

Le langage SQL (STRUCTURED QUERY LANGUAGE)

SQL est le langage consacré aux SGBD relationnels.

Il permet :

La définition des données (data definition language, DDL) :
création et la modification de la structure des données

La manipulation des données (data manipulation language, DML) : interrogation, ajout, mise à jour et suppression de données

La gestion des transactions (transactional control language, TCL) : validation ou annulation des transactions

La gestion du contrôle d'accès (data control language DCL) :
la définition de comptes d'accès et de droits associés

CREATE TABLE

```
CREATE TABLE employees
( employee_id      NUMBER(6)
  CONSTRAINT emp_employee_id PRIMARY KEY
, first_name       VARCHAR2(20)
, last_name        VARCHAR2(25)
  CONSTRAINT emp_last_name_nn NOT NULL
, email            VARCHAR2(25)
  CONSTRAINT emp_email_nn    NOT NULL
  CONSTRAINT emp_email_uk    UNIQUE
, phone_number     VARCHAR2(20)
, hire_date        DATE
  CONSTRAINT emp_hire_date_nn NOT NULL
, job_id           VARCHAR2(10)
  CONSTRAINT emp_job_nn      NOT NULL
, salary           NUMBER(8,2)
  CONSTRAINT emp_salary_ck   CHECK (salary>0)
, commission_pct   NUMBER(2,2)
, manager_id       NUMBER(6)
  CONSTRAINT emp_manager_fk REFERENCES
employees (employee_id)
, department_id    NUMBER(4)
  CONSTRAINT emp_dept_fk     REFERENCES
departments (department_id));
```

Contraintes définies au niveau table

```
CREATE TABLE employees(  
    employee_id      NUMBER(6) ,  
    last_name        VARCHAR2(25) NOT  
NULL,  
    email            VARCHAR2(25) ,  
    salary            NUMBER(8,2) ,  
    commission_pct   NUMBER(2,2) ,  
    hire_date        DATE NOT NULL,  
    ...  
    department_id    NUMBER(4) ,  
    CONSTRAINT emp_dept_fk FOREIGN KEY  
(department_id)  
    REFERENCES  
departments(department_id) ,  
    CONSTRAINT emp_email_uk  
UNIQUE(email)) ;
```

Créer une table à l'aide d'une sous-interrogation

```
CREATE TABLE dept80
AS
SELECT  employee_id, last_name,
        salary*12 ANNSAL,
        hire_date
FROM    employees
WHERE   department_id = 80;
```

```
CREATE TABLE succeeded.
```

```
DESCRIBE dept80
```

Name	Null	Type
-----	-----	-----
EMPLOYEE_ID		NUMBER(6)
LAST_NAME	NOT NULL	VARCHAR2(25)
ANNSAL		NUMBER
HIRE_DATE	NOT NULL	DATE

Insérer de nouvelles lignes

- Insérez une nouvelle ligne contenant des valeurs pour chaque colonne.
- Enumérez les valeurs dans l'ordre par défaut des colonnes de la table.
- Enumérez éventuellement les colonnes indiquées dans la clause `INSERT`.

```
INSERT INTO departments(department_id,  
                        department_name, manager_id, location_id)  
VALUES (70, 'Public Relations', 100, 1700);
```

```
1 rows inserted
```



- Placez les valeurs de type caractère et date entre apostrophes.

Insérer des lignes comprenant des valeurs NULL

- Méthode implicite : Omettre la colonne dans la liste.

```
INSERT INTO departments (department_id,  
                          department_name)  
VALUES (30, 'Purchasing');  
1 rows inserted
```

- Méthode explicite : Indiquer le mot-clé NULL dans la clause VALUES.

```
INSERT INTO departments    
1 rows inserted (100, 'Finance', NULL, NULL);
```

Copier des lignes depuis une autre table

- Ecrivez l'instruction `INSERT` avec une sous-interrogation :

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
SELECT employee_id, last_name, salary, commission_pct
FROM employees
WHERE job_id LIKE '%REP%';
```

4 rows inserted

- N'utilisez pas la clause `VALUES`.
- Le nombre de colonnes de la clause `INSERT` doit correspondre à celui de la sous-interrogation.
- Insérez toutes les lignes renvoyées par la sous-interrogation dans la table `sales_reps`.

Mettre à jour des lignes d'une table

- Si vous indiquez la clause `WHERE`, seules les valeurs d'une ou plusieurs lignes spécifiques sont modifiées :

```
UPDATE employees
SET    department_id = 50
WHERE  employee_id = 113;
```

1 rows updated

- Si vous omettez la clause `WHERE`, les valeurs de toutes les lignes de la table sont modifiées :

```
UPDATE    copy_emp
SET       department_id = 110;
```

22 rows updated

- Introduire la valeur d'une colonne par `NULL`.

Supprimer des lignes d'une table

- Si vous indiquez la clause `WHERE`, des lignes spécifiques sont supprimées :

```
DELETE FROM departments  
WHERE department_name = 'Finance';
```

```
1 rows deleted
```

- Si vous omettez la clause `WHERE`, toutes les lignes de la table sont supprimées :

```
DELETE FROM copy_emp;
```

```
22 rows deleted
```

Transactions de base de données

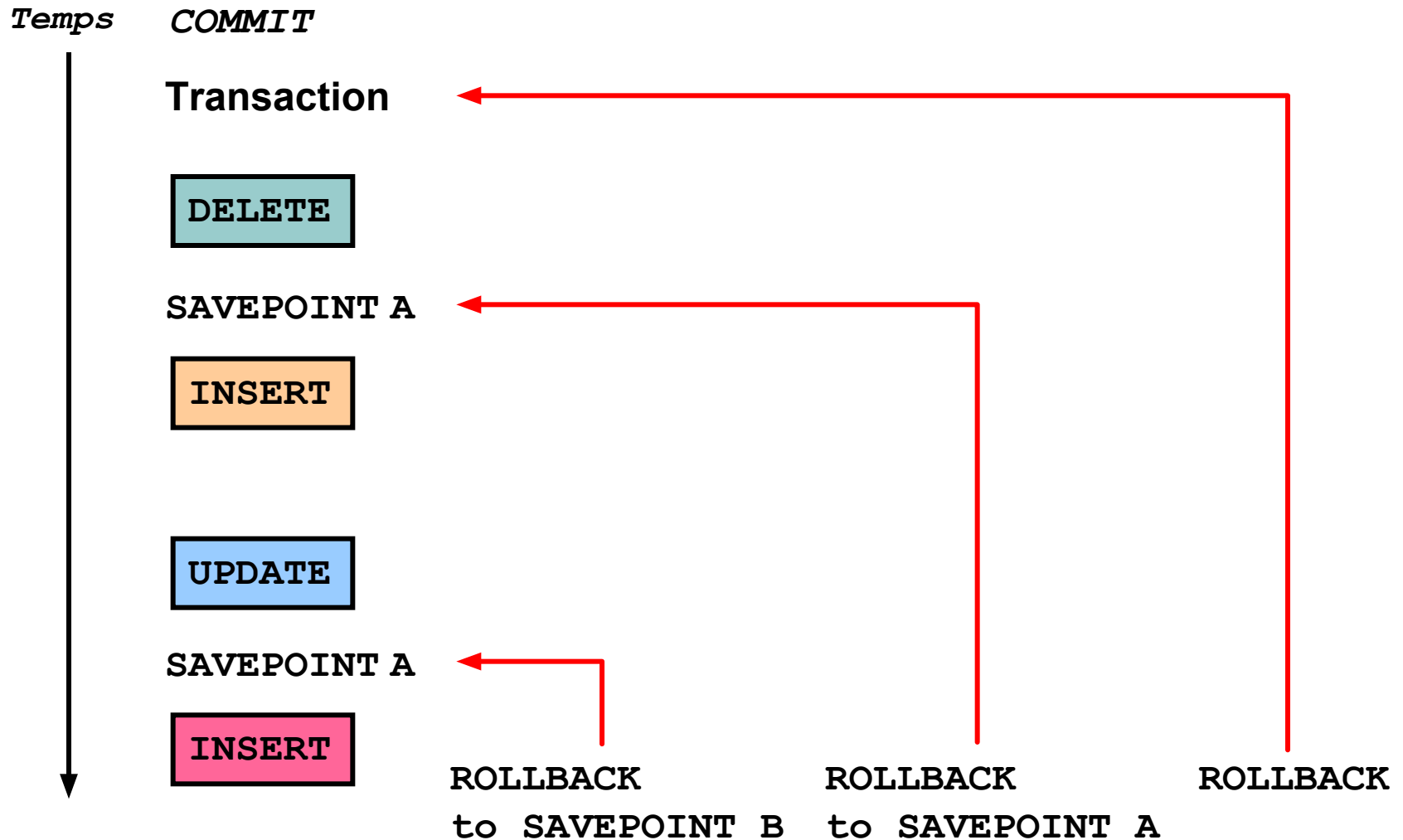
Une transaction de base de données se compose des éléments suivants :

- Des instructions LMD qui apportent une modification cohérente aux données (Insert, Update, Delete)
- Une instruction LDD (Create table, Create index...)
- Une instruction LCD (langage de contrôle de données: Rollback, Commit)

Transactions de base de données : Début et fin

- Elles commencent avec l'exécution de la première instruction SQL LMD.
- Elles finissent lorsque l'un des événements suivants se produit :
 - Une instruction COMMIT ou ROLLBACK est exécutée.
 - Une instruction LDD ou LCD est exécutée (validation automatique).
 - L'utilisateur quitte SQL Developer ou SQL*Plus.
 - Le système connaît une défaillance.

Instructions explicites de contrôle des transactions



Etat des données avant exécution de l'instruction COMMIT ou ROLLBACK

- L'état antérieur des données peut être récupéré.
- L'utilisateur actuel peut visualiser les résultats des opérations LMD à l'aide de l'instruction `SELECT`.
- Les autres utilisateurs *ne peuvent pas* afficher les résultats des instructions LMD exécutées par l'utilisateur actuel.
- Les lignes affectées sont *verrouillées*. Les autres utilisateurs ne peuvent donc pas modifier les données de ces lignes.

Extraire les lignes d'une table

Sélectionner toutes les colonnes

```
SELECT *  
FROM departments;
```

Afficher les tables appartenant à l'utilisateur courant

```
select * from tab
```

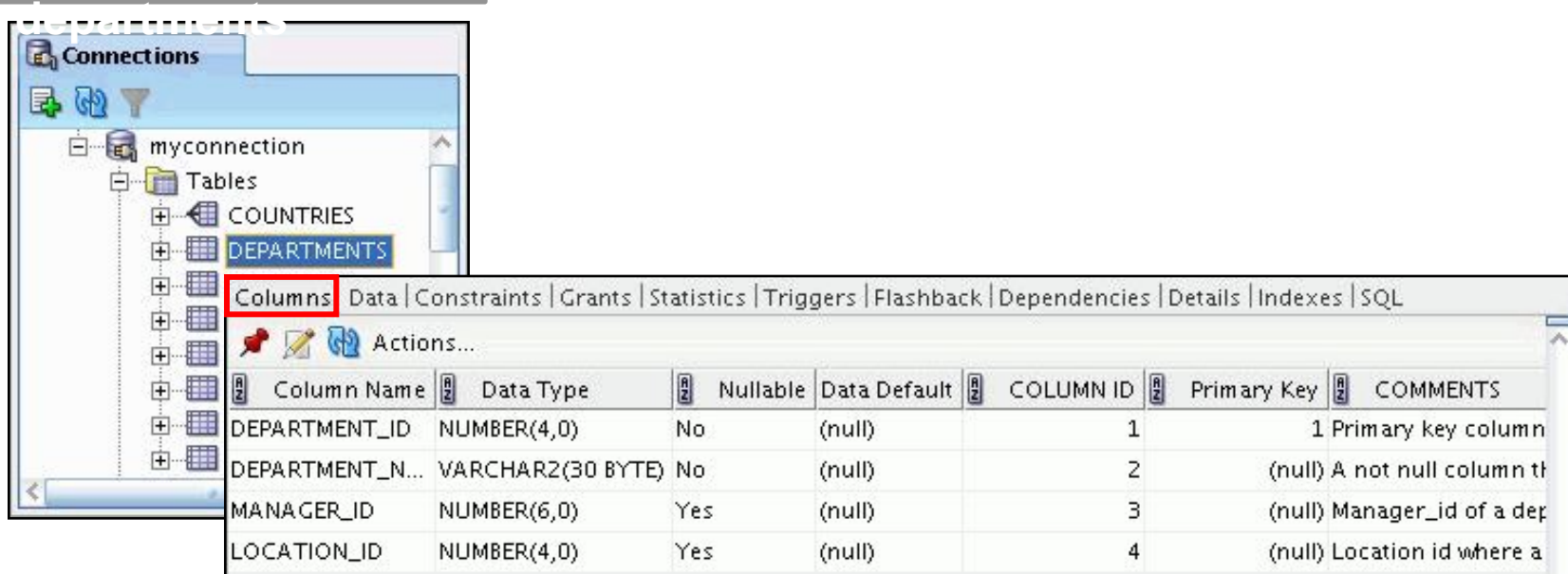
Sélectionner des colonnes spécifiques

```
SELECT department_id, location_id  
FROM departments;
```

Afficher la structure d'une table

- Utilisez la commande `DESCRIBE` pour afficher la structure d'une table.
- Vous pouvez aussi sélectionner la table dans l'arborescence Connections et cliquer sur l'onglet Columns.

DESCRIBE



Column Name	Data Type	Nullable	Data Default	COLUMN ID	Primary Key	COMMENTS
DEPARTMENT_ID	NUMBER(4,0)	No	(null)	1	1	Primary key column
DEPARTMENT_N...	VARCHAR2(30 BYTE)	No	(null)	2	(null)	A not null column th
MANAGER_ID	NUMBER(6,0)	Yes	(null)	3	(null)	Manager_id of a dep
LOCATION_ID	NUMBER(4,0)	Yes	(null)	4	(null)	Location id where a

Lignes en double

Par défaut, pour les interrogations, toutes les lignes sont affichées (y compris celles en double).

1

```
SELECT department_id  
FROM employees;
```

	DEPARTMENT_ID
1	10
2	20
3	20
4	110
5	110

...

2

```
SELECT DISTINCT department_id  
FROM employees;
```

	DEPARTMENT_ID
1	(null)
2	20
3	90
4	110
5	50
6	80
7	10
8	60

Restreindre les

```
SELECT last_name, salary
FROM employees
WHERE salary <= 3000 ;
```

```
SELECT last_name, salary
FROM employees
WHERE salary BETWEEN 2500 AND 3500 ;
```

```
SELECT employee_id, last_name, salary, manager_id
FROM employees
WHERE manager_id IN (100, 101, 201) ;
```

```
SELECT last_name
FROM employees
WHERE last_name LIKE '_o%' ;
```

```
SELECT employee_id, first_name || ' ' || last_name, salary*12 "Salaire
annuel"
FROM employees
WHERE first_name LIKE 'A%'
```

Restreindre les données

Rechercher les valeurs

NULL

```
SELECT last_name, manager_id
FROM   employees
WHERE  manager_id IS NULL
ORDER BY last_name
```

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

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

Tri

```
SELECT last_name, job_id, department_id,  
hire_date  
FROM employees  
ORDER BY hire_date ;
```

Tri selon une colonne repérée par sa position dans la clause

```
SELECT last_name, job_id, department_id,  
hire_date  
FROM employees  
ORDER BY 3;
```

Tri

```
SELECT last_name, job_id, department_id,  
hire_date  
FROM employees  
ORDER BY hire_date DESC ;
```

Tri par alias de

```

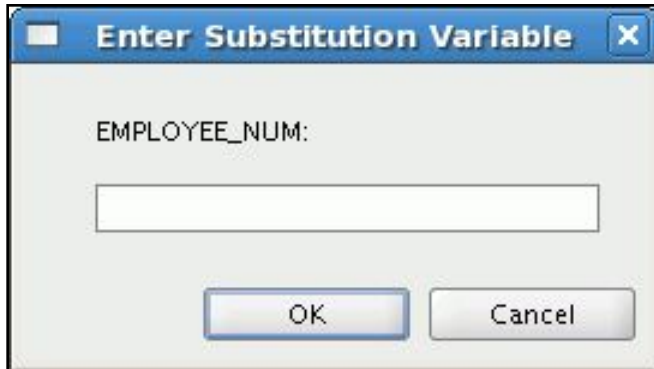
-- colonne
SELECT employee_id, last_name, salary*12 annsal
FROM employees
ORDER BY annsal ;

```

Utiliser une variable de substitution avec esperluette simple

Utilisez une variable comprenant une esperluette d'interprétation (&) comme préfixe pour inviter l'utilisateur à entrer une valeur :

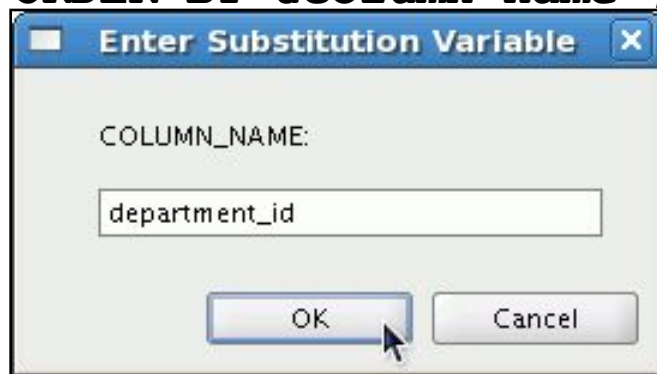
```
SELECT employee_id, last_name, salary,  
department_id  
FROM employees  
WHERE employee_id = &employee_num ;
```



Utiliser une variable de substitution avec esperluette double

Utilisez l'esperluette d'interprétation double (&&) si vous souhaitez réutiliser la valeur de la variable sans solliciter l'utilisateur à chaque fois :

```
SELECT employee_id, last_name, job_id,   
&&column_name  
FROM   
ORDER BY &column_name;
```



	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	200	Whalen	AD_ASST	10
2	201	Hartstein	MK_MAN	20
3	202	Fay	MK_REP	20

...

Utiliser les fonctions de conversion de casse

Affichez le numéro d'employé, le nom et le numéro de département de l'employé Higgins :

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  last_name = 'higgins';
```

0 rows selected

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  LOWER(last_name) = 'higgins';
```


	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	205	Higgins	110

Utiliser la fonction SYSDATE

SYSDATE est une fonction qui renvoie :

- la date
- l'heure

```
SELECT sysdate  
FROM dual;
```

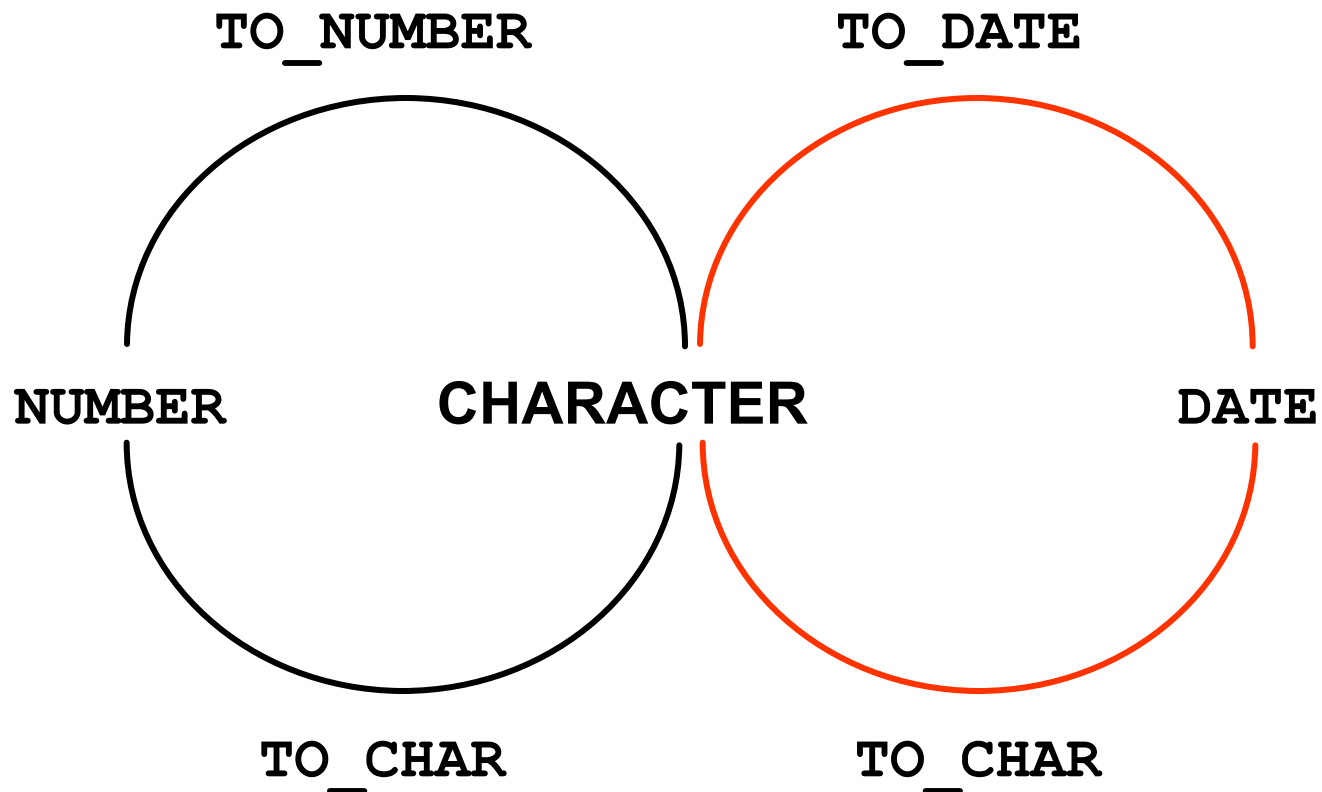
	 SYSDATE
1	10-JUN-09

Utiliser les fonctions ROUND et TRUNC avec des dates

Hypothèse : SYSDATE = '25-JUL-03' :

Fonction	Résultat
ROUND (SYSDATE, 'MONTH')	01-AUG-03
ROUND (SYSDATE , 'YEAR')	01-JAN-04
TRUNC (SYSDATE , 'MONTH')	01-JUL-03
TRUNC (SYSDATE , 'YEAR')	01-JAN-03

Fonctions de conversion





Fonctions de conversion

```
SELECT TO_CHAR(salary, '$99,999.00') SALARY
FROM   employees
WHERE  last_name = 'Ernst';
```

	 SALARY
1	\$6,000.00

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')
FROM   employees;
```

	 LAST_NAME	 TO_CHAR(HIRE_DATE,'DD-MON-YYYY')
1	Whalen	17-Sep-1987
2	King	17-Jun-1987
3	Kochhar	21-Sep-1989

```
select round(sysdate - to_date('11-08-2010',
'DD-MM-YYYY'), 1) FROM dual;
```

La fonction NVL

Convertit une valeur NULL en valeur réelle

1

```
SELECT last_name, salary, NVL(commission_pct, 0),  
       (salary*12) + (salary*12*NVL(commission_pct, 0)) AN_SAL  
FROM employees;
```

2

	LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
1	Whalen	4400	0	52800
2	Hartstein	13000	0	156000
3	Fay	6000	0	72000
4	Higgins	12000	0	144000
5	Gietz	8300	0	99600
6	King	24000	0	288000
7	Kochhar	17000	0	204000
8	De Haan	17000	0	204000
9	Hunold	9000	0	108000
10	Ernst	6000	0	72000

1

2

ORACLE

Les expressions CASE et DECODE

Elles facilitent les interrogations conditionnelles en faisant le travail d'une instruction IF-THEN-ELSE :

```
SELECT last_name, job_id, salary,  
       CASE job_id WHEN 'IT_PROG' THEN 1.10*salary  
                   WHEN 'ST_CLERK' THEN 1.15*salary  
                   WHEN 'SA_REP' THEN 1.20*salary  
       ELSE salary END "REVISED_SALARY"  
FROM employees;
```

```
SELECT last_name, job_id, salary,  
       DECODE(job_id, 'IT_PROG', 1.10*salary,  
               'ST_CLERK', 1.15*salary,  
               'SA_REP', 1.20*salary,  
               salary)  
       REVISED_SALARY  
FROM employees;
```

Les fonctions de groupe

```
SELECT AVG(salary), MAX(salary),  
       MIN(salary), SUM(salary)  
FROM   employees  
WHERE  job_id LIKE '%REP%';
```

```
SELECT MIN(hire_date), MAX(hire_date)  
FROM   employees;
```

Créer des groupes de données avec la clause GROUP BY

Toutes les colonnes de la liste `SELECT` qui ne figurent pas dans des fonctions de groupe doivent être présentes dans la clause `GROUP BY`.

```
SELECT    department_id, AVG(salary)
FROM      employees
GROUP BY  department_id ;
```

	DEPARTMENT_ID	AVG(SALARY)
1	(null)	7000
2	20	9500
3	90	19333.333333333333...
4	110	10150
5	50	3500
6	80	10033.333333333333...
7	10	4400
8	60	6400

Interrogations non autorisées avec les fonctions de groupe

Toute colonne ou expression de la liste `SELECT` qui n'est pas une fonction d'agrégation doit figurer dans la clause `GROUP BY` :

```
SELECT department_id, COUNT(last_name)
FROM employees;
```

ORA-00937: not a single-group group function
00937. 00000 - "not a single-group group function"

Il est nécessaire d'ajouter une clause `GROUP BY` pour compter les noms associés à chaque `department_id`.

```
SELECT department_id, job_id, COUNT(last_name)
FROM employees
GROUP BY department_id;
```

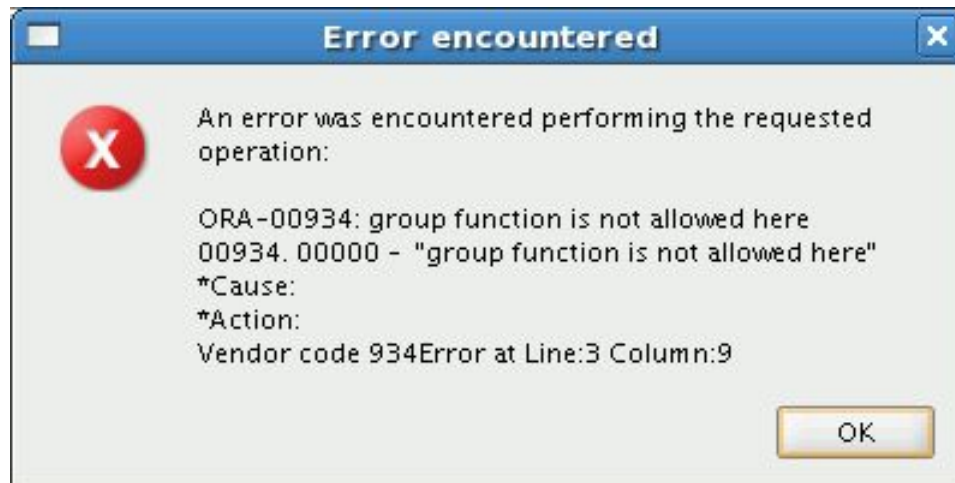
ORA-00979: not a GROUP BY expression
00979. 00000 - "not a GROUP BY expression"

Ajoutez `job_id` dans la clause `GROUP BY` ou supprimez la colonne `job_id` de la liste `SELECT`.

Interrogations non autorisées avec les fonctions de groupe

- Vous ne pouvez pas utiliser la clause `WHERE` pour restreindre des groupes.
- Pour cela, vous pouvez utiliser la clause `HAVING`.
- Vous ne pouvez pas utiliser de fonctions de groupe dans la clause `WHERE`.

```
SELECT    department_id, AVG(salary)
FROM      employees
WHERE     AVG(salary) > 8000
GROUP BY  department_id;
```



Vous ne pouvez pas utiliser la clause `WHERE` pour restreindre des groupes.

Restreindre les résultats d'un groupe avec la clause HAVING

```
SELECT    department_id, MAX(salary)
FROM      employees
GROUP BY  department_id
HAVING    MAX(salary)>10000 ;
```

	DEPARTMENT_ID	MAX(SALARY)
1	20	13000
2	90	24000
3	110	12000
4	80	11000

Utiliser la fonction COUNT

COUNT (*) renvoie le nombre de lignes d'une table :

1

```
SELECT COUNT(*)  
FROM employees  
WHERE department_id = 50;
```

	COUNT(*)
1	5

COUNT(*expr*) renvoie le nombre de lignes comportant des valeurs non NULL pour *expr* :

2

```
SELECT COUNT(commission_pct)  
FROM employees  
WHERE department_id = 80;
```

	COUNT(COMMISSION_PCT)
1	3

Obtenir des données à partir de plusieurs tables: jointure

EMPLOYEES

	EMPLOYEE_ID	DEPARTMENT_ID
1	200	10
2	201	20
3	202	20
4	205	110
5	206	110
6	100	90
7	101	90
8	102	90
9	103	60
10	104	60

...

Clé
étrangère

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME
1	10	Administration
2	20	Marketing
3	50	Shipping
4	60	IT
5	80	Sales
6	90	Executive
7	110	Accounting
8	190	Contracting

Clé
primaire

Extraire des enregistrements avec la clause USING

```
SELECT employee_id, last_name,  
       location_id, department_id  
FROM   employees JOIN departments  
USING (department_id) ;
```

	<small>A Z</small> EMPLOYEE_ID	<small>A Z</small> LAST_NAME	<small>A Z</small> LOCATION_ID	<small>A Z</small> DEPARTMENT_ID
1	200	Whalen	1700	10
2	201	Hartstein	1800	20
3	202	Fay	1800	20
4	144	Vargas	1500	50
5	143	Matos	1500	50
6	142	Davies	1500	50
7	141	Rajs	1500	50
8	124	Mourgos	1500	50

...

18	206	Gietz	1700	110
19	205	Higgins	1700	110

Extraire des enregistrements avec la clause ON

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON     (e.department_id = d.department_id);
```

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	200	Whalen	10	10	1700
2	201	Hartstein	20	20	1800
3	202	Fay	20	20	1800
4	144	Vargas	50	50	1500
5	143	Matos	50	50	1500
6	142	Davies	50	50	1500
7	141	Rajs	50	50	1500
8	124	Mourgos	50	50	1500
9	103	Hunold	60	60	1400
10	104	Ernst	60	60	1400
11	107	Lorentz	60	60	1400

...

Créer des jointures à trois liens à l'aide de la clause ON

```
SELECT employee_id, city, department_name
FROM   employees e
JOIN   departments d
ON     d.department_id = e.department_id
JOIN   locations l
ON     d.location_id = l.location_id;
```

	EMPLOYEE_ID	CITY	DEPARTMENT_NAME
1	100	Seattle	Executive
2	101	Seattle	Executive
3	102	Seattle	Executive
4	103	Southlake	IT
5	104	Southlake	IT
6	107	Southlake	IT
7	124	South San Francisco	Shipping
8	141	South San Francisco	Shipping
9	142	South San Francisco	Shipping

...

Appliquer des conditions supplémentaires à une jointure

Utilisez la clause `AND` ou la clause `WHERE` pour appliquer des conditions supplémentaires :

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON     (e.department_id = d.department_id)  
AND    e.manager_id = 149 ;
```

O

u

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON     (e.department_id = d.department_id)  
WHERE  e.manager_id = 149 ;
```

Renvoyer des enregistrements sans correspondance directe à l'aide de jointures externes

DEPARTMENTS

	DEPARTMENT_NAME	DEPARTMENT_ID
1	Administration	10
2	Marketing	20
3	Shipping	50
4	IT	60
5	Sales	80
6	Executive	90
7	Accounting	110
8	Contracting	190

Le département 190 ne comporte aucun employé.

L'employé "Grant" n'a été associé à un ID de département.

Equijointure avec EMPLOYEES

	DEPARTMENT_ID	LAST_NAME
1	10	Whalen
2	20	Hartstein
3	20	Fay
4	110	Higgins
5	110	Gietz
6	90	King
7	90	Kochhar
8	90	De Haan
9	60	Hunold
10	60	Ernst

...

18	80	Abel
19	80	Taylor

RIGHT OUTER JOIN

```
SELECT e.last_name, d.department_id, d.department_name
FROM   employees e RIGHT OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```




	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Davies	50	Shipping
5	Vargas	50	Shipping
6	Rajs	50	Shipping
7	Mourgos	50	Shipping
8	Matos	50	Shipping

...

18	Higgins	110	Accounting
19	Gietz	110	Accounting
20	(null)	190	Contracting

LEFT OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e LEFT OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```

	 LAST_NAME	 DEPARTMENT_ID	 DEPARTMENT_NAME
1	Whalen	10	Administration
2	Fay	20	Marketing
3	Hartstein	20	Marketing
4	Vargas	50	Shipping
5	Matos	50	Shipping

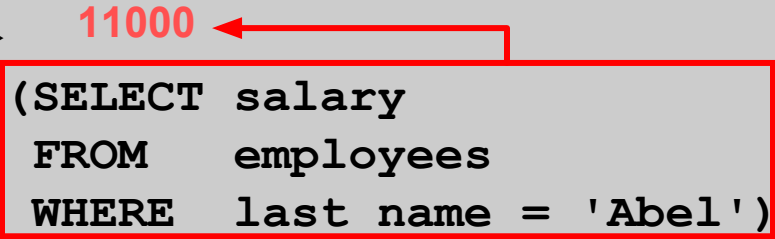
...

16	Kochhar	90	Executive
17	King	90	Executive
18	Gietz	110	Accounting
19	Higgins	110	Accounting
20	Grant	(null)	(null)

Sous-interrogation



- Qui a un salaire supérieur à celui d'Abel ?

```
SELECT last_name, salary
FROM employees
WHERE salary > 11000
      (SELECT salary
        FROM employees
        WHERE last_name = 'Abel');
```

A red arrow points from the value '11000' to the subquery. A red box highlights the subquery itself. The subquery is: (SELECT salary FROM employees WHERE last_name = 'Abel');

	LAST_NAME	SALARY
1	Hartstein	13000
2	Higgins	12000
3	King	24000
4	Kochhar	17000
5	De Haan	17000

Exécuter des sous-interrogations monolignes

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id =  SA_REP
AND    salary >  8600
        (SELECT job_id
         FROM   employees
         WHERE  last_name = 'Taylor')
        (SELECT salary
         FROM   employees
         WHERE  last_name = 'Taylor');
```

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

La sous interrogation renvoie plusieurs lignes

Employés qui ne sont pas programmeurs en informatique et dont le salaire est inférieur à celui de n'importe quel programmeur

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

Employés dont le salaire est inférieur à celui de tous les employés dont l'ID de poste est IT_PROG et dont le poste n'est pas IT_PROG.

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

9000, 6000, 4200

Les opérateurs ensembliste

```
SELECT employee_id,  
       job_id  
FROM   employees  
UNION  
SELECT employee_id,  
       job_id  
FROM   job_history;
```

```
SELECT employee_id, job_id, department_id  
FROM   employees  
UNION ALL  
SELECT employee_id, job_id, department_id  
FROM   job_history  
ORDER BY employee_id;
```

Les opérateurs ensembliste

```
SELECT employee_id, job_id
FROM   employees
INTERSECT
SELECT employee_id, job_id
FROM   job_history;
```

```
SELECT employee_id
FROM   employees
MINUS
SELECT employee_id
FROM   job_history;
```

Objets de base de données

Objet	Description
Table	Unité élémentaire de stockage, constituée de lignes
Vue	Représente de façon logique des sous-ensembles de données issus d'une ou de plusieurs tables
Séquence	Génère des valeurs numériques
Index	Améliore les performances de certaines interrogations d'extraction de données
Synonyme	Attribue un autre nom aux objets

Vue

- Créez la vue EMPVU80, qui contient les détails relatifs aux employés du département 80 :

```
CREATE VIEW empvu80
AS SELECT employee_id, last_name, salary
FROM employees
WHERE department_id = 80;
```

```
CREATE VIEW succeeded.
```

- Décrivez la structure de la vue à l'aide de la commande DESCRIBE de iSQL*Plus :

```
DESCRIBE empvu80
```

- Extraire les données à l'aide d'une vue

```
SELECT * FROM empvu80
```

Utiliser la clause WITH CHECK OPTION

- La clause WITH CHECK OPTION permet de garantir que les opérations LMD effectuées sur la vue concernent uniquement les lignes sélectionnées par celle-ci :

```
CREATE OR REPLACE VIEW empvu20
AS SELECT *
   FROM   employees
  WHERE   department_id = 20
  WITH CHECK OPTION CONSTRAINT empvu20_ck ;
```

- Toute tentative d'insertion (INSERT) d'une ligne dont la valeur `department_id` est différente de 20 ou de mise à jour (UPDATE) de l'ID de département d'une ligne dans la vue échoue car il s'agit d'une violation de la contrainte WITH CHECK OPTION.

Séquence: générateur de nombre entier

```
CREATE SEQUENCE dept_deptid_seq  
            INCREMENT BY 10  
            START WITH 120  
            MAXVALUE 9999;
```

Utiliser une séquence

- Insérez un nouveau département nommé Support à l'emplacement dont l'ID = 2500 :

```
INSERT INTO departments (department_id,  
                        department_name, location_id)  
VALUES                (dept_deptid_seq.NEXTVAL,  
1 rows inserted      'Support', 2500);
```

- Affichez la valeur actuelle de la séquence

```
SELECT dept_deptid_seq.CURRVAL  
FROM   dual;
```

Index

Améliore les performances de certaines interrogations

Créé automatiquement : Un index unique est créé automatiquement lorsque vous définissez une contrainte PRIMARY KEY ou UNIQUE dans la définition d'une table

Ou

Créé manuellement

```
CREATE INDEX emp_last_name_idx  
ON      employees(last_name);
```

Synonymes

- Définissez un nom abrégé pour la vue DEPT_SUM_VU :

```
CREATE SYNONYM d_sum  
FOR dept_sum_vu;
```

Supprimer des objets de base de données

```
DROP TABLE emp;
```

```
DROP view empvu80;
```

```
DROP index emp_last_name_idx;
```

```
DROP SYNONYM d_sum;
```