |
| 4 | Chrysler | Intrepid | 2000 | blanc | 25 000,00 $ | 6 | 160 |
| 5 | Chrysler | Voyager | 2001 | blanc | 32 000,00 $ | 6 | 165 |
| 6 | Suzuki | Kia | 2000 | bleu | 23 500,00 $ | 4 | 115 |
| 7 | Chevrolet | Cavalier | 2000 | rouge | 18 795,00 $ | 4 | 115 |
| 8 | Mazda | 626 | 2001 | blanc | 26 795,00 $ | 6 | 135 |
| 9 | Daewoo | Lemans | 2001 | bleu | 11 995,00 $ | 4 | 90 |
| 10 | Pontiac | Lemans | 2000 | rouge | 28 795,00 $ | 6 | 135 |
| 11 | Toyota | Camry | 1999 | blanc | 28 995,00 $ | 6 | 175 |
| 12 | Ford | Mercury | 2001 | rouge | 33 995,00 $ | 6 | 175 |
| 13 | Ford | Grand Marquis | 2001 | rouge | 29 995,00 $ | 6 | 165 |
| 14 | Nissan | Pathfinder | 2000 | rouge | 38 995,00 $ | 6 | 175 |
| 15 | Nissan | Sentra | 1999 | bleu | 14 000,00 $ | 4 | 115 |
| 16 | Toyota | Echo | 2001 | blanc | 15 995,00 $ | 4 | 105 |
| 17 | Toyota | Camry | 2000 | bleu | 29 000,00 $ | 6 | 165 |
| 18 | Chevrolet | Sunfire | 2001 | rouge | 19 500,00 $ | 6 | 115 |
| 19 | Chevrolet | Impala | 1964 | blanc | 8 000,00 $ | 8 | 145 |
| 20 | Mazda | Protégé | 1999 | blanc | 15 500,00 $ | 4 | 115 |
| 21 | Mazda | Protégé | 2000 | bleu | 16 500,00 $ | 4 | 125 |
| 22 | Mazda | Protégé | 2001 | rouge | 17 450,00 $ | 4 | 135 |
| 23 | Chrysler | Jeep | 2000 | vert | 25 495,00 $ | 6 | 135 |
| 24 | Chrysler | New Yorker | 1985 | bleu | 18 995,00 $ | 8 | 165 |
| 25 | Nissan | Pathfinder | 2001 | blanc | 37 995,00 $ | 6 | 175 |
| 26 | Mazda |  | 2000 | rouge | 19 000,00 $ | 4 | 135 |
| 27 | Mazda | Protégé |  | blanc | 17 000,00 $ | 4 | 115 |
| 28 | Mazda | 626 | 2001 |  | 26 500,00 $ | 6 | 165 |
| 29 | Toyota | Camry | 2000 |  | 0,00 $ | 0 | 0 |
| 30 | Pontiac | Sunfire | 2001 | Rouge | 20 000,00 $ | 6 |  |
| 31 | Pontiac | Sunfire | 2000 | Rouge | 19 995,00 $ | 6 |  |
| 32 | Nissan |  |  |  |  |  |  |
| 33 | Chevrolet | Cavalier | 2000 | Vert | 17 500,00 $ | 4 | 115 |
| 34 | Mazda | 626 | 2001 |  | 25 600,00 $ | 6 | 165 |
| 37 | Nissan | Pathfinder | 2000 | Bleu | 20 000,00 $ | 9 | 165 |

### ORDER BY

Par défaut

SELECT < liste\_expressions >  
FROM < liste\_tab\_req >  
ORDER BY < champ1 [ASC|DESC] >  
 [< champ1 [ASC|DESC] >];

* **ReqAuto1**
* Donnez la marque et le modèle de toutes les automobiles en stock, les résultats devront être classés en ordre de marque.

SELECT AutoMarque, AutoModèle  
FROM Autos  
ORDER BY AutoMarque;

* 35 résultats :

| **Marque** | **Modèle** |
| --- | --- |
| Chevrolet | Impala |
| Chevrolet | Cavalier |
| Chevrolet | Sunfire |
| Chevrolet | Cavalier |
| Chrysler | Voyager |
| Chrysler | New Yorker |
| Chrysler | Jeep |
| Chrysler | Intrepid |
| Daewoo | Lemans |
| Ford | Grand Marquis |
| Ford | Mercury |
| Ford | Grand Marquis |
| Mazda | 626 |
| Mazda | 626 |
| … | … |

Note :

Dans ce cours, il n’y aura pas d’autre clause à la suite de *ORDER BY* dans une instruction *SELECT* en SQL (nous n’utiliserons pas [*WITH OWNERACCESS OPTION* durant la session])

### GROUP BY

SELECT < liste\_expressions >  
FROM < liste\_tab\_req >  
GROUP BY < liste\_de\_champs >  
[HAVING < condition >]

* Permet de regrouper des enregistrements de sortie (résultats) selon les valeurs identiques des champs spécifiés.
* **ReqAuto2**

SELECT AutoMarque  
FROM Autos  
GROUP BY AutoMarque;

* 9 résultats :

| **Marque** |
| --- |
| Chevrolet |
| Chrysler |
| Daewoo |
| Ford |
| Mazda |
| Nissan |
| Pontiac |
| Suzuki |
| Toyota |

* Dans Access, le menu contextuel de la grille avec la commande *Σ Totaux* fait apparaître une ligne opération. Par la suite, cette ligne nous donne accès à d’autres opérations.
* Dans le cas de cette requête les résultats sont identiques à :

SELECT DISTINCT AutoMarque  
FROM Autos;

Note

*DISTINCT* et *GROUP BY* ne sont pas incompatibles, mais *DISTINCT* devient inutile!

SELECT DISTINCT AutoMarque  
FROM Autos  
GROUP BY AutoMarque;

Correct syntaxiquement, mais je pénaliserai!

* **ReqAuto3**

SELECT AutoMarque, AutoModèle  
FROM Autos  
GROUP BY AutoMarque, AutoModèle;

* 22 résultats :

| **Marque** | **Modèle** |
| --- | --- |
| Chevrolet | Cavalier |
| Chevrolet | Impala |
| Chevrolet | Sunfire |
| Chrysler | Intrepid |
| Chrysler | Jeep |
| Chrysler | New Yorker |
| Chrysler | Voyager |
| Daewoo | Lemans |
| Ford | Grand Marquis |
| Ford | Mercury |
| Mazda |  |
| Mazda | 626 |
| Mazda | Protégé |
| Nissan |  |
| Nissan | Pathfinder |
| Nissan | Sentra |
| Pontiac | Lemans |
| Pontiac | Sunfire |
| Suzuki | Kia |
| Toyota | Camry |
| Toyota | Corolla |
| Toyota | Echo |

* Access a dû trier sur les 2 champs avant d’enlever les doublons. Si des valeurs sont absentes (null), elles précèdent les valeurs non-nulles.

#### Contraintes et règles

* Tout Champ d’affichage (liste dans le *SELECT*) **DOIT** se retrouver dans la clause *GROUP BY.*
* Le *GROUP BY* peut utiliser un champ sans que celui-ci ne soit affiché par la requête.
* Le *GROUP BY* regroupe aussi les valeurs nulles.
* **ReqAuto4**

SELECT AutoMarque, AutoModèle  
FROM Autos  
GROUP BY AutoMarque, AutoModèle, AutoAnnée;

* 29 résultats :

| **Marque** | **Modèle** |  | **Marque** | **Modèle** |
| --- | --- | --- | --- | --- |
| Chevrolet | Cavalier |  | Nissan |  |
| Chevrolet | Impala |  | Nissan | Pathfinder |
| Chevrolet | Sunfire |  | Nissan | Pathfinder |
| Chrysler | Intrepid |  | Nissan | Sentra |
| Chrysler | Jeep |  | Pontiac | Lemans |
| Chrysler | New Yorker |  | Pontiac | Sunfire |
| Chrysler | Voyager |  | Pontiac | Sunfire |
| Daewoo | Lemans |  | Suzuki | Kia |
| Ford | Grand Marquis |  | Toyota | Camry |
| Ford | Grand Marquis |  | Toyota | Camry |
| Ford | Mercury |  | Toyota | Corolla |
| Mazda |  |  | Toyota | Echo |
| Mazda | 626 |  |  |  |
| Mazda | Protégé |  |  |  |
| Mazda | Protégé |  |  |  |
| Mazda | Protégé |  |  |  |
| Mazda | Protégé |  |  |  |

* Pour mieux comprendre, si on ajoutait dans la liste du *SELECT* AutoAnnée, le résultat serait le suivant :

| **Marque** | **Modèle** | **Année** |
| --- | --- | --- |
| … |  |  |
| Daewoo | Lemans | 2001 |
| Ford | Grand Marquis | 2000 |
| Ford | Grand Marquis | 2001 |
| Ford | Mercury | 2001 |
| Mazda |  | 2000 |
| Mazda | 626 | 2001 |
| Mazda | Protégé |  |
| Mazda | Protégé | 1999 |
| Mazda | Protégé | 2000 |
| Mazda | Protégé | 2001 |
| … |  |  |

* Si on ajoute *DISTINCT* est-ce que ça changerait les résultats.

Oui! Mais si on ajoute l’affichage de AutoAnnée, non.

#### Explications

* *GROUP BY* sélectionne les valeurs des champs identifiés et/ou regroupés.
* Met implicitement en ordre sur le 1er champ de regroupement, ensuite le 2e,…
* Élimine les doublons (en mémoire).
* Par exemple :

| **Marque** | **Modèle** | **Année** | **Donne** | **Marque** | **Modèle** |
| --- | --- | --- | --- | --- | --- |
| Mazda | 626 | 2001 |  | Mazda | 626 |
| Mazda | 626 | 2001 |  |  |  |
| Mazda | 626 | 2001 |  |  |  |

Mais

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Mazda | Protégé |  | **Donne** | Mazda | Protégé |
| Mazda | Protégé | 1999 |  | Mazda | Protégé |
| Mazda | Protégé | 2000 |  | Mazda | Protégé |
| Mazda | Protégé | 2001 |  | Mazda | Protégé |

### Fonctions d’agrégations

* Elles se présentent dans < liste\_de\_selection >
* Impliquent un regroupement, donc sont souvent associées à une clause *GROUP BY*

#### Fonction Count

* La fonction COMPTE calcule le nombre d’enregistrements répondant à la requête.

SELECT COUNT(< expression >) AS < nom\_entête >  
FROM < liste\_tab\_req >;

* **ReqAuto5**

SELECT COUNT(AutoMarque) AS Nb  
FROM Autos;

* 1 résultat :

| **Nb** |
| --- |
| 35 |

* **ReqAuto6**

SELECT COUNT(AutoAnnée) AS NbAnnées  
FROM Autos;

* 1 résultat :

| **NbAnnées** |
| --- |
| 33 |

Remarques

* Dans *COUNT(exp)*, si *exp* est un champ, les valeurs nulles seront ignorées
* Si on écrit *COUNT(\*)* les valeurs nulles seront comptées.
* **ReqAuto7**

SELECT AutoMarque AS Marque, AutoModèle AS Modèle,  
 COUNT (AutoAnnée) AS Compte  
FROM Autos  
GROUP BY AutoMarque, AutoModèle;

* Résultats :

| **Marque** | **Modèle** | **Compte** |
| --- | --- | --- |
| Chevrolet | Cavalier | 2 |
| Chevrolet | Impala | 1 |
| Chevrolet | Sunfire | 1 |
| Chrysler | Intrepid | 1 |
| Chrysler | Jeep | 1  Ici, ça veut dire aucune valeur dans le champ AutoAnnée |
| Chrysler | New Yorker | 1 |
| Chrysler | Voyager | 1 |
| Daewoo | Lemans | 1 |
| Ford | Grand Marquis | 2 |
| Ford | Mercury | 1 |
| Mazda |  | 1 |
| Mazda | 626 | 3 |
| Mazda | Protégé | 3 |
| Nissan |  | 0 |
| Nissan | Pathfinder | 3 |
| Nissan | Sentra | 2 |
| Pontiac | Lemans | 1 |
| Pontiac | Sunfire | 2 |
| Suzuki | Kia | 1 |
| Toyota | Camry | 3 |
| Toyota | Corolla | 1 |
| Toyota | Echo | 1 |

* Si la requête était :

SELECT AutoMarque AS Marque, AutoModèle AS Modèle,  
 COUNT (\*) AS Compte  
FROM Autos  
GROUP BY AutoMarque, AutoModèle;

* Les résultats :

| **Marque** | **Modèle** | **Compte** |  | **Marque** | **Modèle** | **Compte** |
| --- | --- | --- | --- | --- | --- | --- |
| Chevrolet | Cavalier | 2 |  | Mazda | Protégé | **4** |
| Chevrolet | Impala | 1 |  | Nissan |  | **1** |
| Chevrolet | Sunfire | 1 |  | Nissan | Pathfinder | 3 |
| Chrysler | Intrepid | 1 |  | Nissan | Sentra | 2 |
| Chrysler | Jeep | 1 |  | Pontiac | Lemans | 1 |
| Chrysler | New Yorker | 1 |  | Pontiac | Sunfire | 2 |
| Chrysler | Voyager | 1 |  | Suzuki | Kia | 1 |
| Daewoo | Lemans | 1 |  | Toyota | Camry | 3 |
| Ford | Grand Marquis | 2 |  | Toyota | Corolla | 1 |
| Ford | Mercury | 1 |  | Toyota | Echo | 1 |
| Mazda |  | 1 |  |  |  |  |
| Mazda | 626 | 3 |  |  |  |  |

* **ReqAuto8**

SELECT AutoMarque, AutoModèle,  
 COUNT(AutoAnnée & AutoCouleur) AS Compte  
FROM Autos  
GROUP BY AUTOMARQUE, AutoModèle;

* Résultats :

| **Marque** | **Modèle** | **Compte** |  | **Marque** | **Modèle** | **Compte** |
| --- | --- | --- | --- | --- | --- | --- |
| Chevrolet | Cavalier | 2 |  | Mazda | Protégé | **4** |
| Chevrolet | Impala | 1 |  | Nissan |  | **0** |
| Chevrolet | Sunfire | 1 |  | Nissan | Pathfinder | 3 |
| Chrysler | Intrepid | 1 |  | Nissan | Sentra | 2 |
| Chrysler | Jeep | 1 |  | Pontiac | Lemans | 1 |
| Chrysler | New Yorker | 1 |  | Pontiac | Sunfire | 2 |
| Chrysler | Voyager | 1 |  | Suzuki | Kia | 1 |
| Daewoo | Lemans | 1 |  | Toyota | Camry | 3 |
| Ford | Grand Marquis | 2 |  | Toyota | Corolla | 1 |
| Ford | Mercury | 1 |  | Toyota | Echo | 1 |
| Mazda |  | 1 |  |  |  |  |
| Mazda | 626 | 3 |  |  |  |  |

* Pourquoi?

Parce que Count Protégé (20-1999blanc, 21-2000bleu, 22-2001rouge, 27-blanc) donne 4 et Nissan avec pas de marque n’a pas de couleur non plus, donc =0.

Remarque

Quand une fonction d’agrégat est présente dans la liste de *SELECT*, la présence de tout champ dans cette liste **oblige** l’utilisation du *GROUP BY* présentant ce ou ces champs.

conseil

1. Faire le *SELECT* sans *GROUP BY* avec tous les champs et expressions impliquées, en appliquant des *ORDER BY* (pour voir ce qui sera regroupé)*.*
2. Appliquer (ajouter) le *GROUP BY* et les fonctions d’agrégat.

#### Fonctions MAX et MIN

* **ReqAuto9**

SELECT AutoModèle, MAX(AutoPrix) AS Gros, MIN(AutoPrix) AS Petit  
FROM Autos  
GROUP BY AutoModèle;

* Résultats :

| **Modèle** | **Gros** | **Petit** |
| --- | --- | --- |
|  | 19 000,00 $ | 19 000,00 $ |
| 626 | 26 795,00 $ | 25 600,00 $ |
| Camry | 29 000,00 $ | 0,00 $ |
| Cavalier | 18 795,00 $ | 17 500,00 $ |
| Corolla | 19 500,00 $ | 19 500,00 $ |
| Echo | 15 995,00 $ | 15 995,00 $ |
| Grand Marquis | 29 995,00 $ | 26 500,00 $ |
| Impala | 8 000,00 $ | 8 000,00 $ |
| Intrepid | 25 000,00 $ | 25 000,00 $ |
| Jeep | 25 495,00 $ | 25 495,00 $ |
| Kia | 23 500,00 $ | 23 500,00 $ |
| Lemans | 28 795,00 $ | 11 995,00 $ |
| Mercury | 33 995,00 $ | 33 995,00 $ |
| New Yorker | 18 995,00 $ | 18 995,00 $ |
| Pathfinder | 38 995,00 $ | 20 000,00 $ |
| Protégé | 17 450,00 $ | 15 500,00 $ |
| Sentra | 14 000,00 $ | 13 565,00 $ |
| Sunfire | 20 000,00 $ | 19 500,00 $ |
| Voyager | 32 000,00 $ | 32 000,00 $ |

* *MAX* : plus grande valeur dans ce champ parmi les enregistrements regroupés.
* *MIN* : plus petite valeur dans ce champ parmi les enregistrements regroupés.
* On peut aussi faire :

SELECT AutoMarque, MIN(AutoPrix/100) AS MIN,  
 MAX(AutoPrix \*100) AS Max  
FROM Autos  
GROUP BY AutoMarque;

Expression calculée

#### SUM et AVG

* La fonction SOMME totalise les valeurs numériques contenues dans un champ spécifié dans la requête.
* Les enregistrements ayant une valeur nulle dans ce champ sont ignorés.
* La fonction MOYENNE évalue la moyenne des valeurs numériques non nulles, pour un champ (ou expression spécifié).
* **ReqAuto10**

SELECT AutoMarque, AutoModèle,  
 COUNT(AutoPuissance) AS Nb,  
 SUM(AutoPuissance) AS Somme,  
 AVG(AutoPuissance) AS Moyenne  
FROM Autos  
GROUP BY AutoMarque, AutoModèle;

* Résultats :

| **Marque** | **Modèle** | **Nb** | **Somme** | **Moyenne** |
| --- | --- | --- | --- | --- |
| Chevrolet | Cavalier | 2 | 230 | 115 |
| Chevrolet | Impala | 1 | 145 | 145 |
| Chevrolet | Sunfire | 1 | 115 | 115 |
| Chrysler | Intrepid | 1 | 160 | 160 |
| Chrysler | Jeep | 1 | 135 | 135 |
| Chrysler | New Yorker | 1 | 165 | 165 |
| Chrysler | Voyager | 1 | 165 | 165 |
| Daewoo | Lemans | 1 | 90 | 90 |
| Ford | Grand Marquis | 2 | 340 | 170 |
| Ford | Mercury | 1 | 175 | 175 |
| Mazda |  | 1 | 135 | 135 |
| Mazda | 626 | 3 | 465 | 155 |
| Mazda | Protégé | 4 | 490 | 122,5 |
| Nissan |  | 0 |  |  |
| … |  |  |  |  |

#### FIRST et LAST

* Renvoie la première (ou dernière) valeur trouvée dans un champ (ou expression) des enregistrements résultants de la requête.
* PREMIER et DERNIER tiennent compte des regroupements présents.
* **ReqAuto11**

SELECT FIRST(AutoPrix) AS PremierDeAutoPrix, LAST(AutoModèle) AS DernierDeAutoModèle,  
 MAX(AutoAnnée) AS MaxDeAutoAnnée, AutoMarque  
FROM Autos  
GROUP BY AutoMarque  
ORDER BY LAST(AutoModèle);

* Résultats :

| **PremierDeAutoPrix** | **DernierDeAutoModèle** | **MaxDeAutoAnnée** | **Marque** |
| --- | --- | --- | --- |
| 26 795,00 $ | 626 | 2001 | Mazda |
| 19 500,00 $ | Camry | 2001 | Toyota |
| 18 795,00 $ | Cavalier | 2001 | Chevrolet |
| 26 500,00 $ | Grand Marquis | 2001 | Ford |
| 23 500,00 $ | Kia | 2000 | Suzuki |
| 11 995,00 $ | Lemans | 2001 | Daewoo |
| 25 000,00 $ | New Yorker | 2001 | Chrysler |
| 13 565,00 $ | Pathfinder | 2001 | Nissan |
| 28 795,00 $ | Sunfire | 2001 | Pontiac |

Trié sur ce champ

Attention

Si FIRST ou LAST ne fait pas partie d’une instruction avec regroupement et/ou le champ impliqué n’est pas déjà trié, la valeur dans ce champ de sortie n’a pas nécessairement de signification utile.

### Expression

* Access-SQL nous offre beaucoup de fonctions intégrées, qui peuvent être utilisées dans des instructions SQL.
* Par exemple, pour les dates :
* Année(exp\_date) Year(exp\_date)
* Date() Date()
* EstDate(exp\_date) IsDate(exp\_date)
* Maintenant() Now()
* Mois(exp\_date) Mois(exp\_date)
* MonthName(month [; abbreviate]) où abbreviate est un booléen
* DateDiff(intervalle; date1; date2 [;firstweekday; firstweek])

Attention

* Les fonctions intégrées ne font pas partie des mots réservés de SQL
* Selon le SGBD utilisé, ces mêmes fonctions peuvent avoir des noms différents (ou ne pas exister)
* Leur emploi rend l’instruction SQL moins portable d’un SGBD à un autre…
* **ReqM**

SELECT ClientNom, ClientDepuis, Month(ClientDepuis) AS Mois,  
 MonthName(Month(ClientDepuis)) AS LeMois  
FROM Clients;

* Résultats :

| **ClientNom** | **ClientDepuis** | **Mois** | **LeMois** |
| --- | --- | --- | --- |
| Animaux Câlins | 1995-03-19 | 3 | mars |
| Jean Adams | 1993-01-01 | 1 | janvier |
| Bruce Adams | 1992-10-22 | 10 | octobre |
| Animaux exotiques S.A. | 1992-04-08 | 4 | avril |
| Animaux Familiers | 1994-07-05 | 7 | juillet |
| Aquarium de Baillargues | 1992-07-16 | 7 | juillet |
| Stéphane Brun | 1995-09-01 | 9 | septembre |
| Jean Bruneteau | 1992-09-02 | 9 | septembre |
| Bow Wow House | 1992-09-08 | 9 | septembre |
| Châtons Calins | 1993-05-06 | 5 | mai |
| … |  |  |  |

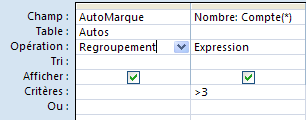
### Clause HAVING

* Permet d’établir une restriction portant sur un champ de sortie de regroupement.
* **ReqAuto12**
* Combien avons-nous de marques pour lesquelles il y a plus que 3 véhicules?

SELECT AutoMarque, COUNT(\*) AS Nombre  
FROM Autos  
GROUP BY AutoMarque  
HAVING COUNT(\*) > 3;

* Résultats :

| **Marque** | **Nombre** |
| --- | --- |
| Chevrolet | 4 |
| Chrysler | 4 |
| Mazda | 8 |
| Nissan | 6 |
| Toyota | 5 |

* Avec QBE :
* « Expression » génère *HAVING*

Remarques

* On ne peut pas faire : *Having nombre>3*
* *HAVING* est toujours associé à un *GROUP BY*
* Une instruction *WHERE* s’applique avant le regroupement des résultats, *HAVING* s’applique (normalement) après.
* **ReqAuto13**
* Quelles sont les marques qui ont au moins 2 autos de couleur « bleu ».

SELECT AutoMarque, COUNT(\*) AS Nombre  
FROM Autos  
WHERE AutoCouleur = "bleu"  
GROUP BY AutoMarque  
HAVING COUNT(\*)>1;

* Résultats :

| **Marque** | **Nombre** |
| --- | --- |
| Nissan | 3 |
| Toyota | 2 |

* On aurait les mêmes résultats avec :

SELECT AutoMarque, COUNT(\*) AS Nb  
FROM Autos  
GROUP BY AutoMarque, AutoCouleur  
HAVING (Count(\*)>1) AND (AutoCouleur = "bleu")

* Mais ce n’est pas toujours équivalent de faire une migration d’une restriction d’un *WHERE* vers un *HAVING.*
* Par exemple, si on change la restriction en haut par (AutoCouleur <> "bleu") les résultats seront différents car on ajoute un regroupement par couleur pour restreindre avec le *HAVING.* Il comptera donc les groupes de couleurs et non les couleurs à l’intérieur.

***Exercices sur Films.mdb***

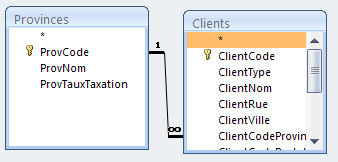
### Requêtes avec paramètres

* Access-SQL offre une instruction pour déclarer des paramètres dans une requête.

Parameters <nom\_param> <type> [,…n];

Comprendre que c’est une « variable »

* Elle se place avant le *SELECT*
* **ReqN**
* On veut connaître les clients demeurant dans une province donnée.



SELECT ProvNom, ClientNom, ClientVille  
FROM Provinces INNER JOIN Clients  
 ON Provinces.ProvCode = Clients.ClientCodeProvince  
WHERE ProvNom = UneProvince;

#### Dans la grille de saisie

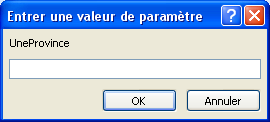
* L’identificateur est placé dans la ligne critère entre [ ].

OU

* Par le menu contextuel dans la partie du haut > Paramètres

#### Exécution

* Access demande une valeur pour le paramètre :



Notes

* Cette boîte de dialogue peut apparaître lorsque notre syntaxe est erronée et que Access tente d’interpréter que c’est un paramètre.
* Si on a 2 paramètres ou plus, Access les demande à tour de rôle.
* **ReqO**
* Je veux connaître les propriétaires qui ont des animaux qui sont nés à un mois donné (paramètre).

PARAMETERS UnMois Text(255);  
SELECT ClientNom, Aninom, AniNaissance  
FROM Clients INNER JOIN Animaux  
 ON Clients.ClientCode = Animaux.AniCodeClient  
WHERE Month(AniNaissance) = UnMois;

### Fonction utilisateur

* Access-SQL permet l’utilisation de fonctions écrites par le programmeur dans une instruction SQL;
* Exemple
* Dans un module j’écris en VBA.

Public Function DonnerAge(ByVal UneDate As Date) As Double  
  
'Objectif: retourner, sous forme d'un nombre réel la différence 'de temps entre aujourd'hui et UneDate en nombre d'années.  
  
Const JOURS\_PAR\_AN As Double = 365.24  
Dim Différence As Double  
Différence = CDbl(Date - UneDate)  
DonnerAge = Différence / JOURS\_PAR\_AN  
  
End Function

#### Utilisation

* **ReqP**
* On veut connaître l’âge des animaux vivants.

SELECT Aninom, Anitype, Aninaissance,  
 DonnerAge(Aninaissance) AS Age  
FROM Animaux  
WHERE NOT AniEstDécédé;

* 122 Résultats :

| **Nom** | **Type** | **Naissance** | **Age** |
| --- | --- | --- | --- |
| Bobo | LAPIN | avr 95 | 13,8922352425802 |
| Presto Chango | LEZARD | mai 95 | 13,8100974701566 |
| Aquilon | MOUFETTE | août 94 | 14,5575511992115 |
| Fido | CHIEN | juin 93 | 15,7239075676268 |
| Porky | COCHON | févr 94 | 15,0531157595006 |
| Rising Sun | CHEVAL | avr 93 | 15,8909210382214 |
| Désirée | CHIEN | févr 94 | 15,0531157595006 |
| Mikos | LOUP | nov 90 | 18,3057715474756 |
| … |  |  |  |

Conseils

* Assurez-vous que la fonction ne produit aucun effet de bord.
* Soyez conscients que ceci impliquera probablement une certaine conversion si on veut porter cette requête vers un autre SGBD.

### Traitement des valeurs nulles

Dans une requête utilisez ce champ, sinon la fonction renvoie une chaîne de longueur nulle.

#### Fonction *Nz*

* *Nz* pour **N**ull to **Z**ero

*Nz*(< expr > [, < valeur\_si\_nulle >])

* **ReqAuto14**

SELECT AutoMarque, Nz(AutoModèle, "inconnu") AS Modèle,  
 Nz(AutoPrix, 0) AS Prix  
FROM Autos;

* Résultats :

| **Marque** | **Modèle** | **Prix** |
| --- | --- | --- |
| Toyota | Corolla | 19500 |
| Ford | Grand Marquis | 26500 |
| Nissan | Sentra | 13565 |
| Chrysler | Intrepid | 25000 |
| Chrysler | Voyager | 32000 |
| Suzuki | Kia | 23500 |
| Chevrolet | Cavalier | 18795 |
| Mazda | 626 | 26795 |
| Daewoo | Lemans | 11995 |
| Pontiac | Lemans | 28795 |
| Toyota | Camry | 28995 |
| Ford | Mercury | 33995 |
| Ford | Grand Marquis | 29995 |
| Nissan | Pathfinder | 38995 |
| Nissan | Sentra | 14000 |
| Toyota | Echo | 15995 |
| Toyota | Camry | 29000 |
| Chevrolet | Sunfire | 19500 |
| Chevrolet | Impala | 8000 |
| Mazda | Protégé | 15500 |
| Mazda | Protégé | 16500 |
| Mazda | Protégé | 17450 |
| Chrysler | Jeep | 25495 |
| Chrysler | New Yorker | 18995 |
| Nissan | Pathfinder | 37995 |
| Mazda | inconnu | 19000 |
| Mazda | Protégé | 17000 |
| Mazda | 626 | 26500 |
| Toyota | Camry | 0 |
| Pontiac | Sunfire | 20000 |
| Pontiac | Sunfire | 19995 |
| Nissan | inconnu | 0 |
| Chevrolet | Cavalier | 17500 |
| Mazda | 626 | 25600 |
| Nissan | Pathfinder | 20000 |

* Peut-on y ajouter un *GROUP BY AutoMarque*?

*Non, si on ajoute un GROUP BY, tout Champ d’affichage (liste dans le* SELECT*) DOIT se retrouver dans la clause* GROUP BY (AutoMarque, AutoModèle, AutoPrix).

#### Fonction iif

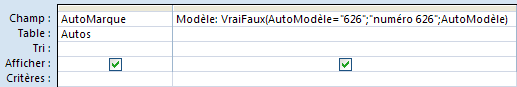
* Équivalent au si() de Excel

Iif(<condition>, <valeur\_si\_vrai>, <valeur\_si\_faux>)

* **ReqAuto15**

SELECT AniNom, Iif(AniSexe = "M", "Male”, "Femelle") as sexe

SELECT AutoMarque,   
 Iif(AutoModèle ="626", "numéro 626", AutoModèle) AS Modèle  
FROM Autos;

* Avec le générateur d’expression :

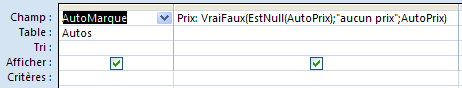
#### Fonction IsNull

* Retourne vrai ou faux

IsNull(< expr >)

**ReqAuto16**

SELECT AutoMarque,  
 Iif(IsNull(AutoPrix), "aucun prix", AutoPrix) AS Prix  
FROM Autos;

* Avec le générateur d’expression :

Ne pas faire :

WHERE AniRace = IsNull(Anirace), pas d’erreur mais pas de résultat!

Faire :

* WHERE Anirace is Null
* WHERE Anirace is not Null  
   **ou**
* WHERE not(AniRace is Null)

#### Formatage

Format(< expr >, < chaîne de format >)

* **ReqAuto17**

SELECT AutoMarque, AutoModèle,  
 Format(AutoPuissance, "00.00 cv") AS Puissance  
FROM Autos

* Résultats :

| **Marque** | **Modèle** | **Puissance** |
| --- | --- | --- |
| Toyota | Corolla | 138,00 cv |
| Ford | Grand Marquis | 175,00 cv |
| Nissan | Sentra | 103,00 cv |
| Chrysler | Intrepid | 160,00 cv |
| Chrysler | Voyager | 165,00 cv |
| Suzuki | Kia | 115,00 cv |
| Chevrolet | Cavalier | 115,00 cv |
| Mazda | 626 | 135,00 cv |
| Daewoo | Lemans | 90,00 cv |
| Pontiac | Lemans | 135,00 cv |
| Toyota | Camry | 175,00 cv |
| Ford | Mercury | 175,00 cv |
| Ford | Grand Marquis | 165,00 cv |
| Nissan | Pathfinder | 175,00 cv |
| Nissan | Sentra | 115,00 cv |
| Toyota | Echo | 105,00 cv |
| Toyota | Camry | 165,00 cv |
| Chevrolet | Sunfire | 115,00 cv |
| Chevrolet | Impala | 145,00 cv |
| Mazda | Protégé | 115,00 cv |
| Mazda | Protégé | 125,00 cv |
| Mazda | Protégé | 135,00 cv |
| Chrysler | Jeep | 135,00 cv |
| Chrysler | New Yorker | 165,00 cv |
| Nissan | Pathfinder | 175,00 cv |
| Mazda |  | 135,00 cv |
| Mazda | Protégé | 115,00 cv |
| Mazda | 626 | 165,00 cv |
| Toyota | Camry | 00,00 cv |
| Pontiac | Sunfire |  |
| Pontiac | Sunfire |  |
| Nissan |  |  |
| Chevrolet | Cavalier | 115,00 cv |
| Mazda | 626 | 165,00 cv |
| Nissan | Pathfinder | 165,00 cv |

### Quelques précisions

#### Sur ORDER BY

* Pas possible sur les types memo ou objetOLE
* C’est possible de faire :

SELECT Nom, Prénom, Age, Ville  
FROM Employés  
ORDER BY 3, 1 DESC;

* 3 étant le 3e élément dans la liste du SELECT et 1 le 1er
* C’est possible, mais moins explicite ou significatif, donc à ne pas faire.

#### Sur GROUP BY

* Si la liste du SELECT ne contient que des fonctions d’agrégat, le GROUP BY n’est pas obligatoire.

SELECT COUNT(…) AS Nb, MAX(…) AS Max  
FROM Table;  
 (ne pas mettre de GROUP BY)

### Sous-Requête

* C’est une instruction SELECT imbriquée dans une autre instruction SELECT.
* Elle est toujours entre parenthèses
* Elle peut servir dans :
* une liste de SELECT (ReqR)
* un WHERE (ReqQ, ReqS )
* un HAVING
* Avec le WHERE:
* Syntaxe 1 :

WHERE <expr> Opération\_de\_comparaison [ANY|ALL] (sous\_requête)

* Où ANY veux dire au moins 1 ( <expr> > ANY (sous-requête) renvoie vrai si expr est au moins plus grande que la plus petit valeur de la sous-requête).
* Où ALL veux tous (<expr> > ALL (sous-requête) renvoie vrai si expr est plus grande que toutes les valeurs de la sous-requête, donc plus grande que la plus grande).
* Syntase 2 :

WHERE <expr> [NOT] IN (sous-requête)

* Où IN renvoie vrai si expr se retrouve dans les résultats de la sous-requête
* Syntaxe 3 :

WHERE [NOT] EXISTS (sous-requête)

SELECT AniNom FROM Animaux

WHERE EXISTS (SELECT AniCode FROM Animaux WHERE AniSexe = ‘M’)

* Où EXISTS renvoie vrai s’il y a des résultats à la sous-requête.
* **ReqQ**
* Quels sont les animaux qui sont plus vieux que la moyenne des animaux appartenant à des clients qui sont des établissements commerciaux (type 2)? On veut les noms des animaux les plus jeunes en premier avec leur date de naissance et leur âge.

SELECT AniNom, AniNaissance, DonnerAge(AniNaissance) AS Âge  
FROM Animaux  
WHERE AniNaissance < ( SELECT AVG(AniNaissance)  
 FROM Clients INNER JOIN Animaux  
 ON Animaux.AniCodeClient =   
 Clients.ClientCode  
 WHERE ClientType = 2)  
ORDER BY AniNaissance DESC;

* Résultats :

| **Nom** | **Naissance** | **Âge** |
| --- | --- | --- |
| Tigre | févr 91 | 18,0703099331946 |
| Ming | févr 91 | 18,0703099331946 |
| Barney | févr 91 | 18,0703099331946 |
| Jerry | févr 91 | 18,0703099331946 |
| Clown | févr 91 | 18,0703099331946 |
| Jacquot | déc 90 | 18,2400613295367 |
| Mikos | nov 90 | 18,3221991019604 |
| Marcus | nov 90 | 18,3221991019604 |
| Vanessa | nov 90 | 18,3221991019604 |
| Floppy | oct 90 | 18,4070748001314 |
| Muffy | sept 90 | 18,489212572555 |
| Eclair | août 90 | 18,5740882707261 |
| Eclair | juil 90 | 18,6589639688972 |
| Babar | juil 90 | 18,6589639688972 |
| Ombre | mai 90 | 18,8259774394918 |
| Buck | avr 90 | 18,9081152119155 |
| Samson | avr 90 | 18,9081152119155 |
| Swinger | mars 90 | 18,9929909100865 |
| Sandy | mars 90 | 18,9929909100865 |
| Doudoune | mars 90 | 18,9929909100865 |
| Gizmo | mars 90 | 18,9929909100865 |
| Tortueux | mars 90 | 18,9929909100865 |
| Simplet | févr 90 | 19,0696528310152 |
| Tigre | févr 90 | 19,0696528310152 |
| Ben | janv 90 | 19,1545285291863 |
| Rose | nov 89 | 19,321541999781 |
| Brunaud | sept 89 | 19,4885554703756 |
| Caruso | avr 89 | 19,9074581097361 |
| Martin | avr 89 | 19,9074581097361 |
| Grimpeuse | mars 89 | 19,9923338079071 |
| Margo | mars 89 | 19,9923338079071 |
| … |  |  |

* **ReqR**
* Pour chaque client on veut son nombre actuel d’animaux et le pourcentage de ce nombre par rapport au nombre d’animaux vivants connus par la clinique. On veut les résultats en ordre décroissant de pourcentage.

SELECT ClientNom, COUNT(\*) AS NbAnimaux,  
 ( SELECT COUNT(\*) FROM Animaux WHERE NOT AniEstDécédé) AS Total,  
 FORMAT (COUNT(\*)/Total, "0.00%") AS Pourcentage

FROM Clients INNER JOIN Animaux ON Clients.ClientCode=Animaux.AniCodeClient  
WHERE Not AniEstDécédé  
GROUP BY ClientNom  
ORDER BY 4 DESC;

* Résultats :

| **ClientNom** | **NbAnimaux** | **Total** | **Pourcentage** |
| --- | --- | --- | --- |
| Le Monde des Oiseaux | 7 | 122 | 5,74% |
| UFR Sciences de la Vie | 6 | 122 | 4,92% |
| Royaume des Animaux (le) | 6 | 122 | 4,92% |
| Aquarium de Baillargues | 5 | 122 | 4,10% |
| Patricia Irribarado | 5 | 122 | 4,10% |
| George Grandjean | 4 | 122 | 3,28% |
| Animaux Câlins | 4 | 122 | 3,28% |
| Office Naturel de l'Hérault | 4 | 122 | 3,28% |
| La Grange au Poil | 3 | 122 | 2,46% |
| La Cuve à Poissons SARL | 3 | 122 | 2,46% |
| Zoo Municipal de Ranier | 3 | 122 | 2,46% |
| Pépiements | 3 | 122 | 2,46% |
| Philippe Primenat | 3 | 122 | 2,46% |
| La Maison de l'Animal | 3 | 122 | 2,46% |
| Des Oiseaux et des Hommes | 3 | 122 | 2,46% |
| Sandra Young | 3 | 122 | 2,46% |
| Bow Wow House | 3 | 122 | 2,46% |
| Au Chat Malin | 3 | 122 | 2,46% |
| Animaux Familiers | 3 | 122 | 2,46% |
| Zoo du Mont Pilote | 3 | 122 | 2,46% |
| Haute Protection | 2 | 122 | 1,64% |
| Châtons Calins | 2 | 122 | 1,64% |
| Barbara Willenberg | 2 | 122 | 1,64% |
| Jean Adams | 2 | 122 | 1,64% |
| Koalas et Perruches | 2 | 122 | 1,64% |
| Christian Praduit | 2 | 122 | 1,64% |
| Réserve Naturelle de Nîmes | 2 | 122 | 1,64% |
| L'Ile aux Serpents | 2 | 122 | 1,64% |
| Marguerite McKinley | 2 | 122 | 1,64% |
| Zoo Municipal d'Angliers | 2 | 122 | 1,64% |
| World Fauna | 2 | 122 | 1,64% |
| Pierrette Grandon | 2 | 122 | 1,64% |
| Plein d'Amour | 2 | 122 | 1,64% |
| … |  |  |  |

* **ReqS**
* Quels sont les animaux ayant fait une visite depuis le 3 juillet 1995, dont le nom commence par « M » et qui appartiennent à un client qui n’est pas un établissement public (clienttype=2)?
* **Version sans sous-requête :**

SELECT ClientNom, AniNom, VisitDate  
FROM (Clients INNER JOIN Animaux  
 ON Clients.ClientCode= Animaux.AniCodeClient)  
 INNER JOIN Visites ON Animaux.AniCode = Visites.VisitCodeAni  
WHERE VisitDate >= #7-3-95# AND AniNom LIKE "M\*"  
 AND ClientType <> 2;

* L’intérêt d’une sous-requête vient du fait que, si c’est possible, Access l’évalue qu’une seule fois.
* **Version avec sous-requête :**

SELECT AniNom, VisitDate  
FROM Animaux INNER JOIN Visites  
 ON Animaux.AniCode = Visites.VisitCodeAni  
WHERE VisitDate >= #7-3-95# AND AniNom LIKE "M\*"  
 AND AniCodeClient IN   
 (SELECT ClientCode FROM Clients   
 WHERE ClientType <> 2);

### Opérateur LIKE

* Syntaxe :

<expr> LIKE "modèle"

* Exemple 1

SELECT JoueurNom FROM Joueurs WHERE JoueurNom LIKE "\*krop\*"

Nom LIKE "L\*"

Chiffre

* Exemple 2

Numéro LIKE "A#BB"

De A à L

* Exemple 3

Nom LIKE "[A-L]\*"

sauf A à L

* Exemple 4

Nom LIKE "\*##[!A-L]\*"

### Opérateur Union

* Fusionne les résultats de 2 ou plusieurs requêtes en éliminant les doublons.
* Syntaxe :

[Table] *requête1*UNION [ALL]  
[Table] *requête2*  
[UNION [ALL] [Table] *requêteN*]

* **ReqFilms1**
* On veut connaître quels sont les acteurs et les réalisatrices canadiens encore vivants.

SELECT ActNom AS Nom, ActPrénom AS Prénom, ActSexe AS Sexe,  
 "Acteur" AS Information  
FROM Acteurs INNER JOIN Nationalités  
 ON Acteurs.ActNationalité = Nationalités.NatCode  
WHERE ActSexe = "M" AND ActAnDécès = 0 AND NatNom = "canadien"  
  
UNION  
  
SELECT RealNom, RealPrénom AS Prénom, RealSexe, "Réalisatrice" AS Information  
FROM Réalisateurs INNER JOIN Nationalités  
 ON Réalisateurs.RealNationalité = Nationalités.NatCodeRégion  
WHERE RealSexe = "F" AND RealAnDécès = 0 AND NatNom ="Canadien"  
  
ORDER BY Prénom;

Remarques

* Les 2 requêtes doivent avoir, le même nombre d’expressions dans le SELECT, mais avec des types « compatibles ».
* Le ORDER BY porte sur l’ensemble des résultats des 2 requêtes et présente des champs (ou expression) appartenant à la 1ère requête.
* Si on créé les requête au préalable, il est possible de faire.

Table reqActCanadiens  
UNION  
Table reqRealCanadiens

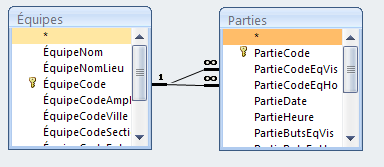
Note

Ici il n’est pas possible d’ajouter un *ORDER BY* car il doit porter sur des champs spécifiés dans la 1ère requête, ce qui n’est pas le cas ici.

* Clause *ALL*
* Si elle est absente, les doublons dans les résultats sont éliminés
* Si on veut tout de même ces doublons on ajoute *ALL*
* Précisions
* Chaque requête peut avoir son *GROUP BY*
* Tout *ORDER BY* ne s’appliquant pas à l’ensemble des résultats est inopérant.

### Exemple avec LNH.mdb

* Quelles sont les parties jouées par les « Devils »?

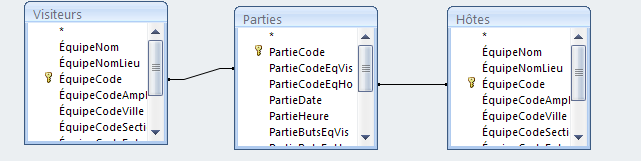


* **ReqPartiesDevils Version 1**

SELECT ÉquipeNom, PartieDate, PartieHeure  
FROM Équipes INNER JOIN Parties  
 ON Équipes.ÉquipeCode=Parties.PartieCodeEqVis  
WHERE Équipes.ÉquipeNom="devils" AND  
 Parties.PartieRésultatEqVis<>"à venir";  
  
UNION  
  
SELECT ÉquipeNom, PartieDate, PartieHeure  
FROM Équipes INNER JOIN Parties  
 ON Équipes.ÉquipeCode=Parties.PartieCodeEqHo  
WHERE ÉquipeNom="devils" AND PartieRésultatEqVis<>"à venir";

Remarques

* On peut identifier une table comme ceci : *NomTable AS AliasTable*
* Aux endroits où le nom de la table doit apparaître on le remplace par son alias

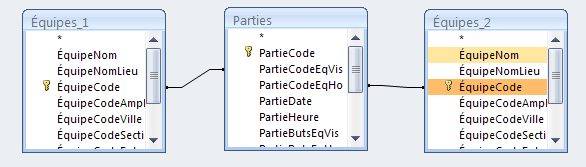


SELECT Visiteurs.ÉquipeNom, PartieDate, PartieHeure,  
 Hôtes.ÉquipeNom  
FROM Équipes AS Hôtes INNER JOIN  
 (Équipes AS Visiteurs INNER JOIN Parties  
 ON Visiteurs.ÉquipeCode=Parties.PartieCodeEqVis)  
 ON Hôtes.ÉquipeCode=Parties.PartieCodeEqHo  
WHERE (Visiteurs.ÉquipeNom="devils" OR  
 Hôtes.ÉquipeNom = "devils")  
 AND PartieRésultatEqVis<>"à venir";

* **ReqPartiesDevils Version 2**

SELECT ÉquipeNom, PartieDate, PartieHeure  
FROM Équipes INNER JOIN Parties  
 ON Équipes.ÉquipeCode=Parties.PartieCodeEqVis  
 Or ÉquipeCode=PartieCodeEqHo  
WHERE ÉquipeNom="devils" AND PartieRésultatEqVis<>"à venir";

* **ReqPartiesDevils Version 3**

****

* Access écrira :

…

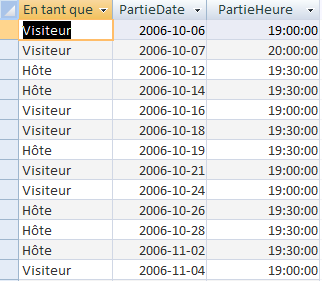
FROM Équipes AS Équipes\_2 INNER JOIN  
 (Équipes AS Équipes\_1 INNER JOIN Parties  
 ON Équipes\_1.ÉquipeCode = Parties.PartieCodeEqVis)  
 ON Équipes\_2.ÉquipeCode = Parties.PartieCodeEqHo

…

* On peut imaginer une version 4, si dans la version 1 on a ReqPartiesDevilsVis et ReqPartiesDevilsHo.

Table ReqPartieDevilsVis  
UNION  
Table ReqPartieDevilsHo

Discussions

* Version 4 : telle qu’écrite empêche de faire un ORDER BY
* Versions 1,2 et 4 : empêche de faire afficher le nom de l’équipe adversaire
* Avec la version 2 il serait possible de montrer
* 
* SELECT IIf(ÉquipeCode=PartieCodeEqVis,"Visiteur","Hôte") AS [En tant que],  
   Parties.PartieDate, Parties.PartieHeure  
  FROM Équipes INNER JOIN Parties  
   ON Équipes.ÉquipeCode=Parties.PartieCodeEqVis  
   Or ÉquipeCode=PartieCodeEqHo  
  WHERE ÉquipeNom="devils" AND PartieRésultatEqVis<>"à venir";

## Les Requêtes d’action

* Elles sont disponibles via le groupe *Type de requête* de l’onglet *Créer*.

### Création de table

* Syntaxe :

SELECT <liste\_de\_champ> INTO <nom\_nouvelle\_table  
FROM <source\_de\_données>  
[WHERE <cond>]

* **ReqT**
* On veut avoir une (nouvelle) table ne contenant que des animaux vivants.

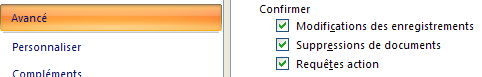
SELECT Animaux.\* INTO AnimauxVivants  
FROM Animaux  
WHERE NOT AniEstDécédé;

* Voici la démarche dans l’interface QBE

1. Créer une nouvelle requête
2. Placer la source de données dans la fenêtre (table, tables jointes, requête)
3. Sélectionner tous les champs
4. Établir les critères
5. Bouton *Création de table* du groupe *Type de requête*
6. Donner un nom à cette nouvelle table

Note

À l’exécution Acess nous demande une confirmation, à cause de l’option suivante :



* La nouvelle table créée avec cette requête :

1. a la même structure que la table Animaux, donc des champs identiques en type et en taille.
2. contient des enregistrements rencontrant le critère de restriction.
3. ne conserve pas des propriétés comme légende, nulle interdit, indexé (donc **clé primaire non reproduite**), …

* **ReqJoueursÉquipe**
* On veut créer une table ne contenant que les noms et numéros de chandail des joueurs d’une équipe donnée.

PARAMETERS Équipe Text(255);  
SELECT JouNom, JouPrénom, ÉquipeNom, JouNoChandail  
 INTO JoueursDeÉquipe  
FROM Joueurs INNER JOIN Équipes  
 ON Joueurs.JouCodeÉquipe=Équipes.ÉquipeCode  
WHERE JouEstActif And ÉquipeNom=Équipe;

### Mise à jour

* Permet de modifier un ou plusieurs champs de plusieurs enregistrements d’une table ou de plusieurs tables jointes.
* Syntaxe

UPDATE <exp\_tab\_req>  
SET <liste\_exp>  
[WHERE <critères>]

* **ReqU**
* Un chien femelle aura un type animal « chienne » et une race « canine ».

UPDATE Animaux  
SET Anitype = "chienne", AniRace = "canine"  
WHERE AniType="chien" AND AniSexe="F";

* **ReqVAvantV**

SELECT Aninom, AniEstStérilisé, ClientCodeProvince  
FROM AnimauxOntariensStérélisés INNER JOIN Clients  
 ON AnimauxOntariensStérélisés.AniCodeClient =  
 Clients.ClientCode  
WHERE ClientCodeProvince = "ON"  
ORDER BY AniEstStérilisé;

* Résultats :

| **Nom** | **Stérilisé** | **ClientCodeProvince** |
| --- | --- | --- |
| Fi Fi | Oui | ON |
| C.C. | Oui | ON |
| Gizmo | Oui | ON |
| Doudoune | Oui | ON |
| Roméo | Oui | ON |
| César | Oui | ON |
| Désirée | Oui | ON |
| Tigre | Oui | ON |
| Micro | Oui | ON |
| Eclair | Oui | ON |
| Tonnerre | Oui | ON |
| Sylvester | Oui | ON |
| Juliette | Oui | ON |
| Simplet | **Non** | ON |
| Ren | **Non** | ON |
| Stimpy | **Non** | ON |
| Clone | **Non** | ON |
| Strident | **Non** | ON |
| Samson | **Non** | ON |
| Dalila | **Non** | ON |
| Gabriel | **Non** | ON |
| Adam | **Non** | ON |
| Tueur | **Non** | ON |
| Fuyant | **Non** | ON |
| Hop | **Non** | ON |
| Swinger | **Non** | ON |
| Muffy | **Non** | ON |
| Beauté | **Non** | ON |
| Oren Girl | **Non** | ON |
| Babar | **Non** | ON |
| Benjamin | **Non** | ON |
| Blanchette | **Non** | ON |
| Ouatte | **Non** | ON |
| Sauteur | **Non** | ON |
| Houpette | **Non** | ON |
| Sauvage | **Non** | ON |
| Sammie | **Non** | ON |

* **ReqV**
* La clinique a stérilisé tous les animaux appartenant à des clients ontariens.

UPDATE AnimauxOntariensStérélisés INNER JOIN  
 (Clients INNER JOIN Provinces  
 ON Clients.ClientCodeProvince = Provinces.ProvCode)  
 ON AnimauxOntariensStérélisés.AniCodeClient =  
 Clients.ClientCode  
SET AniEstStérilisé = true  
WHERE ProvNom = "Ontario";

### Suppression d’enregistrements

* Pour supprimer 1 ou plusieurs enregistrements d’une table.

DELETE [Table.\*]  
FROM Table  
[WHERE <cond>]

* **ReqJoueursSupprimés**

DELETE \*  
FROM JoueursSupprimé;

* Détruit tous les enregistrements de la table sans supprimer la table elle-même (la structure est préservée).
* **Attention à l’intégrité référentielle.**
* Pourquoi y-a-t-il « 200 » et « 6 » d’inscrit dans le premier enregistrement?

Parce que ce sont des valeurs par défaut.

* **ReqJoueursSupprimésBruins**

DELETE JoueursSupprimés2.\*  
FROM JoueursSupprimés2 INNER JOIN Équipes  
 ON JoueursSupprimés2.JouCodeÉquipe = Équipes.ÉquipeCode  
WHERE ÉquipeNom="Bruins";

* Ne supprime que les joueurs des Bruins

Remarque

* Ne pas oublier que si les suppressions en cascade sont autorisées dans la table où se font les DELETE, des enregistrements de d’autres tables seront possiblement supprimés.

### Supprimer une table

* **ReqSuppriméTableJoueurs**

DROP TABLE JoueursSupprimés2;

* Supprime les enregistrements et la structure.

### Ajout d’enregistrements

#### Syntaxe 1

INSERT [INTO] <nom\_Table> [(liste\_de\_champs)]  
SELECT <liste\_de\_Select>  
FROM <Table\_Req>  
[WHERE <cond>]

* Permet d’ajouter plusieurs enregistrements dans une table.
* **ReqCréerJoueurAvant**
* Disons qu’on veut créer une table JoueursAvant vide.

CREATE TABLE JoueursAvant (AvantNomPrénom text(255),  
 AvantCode integer,  
 AvantCodePosition integer,  
 AvantÉquipe text(255) );

* **ReqRemplirJoueursAvant**
* On veut lui ajouter les joueurs actuels de la section « Nord-est ».

INSERT INTO JoueursAvant(AvantNomPrénom, AvantCode,  
 AvantCodePosition, AvantÉquipe)  
SELECT JouNom & ", " & JouPrénom AS Nom, JouCode,  
 JouCodePosition, ÉquipeNom  
FROM (Sections INNER JOIN Équipes  
 ON Sections.SectCode =Équipes.ÉquipeCodeSection)  
 INNER JOIN Joueurs ON Équipes.ÉquipeCode=  
 Joueurs.JouCodeÉquipe  
WHERE SectNom ="Section Nord-Est" AND JouCodePosition >=2  
 AND JouCodePosition <=4 AND JouEstActif;

Note

La liste de champ de la table d’insertion doit coïncider avec la liste de SELECT

* Exemple d’un cas plus simple
* On a une table JoueursXXX de même structure que la table Joueurs.
* On pourrait faire :

INSERT INTO JoueursXXX  
SELECT \*  
FROM Joueurs

* On se retrouverait avec 2 tables identiques si on avait aucun enregistrement à l’intérieur de la table JoueursXXX

Note

Cette requête fonctionne uniquement si JoueursXXX a exactement **la même structur**e que la table Joueurs et **les mêmes identificateurs** de champ, sinon il faut identifier une liste pour la table d’insertion, comme on a fait dans le premier exemple.

Important

* INSERT INTO table : ajoute des enregistrements à la table
* SELECT … INTO table : crée une nouvelle table ou écrase la table actuelle

#### Syntaxe 2

INSERT INTO <nom\_table> [(champ1,champ2,… champn)]  
VALUES (val\_champ1, val\_champ2,… val\_champn)

* On a une table Articles
* ArtCode numérotation automatique
* ArtNom texte
* ArPrix monétaire
* ArtDisponible oui/non

INSERT INTO Articles (ArPrix, ArtDisponible, ArtCode, ArtNom)  
VALUES (22.50, true, 8, "Couteau")

* On pourrait faire aussi :

INSERT INTO Articles  
VALUES (22, "lames de rasoir", 0.55, yes)

Note

Si on ne met pas de liste de champs, toutes les valeurs doivent y être et de bon type.

* Comme ArtCode est une numérotation automatique les 2 requêtes suivantes fonctionnent!

INSERT INTO Articles (ArtNom, ArtPrix, ArtDisponible)  
VALUES ("ça fonctionne!", 3.45, no)

* Si ArtPrix et ArtDisponible ont des valeurs par défaut

INSERT INTO Articles (ArtNom)  
VALUES ("ça fonctionne aussi!!!")

* VALUES accepte, en fait, une liste d’expression

INSERT INTO Articles (ArtNom, ArtPrix)  
VALUES ("bidule", 15\*10)

**OU**

INSERT INTO Articles (ArtNom, ArtPrix)  
VALUES ("cossin", abs(-55))

Attention :

INSERT INTO Articles (ArtNom, ArtPrix)  
VALUES ("kessécadonne", curieux)

Avec l’interface, Access pensera que *curieux* est un paramètre! Mais dans l’exécution du code VBA, on devrait avoir une erreur.

### Exemples avec LNH.mdb

#### Exemple 1

* Req1>PoidsCanadiens
* Nommez un joueur actif dont le poids est supérieur au poids de tout joueur des canadiens.

SELECT Top 1 JouNom, JouPrénom, JouPoids  
FROM Joueurs  
WHERE JouEstActif AND JouPoids > ALL (SELECT Joupoids  
 FROM Joueurs INNER JOIN Équipes  
 ON Équipes.ÉquipeCode=Joueurs.JouCodeÉquipe  
 WHERE ÉquipeNom = "Canadiens" AND JouEstActif)

* Si on utilisait ANY à la place de ALL, ça voudrait dire qu’on voudrait un joueur ayant poids plus grand que n’importe quel joueur des Canadiens (au moins plus grand que le plus petit des poids).

#### Exemple 2

* Souvent une sous-requête peut être traitée avec une requête utilisant une autre requête.
* On veut connaître une équipe qui n’a pas fait exactement 6 buts durant une partie à la maison.
* ReqÉquipePas6ButsMaisonAvecSS

SELECT TOP 10 ÉquipeNom  
FROM Équipes  
WHERE ÉquipeNom NOT IN (SELECT DISTINCT ÉquipeNom  
 FROM Équipes INNER JOIN Parties  
 ON Équipes.ÉquipeCode=Parties.PartieCodeEqHo  
 WHERE PartieButsEqHo=6);

* ReqÉquipePas6ButsMaisonSansSS

SELECT TOP 1 Équipes.ÉquipeNom  
FROM Équipes LEFT JOIN Req6Buts  
 ON Équipes.ÉquipeNom = Req6Buts.ÉquipeNom  
WHERE Req6Buts.ÉquipeNom is null;