

PL / SQL



PL / SQL

Module 1 – Présentation de l'écosystème ORACLE



Objectifs

- Découvrir Oracle
- Découvrir l'environnement logiciel
- Savoir créer un utilisateur Oracle fonctionnel
- Découvrir la notion de base de données Oracle



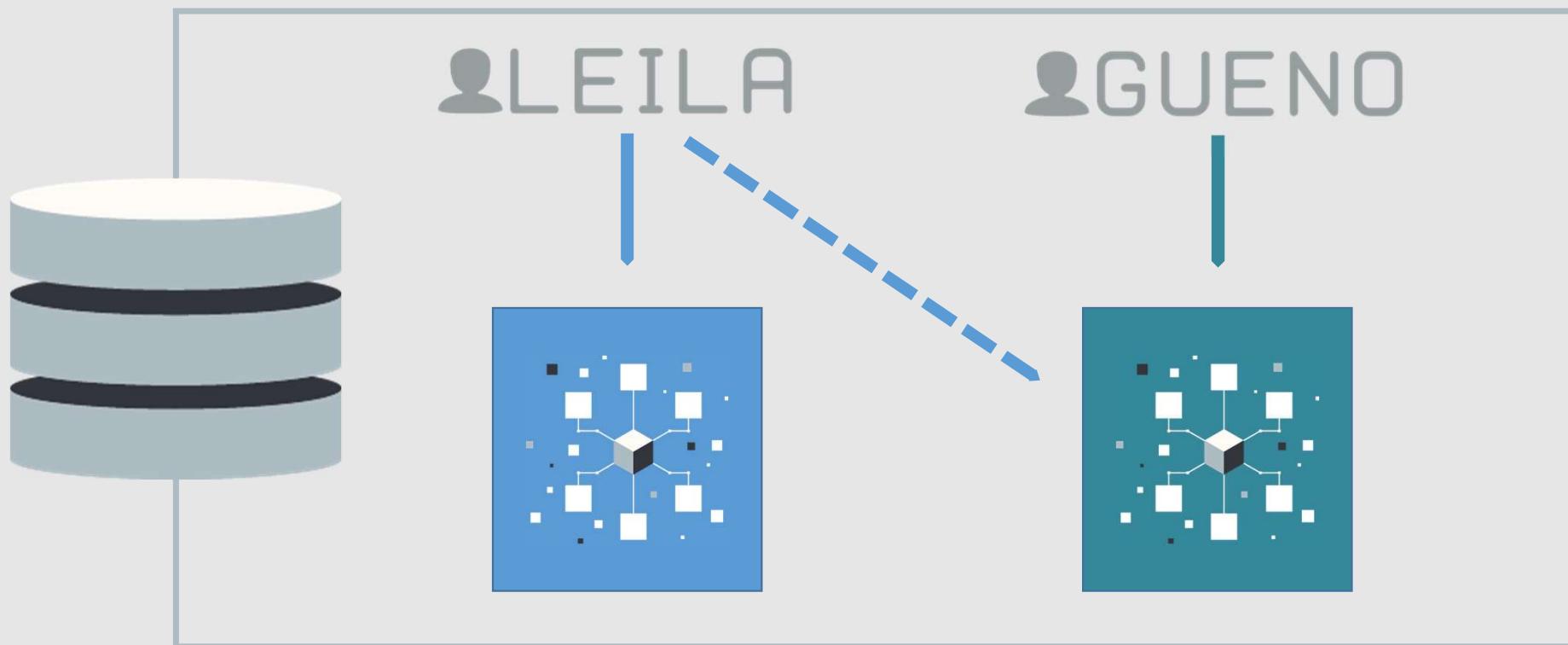
Présentation de l'écosystème ORACLE

L'entreprise



Présentation de l'écosystème ORACLE

Notion de base de données chez Oracle



Présentation de l'écosystème ORACLE

Création d'un utilisateur avec des droits

```
create role eni;
grant connect, resource to eni;
grant create procedure to eni;
grant create trigger to eni;

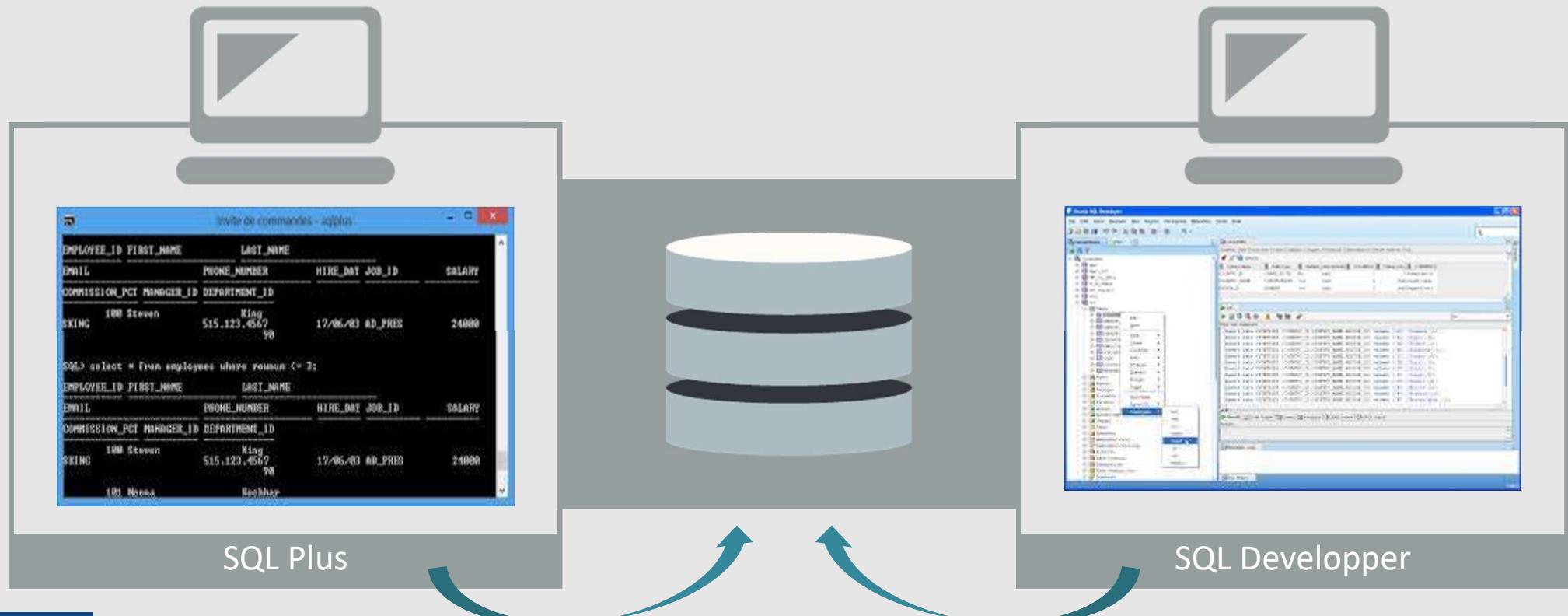
create user ul
identified by x
default tablespace users
temporary tablespace temp
quota unlimited on users;

grant eni to ul;
```



Présentation de l'écosystème ORACLE

L'environnement logiciel



Présentation de l'écosystème ORACLE
SQL Plus

Démonstration



Présentation de l'écosystème ORACLE
SQL Developer

Démonstration



Conclusion

- Vous connaissez l'entreprise Oracle
- Vous avez compris le concept de schéma
- Vous avez découvert SQL Plus
- Vous savez créer un nouvel utilisateur
- Vous avez découvert SQL Developer



PL / SQL

Module de rappel – La gestion des tables sous Oracle



Objectifs

- Rappel des instructions du DDL
- Découverte des particularités Oracle



La gestion des tables sous Oracle

La dette technique



Inventeur du concept de wiki et
a permis l'élaboration de
Wikipédia

-Ward Cunningham-

La gestion des tables sous Oracle

La dette technique

Quand on code au plus vite et de manière non optimale, on contracte une dette technique que l'on rembourse tout au long de la vie du projet sous forme de temps de développement de plus en plus long et de bugs de plus en plus fréquents.



La gestion des tables sous Oracle

Les conventions de nommage

- Mots clés SQL en majuscules
- Nom des tables au pluriel et en snake_case
- Nom des colonnes explicites et en snake_case

https://docs.oracle.com/cd/A97630_01/appdev.920/a96624/02_funds.htm



La gestion des tables sous Oracle

Les commentaires de code

```
/* Je suis  
un commentaire sur  
plusieurs lignes */
```

```
--Je suis un commentaire sur une ligne
```

La gestion des tables sous Oracle

Les principaux types caractères

CHAR (Size)

VARCHAR (Size)

LONG



La gestion des tables sous Oracle

Le principal type numérique

NUMBER(precision, scale)

Valeur réelle	Spécification de la colonne	Valeur stockée
123,89	NUMBER	123,89
123,89	NUMBER(3)	124
123,89	NUMBER(3,2)	Impossible
123,89	NUMBER(6,-2)	100
0,000127	NUMBER(4,5)	0,00013



La gestion des tables sous Oracle

Les principaux types dates

DATE

TIMESTAMP



La gestion des tables sous Oracle

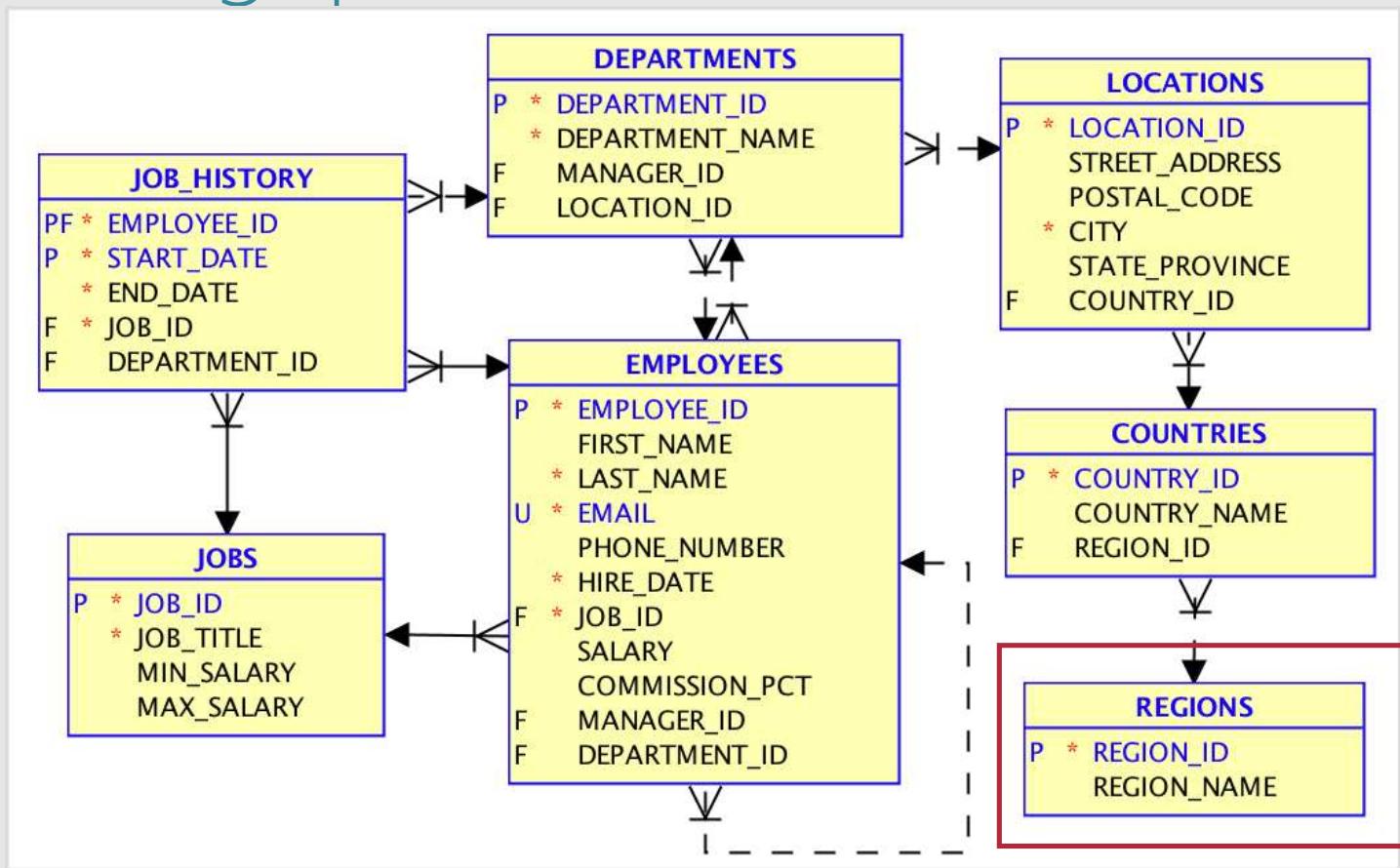
Le principal type multimédia

BLOB



La gestion des tables sous Oracle

Le schéma logique de base de données



La gestion des tables sous Oracle

La création de la table REGIONS

```
CREATE TABLE regions
(
    region_id      NUMBER PRIMARY KEY,
    region_name    VARCHAR2(25) DEFAULT 'XXX'
);
```

La gestion des tables sous Oracle

Les commentaires posés sur les objets

```
COMMENT ON TABLE regions
IS 'Regions table that contains region numbers and names. Contains 4 rows; references with the Countries table.'

COMMENT ON COLUMN regions.region_id
IS 'Primary key of regions table.'

COMMENT ON COLUMN regions.region_name
IS 'Names of regions. Locations are in the countries of these regions.'
```



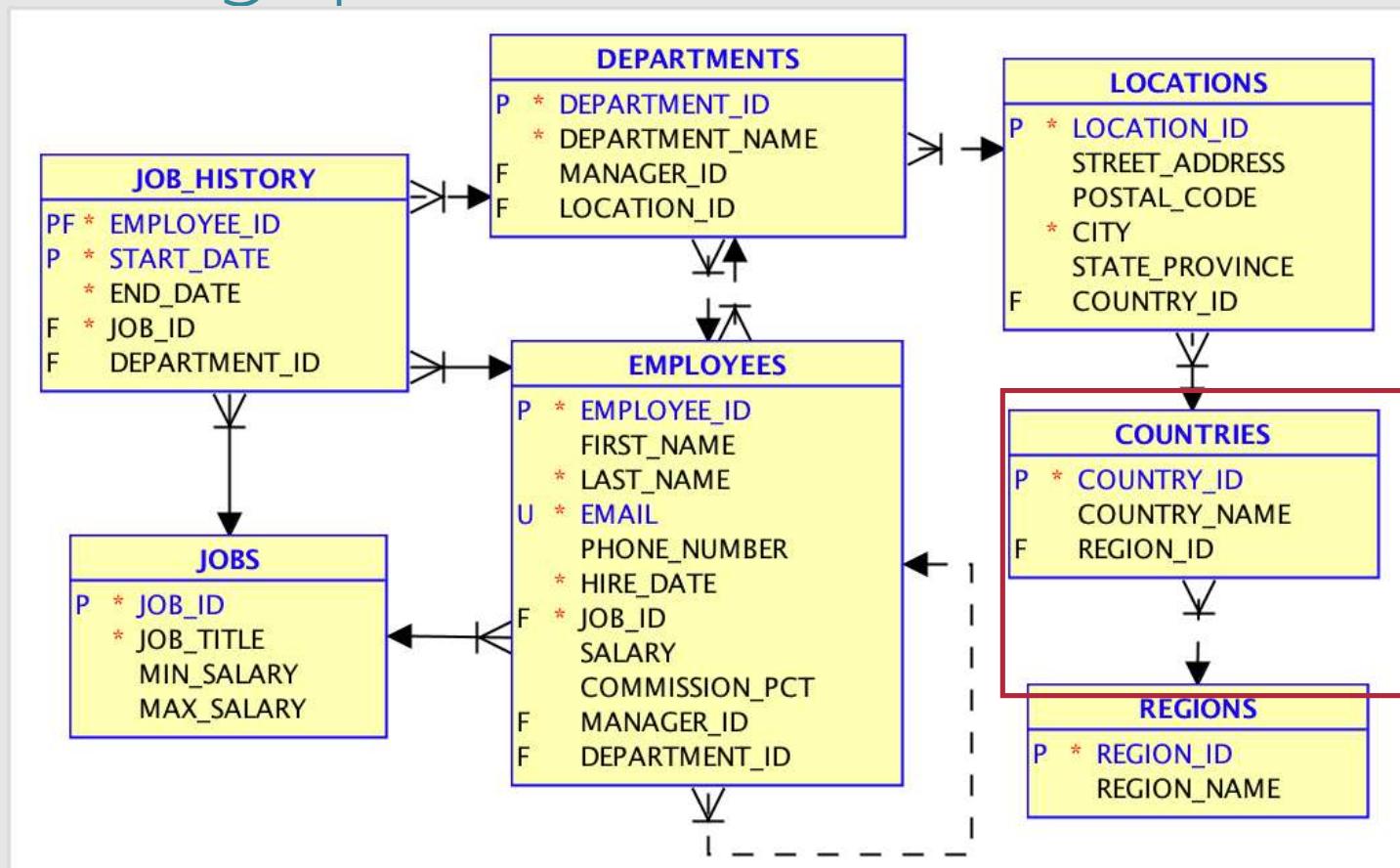
La gestion des tables sous Oracle

La consultation des commentaires

```
SELECT * FROM user_tab_comments;  
  
SELECT * FROM user_col_comments;
```

La gestion des tables sous Oracle

Le schéma logique de base de données



La gestion des tables sous Oracle

La création de la table COUNTRIES

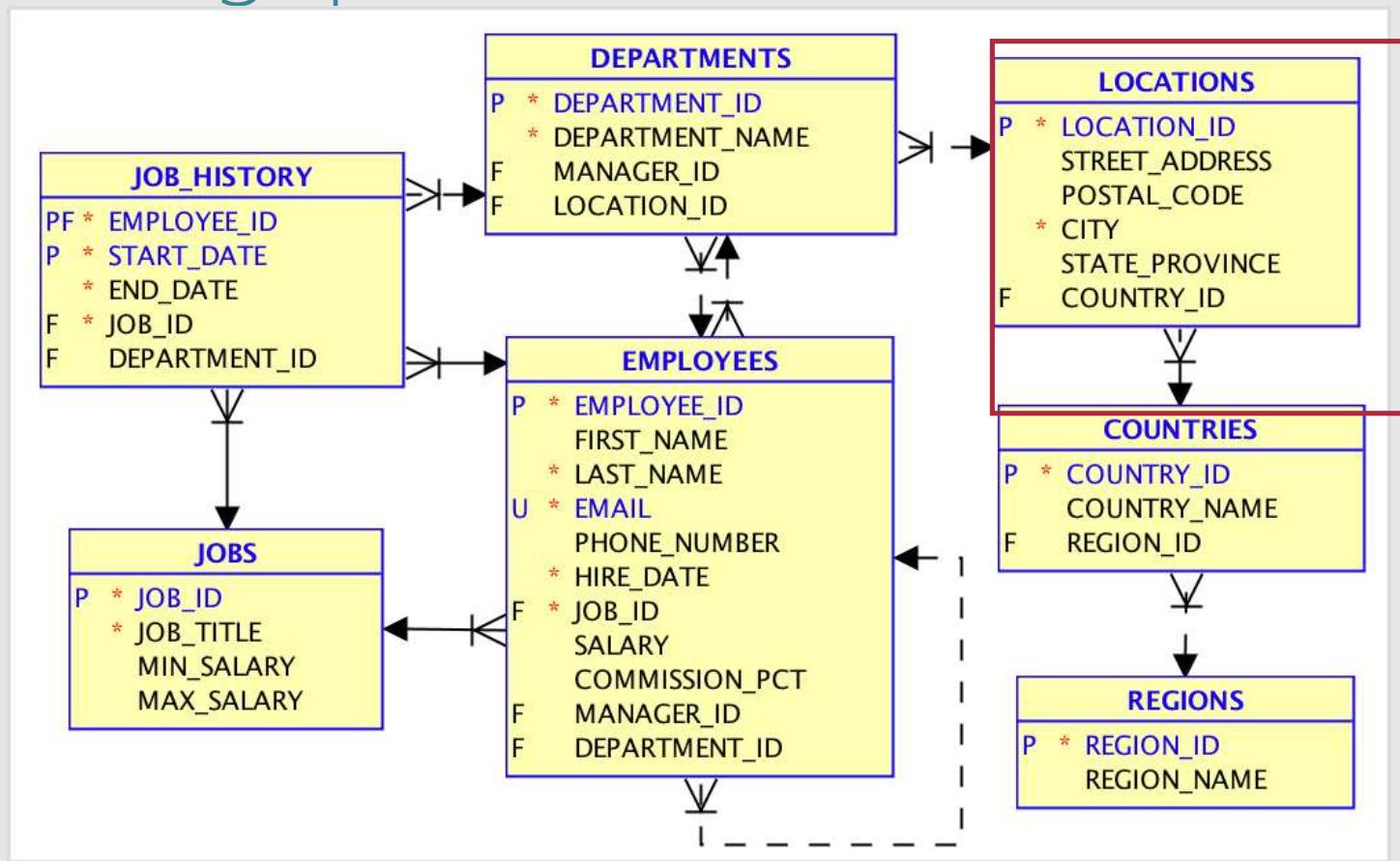
```
CREATE TABLE countries
(
    country_id CHAR(2) CONSTRAINT country_id_nn NOT NULL,
    country_name      VARCHAR2(40),
    region_id        NUMBER,
    CONSTRAINT          country_c_id_pk PRIMARY KEY (country_id)
);

ALTER TABLE countries
ADD (
    CONSTRAINT countr_reg_fk
    FOREIGN KEY (region_id)
    REFERENCES regions(region_id)
);
```



La gestion des tables sous Oracle

Le schéma logique de base de données



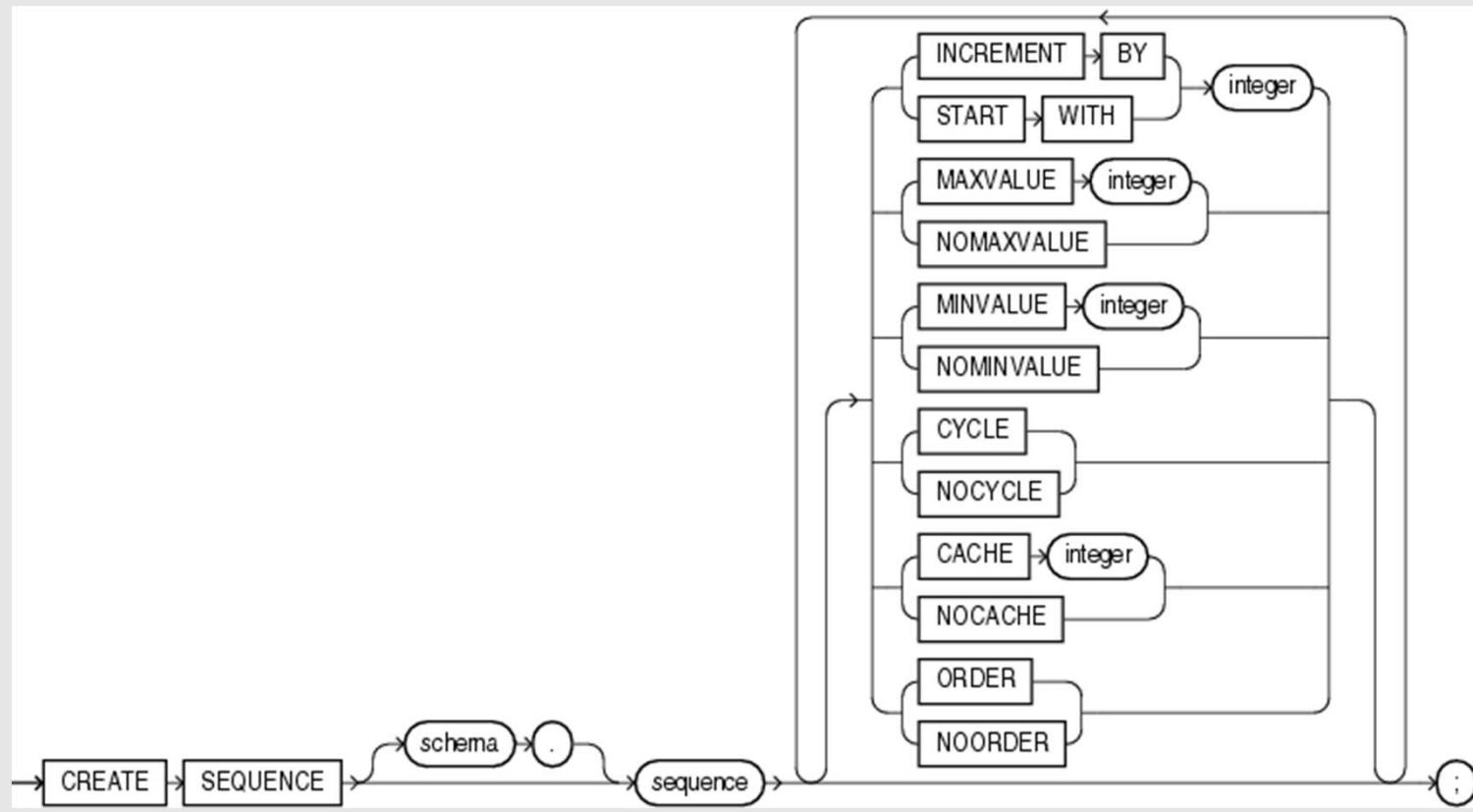
La gestion des tables sous Oracle

La création de la table LOCATIONS

```
CREATE TABLE locations
(
    location_id      NUMBER(4) CONSTRAINT loc_id_pk PRIMARY KEY,
    street_address   VARCHAR2(40),
    postal_code      VARCHAR2(12),
    city              VARCHAR2(30) CONSTRAINT loc_city_nn NOT NULL,
    state_province   VARCHAR2(25),
    country_id       CHAR(2) CONSTRAINT loc_c_id_fk REFERENCES countries(country_id)
);
```

La gestion des tables sous Oracle

La création d'une séquence



La gestion des tables sous Oracle

La création d'une séquence

```
CREATE SEQUENCE locations_seq
    START WITH      3300
    INCREMENT BY   100
    MAXVALUE       9900
    NOCACHE
    NOCYCLE;
```

La gestion des tables sous Oracle

L'utilisation d'une séquence

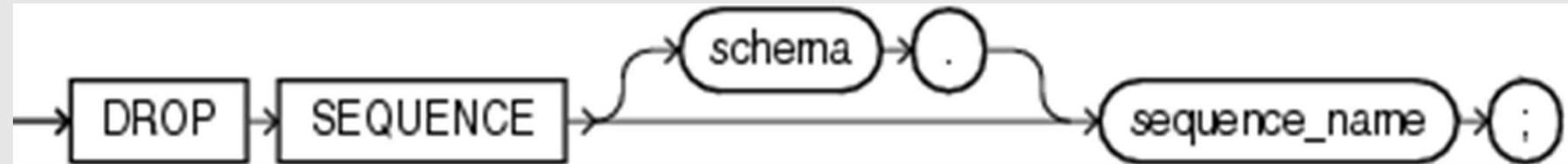
```
locations_seq.NEXTVAL
```

```
locations_seq.CURRVAL
```



La gestion des tables sous Oracle

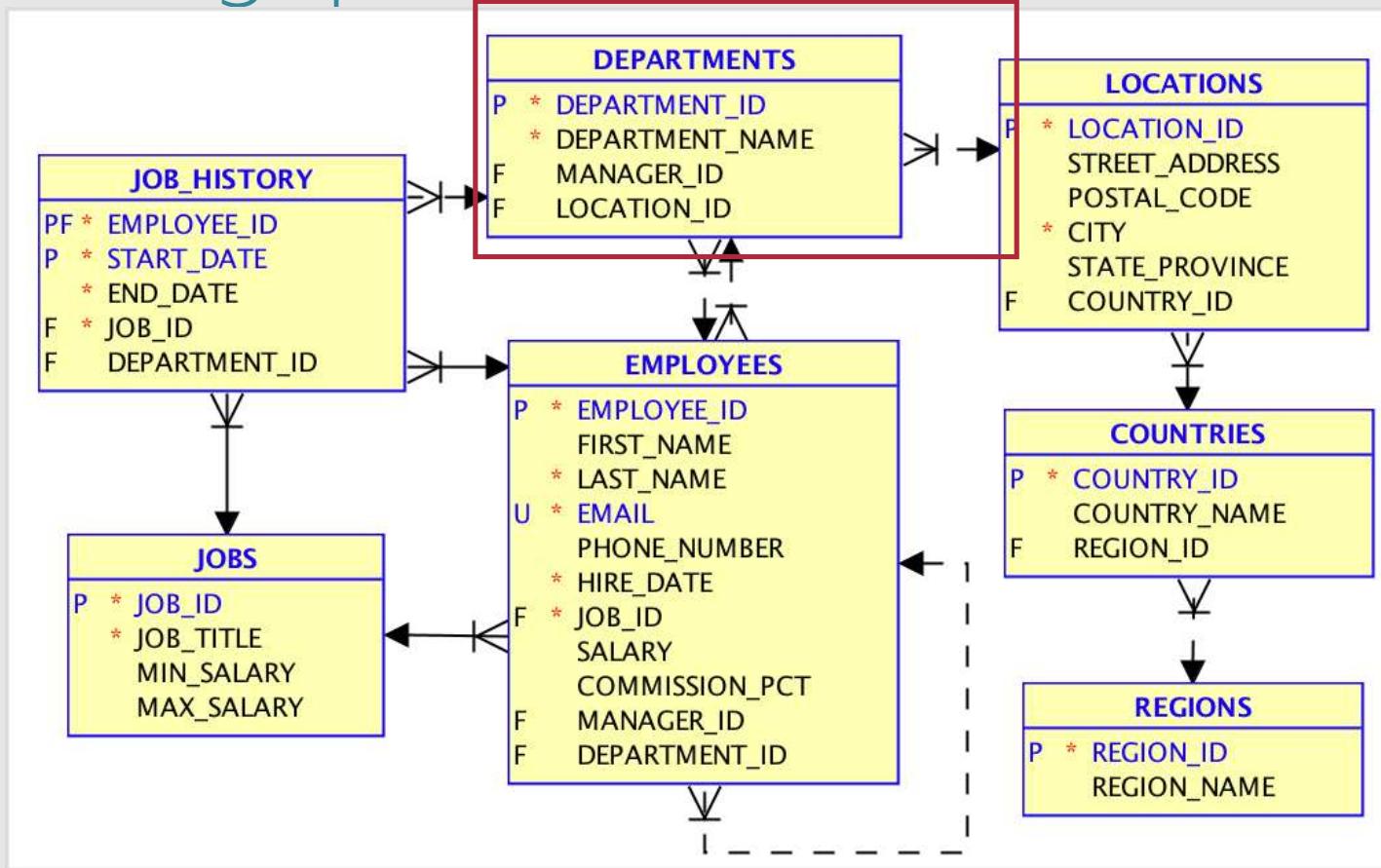
La suppression d'une séquence



```
DROP SEQUENCE departments_seq;  
DROP SEQUENCE employees_seq;  
DROP SEQUENCE locations_seq;
```

La gestion des tables sous Oracle

Le schéma logique de base de données



La gestion des tables sous Oracle

La création de la table DEPARTMENTS

```
CREATE TABLE departments
(
    department_id      NUMBER(4),
    department_name   VARCHAR2(30) CONSTRAINT dept_name_nn NOT NULL,
    manager_id        NUMBER(6),
    location_id       NUMBER(4)
);

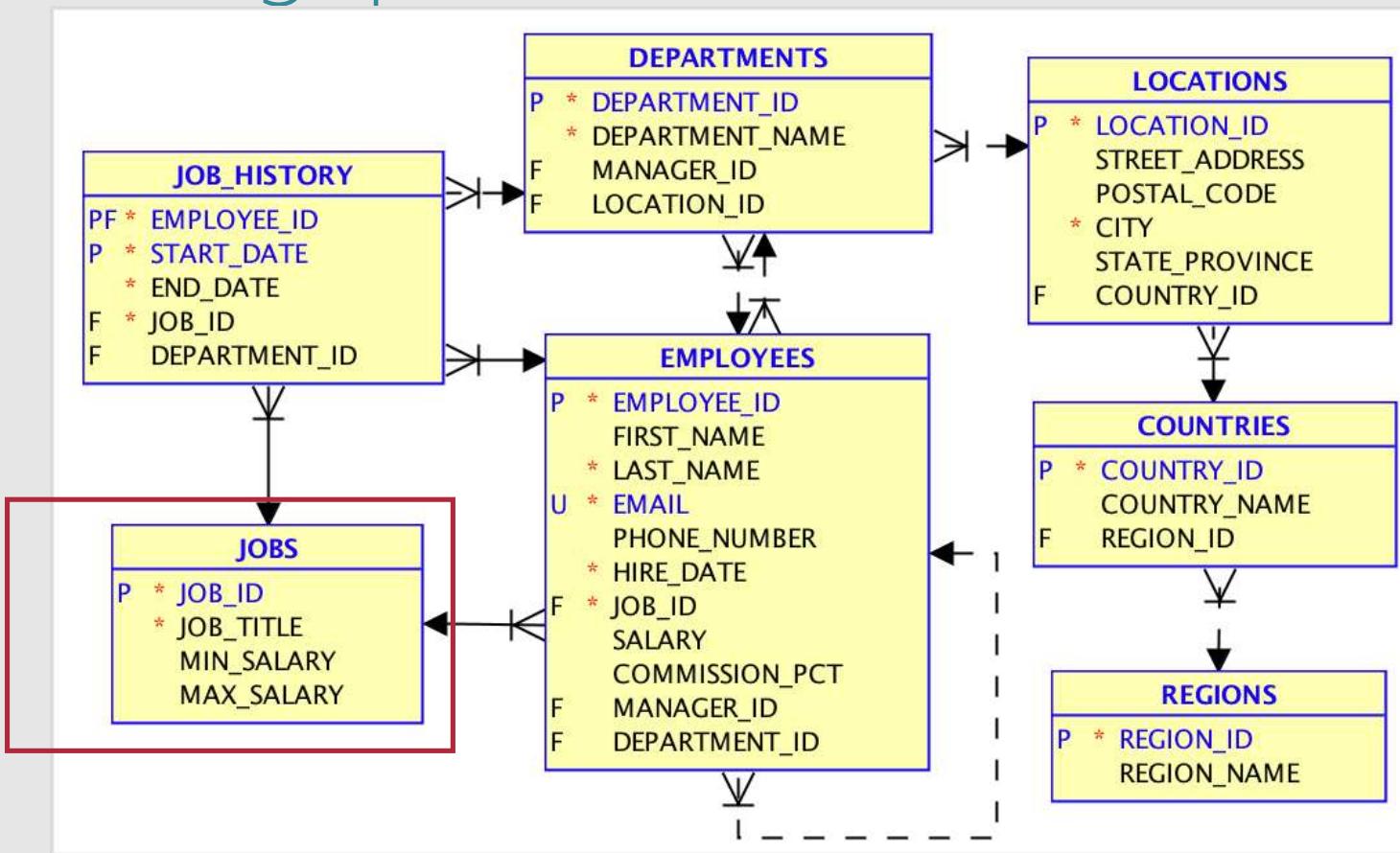
ALTER TABLE departments
ADD (
    CONSTRAINT dept_id_pk PRIMARY KEY (department_id),
    CONSTRAINT dept_loc_fk FOREIGN KEY (location_id) REFERENCES locations (location_id)
);

CREATE SEQUENCE departments_seq
    START WITH      280
    INCREMENT BY   10
    MAXVALUE       9990
    NOCACHE
    NOCYCLE;
```



La gestion des tables sous Oracle

Le schéma logique de base de données



La gestion des tables sous Oracle

La création de la table JOBS

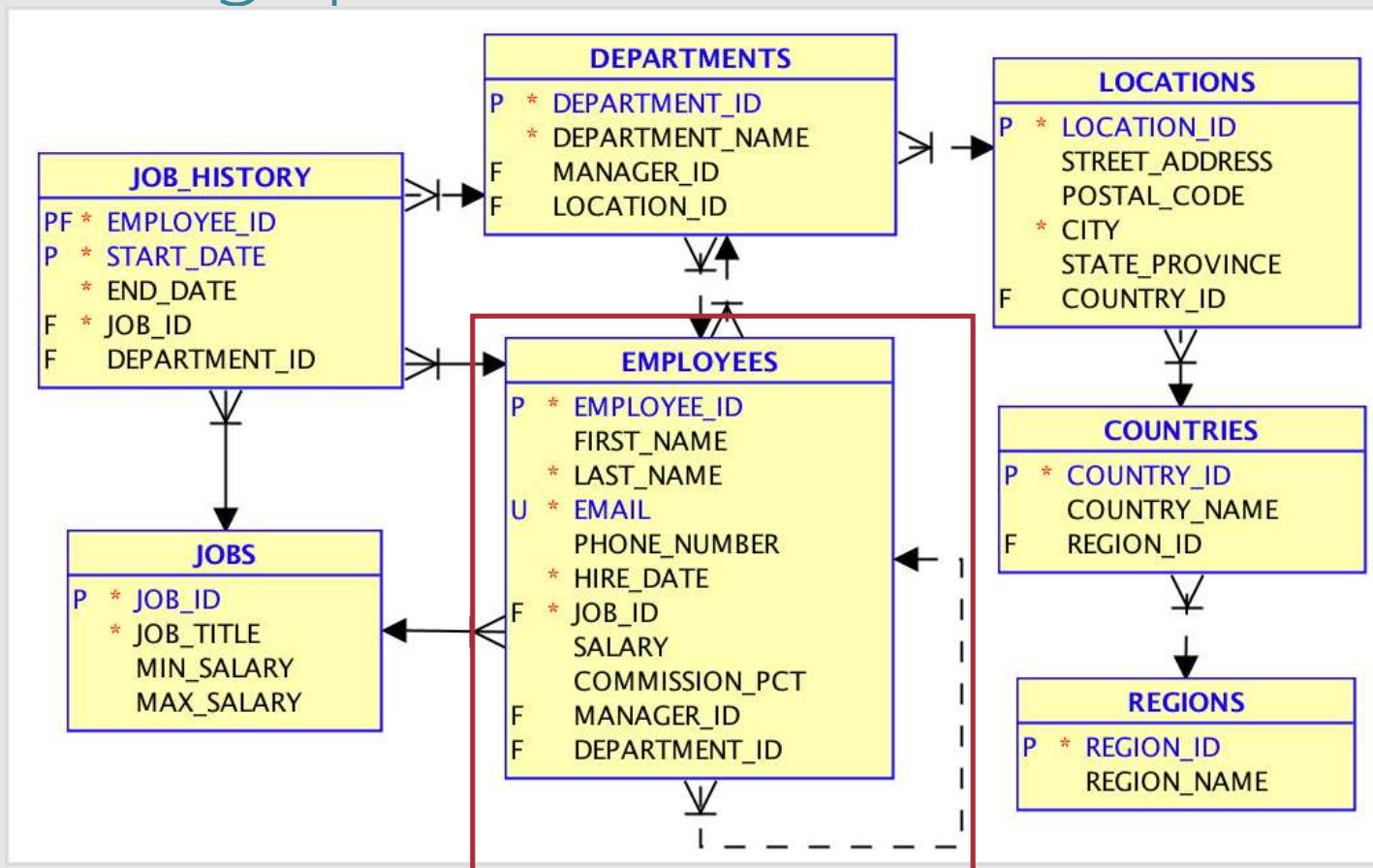
```
CREATE TABLE jobs
(
    job_id          VARCHAR2(10),
    job_title       VARCHAR2(35)
    CONSTRAINT      job_title_nn  NOT NULL,
    min_salary      NUMBER(6),
    max_salary      NUMBER(6)
);

CREATE UNIQUE INDEX job_id_pk ON jobs (job_id) ;

ALTER TABLE jobs
ADD (
    CONSTRAINT job_id_pk PRIMARY KEY(job_id)
) ;
```

La gestion des tables sous Oracle

Le schéma logique de base de données



La gestion des tables sous Oracle

La création de la table EMPLOYEES

```
CREATE TABLE employees
(
    employee_id      NUMBER(6),
    first_name       VARCHAR2(20),
    last_name        VARCHAR2(25) CONSTRAINT emp_last_name_nn NOT NULL,
    email            VARCHAR2(25) CONSTRAINT emp_email_nn NOT NULL,
    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),
    commission_pct   NUMBER(2,2),
    manager_id       NUMBER(6),
    department_id    NUMBER(4),
    ,CONSTRAINT emp_salary_min CHECK (salary > 0)
    ,CONSTRAINT emp_email_uk UNIQUE (email)
);
```

La gestion des tables sous Oracle

L'intégration de la table EMPLOYEES

```
ALTER TABLE employees
ADD (
    CONSTRAINT      emp_emp_id_pk PRIMARY KEY (employee_id),
    CONSTRAINT      emp_dept_fk FOREIGN KEY (department_id) REFERENCES departments,
    CONSTRAINT      emp_job_fk FOREIGN KEY (job_id) REFERENCES jobs (job_id),
    CONSTRAINT      emp_manager_fk FOREIGN KEY (manager_id) REFERENCES employees
) ;

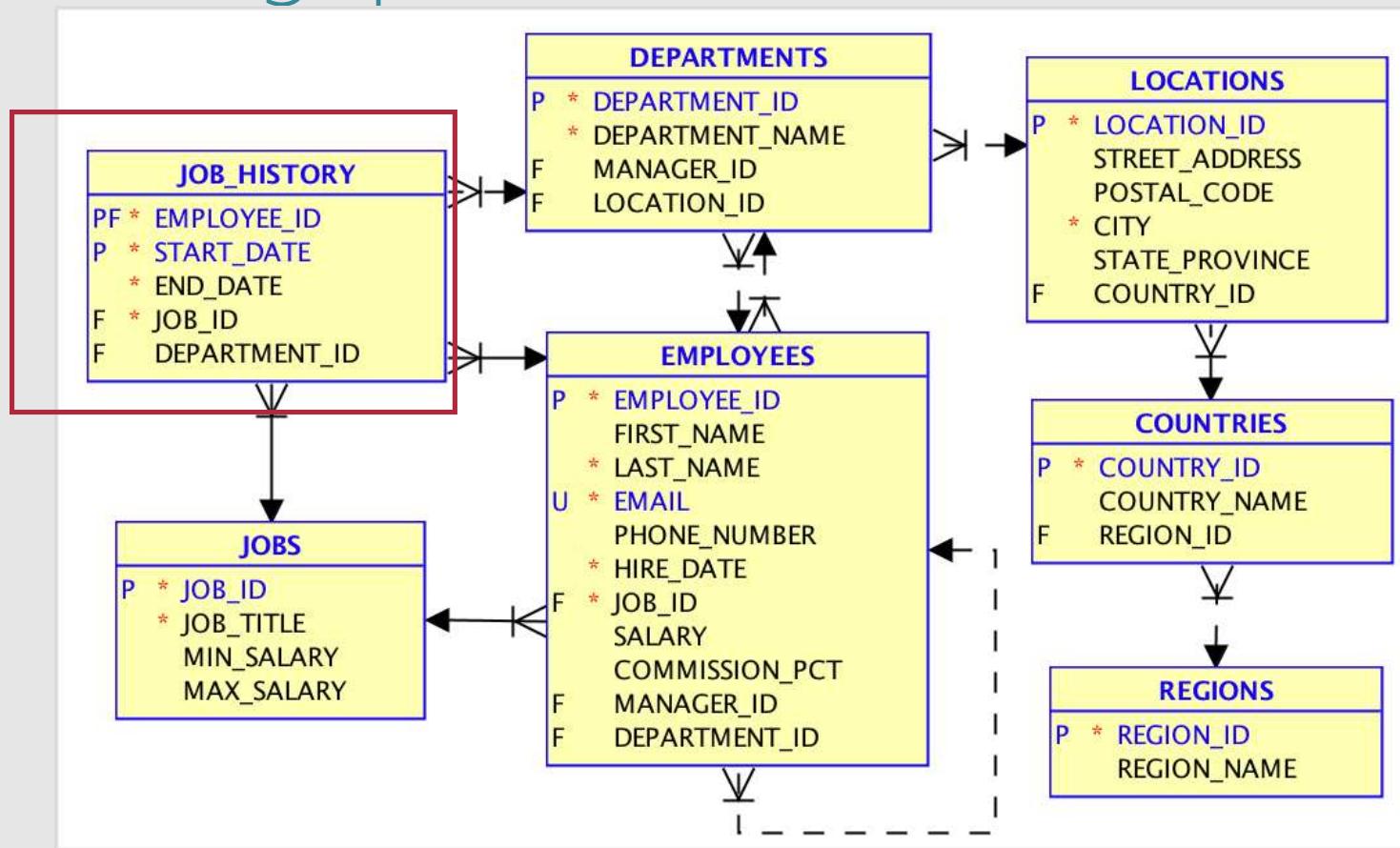
ALTER TABLE departments
ADD (
    CONSTRAINT dept_mgr_fk FOREIGN KEY (manager_id) REFERENCES employees (employee_id)
) ;

CREATE SEQUENCE employees_seq
    START WITH      207
    INCREMENT BY   1
    NOCACHE
    NOCYCLE;
```



La gestion des tables sous Oracle

Le schéma logique de base de données



La gestion des tables sous Oracle

Les contraintes portant sur plusieurs colonnes

```
CREATE TABLE job_history
(
    employee_id      NUMBER(6) CONSTRAINT      jhist_employee_nn NOT NULL,
    start_date       DATE CONSTRAINT        jhist_start_date_nn NOT NULL,
    end_date         DATE CONSTRAINT        jhist_end_date_nn NOT NULL,
    job_id           VARCHAR2(10) CONSTRAINT     jhist_job_nn NOT NULL,
    department_id    NUMBER(4),
    CONSTRAINT        jhist_date_interval CHECK (end_date > start_date)
);

CREATE UNIQUE INDEX jhist_emp_id_st_date_pk ON job_history (employee_id, start_date) ;

ALTER TABLE job_history
ADD (
    CONSTRAINT jhist_emp_id_st_date_pk PRIMARY KEY (employee_id, start_date),
    CONSTRAINT      jhist_job_fk FOREIGN KEY (job_id) REFERENCES jobs,
    CONSTRAINT      jhist_emp_fk FOREIGN KEY (employee_id) REFERENCES employees,
    CONSTRAINT      jhist_dept_fk FOREIGN KEY (department_id) REFERENCES departments
) ;
```

La gestion des tables sous Oracle

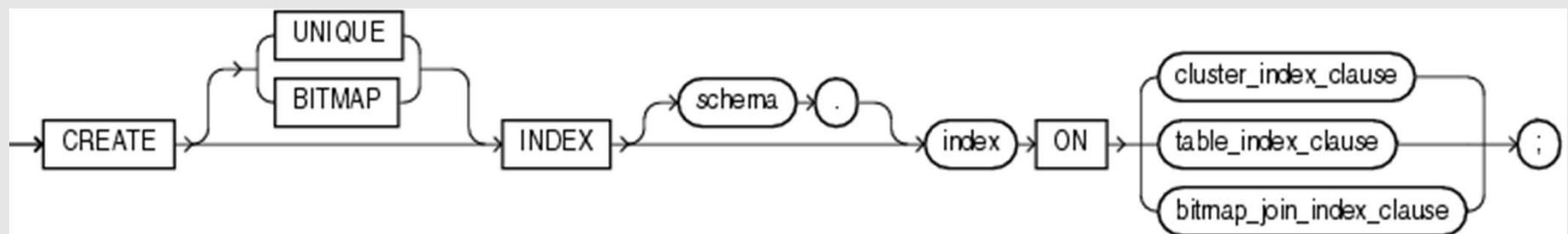
Le vidage d'une table

```
TRUNCATE TABLE informations;
```



La gestion des tables sous Oracle

La syntaxe de création d'un index



La gestion des tables sous Oracle

Exemple de création d'un index

```
CREATE INDEX emp_department_ix ON employees (department_id);

CREATE INDEX emp_job_ix ON employees (job_id);

CREATE INDEX emp_manager_ix  ON employees (manager_id);

CREATE INDEX emp_name_ix ON employees (last_name, first_name);

CREATE INDEX dept_location_ix ON departments (location_id);

CREATE INDEX jhist_job_ix ON job_history (job_id);

CREATE INDEX jhist_employee_ix ON job_history (employee_id);

CREATE INDEX jhist_department_ix ON job_history (department_id);

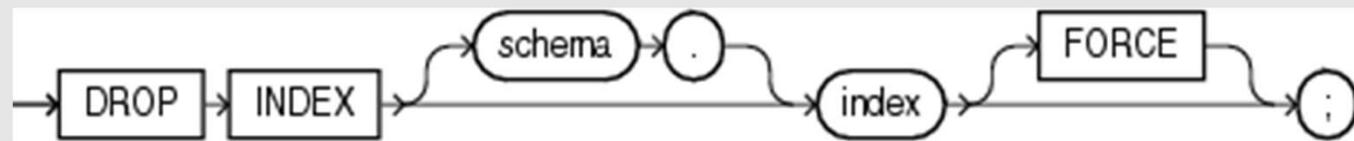
CREATE INDEX loc_city_ix ON locations (city);

CREATE INDEX loc_state_province_ix ON locations (state_province);

CREATE INDEX loc_country_ix ON locations (country_id);
```

La gestion des tables sous Oracle

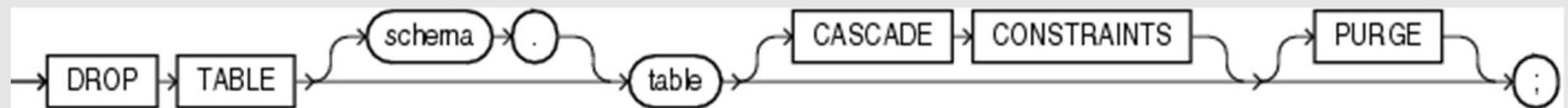
La suppression d'un index



```
DROP INDEX dept_location_ix
```

La gestion des tables sous Oracle

La suppression d'une table



```
DROP TABLE regions      CASCADE CONSTRAINTS;
DROP TABLE departments   CASCADE CONSTRAINTS;
DROP TABLE locations    CASCADE CONSTRAINTS;
DROP TABLE jobs          CASCADE CONSTRAINTS;
DROP TABLE job_history   CASCADE CONSTRAINTS;
DROP TABLE employees     CASCADE CONSTRAINTS;
DROP TABLE countries     CASCADE CONSTRAINTS;
```

La gestion des tables sous Oracle

La suppression d'une contrainte

```
ALTER TABLE locations DROP CONSTRAINT loc_city_nn;
```



La gestion des tables sous Oracle

L'ajout ou suppression d'une colonne

```
ALTER TABLE departments ADD dn VARCHAR2(300);
```

```
ALTER TABLE departments DROP COLUMN dn;
```

La gestion des tables sous Oracle

Le changement de nom d'une colonne

```
ALTER TABLE regions  
RENAME COLUMN region_name TO region_true_name;
```



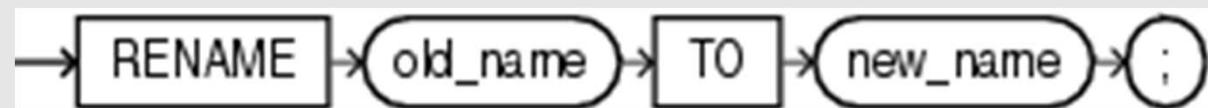
La gestion des tables sous Oracle

La modification d'une colonne

```
ALTER TABLE  
    departments  
MODIFY  
    department_name VARCHAR2(50);
```

La gestion des tables sous Oracle

Le changement de nom d'une table



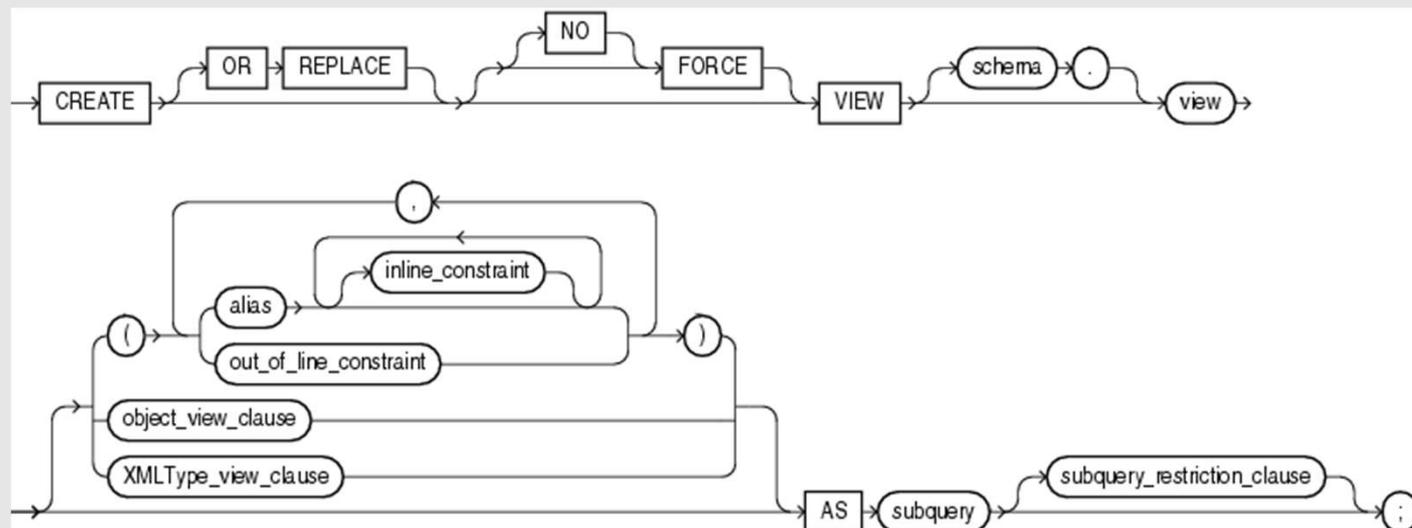
La gestion des tables sous Oracle

La définition d'une colonne identité

```
CREATE TABLE informations
(
    informations_id NUMBER GENERATED AS IDENTITY,
    informations_text VARCHAR(500)
)
```

La gestion des tables sous Oracle

La syntaxe de création d'une vue



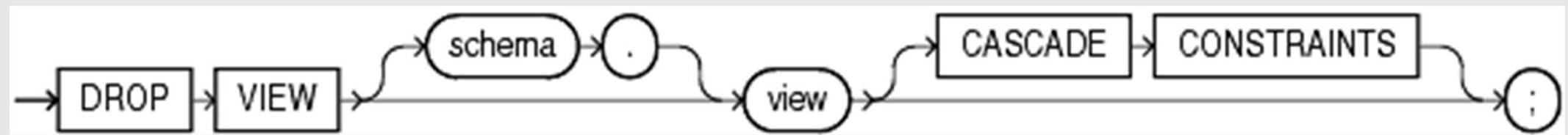
La gestion des tables sous Oracle

Exemple de création d'une vue

```
CREATE OR REPLACE FORCE VIEW regions_countries_view
AS
SELECT
    r.region_name,
    c.country_name
FROM
    regions r
    JOIN countries c ON r.region_id = c.region_id;
```

La gestion des tables sous Oracle

La suppression d'une vue



La gestion des tables sous Oracle

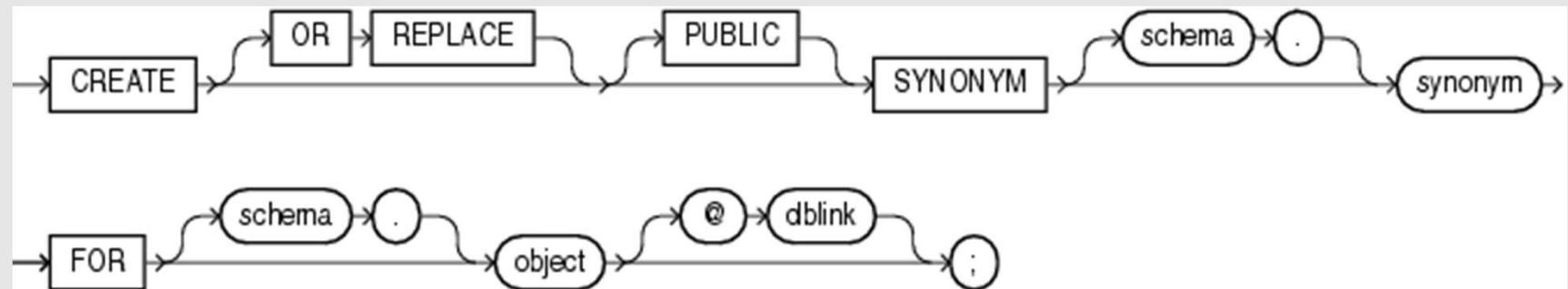
Exemple de suppression d'une vue

```
DROP VIEW regions_countries_view;
```



La gestion des tables sous Oracle

La syntaxe de création d'un synonyme



```
CREATE SYNONYM history FOR job_history;
```

La gestion des tables sous Oracle

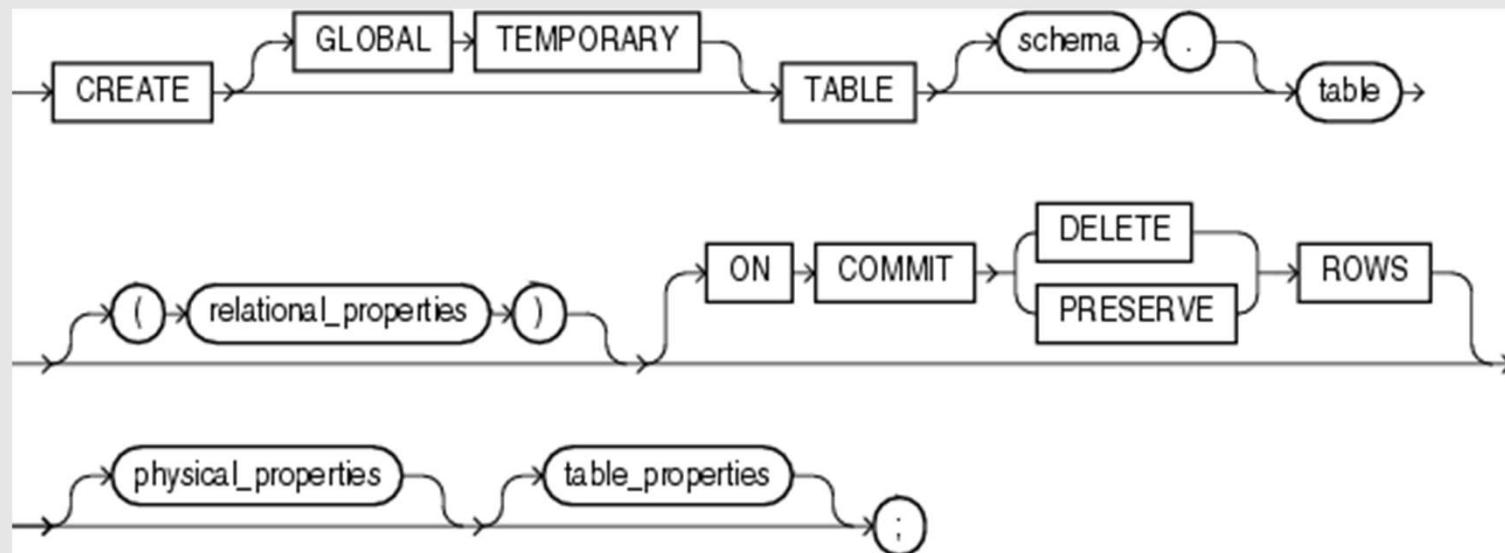
La suppression d'un synonyme



```
DROP SYNONYM history;
```

La gestion des tables sous Oracle

La syntaxe de création d'une table temporaire



La gestion des tables sous Oracle

Exemple de création d'une table temporaire

```
CREATE GLOBAL TEMPORARY TABLE employees_sel
(
    id  VARCHAR2(256),
    end_date      DATE
) ON COMMIT PRESERVE ROWS ;
```

Conclusion

- Vous savez mettre en place une base de données sous Oracle



PL / SQL

Module de rappel – La gestion des données sous Oracle



Objectifs

- Rappel du DML
- Découverte des particularités Oracle



La gestion des données sous Oracle

L'insertion d'enregistrement



`INSERT INTO Table VALUES (valeur_1, valeur_2, valeur_3)`

`INSERT INTO Table (colonne_1, colonne_3) VALUES (valeur_1, valeur_3)`



Il faut que tous les attributs NOT NULL soient définis lors d'un INSERT.

La gestion des données sous Oracle

Exemples d'insertions d'enregistrements

```
INSERT INTO regions(region_id,region_name) VALUES (1, 'Europe') ;  
  
INSERT INTO regions VALUES (2, 'Americas') ;  
  
INSERT INTO regions VALUES ( 3, 'Asia') ;  
  
INSERT INTO regions VALUES ( 4, 'Middle East and Africa') ;
```

La gestion des données sous Oracle

La mise à jour d'enregistrement(s)

UPDATE

UPDATE *Table*

SET *colonne* = *valeur*

SET

UPDATE *Table*

SET *colonne_1* = *valeur*

WHERE *colonne_2* = *condition*



La gestion des données sous Oracle

Exemples de mises à jour d'enregistrements

```
UPDATE regions SET region_name = UPPER(region_name)
```

```
UPDATE regions SET region_name = 'America' WHERE region_name = 'Americas'
```



La gestion des données sous Oracle

La suppression d'enregistrement(s)

DELETE

`DELETE FROM Table`

FROM

`DELETE FROM Table WHERE colonne = condition`

WHERE



La gestion des données sous Oracle

Exemples de suppressions d'enregistrements

```
DELETE FROM regions WHERE region_id = 1;  
DELETE regions;
```



La gestion des données sous Oracle

La validation des modifications

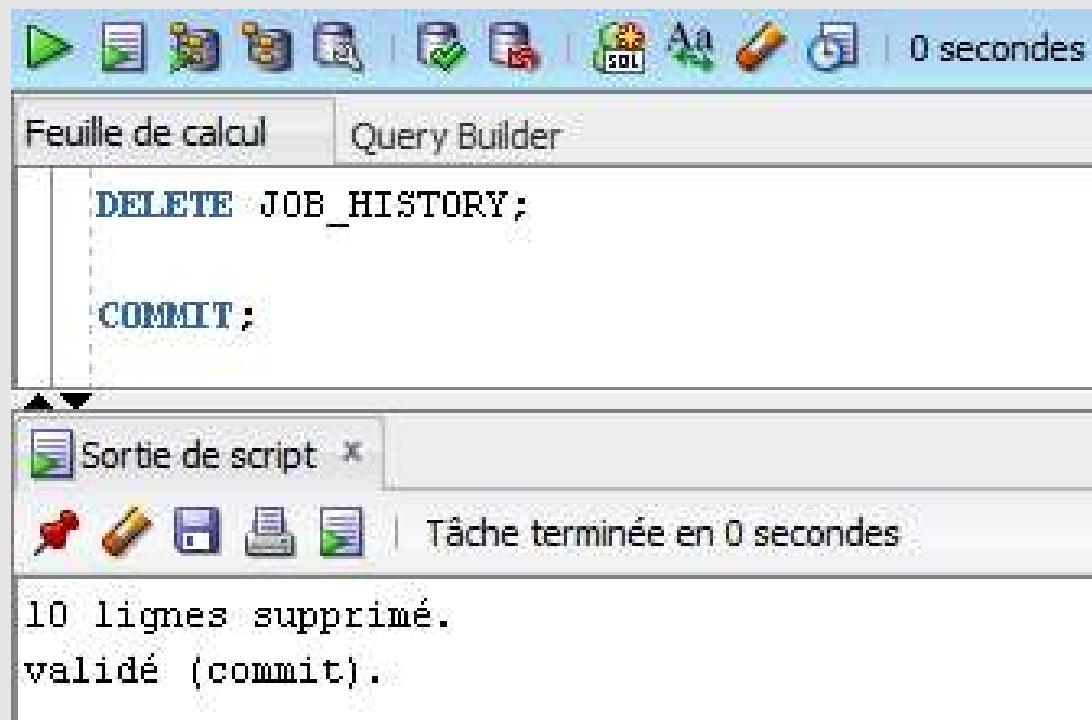
Enregistre en base toutes les insertions, modifications et suppressions réalisées depuis le début de la transaction.

Tant qu'il n'y a pas eu COMMIT, seule la connexion courante voit ses mises à jour.



La gestion des données sous Oracle

La validation des modifications



The screenshot shows the Oracle SQL Developer interface. The top toolbar has various icons for database operations like execute, refresh, and search. The status bar indicates "0 secondes". Below the toolbar, there are two tabs: "Feuille de calcul" (Calculation Sheet) and "Query Builder". The "Feuille de calcul" tab contains the following SQL code:

```
DELETE FROM JOB_HISTORY;  
COMMIT;
```

Below the code, a message window titled "Sortie de script" (Script Output) displays the results of the execution:

```
Tâche terminée en 0 secondes  
10 lignes supprimé.  
validé (commit).
```

La gestion des données sous Oracle

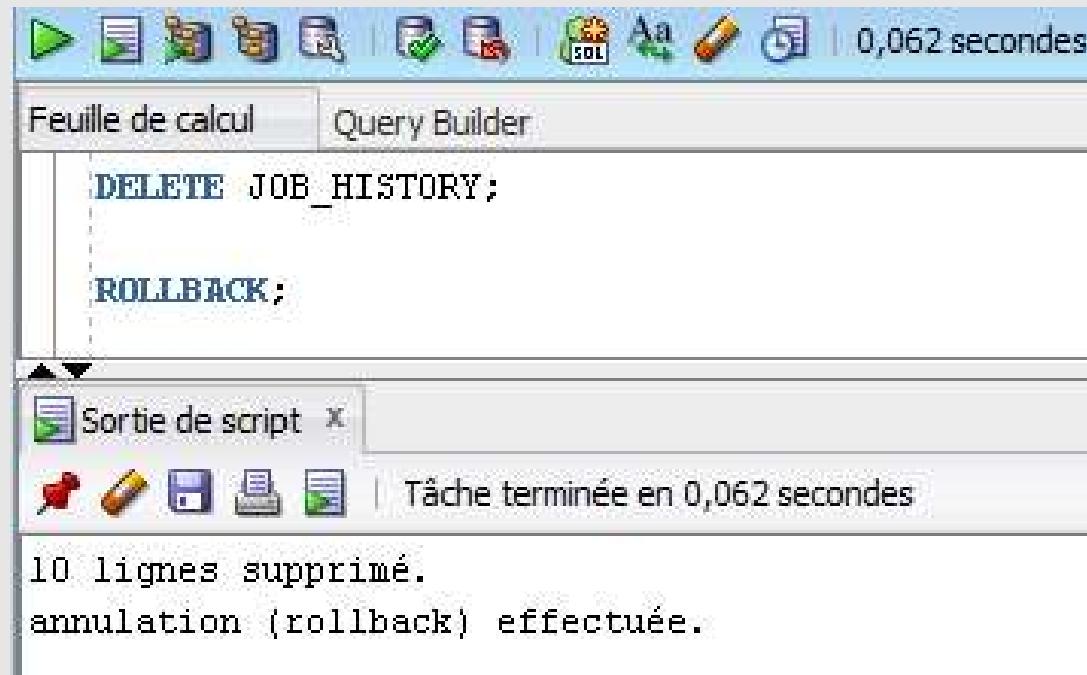
L'invalidation des modifications

Annule toutes les insertions, modifications et suppressions réalisées depuis le début de la transaction.



La gestion des données sous Oracle

L'invalidation des modifications



The screenshot shows the Oracle SQL Developer interface. The top menu bar includes icons for file, edit, search, and various database operations. The title bar indicates the current tab is "Feuille de calcul" (Query Builder). The main area contains the following SQL code:

```
DELETE FROM JOB_HISTORY;  
ROLLBACK;
```

Below the code, a "Sortie de script" (Script Output) window is open, showing the results of the execution:

Tâche terminée en 0,062 secondes

10 lignes supprimé.
annulation (rollback) effectuée.

Conclusion

- Vous savez modifier des données sous Oracle
- Vous savez valider vos modifications
- Vous savez annuler vos modifications



PL / SQL

Module de rappel – l'extraction des données sous Oracle



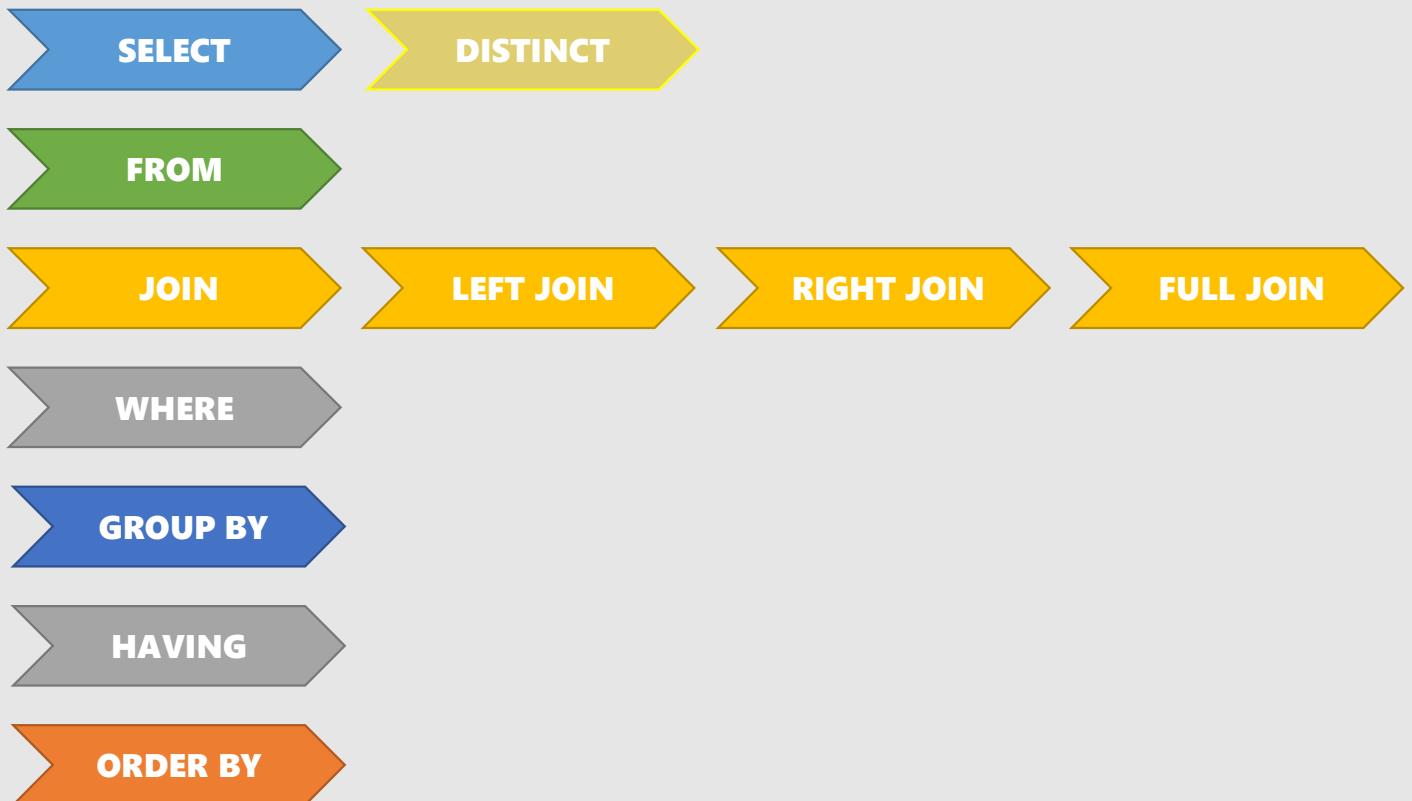
Objectifs

- Rappel sur les extractions de données avec le langage SQL
- Mise en avant des spécificités Oracle
- Découverte des fonctions essentielles Oracle



L'extraction des données sous Oracle

L'instruction SELECT



L'extraction des données sous Oracle

La projection et la restriction

SELECT

`SELECT * FROM Table`

FROM

`SELECT colonne_1, colonne_2 FROM Table`

SELECT

`SELECT * FROM Table WHERE colonne_1 = valeur`

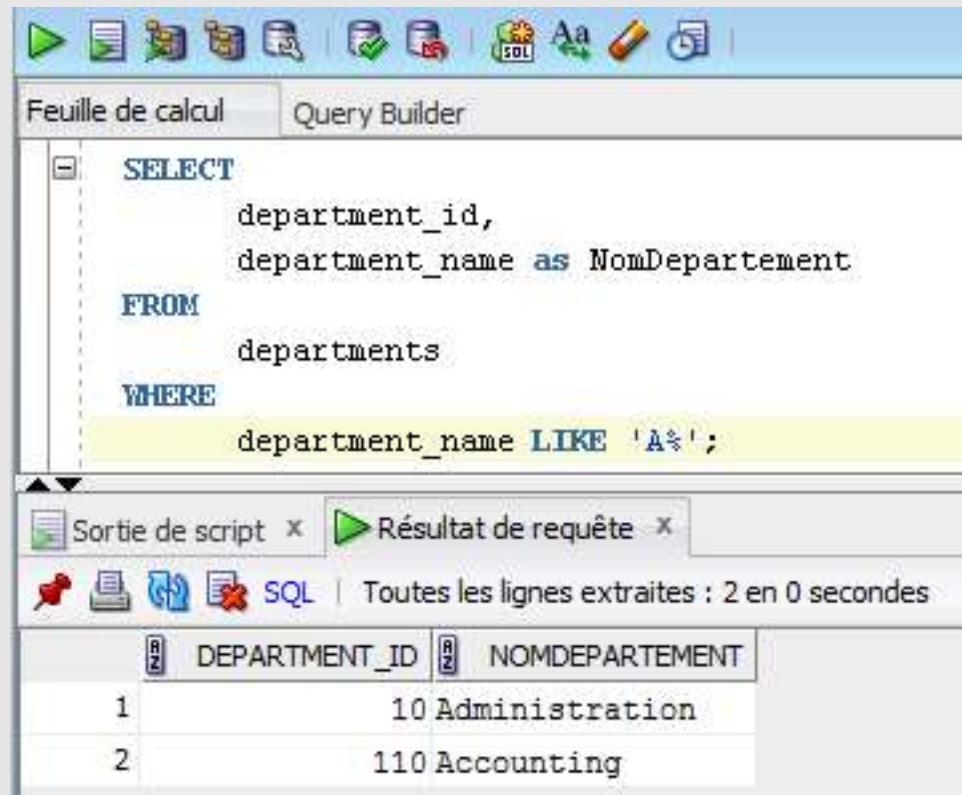
FROM

`SELECT colonne_1 * 2 AS alias FROM Table`

WHERE

L'extraction des données sous Oracle

Exemple de projection et de restriction



The screenshot shows the Oracle SQL Developer interface. The top menu bar includes icons for running, saving, and navigating. Below the menu is a toolbar with various tools. The main window has two tabs: "Feuille de calcul" (Calculation Sheet) and "Query Builder". The "Query Builder" tab is active, displaying the following SQL code:

```
SELECT
    department_id,
    department_name AS NomDepartement
FROM
    departments
WHERE
    department_name LIKE 'A%';
```

The "Résultat de requête" (Query Result) tab is selected at the bottom, showing the output of the query:

	DEPARTMENT_ID	NOMDEPARTEMENT
1	10	Administration
2	110	Accounting

A status message at the bottom of the result tab indicates: "Toutes les lignes extraites : 2 en 0 secondes" (All rows extracted: 2 in 0 seconds).

L'extraction des données sous Oracle

Les calculs d'agrégats

SELECT

```
SELECT MIN(colonne_1) AS minimum,  
       MAX(colonne_1) AS maximum,  
       AVG(colonne_1) AS moyenne  
FROM Table
```

FROM

Table	colonne_1	colonne_2
	2	A
	4	A
	1	B
	6	A
	5	B

Res	minimum	maximum	moyenne
	1	6	3

L'extraction des données sous Oracle

Les fonctions d'agrégation

AVG ()

MIN ()

MAX ()

COUNT ()

SUM ()



L'extraction des données sous Oracle

Exemple de calculs d'agrégats

The screenshot shows the Oracle SQL Developer interface. The top menu bar includes icons for running queries, saving, and navigating. Below the menu is a toolbar with various buttons. The main workspace has two tabs: "Feuille de calcul" (Calculation Sheet) and "Query Builder". The "Query Builder" tab is active, displaying the following SQL query:

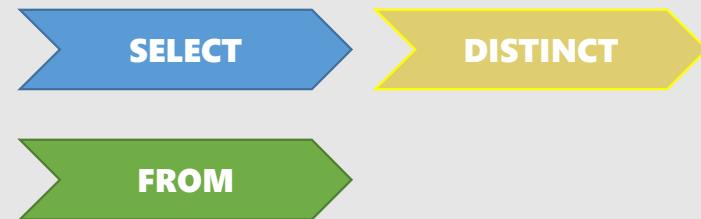
```
SELECT
    COUNT(*) as "nombre total"
FROM
    departments
WHERE
    department_name LIKE 'A%';
```

Below the query editor is a toolbar with icons for script output, results, and other functions. The status bar at the bottom indicates: "Sortie de script x Résultat de requête x" and "Toutes les lignes extraites : 1 en 0 secondes". The results window displays the following table:

nombre total
1
2

L'extraction des données sous Oracle

Le regroupement des lignes : DISTINCT



`SELECT DISTINCT colonne_1 FROM Table`

Table	colonne_1
	A
	A
	B
	A
	B

Res	colonne_1
	A
	B

L'extraction des données sous Oracle

Exemple d'utilisation de la clause DISTINCT

SANS

The screenshot shows the Oracle SQL Developer interface. The top menu bar includes icons for file operations, search, and database connections. Below the menu is a toolbar with various icons. The main workspace is titled "Feuille de calcul" and contains a "Query Builder" tab. A query is displayed:

```
SELECT
    e.last_name
FROM
    employees e
WHERE
    e.last_name IN('King','Smith');
```

Below the query, there are two tabs: "Sortie de script" and "Résultat de requête". The "Résultat de requête" tab shows the output:

LAST_NAME
1 King
2 King
3 Smith
4 Smith

At the bottom of the interface, status information reads: "Toutes les lignes extraites : 4 en 0,015 secondes".

AVEC

The screenshot shows the same Oracle SQL Developer interface. The top menu bar and toolbar are identical. The main workspace shows the same query as the first screenshot, but with the word "DISTINCT" added before the column name:

```
SELECT
    DISTINCT e.last_name
FROM
    employees e
WHERE
    e.last_name IN('King','Smith');
```

Below the query, there are two tabs: "Sortie de script" and "Résultat de requête". The "Résultat de requête" tab shows the output:

LAST_NAME
1 King
2 Smith

At the bottom of the interface, status information reads: "Toutes les lignes extraites : 2 en 0 secondes".

L'extraction des données sous Oracle

Le regroupement des lignes : GROUP BY

SELECT

FROM

GROUP BY

SELECT

MIN(colonne_1) AS minimum,
colonne_2

FROM

Table

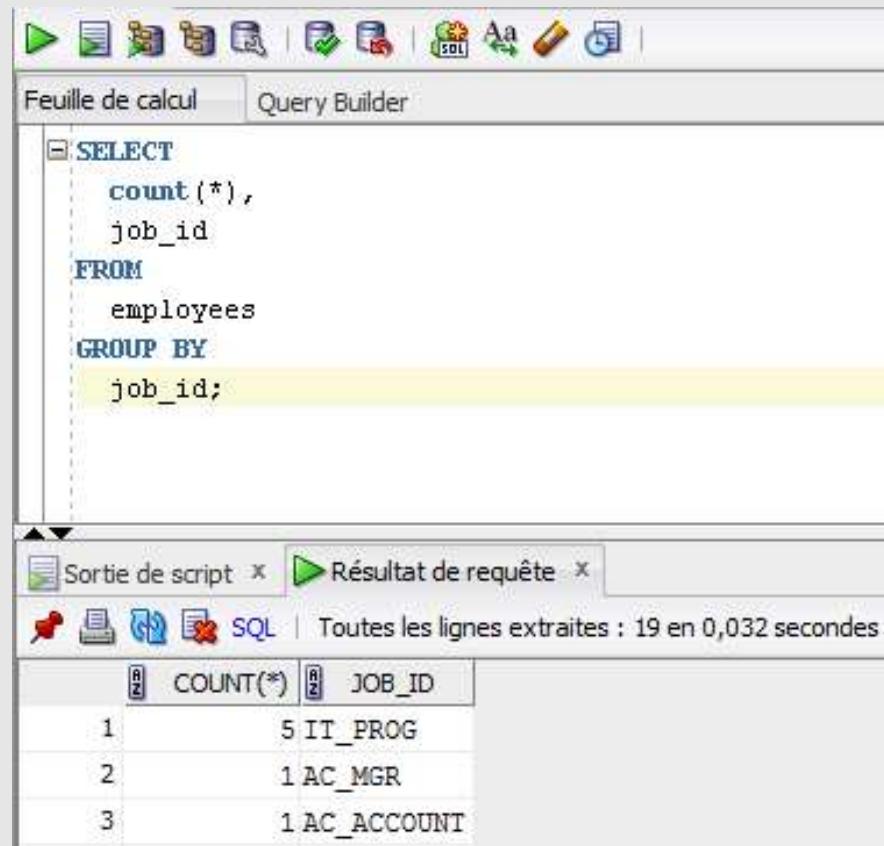
GROUP BY colonne_2

Table	colonne_1	colonne_2	Res	minimum	colonne_2
	2	A		2	A
	4	A		1	B
	1	B			
	6	A			
	5	B			

The diagram illustrates the grouping process. The original table has five rows. The first two rows (values 2 and 4) are grouped together under column A. The last two rows (values 1 and 5) are grouped together under column B. The result table shows the minimum value for each group (2 for group A, 1 for group B) and the corresponding group identifier (A and B respectively).

L'extraction des données sous Oracle

Exemple d'utilisation de la clause GROUP BY



The screenshot shows the Oracle SQL Developer interface. The top window is titled "Query Builder" and contains the following SQL code:

```
SELECT
    count(*),
    job_id
FROM
    employees
GROUP BY
    job_id;
```

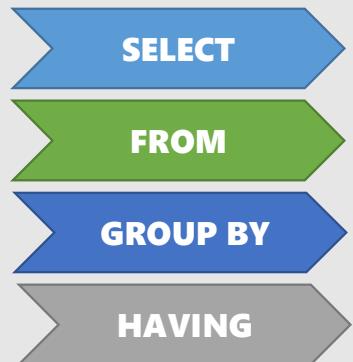
The bottom window is titled "Résultat de requête" (Result of query) and displays the following data:

	COUNT(*)	JOB_ID
1	5	IT_PROG
2	1	AC_MGR
3	1	AC_ACCOUNT

Annotations: The "GROUP BY" clause in the query and the "JOB_ID" column in the results table are highlighted with a yellow background.

L'extraction des données sous Oracle

La restriction sur agrégat



```
SELECT MIN(colonne_1) AS minimum, colonne_2  
FROM Table  
GROUP BY colonne_2  
HAVING COUNT(*) > 2 AND colonne_2 IN ('A', 'B')
```

Table	colonne_1	colonne_2	Res	minimum	colonne_2
	2	A		2	A
	4	A			
	1	B			
	6	A			
	5	B			
	3	C			

The diagram illustrates the execution of the query. Red boxes highlight specific rows in the 'Table' and 'Res' columns. Arrows point from the highlighted rows in the 'Table' to the corresponding row in the 'Res' column.

L'extraction des données sous Oracle

Exemple de restriction sur agrégat

The screenshot shows the Oracle SQL Developer interface. The top window is titled "Feuille de calcul" (Query Builder) and contains the following SQL code:

```
SELECT
    count(*),
    job_id
FROM
    employees
GROUP BY
    job_id
HAVING
    count(*) > 10;
```

The bottom window is titled "Sortie de script" (Script Output) and "Résultat de requête" (Query Result). It displays the results of the executed query:

	COUNT(*)	JOB_ID
1	20	SH_CLERK
2	30	SA_REP
3	20	ST_CLERK

Below the table, a status message reads: "Toutes les lignes extraites : 3 en 0,032 secondes".

L'extraction des données sous Oracle

La jointure



```
SELECT Table1.colonne_1, colonne_3  
FROM Table1  
JOIN Table2 ON Table1.colonne_1 = Table2.colonne_1
```

Table1	colonne_1	colonne_2
	1	A
	2	A
	4	B

Table2	colonne_1	colonne_3	colonne_4
	1	A1	XXXX
	3	A1	XXXX
	2	A1	XXXX
	1	A2	XXXX
	3	A2	XXXX
	1	A3	XXXX

Res	colonne_1	colonne_3
	1	A1
	2	A1
	1	A2
	1	A3

L'extraction des données sous Oracle

Exemple de jointure

The screenshot shows the Oracle SQL Developer interface. The top section is a 'Query Builder' window titled 'Feuille de calcul' (Calculation Sheet). It contains the following SQL code:

```
SELECT
    first_name,
    job_title
FROM
    employees e
JOIN jobs j ON j.job_id = e.job_id;
```

The 'JOIN' statement is highlighted with a yellow background. Below the Query Builder is a toolbar with icons for Sortie de script (Script Output), Résultat de requête (Query Result), and other database operations. The status bar indicates 'SQL | 50 lignes extraites en 0,094 secondes' (50 rows extracted in 0.094 seconds).

The bottom section is a 'Résultat de requête' (Query Result) grid displaying the extracted data:

	FIRST_NAME	JOB_TITLE
1	Steven	President
2	Neena	Administration Vice President
3	Lex	Administration Vice President
4	Alexander	Programmer
5	Bruce	Programmer

L'extraction des données sous Oracle

La jointure externe droite

SELECT
FROM
RIGHT JOIN

```
SELECT Table1.colonne_1, colonne_3  
FROM Table1  
RIGHT JOIN Table2 ON Table1.colonne_1 = Table2.colonne_1
```

Table1	colonne_1	colonne_2
	1	A
	2	A
	4	B

Table2	colonne_1	colonne_3	colonne_4
	1	A1	XXXX
	3	A1	XXXX
	2	A1	XXXX
	1	A2	XXXX
	3	A2	XXXX
	1	A3	XXXX

Res	colonne_1	colonne_3
	1	A1
	2	A1
	1	A2
	1	A3
	3	A1
	3	A2

L'extraction des données sous Oracle

Exemple de jointure externe droite

Feuille de calcul Query Builder

```
SELECT
    first_name,
    job_title
FROM
    employees e
JOIN jobs j ON e.job_id = j.job_id;
```

Sortie de script x Résultat de requête x

SQL | 50 lignes extraites en 0 secondes

FIRST_NAME	JOB_TITLE
1 Steven	President
2 Neena	Administration Vice President
3 Lex	Administration Vice President
4 Alexander	Programmer
5 Bruce	Programmer

Feuille de calcul Query Builder

```
SELECT
    first_name,
    job_title
FROM
    employees e
RIGHT JOIN jobs j ON e.job_id = j.job_id;
```

Sortie de script x Résultat de requête x

SQL | 50 lignes extraites en 0 secondes

FIRST_NAME	JOB_TITLE
1 William	Public Accountant
2 Shelley	Accounting Manager
3 Jennifer	Administration Assistant
4 (null)	Fictive job
5 Steven	President
6 Lex	Administration Vice President

L'extraction des données sous Oracle

La jointure externe gauche

SELECT
FROM
LEFT JOIN

```
SELECT Table1.colonne_1, colonne_3  
FROM Table1  
LEFT JOIN Table2 ON Table1.colonne_1 = Table2.colonne_1
```

Table1	Attribut_1	Attribut_2	Table2	Attribut_1	Attribut_3	Attribut_4	Res	Attribut_1	Attribut_3
	1	A		1	A1	XXXX		1	A1
	2	A		3	A1	XXXX		2	A1
	4	B		2	A1	XXXX		1	A2
				1	A2	XXXX		1	A3
				3	A2	XXXX		4	NULL
				1	A3	XXXX			

L'extraction des données sous Oracle

Exemple de jointure externe gauche

Feuille de calcul Query Builder

```
SELECT
    first_name,
    job_title
FROM
    employees e
LEFT JOIN jobs j ON e.job_id = j.job_id;
```

Sortie de script x Résultat de requête x

SQL | Toutes les lignes extraites : 108 en 0,078 secondes

	FIRST_NAME	JOB_TITLE
103	Pat	Marketing Representative
104	Susan	Human Resources Representa...
105	Hermann	Public Relations Represent...
106	Shelley	Accounting Manager
107	William	Public Accountant
108	Anthony	(null)

L'extraction des données sous Oracle

La jointure externe complète

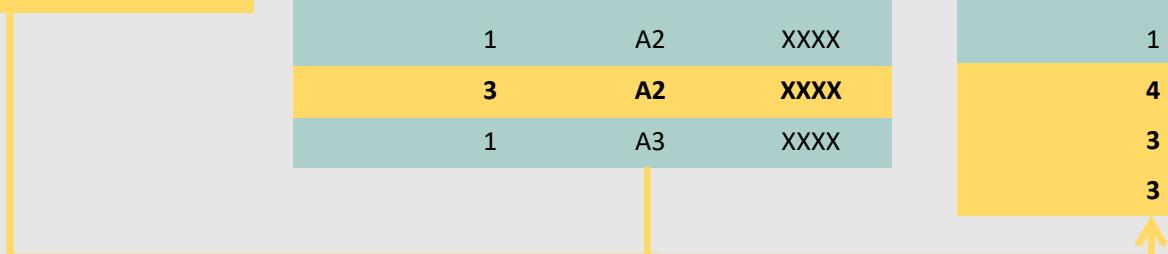


```
SELECT Table1.attribut_1, attribut_3  
FROM Table1  
FULL JOIN Table2 ON Table1.attribut_1 = Table2.attribut_1
```

Table1	Attribut_1	Attribut_2
1	A	
2	A	
4	B	

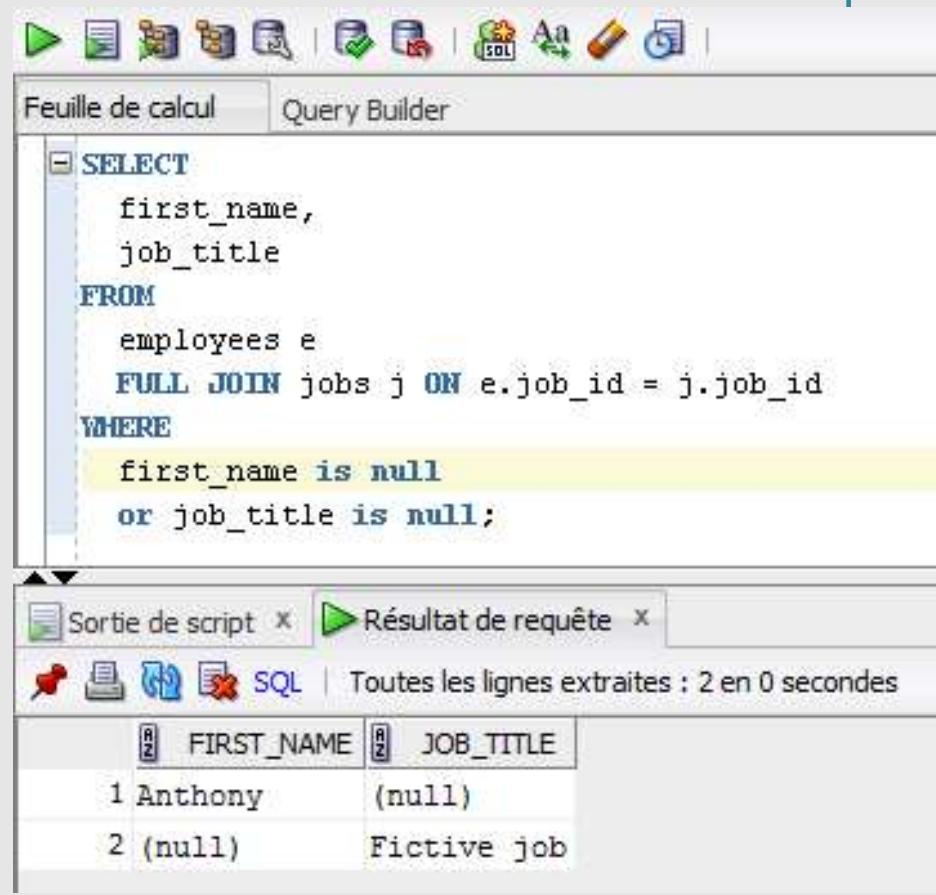
Table2	Attribut_1	Attribut_3	Attribut_4
1	A1	XXXX	
3	A1	XXXX	
2	A1	XXXX	
1	A2	XXXX	
3	A2	XXXX	
1	A3	XXXX	

Res	Attribut_1	Attribut_3
	1	A1
	2	A1
	1	A2
	1	A3
4	NULL	
3	A1	
3	A2	



L'extraction des données sous Oracle

Exemple de jointure externe complète



The screenshot shows the Oracle SQL Developer interface. The top menu bar includes icons for running queries, saving, and navigating. Below the menu is a toolbar with various tools. The main window has two tabs: "Feuille de calcul" (Calculation Sheet) and "Query Builder". The "Query Builder" tab is active, displaying the following SQL query:

```
SELECT
    first_name,
    job_title
FROM
    employees e
    FULL JOIN jobs j ON e.job_id = j.job_id
WHERE
    first_name is null
    or job_title is null;
```

Below the query editor is a toolbar with icons for script output, results, and other functions. The status bar indicates "Toutes les lignes extraites : 2 en 0 secondes" (All rows extracted: 2 in 0 seconds). The results grid displays the following data:

	FIRST_NAME	JOB_TITLE
1	Anthony	(null)
2	(null)	Fictive job

L'extraction des données sous Oracle

Le tri



```
SELECT colonne_1,  
       colonne_2  
  FROM Table  
 ORDER BY colonne_1 ASC [DESC]
```

The diagram illustrates a data transformation process. On the left, there is a table labeled 'Table' with columns 'Attribut_1' and 'Attribut_2'. The data is as follows:

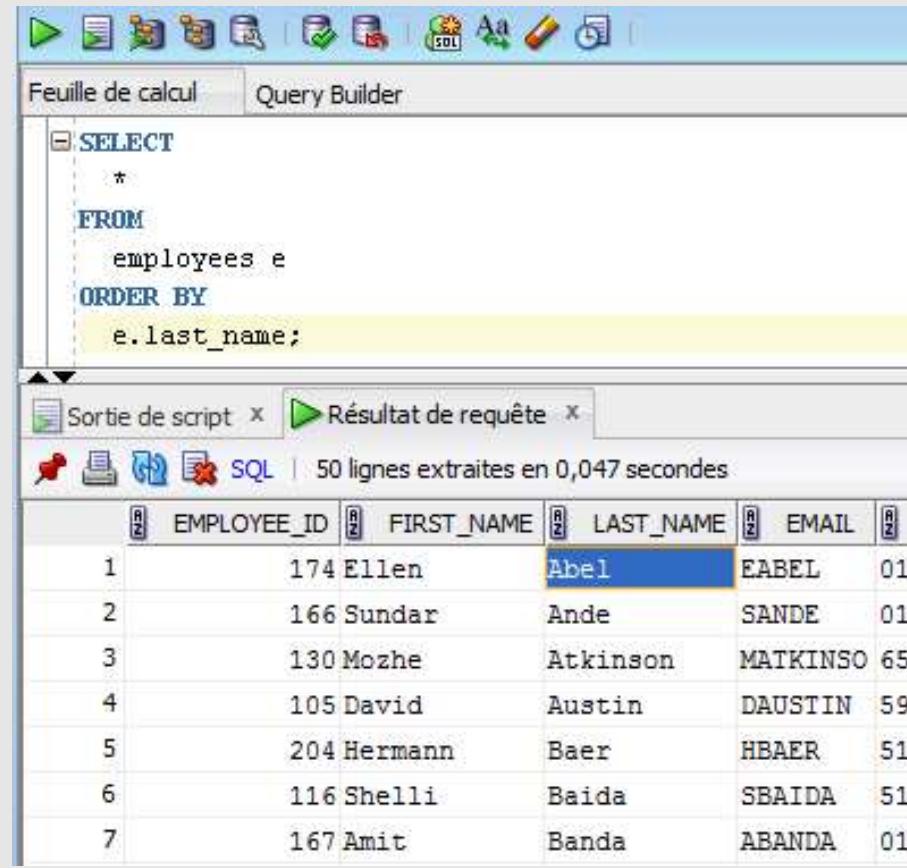
Table	Attribut_1	Attribut_2
	2	A
	4	A
	1	B
	6	A
	5	B

An orange arrow points from this table to a second table on the right, labeled 'Res' (short for Result). This table has columns 'Attribut_1' and 'Attribut_2'. The data is as follows:

Res	Attribut_1	Attribut_2
	1	B
	2	A
	4	A
	5	B
	6	A

L'extraction des données sous Oracle

Exemple de tri



The screenshot shows the Oracle SQL Developer interface. The top menu bar includes icons for running, saving, and connecting. Below the menu is a toolbar with various buttons. The main window has two tabs: "Feuille de calcul" (Calculation Sheet) and "Query Builder". The "Query Builder" tab is active, displaying the following SQL code:

```
SELECT
  *
FROM
  employees e
ORDER BY
  e.last_name;
```

Below the code, there are two tabs: "Sortie de script" (Script Output) and "Résultat de requête" (Query Result). The "Résultat de requête" tab is selected, showing the results of the query. The results are presented in a table with the following columns: EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, and PHONE_NUMBER. The data is sorted by LAST_NAME in ascending order. The first few rows of the result set are:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER
1	Ellen	Abel	EABEL	011
2	Sundar	Ande	SANDE	011
3	Mozhe	Atkinson	MATKINSO	650
4	David	Austin	DAUSTIN	590
5	Hermann	Baer	HBAER	515
6	Shelli	Baida	SBAIDA	515
7	Amit	Banda	ABANDA	011

L'extraction des données sous Oracle

La documentation Oracle

ORACLE Help Center Welcome Thierry▼

Home / Database / Oracle Database Online Documentation, 10g Release 2 (10.2) / Administration

Database SQL Reference

This Book This Release

Feedback Download Share to:

Categories

- Home
- Master Index
- Master Glossary

UPPER

Syntax

→ **UPPER** [(**char**)]

Description of the illustration upper.gif

Purpose

UPPER returns **char**, with all letters uppercase. **char** can be any of the datatypes **CHAR**, **VARCHAR2**, **NCHAR**, **NVARCHAR2**, **CLOB**, or **NCLOB**. The return value is the same datatype as **char**. The database sets the case of the characters based on the binary mapping defined for the underlying character set. For linguistic-sensitive uppercase, please refer to **NLS_UPPER**.

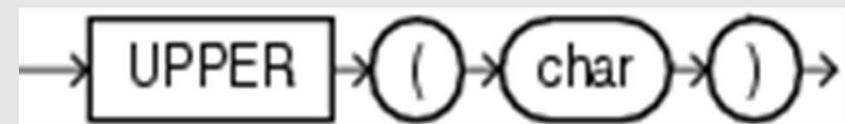
Examples

The following example returns each employee's last name in uppercase:

```
SELECT UPPER(last_name) "Uppercase"
  FROM employees;
```

L'extraction des données sous Oracle

Quelques fonctions essentielles Oracle



L'extraction des données sous Oracle

Quelques fonctions essentielles Oracle



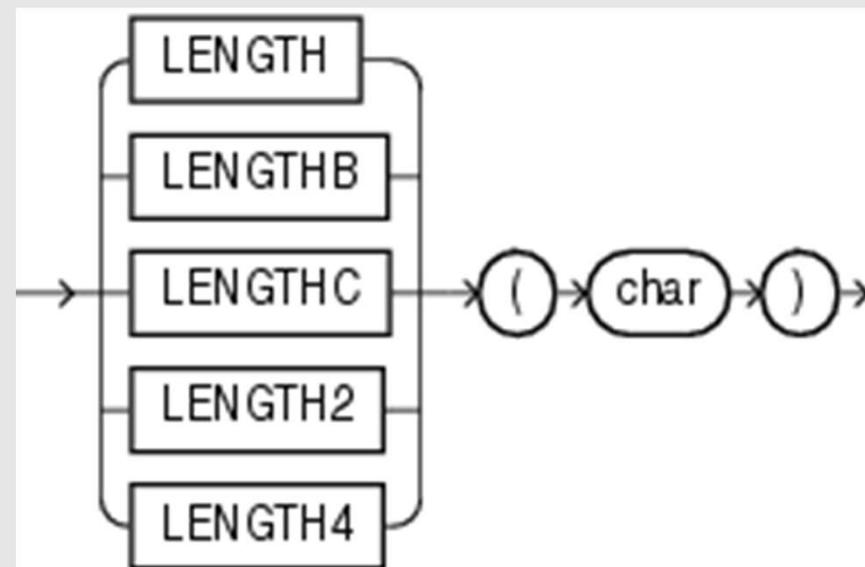
L'extraction des données sous Oracle

Quelques fonctions essentielles Oracle



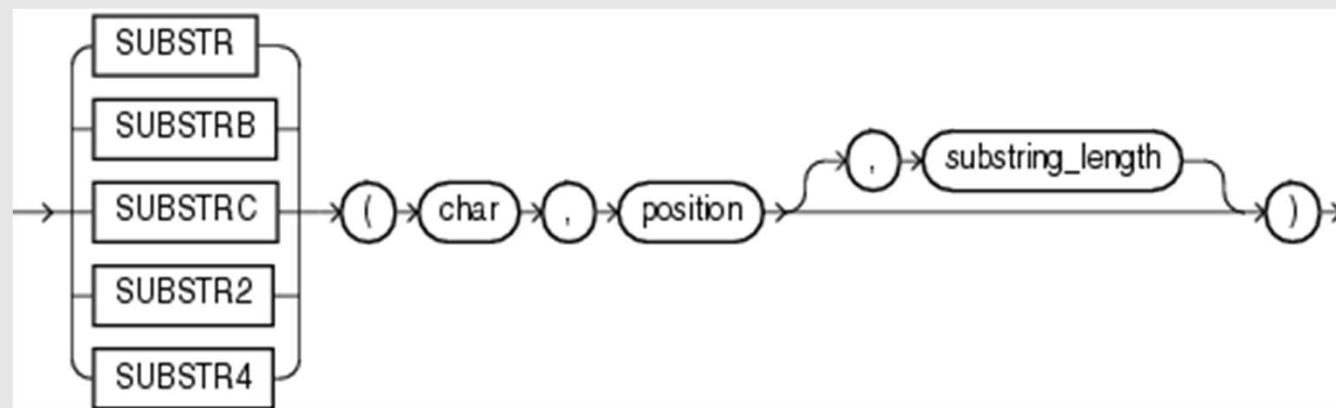
L'extraction des données sous Oracle

Quelques fonctions essentielles Oracle



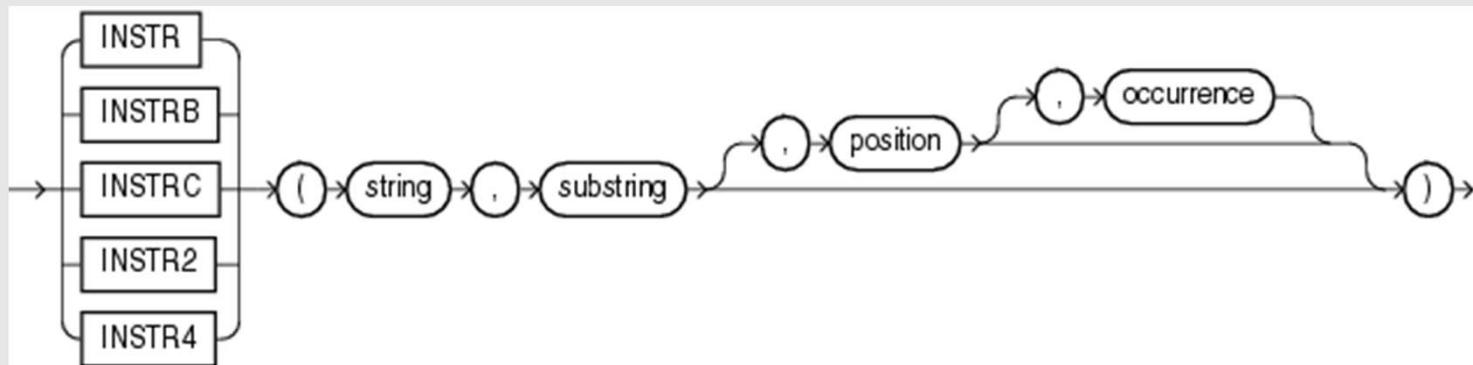
L'extraction des données sous Oracle

Quelques fonctions essentielles Oracle



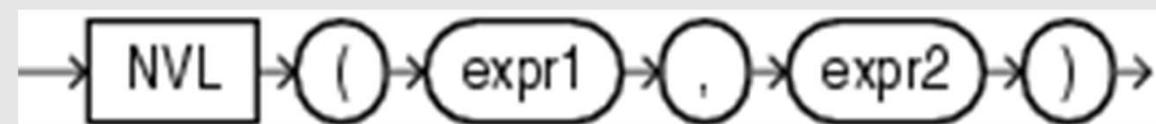
L'extraction des données sous Oracle

Quelques fonctions essentielles Oracle



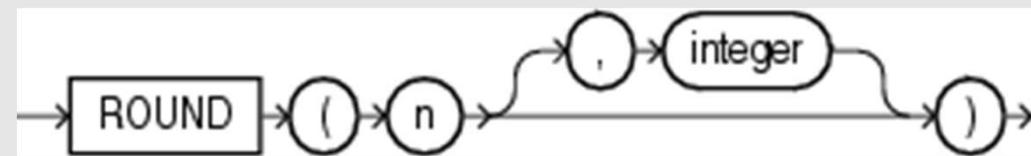
L'extraction des données sous Oracle

Quelques fonctions essentielles Oracle



L'extraction des données sous Oracle

Quelques fonctions essentielles Oracle



L'extraction des données sous Oracle

Quelques fonctions essentielles Oracle

→ **SYSDATE** →

L'extraction des données sous Oracle

Quelques fonctions essentielles Oracle



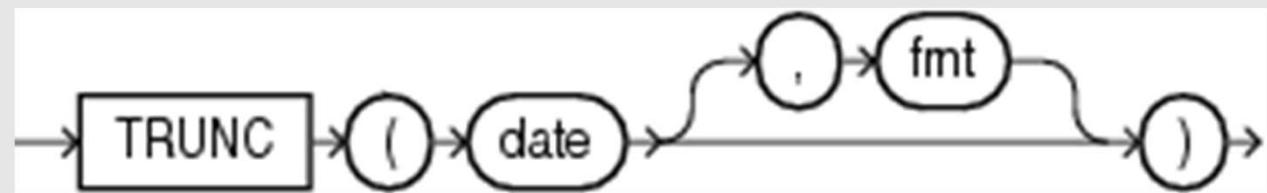
L'extraction des données sous Oracle

Quelques fonctions essentielles Oracle



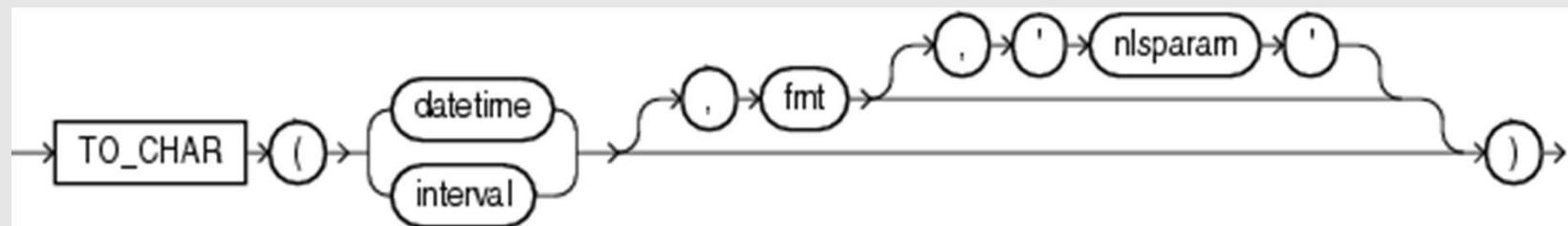
L'extraction des données sous Oracle

Quelques fonctions essentielles Oracle



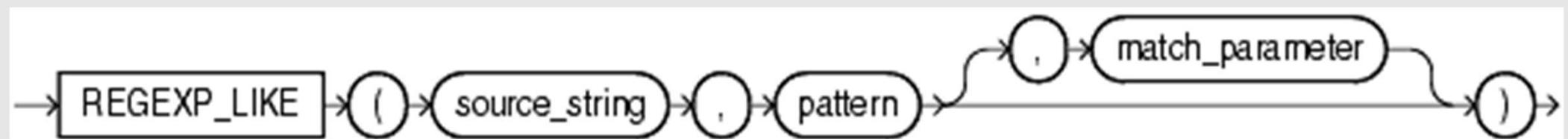
L'extraction des données sous Oracle

Quelques fonctions essentielles Oracle



L'extraction des données sous Oracle

Quelques fonctions essentielles Oracle



L'extraction des données sous Oracle

Quelques fonctions essentielles Oracle

Démonstration



Conclusion

- Vous savez extraire des données sous Oracle
- Vous avez découvert des fonctions courantes Oracle
- Vous avez découvert la syntaxe de la documentation Oracle



PL / SQL

Module 2 – Le langage PL / SQL



Objectifs

- Savoir expliquer ce qu'est le PL / SQL
- Comprendre l'intérêt du PL / SQL

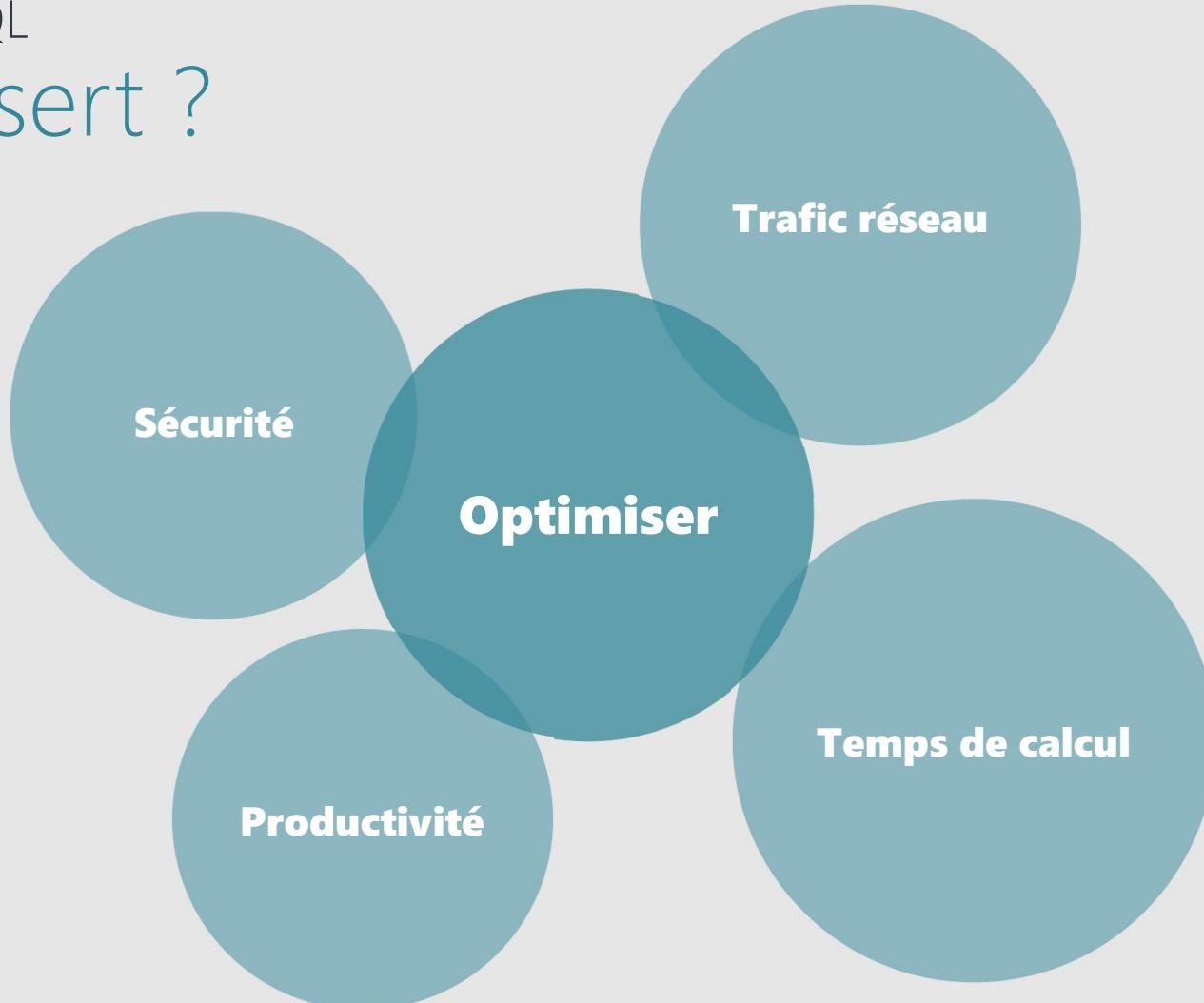


Le langage PL / SQL C'est quoi ?



Procedural Language / Structured Query Language

Le langage PL / SQL À quoi ça sert ?



Conclusion

- Vous savez que le PL/SQL est un langage procédural
- Vous savez que le PL/SQL s'exécute sur le serveur
- Vous savez que le PL/SQL permet d'accéder aux données de manière optimisée

PL / SQL

Module 3 – Les blocs PL/SQL



Objectifs

- Sensibilisation aux conventions de nommage
- Comprendre ce qu'est un bloc PL/SQL
- Connaître les différents types de blocs PL/SQL
- Découvrir le plus petit bloc PL/SQL du monde
- Connaître les sections possibles d'un bloc PL/SQL



Les blocs PL / SQL

La dette technique

Quand on code au plus vite et de manière non optimale, on contracte une dette technique que l'on rembourse tout au long de la vie du projet sous forme de temps de développement de plus en plus long et de bugs de plus en plus fréquents.



Les blocs PL / SQL

Les clés pour réduire la dette technique

```
nombre_adherents INT;
```

```
UPDATE  
    benevoles  
SET  
    actif = 1;
```

https://docs.oracle.com/cd/A97630_01/appdev.920/a96624/02_funds.htm



Les blocs PL / SQL

Définition d'un bloc PL/SQL



Les blocs PL / SQL

Le bloc interne

Envoi une requête avec le nom du bloc à exécuter



Le bloc stocké est exécuté sur le serveur

Les blocs PL / SQL

Le bloc externe

Envoi une requête avec le contenu du bloc à exécuter



Le bloc est exécuté sur le serveur

Les blocs PL / SQL

Le plus petit bloc du monde

```
BEGIN  
    NULL; --Traitement  
END;  
/
```

Les blocs PL / SQL

La section DECLARE

```
DECLARE  
    --Declaration des variables  
BEGIN  
    NULL; --Traitement  
END;  
/
```

Les blocs PL / SQL

La section EXCEPTION

```
DECLARE
    --Déclaration des variables
BEGIN
    NULL; --Traitement
EXCEPTION
    --Section de gestion des erreurs
END;
/
```

Conclusion

- Vous savez qu'un bloc PL/SQL est un programme
- Vous savez que les blocs PL/SQL sont exécutés sur le serveur de BDD
- Vous savez qu'un bloc associé à un nom est un bloc interne
- Vous savez qu'un bloc sans nom est un bloc externe ou anonyme
- Vous connaissez les sections DECLARE, BEGIN et EXCEPTION
- Vous savez que les sections DECLARE et EXCEPTION sont facultatives



PL / SQL

Module 4 – La section DECLARE



La section DECLARE

Objectifs

- Connaître les types simples de variables
- Savoir déclarer une variable



La section DECLARE

Les types de données

Tous les types SQL

BOOLEAN

PLS_INTEGER

Sous-types



La section DECLARE

La déclaration d'une variable

```
nom_de_la_variable [CONSTANT] TYPE [NOT NULL] [:= expression];
```



La section DECLARE

Des exemples de déclarations

```
DECLARE
    compteur PLS_INTEGER;
    maximum CONSTANT PLS_INTEGER := 500;
    resultat BOOLEAN NOT NULL := TRUE;
    prenom VARCHAR2(30) := 'Anthony';
BEGIN
    NULL; --Traitement
END;
/
```

La section DECLARE

Conclusion

- Vous connaissez les types simples de variables
- Vous savez déclarer une variable
- Vous avez découvert les types BOOLEAN et PLS_INTEGER
- Vous savez qu'il existe des sous-types
- Vous savez que vous pouvez créer vos propres sous types



PL / SQL

Module 5 – La section BEGIN



La section BEGIN

Objectifs

- Savoir utiliser des variables
- Savoir utiliser des structures de contrôle
- Savoir utiliser des types complexes
- Savoir afficher des messages pour débuguer



La section BEGIN

Les commentaires de code

```
/* Je suis  
    un commentaire sur  
    plusieurs lignes */
```

```
--Je suis un commentaire sur une ligne
```

La section BEGIN
L'affection

:=

INTO

```
nombre_de_mots := 14540;  
  
SELECT count(*) INTO total FROM regions;
```

La section BEGIN

Le package DBMS_OUTPUT

DBMS_OUTPUT pour déboguer

```
DBMS_OUTPUT.PUT_LINE('Valeur de la variable nommée code_agence : ' || code_agence);
```

La commande SET SERVEROUTPUT ON active les fonctions du package DBMS_OUTPUT.



La section BEGIN

Le traitement conditionnel IF ... THEN ... ELSE

```
DECLARE
    couleur_drapeau VARCHAR(30) := 'VERT';
BEGIN
    IF couleur_drapeau = 'ROUGE' THEN
        DBMS_OUTPUT.PUT_LINE('Baignade interdite');
    ELSIF couleur_drapeau = 'ORANGE' THEN
        DBMS_OUTPUT.PUT_LINE('Baignade dangereuse');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Baignade autorisée youhouuu ! :)');
    END IF;
END;
```

La section BEGIN

Le traitement conditionnel CASE avec valeur

```
DECLARE
    couleur_drapeau VARCHAR(30) := 'VERT';
BEGIN
    CASE couleur_drapeau
        WHEN 'ROUGE' THEN
            DBMS_OUTPUT.PUT_LINE('Baignade interdite !!!');
        WHEN 'ORANGE' THEN
            DBMS_OUTPUT.PUT_LINE('Attention, la mer est dangereuse');
        WHEN 'VERT' THEN
            DBMS_OUTPUT.PUT_LINE('Tous à l''eau !!!');
        WHEN 'NOIR' THEN
            DBMS_OUTPUT.PUT_LINE('Marée noire');
        ELSE
            DBMS_OUTPUT.PUT_LINE('Drapeau non repertorié');
    END CASE;
END;
```

La section BEGIN

Le traitement conditionnel CASE avec condition

```
DECLARE
    couleur_drapeau VARCHAR(30) := 'VERT';
BEGIN
    CASE
        WHEN couleur_drapeau = 'VERT' THEN
            DBMS_OUTPUT.PUT_LINE('Le drapeau est vert.');
        WHEN couleur_drapeau = 'ORANGE' THEN
            DBMS_OUTPUT.PUT_LINE('Le drapeau est orange');
        WHEN couleur_drapeau = 'ROUGE' THEN
            DBMS_OUTPUT.PUT_LINE('Le drapeau est rouge.');
        ELSE
            DBMS_OUTPUT.PUT_LINE('Je ne connais pas cette couleur.');
    END CASE;
END;
```

La section BEGIN

Le traitement répétitif LOOP

```
DECLARE
    index_loop INTEGER := 0;
BEGIN
    LOOP
        IF index_loop = 3 THEN
            EXIT;
        ELSE
            DBMS_OUTPUT.PUT_LINE(index_loop);
        END IF;
        index_loop := index_loop +1;
    END LOOP;
END;
```

La section BEGIN

Le traitement répétitif FOR

```
BEGIN
    FOR i IN 0..3 LOOP
        DBMS_OUTPUT.PUT_LINE(i);
    END LOOP;
END;
```

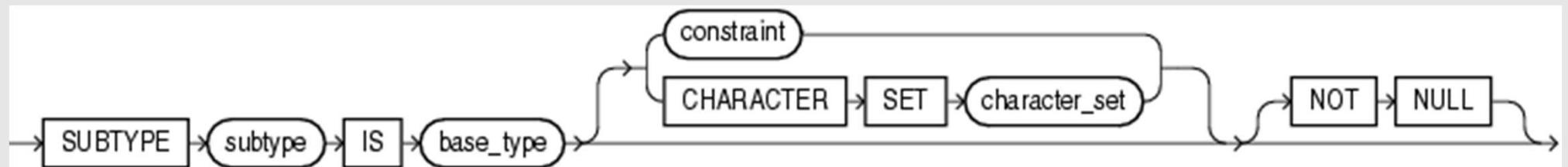
La section BEGIN

Le traitement répétitif WHILE

```
DECLARE
    index_while INTEGER := 0;
BEGIN
    WHILE (index_while < 3) LOOP
        DBMS_OUTPUT.PUT_LINE(index_while);
        index_while := index_while + 1;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('FIN : index_while = ' || index_while);
END;
```

La section BEGIN

Les sous-types



La section BEGIN

Les types définis par l'utilisateur

```
SUBTYPE nom_sous_type IS type_de_base[ (contrainte) ] [NOT NULL];
```



La section BEGIN

Exemple de types définis par l'utilisateur

```
SET SERVEROUTPUT ON;

DECLARE
    SUBTYPE salaire IS NUMBER(4);
    SUBTYPE date_nn IS DATE NOT NULL;

    premiere_annee salaire := 2000;
    premier_jour date_nn := SYSDATE;

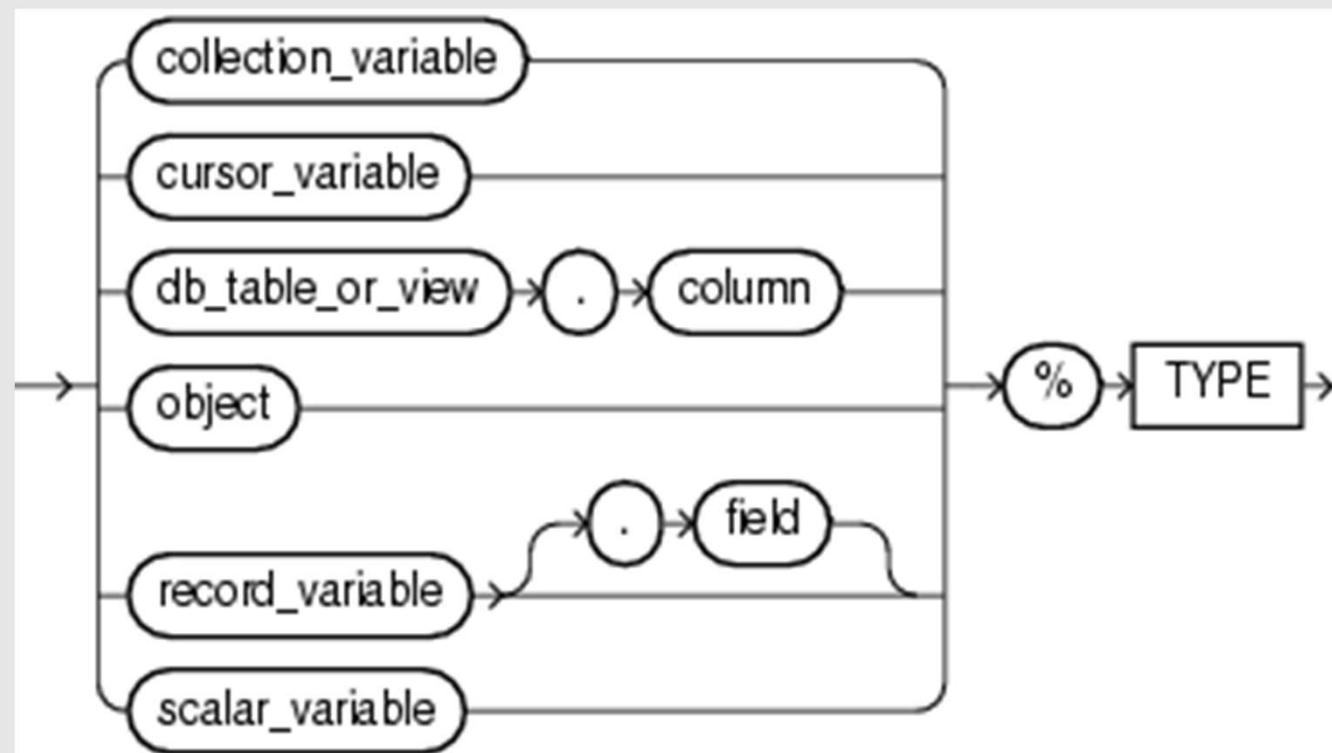
BEGIN

    DBMS_OUTPUT.PUT_LINE(premiere_annee);
    DBMS_OUTPUT.PUT_LINE(premier_jour);

END;
```

La section BEGIN

L'attribut %TYPE



La section BEGIN

L'attribut %TYPE

```
nom_variable nom_table.nom_colonne%TYPE;
```



La section BEGIN

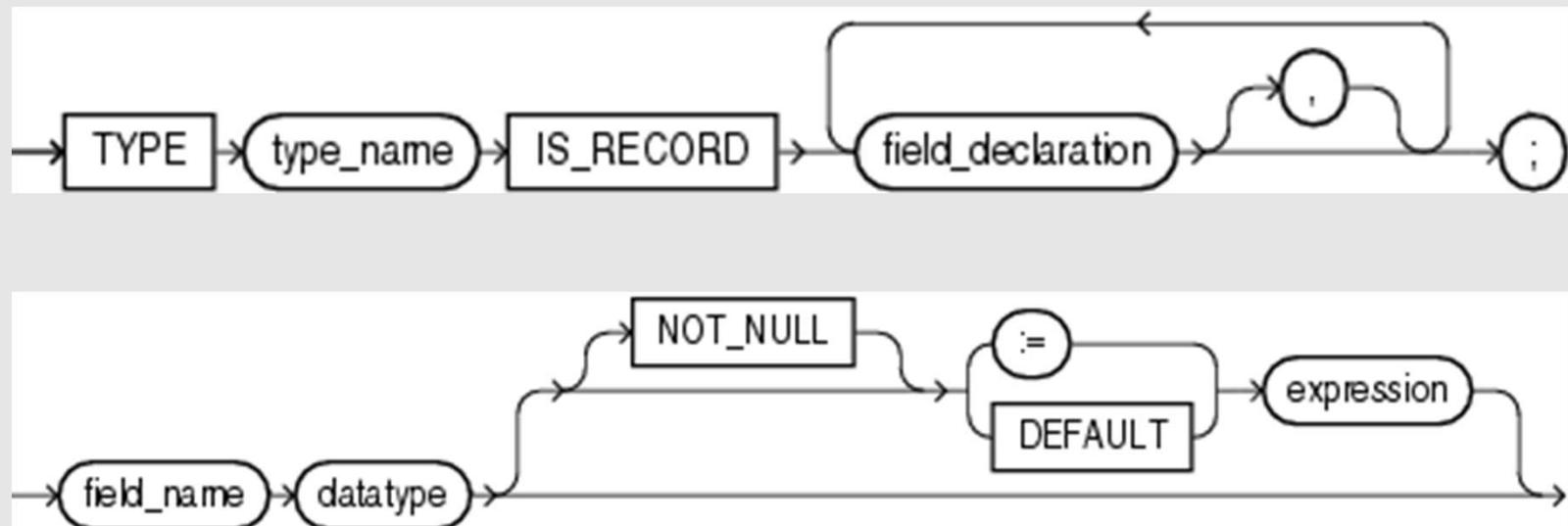
Exemple d'utilisation de l'attribut %TYPE

```
SET SERVEROUTPUT ON;

DECLARE
    continent regions.region_name%TYPE;
BEGIN
    continent := 'Europe';
    DBMS_OUTPUT.PUT_LINE(continent);
END;
```

La section BEGIN

Les enregistrements de type RECORD



La section BEGIN

Exemple d'utilisation du type RECORD

```
SET SERVEROUTPUT ON;

DECLARE

    TYPE fiche_parking IS RECORD
    (
        prenom VARCHAR2(50),
        nom VARCHAR2(50) NOT NULL := 'XXX',
        nombre NUMBER NOT NULL DEFAULT 1
    );

    fiche1 fiche_parking;

BEGIN

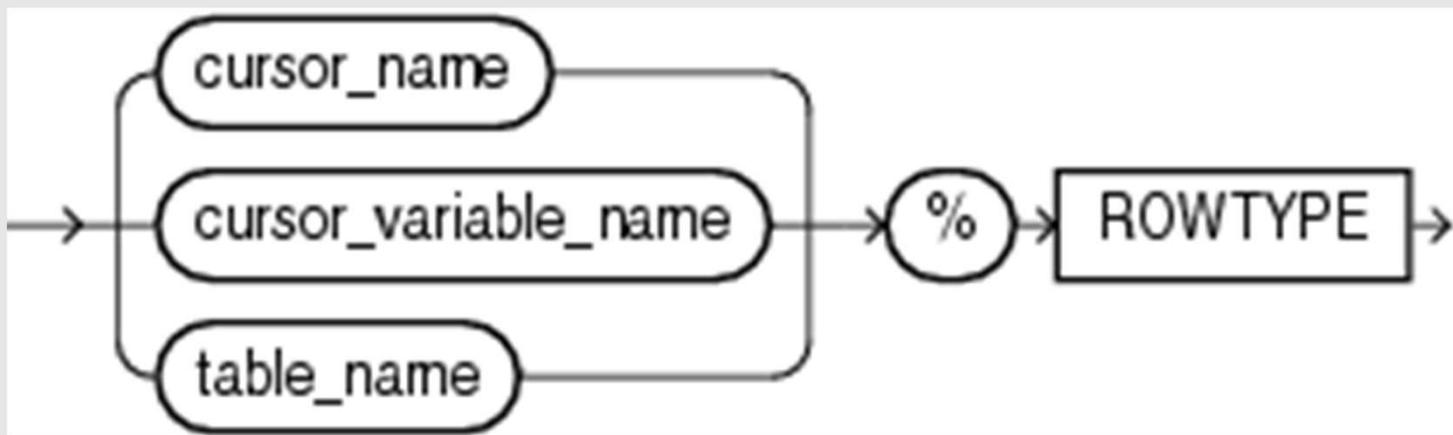
    fiche1.prenom := 'Anthony';
    fiche1.nom := 'Cosson';

    DBMS_OUTPUT.PUT_LINE('Prenom : ' || fiche1.prenom);
    DBMS_OUTPUT.PUT_LINE('Nom : ' || fiche1.nom);
    DBMS_OUTPUT.PUT_LINE('Nombre voiture : ' || fiche1.nombre);

END;
```

La section BEGIN

L'attribut %ROWTYPE

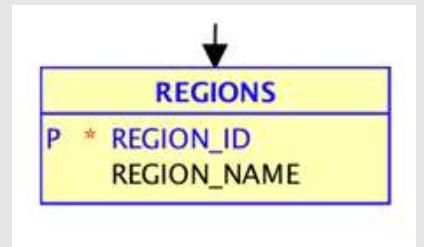


La section BEGIN

Exemple d'utilisation de l'attribut %ROWTYPE

```
SET SERVEROUTPUT ON;

DECLARE
    region_rec regions%ROWTYPE;
BEGIN
    region_rec.region_id = 5;
    region_rec.region_name = 'Antarctique';
    DBMS_OUTPUT.PUT_LINE('Id de la region : ' || region_rec.region_id);
    DBMS_OUTPUT.PUT_LINE('Nom de la region : ' || region_rec.region_name);
END;
```



La section BEGIN

Les collections

TYPE	Nombre d'éléments	Indexation	Doit être initialisé
INDEX BY TABLE (Tableau associatif (clé/valeur))	Illimité	Alphanumérique	Non
NESTED TABLE (Tableau imbriqué)	Illimité	Entier	Oui
VARRAY (Taille variable)	Limité	Entier	Non



La section BEGIN

La collection de type INDEX BY TABLE

Ensemble de paires clé-valeur

Clé numérique ou alpha numérique

Taille dynamique

Stockage temporaire de données



La section BEGIN

Exemple de collection INDEX BY TABLE

```
SET SERVEROUTPUT ON;

DECLARE

    TYPE population IS TABLE OF NUMBER INDEX BY VARCHAR(50);

    population_ville population;

    ville VARCHAR2(50);

BEGIN

    population_ville('Soliers') := 2151;
    population_ville('Caen') := 106538;
    population_ville('Limoges') := 134577;
    population_ville('Rennes') := 213454;

    ville := population_ville.FIRST;

    WHILE ville IS NOT NULL LOOP
        DBMS_Output.PUT_LINE('La population de ' || ville
        || ' est de ' || TO_CHAR(population_ville(ville)) || ' habitants');

        ville := population_ville.NEXT(ville);
    END LOOP;
END;
```

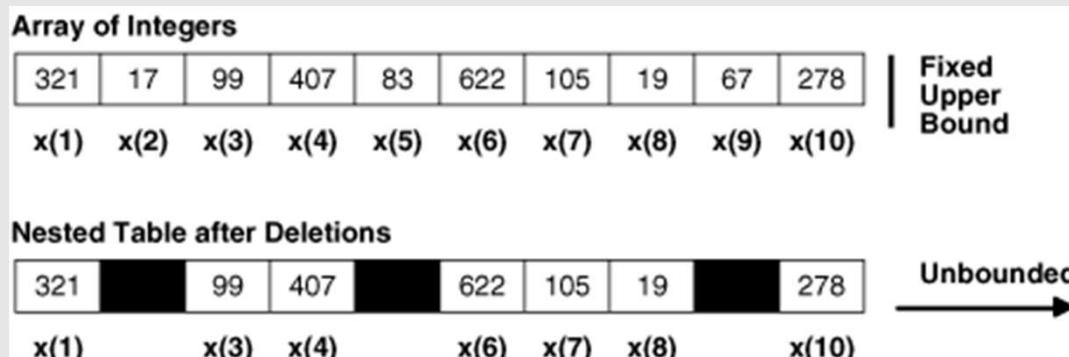
La section BEGIN

La collection de type NESTED TABLE

Taille dynamique

Index numérique

Données parsemées



La section BEGIN

Exemple de collection NESTED TABLE

```
SET SERVEROUTPUT ON;

DECLARE

    TYPE tableau_noms_objets IS TABLE OF VARCHAR2(50);

    tableau_musique tableau_noms_objets;

    objet_courant VARCHAR2(50);
BEGIN

    tableau_musique := tableau_noms_objets('Guitare','Tambour','Batterie','Basse');

    tableau_musique.DELETE(2);

    DBMS_OUTPUT.PUT_LINE(tableau_musique(1));
    --DBMS_OUTPUT.PUT_LINE(tableau_musique(2));
    DBMS_OUTPUT.PUT_LINE(tableau_musique(3));

END;
```

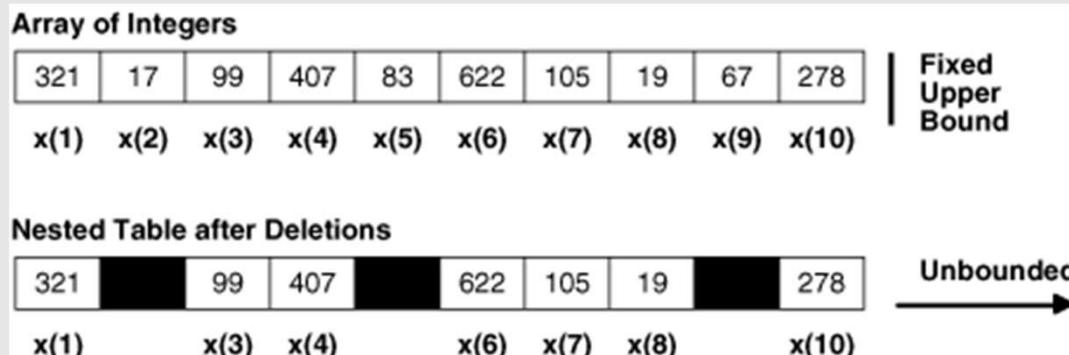
La section BEGIN

La collection de type VARRAY

Taille fixe

Index numérique

Données denses



La section BEGIN

Exemple de collection VARRAY

```
SET SERVEROUTPUT ON;

DECLARE

    TYPE tableau_couleur IS VARRAY(5) OF VARCHAR2(50);

    france tableau_couleur := tableau_couleur('Bleu','Blanc','Rouge');

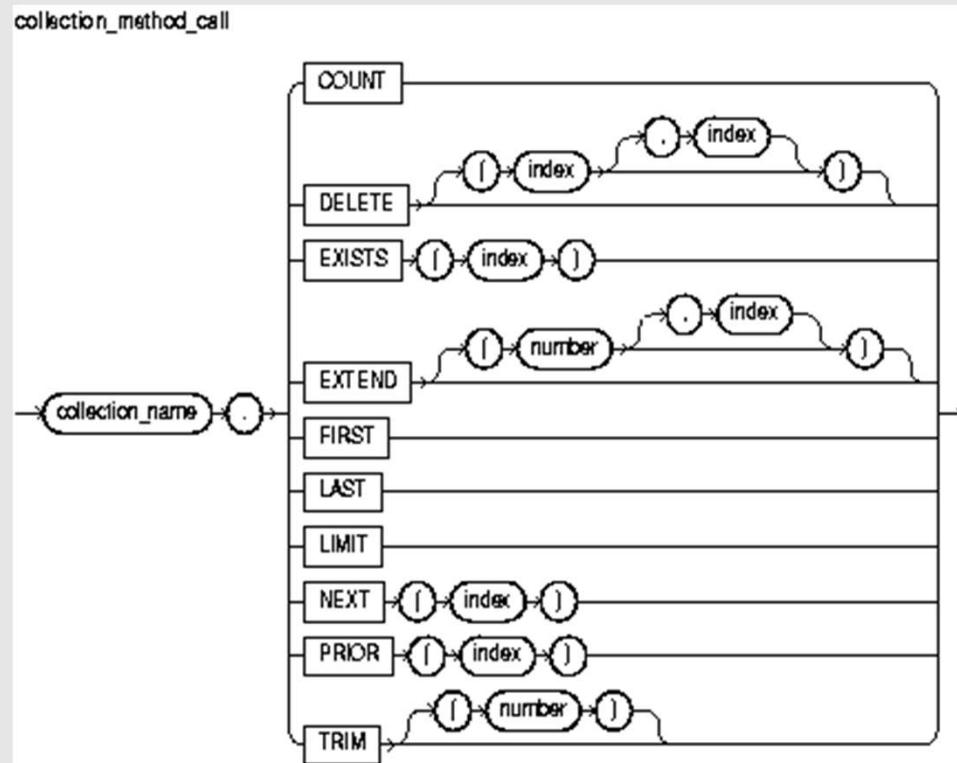
BEGIN

    DBMS_OUTPUT.PUT_LINE(france(1));
    DBMS_OUTPUT.PUT_LINE(france(2));
    DBMS_OUTPUT.PUT_LINE(france(3));

END;
```

La section BEGIN

Méthodes associées aux collections



La section BEGIN

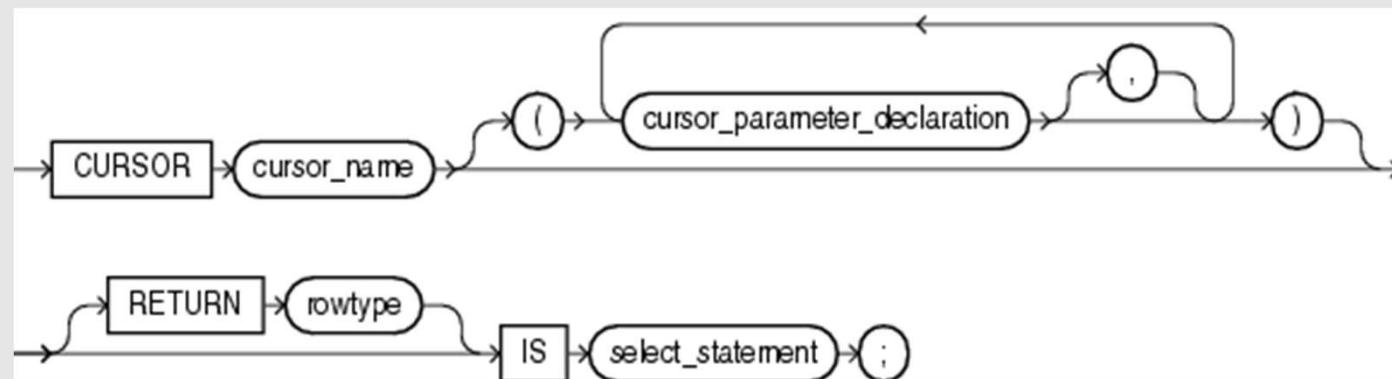
Le curseur explicite

Un curseur permet de stocker le résultat d'une requête
afin de le traiter ligne par ligne



La section BEGIN

La déclaration d'un curseur explicite



La section BEGIN

L'utilisation du curseur explicite : 1^{ère} méthode

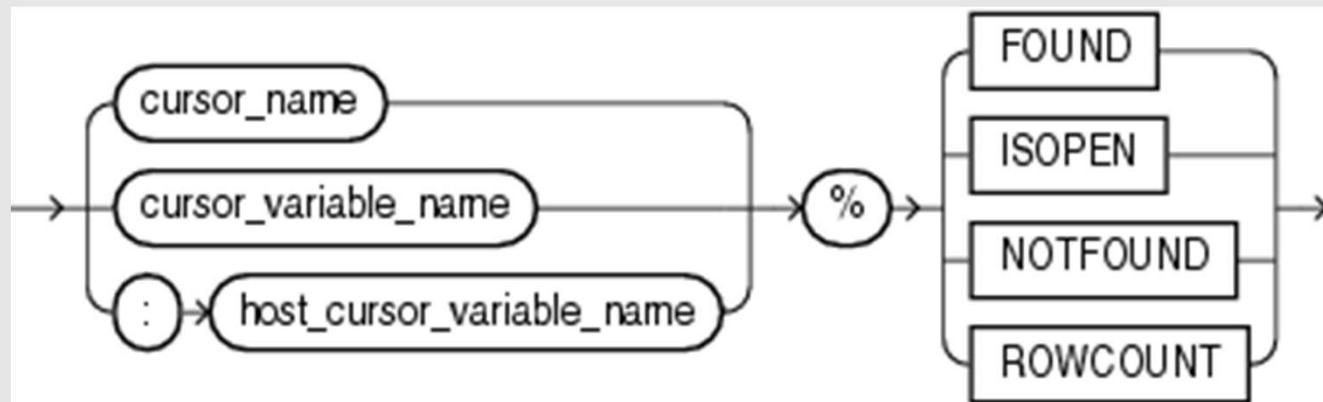
Fonctionnement :

1. Déclaration du curseur
2. Ouverture du curseur
3. Récupération du résultat ligne par ligne
4. Fermeture du curseur



La section BEGIN

Les attributs de curseur



La section BEGIN

L'utilisation du curseur explicite : 1ère méthode

```
SET SERVEROUTPUT ON;

DECLARE
    CURSOR cursor_employees_it IS SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID = 60;
    employee employees%ROWTYPE;
BEGIN
    OPEN cursor_employees_it;

    LOOP
        FETCH cursor_employees_it INTO employee;
        DBMS_OUTPUT.PUT_LINE(employee.last_name);
        EXIT WHEN cursor_employees_it%NOTFOUND;
    END LOOP;

    CLOSE cursor_employees_it;
END;
```

La section BEGIN

L'utilisation du curseur explicite : 2^{ème} méthode

FOR ... IN ... LOOP



La section BEGIN

L'utilisation du curseur explicite : 2ème méthode

```
SET SERVEROUTPUT ON;

DECLARE
    CURSOR cursor_employees_it IS SELECT first_name, last_name FROM EMPLOYEES WHERE DEPARTMENT_ID = 60;
    employee employees%ROWTYPE;
BEGIN

    FOR employee IN cursor_employees_it LOOP
        DBMS_OUTPUT.PUT_LINE(employee.first_name);
    END LOOP;
END;
```



La section BEGIN

Le verrou d'intention FOR UPDATE

Permet de verrouiller la table ou les champs que l'on va mettre à jour à partir du curseur implicite



La section BEGIN

Le verrou d'intention FOR UPDATE

```
SET SERVEROUTPUT ON;

DECLARE
    CURSOR cursor_employees_it IS SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID = 60 FOR UPDATE;
BEGIN

    FOR employee IN cursor_employees_it LOOP
        UPDATE employees SET last_name = UPPER(last_name) WHERE employee_id = employee.employee_id;
        DBMS_OUTPUT.PUT_LINE(employee.first_name || ' ' || employee.last_name);
    END LOOP;

    COMMIT;
END;
```



PL / SQL

Le verrou d'intention FOR UPDATE OF

```
SET SERVEROUTPUT ON;

DECLARE
  CURSOR cursor_employees_it IS SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID = 60 FOR UPDATE OF last_name;
BEGIN

  FOR employee IN cursor_employees_it LOOP
    UPDATE employees SET last_name = UPPER(last_name) WHERE employee_id = employee.employee_id;
    DBMS_OUTPUT.PUT_LINE(employee.first_name || ' ' || employee.last_name);
  END LOOP;

  COMMIT;
END;
```



La section BEGIN

La clause WHERE CURRENT OF du curseur explicite

Permet de simplifier la clause WHERE des requêtes de modification



La section BEGIN

Exemple d'utilisation de la clause WHERE CURRENT OF

```
SET SERVEROUTPUT ON;

DECLARE
    CURSOR cursor_employees_it IS SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID = 60 FOR UPDATE OF last_name;
BEGIN
    FOR employee IN cursor_employees_it LOOP
        UPDATE employees SET last_name = UPPER(last_name) WHERE CURRENT OF cursor_employees_it;
        DBMS_OUTPUT.PUT_LINE(employee.first_name || ' ' || employee.last_name);
    END LOOP;
    COMMIT;
END;
```



La section BEGIN

Le curseur explicite paramétrable

Un curseur peut être paramétrable



La section BEGIN

Exemple de curseur explicite paramétrable

```
SET SERVEROUTPUT ON;

DECLARE
    CURSOR cursor_employees_it(dep NUMBER) IS SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID = dep FOR UPDATE OF last_name;
BEGIN
    FOR employee IN cursor_employees_it(60) LOOP
        UPDATE employees SET last_name = UPPER(last_name) WHERE CURRENT OF cursor_employees_it;
        DBMS_OUTPUT.PUT_LINE(employee.first_name || ' ' || employee.last_name);
    END LOOP;
    COMMIT;
END;
```



La section BEGIN

L'utilisation du curseur explicite : 3^{ème} méthode

FOR ... IN ... LOOP



La section BEGIN

L'utilisation curseur explicite : 3^{ème} méthode

```
SET SERVEROUTPUT ON;

BEGIN

  FOR employee IN (SELECT first_name, last_name FROM EMPLOYEES WHERE DEPARTMENT_ID = 60) LOOP
    DBMS_OUTPUT.PUT_LINE(employee.first_name);
  END LOOP;

END;
```

La section BEGIN

Le curseur implicite

- C'est un curseur de session déclaré et géré implicitement par PL/SQL.
- Il contient des informations à propos des dernières instructions DML effectuées.
- Utilisation :
 - SQL%FOUND
 - SQL%ROWCOUNT



La section BEGIN

Bloc anonyme

Démonstration



La section BEGIN

Conclusion

- Vous connaissez des types de données complexes
- Vous savez définir et manipuler un tableau
- Vous savez définir et manipuler une collection
- Vous connaissez la syntaxe des structures de contrôle du langage PL/SQL
- Vous savez définir et manipuler un curseur explicite
- Vous connaissez les attributs de curseur



PL / SQL

Module 6 – La section EXCEPTION



La section EXCEPTION

Objectifs

- Savoir gérer les erreurs de programmation
- Savoir gérer les anomalies utilisateurs



La section EXCEPTION

L'emplacement de la section Exception

```
DECLARE
    --Déclaration des variables
BEGIN
    --Traitement
EXCEPTION
    --Gestion des erreurs
END;
```



La section EXCEPTION

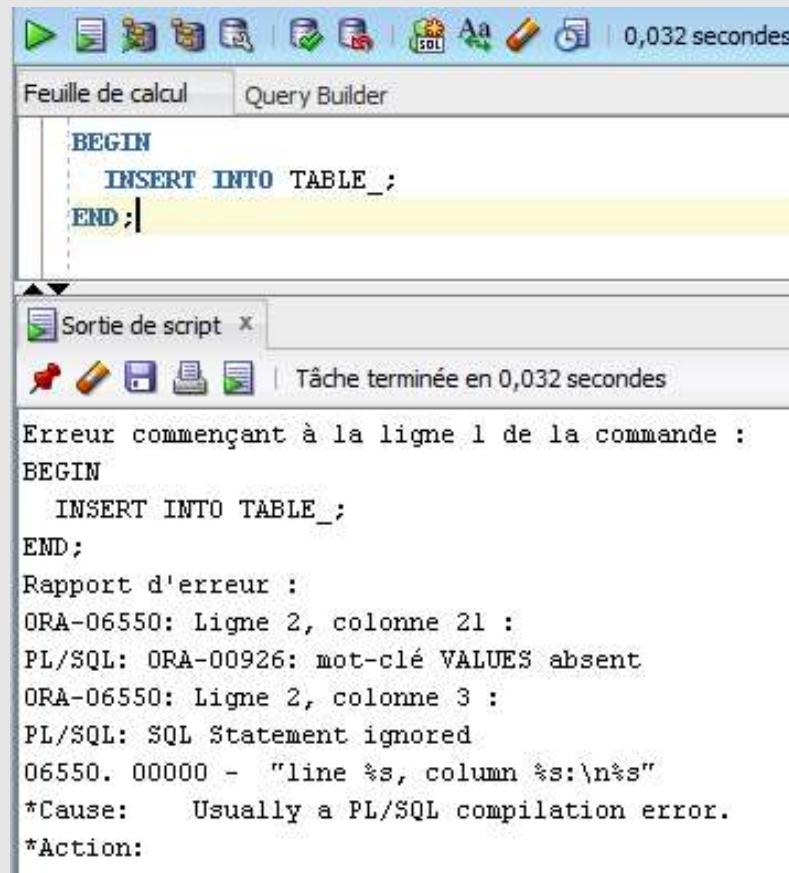
Les différents types d'erreurs

- Erreur de compilation
- Erreur d'exécution
- Erreur utilisateur



La section EXCEPTION

Les erreurs de compilation



The screenshot shows a SQL developer interface. In the top-left panel, there is a code editor with the following PL/SQL block:

```
BEGIN
    INSERT INTO TABLE_;
END ;
```

In the bottom-right panel, titled "Sortie de script", the output is:

```
Tâche terminée en 0,032 secondes
Erreur commençant à la ligne 1 de la commande :
BEGIN
    INSERT INTO TABLE_;
END;
Rapport d'erreur :
ORA-06550: Ligne 2, colonne 21 :
PL/SQL: ORA-00926: mot-clé VALUES absent
ORA-06550: Ligne 2, colonne 3 :
PL/SQL: SQL Statement ignored
06550. 00000 -  "line %s, column %s:\n%s"
*Cause:    Usually a PL/SQL compilation error.
*Action:
```

La section EXCEPTION

Les erreurs d'exécution prédéfinies

- Erreur Oracle
- Exception numérotée
- Exception nommée



La section EXCEPTION

Les erreurs d'exécution prédéfinies

The screenshot shows a PL/SQL anonymous block attempting to select data from a table named 'regions'. The code is as follows:

```
DECLARE
    region regions%ROWTYPE;
BEGIN
    SELECT * INTO region FROM regions WHERE region_id = 21541851;
END;
```

The output window shows the error message:

Erreur commençant à la ligne 1 de la commande :
DECLARE
region regions%ROWTYPE;
BEGIN
SELECT * INTO region FROM regions WHERE region_id = 21541851;
END;
Rapport d'erreur :
ORA-01403: aucune donnée trouvée
ORA-06512: à ligne 4
01403. 00000 - "no data found"
*Cause:
*Action:

The screenshot shows a PL/SQL anonymous block with an 'EXCEPTION' section. The code is as follows:

```
SET SERVEROUTPUT ON;

DECLARE
    region regions%ROWTYPE;
BEGIN
    SELECT * INTO region FROM regions WHERE region_id = 21541851;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Je gère !');
END;
```

The output window shows the message:

Sortie de script x
Tâche terminée en 0,016 secondes
bloc anonyme terminé
Je gère !

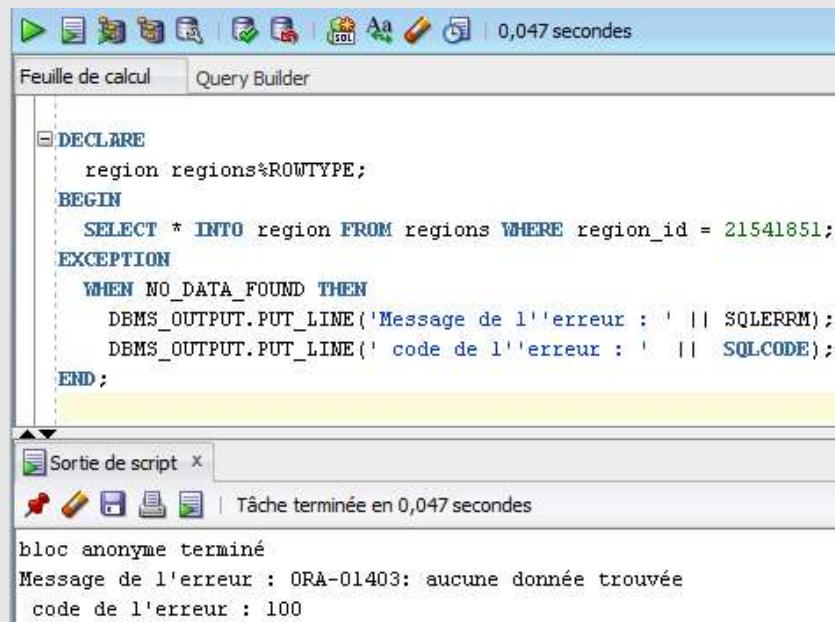
La section EXCEPTION

La liste des exceptions prédéfinies

NOM	NUMERO	SQLCODE
ACCESS_INTO_NULL	ORA-06530	-6530
CASE_NOT_FOUND	ORA-06592	-6592
COLLECTION_IS_NULL	ORA-06531	-6531
CURSOR_ALREADY_OPENED	ORA-06511	-6511
DUP_VAL_ON_INDEX	ORA-00001	-1
INVALID_CURSOR	ORA-01001	-1001
INVALID_NUMBER	ORA-01722	-1722
NO_DATA_FOUND	ORA-01403	+100
PROGRAM_ERROR	ORA-06501	-6501
ROWTYPE_MISMATCH	ORA-06504	-6504
STORAGE_ERROR	ORA-06500	-6500
SUBSCRIPT_BEYOND_COUNT	ORA-06533	-6533
SUBSCRIPT_OUTSIDE_LIMIT	ORA-06532	-6532
SYS_INVALID_ROWID	ORA-01410	-1410
TOO_MANY_ROWS	ORA-01422	-1422
VALUE_ERROR	ORA-06502	-6502
ZERO_DIVIDE	ORA-01476	-1476

La section EXCEPTION

Les fonctions SQLEERRM & SQLCODE



The screenshot shows the Oracle SQL Developer interface. The main window displays a PL/SQL anonymous block:

```
DECLARE
    region regions%ROWTYPE;
BEGIN
    SELECT * INTO region FROM regions WHERE region_id = 21541851;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Message de l''erreur : ' || SQLEERRM);
        DBMS_OUTPUT.PUT_LINE(' code de l''erreur : ' || SQLCODE);
END;
```

The code uses the `SQLEERRM` and `SQLCODE` functions within a `NO_DATA_FOUND` exception handler to output error messages and codes to the database log.

Below the code editor is a "Sortie de script" (Script Output) window:

Sortie de script X
Tâche terminée en 0,047 secondes

bloc anonyme terminé
Message de l'erreur : ORA-01403: aucune donnée trouvée
code de l'erreur : 100

The output window shows the results of the script execution, indicating that the task completed in 0.047 seconds and that the anonymous block was terminated because no data was found, resulting in the ORA-01403 error message and code 100.

La section EXCEPTION

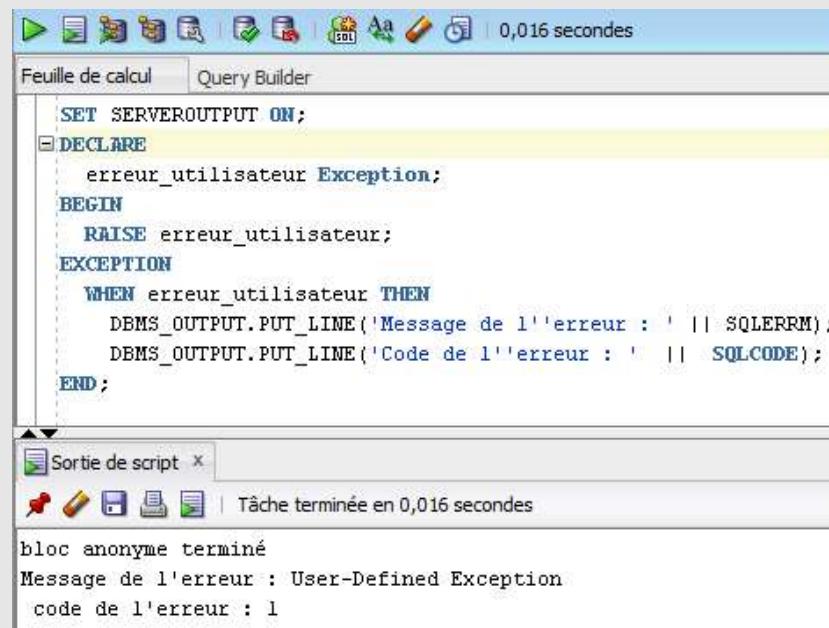
Les erreurs utilisateur

- Définies grâce au type EXCEPTION
- Levées grâce au mot-clé RAISE
- Permettent de se protéger de traitements illogiques



La section EXCEPTION

Les erreurs utilisateur



The screenshot shows the Oracle SQL Developer interface. The top bar includes standard icons and the text "0,016 secondes". Below it, the title bar says "Feuille de calcul Query Builder". The main area contains the following PL/SQL code:

```
SET SERVEROUTPUT ON;
DECLARE
    erreur_utilisateur Exception;
BEGIN
    RAISE erreur_utilisateur;
EXCEPTION
    WHEN erreur_utilisateur THEN
        DBMS_OUTPUT.PUT_LINE('Message de l''erreur : ' || SQLERRM);
        DBMS_OUTPUT.PUT_LINE('Code de l''erreur : ' || SQLCODE);
END;
```

Below the code, a "Sortie de script" window displays the results of the execution:

```
Tâche terminée en 0,016 secondes
bloc anonyme terminé
Message de l'erreur : User-Defined Exception
code de l'erreur : 1
```

La section EXCEPTION

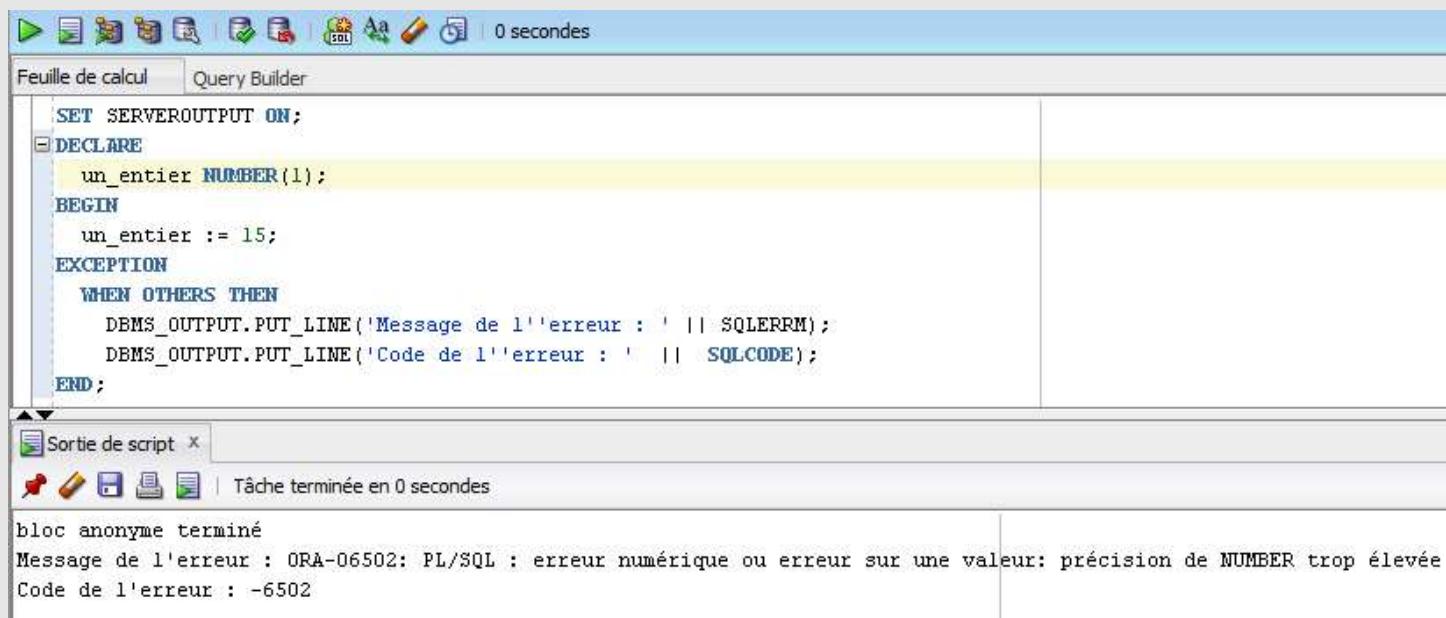
Les erreurs d'exécution non prédéfinies

- Erreurs Oracle
- Définies par un code erreur numérique



La section EXCEPTION

Les erreurs d'exécution non prédéfinies



The screenshot shows the Oracle SQL Developer interface. The main window displays a PL/SQL anonymous block:

```
SET SERVEROUTPUT ON;
DECLARE
    un_entier NUMBER(1);
BEGIN
    un_entier := 15;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Message de l''erreur : ' || SQLERRM);
        DBMS_OUTPUT.PUT_LINE('Code de l''erreur : ' || SQLCODE);
END;
```

Below the code editor, a "Sortie de script" (Script Output) window is open, showing the results of the execution:

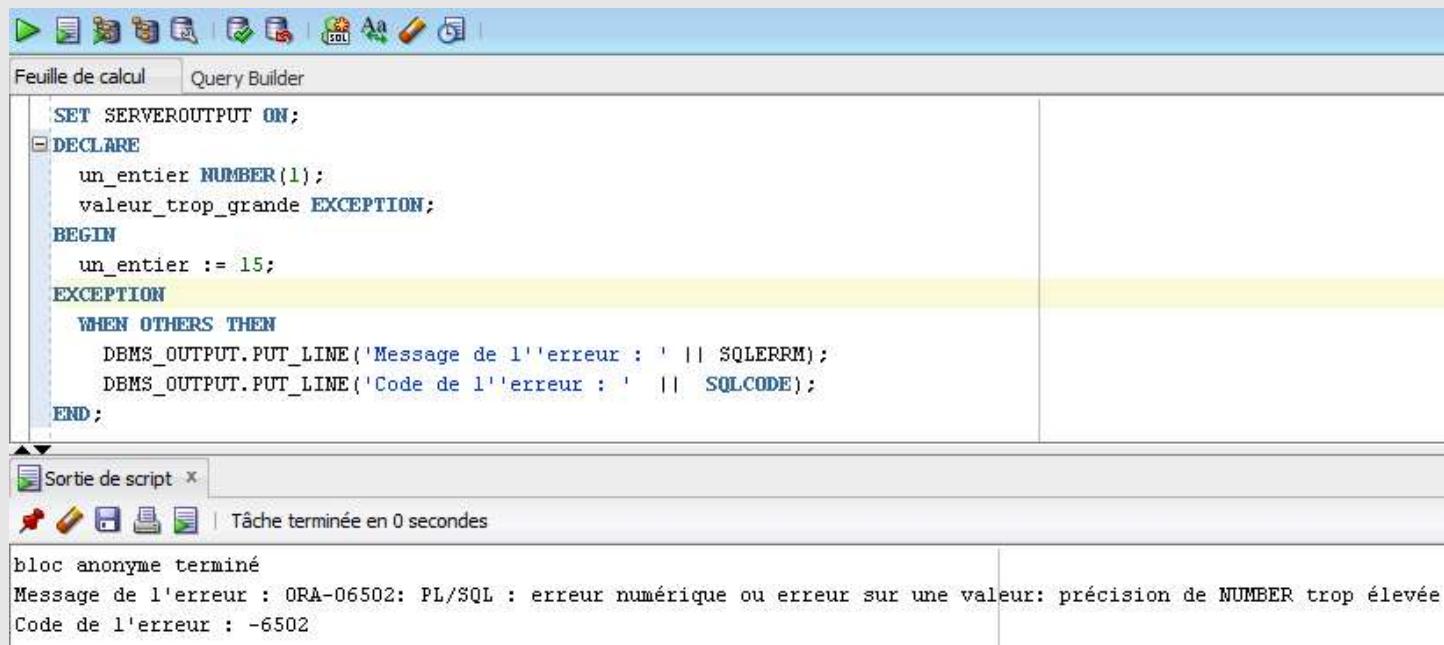
Tâche terminée en 0 secondes

bloc anonyme terminé

Message de l'erreur : ORA-06502: PL/SQL : erreur numérique ou erreur sur une valeur: précision de NUMBER trop élevée
Code de l'erreur : -6502

La section EXCEPTION

La directive de compilation PRAGMA EXCEPTION_INIT



The screenshot shows the Oracle SQL Developer interface. The top menu bar has icons for running, saving, and zooming. Below it, the tabs "Feuille de calcul" and "Query Builder" are visible, with "Feuille de calcul" selected.

The main code editor window contains the following PL/SQL code:

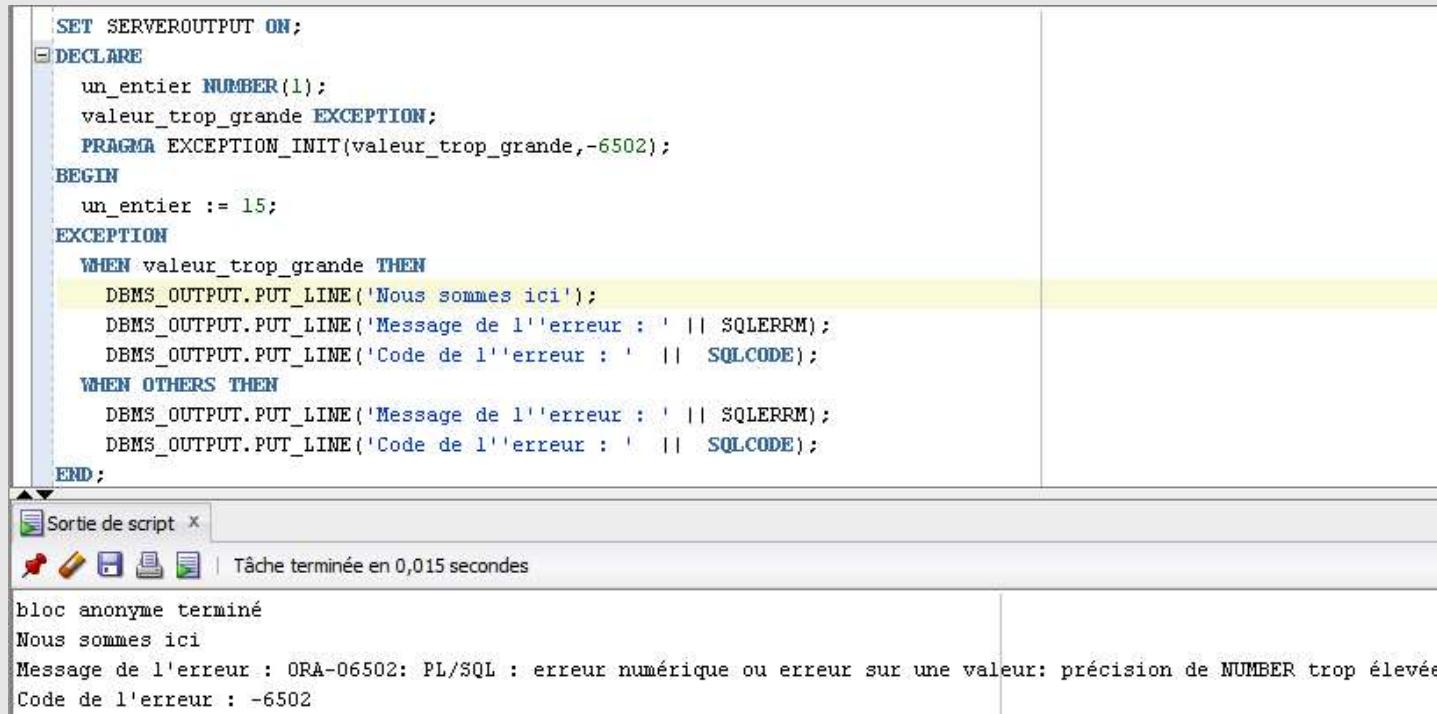
```
SET SERVEROUTPUT ON;
DECLARE
    un_entier NUMBER(1);
    valeur_trop_grande EXCEPTION;
BEGIN
    un_entier := 15;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Message de l''erreur : ' || SQLERRM);
        DBMS_OUTPUT.PUT_LINE('Code de l''erreur : ' || SQLCODE);
END;
```

Below the code editor is a "Sortie de script" (Script Output) window. It shows the status "Tâche terminée en 0 secondes" (Task completed in 0 seconds). The output pane displays the error message:

bloc anonyme terminé
Message de l'erreur : ORA-06502: PL/SQL : erreur numérique ou erreur sur une valeur: précision de NUMBER trop élevée
Code de l'erreur : -6502

La section EXCEPTION

La directive de compilation PRAGMA EXCEPTION_INIT



The screenshot shows a PL/SQL anonymous block in the SQL Developer interface. The code uses the `PRAGMA EXCEPTION_INIT` directive to map the error `ORA-06502` to the exception `valeur_trop_grande`. It then catches both `valeur_trop_grande` and `OTHERS` exceptions to output the error message and code. The execution output window shows the successful completion of the task and the resulting error message.

```
SET SERVEROUTPUT ON;
DECLARE
    un_entier NUMBER(1);
    valeur_trop_grande EXCEPTION;
    PRAGMA EXCEPTION_INIT(valeur_trop_grande,-6502);
BEGIN
    un_entier := 15;
EXCEPTION
    WHEN valeur_trop_grande THEN
        DBMS_OUTPUT.PUT_LINE('Nous sommes ici');
        DBMS_OUTPUT.PUT_LINE('Message de l''erreur : ' || SQLERRM);
        DBMS_OUTPUT.PUT_LINE('Code de l''erreur : ' || SQLCODE);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Message de l''erreur : ' || SQLERRM);
        DBMS_OUTPUT.PUT_LINE('Code de l''erreur : ' || SQLCODE);
END;
```

Sortie de script X

Sortie de script X | Tâche terminée en 0,015 secondes

bloc anonyme terminé

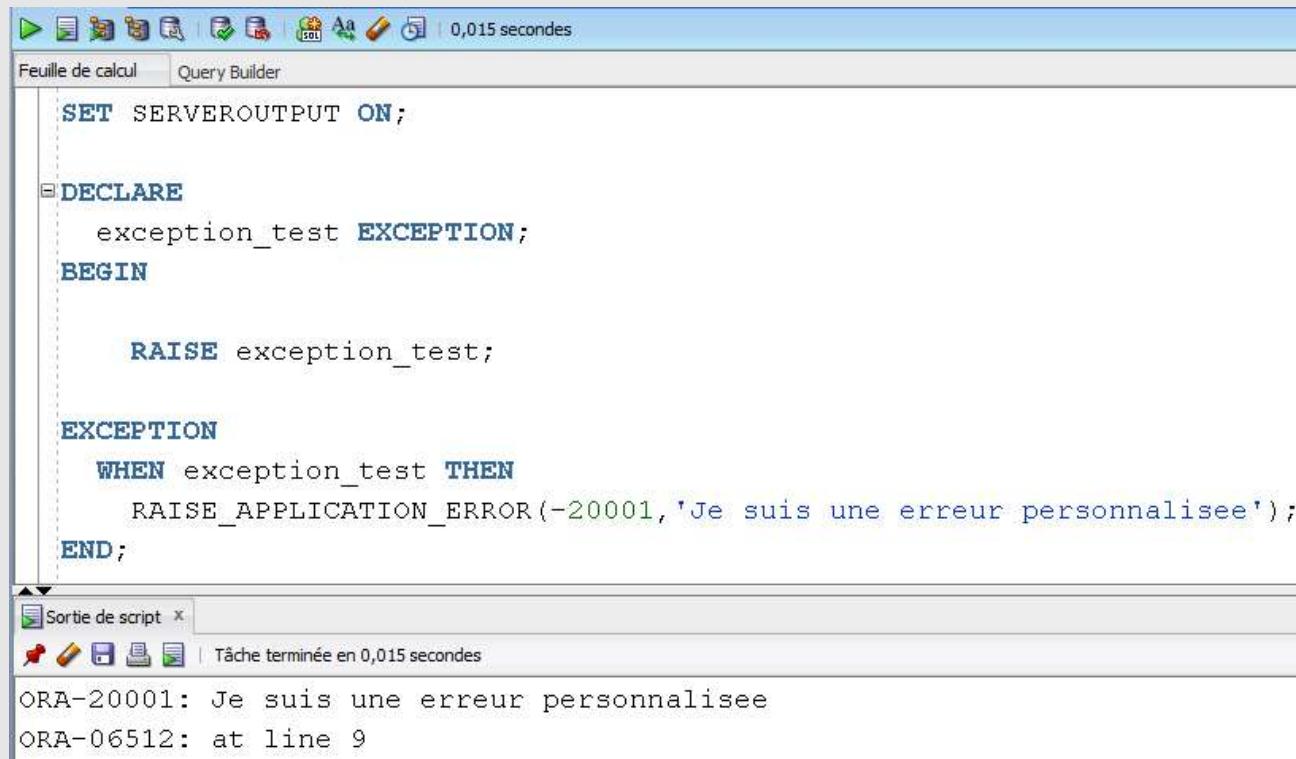
Nous sommes ici

Message de l'erreur : ORA-06502: PL/SQL : erreur numérique ou erreur sur une valeur: précision de NUMBER trop élevée

Code de l'erreur : -6502

La section EXCEPTION

La procédure RAISE_APPLICATION_ERROR



The screenshot shows a SQL developer interface with a query builder window. The code is as follows:

```
SET SERVEROUTPUT ON;

DECLARE
    exception_test EXCEPTION;
BEGIN
    RAISE exception_test;
EXCEPTION
    WHEN exception_test THEN
        RAISE_APPLICATION_ERROR(-20001, 'Je suis une erreur personnalisée');
END;
```

In the output pane, the error message is displayed:

```
ORA-20001: Je suis une erreur personnalisée
ORA-06512: at line 9
```

La section EXCEPTION

La propagation des exceptions 1/3

```
SET SERVEROUTPUT ON;

DECLARE
    exception_test EXCEPTION;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Je suis avant l''erreur');
    RAISE exception_test;
    DBMS_OUTPUT.PUT_LINE('Je suis apres l''erreur');
EXCEPTION
    WHEN exception_test THEN
        DBMS_OUTPUT.PUT_LINE('Je traite l''erreur');
END;
```

La section EXCEPTION

La propagation des exceptions 2/3

```
SET SERVEROUTPUT ON;

DECLARE
    exception_test EXCEPTION;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Je suis avant le sous bloc');
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Je suis avant l''erreur');
        RAISE exception_test;
        DBMS_OUTPUT.PUT_LINE('Je suis après l''erreur');
    END;
    DBMS_OUTPUT.PUT_LINE('Je suis après le sous bloc');
EXCEPTION
    WHEN exception_test THEN
        DBMS_OUTPUT.PUT_LINE('Je traite l''erreur');
END;
```

La section EXCEPTION

La propagation des exceptions 3/3

```
SET SERVEROUTPUT ON;

DECLARE
    exception_test EXCEPTION;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Je suis avant le sous bloc');
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Je suis avant l''erreur');
        RAISE exception_test;
        DBMS_OUTPUT.PUT_LINE('Je suis apres l''erreur');
    EXCEPTION
        WHEN exception_test THEN
            DBMS_OUTPUT.PUT_LINE('Je traite l''erreur dans le sous bloc');
    END;
    DBMS_OUTPUT.PUT_LINE('Je suis après le sous bloc');
EXCEPTION
    WHEN exception_test THEN
        DBMS_OUTPUT.PUT_LINE('Je traite l''erreur');
END;
```

Conclusion

- Vous savez ce qu'est une erreur utilisateur
- Vous savez ce qu'est une erreur prédefinie & non prédefinie
- Vous savez utiliser le type EXCEPTION
- Vous savez lever une erreur l'instruction avec RAISE
- Vous savez associer une erreur non prédefinie à une variable EXCEPTION
- Vous savez associer un message à une erreur avec RAISE_APPLICATION_ERROR
- Vous connaissez le mécanisme de propagation des erreurs



PL / SQL

Module 7 – Les procédures stockées



Objectifs

- Savoir créer une procédure stockée
- Savoir utiliser les différents types de paramètres pour les procédures stockées
- Savoir appeler une procédure stockée



Les procédures stockées

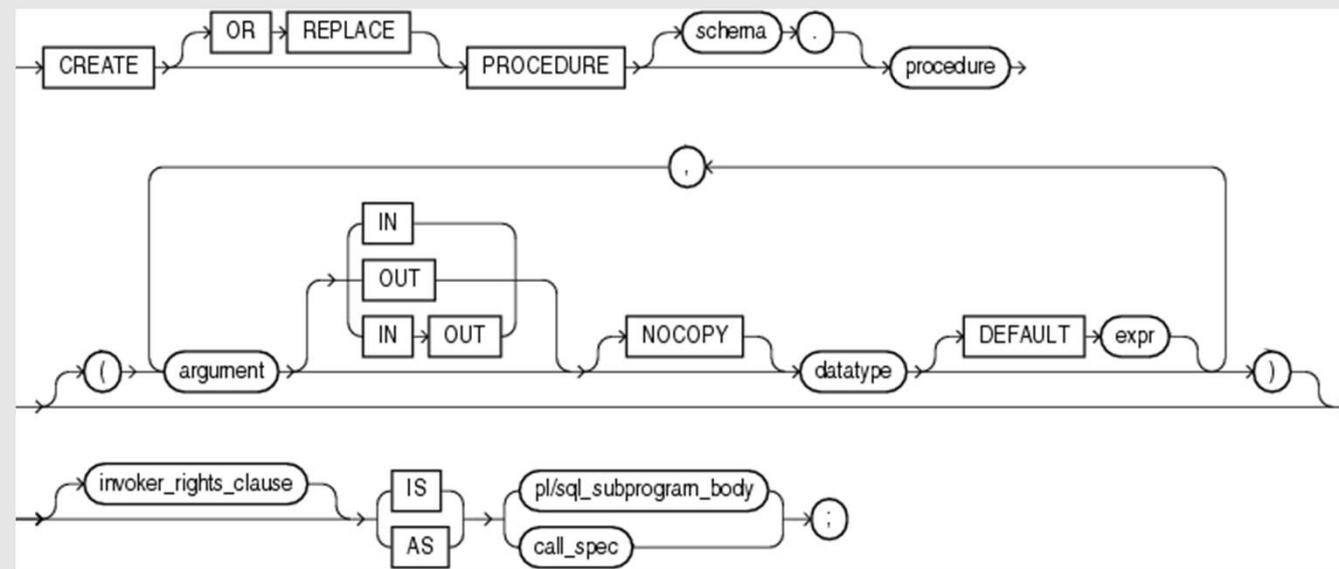
Définition

- Programme défini par l'utilisateur
- Programme stocké sur le serveur
- Bloc PL/SQL associé à un nom



Les procédures stockées

La syntaxe de création



Les procédures stockées

Exemple de création d'une procédure stockée

```
SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE afficher_employes
AS
  CURSOR cursor_employees_it IS SELECT first_name, last_name FROM EMPLOYEES WHERE DEPARTMENT_ID = 60;
  employee employees%ROWTYPE;
BEGIN

  FOR employee IN cursor_employees_it LOOP
    DBMS_OUTPUT.PUT_LINE(employee.first_name);
  END LOOP;

END;
```



Les procédures stockées

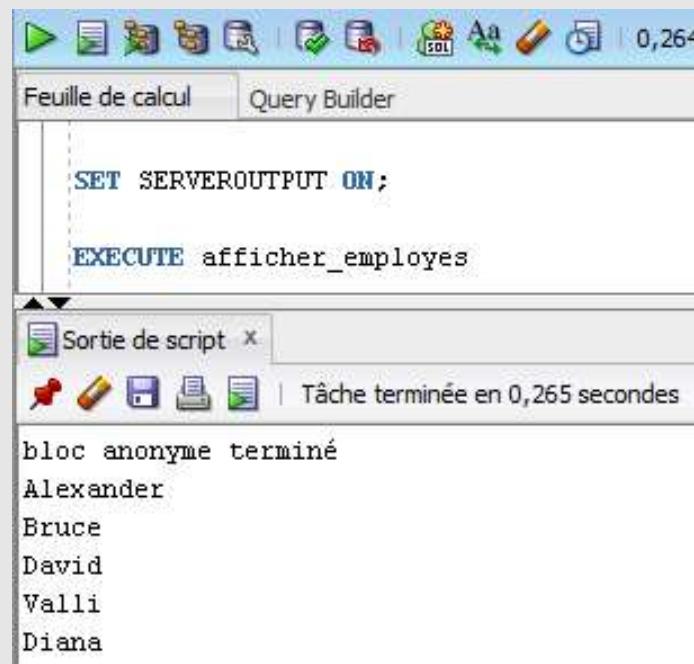
L'appel à une procédure stockée

- En utilisant la commande EXECUTE
- En utilisant le nom de la procédure stockée



Les procédures stockées

L'appel par la commande EXECUTE



The screenshot shows the Oracle SQL Developer interface. In the top menu bar, there are several icons and the text "0,264". Below the menu is a toolbar with icons for running scripts, opening files, and other database operations. The main window has two tabs: "Feuille de calcul" (Calculation Sheet) and "Query Builder". The "Query Builder" tab is active, containing the following PL/SQL code:

```
SET SERVEROUTPUT ON;  
EXECUTE afficher_employes;
```

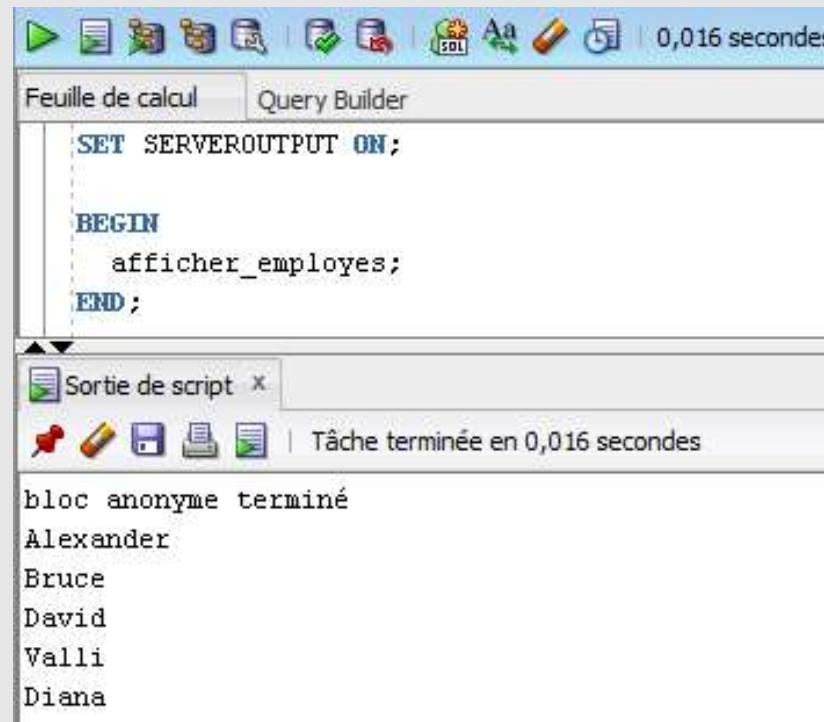
Below the code editor is a "Sortie de script" (Script Output) window. It displays the output of the executed procedure:

bloc anonyme terminé
Alexander
Bruce
David
Valli
Diana

The output window also indicates that the task was completed in 0,265 seconds.

Les procédures stockées

L'appel à partir d'un autre bloc



The screenshot shows the Oracle SQL Developer interface. In the top toolbar, there are various icons for navigating between tabs, running queries, and managing sessions. The status bar at the bottom indicates "0,016 secondes". Below the toolbar, there are two tabs: "Feuille de calcul" (Calculation Sheet) and "Query Builder". The "Feuille de calcul" tab is active and contains the following PL/SQL code:

```
SET SERVEROUTPUT ON;

BEGIN
    afficher_employes;
END;
```

Below the code, a message box titled "Sortie de script" (Script Output) displays the results of the execution:

Sortie de script

Tâche terminée en 0,016 secondes

bloc anonyme terminé

Alexander
Bruce
David
Valli
Diana

Les procédures stockées

Les paramètres de procédure

- IN
- OUT
- IN OUT



Les procédures stockées

Les paramètres IN

- Type par défaut
- Paramètre d'entrée



Les procédures stockées

Exemple de paramètre d'entrée

```
SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE afficher_employes(id_department IN NUMBER)
AS
    CURSOR cursor_employees_it IS SELECT first_name, last_name FROM EMPLOYEES WHERE DEPARTMENT_ID = id_department;
    employee employees%ROWTYPE;
BEGIN
    FOR employee IN cursor_employees_it LOOP
        DBMS_OUTPUT.PUT_LINE(employee.first_name);
    END LOOP;
END;
```



Les procédures stockées

Les paramètres OUT

- Paramètre de sortie



Les procédures stockées

Exemple de paramètre de sortie

```
SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE afficher_employes(id_department IN NUMBER, total_employes OUT NUMBER)
AS
    CURSOR cursor_employees_it IS SELECT first_name, last_name FROM EMPLOYEES WHERE DEPARTMENT_ID = id_department;
    employee employees%ROWTYPE;
BEGIN

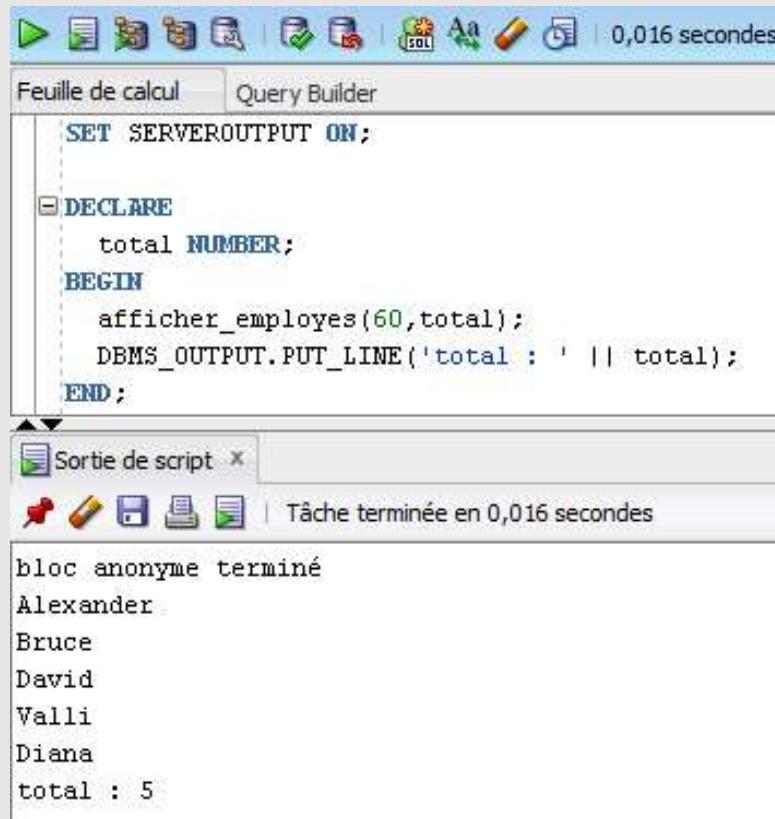
    SELECT COUNT(*) INTO total_employes FROM employees WHERE DEPARTMENT_ID = id_department;

    FOR employee IN cursor_employees_it LOOP
        DBMS_OUTPUT.PUT_LINE(employee.first_name);
    END LOOP;
END;
```



Les procédures stockées

Le passage de paramètres à une procédure



The screenshot shows the Oracle SQL Developer interface. The top menu bar includes icons for running, saving, and zooming, along with a toolbar and a status bar indicating "0,016 secondes". The main window has tabs for "Feuille de calcul" and "Query Builder", with "Feuille de calcul" selected. The code area contains the following PL/SQL block:

```
SET SERVEROUTPUT ON;

DECLARE
    total NUMBER;
BEGIN
    afficher_employes(60,total);
    DBMS_OUTPUT.PUT_LINE('total : ' || total);
END;
```

Below the code is a "Sortie de script" (Script Output) window. It shows the results of the execution:

```
bloc anonyme terminé
Alexander
Bruce
David
Valli
Diana
total : 5
```

The output window also displays the message "Tâche terminée en 0,016 secondes" (Task completed in 0,016 seconds).

Les procédures stockées

Les paramètres IN OUT

- Paramètres d'entrées et de sorties



Les procédures stockées

Exemple de paramètre entrée / sortie

```
SET SERVEROUTPUT ON;

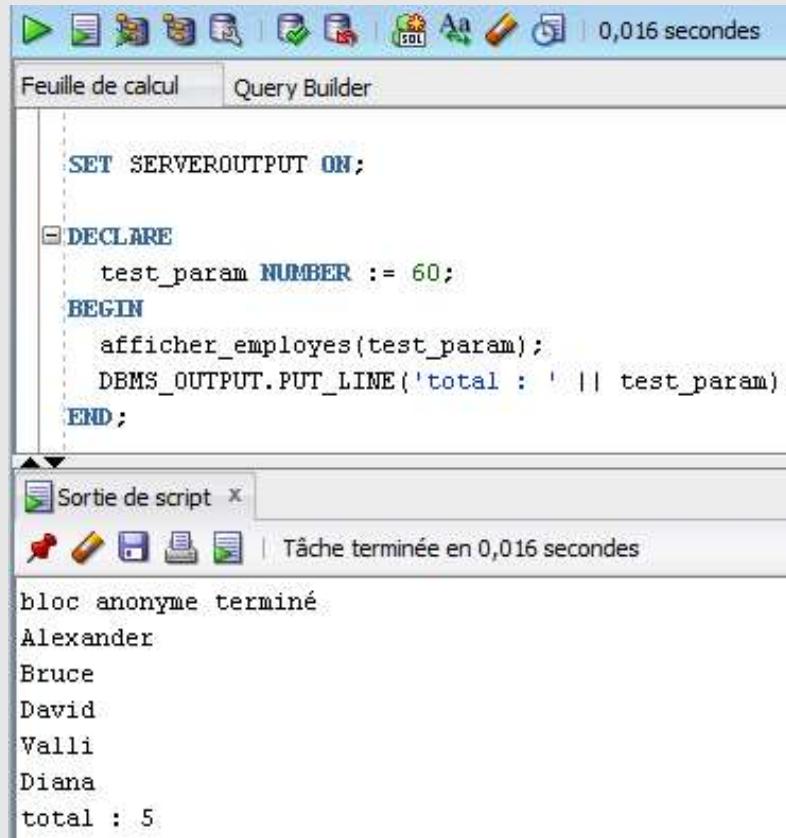
CREATE OR REPLACE PROCEDURE afficher_employes(info IN OUT NUMBER)
AS
    CURSOR cursor_employees_it IS SELECT first_name, last_name FROM EMPLOYEES WHERE DEPARTMENT_ID = info;
    employee employees%ROWTYPE;
BEGIN

    FOR employee IN cursor_employees_it LOOP
        DBMS_OUTPUT.PUT_LINE(employee.first_name);
    END LOOP;

    SELECT COUNT(*) INTO info FROM employees WHERE DEPARTMENT_ID = info;
END;
```

Les procédures stockées

L'utilisation d'un paramètre entrée / sortie



The screenshot shows the Oracle SQL Developer interface. The main window displays a PL/SQL anonymous block:

```
SET SERVEROUTPUT ON;

DECLARE
    test_param NUMBER := 60;
BEGIN
    afficher_employes(test_param);
    DBMS_OUTPUT.PUT_LINE('total : ' || test_param);
END;
```

The bottom window, titled "Sortie de script", shows the execution results:

```
bloc anonyme terminé
Alexander
Bruce
David
Valli
Diana
total : 5
```

Les procédures stockées

Démonstration



Conclusion

- Vous savez créer des procédures stockées
- Vous savez utiliser des procédures stockées
- Vous connaissez les différents types de paramètres applicables aux procédures stockées



PL / SQL

Module 8 – Les fonctions



Objectifs

- Savoir créer une fonction utilisateur
- Savoir utiliser une fonction utilisateur
- Savoir exploiter la valeur renvoyée par une fonction



Les fonctions

Définition

- Bloc de code nommé et stocké sur le serveur
- Retourne toujours une et seule une valeur



Les fonctions

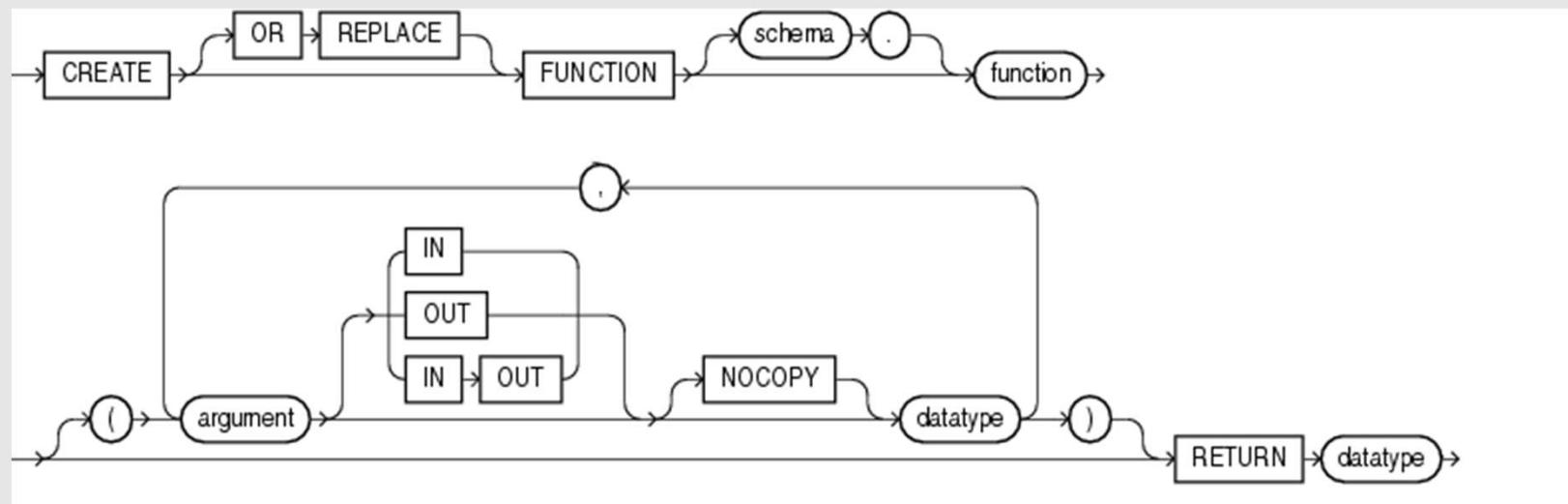
Les spécificités

- Les paramètres d'entrées sont exclusivement de type IN
- Les paramètres d'entrées doivent être de type SQL et non PL/SQL
- Le paramètre de retour doit être de type SQL et non PL/SQL
- Les fonctions ne doivent pas faire de DML (INSERT, UPDATE, DELETE)



Les fonctions

La syntaxe de création



Les fonctions

Exemple de création d'une fonction

```
CREATE OR REPLACE FUNCTION multiplier_par_deux(nombre_a_multiplier IN NUMBER)
RETURN NUMBER
IS
resultat NUMBER;
BEGIN
    resultat := nombre_a_multiplier * 2;
    RETURN resultat;
END;
```

Les fonctions

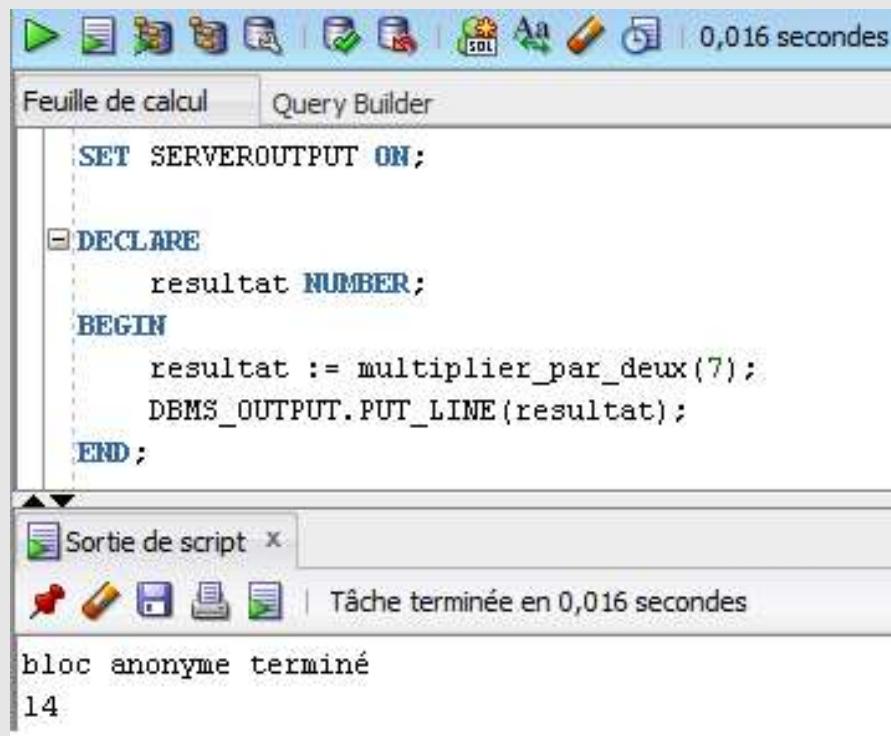
L'appel à une fonction

- Depuis un bloc PL/SQL
- Depuis une requête SQL



Les fonctions

L'appel à partir d'un bloc



The screenshot shows the Oracle SQL Developer interface. The top bar includes standard icons and the text "0,016 secondes". Below the bar, there are two tabs: "Feuille de calcul" (Calculation Sheet) and "Query Builder". The main area displays an anonymous PL/SQL block:

```
SET SERVEROUTPUT ON;

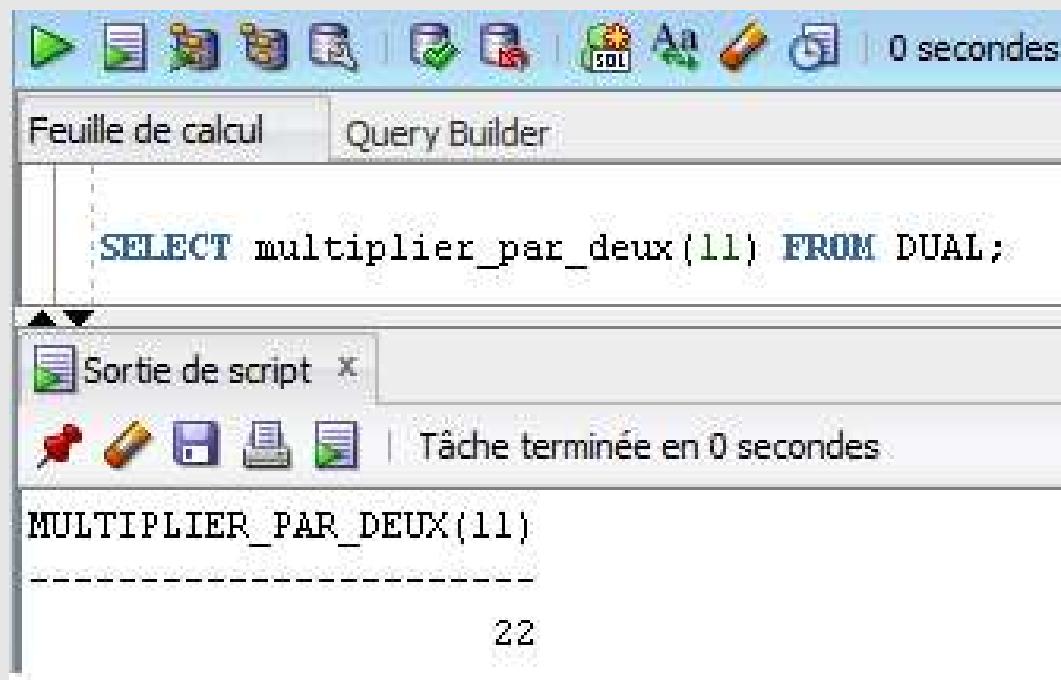
DECLARE
    resultat NUMBER;
BEGIN
    resultat := multiplier_par_deux(7);
    DBMS_OUTPUT.PUT_LINE(resultat);
END;
```

Below the code, a "Sortie de script" (Script Output) window is open, showing the result of the execution:

```
Tâche terminée en 0,016 secondes
bloc anonyme terminé
14
```

Les fonctions

L'appel à partir d'une requête



The screenshot shows the Oracle SQL Developer interface. The top menu bar includes icons for running, saving, and zooming, followed by a toolbar with various buttons. The title bar displays "Feuille de calcul" and "Query Builder". The main area contains a SQL query:

```
SELECT multiplier_par_deux(11) FROM DUAL;
```

Below the query, a message indicates the task was completed in 0 seconds:

Sortie de script | Tâche terminée en 0 secondes

The results pane shows the output of the function call:

```
MULTIPLIER_PAR_DEUX(11)
-----
22
```

Les fonctions

Démonstration



Conclusion

- Vous savez créer des fonctions
- Vous savez utiliser des fonctions
- Vous savez exploiter la valeur renvoyée par la fonction



PL / SQL

Module 9 – Les déclencheurs de base de données



Les déclencheurs de base de données

Objectifs

- Savoir expliquer ce qu'est un trigger
- Savoir créer un trigger



Les déclencheurs de base de données

Définition

- Trigger = Déclencheur
- Traitement s'exécutant automatiquement lorsqu'un évènement (insertion, suppression, mise à jour) se produit sur une table ou une vue



Les déclencheurs de base de données

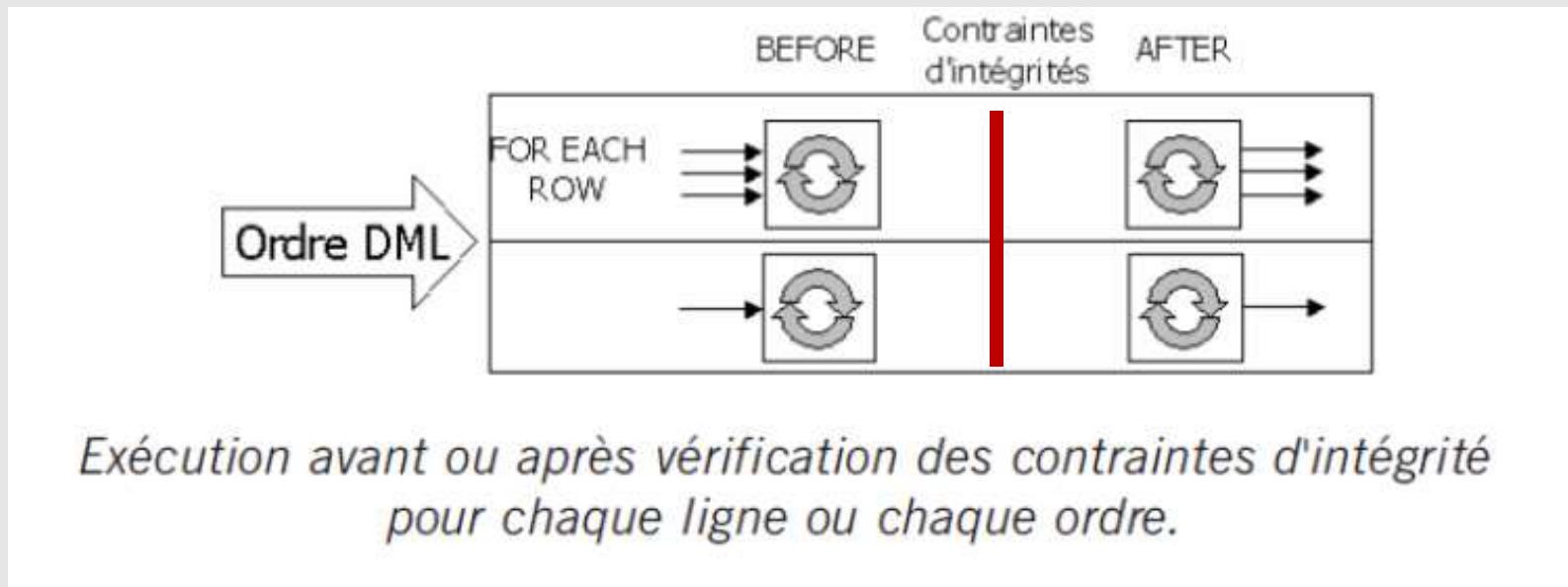
Les spécificités

- Blocs associés à un nom
- Peuvent appeler des sous-programmes
- Non paramétrables
- COMMIT et ROLLBACK interdits



Les déclencheurs de base de données

Le principe de fonctionnement



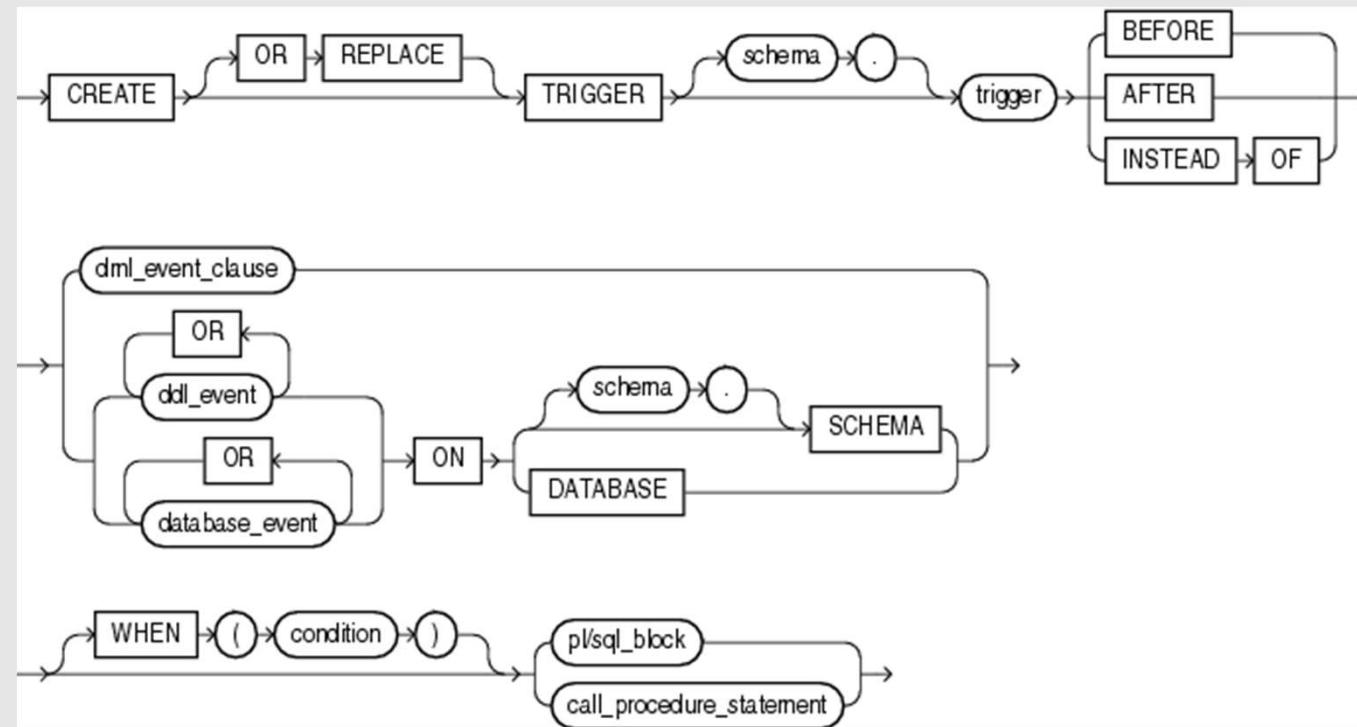
Les déclencheurs de base de données

Les différents types de trigger de table

TRIGGER	BEFORE	STATEMENT	INSERT	SELECT	
			UPDATE	SELECT	
			DELETE	SELECT	
		ROW	INSERT	SELECT	New
			UPDATE		New / Old
			DELETE		New
	AFTER	STATEMENT	INSERT	SELECT	
			UPDATE	SELECT	
			DELETE	SELECT	
		ROW	INSERT		New
			UPDATE		New / Old
			DELETE		New

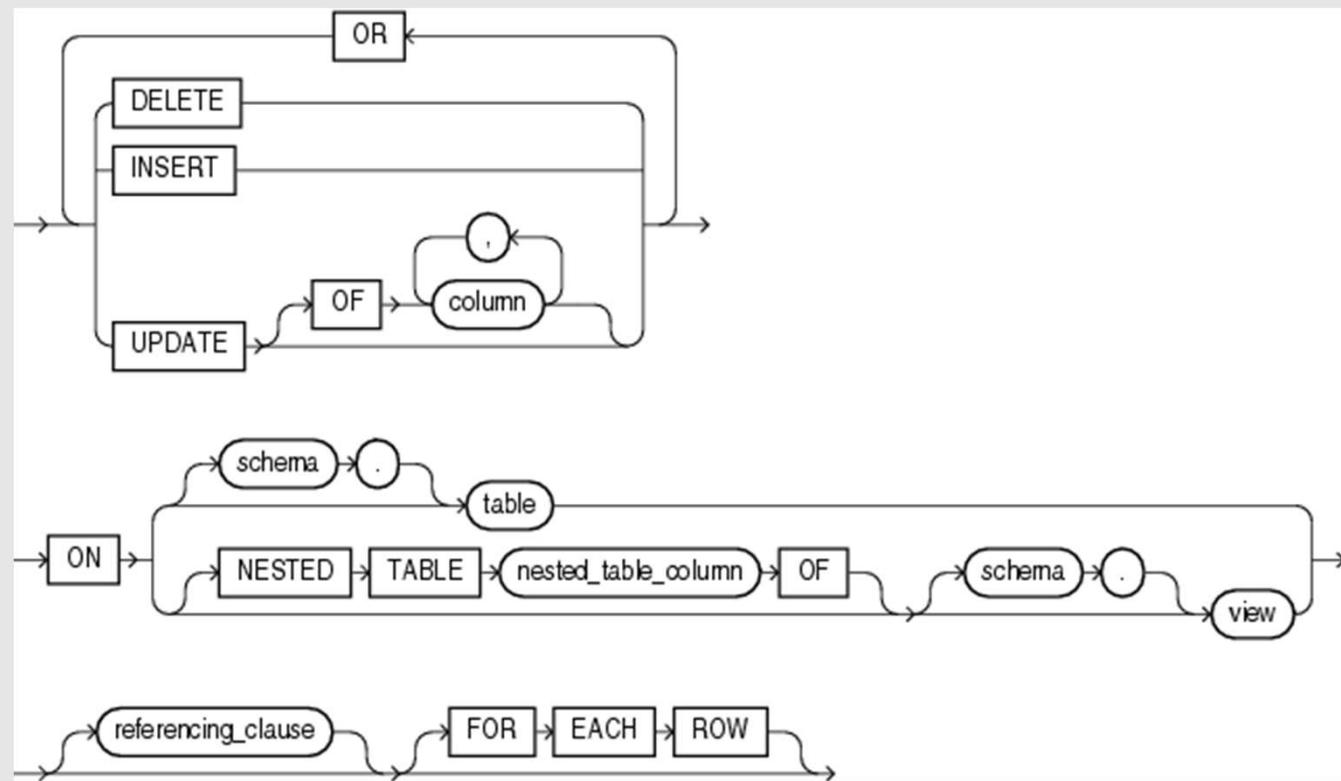
Les déclencheurs de base de données

La syntaxe de création 1/2



Les déclencheurs de base de données

La syntaxe de création 2/2



Les déclencheurs de base de données

Les prédictats de trigger

- INSERTING
- DELETING
- UPDATING



Les déclencheurs de base de données

L'utilisation des prédictats

```
CREATE OR REPLACE TRIGGER securisation_horaires
  BEFORE INSERT OR UPDATE OR DELETE ON employees
BEGIN
  IF TO_CHAR (SYSDATE, 'HH24:MI') NOT BETWEEN '08:00' AND '18:00' OR TO_CHAR (SYSDATE, 'DY') IN ('SAT', 'SUN') THEN
    IF INSERTING THEN
      RAISE_APPLICATION_ERROR (-20205, 'Vous ne pouvez pas faire d''inscriptions hors des heures normales de bureau');
    END IF;

    IF DELETING THEN
      RAISE_APPLICATION_ERROR (-20206, 'Vous ne pouvez pas faire de suppressions hors des heures normales de bureau');
    END IF;

    IF UPDATING THEN
      RAISE_APPLICATION_ERROR (-20207, 'Vous ne pouvez pas faire de mises à jour hors des heures normales de bureau');
    END IF;
  END IF;
END;
```



Les déclencheurs de base de données

Les variables implicites

- OLD
- NEW



Les déclencheurs de base de données

L'appel aux variables OLD et NEW

```
CREATE OR REPLACE TRIGGER securisation_salaire
BEFORE UPDATE ON employees
FOR EACH ROW
BEGIN
    IF(:OLD.salary > :NEW.salary) THEN
        RAISE_APPLICATION_ERROR(-20210,'Impossible de réduire un salaire');
    END IF;
END;
```

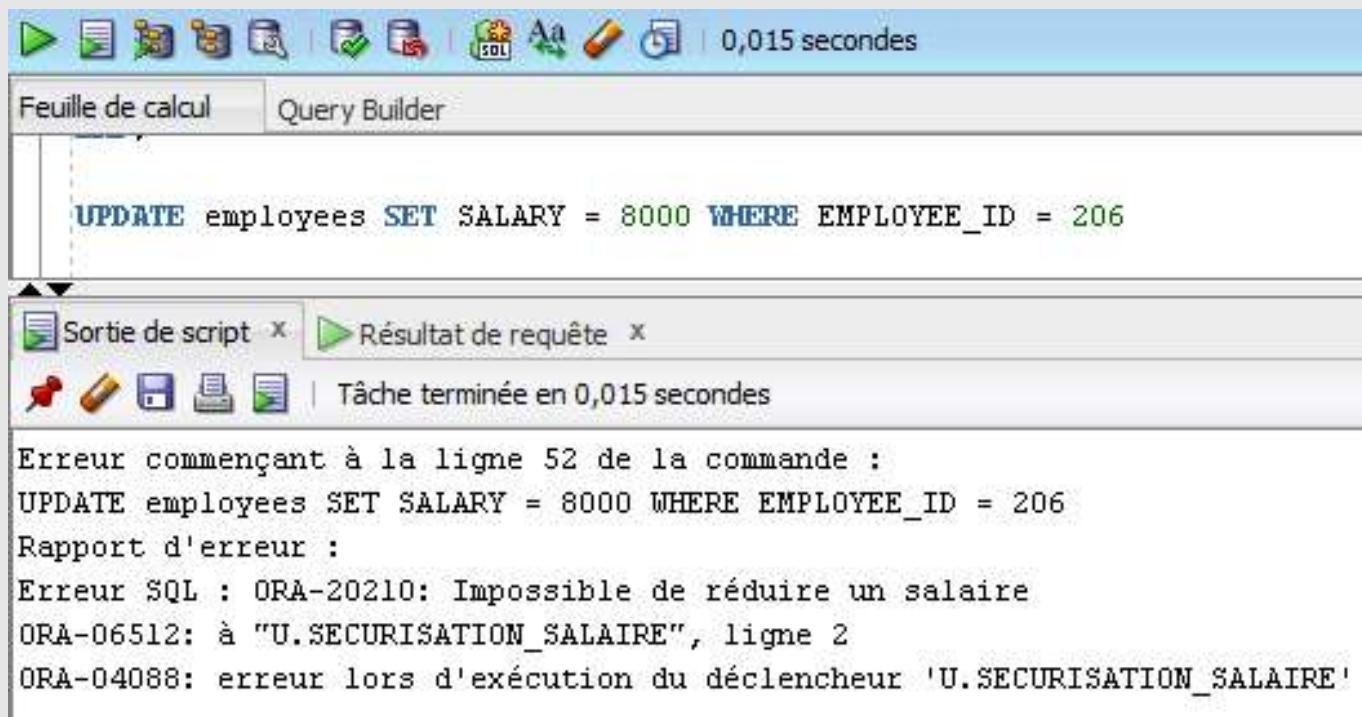
Les déclencheurs de base de données

La clause WHEN

```
CREATE OR REPLACE TRIGGER securisation_salaire
BEFORE UPDATE ON employees
FOR EACH ROW
WHEN(OLD.salary > NEW.salary)
BEGIN
    RAISE_APPLICATION_ERROR(-20210,'Impossible de réduire un salaire');
END;
```

Les déclencheurs de base de données

L'exécution du trigger



The screenshot shows a SQL developer interface. In the top query editor, there is a single UPDATE statement:

```
UPDATE employees SET SALARY = 8000 WHERE EMPLOYEE_ID = 206
```

In the bottom results window, an error message is displayed:

Sortie de script x Résultat de requête x

Tâche terminée en 0,015 secondes

Erreur commençant à la ligne 52 de la commande :

```
UPDATE employees SET SALARY = 8000 WHERE EMPLOYEE_ID = 206
```

Rapport d'erreur :

```
Erreur SQL : ORA-20210: Impossible de réduire un salaire
ORA-06512: à "U.SECURISATION_SALAIRE", ligne 2
ORA-04088: erreur lors d'exécution du déclencheur 'U.SECURISATION_SALAIRE'
```

Les déclencheurs de base de données

Les triggers sur vues

- La clause INSTEAD OF = Au lieu de
- Permet d'insérer, de modifier ou de supprimer à partir d'une vue **multitable**
- Utilise la clause FOR EACH implicitement



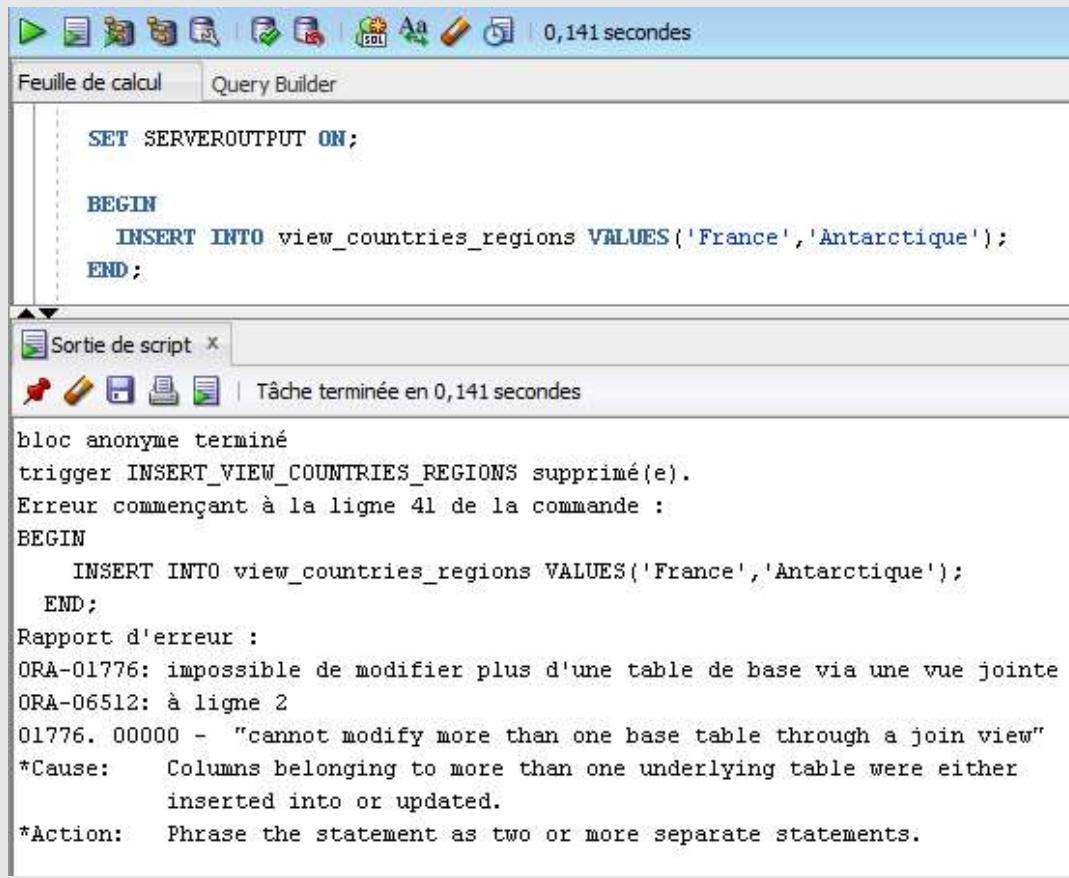
Les déclencheurs de base de données

Exemple de trigger sur vue 1/4

```
CREATE OR REPLACE VIEW view_countries_regions
AS
  SELECT
    c.country_name AS country,
    r.region_name AS region
  FROM
    countries c
  JOIN regions r ON c.region_id = r.region_id;
```

Les déclencheurs de base de données

Exemple de trigger sur vue 2/4



The screenshot shows a SQL developer interface with the following content:

```
SET SERVEROUTPUT ON;

BEGIN
    INSERT INTO view_countries_regions VALUES('France','Antarctique');
END;
```

Below the code, the "Sortie de script" (Script Output) window displays the results of the execution:

```
bloc anonyme terminé
trigger INSERT_VIEW_COUNTRIES_REGIONS supprimé(e).
Erreur commençant à la ligne 41 de la commande :
BEGIN
    INSERT INTO view_countries_regions VALUES('France','Antarctique');
END;
Rapport d'erreur :
ORA-01776: impossible de modifier plus d'une table de base via une vue jointe
ORA-06512: à ligne 2
01776. 00000 -  "cannot modify more than one base table through a join view"
*Cause:    Columns belonging to more than one underlying table were either
           inserted into or updated.
*Action:   Phrase the statement as two or more separate statements.
```

Les déclencheurs de base de données

Exemple de trigger sur vue 3/4

```
CREATE OR REPLACE TRIGGER insert_view_countries_regions
INSTEAD OF INSERT ON view_countries_regions
DECLARE
    total_region NUMBER;
    new_id_region NUMBER;
    total_country NUMBER;
    new_id_country NUMBER;
BEGIN

    SELECT COUNT(*) INTO total_region FROM REGIONS WHERE REGION_NAME = :NEW.region;
    IF total_region = 0 THEN
        SELECT (MAX(REGION_ID)+1) INTO new_id_region FROM REGIONS;
        INSERT INTO regions VALUES(new_id_region,:NEW.region);
    END IF;

    SELECT COUNT(*) INTO total_country FROM countries WHERE COUNTRY_NAME = :NEW.country;
    IF total_country = 0 THEN
        SELECT (MAX(REGION_ID)+1) INTO new_id_country FROM countries;
        INSERT INTO countries VALUES(new_id_country,:NEW.country,new_id_region);
    END IF;
END;
```

Les déclencheurs de base de données

Exemple de trigger sur vue 4/4

The screenshot shows the Oracle SQL Developer interface. The top toolbar has various icons for database navigation and management. The main window title bar says "Feuille de calcul" and "Query Builder". Below the title bar, the status bar indicates "0,015 secondes". The code area contains the following PL/SQL block:

```
SET SERVEROUTPUT ON;

BEGIN
    INSERT INTO view_countries_regions VALUES('Chili','Antarctique');
END;
```

The code is highlighted in blue. A yellow horizontal bar highlights the line "END;". Below the code area, there is a "Sortie de script" (Script Output) window. Its title bar says "Sortie de script x". The status bar in this window says "Tâche terminée en 0,015 secondes". The output message in the window is "bloc anonyme terminé".

Les déclencheurs de base de données

La suppression d'un trigger

The screenshot shows the Oracle SQL Developer interface. The top toolbar has various icons for database navigation and management. The main window title bar says "Feuille de calcul" and "Query Builder". Below the title bar, there is a SQL editor pane containing the following command:

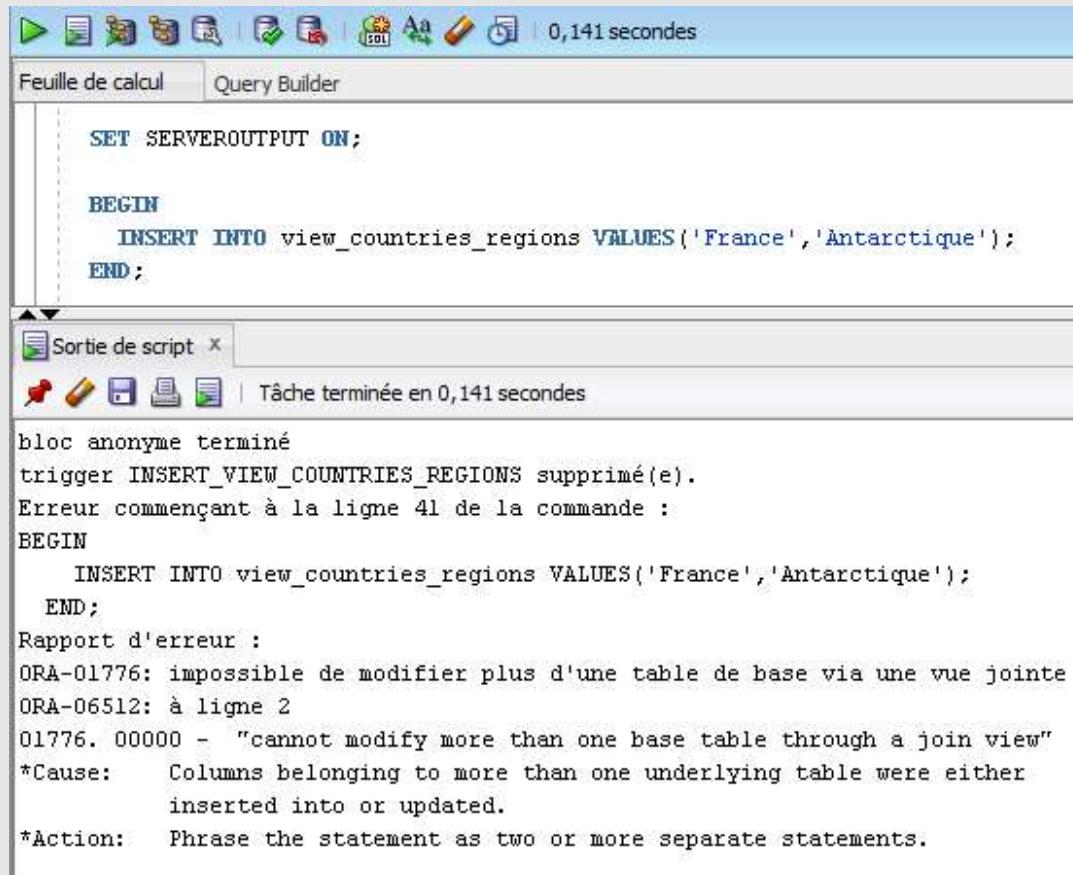
```
DROP TRIGGER insert_view_countries_regions;
```

Below the editor is a "Sortie de script" (Script Output) window. It shows the status message "Tâche terminée en 0,639 secondes" (Task completed in 0,639 seconds). The output pane displays the results of the executed command:

```
bloc anonyme terminé
trigger INSERT_VIEW_COUNTRIES_REGIONS supprimé(e).
```

Les déclencheurs de base de données

La suppression d'un trigger



The screenshot shows a SQL developer interface with a query builder window containing the following PL/SQL code:

```
SET SERVEROUTPUT ON;

BEGIN
    INSERT INTO view_countries_regions VALUES('France','Antarctique');
END;
```

The output window below shows the execution results:

```
Sortie de script x
Tâche terminée en 0,141 secondes

bloc anonyme terminé
trigger INSERT_VIEW_COUNTRIES_REGIONS supprimé(e).
Erreur commençant à la ligne 41 de la commande :
BEGIN
    INSERT INTO view_countries_regions VALUES('France','Antarctique');
END;
Rapport d'erreur :
ORA-01776: impossible de modifier plus d'une table de base via une vue jointe
ORA-06512: à ligne 2
01776. 00000 -  "cannot modify more than one base table through a join view"
*Cause:    Columns belonging to more than one underlying table were either
           inserted into or updated.
*Action:   Phrase the statement as two or more separate statements.
```

Les déclencheurs de base de données

Démonstration



Conclusion

- Vous comprenez le mécanisme des triggers
- Vous savez créer des triggers de table
- Vous savez créer des triggers de vue



PL / SQL

Module 10 – Les packages



Objectifs

- Comprendre l'utilité des packages
- Savoir créer un package
- Savoir utiliser un élément de package



Les packages

Définition

- Objet du schéma
- Permet d'améliorer la gestion des objets PL/SQL
- Permet de regrouper un ensemble d'objets homogènes



Les packages

Les avantages

- Structuration du développement
- Conception orientée objet
- Développement des composants
- Amélioration des performances



Les packages

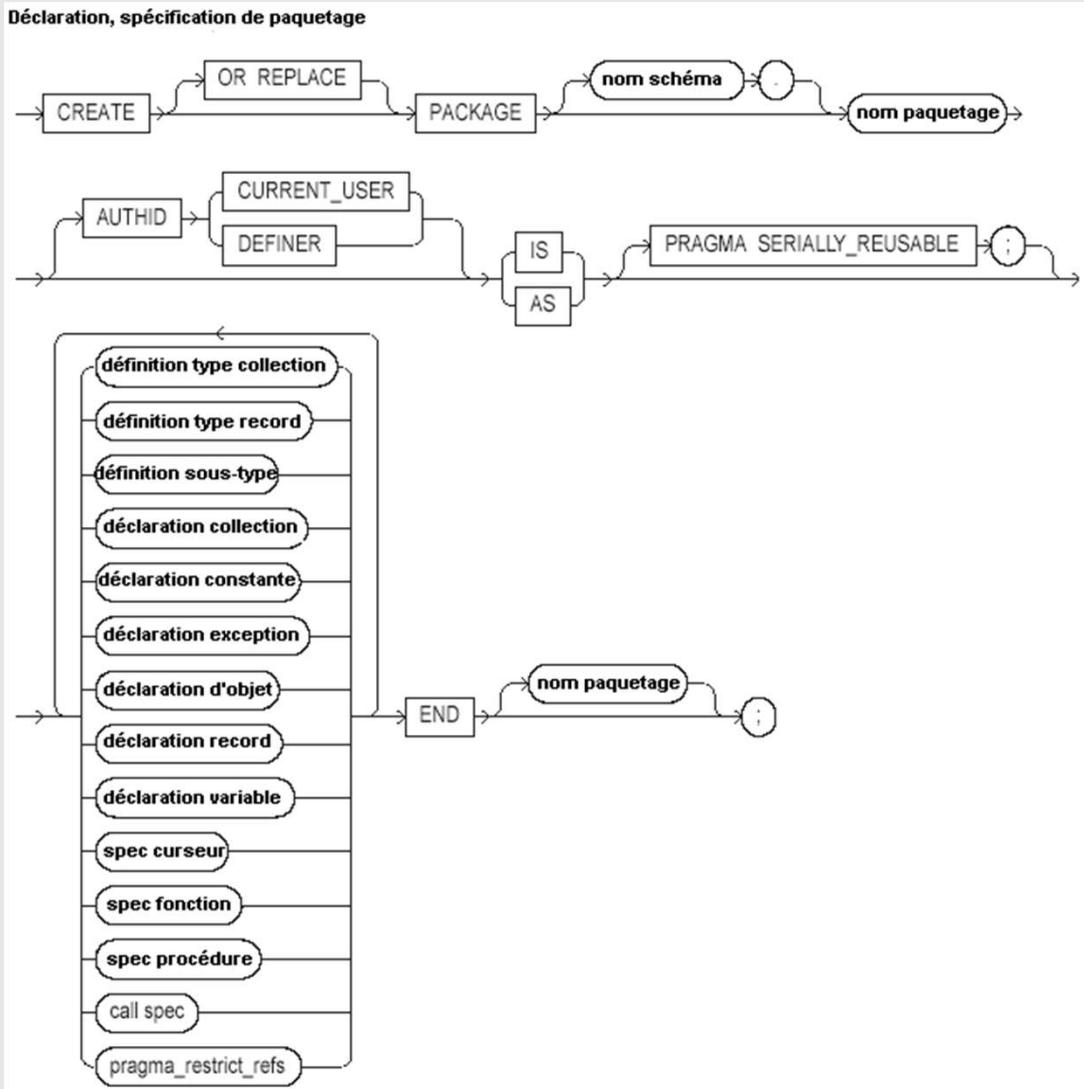
Les spécificités

- Composé de deux parties distinctes
 - Spécification
 - Corps
- Eléments pouvant être encapsulés
 - Variable
 - Constante
 - Exception
 - Procédure
 - Fonction



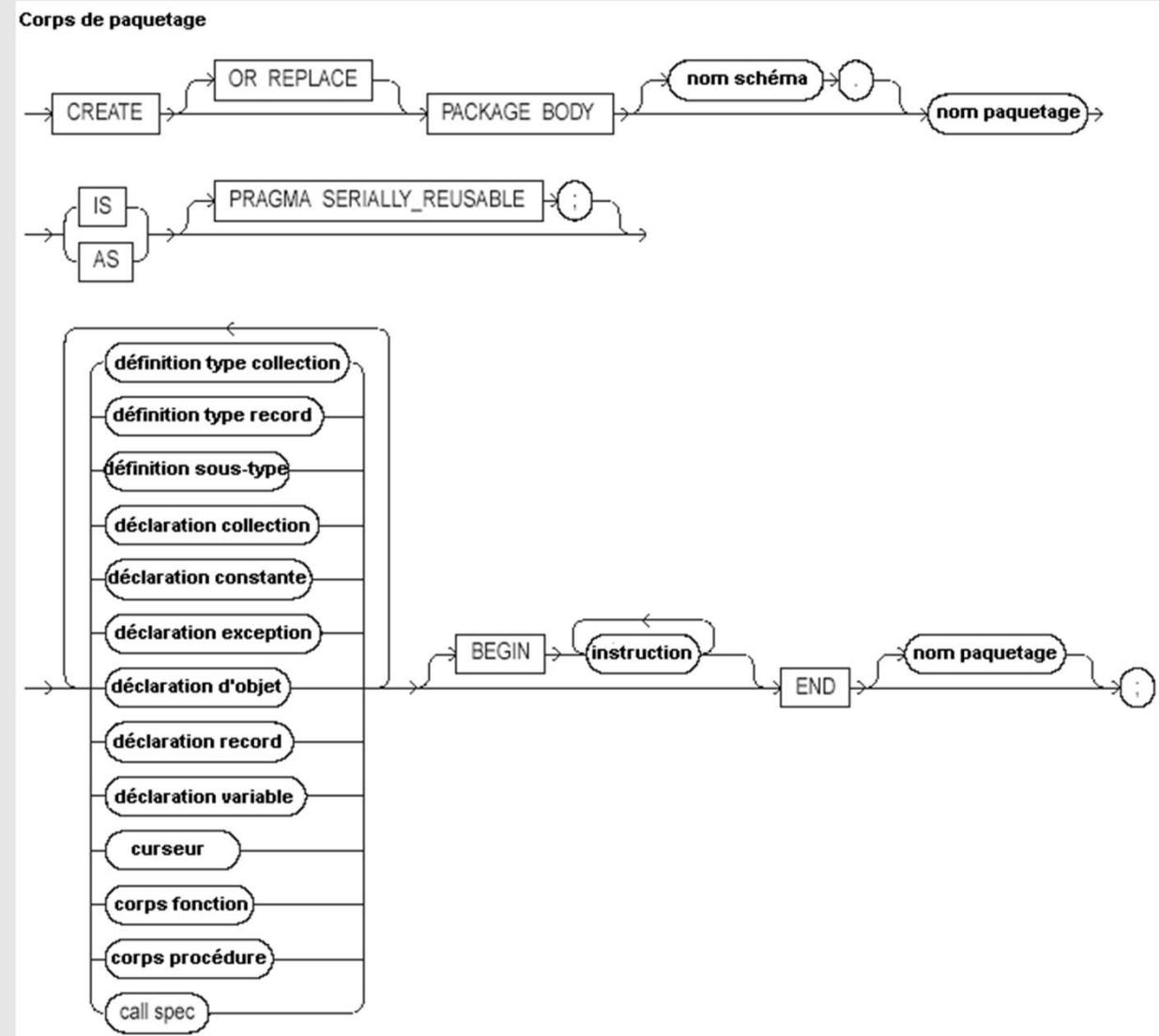
Les packages

La syntaxe de création de la partie spécification



Les packages

La syntaxe de création de la partie corps



Les packages

Le processus de création

1. Créer la partie Spécification
2. Créer la partie Corps



Les packages

Exemple d'une partie spécification

```
CREATE OR REPLACE PACKAGE pkg_regions
IS
    nom_region regions.region_name%TYPE;
    nom_region_default CONSTANT regions.region_name%TYPE := 'Europe';
    FUNCTION get_nombre_regions RETURN NUMBER;
    PROCEDURE formatter_regions;
END;
```



Les packages

Exemple d'une partie corps

```
CREATE OR REPLACE PACKAGE BODY pkg_regions
IS

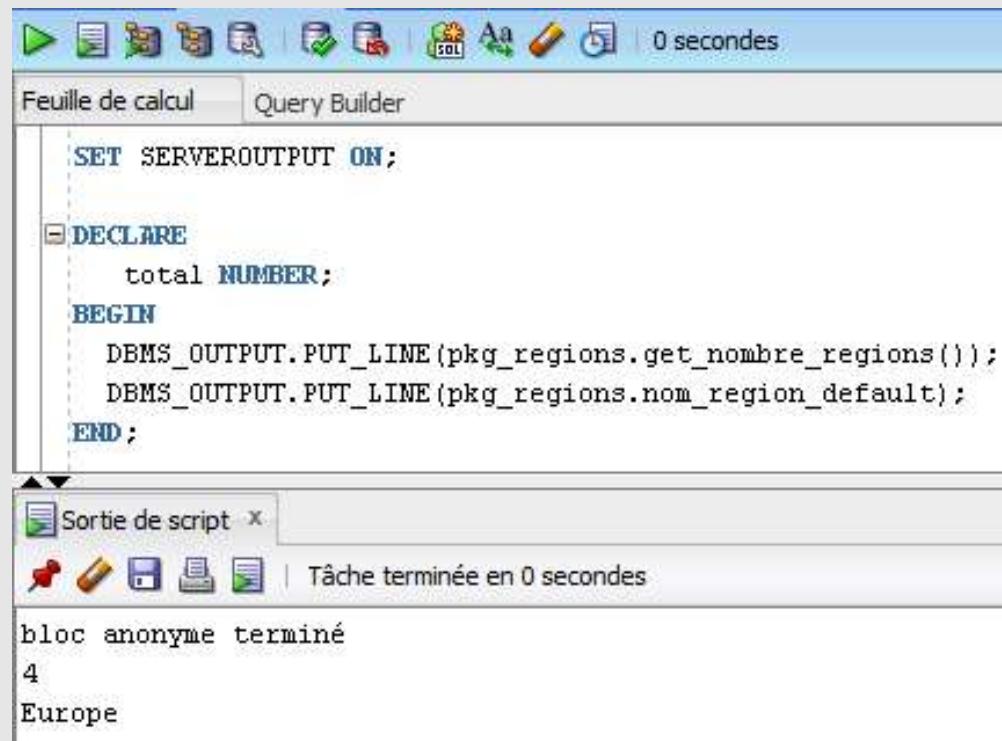
    FUNCTION get_nombre_regions RETURN NUMBER
    IS
        total NUMBER;
    BEGIN
        SELECT COUNT(*) INTO total FROM regions;
        RETURN total;
    END;

    PROCEDURE formatter_regions
    IS
    BEGIN
        UPDATE regions SET region_name = UPPER(region_name);
    END;

END;
```

Les packages

L'appel à un composant de package



```
SET SERVEROUTPUT ON;

DECLARE
    total NUMBER;
BEGIN
    DBMS_OUTPUT.PUT_LINE(pkg_regions.get_nombre_regions());
    DBMS_OUTPUT.PUT_LINE(pkg_regions.nom_region_default);
END;
```

Sortie de script

bloc anonyme terminé

4

Europe

Les packages

La suppression d'un package

```
DROP PACKAGE pkg_regions;
```



Les packages

Conclusion

- Vous comprenez l'utilité des packages
- Vous savez créer un package
- Vous savez supprimer un package
- Vous savez utiliser un package



PL / SQL

Module 11 – Le SQL dynamique



Objectifs

- Savoir expliquer ce qu'est le SQL dynamique
- Savoir quels sont les avantages du SQL dynamique
- Savoir utiliser le SQL dynamique



Le SQL dynamique

Les spécificités

- Permet de créer des traitements génériques
- Permet de construire dynamiquement des requêtes



Le SQL dynamique

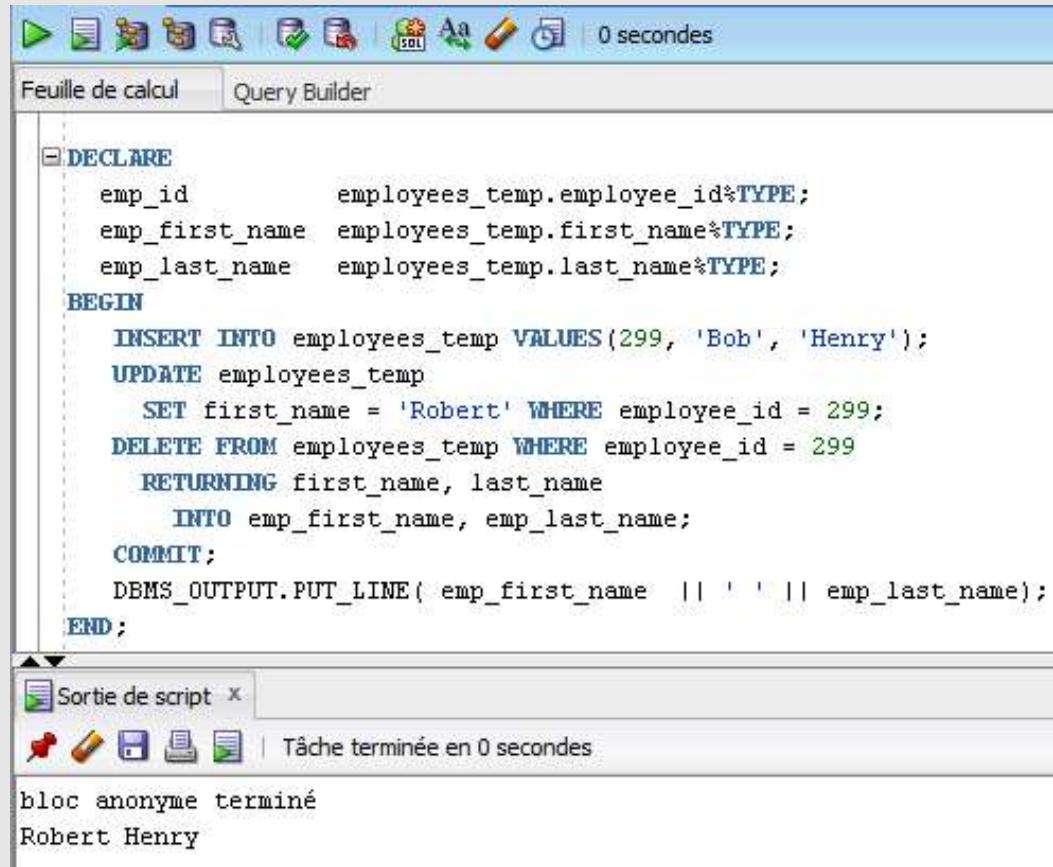
La clause RETURNING

```
CREATE TABLE employees_temp
AS SELECT employee_id, first_name, last_name
FROM employees;

DECLARE
    emp_id          employees_temp.employee_id%TYPE;
    emp_first_name  employees_temp.first_name%TYPE;
    emp_last_name   employees_temp.last_name%TYPE;
BEGIN
    INSERT INTO employees_temp VALUES(299, 'Bob', 'Henry');
    UPDATE employees_temp
        SET first_name = 'Robert' WHERE employee_id = 299;
    DELETE FROM employees_temp WHERE employee_id = 299
        RETURNING first_name, last_name
        INTO emp_first_name, emp_last_name;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE( emp_first_name || ' ' || emp_last_name);
END;
```

Le SQL dynamique

Exemple d'utilisation de la clause RETURNING



The screenshot shows the Oracle SQL Developer interface with a query editor window. The code in the editor is:

```
DECLARE
    emp_id      employees_temp.employee_id%TYPE;
    emp_first_name  employees_temp.first_name%TYPE;
    emp_last_name   employees_temp.last_name%TYPE;
BEGIN
    INSERT INTO employees_temp VALUES(299, 'Bob', 'Henry');
    UPDATE employees_temp
        SET first_name = 'Robert' WHERE employee_id = 299;
    DELETE FROM employees_temp WHERE employee_id = 299
        RETURNING first_name, last_name
        INTO emp_first_name, emp_last_name;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE( emp_first_name || ' ' || emp_last_name);
END;
```

Below the editor, a 'Sortie de script' (Script Output) window is open, showing the result of the execution:

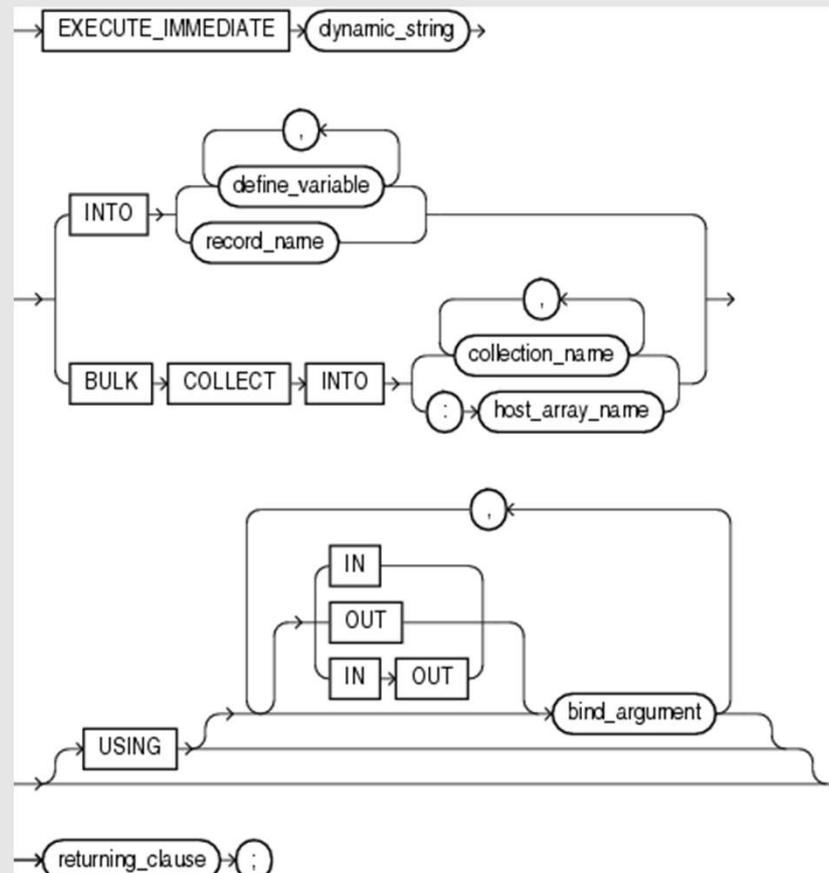
Tâche terminée en 0 secondes

bloc anonyme terminé

Robert Henry

Le SQL dynamique

La syntaxe de l'instruction EXECUTE IMMEDIATE



Le SQL dynamique

Exemple d'utilisation du EXECUTE IMMEDIATE

```
DECLARE
    requete VARCHAR2(256) ;
    nouveau_nom      VARCHAR(20) := 'Jojo';
    id_employe       employees.employee_id%TYPE := 100;
BEGIN
    requete := 'UPDATE employees SET FIRST_NAME = :1 WHERE employee_id = :2';
    EXECUTE IMMEDIATE requete USING nouveau_nom, id_employe;
END ;
```

Le SQL dynamique

Exemple d'utilisation du EXECUTE IMMEDIATE

The screenshot shows the Oracle SQL Developer interface. The top menu bar includes icons for file, edit, search, and database operations, along with a toolbar below it. The main window has tabs for 'Feuille de calcul' and 'Query Builder'. The code editor contains the following PL/SQL block:

```
DECLARE
    requete VARCHAR2(256) ;
    nouveau_nom      VARCHAR(20) := 'Jojo';
    id_employe       employees.employee_id%TYPE := 100;
BEGIN
    requete := 'UPDATE employees SET FIRST_NAME = :1 WHERE employee_id = :2';
    EXECUTE IMMEDIATE requete USING nouveau_nom, id_employe;
END ;
```

Below the code, a 'SELECT * FROM employees WHERE employee_id = 100;' statement is highlighted in yellow.

The bottom section shows the execution results in three tabs: 'Sortie de script', 'Enregistrer la trace automatiquement', and 'Résultat de requête'. The 'Résultat de requête' tab displays the following table:

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE
1	100	Jojo	King	SKING	515.123.4567	17/06/03

Le SQL dynamique

Exemple d'utilisation du EXECUTE IMMEDIATE

```
SET SERVEROUTPUT ON;

DECLARE
    total number;
    table_recherche VARCHAR(30) := 'regions';
    requete VARCHAR(100);
BEGIN
    requete := 'SELECT COUNT(*) FROM ' || table_recherche;
    EXECUTE IMMEDIATE requete INTO total;
    DBMS_OUTPUT.PUT_LINE(total);
END ;
```

Conclusion

- Vous savez expliquer ce qu'est le SQL dynamique
- Vous savez quels sont les avantages du SQL dynamique
- Vous savez utiliser le SQL dynamique



PL / SQL

Module 12 – Les transactions



Objectifs

- Savoir expliquer ce qu'est une transaction
- Savoir gérer des transactions
- Savoir expliquer ce qu'est une transaction autonome



Les transactions

Définition

- Gère un ensemble d'instructions du DML.
- S'assure que toutes les instructions ont été effectuées avant de les appliquer définitivement.
- Permet un retour en arrière en cas d'erreur.



Les transactions

Les spécificités

- Une seule transaction principale active
- Il y a toujours une transaction principale active en cours
- Le démarrage d'une transaction principale est automatique
- La transaction peut être validée ou invalidée
- La transaction permet d'assurer la cohérence des données vues par chaque utilisateur connecté



Les transactions

La validation d'une transaction : COMMIT

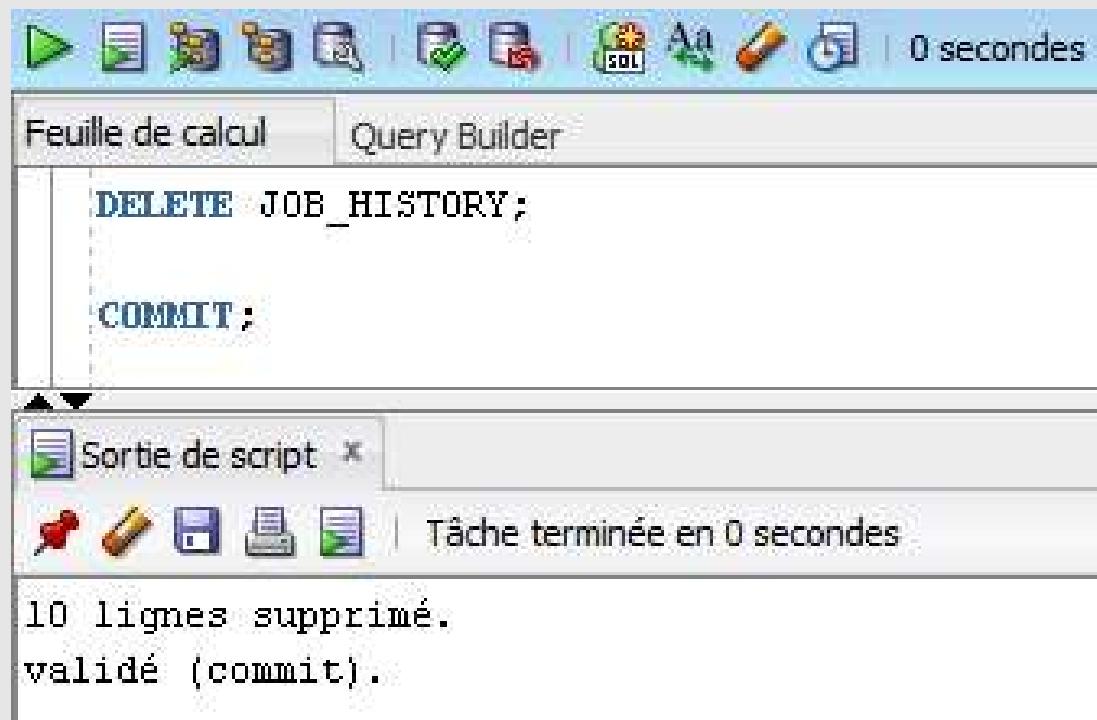
Enregistre en base toutes les insertions, modifications et suppressions.

Tant qu'il n'y a pas eu COMMIT, seule la connexion courante voit ses mises à jour.



Les transactions

Exemple de validation d'une transaction



The screenshot shows a database interface with a toolbar at the top containing various icons. Below the toolbar, there are two tabs: "Feuille de calcul" (selected) and "Query Builder". The main area contains the following SQL code:

```
DELETE FROM JOB_HISTORY;  
COMMIT;
```

Below the code, a message box titled "Sortie de script" displays the results of the execution:

Tâche terminée en 0 secondes

10 lignes supprimé.
validé (commit).

Les transactions

L'invalidation d'une transaction : ROLLBACK

Annule toutes les insertions, modifications et suppressions depuis le début de la transaction.



Les transactions

Exemple d'invalidation d'une transaction

The screenshot shows a database interface with the following details:

- Toolbar:** Includes icons for play, stop, refresh, and various database operations.
- Top Status Bar:** Shows "0,062 secondes".
- Tab Bar:** "Feuille de calcul" is selected, while "Query Builder" is also visible.
- Script Area:** Contains the following SQL code:

```
DELETE FROM JOB_HISTORY;  
ROLLBACK;
```
- Output Window:** Titled "Sortie de script", it shows the results of the executed command:

```
Tâche terminée en 0,062 secondes  
10 lignes supprimé.  
annulation (rollback) effectuée.
```

Les transactions

Les transactions autonomes

- Transaction devant être exécutée indépendamment de la transaction appelante.
- La transaction appelante est suspendue temporairement.
- La transaction autonome est contrôlée par les ordres COMMIT et ROLLBACK.
- Mécanisme activé grâce à la directive de compilation (**pragma**)
AUTONOMOUS_TRANSACTION.



Les transactions

Exemple de transaction autonome

```
CREATE OR REPLACE
PROCEDURE add_information(message VARCHAR2) IS
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
    INSERT INTO informations (INFORMATIONS_TEXT) VALUES (message);
    COMMIT;
END;
```

Les transactions

Exemple de transaction autonome

```
BEGIN
    INSERT INTO regions values (100,'Paradis');
    COMMIT;
    add_information('Region insérée');
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        add_information('La region existe déjà');
        ROLLBACK;
    WHEN OTHERS THEN
        add_information('Erreur inconnue à l''insertion');
        ROLLBACK;
END;
```

Les transactions

Conclusion

- Vous savez utiliser les transactions
- Vous savez utiliser les transactions autonomes

