Le langage SQL
Langage d'Intérrogation de Données

Ines BAKLOUTI

ines.baklouti@esprit.tn

Ecole Supérieure Privée d'Ingénierie et de Technologies

Plan

Plan

Instruction SELECT

Syntaxe

SELECT * | { [DISTINCT] <colonne> | <expression> [alias],...}
FROM <nom_table>;

- SELECT : indique les colonnes à afficher
- DISTINCT : supprime les doublons
- FROM : indique les tables contenant les colonnes

Exemple 1

SELECT * FROM departments;

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700

Plus de 10 lignes sont disponibles. Augmentez le sélecteur de lignes pour afficher plus de lignes.

Instruction SELECT

Exemple 2

SELECT department_id, department_name FROM departments;

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
30	Purchasing
40	Human Resources
50	Shipping
60	IT
70	Public Relations
80	Sales
90	Executive
100	Finance

Plus de 10 lignes sont disponibles. Augmentez le sélecteur de lignes pour afficher plus de lignes.

Exemple 3

SELECT DISTINCT department_id FROM employees;

DEPARTMENT_ID
100
30
-
90
20
70
110

Expressions arithmétiques

- Expressions contenant des données de type NUMBER, DATE et des opérateurs arithmétiques

Opérateur	Description
+	Addition
-	Soustraction
*	Multiplication
/	Division

Exemple

SELECT last_name, first_name, salary+1000, commission_pct*100 FROM employees;

LAST_NAME	FIRST_NAME	SALARY+1000	COMMISSION_PCT*100
Higgins	Shelley	13000	-
Gietz	William	9300	-
Cambrault	Gerald	12000	30
Zlotkey	Eleni	11500	20
Tucker	Peter	11000	30
Bernstein	David	10500	25
Hall	Peter	10000	25

Valeur NULL

- NULL représente une valeur non disponible, non affectée
- NULL est différente de zéro, espace ou chaîne vide

Exemple

SELECT last_name, first_name, commission_pct FROM employees;

LAST_NAME	FIRST_NAME	COMMISSION_PCT
Higgins	Shelley	-
Gietz	William	-
Cambrault	Gerald	,3
Zlotkey	Eleni	,2
Tucker	Peter	,3
Bernstein	David	,25
Hall	Peter	,25
Olsen	Christopher	,2
Cambrault	Nanette	,2

Valeur NULL et expressions arithméthiques

- Les expressions arithmétiques comportant une valeur NULL renvoient toujours une valeur NULL

Exemple

SELECT last_name, salary, salary+commission_pct, salary*commission_pct FROM employees;

LAST_NAME	SALARY	SALARY+COMMISSION_PCT	SALARY*COMMISSION_PCT
Higgins	12000	-	-
Gietz	8300	-	-
Cambrault	11000	11000,3	3300
Zlotkey	10500	10500,2	2100
Tucker	10000	10000,3	3000
Bernstein	9500	9500,25	2375
Hall	9000	9000,25	2250

Allias de colonne

- Un alias de colonne :
  - Renomme un entête de colonne
  - Est utile avec les calculs
  - Suit immédiatement le nom d'une colonne (le mot clé facultatif AS peut également être utilisé entre le nom de la colonne et l'alias)
  - Nécessité des guillemets ("alias") s'il contient des espaces ou des caractères spéciaux (# $), ou s'il distingue les majuscules des minuscules

Exemple

SELECT last_name nom, first_name AS prénom, salary*12 "revenu annuel" FROM employees;

NOM	PRÉNOM	Revenu Annuel
King	Steven	288000
Kochhar	Neena	204000
De Haan	Lex	204000
Hunold	Alexander	108000
Ernst	Bruce	72000
Austin	David	57600
Pataballa	Valli	57600
Lorentz	Diana	50400

Opérateur de concaténation

- Concatène des colonnes ou des chaines de caractères
- Est représenté par le symbole ||
- La colonne résultante est une expression de type caratère

Exemple

SELECT department_id||' ** '||department_name AS "département" from departments;

Département
10 ** Administration
20 ** Marketing
30 ** Purchasing
40 ** Human Resources
50 ** Shipping
60 ** IT
70 ** Public Relations
80 ** Sales

Restriction de données: la clause WHERE

- Restreindre les lignes renvoyées à l'aide la clause WHERE

Syntaxe

SELECT * | { [DISTINCT] <colonne> | <expression> [alias],...}
FROM <nom_table>
[WHERE <condition(s)>];

Exemple

SELECT employee_id, last_name, department_id
FROM employees
WHERE department_id= 80;

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
145	Russell	80
146	Partners	80
147	Errazuriz	80
148	Cambrault	80
149	Zlotkey	80
150	Tucker	80
151	Bernstein	80
152	Hall	80
153	Olsen	80

Chaînes de caractères et dates

- Les chaînes de caractères et les dates sont incluses entre apostrophes.
- Les valeurs de type caractère distinguent les majuscules des minuscules
- Les valeurs de type date sont sensibles au format
- Le format de date par défaut est DD-MM-RR

Exemples

1. SELECT employee_id,first_name FROM employees WHERE first_name='James';

EMPLOYEE_ID	FIRST_NAME
127	James
131	James

2. SELECT employee_id,first_name FROM employees WHERE first_name='JAMES';
   => aucune ligne sélectionnée

Opérateurs de comparaison

Opérateur	Description
=	Egal à
<	Inférieur à
<=	Inférieur à ou égal
>	Supérieur à
>=	Supérieur à ou égale à
<> ou !=	Différent
BETWEEN val1 AND val2	Valeur comprise entre val1 et val2
In (val1,val2,…,valN)	Appartient à une liste de valeurs
Like	Correspond à un modèle de chaînes de caractères
IS NULL	Correspond à une valeur NULL

Opérateurs de comparaison

L'opérateur BETWEEN

Exemple 1

SELECT first_name, salary FROM employees
WHERE salary BETWEEN 15000 AND 20000;

FIRST_NAME	SALARY
Neena	17000
Lex	17000

Exemple 2

SELECT first_name, salary FROM employees
WHERE first_name BETWEEN 'V' AND 'X';

FIRST_NAME	SALARY
Valli	4800
William	7400
Winston	3200
Vance	2800
William	8300

Opérateurs de comparaison

L'opérateur IN

Exemple 1

```
SELECT first_name, department_id FROM employees
WHERE department_id IN (10,20);
```

FIRST_NAME	DEPARTMENT_ID
Pat	20
Michael	20
Jennifer	10

Exemple 2

```
SELECT first_name, last_name FROM employees
WHERE first_name IN ('James','David','Diana');
```

FIRST_NAME	LAST_NAME
David	Austin
David	Bernstein
James	Landry
David	Lee
Diana	Lorentz
James	Marlow

Opérateurs de comparaison

L'opérateur LIKE

- L'opérateur LIKE permet de rechercher des chaînes de caractères à l'aide de caractères génériques
- Les conditions de recherche peuvent contenir des caractères ou des nombres littéraux
  - % représente Zéro ou plusieurs caractères
  - _ représente un caractère

Exemple

```
SELECT first_name FROM employees
WHERE first_name LIKE 'S_e%' ;
```

FIRST_NAME
Shelli
Stephan
Shelley
Steven
Steven
Stephen

Opérateurs de comparaison

L'opérateur IS NULL

- L'opérateur IS NULL permet de tester la présence de valeurs NULL.

Exemple

SELECT first_name, manager_id FROM employees
WHERE manager_id IS NULL;

FIRST_NAME	MANAGER_ID
Steven	-
Stephan	-

Opérateurs logiques

OPERATEUR	DESCRIPTION
AND	Retourne TRUE si les deux conditions sont VRAIES
OR	Retourne TRUE si au moins une des conditions est VRAIE
NOT	Inverse la valeur de la condition • TRUE si la condition est FAUSSE • FALSE si la condition est VRAIE

Opérateurs logiques

L'opérateur AND

Exemple

SELECT last_name, job_id, salary FROM employees
WHERE salary >=10000 AND job_id LIKE '%MAN%';

LAST_NAME	JOB_ID	SALARY
Russell	SA_MAN	14000
Partners	SA_MAN	13500
Errazuriz	SA_MAN	12000
Cambrault	SA_MAN	11000
Zlotkey	SA_MAN	10500
Hartstein	MK_MAN	13000

Opérateurs logiques

L'opérateur OR

Exemple

SELECT last_name, job_id, salary FROM employees
WHERE salary >=10000 OR job_id LIKE '%MAN%';

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000
Kochhar	AD_VP	17000
De Haan	AD_VP	17000
Greenberg	FI_MGR	12000
Raphaely	AC_MGR	12000
Tobias	AC_MGR	12000
Weiss	ST_MAN	8000
Fripp	ST_MAN	8200
Kaufling	ST_MAN	7900
Vollman	ST_MAN	6500
Plus de 10 lignes sont disponibles. Augmentez le sélecteur de lignes pour afficher plus de lignes.		

Opérateurs logiques

L'opérateur NOT

Exemple

SELECT last_name, job_id FROM employees
WHERE job_id NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP') ;

LAST_NAME	JOB_ID
King	AD_PRES
Kochhar	AD_VP
De Haan	AD_VP
Greenberg	FI_MGR
Faviet	FI_ACCOUNT
Chen	FI_ACCOUNT
Sciarra	FI_ACCOUNT
Urman	FI_ACCOUNT
Popp	FI_ACCOUNT
Raphaely	AC_MGR
Plus de 10 lignes sont disponibles. Augmentez le sélecteur de lignes pour afficher plus de lignes.	

Opérateurs logiques

Règles de priorité

ORDRE DE PRIORITE	OPERATEUR
1	Les parenthèses
2	Tous les opérateurs de comparaison
3	NOT
4	AND
5	OR

Opérateurs logiques

Règles de priorité

Exemple 1

SELECT last_name, job_id, salary FROM employees
WHERE job_id= 'SA_MAN' OR job_id= 'AD_VP' AND salary > 12000;

LAST_NAME	JOB_ID	SALARY
Kochhar	AD_VP	17000
De Haan	AD_VP	17000
Russell	SA_MAN	14000
Partners	SA_MAN	13500
Errazuriz	SA_MAN	12000
Cambrault	SA_MAN	11000
Zlotkey	SA_MAN	10500

Exemple 1

SELECT last_name, job_id, salary FROM employees
WHERE (job_id= 'SA_MAN' OR job_id= 'AD_VP') AND salary > 12000;

LAST_NAME	JOB_ID	SALARY
Kochhar	AD_VP	17000
De Haan	AD_VP	17000
Russell	SA_MAN	14000
Partners	SA_MAN	13500

Tri de données: la clause ORDER BY

- La clause ORDER BY:
  - permet de triez les lignes extraites
    - ASC : ordre croissant (par défaut)
    - DESC : ordre décroissant
  - toujours la dernière clause dans l'instruction SELECT

Syntaxe

SELECT * | { [DISTINCT] <colonne> | <expression> [alias],...}
FROM <nom_table>;
[WHERE condition(s)]
[ORDER BY {<colonne>, <expression>, <alias>} [ASC | DESC]] ;

Exemple 1: tri par ordre décroissant

SELECT last_name, hire_date FROM employees
ORDER BY hire_date DESC ; –ou bien ORDER By 2 DESC

LAST_NAME	HIRE_DATE
Derode	26/02/10
Banda	21/04/00
Kumar	21/04/00
Ande	24/03/00
Markle	08/03/00
Lee	23/02/00

Tri de données: la clause ORDER BY

Exemple 2: tri par alias de colonne

SELECT employee_id, last_name, salary*12 "Salaire Annuel"
FROM employees
ORDER BY "Salaire Annuel"; –ou bien ORDER By 3

EMPLOYEE_ID	LAST_NAME	Salaire Annuel
132	Olson	25200
128	Markle	26400
136	Philtanker	26400
135	Gee	28800
127	Landry	28800
119	Colmenares	30000
131	Marlow	30000

Exemple 3: tri selon plusieurs colonnes

SELECT employee_id, last_name, salary*12 "Salaire Annuel"
FROM employees
ORDER BY "Salaire Annuel", last_name DESC; —ou bien ORDER By 3, 2 DESC

EMPLOYEE_ID	LAST_NAME	Salaire Annuel
132	Olson	25200
136	Philtanker	26400
128	Markle	26400
127	Landry	28800
135	Gee	28800
144	Vargas	30000
182	Sullivan	30000

Plan

Les fonctions

- Il existe 3 types de fonctions dans le language SQL
  - Fonctions mono-ligne: manipulent une seule ligne et ramènent un seul résultat
  - Fonctions analytiques: manipulent plusieurs lignes et ramènent un plusieurs résultats
  - Fonctions multi-lignes: manipulent plusieurs lignes et ramènent un seul résultat

Les fonctions de caractères

Fonctions de conversion majuscules/minuscules

Fonction	Exemple	Résultat	Description
LOWER(chaine)	LOWER('SQL Course')	sql course	convertit les caractères majuscules de «chaine» en minuscules
UPPER(chaine)	UPPER('SQL Course')	SQL COURSE	convertit les caractères minuscules de «chaine» en majuscules
INITCAP(chaine)	INITCAP('SQL Course')	Sql Course	convertit l'initiale de chaque mot de «chaine» en majuscule et les caractères suivants en minuscules

Exemple

SELECT first_name, lower(first_name) , upper(first_name), initcap(first_name) FROM employees;

FIRST_NAME	LOWER(FIRST_NAME)	UPPER(FIRST_NAME)	INITCAP(FIRST_NAME)
Ellen	ellen	ELLEN	Ellen
Sundar	sundar	SUNDAR	Sundar
Mozhe	mozhe	MOZHE	Mozhe
David	david	DAVID	David
Hermann	hermann	HERMANN	Hermann
Shelli	shelli	SHELLI	Shelli

Les fonctions de caractères

Fonctions de manipulation de caractères

Fonction	Exemple	Résultat	Description
CONCAT(chain1,chaine2)	CONCAT('Hello', 'World')	HelloWorld	concatène la première «chaine1» avec «chaine2 » (comme l'opérateur ||)
SUBSTR(chaine,pos[,long])	SUBSTR('HelloWorld',1,5)	Hello	extrait une sous chaîne de «chaine» à partir de la position «pos» et de longueur «long»
LENGTH(chaine)	LENGTH('HelloWorld')	10	taille d'une chaîne de caractères
INSTR(chaine1,chaine2)	INSTR('HelloWorld', 'W')	6	position de « chaine2» dans «chaine1»
LPAD(chaine,long [,car])	LPAD(salary,10,'*')	*****24000	complète (ou tronque) «chaine» à gauche à la longueur «long» par le caractère «car» (espace par défaut)
RPAD(chaine,long [,car])	RPAD(salary, 10, '*')	24000*****	complète (ou tronque) «chaine» à droite à la longueur «long» par le caractère «car» (espace par défaut)
LTRIM(chaine [,car])	LTRIM('abbabaa','a')	bbabaa	supprime les caractères à gauche de la chaîne «chaine» tant qu'ils appartiennent à l'ensemble de caractères «car» (espace par défaut)
RTRIM(chaîne [,car])	RTRIM('abbabaa','a')	abbab	supprime les caractères à droite de la chaîne «chaine» tant qu'ils appartiennent à l'ensemble de caractères «car» (espace par défaut)
REPLACE(chaine,ch1,ch2)	REPLACE('JACK and JUE','J','BL')	BLACK and BLUE	remplace toutes les occurrences de «ch1» dans «chaine» par «ch2»
TRANSLATE(chaine,ch1,ch2)	TRANSLATE('JACK and JUE','J','BL')	BACK and BUE	remplace chaque caractère de «ch1» dans «chaine» par son correspondant dans «ch2»
ASCII(chaine)	ASCII('Black')	66	retourne le code ASCII du premier caractère de «chaine»
CHR(n)	CHR(98)	b	retourne le caractère ayant le code ASCII «n»

Les fonctions de caractères

Fonctions de manipulation de caractères

Exemple

```
SELECT first_name, last_name, job_id,
CONCAT(first_name,last_name) "Nom et prénom",
LENGTH (last_name) "longueur nom",
INSTR(last_name,'a') "position a",
LPAD(last_name,10,'*'),
RPAD(last_name,10,'*')
FROM employees
WHERE SUBSTR(job_id, 4) = 'REP';
```

FIRST_NAME	LAST_NAME	JOB_ID	Nom Et Prénom	Longueur Nom	Position A	LPAD(LAST_NAME,10,'*')	RPAD(LAST_NAME,10,'*')
Peter	Tucker	SA_REP	PeterTucker	6	0	****Tucker	Tucker****
David	Bernstein	SA_REP	DavidBernstein	9	0	*Bernstein	Bernstein*
Peter	Hall	SA_REP	PeterHall	4	2	******Hall	Hall******
Christopher	Olsen	SA_REP	ChristopherOlsen	5	0	*****Olsen	Olsen*****
Nanette	Cambrault	SA_REP	NanetteCambrault	9	2	*Cambrault	Cambrault*
Oliver	Tuvault	SA_REP	OliverTuvault	7	4	***Tuvault	Tuvault***
Janette	King	SA_REP	JanetteKing	4	0	******King	King******
Patrick	Sully	SA_REP	PatrickSully	5	0	*****Sully	Sully*****
Allan	McEwen	SA_REP	AllanMcEwen	6	0	****McEwen	McEwen****

Les fonctions numériques

Fonction	Exemple	Résultat	Description
ABS(n)	ABS(-5)	5	Valeur absolue de n
MOD(n1, n2)	MOD(15,4)	3	reste de division de n1 par n2
POWER(n, e)	POWER(2,3)	8	n à la puissance e
SIGN(n)	SIGN(6)	1	Retourne -1 si n<0, 0 si n=0, 1 si n>0
SQRT(n)	SQRT(4)	16	racine carrée de n
ROUND(n[, p])	ROUND(45.926, 2) ROUND(45.924, 2) ROUND(-45.925,2)	45.93 45.92 -45.93	arrondit n à la précision p (0 par défaut)
TRUNC(n[, p])	TRUNC(45.926, 2) TRUNC(45.924, 2) TRUNC(-45.925,2)	45.92 45.92 -45.92	tronque n à la précision p (0 par défaut)
FLOOR(n)	FLOOR(45.926) FLOOR(-45.926)	45 -46	retourne la partie entière inférieure (si n<x<n+1 alors FLOOR(x)=n)
CEIL(n)	CEIL(45.926) CEIL(-45.926)	46 -45	retourne la partie entière supérieure (si n<x<n+1 alors CEIL(x)=n+1)
GREATEST(n1, n2,...)	GREATEST(4,-1,7)	7	maximum de n1, n2,...
LEAST(n1, n2,...)	LEAST(4,-1,7)	-1	minimum de n1, n2,...

Les fonctions numériques

Exemple

```
SELECT commission_pct+0.2,
ROUND(commission_pct+0.2),
TRUNC(commission_pct+0.2),
FLOOR(commission_pct+0.2),
CEIL(commission_pct+0.2)
FROM employees where department_id=80;
```

COMMISSION_PCT+0.2	ROUND(COMMISSION_PCT+0.2)	TRUNC(COMMISSION_PCT+0.2)	FLOOR(COMMISSION_PCT+0.2)	CEIL(COMMISSION_PCT+0.2)
,6	1	0	0	1
,5	1	0	0	1
,5	1	0	0	1
,5	1	0	0	1
,4	0	0	0	1
,5	1	0	0	1
,45	0	0	0	1

Les fonctions de dates

- Dans la base de données Oracle, les dates sont stockées dans un format numériques interne: siècle, année, mois, jour, heures, minutes et secondes
- Le format de date par défaut est 'DD-MON-YY'
- SYSDATE est une fonction qui renvoie :
  - La date
  - L'heure

Les fonctions de dates

Calcul arithmétique sur les dates

- Calcul arithmétique sur des dates:
  - ajout ou soustraction d'un nombre de jour à une date afin d'obtenir une date résultante
  - Ajout ou soustraction d'un nombre d'heures à une date en divisant le nombre d'heures par 24
  - soustraction d'une date d'une autre afin de déterminer le nombre de jours entre les deux dates

Exemple

```
SELECT first_name,
(SYSDATE-hire_date) AS jours,
(SYSDATE-hire_date)/7 AS semaines
FROM employees;
```

FIRST_NAME	JOURS	SEMAINES
Steven	9634,48894675925925925925925925925926	1376,35556382275132275132275132275132275
Neena	8807,48894675925925925925925925925926	1258,21270667989417989417989417989417989
Lex	7597,48894675925925925925925925925926	1085,35556382275132275132275132275132275
Alexander	8703,48894675925925925925925925925926	1243,35556382275132275132275132275132275
Bruce	8200,48894675925925925925925925925926	1171,49842096560846560846560846560846561
David	5973,48894675925925925925925925925926	853,355563822751322751322751322751322751

Les fonctions de dates

Fonction	Exemple	Résultat	Description
SYSDATE ou CURRENT_DATE			retourne la date et l'heure courante du système d'exploitation hôte
MONTHS_BETWEEN(d1,d2)	MONTHS_BETWEEN ('01-SEP-95','11-JAN-94')	19.6774194	Nombre de mois entre deux dates d1 et d2
ADD_MONTHS(d,j)	ADD_MONTHS ('11-JAN-94',6)	11-JUL-94	ajoute j mois à une date d
NEXT_DAY(d,j)	NEXT_DAY ('01-SEP-95','FRIDAY')	08-SEP-95	retourne la date du jour j qui suit la date d
LAST_DAY(d)	LAST_DAY ('01-FEB-95')	28-FEB-95	retourne le dernier jour du mois de la date d
ROUND(d, p)	•ROUND(to_date('25-07-03','DD-MM-YY'),'MONTH') •ROUND(to_date('25-07-03','DD-MM-YY'),'YEAR')	01-08-03 01-01-04	arrondit la date d à la précision p. La précision est indiquée en utilisant un des masques de mise en forme de la date
TRUNC(d, p)	•TRUNC(to_date('25-07-03','DD-MM-YY'),'MONTH') •TRUNC(to_date('25-07-03','DD-MM-YY'),'YEAR')	01-07-03 01-01-03	tronque la date d à la précision p
EXTRACT(p FROM d)	extract (day from to_date('25-07-03','DD-MM-YY'))	25	extrait la précision p de la date d

Les fonctions de dates

Exemple

```
SELECT systimestamp,
extract(year from systimestamp) as Année,
extract(month from systimestamp) as Mois,
extract(day from systimestamp) as Jour,
extract(hour from systimestamp) as Hour,
extract(minute from systimestamp) as Minutes,
extract(second from systimestamp) as Secondes
FROM dual;
```

SYSTIMESTAMP	ANNÉE	MOIS	JOUR	HOUR	MINUTES	SECONDES
01-NOV. -13 12.48.03,354000 PM +01:00	2013	11	1	11	48	3,354

Les fonctions de conversion

Conversion implicite

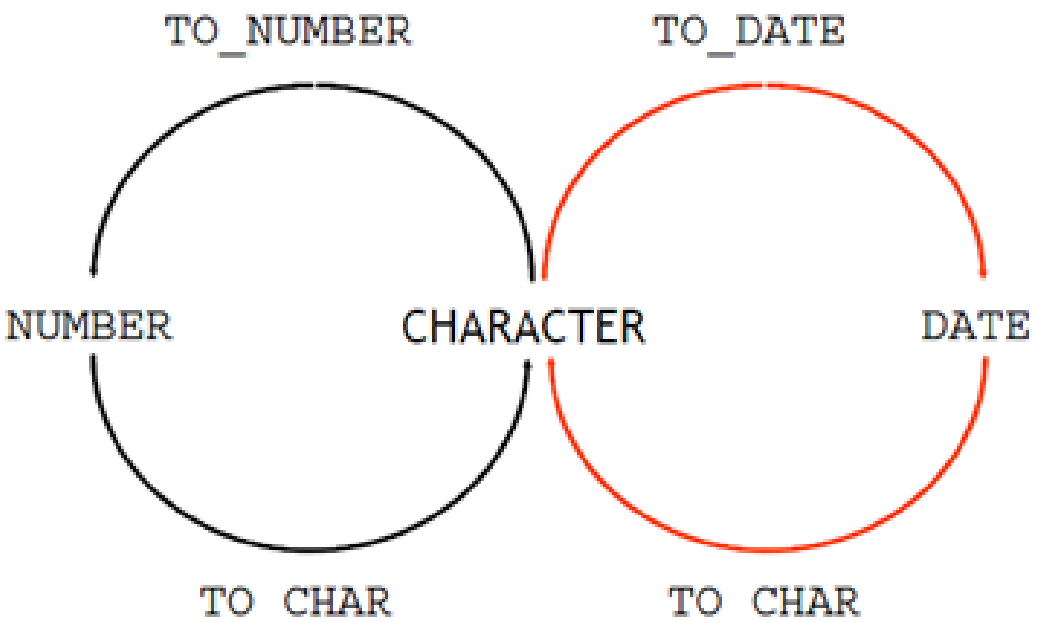
- Pour les affectations, le serveur Oracle peut convertir automatiquement les types de données suivants:

FROM	TO
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE

- L'expression salary=‘2000’ entraîne la conversion implicite de la chaîne ‘2000’ en valeur numérique 2000
- L'expression hire_date>'01-Jan-90' entraîne la conversion implicite de la chaîne '01-Jan-90' en date

Les fonctions de conversion

Conversion explicite



Les fonctions de conversion

Conversion explicite: fonction TO_CHAR

Fonction	Description
TO_CHAR (date, format)	permet de convertir une date en chaîne de caractère en fonction d'un format • YYYY année • YY deux derniers chiffres de l'année • WW numéro de la semaine dans l'année • MM numéro du mois • MONTH nom du mois • MON nom abrévié du mois • DDD numéro du jour dans l'année • DD numéro du jour dans le mois • D numéro du jour dans la semaine (1 pour lundi, 7 pour dimanche) • HH ou HH12 heure (sur 12 heures) • HH24 heure (sur 24 heures) • MI minutes • SS secondes
TO_CHAR (nombre, format)	permet de convertir un nombre en chaîne de caractère en fonction d'un format • 9 représente un chiffre (non représenté si non significatif) • 0 représente un chiffre (présent même si non significatif) • . point décimal apparent • , une virgule apparaîtra à cet endroit • $ un $ précédera le premier chiffre significatif • B le nombre sera représente par des blancs s'il vaut zéro • MI le signe négatif sera à droite • PR un nombre négatif sera entre <>

Les fonctions de conversion

Conversion explicite:fonction TO_CHAR

Exemple 1

SELECT last_name, TO_CHAR(hire_date, 'fm DD Month YYYY') AS HIREDATE FROM employees;

LAST_NAME	HIREDATE
King	17 Juin 1987
Kochhar	21 Septembre 1989
De Haan	13 Janvier 1993
Hunold	3 Janvier 1990
Ernst	21 Mai 1991

- fm permet de supprimer les espaces de remplissage ou les zéros de début

Exemple 2

SELECT first_name, TO_CHAR(salary, '$99,000.00') SALARY FROM employees;

FIRST_NAME	SALARY
Steven	$24,000.00
Neena	$17,000.00
Lex	$17,000.00
Alexander	$9,000.00
Bruce	$6,000.00
David	$4,800.00

Les fonctions de conversion

Conversion explicite:fonctions TO_DATE / TO_NUMBER

Fonction	Description
TO_DATE (chaîne, format)	permet de convertir une chaîne de caractères en donnée de type date. Le format est identique à celui de la fonction TO_CHAR
TO_NUMBER (chaîne, format)	convertit une chaîne de caractères en nombre dans le format donné (quand la chaîne de caractères est composée de caractères numériques)

Exemple 1

select first_name, hire_date from emp
where hire_date>to_date('01/01/1982','DD-MM-YYYY');

FIRST_NAME	HIRE_DATE
Michael	17/02/96
Pat	17/08/97

Exemple 2

Select first_name, salary from employees
where salary>=to_number('15000');

FIRST_NAME	SALARY
Steven	24000
Neena	17000
Lex	17000

Autres fonctions

Fonction DECODE

- La fonction decodepermet de faire un traitement conditionnel sur les données :

Syntaxe

Decode (expr, val1, res1, val2, res2, ... ValN, resN, default)
retourne res1 si expr = val1, res2 si expr=val2,...,resN si expr=valN sinon default

Exemple

SELECT first_name,department_id, decode(department_id,10, 'ACCOUNTING', 20, 'RESEARCH', 'DEP. INCONNU ') AS "NOM DEPARTEMENT" FROM employees;

FIRST_NAME	DEPARTMENT_ID	NOM DEPARTEMENT
Jennifer	10	ACCOUNTING
Michael	20	RESEARCH
Pat	20	RESEARCH
Den	30	DEP. INCONNU
Alexander	30	DEP. INCONNU

Autres fonctions

Fonction NVL / NVL2

- NVL(expr,val): retourne val si expr est NULL
- NVL2(expr,val1,val2): retourne val1 si expr est NOT NULL et val2 si expr est NULL
- expr peut être de type date, les caractère et valeur numérique. Les types de données de expr et val doivent correspondre.

Exemple

```
SELECT first_name, salary, commission_pct,
NVL(commission_pct,0),
NVL(to_char(commission_pct), 'Pas de commission') AS "commission ?",
NVL2(commission_pct,commission_pct*salary,0) AS "commision",
to_char(NVL2(commission_pct,commission_pct*100/salary,0)) || '%' AS "pourcentage commission"
FROM employees;
```

FIRST_NAME	SALARY	COMMISSION_PCT	NVL(COMMISSION_PCT,0)	Commission ?	Commision	Pourcentage Commission
Steven	24000	-	0	Pas de commission	0	0%
Neena	17000	-	0	Pas de commission	0	0%
Lex	17000	-	0	Pas de commission	0	0%
John	14000	,4	,4	,4	5600	,002857142857142857142857142857142857143%
Karen	13500	,3	,3	,3	4050	,002222222222222222222222222222222222222%
Michael	13000	-	0	Pas de commission	0	0%

Autres fonctions

Fonction NULLIF

- NULLIF (expr1, expr2) : retourne NULL si expr1= expr2, sinon retourne expr1

Exemple

SELECT first_name, LENGTH(first_name) nbr1, last_name, LENGTH(last_name) nbr2, NULLIF(LENGTH(first_name), LENGTH(last_name)) result FROM employees;

FIRST_NAME	NBR1	LAST_NAME	NBR2	RESULT
Amit	4	Banda	5	4
Sarah	5	Bell	4	5
Britney	7	Everett	7	-
Daniel	6	Faviet	6	-

Autres fonctions

Fonction COALESCE

- Coalesce (exp1,expr2,expr3,. . .) : retourne la première valeur non nulle

Exemple

select Coalesce(NULL,1,NULL,7) from dual;
=> retourne 1

Autres fonctions

Fonction CASE

- La fonction case évalue une liste de conditions et retourne un résultat parmi les cas possibles

Syntaxe

1. case <expression>
   when <valeur1> then <resultat1>
   .
   .
   .
   when <valeurN> then <resultatN>
   else resultat
   end
2. case
   when <condition1> then <resultat1>
   .
   .
   .
   when <conditionN> then <resultatN>
   else resultat
   end

Autres fonctions

Fonction CASE

Exemple

```
SELECT first_name, department_id,
case department_id
when 10 then 'Accounting'
when 20 then 'RESEARCH'
else 'INCONNU'
end as departement
FROM employees;
```

FIRST_NAME	DEPARTMENT_ID	DEPARTEMENT
Jennifer	10	Accounting
Michael	20	RESEARCH
Pat	20	RESEARCH
Den	30	INCONNU
Alexander	30	INCONNU

Les fonctions analytiques

- Les fonctions analytiques calculent une valeur globale basée sur un groupe de lignes. Ils diffèrent des fonctions de groupe (ou d'agrégation) en ce qu'ils renvoient plusieurs lignes pour chaque groupe.
- Les fonctions analytiques sont la dernière série d'opérations effectuées dans une requête à l'exception de la clause finale ORDER BY. Par conséquent, elles analytiques ne peuvent apparaître que dans la liste de sélection ou clause ORDER BY.

Syntaxe d'une fonction analytique

fonction_analytique(expression) OVER([clause_partitionnement] [clause_ordre])

- clause_partitionnement: sous forme PARTITION BY expression1,expression2,...,expressionN : définit les groupes de partitionnement
- clause_ordre: sous forme ORDER BY expression1,expression2,...,expressionN [NULLSFIRST|LAST] : définit l'ordre à l'intérieur de chaque partition
  - NULLSFIRST/LAST: indique si les valeurs nulles seront en premier ordre/dernier ordre

Les fonctions analytiques

Fonction ROW_NUMBER()

- La fonction row_number() retourne le numéro séquentiel d'une ligne dans une partition de résultats, en commençant à 1 pour la première ligne de chaque partition.

Exemple 1

SELECT employee_id,department_id, salary, row_number() over(order by salary DESC) "N° Salaire" FROM employees ;

EMPLOYEE_ID	DEPARTMENT_ID	SALARY	N° Salaire
100	90	24000	1
101	90	17000	2
102	90	17000	3
145	80	14000	4
146	80	13500	5
201	20	13000	6
108	100	12000	7

Les fonctions analytiques

Fonction ROW_NUMBER()

Exemple 2

SELECT employee_id,department_id, salary, row_number() over(partition by department_id order by salary DESC) "Rang Salaire" FROM employees;

EMPLOYEE_ID	DEPARTMENT_ID	SALARY	N° Salaire
200	10	4400	1
201	20	13000	1
202	20	6000	2
114	30	12000	1
117	30	12000	2
115	30	3100	3
116	30	2900	4
118	30	2600	5
119	30	2500	6

Exemple 3

SELECT employee_id,department_id,job_id, salary, row_number() over(partition by department_id,job_id order by salary DESC) "Rang Salaire" FROM employees;

EMPLOYEE_ID	DEPARTMENT_ID	JOB_ID	SALARY	N° Salaire
200	10	AD_ASST	4400	1
201	20	MK_MAN	13000	1
202	20	MK_REP	6000	1
114	30	AC_MGR	12000	1
117	30	AC_MGR	12000	2
115	30	PU_CLERK	3100	1
116	30	PU_CLERK	2900	2
118	30	PU_CLERK	2600	3

Les fonctions analytiques

Fonction RANK()

- La fonction rank() retourne le rang dechaque ligne au sein de la partition d'un ensemble de résultats

Exemple

SELECT employee_id,department_id, salary, rank() over(partition by department_id order by salary DESC) "Rang Salaire" FROM employees;

EMPLOYEE_ID	DEPARTMENT_ID	SALARY	Rang Salaire
200	10	4400	1
201	20	13000	1
202	20	6000	2
114	30	12000	1
117	30	12000	1
115	30	3100	3
116	30	2900	4
118	30	2600	5
119	30	2500	6

Les fonctions analytiques

Fonction DENSE_RANK()

- La fonction dense_rank() retourne le rang des lignes à l'intérieur de la partition d'un ensemble de résultats,sans aucun vide dans le classement

Exemple

SELECT employee_id,department_id, salary, dense_rank() over(partition by department_id order by salary DESC) "Rang Salaire" FROM employees;

EMPLOYEE_ID	DEPARTMENT_ID	SALARY	Rang Salaire
200	10	4400	1
201	20	13000	1
202	20	6000	2
114	30	12000	1
117	30	12000	1
115	30	3100	2
116	30	2900	3
118	30	2600	4

Les fonctions analytiques

Fonction FIRST_VALUE()

- La fonction first_value() retourne la première valeur d'une partition

Exemple

SELECT employee_id,department_id, salary, first_value(salary) over(partition by department_id order by salary) as first_valeur from employees;

EMPLOYEE_ID	DEPARTMENT_ID	SALARY	FIRST_VALEUR
200	10	4400	4400
202	20	6000	6000
201	20	13000	6000
119	30	2500	2500
118	30	2600	2500
116	30	2900	2500
115	30	3100	2500
117	30	12000	2500
114	30	12000	2500

Les fonctions analytiques

Fonction LAST_VALUE()

- La fonction last_value() retourne la dernière valeur d'une partition

Exemple

SELECT employee_id,department_id, salary, last_value(salary) over(partition by department_id order by salary) as last_valeur from employees;

EMPLOYEE_ID	DEPARTMENT_ID	SALARY	LAST_VALEUR
200	10	4400	4400
202	20	6000	6000
201	20	13000	13000
119	30	2500	2500
118	30	2600	2600
116	30	2900	2900
115	30	3100	3100
117	30	12000	12000
114	30	12000	12000

Les fonctions multi-lignes

- Les fonctions multi-lignes (appelées aussi de groupe ou d'agrégation) opèrent sur des ensembles de lignes afin de renvoyer un seul résultat par groupe.
- Les fonctions de groupe les plus utilisées:
  - AVG([distinct | all] expr) : valeur moyenne en ignorant les valeurs NULL
  - COUNT ([* | distinct | all] expr) : nombre de lignes où expr est différente de NULL. Le caractère * comptabilise toutes les lignes sélectionnées
  - MAX ([distinct | all] expr) : valeur maximale en ignorant les valeurs NULL
  - MIN([distinct | all] expr) : valeur minimale en ignorant les valeurs NULL
  - STDDEV([distinct | all] expr) : ecart-type en ignorant les valeurs NULL
  - SUM([distinct | all] expr) : somme en ignorant les valeurs NULL
  - VARIANCE([distinct | all] expr) : variance en ignorant les valeurs NULL

Les fonctions multi-lignes

Syntaxe

SELECT [colonne,] fonction_groupe(colonne), ...
FROM <nom_table>
[WHERE <condition>]
[GROUP BY colonne]
[ORDER BY colonne];

Exemple 1

SELECT trunc(AVG(salary),3), SUM(salary), MAX(hire_date), MIN(hire_date)
FROM employees
WHERE department_id in(80,90);

TRUNC(AVG(SALARY),3)	SUM(SALARY)	MAX(HIRE_DATE)	MIN(HIRE_DATE)
9797,297	362500	21/04/00	17/06/87

Les fonctions multi-lignes

Exemple 2

```
SELECT COUNT(*)
FROM employees
WHERE department_id in(80,90);
```

COUNT(*)
37

=>retourne le nombre de lignes qui vérifient la condition de la clause WHERE

Exemple 3

```
SELECT COUNT(commission_pct) "count", COUNT(all commission_pct) "all",
COUNT(DISTINCT commission_pct) "distinct"
FROM employees
WHERE department_id in(80,90);
```

Count	All	Distinct
34	34	7

=>COUNT(expr) /COUNT(all expr): retourne le nombre de ligne ayant des valeurs non NULL de expr
=>COUNT(distinct expr): retourne le nombre de ligne ayant des valeurs non NULL et DISTINCT de expr

Les fonctions multi-lignes

Remarque

- Les fonctions de groupe ignorent les valeurs NULL de la colonne

Exemple:

SELECT trunc(AVG(commission_pct) ,3) FROM employees;

TRUNC(AVG(COMMISSION_PCT),3)
,222

- La fonction NVL force les fonctions de groupe à inclure les valeurs NULL

Exemple:

SELECT trunc(AVG(NVL(commission_pct,0)) ,3) FROM employees;

TRUNC(AVG(NVL(COMMISSION_PCT,0)),3)
,072

Les fonctions multi-lignes

La clause GROUP BY

Exemple 1

SELECT department_id, trunc(AVG(salary),3)
FROM employees WHERE department_id in(80,90)
GROUP BY department_id;

DEPARTMENT_ID	TRUNC(AVG(SALARY),3)
90	19333,333
80	8955,882

Notez Bien

Toute colonne ou expression de la liste SELECT qui ne constitue pas une fonction d'agrégation doit figurer dans la clause GROUP BY

Exemple:

SELECT department_id, job_id, trunc(AVG(salary),3)
FROM employees WHERE department_id in(80,90)
GROUP BY department_id, job_id

DEPARTMENT_ID	JOB_ID	TRUNC(AVG(SALARY),3)
80	SA_MAN	12200
80	SA_REP	8396,551
90	AD_PRES	24000
90	AD_VP	17000

Les fonctions multi-lignes

La clause HAVING

- La clause HAVING permet de restreindre l'affichage des résultats de groupes à ceux qui vérifient la condition dans cette clause
  - Les lignes sont regroupées
  - La fonction de groupe est appliquée
  - Les groupes qui correspondent à la clause HAVING sont affichés

Syntaxe

```
SELECT [colonne,] fonction_groupe(colonne), ...
FROM <nom_table>
[WHERE <condition>]
[GROUP BY colonne]
[HAVING <condition_groupe>]
[ORDER BY colonne];
```

Les fonctions multi-lignes

La clause HAVING

Exemple 1

```
SELECT department_id, MAX(salary)
FROM employees
WHERE job_id LIKE '%REP'
GROUP BY department_id
ORDER BY department_id;
```

DEPARTMENT_ID	MAX(SALARY)
20	6000
40	6500
70	10000
80	11500
-	7000

Exemple 2

```
SELECT department_id, MAX(salary)
FROM employees
WHERE job_id LIKE '%REP'
GROUP BY department_id
HAVING MAX(salary)>=10000
ORDER BY department_id;
```

DEPARTMENT_ID	MAX(SALARY)
70	10000
80	11500

Les fonctions multi-lignes

L'opérateur ROLLUP

- L'opérateur ROLLUP : calcule des agrégats (SUM, COUNT, MAX, MIN, AVG) à tous les niveaux de totalisation sur une hiérarchie de dimensions et calcule le total général selon l'ordre de gauche à droite dans la clause GROUP BY
- S'il y a n colonnes de regroupements, GROUP BY ROLLUP génère n+1 niveaux de totalisation
- ROLLUP (a, b, c)
  - (a, b, c)
  - (a, b)
  - (a)
  - ()

Les fonctions multi-lignes

L'opérateur ROLLUP

Exemple

```
SELECT Department_id, JOB_id,manager_id, SUM (SALARY)
FROM EMPLOYEES
WHERE DEPARTMENT_ID in (10,20,30)
GROUP BY Rollup(Department_id, JOB_id,manager_id);
```

DEPARTMENT_ID	JOB_ID	MANAGER_ID	SUM(SALARY)
10	AD_ASST	101	4400
10	AD_ASST	-	4400
10	-	-	4400
20	MK_MAN	100	13000
20	MK_MAN	-	13000
20	MK_REP	201	6000
20	MK_REP	-	6000
20	-	-	19000
30	AC_MGR	100	12000
30	AC_MGR	114	12000
30	AC_MGR	-	24000
30	PU_CLERK	114	11100
30	PU_CLERK	-	11100
30	-	-	35100
-	-	-	58500

Les fonctions multi-lignes

L'opérateur CUBE

- L'opérateur CUBE : calcule des agrégats (SUM, COUNT, MAX, MIN, AVG) à différents niveaux d'agrégation comme ROLLUP mais de plus permet de calculer toutes les combinaisons d'agrégations :
- L'opérateur CUBE: calcule des sous-totaux pour toutes les combinaisons possibles d'un ensemble de colonnes de regroupement
- Si la clause CUBE contient n colonnes, CUBE calcule 2^n combinaisons de totaux
- CUBE (a, b, c)
  - (a, b, c)
  - (a, b)
  - (a, c)
  - (a)
  - (b, c)
  - (b)
  - (c)
  - ()

Les fonctions multi-lignes

L'opérateur CUBE

Exemple

```
SELECT Department_id, JOB_id, SUM (SALARY)
FROM EMPLOYEES
WHERE DEPARTMENT_ID in (10,20,30)
GROUP BY Cube(Department_id, JOB_id);
```

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
-	-	58500
-	AC_MGR	24000
-	MK_MAN	13000
-	MK_REP	6000
-	AD_ASST	4400
-	PU_CLERK	11100
10	-	4400
10	AD_ASST	4400
20	-	19000
20	MK_MAN	13000
20	MK_REP	6000
30	-	35100
30	AC_MGR	24000
30	PU_CLERK	11100

Les fonctions multi-lignes

La fonction GROUPING

- Les lignes de totaux correspondent génaralement aux lignes ayant des valeurs NULL
  =>possibilité de confusion si les lignes contiennent déja des valeurs NULL !
- La fonction GROUPING permet d'éliminer cette ambiguité. Elle accepte une seule colonne comme paramètre et retourne:
  - 1 si la colonne contient une valeur null généré dans le cadre d'un sous-total par un ROLLUP ou CUBE
  - 0 pour une autre valeur, y compris les valeurs NULL stockées

Les fonctions multi-lignes

La fonction GROUPING

Exemple

SELECT Department_id, JOB_id, SUM (SALARY),
GROUPING(Department_id), GROUPING(JOB_id)
FROM EMPLOYEES
WHERE DEPARTMENT_ID in (10,20,30)
GROUP BY Cube(Department_id, JOB_id);

DEPARTMENT_ID	JOB_ID	SUM(SALARY)	GROUPING(DEPARTMENT_ID)	GROUPING(JOB_ID)
-	-	58500	1	1
-	AC_MGR	24000	1	0
-	MK_MAN	13000	1	0
-	MK_REP	6000	1	0
-	AD_ASST	4400	1	0
-	PU_CLERK	11100	1	0
10	-	4400	0	1
10	AD_ASST	4400	0	0
20	-	19000	0	1
20	MK_MAN	13000	0	0
20	MK_REP	6000	0	0
30	-	35100	0	1
30	AC_MGR	24000	0	0
30	PU_CLERK	11100	0	0

Plan

Les sous intérrogations

Interrogation principale:

Quels employés ont un salaire plus élevé que celui d'Abel?

Sous-interrogation:

Quel est le salaire d'Abel?

Les sous intérrogations

Syntaxe

SELECT <colonne1>[, <colonne2>,..., <colonneN>]
FROM <nom_table>
WHERE <expression> OPERATEUR (SELECT SELECT <colonne1>[, <colonne2>,...,
<colonneN>] FROM <nom_table>)

- Mettre les sous-interrogations entre parenthèses
- La clause order by de la sous-interrogation n'est pas nécessaire
- Utilisez des opérateurs de comparaison monolignes avec les sous-interrogations monolignes, et des opérateurs de comparaison multilignes avec les sous interrogations multilignes
  - opérateurs mono-ligne (>,>= ,<, <=, . . .)
  - opérateurs multi-lignes (IN, ALL, ANY)
- La sous-interrogation (requête interne) est exécutée une seule fois avant la requête principale
- Le résultat de la sous-interrogation est utilisé par la requête principale (requête externe)
- Une sous-interrogation est utilisée dans les clauses suivantes :
  - WHERE
  - HAVING
  - FROM

Les sous intérrogations monoligne

- Renvoient une seule ligne
- Utilisent des operateurs de comparaison monolignes (= , >, >= , < , <= , <>)

Exemple 1

```
SELECT last_name, job_id, salary
FROM employees
WHERE job_id =( SELECT job_id FROM employees WHERE employee_id=124 );
```

LAST_NAME	JOB_ID	SALARY
Weiss	ST_MAN	8000
Fripp	ST_MAN	8200
Kaufling	ST_MAN	7900
Vollman	ST_MAN	6500
Mourgos	ST_MAN	5800

Les sous intérrogations monoligne

Exemple 2

```
SELECT last_name, job_id, salary
FROM employees
WHERE salary =(SELECT max(salary) FROM employees )
```

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000

Exemple 3

```
SELECT department_id, min(salary)
FROM employees
GROUP BY department_id
HAVING min(salary) > (SELECT MIN(salary) FROM employees WHERE department_id= 20)
```

DEPARTMENT_ID	MIN(SALARY)
100	6900
-	7000
90	17000
70	10000
110	8300
80	6100
40	6500

Les sous intérrogations multi-lignes

- Renvoient plusieurs lignes
- Utilisent des opérateurs de comparaison multiligne (IN, ANY, ALL)

Exemple 1

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary > ANY (SELECT salary FROM employees WHERE job_id = 'SA_MAN')
AND job_id <> 'SA_MAN';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
100	King	AD_PRES	24000
102	De Haan	AD_VP	17000
101	Kochhar	AD_VP	17000
201	Hartstein	MK_MAN	13000
205	Higgins	AC_MGR	12000
114	Raphaely	AC_MGR	12000
108	Greenberg	FI_MGR	12000
117	Tobias	AC_MGR	12000
168	Ozer	SA_REP	11500
174	Abel	SA_REP	11000

Les sous intérrogations multi-lignes

Exemple 2

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary > ALL (SELECT salary FROM employees WHERE job_id = 'SA_MAN')
AND job_id <> 'SA_MAN';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
100	King	AD_PRES	24000
101	Kochhar	AD_VP	17000
102	De Haan	AD_VP	17000

Plan

Les jointures

- Une jointure permet d'extraire des données à partir de plusieurs tables (et / ou vues) en utilisant des conditions de jointure
- La condition de jointure peut être exprimée:
  - dans la clause WHERE: WHERE table1.C1=table2.C1
  - dans la clause ON: ON table1.C1=table2.C1
- Précéder le nom de la colonne par le nom de la table lorsque nom de la colonne figure dans plusieurs tables
- Il existe deu types de jointure:
  - jointure interne
  - jointure externe
- Dans ce qui suit on utilisera les tables tab1 et tab2 suivantes pour les exemples:

A	B	C
1	a1	2
2	a2	3
3	a3	4
4	a4	5
5	a5	6
6	a6	7
7	a7	8

A	B	C
3	a3	4
4	a4	5
5	a5	6
6	a6	7
7	a7	8
8	a8	9
9	a9	10

Jointure interne

- Une jointure interne (appelé aussi jointure simple) est une jointure de deux tables ou plus qui retourne uniquement les lignes qui satisfont la condition de jointure

Syntaxe

SELECT T1.colonne1, . . . ,T1.colonneN,T2.colonne1, . . . ,T2.colonneM
FROM T1 [INNER] JOIN T2
ON <condition_jointure>
WHERE <condition>

Exemple

SELECT tab1.a, tab1.b, tab2.a,tab2.b
FROM tab1 INNER JOIN tab2
ON tab1.a=tab2.a and tab1.b=tab2.b;

A	B	A	B
3	a3	3	a3
4	a4	4	a4
5	a5	5	a5
6	a6	6	a6
7	a7	7	a7

Jointure externe

- Une jointure externe étend le résultat d'une jointure interne
- Une jointure externe renvoie toutes les lignes qui satisfont la condition de jointure et renvoie également une partie ou l'ensemble des lignes d'une table pour lesquelles aucune ligne de l'autre table satisfait la condition de jointure. Il existe 3 types de jointures externe:
  - jointure externe gauche: jointure entre A et B => afficher les lignes de A qui ne satisfont pas la condition de jointure
  - jointure externe droite: jointure entre A et B => afficher les lignes de B qui ne satisfont pas la condition de jointure
  - jointure externe complète: jointure entre A et B => afficher les lignes de A et B qui ne satisfont pas la condition de jointure

Syntaxe

SELECT T1.colonne1, . . . ,T1.colonneN,T2.colonne1, . . . ,T2.colonneM
FROM T1 {LEFT | RIGHT | FULL} [OUTER] JOIN T2
ON <condition_jointure>
WHERE <condition>

Jointure externe

Exemple 1:
SELECT tab1.a, tab1.b, tab2.a,tab2.b
FROM tab1 LEFT OUTER JOIN tab2
ON tab1.a=tab2.a and tab1.b=tab2.b;

A	B	A	B
3	a3	3	a3
4	a4	4	a4
5	a5	5	a5
6	a6	6	a6
7	a7	7	a7
2	a2	-	-
1	a1	-	-

Exemple 2:
SELECT tab1.a, tab1.b, tab2.a,tab2.b
FROM tab1 RIGHT OUTER JOIN tab2
ON tab1.a=tab2.a and tab1.b=tab2.b;

A	B	A	B
3	a3	3	a3
4	a4	4	a4
5	a5	5	a5
6	a6	6	a6
7	a7	7	a7
-	-	9	a9
-	-	8	a8

Exemple 3:
SELECT tab1.a, tab1.b, tab2.a,tab2.b
FROM tab1 FULL OUTER JOIN tab2
ON tab1.a=tab2.a and tab1.b=tab2.b;

A	B	A	B
3	a3	3	a3
4	a4	4	a4
5	a5	5	a5
6	a6	6	a6
7	a7	7	a7
2	a2	-	-
1	a1	-	-
-	-	8	a8
-	-	9	a9

Equijointure / Non-equijointure

- Une équijointure est une jointure avec une condition de jointure contenant un opérateur d'égalité (= , LIKE , etc)

Exemple 1:
SELECT tab1.a, tab1.b, tab2.a,tab2.b
FROM tab1 LEFT OUTER JOIN tab2
ON tab1.a=tab2.a;

A	B	A	B
3	a3	3	a3
4	a4	4	a4
5	a5	5	a5
6	a6	6	a6
7	a7	7	a7
2	a2	-	-
1	a1	-	-

- Une non-équijointure est une jointure avec une condition de jointure contenant un opérateur d'inégalité (< ,<= , > , >= , BETWEEN , etc)

Exemple 2:
SELECT tab1.a, tab2.a, tab2.c
FROM tab1 INNER JOIN tab2 $
ON tab1.a>=tab2.a and tab1.a<=tab2.c;

A	A	C
7	7	8
7	6	7
6	6	7
6	5	6
5	5	6
5	4	5
4	4	5
4	3	4
3	3	4

Auto-jointure

- Une auto-jointure est une jointure d'une table avec elle-même (réflexivité)

Exemple

```
SELECT t1.a, t2.b, t2.c
FROM tab1 t1 INNER JOIN tab1 t2
ON t2.c=t1.a
```

A	B	C
2	a1	2
3	a2	3
4	a3	4
5	a4	5
6	a5	6
7	a6	7

Jointure naturelle

- Une jointure naturelle est basée sur toutes les colonnes des deux tables portant le même nom
- Elle sélectionne les lignes des deux tables dont les valeurs sont identiques dans toutes les colonnes qui correspondent
- Si les colonnes portant le même nom présentent des types de données différents, une erreur est renvoyée

Exemple

```
SELECT department_id, department_name, location_id, city
FROM departments NATURAL JOIN locations
WHERE location_id <>1700;
                =
SELECT department_id, department_name, location_id, city
FROM departments INNER JOIN locations
USING (location_id)
WHERE location_id <>1700;
                =
SELECT department_id, department_name, departments.location_id, city
FROM departments INNER JOIN locations
ON departments.location_id = locations.location_id
WHERE departments.location_id <>1700;
```

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
20	Marketing	1800	Toronto
40	Human Resources	2400	London
50	Shipping	1500	South San Francisco
60	IT	1400	Southlake
70	Public Relations	2700	Munich
80	Sales	2500	Oxford

Produit cartésien

- On obtient un produit cartésien lorsque :
  - Une condition de jointure est omise
  - Une condition de jointure est incorrecte
- A chaque ligne de la table 1 sont jointes toutes les lignes de la table 2
- Le nombre de lignes renvoyées est égal n1*n2 où n1 est le nombre de lignes de la table 1 n2 est le nombre de lignes de la table 2

Exemple

```
SELECT tab1.a, tab2.b
FROM tab1 CROSS JOIN tab2
WHERE tab1.a IN(1,2) ;
```

A	B
1	a5
1	a6
1	a7
2	a5
2	a6
2	a7

Plan

Les opérateurs ensemblistes

OPERATEUR	DESCRIPTION
INTERSECT	Ramène toutes les lignes communes aux deux requêtes
UNION	Toutes les lignes distinctes ramenées par les deux requêtes
UNION ALL	Toutes les lignes ramenées par les deux requêtes y compris les doublons
MINUS	Toutes les lignes ramenées par la première requête sauf les lignes ramenées par la seconde requête

L'opérateur UNION

- Le nombre de colonnes et le type des colonnes doivent être identiques dans les 2 ordres SELECT
- L'opérateur UNION intervient sur toutes les colonnes

Exemple 1

SELECT employee_id, job_id, salary FROM employees
WHERE job_id IN('AC_MGR','AD_VP')ORDER BY employee_id;

EMPLOYEE_ID	JOB_ID	SALARY
101	AD_VP	17000
102	AD_VP	17000
114	AC_MGR	12000
117	AC_MGR	12000
205	AC_MGR	12000
298	AD_VP	6000

Exemple 2

SELECT employee_id, job_id, salary FROM employees
WHERE salary >= 15000 ORDER BY employee_id;

EMPLOYEE_ID	JOB_ID	SALARY
100	AD_PRES	24000
101	AD_VP	17000
102	AD_VP	17000

L'opérateur UNION

Exemple 3

SELECT employee_id, job_id, salary FROM employees WHERE job_id IN('AC_MGR','AD_VP')

UNION

SELECT employee_id, job_id, salary FROM employees WHERE salary >= 15000
ORDER BY employee_id;

EMPLOYEE_ID	JOB_ID	SALARY
100	AD_PRES	24000
101	AD_VP	17000
102	AD_VP	17000
114	AC_MGR	12000
117	AC_MGR	12000
205	AC_MGR	12000
298	AD_VP	6000

L'opérateur UNION ALL

- Renvoie le même résultat que UNION sans élimination des doublons

Exemple

SELECT employee_id, job_id, salary FROM employees WHERE job_id IN('AC_MGR','AD_VP')

UNION ALL

SELECT employee_id, job_id, salary FROM employees WHERE salary >= 15000
ORDER BY employee_id;

EMPLOYEE_ID	JOB_ID	SALARY
100	AD_PRES	24000
101	AD_VP	17000
101	AD_VP	17000
102	AD_VP	17000
102	AD_VP	17000
114	AC_MGR	12000
117	AC_MGR	12000
205	AC_MGR	12000
298	AD_VP	6000

L'opérateur INTERSECT

Exemple

SELECT employee_id, job_id, salary FROM employees WHERE job_id IN('AC_MGR','AD_VP')

INTERSECT

SELECT employee_id, job_id, salary FROM employees WHERE salary >= 15000
ORDER BY employee_id;

EMPLOYEE_ID	JOB_ID	SALARY
101	AD_VP	17000
102	AD_VP	17000

L'opérateur MINUS

Exemple 1

```
SELECT employee_id, job_id, salary
FROM employees
WHERE job_id IN('AC_MGR','AD_VP')

MINUS

SELECT employee_id, job_id, salary
FROM employees
WHERE salary >= 15000
ORDER BY employee_id;
```

EMPLOYEE_ID	JOB_ID	SALARY
114	AC_MGR	12000
117	AC_MGR	12000
205	AC_MGR	12000
298	AD_VP	6000

Exemple 2

```
SELECT employee_id, job_id, salary
FROM employees
WHERE salary >= 15000

MINUS

SELECT employee_id, job_id, salary
FROM employees
WHERE job_id IN('AC_MGR','AD_VP')
ORDER BY employee_id;
```

EMPLOYEE_ID	JOB_ID	SALARY
100	AD_PRES	24000