Formation base de données et Langage SQL pour non-informaticiens

1

Langage SQL

Participants et pré requis

- **Participants**
- Chargé de reporting ou d'analyse, assistant(e), toute personne ayant des besoins d'interrogation ou de mises à jour simples d'une base de données avec le langage SQL.

- **Prérequis**
- Aucune connaissance particulière.
- Formation commune à toutes les bases relationnelles (Oracle, SQL Server, DB2, PostGreSQL, MySQL, Access, SQL Lite...).

2

Langage SQL

La Formatrice

- Laure BERENGUER, consultante et formatrice indépendante

- Page du site internet (exercices et support) :

http://berenguer-formation-conseil.fr/langage-sql-adventureworksdw2019/

- laure@berenguer.onmicrosoft.com
- http://www.linkedin.com/in/laure-berenguer38/

3

Langage SQL

Objectif de la formation

- Comprendre le principe et le contenu d'une base de données relationnelle
- Créer des requêtes pour extraire des données suivant différents critères
- Utiliser des calculs simples et des agrégations de données
- Réaliser des requêtes avec des jointures, pour restituer les informations de plusieurs tables
- Combiner les résultats de plusieurs requêtes

4

Langage SQL

Le Sommaire

5

Langage SQL

INTRODUCTION AUX BASES DE DONNÉES

6

Eléments constitutifs

Requêtes mono-tables

- Les tables
- Les champs (colonnes) et les types de données
- Les enregistrements (lignes)

Requêtes multi-tables :

- Le modèle Physique de données (MPD) **indispensable**
- Les clés : clés primaires et clés étrangères
- Les relations entre les tables

7

Qu'est-ce qu'un serveur et une base de données ?

- Un serveur de gestion de base de données (SGBD) est un logiciel qui permet de stocker et gérer les bases de données sur un serveur.
- Une base de données est un ensemble structuré et organisé en tables permettant le stockage de grandes quantités d'informations afin d'en faciliter l'exploitation : ajout, mise à jour, recherche et analyse des données.

8

Qu'est-ce qu'une table et des champs ?

- Une **table** correspondant à une **entité particulière** : client, produit, commande, employé, etc.
- Chaque **enregistrement/ligne d'une table** correspondant à un **individu statistique différent.**
- Chaque enregistrement contient un **identifiant** – souvent la première colonne/champ de la table – puis **d'autres colonnes** qui apportent des informations relatives à l'individu/enregistrement en question.

Id Client	Nom	Prénom	Ville
1	DUPONT	JOHN	PARIS
2	DUPONT	PAUL	LYON
3	BART	ELODIE	MARSEILLE

9

Qu'est-ce que les clés et les relations ?

- Une **clé primaire (Primary Key = PK)** est l'identifiant de l'enregistrement d'une table : l'identifiant/numéro du client de la table client. Chaque clé primaire **est unique** et il ne peut pas y avoir deux fois le même client dans la table.
- Une **clé étrangère** (**FK**) est l'identifiant d'un enregistrement provenant d'une autre table. Elle fait toujours référence à une PK dans une autre table : l'identifiant du client dans la table commande. Elle peut donc être en doublons si le client a commandé plusieurs fois par exemple.
- C'est la relation d'égalité entre les deux qui permet la relation entre la table client et la table commande.

Id Client : clé primaire	Nom	Prénom	Ville
1	DUPONT	JOHN	PARIS
2	DUPONT	PAUL	LYON
3	BART	ELODIE	MARSEILLE

Num Commande	Id Client : clé étrangère	Date
12	1	30-déc-21
13	1	31-déc-21
14	2	31-déc-21

Base de données : Structure

Les types de données courants

Texte

- Char(n) 0 à 255
- Char(10) : 10 octets sur le disque /et en mémoire même si la valeur du champ est « Paris »
- Varchar(10) : 5 octets sur disque/mémoire si le champ contient « Paris »

- **Numérique (entier ou décimal)**
- Int : integer (Entier)
- Decimal (x,y) : Decimal (5,2)
- Numeric (x,y) : Identique à Decimal
- Float : Réservé aux calculs scientifiques (très gourmand en mémoire)

- **Date**
- DateTime (Année, mois, jour, heures, minutes, secondes, ms)
- Date (Année, mois, jour)

11

Base de données :

Conclusion

Une base de données est :

- Un ensemble de données organisées et reliées
- Dans (et entre) différentes tables composées
- De champs (dont certains clés primaires)
- De types numériques, textes ou dates

Qui nous permet d'enregistrer, modifier et d'extraire de l'information !

12

Langage SQL

EXTRAIRE LES DONNEES D'UNE TABLE

Extraire les données d'une table

Pour extraire les données d'une (ou plusieurs) table en SQL il faut :

Bien connaître le modèle physique (MPD) du sous-système de base de données, le contenu, les règles métiers..

Et bien savoir ce que l'on veut !!

SELECT et la syntaxe SQL

SELECT est la commande de base du SQL destinée à **extraire des données**

3 mots clés :

```
SELECT
        Champ1,
        Champ2
FROM
        Table
WHERE
        Champ3 ='test'
```

Cette requête SQL va **sélectionner et afficher** (SELECT) le champ « nom_du_champ »

provenant (FROM) du tableau appelé « nom_du_tableau ».

Contenant (**condition**) quand le «nom_du_champ» est

15

SELECT et la syntaxe SQL (2)

3 types de select:

```
SELECT *
        FROM
```

Retourne et affiche l'ensemble des champs (colonnes) et des lignes. (A éviter)

```
SELECT TOP 100 *
        FROM
```

Retourne l'ensemble des champs (colonnes) et seulement les 100 premières lignes : permet de visualiser les données et ce que contient la table sans problématique de performance.

16

Filtrage des données :
La clause DISTINCT

```
use comptoir

select Distinct Clientpays
From T_Client
```

Clientpays
Argentina
Austria
Belgium
Brazil
Canada
Denmark
DEUTSCHLAND
Finland
FRANCE

Le DISTINCT se fait sur l'ensemble des champs du Select

Il permet de supprimer les doublons sur les lignes.

17

Les commentaires en SQL

- Introduits par « -- » (une seule ligne
- Entourés par /*,,,,,,,,,*/ (plusieurs lignes)

```
/*
Afficher les 100 premieres lignes
Nom, region, Pays à partir de la table client
*/
-- Triées par pays (croissant)
use comptoir
select TOP 100
ClientNom,
ClientRegion,
ClientPays

from T_Client
order by ClientPays
```

18

La commande SELECT :

La clause TOP à travers différents SGBD

```
-- SQL SERVEUR
Select TOP 10 *
From DimCustomer

-- ORACLE
Select *
From DimCustomer
WHERE ROWNUM <=10

-- MYSQL
Select *
From DimCustomer
LIMIT 10
```

19

La commande SELECT :

Sensibilité à la casse (majuscule / minuscule)

Ces 2 requêtes produisent un résultat **différent** si le serveur (ou la base) est sensible à la casse

```
use COMPTOIR_OLTP
select * from
T_Client
Where ClientVille ='paris'
```

ou

```
use comptoir_OLTP
select * from
t_CLIENT
Where ClientVille ='Paris'
```

20

Renommer les colonnes

```
-- Il existe deux manières de renommer les colonnes
:

use comptoir
Select CLIST as 'Nom Client',
   TILL as 'Tel Client'
From T_Client



Select 'Nom Client'= CLIST,
   'Tel Client'= TILL
From T_Client
```

21

La commande SELECT :

La clause ORDER BY

```
-- La clause ORDER BY permet de trier les données. C'est
la dernière clause dans l'ordre de la syntaxe SQL :

use comptoir
Select ClientNom, ClientRegion, clientpays
From T_Client
order by ClientPays, ClientVille desc

-- ASC signifie ascendant : pas besoin de l'écrire dans
SQL SERVER.
-- DESC signifie descendant : besoin de le mentionner car
par défaut, la clause ORDER BY trie par ordre ascendant.
```

Attention, pour les autres SGBD, mettre ***ORDER BY clientpays ASC***

22

La commande SELECT :

La clause WHERE : Les critères simples = et <>

Afficher les clients non parisiens dont le nom commence par P ou F et qui ont plus de 30 ans

```
SELECT * FROM T_Client
WHERE ClientPays = 'France' AND
  ClientVille <> 'Paris'
AND AGE > 30          -- cote/' pas obligatoire car numérique
```

Texte :	**Numérique** :	
	=25	égal à la valeur 25
= 'France'		
	>50	strictement supérieur à 50
<> 'Spain'	>= 50	supérieur ou égal à 50
	< 50	strictement inférieur à 50
	<= 50	inférieur ou égal à 50
	<>25	différent de 25

23

La commande SELECT :

La clause WHERE : (not) IN : (pas) dans la liste suivante

```
select * from t_client
where ClientPays
Not in ('France','italy')
```

Je sélectionne toutes les colonnes et lignes de ma table t_client.
Quand le pays du client n'est pas, ni l'Italie, ni la France.

```
Select *
from Orders
where
Year(orderdate) in (2008,2009,2012)
and
Month(orderdate)  in (4,10)
```

Year & Month : fonctions intégrées présentées dans la section ***Fonctions intégrées de date.***
Elles permettent d'aller chercher/piocher une année dans une date de type DateTime ou Date. *Correspondent aux fonctions ANNEE et MOIS dans Excel.*

Ici je sélectionne les années 2008, 2009 et 2012 dans le champ order date puis les mois d'avril et d'octobre (4 et 10).

24

La commande SELECT :

La clause WHERE : (not)LIKE

La commande LIKE permet de sélectionner ce qui contient de partout, ou juste au début ou juste à la fin :

```
SELECT * FROM T_Client
WHERE ClientNom like 'S%'
-- Ce qui commence par S car ensuite pourcentage qui
signifie "qu'importe ce qu'il y a après".

SELECT * FROM T_Client
WHERE ClientNom like '%S'
-- Ce qui finit par B car ensuite pourcentage qui signifie
"qu'importe ce qu'il y a avant".

SELECT * FROM T_Client
WHERE ClientNom like '%S%'
-- Ce qui commence par B car ensuite pourcentage qui
signifie "qu'importe ce qu'il y a avant et après".
```

25

La commande SELECT :

La clause WHERE : Between Vs > & < (Entre)

La commande Between va permettre de gérer des intervalles et éviter d'écrire deux conditions (plus grand et plus petit) :

```
select *
From T_Commande
where CdeDate >= '20040705'
AND CdeDate < '20040715'

select *
From T_Commande
where CdeDate between '20040705' AND '20040714'

-- La commande Between prendra 3 fois moins de
temps à l'exécution que l'exemple au dessus.
```

CdeNum	ClientID	EmployeID	CdeDate
10249	TOMSP	6	05/07/2004
10250	HANAR	4	08/07/2004
10251	VICTE	3	08/07/2004
10252	SUPRD	4	09/07/2004
10253	HANAR	3	10/07/2004
10254	CHOPS	5	11/07/2004
10255	RICSU	9	12/07/2004

26

La commande SELECT :

La clause WHERE : Synthèse

= <>	exactement égal/différent à une seule valeur *where ClientPays <> 'France'*
IN / NOT IN	exactement égal/différent à un ensemble de valeurs *where ClientPays not in ('France','Espagne')* ~~*where (ClientPays <> 'France' or*~~ ~~*ClientPays <> 'Espagne)*~~
LIKE / NOT LIKE	contient/commence/finit par telle(s) caractère(s) *where ClientPays like '%rance%'* *Si plus de deux critères sur la même colonne (OR + LIKE) :* *where (ClientPays not like 'Franc%'* *OR* *ClientPays not like '%pagne')*
Between .. and…	entre telle et telle valeur *where nombers between 5 and 7*

La commande SELECT :

Règles d'écritures : (erreurs surlignées en rouge)
Les erreurs de code sont souvent les mêmes

```
use ContosoRetailDW
select
FirstName,
LastName,
Education
From DimCustomer
Where Education = 'Bachelor'

use comptoir
Select
YEAR(Cdedate) as Année,
COUNT(Cdenum) as 'Nombre de
commandes'
From T_commande
Group by YEAR(cdedate)
```

Les champs sélectionnés doivent être séparés d'une virgule (comme lorsque l'on énumère plusieurs exemples (sauf pour le dernier qui clôture)

Lorsque l'on renomme un champ, calcul ou lorsque l'on met une condition : ne pas oublier de mettre des cotes

Lorsque l'on fait un calcul (utilisation d'une fonction d'agrégation) = il faut bien remettre les autres champs du select dans le Group By

La commande SELECT :

La clause Group by :Astuces et utilisation (1/3)
Je compte/somme/etc par les items d'autre(s) colonne(s)

Afin de ne pas le rater, faire **attention à la formulation** : le **mot PAR** est un indice. S'il est dans la phrase et qu'il n'est pas précédé par trier alors c'est qu'un Group By est nécessaire.

Exemples :
- Calculer la **somme des montants des ventes par catégorie de produits** ?
- Calculer le **nombre de clients par ans et par mois** ?
- Calculer **la moyenne des coûts par classe et type de produits** ?

Quels sont les **trois points communs** de tous ces exemples :
- Il y a un calcul, une **agrégation** (count, sum, average, min, max) à faire,
- **Par** quelque chose : pour chaque item des colonnes précisées ensuite (un résultat/le nombre de clients pour chaque année et mois différent),
- Les colonnes par lesquelles nous souhaitons calculer pour chacune de leurs valeurs/Items. Ces **colonnes précisées après le PAR** seront celles à **mentionnées dans le GROUP BY**.

29

La commande SELECT :

La clause Group by :Astuces et utilisation (2/3)
Je compte/somme/etc par les items d'autre(s) colonne(s)

Exemples :
- Calculer la **somme des montants des ventes par catégorie de produits** ?

Commencer par le calcul

```
Select SUM(montant_ventes) as 'Somme montant des ventes'
From TVentes
```

Il ne peut y avoir qu'une seule ligne de résultat correspondant au montant des ventes de toutes les ventes, de toutes les lignes de la table. Le calcul agrège toutes les lignes.

Ecrire le **PAR**
- 1 : Par catégorie de produits : ce qui signifie que la colonne correspondante doit être **écrite dans le Group By afin qu'il regroupe pour chaque catégorie différente**, qu'il calcule pour chaque catégorie différente. Il n'y aura donc pas un seul résultat mais autant de résultats qu'il y a de catégories différentes.

```
Select SUM(montant_ventes) as 'Somme montant des ventes'  ,categorieProd (2)
From TVentes
GROUP BY categorieProd (1)
```

- 2 : Pour finir, afin **d'afficher les catégories de produits, copier/coller le contenu du Group by dans le SELECT** afin de voir la colonne !

30

La commande SELECT :

La clause Group by :Astuces et utilisation (3/3)
Je compte/somme/etc par les items d'autre(s) colonne(s)

Exemple :
- Calculer **le nombre de ventes et la moyenne des coûts par classe et type de produits** ?

1. Commencer par les calculs

```
Select COUNT(IDVENTE) as 'nombre de ventes',
       AVG(TOTALCOST) as 'Moyenne des coûts'
From TVentes
```

2. Ecrire le **PAR**

- Par classe et type de produits : ce qui signifie que les colonnes correspondantes doivent être dans le **Group By afin qu'il regroupe pour chaque classe et type de produit différents.**

```
Select COUNT(IDVENTE) as 'nombre de ventes',
       AVG(TOTALCOST) as 'Moyenne des coûts' , classe, type   (2)
From TVentes
GROUP BY classe, type   (1)
```

- 2 : Pour finir, **copier/coller le contenu du Group by dans le SELECT** afin de voir le contenu des colonnes classe et type !

31

La commande SELECT :

La clause Having : différence avec le Where

```
select ColorName,
   COUNT(productkey) as 'nombre de produits'
from DimProduct
where ProductKey between 2 and 1216
-- la clause where est en 3ème position dans la
syntaxe, AVANT le regroupement.
Il s'effectue sur un CHAMP de la base de données.
→ Quand l'identifiant du produit est égal à un numéro
entre 2 et 1216.
group by colorname

having COUNT(productkey) > 50
-- la clause having est en 5ème position dans la
syntaxe, APRES le regroupement.
Il s'effectue sur une FONCTION D'AGREGATION, calcul
après regroupement.
→ Quand le nombre de produit - count(productkey) //
fonction d'agrégation – PAR couleur est supérieur à 50.
```

32

La commande SELECT :

La clause CASE

Le CASE permet de **créer une nouvelle colonne** à partir de **conditions sur d'autres colonnes.**

Si dans la colonne genre, il y a la valeur M, alors Homme dans nouvelle colonne…

Si le montant des ventes par client est supérieur a tant, alors bon client…

```
select CdeNum, SUM(qte_vente) as
                    'somme des ventes',
   CASE
WHEN SUM(Qte_vente) < 50 THEN 'Faible'
WHEN SUM(Qte_vente) > 70 THEN 'Elevé'
ELSE 'Modéré'
   END as 'Qualification du montant de
ventes'

From T_DetaillerCommande
Group by CdeNum
```

CdeNum	somme des ventes	Qualification du montant de ventes
10411	74	Elevé
10743	28	Faible
11075	42	Faible
10388	75	Elevé
10720	29	Faible
11052	40	Faible
10457	36	Faible
10789	93	Elevé
10434	24	Faible
10766	115	Elevé
10835	17	Faible
10906	15	Faible
10912	100	Elevé
10265	50	Modéré

33

La commande SELECT :

La clause CASE (exemples dates)

```
-- Combien de commandes par type d'envoi ?
select COUNT(*) as 'nombre de commandes',

CASE
WHEN CommandeShippedDate IS NULL THEN 'Commande non
                                       envoyée'
WHEN DATEDIFF(Day,cdedate,CommandeShippedDate) < 5 THEN
                                       'Envoi rapide'
WHEN DATEDIFF(Day,cdedate,CommandeShippedDate) < 10 THEN
                                       'Envoi moyen'
ELSE 'Envoi lent'
   END as 'Type d'envoi'

From T_Commande

group by
CASE WHEN CommandeShippedDate IS NULL THEN 'Commande non envoyée'
WHEN DATEDIFF(Day,cdedate,CommandeShippedDate) < 5 THEN 'Envoi rapide'
WHEN DATEDIFF(Day,cdedate,CommandeShippedDate) < 10 THEN 'Envoi moyen'
ELSE 'Envoi lent' END
```

34

La commande SELECT :

La clause CASE (Oracle)

```
-- On peut mettre autant de conditions sur autant de colonnes
que l'on souhaite pour créer notre colonne conditionnelle :

select COUNT(customerkey) as 'nombre de clients',
   CASE
WHEN gender IS NULL THEN 'Entreprise'
WHEN Gender='M' AND MaritalStatus='M' THEN 'Homme marié'
WHEN Gender='F' AND MaritalStatus='M' THEN 'Femme mariée'
WHEN Gender='M' AND MaritalStatus='S' THEN 'Homme
                                           célibataire'
WHEN Gender='F' AND MaritalStatus='S' THEN 'Femme
                                           célibataire'
   END as 'genre et situation familiale'
From DimCustomer

Group by Gender, MaritalStatus
```

35

Langage SQL

LES CALCULS ET FONCTIONS INTÉGRÉES

36

Les calculs / Fonctions intégrées

- Calculs simples
- Fonctions d'agrégations
- Fonctions DATE
- Fonctions TEXTE
- Fonctions CONVERT

Les calculs / fonctions intégrées

Champs calculé simple

```
-- Possibilité de faire des soustractions,
multiplications et divisions assez facilement :

select top 10
SalesAmount as 'vente',
DiscountAmount as 'remise',
SalesAmount - DiscountAmount as 'hors remise'
From FactOnlineSales
```

vente	remise	hors remise
10,36	2,59	7,77
3,984	0,996	2,988
39,968	9,992	29,976
23,992	5,998	17,994
207,992	51,998	155,994
319,2	79,8	239,4
3,792	0,948	2,844
0	0,948	-0,948
166,4	41,6	124,8
0	41,6	-41,6

Les calculs / fonctions intégrées

Les fonctions d'agrégations

Les instructions d'agrégation permettent des opérations comme le comptage ou les sommes.

Les principales fonctions d'agrégation sont :

- AVG(<champs>), COUNT(<champs>), SUM (<champs>), MIN et MAX.

```
-- nombres de commandes passées en 2004
Select COUNT(cdenum) as 'nombre de commandes'
from t_commande
where YEAR(cdedate) = 2004

-- Le poids moyen des commandes par ville
Select AVG(CommandePoids) as 'Moy poids',ShipCity
from T_Commande
group by ShipCity

-- Chiffre d'affaires global
select SUM(Qte_vente * PU_Vente) as CA
from T_DetaillerCommande
```

39

Les calculs / fonctions intégrées

Les fonctions d'agrégations

GROUP BY ou La clause OVER (partition order by)

```
SELECT
salesorderID,
sum(orderqty) as 'total',
avg(orderqty) as 'moyenne',
count(Orderqty) as 'nombre',
min(orderqty) as 'min',
max(Orderqty) as 'max'
FROM Sales.SalesOrderDetail
WHERE SalesOrderID IN(43659,43664)
group by salesorderId
order by SalesOrderID
```

Utilisation du group by : résultat

Mais nous pouvons aussi utiliser un partitionnement : Clause OVER

	SalesOrderID	ProductID	OrderQty	Total	Avg	Count	Min	Max
1	43659	776	1	26	2	12	1	6
2	43659	777	3	26	2	12	1	6
3	43659	778	1	26	2	12	1	6
4	43659	771	1	26	2	12	1	6
5	43659	772	1	26	2	12	1	6
6	43659	773	2	26	2	12	1	6
7	43659	774	1	26	2	12	1	6
8	43659	714	3	26	2	12	1	6
9	43659	716	1	26	2	12	1	6
10	43659	709	6	26	2	12	1	6
11	43659	712	2	26	2	12	1	6
12	43659	711	4	26	2	12	1	6

```
SELECT
    SalesOrderID,
    ProductID,
    OrderQty
    ,SUM(OrderQty) OVER(PARTITION BY SalesOrderID order by salesorderID) AS 'Total'
    ,AVG(OrderQty) OVER(PARTITION BY SalesOrderID) AS 'Avg'
    ,COUNT(OrderQty) OVER(PARTITION BY SalesOrderID) AS 'Count'
    ,MIN(OrderQty) OVER(PARTITION BY SalesOrderID) AS 'Min'
    ,MAX(OrderQty) OVER(PARTITION BY SalesOrderID) AS 'Max'
FROM Sales.SalesOrderDetail
WHERE SalesOrderID IN(43659,43664);
```

40

Les calculs / fonctions intégrées

Les fonctions d'agrégations
GROUP BY ou La clause OVER (partition order by)
Les clauses Row_number et rank

```
select ClientNom, ClientID
from T_Client
where ClientID like 'A%'
order by ClientNom asc

select
   ROW_NUMBER() OVER(ORDER BY clientnom ASC) AS Row#, clientNom, clientID
from T_Client
where ClientID like 'A%'
```

150 %

Résultats | Messages

	Row#	clientNom	clientID
1	1	Alfreds Futterkiste	ALFKI
2	2	Ana Trujillo Emparedados y helados	ANATR
3	3	Antonio Moreno Taquería	ANTON
4	4	Around the Hom	AROUT

41

Les calculs / fonctions intégrées

Les fonctions : Date (SQL Server)

```
select
CdeNum,
CdeDate,
GETDATE() as 'date jour serveur',
year(cdedate) as 'year',
month(cdedate) as 'mois',
Datepart(weekday, cdedate) as 'jour semaine',
Datepart(Year, cdedate) as 'année',
Datepart(QUARTER, cdedate) as 'trimestre',
DATEDIFF(day,cdedate,GETDATE()) as 'delai en jours',
DATEDIFF(Month,cdedate,GETDATE()) as 'delai en mois',
DATEDIFF(YEAR,cdedate,GETDATE()) as 'delai en année'
From T_Commande
```

CdeNum	CdeDate	date jour serveur	year	mois	jour semaine	année	trimestre	delai en jours	délai en mois	délai en année
10249	05/07/2004	2021-12-31 14:13:13.653	2004	7	1	2004	3	6388	209	17
10250	08/07/2004	2021-12-31 14:13:13.653	2004	7	4	2004	3	6385	209	17
10319	02/10/2004	2021-12-31 14:13:13.653	2004	10	6	2004	4	6299	206	17
10875	06/02/2006	2021-12-31 14:13:13.653	2006	2	1	2006	1	5807	190	15

42

Les calculs / fonctions intégrées

Les fonctions : Date (SQL Server) : Exemples

```
-- Que permettent de faire les requêtes suivantes ?
select top 100 Laminate_Id, Start_Time
From T_Commande
Where DATEDIFF(MINUTE, Start_Time,getdate())< 60

select CdeNum, DATEDIFF(day, CdeDate,
CommandeShippedDate) as délai
From T_Commande
where YEAR(cdedate) = 2006

select employenom
From T_Employe
where DATEDIFF(year, employedteentree, getdate()) > 15

-- La commande datediff permet de calculer un
délai/différence entre 2 dates.
-- Il faut préciser 3 éléments : L'intervalle (année,
mois, minutes, etc.), la date de début, la date de fin.
```

43

Les calculs / fonctions intégrées

Chaînes de caractères (SQL Server)

```
select
LEFT(ClientPays, 3) as '3ères car gch',
SUBSTRING(clientpays,3,2) as 'extrait du texte',
UPPER(clientpays) as 'en maj',
LOWER(clientpays) as 'en min',
LEN(clientpays) as 'nb caractères',
clientcontactfonction + clientid, -- espaces du champ de
type CHAR à droite…
Rtrim(clientcontactfonction) + ' ' + clientid as 'supp
                                    espaces à droite'
From T_Client
```

3ères car gch	extrait du texte	en maj	en min	nb caractères	(No column name)		supp espaces drte
DEU	UT	DEUTSCHLAND	deutschland	11	Sales Representative	ALFKI	Sales Representative ALFKI
Mex	xi	MEXICO	mexico	6	Owner	ANATR	Owner ANATR
Mex	xi	MEXICO	mexico	6	Owner	ANTON	Owner ANTON
UK		UK	uk	2	Sales Representative	AROUT	Sales Representative AROUT

44

Les calculs / fonctions intégrées

Convertir : La fonction CAST

```
select
cast(25.2 as int) + cast(15.2 as int) + cast(33.6 as int) as
result
convert(int, 25.2) = 25
→ 73: l'arrondi est calculé au nombre inférieur

-----------------------------------------
select getdate() as 'date entière',
convert (date, getdate()) as 'date sans heures minutes
secondes',
-- autres types de conversions mais en TEXTE attention :
convert(varchar(20),getdate(),108) 'date format 108',
convert(varchar(20),getdate(),107) 'date format 107',
convert(varchar(20),getdate(),113) 'date format 113',
convert(varchar(20),getdate(),103) 'date format 103'
```

date entière	date sans heures minutes secondes	date format 108	date format 107	date format 113	date format 103
2021-12-31 14:44:16.620	31/12/2021	14:44:16	déc 31, 2021	31/12/2021 14:44	31/12/2021

45

Langage SQL

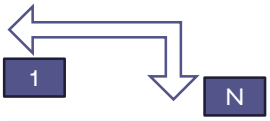
LES JOINTURES

46

Structure d'une table : les clés

- Une **clé primaire (Primary Key = PK)** est l'identifiant de l'enregistrement d'une table. Une **clé étrangère** (**FK**) est l'identifiant d'un enregistrement provenant d'une autre table.
- C'est la relation d'égalité entre les deux qui permet la relation entre la table client et la table commande.
- La cardinalité permet de comprendre la relation entre clé primaire et clé étrangère : 1- N signifiant une ligne dans table1(client – PK) peut correspondre à plusieurs lignes (max) dans table2 (commande – FK).

IdClient : clé primaire PK	Nom	Prénom	Ville
1	DUPONT	JOHN	PARIS
2	DUPONT	PAUL	LYON
3	BART	ELODIE	MARSEILLE



NumCommande	Idclient : clé étrangère FK	Date
12	1	30-déc-21
13	1	31-déc-21
14	2	31-déc-21

47

Les jointures

Rôle des clés

CLIENT

Idclt	Prénom
1	John
2	Paul
3	Elodie

COMMANDE

Id cmd	Date	Idclt	Idp
12	30-déc-21	1	2
13	31-déc-21	1	1
14	31-déc-21	2	3

PRODUIT

IDp	Produit
1	Vélo
2	Trotinette
3	Skate

PK (clés primaires) en rouge. *FK (clés étrangères) en vert.*

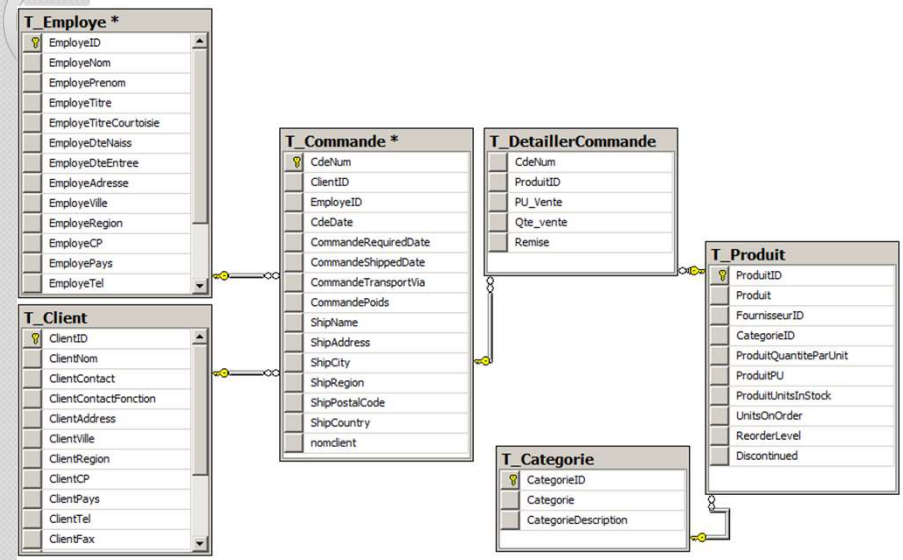
- L'égalité à retranscrire est entre la clé primaire et la clé étrangère.
- Si les deux champs contenant ses clés ont le même nom alors il faudra préciser le nom de la table afin que le code soit le plus précis possible sinon cela ne marchera pas.

Nomtable1.nomchampPK = Nomtable2.nomchampFK
CLIENT.idclt = COMMANDE.idclt

48

Les jointures

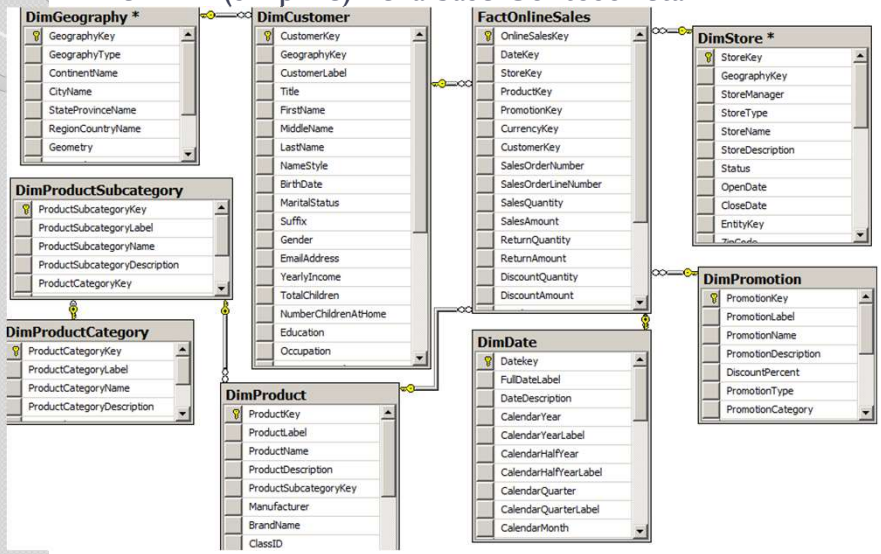
Lire un modèle relationnel (ou physique / MPD)



49

Les jointures

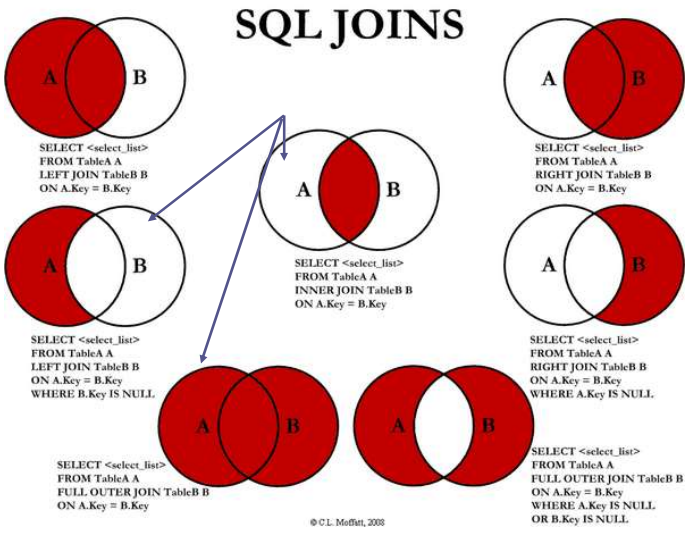
Le MPD (simplifié) de la base ContosoRetailDW



50

Les jointures

Les jointures Vue d'ensemble



51

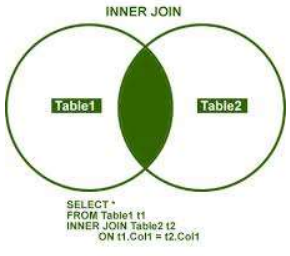
Les jointures

INNER JOIN (schéma) – Interne – à l'Intersection = JOIN

C'est **LA** Jointure **par défaut** qui compare deux tables et retourne tous les enregistrements comportant une **CORRESPONDANCE PARFAITE** (en règle **quasi générale**) PK vers FK.

Correspond à une RechercheV avec une correspondance. S'il ne trouve pas un élément qui est dans la table1 dans la seconde table, alors il ne le garde pas.

Si le client est dans la table client mais pas dans la table commande alors le JOIN va l'exclure.



52

Les jointures

INNER JOIN (schéma) – Interne = JOIN

Table client (A)

Idclt	Prénom
1	Samira
2	Caroline
3	Jordane
4	Pascal

Table commande (B)

Id cmd	Date	Idclt
1	27-févr	1
2	28-févr	1
3	28-févr	2

1, clé - clé primaire

N, Infini (8) - clé étrangère

```
select prénom, idcmd, date
FROM CLIENT JOIN COMMANDE
    ON CLIENT.Idclt = COMMANDE.Idclt
```

avec renommage de table

```
select prénom, idcmd, date
FROM CLIENT as A JOIN COMMANDE as B
    ON A.Idclt = B.Idclt
```

Les clients qui ont commandé

Prénom	Id cmd	Date
Samira	1	27-févr
Samira	2	28-févr
Caroline	3	28-févr

53

Les jointures

Renommer les tables (alias)

```
select D.CalendarYear, S.StoreName,
sum(IS.SalesAmount) as 'total montant ventes'

From FactInternetSales as IS
       JOIN DimProduct as P        – Appel 2 premières tables +
                                     type jointure
  ON IS.ProductKey = P.ProductKey – Quelle relation -> PK=FK

       JOIN DimPromotion as PR       - Appel nouvelle table
  ON IS.PromotionKey = PR.PromotionKey    - relation
       JOIN DimDate as D             - Appel nouvelle table
  ON IS.Datekey = D.DateKey          - relation
       JOIN DimStore as S            - Appel nouvelle table
  ON IS.StoreKey = S.StoreKey        - relation

where MONTH(D.datekey) = 3
Group by D.CalendarYear, S.StoreName
ORDER BY D.CalendarYear, S.StoreName
```

54

Les jointures

Points de vigilance : Distinct et ID/Nom

1. LE DISTINCT

À partir du moment où on fait une ou plusieurs jointures, on crée une table de correspondance énorme. Donc si un client a fait plusieurs commandes, il apparaîtra sur plusieurs lignes même si on affiche dans la requête finale, uniquement son nom.

→ Faire un test avec le **Distinct**. Si résultat différent, c'est qu'il est nécessaire car il y a des doublons.

1. SELECT SUR DES NOMS (personnes, produits, etc.)

Lorsque l'on affiche des noms de personnes, de produits, etc. on a tendance à mettre Select DISTINCT NOM et d'exécuter la requête telle quelle.
Le souci, c'est qu'on peut avoir plusieurs PIERRE DUPONT, non ?
On peut également avoir plusieurs produits qui ont le même nom mais pas les mêmes caractéristiques et qui sont donc différents ?

→ Pensez à toujours afficher l'ID afin d'éviter de supprimer des doublons qui n'en sont pas et d'avoir les bons effectifs finaux !

*Select distinct **IDProduit,** Nomproduit*
*Select distinct **Idclient,** Nomclient, Prenomclient*
Etc.

55

Les jointures

« Old School »

```
Select A.ClientNom, B.CdeDate, C.employenom

From t_client as A, T_commande B, T_employe C
-- les tables sont toutes appelées les unes après les
autres dans le From, séparées des virgules

where A.ClientID=B.ClientID And
B.EmployeID=C.EmployeID And
A.ClientPays = 'France'
ce qui est dans la clause ON, qui permet d'établir
l'égalité entre clés se trouve içi dans le WHERE
mélangé au reste
```

56

Les jointures

LEFT JOIN : Jointure externe

C'est **le second type de** Jointure **par défaut** qui compare deux tables et retourne tous les enregistrements comportant une **CORRESPONDANCE PARFAITE ou non.**

Correspond à une RechercheV Excel avec les correspondances parfaites et les NA. Il garde tous les éléments de la table1 même s'il ne les retrouve pas dans l'autre table.

Préciser dans quelle table, sont tous les éléments : Left si la table est à gauche du JOIN dans le code, Right, si elle se situe à droite.

Cela permet de récupérer **tous les clients :**

- S'ils ont fait une commande, on pourra récupérer information de la table commande donc le contenu des colonnes de l'autre table.

- S'ils n'ont jamais fait de commande, il apparaitra des NULL dans toutes les colonnes de l'autre table afficher via le SELECT. (correspondant aux NA du rechercheV sur Excel).

57

Les jointures

LEFT JOIN : Jointure externe avec Condition NULL

C'est **le troisième type de** Jointure **par défaut** qui compare deux tables et retourne tous les enregistrements comportant **AUCUNE CORRESPONDANCE** entre deux tables**.**

Correspond à une RechercheV Excel avec seulement des NA en guise de résultats. Il garde que les éléments de la table1 qui ne sont pas dans l'autre table.

Cela permet de **récupérer les clients qui n'ont pas fait de commande** :

- On précise toujours que l'on souhaite tous les clients (table1) avec LEFT ou RIGHT,

- S'ils n'ont jamais fait de commande, il apparaitra des NULL dans toutes les colonnes de l'autre table donc on va utiliser une **condition sur la valeur NULL dans une colonne de l'autre table**.

Mettre la condition NULL sur une colonne **obligatoire** afin d'être sûr d'avoir des non-correspondance donc **clés primaires ou clés étrangères**.

58

Les jointures

LEFT JOIN : Jointure externe (exemple)

Table client (A)

Idclt	Prénom
1	Samira
2	Caroline
3	Jordane
4	Pascal

Table commande (B)

Id cmd	Date cmd	Idclt	Date livraison
1	27-févr	1	03-févr
2	28-févr	1	06-févr
3	28-févr	2	null

1, clé - clé primaire

N, Infini (8) - clé étrangère

```
select prénom, idcmd, date
FROM CLIENT LEFT JOIN COMMANDE
ou FROM COMMANDE RIGHT JOIN CLIENT
   ON CLIENT.Idclt = COMMANDE.Idclt
```

```
select prénom, idcmd, date
FROM CLIENT as A LEFT JOIN COMMANDE as B
ou FROM COMMANDE as B RIGHT JOIN CLIENT as A
   ON A.Idclt = B.Idclt
```

Tous les clients (avec ou sans commande)

Prénom	Id cmd	Date cmd	Date livraison
Samira	1	27-févr	03-févr
Samira	2	28-févr	06-févr
Caroline	3	28-févr	null
Jordane	null	null	null
Pascal	null	null	null

S'il n'y a pas de correspondance : toutes les colonnes sont vides car le client n'a jamais commandé.

Si je souhaite **QUE LES CLIENTS SANS COMMANDE** : Condition (where) sur une colonne de la table commande.
Eviter les colonnes contenant possiblement des Null. Prendre des clés (primaires ou étrangères).

59

Les jointures

LEFT JOIN : Exemple

```
Select

A.SOHNUM as 'numéro de commande sur table commande',

B.SOHNUM as 'numéro de commande sur table livraison'


From SORDER A LEFT JOIN DELIVERYD B
        ON A.SOHNUM = B.SOHNUM

WHERE B.SOHNUM IS NULL
```

SELECT : J'affiche des champs provenant des deux tables.

FROM : Je prends toutes les lignes de la table A (ordres de commandes) : LEFT indiquant de prendre ce qui est à gauche du JOIN.

WHERE : Quand le champs SOHNUM est inexistant (NULL) dans la table b (livraisons)

→ Les commandes qui n'ont pas été livrées !

60

Les jointures
Synthèse et Méthodologie (1/2)

1. Quelles sont les tables dont j'ai besoin en fonction de la demande ?

FROM : Je notifie chaque table que je dois appeler dans la clause From.

2. Quelle(s) est(sont) la(es) colonne(s) de correspondance ?

FROM : En utilisant le schéma de la base de de données (MPD : modèle physique de données), je regarde la(es) relation(s) entre les tables que je dois appeler pour écrire ma requête. La clé ou 1 d'un côté signifie qu'il y a la clé primaire du côté de la table en question et le signe de l'infini ou N signifie qu'il a la clé étrangère du côté de l'autre table. Cela me permet de récupérer ma colonne de correspondance pour relier deux tables entre elles.

61

Les jointures
Synthèse et Méthodologie (2/2)

3. Les autres clause ?

Je gère mes clauses Select, Where, group by, Having et Order by comme vu précédemment. Bien préciser le nom de la table avant le nom du champ si ce dernier existe dans plusieurs tables différentes utilisées.

5. Est-ce que je veux exclure ou inclure des données d'une table par rapport à une autre ?

Est-ce une jointure d'inclusion ? → Je laisse le JOIN

Est-ce une jointure d'exclusion ? → LEFT JOIN ou RIGHT JOIN selon l'ordre d'écriture des tables.

Ne pas oublier ensuite la condition NULL (dans la table où l'on ne souhaite pas retrouver les données.

62

La commande SELECT :
SELECT UNION / ALL / INTERSECT / EXCEPT

```
select clientid from t_client
                UNION
Select clientid from t_commande
```

UNION → permet de récupérer les lignes des deux requêtes, tout en supprimant les doublons si des éléments (client) apparaissent dans les deux tables.

UNION ALL → permet de faire exactement la même chose, tout en conservant les doublons.

INTERSECT → permet de récupérer que les éléments (client) communs qui sont dans la première mais aussi dans la seconde table. Si le clientid n'existe que dans la table t_client, alors la commande ne permet pas de l'afficher.

EXCEPT → permet de récupérer que les éléments (client) qui sont dans la requête qui précède EXCEPT (celle du haut) mais qui ne sont pas dans la suivante (commande).

63

Langage SQL
LES SOUS REQUÊTES

64

Utilisation de sous-requêtes

- Écriture de sous-requêtes simples
- Écriture de sous-requêtes corrélées
- Utilisation du prédicat Exists avec les sous-requêtes
- Utilisation du IN, ALL, ANY, SOME

Sous requêtes :

Dans la clause WHERE : In ou NOT IN (requête simples)
L'inclusion / l'exclusion ET/OU comparer des listings

```
-- Comparaison de listing d'une table sur l'autre donc
possibilité d'écrire une sous-requête dans le where :
Select clientnom
from t_client
where ClientID IN (SELECT distinct ClientID from T_Commande)

-- En version Jointure :
SELECT distinct ClientNom
From T_client JOIN t_commande ON
T_client.clientID=T_commande.clientID
```

-- Seulement dans le cas d'une comparaison entre une table et une autre où il n'y a pas besoin de jointure (et donc peut être remplacé par une jointure) MAIS POUR TOUS LES AUTRES CAS, ce sera obligatoirement une SOUS-REQUETE et possiblement des jointures à l'intérieur ☺

Les performances : De quel côté sont-elles favorables ?

Sous requêtes :

Dans la clause WHERE/Having : = > < et ALL / ANY

```
-- Afficher le nombre de commandes en 2006 que si plus
importante qu'en 2004

-- nombre de commandes en 2006 – 600 commandes
Select COUNT(cdenum) as 'nombre de commandes'
from T_Commande
where YEAR(cdedate) = 2006
having COUNT(cdenum) >

-- nombre de commandes en 2004 – 550 commandes
(Select COUNT(cdenum) as 'nombre de commandes'
from T_Commande
where YEAR(cdedate) = 2004)
```

- On peut rajouter ALL : signifie si supérieur à toutes les valeurs de la sous-requête.
- Ou ANY = signifie supérieur à au moins une seule valeur.

67

Sous requêtes :

La sous-requête corrélée

-- Faire remonter les pays qui avaient plus de commandes en 2006 qu'en 2004 :

```
Select shipcountry,
COUNT(cdenum) as 'nombre de commandes'
from T_Commande
where YEAR(cdedate) = 2006
group by shipcountry
having COUNT(cdenum) >

-- nombre de commandes en 2004
(Select COUNT(cdenum) as 'nombre de commandes'
from T_Commande cmd – Renommer table pour être précis
 where YEAR(cdedate) = 2004
AND cmd.ShipCountry=T_Commande.ShipCountry – Faire une
jointure entre le country de la sous-requête (cmd) et
celui de la requête principale (t_commande)
group by shipcountry)
```

68

Sous requêtes :

Dans la clause WHERE : EXISTS (requêtes corrélées)

```
SELECT reference_art
  FROM ARTICLES
  WHERE NOT EXISTS(SELECT *
    FROM LIGNES_CDE
    WHERE LIGNES_CDE.reference_art= ARTICLES.reference_art);
```

3 différences avec les requêtes simples :

-Dans la condition where et le prédicat Exists : aucun champ car on va lier les requêtes par les tables entre la 1ère requête et la seconde : requêtes corrélées,

- Dans la sous-requête, on va utiliser l'ancienne syntaxe des jointures : tables énumérées dans le From et clés reliées dans le where,

- Dans le where de la sous-requête : on lie la clé de la table de la sous-requête à la clé de la table de la première requête (corrélation des requêtes)

--> Permet d'avoir plusieurs choses dans le select de la sous-requête

69

Sous requêtes :

Dans la clause FROM

```
select AVG([nombre de produits]) as 'moyenne'
From

(
Select COUNT(productkey) as 'nombre de produits'
From DimProduct
Group by ColorName
) as toto
```

- Il faut **décomposer le calcul** (étape 1, sous-requête Puis étape 2/ finale : requête principale.

- Pour que **la sous-requête soit considérée comme une table**, cela nécessite :
  - 1: des parenthèses,
  - 2: un nom de table,
  - 3: des noms/entêtes de colonnes

70

Sous requêtes :

Dans le SELECT

```
select
DC.CustomerKey,
DC.LastName,
avg(salesamount) as 'moyenne client',

(select avg(salesamount) from FactOnlineSales)  as 'moy',

(select avg(salesamount) from FactSales)  as
'moy_gnrle_VentesMagasin'

from FactOnlineSales FOS join DimCustomer DC
        on FOS.CustomerKey=DC.CustomerKey
group by DC.CustomerKey, DC.LastName
```


Les sous-requêtes

Synthèse et Méthodologie

1.Est-ce que je souhaite comparer des listings ? Exclure des données de l'un par rapport à l'autre ?

Je vais devoir créer une requête imbriquée dans le WHERE.

Attention : est-ce que je peux remplacer cette requête par une jointure ?

2. Est-ce que j'ai besoin de préparer des requêtes intermédiaires afin d'effectuer des requêtes dessus (fonction d'agrégation sur une autre fonction d'agrégation ou tout autres traitements successifs) ?

Je vais devoir créer une requête imbriquée dans le FROM afin de créer une table temporaire / table dérivée

3. Est-ce que je souhaite récupérer une valeur dans mon select qui soit le résultat d'une autre requête ?

Je vais devoir créer une requête imbriquée dans le SELECT afin de créer un nouveau champ.

En général :

Règles à ne pas oublier !!!

- Toujours préciser les colonnes retourner par la recherche (le « select * » est très gourmand),
- Ne joindre que les tables nécessaires,
- Ne pas oublier le DISTINCT, souvent nécessaire avec les jointures,
- Pour augmenter la lisibilité des requêtes, renommer vos tables (alias) et colonnes,
- Documenter la requête avant de l'écrire :
  - But
  - Auteur
  - Date

ANNEXES

Sommaire / Plan détaillé du support de formation

Schémas des bases de données

Liens web utiles

Sommaire / Plan détaillé (1/2)

Sommaire / Plan détaillé (2/2)

La commande SELECT :

L'ordre de sa syntaxe (1/2)

1.SELECT sum(ventes) as 'Montant des ventes' → AFFICHAGE / RESULTAT
, year(cdedate) as 'Année, service (Virgules)

2.FROM table1 JOIN table2 → PROVENANCE / DANS QUELLE TABLE ?
(JOIN/ON et/ou virgules)

On table1.PK=table2.FK
JOIN table3
On table2.Pk=Table3.FK

3.WHERE pays = 'France' and Age= 20 → CONDITION ? Où ? FILTRE SUR CHAMP DANS LA BASE DE DONNEES
(AND et/ou OR)

77

La commande SELECT :

L'ordre de sa syntaxe (2/2)

4.GROUP BY year(cdedate), service → REGROUPER PAR
(tout ce qui est ici est à copier-coller dans le Select) (Virgules)

5.HAVING sum(ventes) > 50000 → CONDITION ? Où ? FILTRE SUR une fonction d'agrégation dont le résultat est visible après avoir fait le GROUP BY
(AND et/ou OR)

6.ORDER By CHAMP1 asc/desc → TRIER PAR (Virgules)

78

La commande SELECT :

Quand faut-il mettre des parenthèses ?

Les parenthèses sont **obligatoires dans 3 cas** :

- Lors de l'utilisation des **fonctions intégrées** (en rose dans SQL Server) : Year(champ), count(champ), convert(champ), etc.

- Lors de l'utilisation du **IN / NOT IN** → signifie que l'on souhaite mettre une condition correspondant au listing indiqué à l'intérieur :

Where année_naissance IN (1985, 1988) = *année est égale à 1985 ou 1988*
Where ville NOT IN ('Lyon','Lille') = ville n'est pas Lyon ou Lille

- Lors de l'utilisation de **OR et AND cumulatifs** dans une requête :

Where age = 50 and **(**ville like '%p' or ville like '%f'**)** → afin qu'il ne se mélange pas les pinceaux entre AND et OR

- Lors de l'utilisation **d'une sous requête** : une requête imbriquée dans une autre requête sera toujours entourée de parenthèses.

79

La commande SELECT :

Quand faut-il mettre des cotes ?

Les cotes sont **obligatoires dans 2 cas** :

- **Lors de la création d'un alias de colonne,** si ce dernier contient plusieurs mots :

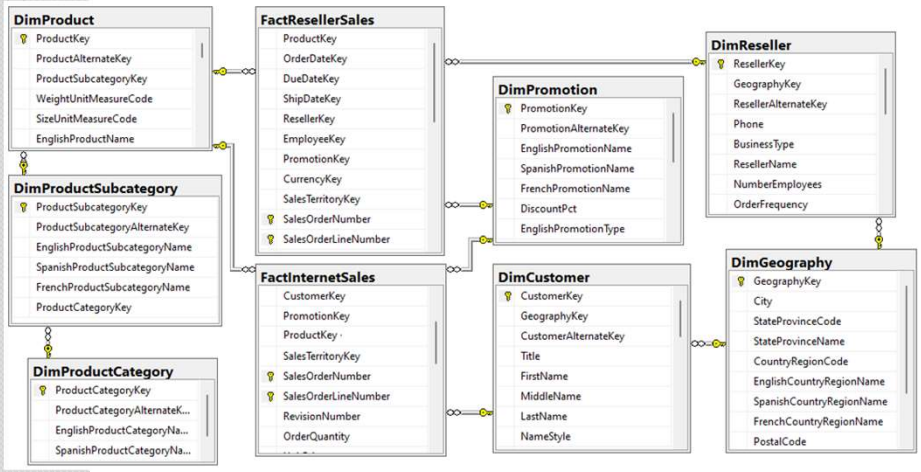
✓ Select ZERTIKL as 'Nom du client' → si j'ai des espaces, alors je dois mettre des cotes.
✓ Select ZERTIKL as 'Nom' → s'il n'y a pas d'espaces, les cotes ne sont pas obligatoires mais peuvent être utilisées.

- **Lors de la création d'une condition sur une valeur,** si le type de donnée de cette dernière n'est pas du numérique :

✓ Where ZERTIKL = 'DUPONT'
✓ Where Age = '50' → la valeur est du numérique, les cotes ne sont pas obligatoires mais peuvent être utilisées.

80

Schémas des bases de données – AdventureWorksDW2019



81

Schémas des bases de données – AdventureWorksDW2019

Relations :

Gauche du schéma :

Table DimproductCategory vers DimproductSubCategory : productcategorykey
Table DimproductSubCategory vers Dimproduct : productsubcategorykey
Table Dimproduct vers FactResellerSales et FactInternetSales : productkey

Milieu :

Tables FactResellerSales et FactInternetSales vers Dimpromotion : promotionkey

Droite :

Table FactResellerSales vers DimReseller : resellerkey
Table Dimreseller vers Dimgeography : geographykey

Table FactInternetSales vers DimCustomer : customerkey
Table DimCustomer vers Dimgeography : geographykey

82

Liens web utiles

Le Web regorge de sites traitant (parfois de façon excellente) du SQL.

Comme il n'est pas possible de les citer tous, en voici une liste tout à fait arbitraire mais représentative de ce qu'on trouve sur le web :

- http://sql.sh/
- https://openclassrooms.com/courses/apprenez-a-programmer-en-vb-net/introduction-au-langage-sql
- http://sql.dev
- https://stackoverflow.com/
- https://technet.microsoft.com/fr-fr/library/ms190750(v=sql.105).aspxeloppez.com/#apprendre-sql

MERCI POUR VOTRE ATTENTION

J'espère que vous repartez confiant et serein autour du langage SQL

Bonne continuation