

SQL

Opération Sur Une Classe

Modélisation des SI
Année: 2004-2005

T.Le, M. Léonard
{thang/leonard}@cui.unige.ch
SYSINF-SES-CUI
UNIVERSITE DE GENEVE



MATIS Geneva Team
Database Laboratory

Contenu

- **Introduction aux requêtes**
- Expressions, conditions et opérateurs
- Fonctions
- Clauses dans le SQL
- Requêtes regroupées
- Sous-requêtes
- Créer une table



Schéma relationnel

CHAUFFEUR (**NoAVS**//Nom,Prénom, Adresse,DateNaissance,Sexe,TotalHeures)

PERMIS (**NoAVS**,Catégorie//DatePermis)

VÉHICULE (**NoVéhicule** //
Catégorie,Marque,NoChassis,NoMoteur,DateCirculation)

LIGNE (**CodeLigne**//Catégorie,TypeLigne,ArretDepart,ArretArrivee)

AFF-VÉHICULE (**NoVéhicule**,DateAff //
StatutVéhicule,CodeLigne,StatutAffectation)

AFF-CHAUFFEUR (**NoAVS**, DateAff // StatutChauffeur,NoSérie)

TRANCHE-SÉRIE (**NoSérie**,CodeLigne//HeureDébut,HeureFin)

HEURE-CHAUFFEUR (**NoVéhicule**,DateAff,**NoAVS**//NbHeures)

3



Définition d'une requête

SELECT att₁, att₂, ..., att_n —————> le format du résultat

FROM classe₁, classe₂, .., classe_m —————> À partir de quelles classes

WHERE condition —————> Les prédicats de sélection



att_i: un nom d'attribut (colonne)

classe_j: un nom de classe (table)

4



Définition d'une requête

Chauffeur

NoAVS	Nom	Prénom	Date Naissance	Adresse	Sexe	Total Heures
000111222001	Lee	Kim	18/04/67	24 General Dufour	M	400
000111222002	Lee	Loulou	15/12/72	Uni Mail CH 1204	F	200
000111222003	Meyer	Boubou	23/12/82	Gare Cornavin	M	300
000111222004	Lambert	Nounou	01/01/77	Chene-Bourg	M	100

Requête: toutes les colonnes et tous les tuples ?

SQL: SELECT *
FROM Chauffeur



L'astérisque (*): sélectionner toutes les colonnes de la table sollicitée

5



Définition d'une requête

Chauffeur

NoAVS	Nom	Prénom	Date Naissance	Adresse	Sexe	Total Heures
000111222001	Lee	Kim	18/04/67	24 General Dufour	M	400
000111222002	Lee	Loulou	15/12/72	Uni Mail CH 1204	F	200
000111222003	Meyer	Boubou	23/12/82	Gare Cornavin	M	300
000111222004	Lambert	Nounou	01/01/77	Chene-Bourg	M	100

Requête: un tuple ?

SQL: SELECT *
FROM Chauffeur
WHERE NoAVS = '000111222002'

6

Définition d'une requête

Chauffeur

NoAVS	Nom	Prénom	Date Naissance	Adresse	Sexe	Total Heures
000111222001	Lee	Kim	18/04/67	24 General Dufour	M	400
000111222002	Lee	Loulou	15/12/72	Uni Mail CH 1204	F	200
000111222003	Meyer	Boubou	23/12/82	Gare Cornavin	M	300
000111222004	Lambert	Nounou	01/01/77	Chene-Bourg	M	100

Requête: une colonne ?

SQL: SELECT Nom
FROM Chauffeur

Lee
Lee
Meyer
Lambert

SELECT DISTINCT Nom
FROM Chauffeur

Lee
Meyer
Lambert



L'option **DISTINCT** élimine les lignes dupliquées

7

Définition d'une requête

Chauffeur

NoAVS	Nom	Prénom	Date Naissance	Adresse	Sexe	Total Heures
000111222001	Lee	Kim	18/04/67	24 General Dufour	M	400
000111222002	Lee	Loulou	15/12/72	Uni Mail CH 1204	F	200
000111222003	Meyer	Boubou	23/12/82	Gare Cornavin	M	300
000111222004	Lambert	Nounou	01/01/77	Chene-Bourg	M	100

Requête: plusieurs colonnes ?

SQL: SELECT Nom, Prénom, Adresse
FROM Chauffeur

8

Définition d'une requête

Chauffeur

NoAVS	Nom	Prénom	Date Naissance	Adresse	Sexe	Total Heures
000111222001	Lee	Kim	18/04/67	24 General Dufour	M	400
000111222002	Lee	Loulou	15/12/72	Uni Mail CH 1204	F	200
000111222003	Meyer	Boubou	23/12/82	Gare Cornavin	M	300
000111222004	Lambert	Nounou	01/01/77	Chene-Bourg	M	100

Requête: plusieurs tuples ?

SQL: **SELECT ***
FROM Chauffeur
WHERE Nom = 'Lee'

9

Contenu

- Introduction aux requêtes
- **Expressions, conditions et opérateurs**
- Fonctions
- Clauses dans le SQL
- Requêtes regroupées
- Sous-requêtes
- Créer une table

10

Expressions

- Une expression renvoie une valeur

– Exemple:

```
SELECT noAVS, nom, prenom FROM chauffeur
WHERE nom = 'Meyer';
```

Expression:

noAVS, nom, prenom

→ valeur : chaîne de caractères

nom = 'Meyer'

→ valeur: Vrai ou Faux

Row #	NOAVS	NOM	PRENOM	DATENAISANCE	ADRESSE	SEXE	TOTALHEURES
1	000111222001	Lee	Kim	18-Apr-1967	24 General Dufour	M	400
2	000111222002	Le	Loulou	15-Dec-1972	Uni Mail CH 1204	F	200
3	000111222003	Meyer	Boubou	23-Dec-1982	Gare Cornavin	M	300
4	000111222004	Lambert	Nounou	1-Jan-1977	Chene-Bourge	M	100

NOAVS	NOM	PRENOM
000111222003	Meyer	Boubou

11

Conditions

- Dans le cas d'interrogation d'une classe
→ projection et sélection

- Si vous voulez trouver un élément ou un groupe particulier d'éléments dans votre tables, vous avez besoin d'une ou plusieurs conditions

- SQL:

– les conditions sont contenues dans la clause *WHERE*

– Syntaxe:

WHERE <Condition de recherche>

12



Conditions

- Condition avec une chaîne de caractères:

WHERE nom = 'Boubou'

Attention: PAS nom = "BouBou"

- Condition avec un entier:

WHERE totalheurs > 100

- Condition avec une date

WHERE datenaissance > '01-Dec-70'

13



Opérateurs

- *Il y a six groupes d'opérateurs:*

- arithmétique,
- comparaison,
- caractère,
- logique et
- divers.

14

Opérateurs d'arithmétiques

- Les opérateurs d'arithmétiques sont **+** (addition), **-** (soustraction), ***** (multiplication) et **/** (division)

- Exemple:

SELECT totalheures/2 **FROM** chauffeur ;

SELECT totalheures, totalheures/2 **AS** demitotal **FROM** chauffeur ;

SELECT (((totalheures*2) - 4)/2) + 2 **AS** total **FROM** chauffeur ;

SELECT nom, prenom, totalheures/2 **FROM** chauffeur ;

SELECT nom, prenom, totalheures, totalheures/2 **AS** demitotal **FROM** chauffeur ;

SELECT nom, prenom, (((totalheures*2) - 4)/2) + 2 **AS** total **FROM** chauffeur ;

NOM	PRENOM	TOTALHEURES/2
Lee	Kim	200
Le	Loulou	100
Meyer	Boubou	150
Lambert	Nounou	50

NOM	PRENOM	TOTALHEURES	DEMITOTAL
Lee	Kim	400	200
Le	Loulou	200	100
Meyer	Boubou	300	150
Lambert	Nounou	100	50

NOM	PRENOM	TOTAL
Lee	Kim	400
Le	Loulou	200
Meyer	Boubou	300
Lambert	Nounou	100



Employez **AS** pour fournir des noms de colonnes

15

Opérateurs de comparaison

- Les opérateurs de comparaison retournent une des trois valeurs:

VRAI, **FAUX**, ou **inconnu**

- Exemple:

SELECT NoAVS, nom, prenom **FROM** chauffeur **WHERE** prenom = 'boubou' ;

SELECT NoAVS, nom, prenom **FROM** chauffeur **WHERE** prenom = 'Boubou' ;

SELECT NoAVS, nom, prenom **FROM** chauffeur **WHERE** prenom = 'BOUBOU' ;

SELECT NoAVS, nom, prenom **FROM** chauffeur **WHERE** prenom IS NULL ;

Row #	NOAVS	NOM	PRENOM
1	000111222001	Lee	Kim
2	000111222002	Le	Loulou
3	000111222003	Meyer	Boubou
4	000111222004	Lambert	Nounou

```
no rows selected
```

NOAVS	NOM	PRENOM
000111222003	Meyer	Boubou

```
no rows selected
```

```
no rows selected
```



La syntaxe de SQL n'est pas cas sensible mais les données sont cas sensible

16

Opérateurs de caractère

Exemple:

SELECT nom, prenom **FROM** chauffeur

WHERE nom **LIKE** 'L%';

SELECT nom, prenom **FROM** chauffeur

WHERE **RTRIM**(nom) **LIKE** 'L_';

SELECT nom || prenom **FROM** chauffeur ;

NOM	PRENOM
Lee	Kim
Le	Loulou
Lambert	Nounou

NOM	PRENOM
Le	Loulou

NOM PRENOM
Lee Kim
Le Loulou
Meyer Boubou
Lambert Nounou



Les caractères génériques (wild-card):

- Les % substituent un groupe de caractères
- Les _ substituent un caractères

Concaténation:

- Le symbole de || (double pipe) enchaîne deux chaînes de caractères

RTRIM : Éliminer des caractères à la droite de la chaîne spécifiée

17

Opérateurs logiques

Exemple:

SELECT * **FROM** chauffeur

WHERE nom **LIKE** 'L%' **AND** totalheures > 100;

SELECT * **FROM** chauffeur

WHERE nom **LIKE** 'L%' **OR NOT** totalheures > 100;

• Une approche plus claire est d'écrire:

SELECT * **FROM** chauffeur

WHERE (nom **LIKE** 'L%') **AND** (totalheures > 100);

SELECT * **FROM** chauffeur

WHERE (nom **LIKE** 'L%') **OR** (**NOT** (totalheures > 100));

18



Divers Opérateurs

■ Les divers opérateurs: **IN**

■ Exemple:

```
SELECT * FROM chauffeur  
WHERE (nom='Boubou') OR (nom='Loulou');
```

- Une approche plus claire est d'écrire :

```
SELECT * FROM chauffeur  
WHERE nom IN ('Boubou','Loulou');
```



IN: comparer une valeur à une liste de valeurs littérales spécifiées

19



Divers Opérateurs

■ Les divers opérateurs: **BETWEEN**

■ Exemple:

```
SELECT * FROM chauffeur  
WHERE (totalheures >= 100) AND (totalheures <=1000);
```

- Une approche plus claire est d'écrire :

```
SELECT * FROM chauffeur  
WHERE totalheures BETWEEN 100 AND 1000;
```



BETWEEN: rechercher des valeurs situées à l'intérieur d'un intervalle (on connaît les valeurs minimale et maximale)

20



Contenu

- Introduction aux requêtes
- Expressions, conditions et opérateurs
- **Fonctions**
- Clauses dans le SQL
- Requêtes regroupées
- Sous-requêtes
- Créer une table

21



Fonctions

- Fonctions d'agrégation
- Fonctions de date et d'heure
- Fonctions de caractère
- Fonctions de conversion
- Diverses fonctions

22

Fonctions d'agrégation

Exemple:

SELECT COUNT(*) FROM chauffeur ;

SELECT COUNT(nom) FROM chauffeur ;

**SELECT AVG(totalheures),
MIN(totalheures), MAX(totalheures)
FROM chauffeur ;**

**SELECT SUM(totalheures)/24 AS totaljours
FROM chauffeur;**

**SELECT SUM(totalheures)/(24*30) AS
totalmois, SUM(totalheures)/24 AS
totaljours FROM chauffeur;**

COUNT(*)			

4			
COUNT(NOM)			

4			
AVG(TOTALHEURES)	MIN(TOTALHEURES)	MAX(TOTALHEURES)	
-----	-----	-----	
250	100	400	
TOTALJOURS			

41.6666667			
TOTALMOIS	TOTALJOURS		
-----	-----		
1.38888889	41.6666667		



COUNT: compter les lignes ou les valeurs d'une colonne ne contenant pas de valeur NULL

SUM: retourner le total des valeurs d'une colonne

AVG: calculer la moyenne des valeurs d'une colonne

MAX; MIN: retourner la valeur maximale / minimale d'une colonne

23

Fonctions de date et d'heure

Exemple:

SELECT SYSDATE FROM DUAL;

SELECT DISTINCT SYSDATE FROM affchauffeur ;

SELECT SYSDATE+1 AS demain FROM DUAL;

**SELECT affdate - 1 AS avant_affdate FROM
affchauffeur WHERE NoAVS = '000111222002';**

SYSDATE	

28-NOV-02	
SYSDATE	

28-NOV-02	
DEMAIN	

29-NOV-02	
AVANT_AFF	

01-JAN-00	



- **SYSDATE** renvoie la date de système

- **DUAL** : cette table est la table spéciale du dictionnaire de données d'Oracle qui a seulement une colonne et une ligne

24

Fonctions de caractère

Exemple:

SELECT UPPER(nom), UPPER(prenom)
FROM chauffeur;

SELECT nom, prenom FROM chauffeur
WHERE LOWER(prenom)='boubou';

SELECT nom, prenom FROM chauffeur
WHERE SUBSTR(nom,1,1)='L';

UPPER (NOM)	UPPER (PRENOM)
LEE	KIM
LE	LOULOU
MEYER	BOUBOU
LAMBERT	NOUNOU

NOM	PRENOM
Meyer	Boubou

NOM	PRENOM
Lee	Kim
Le	Loulou
Lambert	Nounou



- **LOWER** change tous les caractères à la lettre minuscule
- **UPPER** change tous les caractères à la lettre majuscule
- **SUBSTR(chaine_de_caractères, position, nombre_de_caractères)**
 - Le premier argument est la chaîne de caractères à traiter.
 - Le deuxième argument est la position du premier caractère à sortir.
 - Le troisième argument est le nombre de caractères à montrer

25

Fonctions de conversion

Exemple:

SELECT RTRIM(nom) || '-' || TO_CHAR(NoAVS)
as nom_et_noAVS FROM chauffeur;

SELECT NoAVS, nom FROM chauffeur WHERE
datenaissance = TO_DATE ('December 23,
1982', 'MONTH DD, YYYY');

SELECT
TO_CHAR (sysdate, 'DD-MM-YYYY') AS FRENCH,
TO_CHAR (sysdate, 'MM-DD-YYYY') AS
AMERICAN FROM DUAL;

NOM_ET_NOAVS
Lee-000111222001
Le-000111222002
Meyer-000111222003
Lambert-000111222004

NOAVS	NOM
000111222003	Meyer

FRENCH	AMERICAN
28-11-2002	11-28-2002



- **TO_CHAR** convertit un entier / une date en caractère
- **TO_NUMBER** convertit une chaîne de caractères en entier
- **TO_DATE** convertit une chaîne de caractères en date

26



Contenu

- Introduction aux requêtes
- Expressions, conditions et opérateurs
- Fonctions
- **Clauses dans le SQL**
- Requêtes regroupées
- Sous-requêtes
- Créer une table

27



ORDER BY

- **ORDER BY:** Cette clause triera vos résultats

● Exemple:

**SELECT nom, prenom FROM chauffeur
ORDER BY nom;**

NOM	PRENOM
Lambert	Mounou
Lee	Kim
Lee	Loulou
Meyer	Boubou

**SELECT nom, totalheures FROM
chauffeur ORDER BY totalheures
DESC;**

NOM	TOTALHEURES
Lee	400
Meyer	300
Lee	200
Lambert	100

**SELECT nom, datenaissance FROM
chauffeur
ORDER BY nom ASC, datenaissance;**

NOM	DATENAISS
Lambert	01-JAN-77
Lee	18-APR-67
Lee	15-DEC-72
Meyer	23-DEC-82



ASC (défaut) : l'ordre croissant
DESC : l'ordre décroissant

28



Contenu

- Introduction aux requêtes
- Expressions, conditions et opérateurs
- Fonctions
- Clauses dans le SQL
- **Requêtes regroupées**
- Sous-requêtes
- Créer une table

29



SCHEMA RELATIONNEL

PRODUIT (NumProd // LibelléProd, PrixUnitaireProd)
CLIENT (NumClient // Nom, Prénom, AdresseClient)
COMMANDE (NumCde // NumClient, DateCde, Total, AdresseLivraison)
LIGNECDE (NumCde, NumProd // QuantiteProdCde)
LIVRCDE (NumLiv, NumCde // DateLivr, EtatLivr)
LIVRCDEPRODUIT (NumLiv, NumCde, NumProd // QteLivrée)
RENOI (NumLiv, NumCde // DateRenvoi)

30

Requêtes regroupées

- **Le regroupement de données** est un processus pour combiner dans un ordre logique des lignes de données *dont une ou plusieurs colonnes prennent des valeurs identiques*
- Le clause **GROUP BY** s'emploie pour classer des données identiques par groupe

31

Requêtes regroupées

- Quel est le montant moyen des commandes?
- **SQL> SELECT AVG(Total) FROM Commande**

NUMCDE	NUMCLIENT	DATECDE	TOTAL	ADRESSELIVRAISON
10	1	15-Dec-2099	32500	GÉNÉRAL DUFOUR, 24
20	2	20-Jan-2000	24500	RUE DE LYON, 03
30	1	18-Jun-2000	2500	MIREMONT, 29
40	3	20-Sep-2000	6000	RUE DE LAUSANNE, 01
50	1	30-Dec-2000	1600	GÉNÉRAL DUFOUR, 24
60	2	7-Jan-2001	3100	RUE DE GENÈVE, 120

→

AVG(TOTAL)
11700

32

Requêtes regroupées

- Quel est le montant moyen des commandes **pour chaque client** ?
- **SQL> SELECT AVG(Total) FROM Commande
GROUP BY NumClient**

→ **GROUP BY**

NUMCDE	NUMCLIENT	DATECDE	TOTAL	ADRESSELIVRAISON
10	1	15-Dec-2099	32500	GÉNÉRAL DUFOUR, 24
30	1	18-Jun-2000	2500	MIREMONT, 29
50	1	30-Dec-2000	1600	GÉNÉRAL DUFOUR, 24
20	2	20-Jan-2000	24500	RUE DE LYON, 03
60	2	7-Jan-2001	3100	RUE DE GENÈVE, 120
40	3	20-Sep-2000	6000	RUE DE LAUSANNE, 01

→ **AVG(...)**

NUMCLIENT	AVG(TOTAL)
1	12200
2	13800
3	6000

33

Requêtes regroupées

- Quel est le montant moyen des commandes pour chaque client, **pour lesquels le montant moyen est supérieur à 10000** ?
- **SQL> SELECT AVG(Total) FROM Commande
GROUP BY NumClient HAVING AVG(Total) > 10000**

→ **GROUP BY**

NUMCDE	NUMCLIENT	DATECDE	TOTAL	ADRESSELIVRAISON
10	1	15-Dec-2099	32500	GÉNÉRAL DUFOUR, 24
30	1	18-Jun-2000	2500	MIREMONT, 29
50	1	30-Dec-2000	1600	GÉNÉRAL DUFOUR, 24
20	2	20-Jan-2000	24500	RUE DE LYON, 03
60	2	7-Jan-2001	3100	RUE DE GENÈVE, 120
40	3	20-Sep-2000	6000	RUE DE LAUSANNE, 01

→ **AVG(...)**

NUMCLIENT	AVG(TOTAL)
1	12200
2	13800
3	6000

→ **Condition
HAVING**

NUMCLIENT	AVG(TOTAL)
1	12200
2	13800

34

Requêtes regroupées

- Quel est le montant moyen des commandes **de l'année 2000** pour chaque client, **pour lesquels le montant moyen est supérieur à 10000** ?

```
SQL> SELECT AVG(Total) FROM Commande
WHERE DateCde BETWEEN '01-JAN-00' AND '31-DEC-00'
GROUP BY NumClient
HAVING AVG(Total) > 10000
```

35

Requêtes regroupées

NUMCDE	NUMCLIENT	DATECDE	TOTAL	ADRESSELIVRAISON
10	1	15-Dec-2099	32500	GÉNÉRAL DUFOUR, 24
20	2	20-Jan-2000	24500	RUE DE LYON, 03
30	1	18-Jun-2000	2500	MIREMONT, 29
40	3	20-Sep-2000	6000	RUE DE LAUSANNE, 01
50	1	30-Dec-2000	1600	GÉNÉRAL DUFOUR, 24
60	2	7-Jan-2001	3100	RUE DE GENÈVE, 120

→ Condition **WHERE**

NUMCDE	NUMCLIENT	DATECDE	TOTAL	ADRESSELIVRAISON
20	2	20-Jan-2000	24500	RUE DE LYON, 03
30	1	18-Jun-2000	2500	MIREMONT, 29
40	3	20-Sep-2000	6000	RUE DE LAUSANNE, 01
50	1	30-Dec-2000	1600	GÉNÉRAL DUFOUR, 24

→ **GROUP BY**

→ **AVG(..)**

NUMCDE	NUMCLIENT	DATECDE	TOTAL	ADRESSELIVRAISON
30	1	18-Jun-2000	2500	MIREMONT, 29
50	1	30-Dec-2000	1600	GÉNÉRAL DUFOUR, 24
20	2	20-Jan-2000	24500	RUE DE LYON, 03
40	3	20-Sep-2000	6000	RUE DE LAUSANNE, 01

NUMCLIENT	AVG(TOTAL)
1	2050
2	24500
3	6000

NUMCLIENT	AVG(TOTAL)
2	24500

→ Condition **HAVING**

36



Contenu

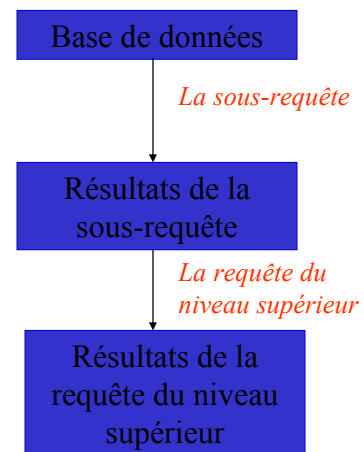
- Introduction aux requêtes
- Expressions, conditions et opérateurs
- Fonctions
- Clauses dans le SQL
- Requêtes regroupées
- **Sous-requêtes**
- Créer une table

37



Sous-requêtes

- Qu'est-ce qu'une sous-requête ?
 - Une **sous-requête** est une requête intégrée à une autre requête.
 - Les résultats d'une **sous-requête** sont utilisés pour déterminer les résultats de la **requête du niveau supérieur**
 - La sous-requête est incluse dans l'une des conditions de recherche **des clauses WHERE ou HAVING**



38

Sous-requêtes

- Donnez la liste de tous les commandes dont le total est supérieure au total de la commande numéro 40

```
SELECT *
FROM Commande
WHERE Total >
(SELECT Total
FROM Commande
WHERE NumCde = 40)
```

La sous-requête

La requête du niveau supérieur

NUMCDE	NUMCLIENT	DATECDE	TOTAL	ADRESSELIVRAISON
10		1 15-Dec-2099	32500	GÉNÉRAL DUFOUR, 24
20		2 20-Jan-2000	24500	RUE DE LYON, 03
30		1 18-Jun-2000	2500	MIREMONT, 29
40		3 20-Sep-2000	6000	RUE DE LAUSANNE, 01
50		1 30-Dec-2000	1600	GÉNÉRAL DUFOUR, 24
60		2 7-Jan-2001	3100	RUE DE GENÈVE, 120

NUMCDE	NUMCLIENT	DATECDE	TOTAL	ADRESSELIVRAISON
40		3 20-Sep-2000	6000	RUE DE LAUSANNE, 01

NUMCDE	NUMCLIENT	DATECDE	TOTAL	ADRESSELIVRAISON
10		1 15-Dec-2099	32500	GÉNÉRAL DUFOUR, 24
20		2 20-Jan-2000	24500	RUE DE LYON, 03

39

Sous-requêtes

- Conditions de recherche des sous-requêtes
 - La condition de comparaison des sous-requêtes**
utilise
 les opérateurs de comparaison
 (= , <> , < , <= , > , >=)
 - La condition d'appartenance à une liste (IN)**
compare
 une valeur à un ensemble de valeurs produit par une sous-requête

40

Sous-requêtes

- Donnez la liste de tous les commandes dont le total est supérieure au total de la commande numéro 40

```
SELECT *
FROM Commande
WHERE Total >
(SELECT Total
FROM Commande
WHERE NumCde = 40)
```

La sous-requête

La requête du niveau supérieur

NUMCDE	NUMCLIENT	DATECDE	TOTAL	ADRESSELIVRAISON
10	1	15-Dec-2099	32500	GÉNÉRAL DUFOUR, 24
20	2	20-Jan-2000	24500	RUE DE LYON, 03
30	1	18-Jun-2000	2500	MIREMONT, 29
40	3	20-Sep-2000	6000	RUE DE LAUSANNE, 01
50	1	30-Dec-2000	1600	GÉNÉRAL DUFOUR, 24
60	2	7-Jan-2001	3100	RUE DE GENÈVE, 120

NUMCDE	NUMCLIENT	DATECDE	TOTAL	ADRESSELIVRAISON
40	3	20-Sep-2000	6000	RUE DE LAUSANNE, 01

NUMCDE	NUMCLIENT	DATECDE	TOTAL	ADRESSELIVRAISON
10	1	15-Dec-2099	32500	GÉNÉRAL DUFOUR, 24
20	2	20-Jan-2000	24500	RUE DE LYON, 03

41

Sous-requêtes

- Affichez les clients qui ont passé des commandes dont la moyenne des totaux est supérieure à 1000

```
SELECT *
FROM client
WHERE numclient IN
( SELECT numclient FROM
commande
GROUP BY numclient HAVING
AVG(total) > 10000 )
```

La sous-requête

La requête du niveau supérieur

NUMCDE	NUMCLIENT	DATECDE	TOTAL	ADRESSELIVRAISON
10	1	15-Dec-2099	32500	GÉNÉRAL DUFOUR, 24
20	2	20-Jan-2000	24500	RUE DE LYON, 03
30	1	18-Jun-2000	2500	MIREMONT, 29
40	3	20-Sep-2000	6000	RUE DE LAUSANNE, 01
50	1	30-Dec-2000	1600	GÉNÉRAL DUFOUR, 24
60	2	7-Jan-2001	3100	RUE DE GENÈVE, 120

NUMCLIENT
1
2

NUMCLIENT	NOM	PRENOM	ADRESSECLIENT
1	MENU	NILOLAS	GÉNÉRAL DUFOUR, 24
2	LEE	ROBERTO	MIREMONT, 46

42



Sous-requêtes

- **La condition d'appartenance à une liste**
 - *Ex: Affichez les clients qui ont passé des commandes dont la moyenne des totaux n'est pas supérieure à 1000*

```
SELECT DISTINCT numclient, nom, prenom, adresse FROM client
WHERE numclient NOT IN
( SELECT numclient FROM commande
GROUP BY numclient HAVING AVG(total) > 1000 )
```

43



Contenu

- Introduction aux requêtes
- Expressions, conditions et opérateurs
- Fonctions
- Clauses dans le SQL
- Requêtes regroupées
- Sous-requêtes
- **Créer une table**

44



Créer une table

1. **PRE:** la table existe déjà?
2. **CREER** cette table
 - Information général
 - Informations sur les colonnes
 - Informations sur les contraintes
3. **POST:** la table est déjà crée ?

45



Créer une table

- **PRE:** la table existe déjà ?

– **Exemple:**

SELECT **TABLE_NAME** **FROM** **TABS** ;

SELECT * **FROM** **TABS**

WHERE **TABLE_NAME** = 'CHAUFFEUR';

TABLE_NAME
AFFCHAUFFEUR
no rows selected



- **TABS** : cette table est la table du dictionnaire qui contient les informations sur les tables des utilisateurs
- **TABLE_NAME** : une colonne de **TABS**

46



Créer une table

- Créer une table:
 - Information général:
 - Quel sera **le nom de la table** ?
 - Informations sur les colonnes:
 - Quelles colonnes devront contenir des données?
 - Quels seront **les noms des colonnes** ?
 - Quels **type de donnée** sera assignée à chaque colonne ?
 - Quelle sera **la longueur** allouée à chaque colonne?
 - Informations sur les contraintes:
 - Quelles colonnes constitueront **la clé primaire**?

47



Créer une table

- Créer une table:
 - Informations sur les colonnes
 - Exemple:
CREATE TABLE chauffeur
(NoAVS CHAR(9),
Nom CHAR(24),
Prenom CHAR(24),
Datenaissance DATE,
Adresse CHAR(80),
Sexe CHAR(1),
Totalheures INTEGER)

Table created.

48



Créer une table

- Informations sur les contraintes
 - Clé primaire:

```
CREATE TABLE chauffeur
(NoAVS CHAR(9) CONSTRAINT pk_chauffeur PRIMARY KEY,
Nom CHAR(24),
Prenom CHAR(24),
Datenaissance DATE,
Adresse CHAR(80),
Sexe CHAR(1),
Totalheures INTEGER)
```

49



Créer une table

- Informations sur les contraintes
 - NOT NULL

```
CREATE TABLE chauffeur
(NoAVS CHAR(9) CONSTRAINT pk_chauffeur PRIMARY KEY,
Nom CHAR(24) CONSTRAINT nn_nom NOT NULL,
Prenom CHAR(24),
Datenaissance DATE,
Adresse CHAR(80),
Sexe CHAR(1),
Totalheures INTEGER)
```

50

Créer une table

- Informations sur les contraintes
 - **Domaine**

```
CREATE TABLE chauffeur
(NoAVS CHAR(9) CONSTRAINT pk_chauffeur PRIMARY KEY,
Nom CHAR(24) CONSTRAINT nn_nom_chauffeur NOT NULL,
Prenom CHAR(24),
Datenaissance DATE,
Adresse CHAR(80),
Sexe CHAR(1) CONSTRAINT ck_sexe_chauffeur
CHECK (sexe = 'M' OR sexe = 'F'),
Totalheures INTEGER CONSTRAINT ck_totalheures_chauffeur
CHECK (totalheures >= 0))
```

51

Créer une table

- POST: la table est déjà créée ?

– **Exemple:**

SELECT TABLE_NAME FROM TABS ;

**SELECT TABLE_NAME,
TABLESPACE_NAME
FROM TABS**

WHERE TABLE_NAME = 'CHAUFFEUR'

DESCRIBE chauffeur ;

TABLE_NAME		

AFFCHAUFFEUR		
CHAUFFEUR		
TABLE_NAME		TABLESPACE_NAME
-----		-----
CHAUFFEUR		USERS
Name	Null?	Type

NOAVS	NOT NULL	NUMBER(38)
NOM	NOT NULL	CHAR(24)
PRENOM		CHAR(24)
DATENAISSANCE		DATE
ADRESSE		CHAR(80)
SEXE		CHAR(1)
TOTALHEURES		NUMBER(38)



Pour modifier en cas d'erreur :
ALTER TABLE chauffeur ;

52

Créer une table

- ... Insérer les données

- **Exemple:**

```
INSERT INTO Chauffeur VALUES ('000111222001','Lee','Kim',  
    '18-APR-67','24 General Dufour','M',400);  
INSERT INTO Chauffeur VALUES ('000111222002','Lee','Loulou',  
    '15-DEC-72','Uni Mail CH 1204','F',200);  
INSERT INTO Chauffeur VALUES ('000111222003','Meyer','Boubou',  
    '23-DEC-82','Gare Cornavin','M',300);  
INSERT INTO Chauffeur VALUES  
    ('000111222004','Lambert','Nounou',  
    '01-JAN-77','Chene-Bourge','M',100);
```

```
1 row created.  
  
1 row created.  
  
1 row created.  
  
1 row created.
```



Pour valider :
SELECT * FROM chauffeur...;

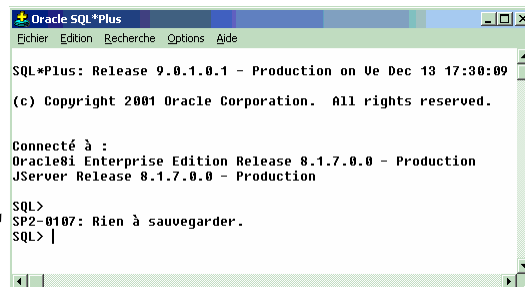
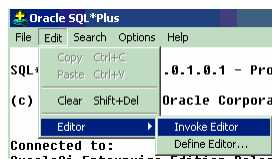
53

Q&A

- **Question:**

des problèmes (SQL*Plus à CUI)

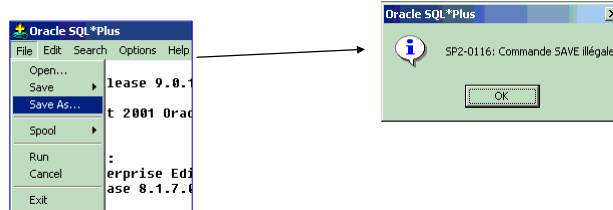
- **Appeler Éditeur**



54

Q&A

- **Question:**
des problèmes (SQL*Plus à CUI)
 - Sauvegarder un script



55

Q&A

- **Answer:** on n'a pas assez de droits ?
- **Tip:**
 - 1. Exécuter une commande de SQL

```
Oracle SQL*Plus
File Edit Search Options Help

SQL*Plus: Release 9.0.1.0.1 - Production on Fri Dec 13 18:03:20 2002
(c) Copyright 2001 Oracle Corporation. All rights reserved.

Connected to:
Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production
JServer Release 8.1.7.0.0 - Production

SQL> SELECT SYSDATE FROM DUAL;

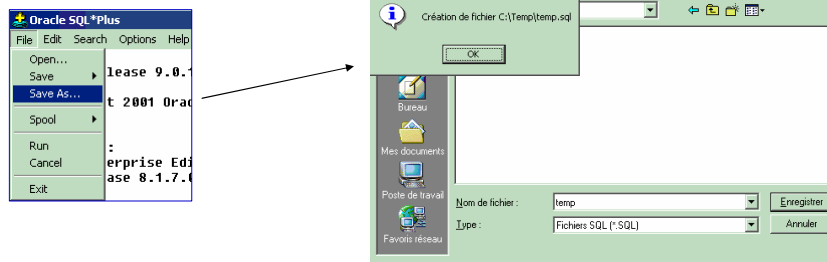
SYSDATE
-----
13-DEC-02
```

56

Q&A

■ Tip:

- 2. Sauvegarder un script dans votre répertoire privé (par exemple: C:\TEMP)



57