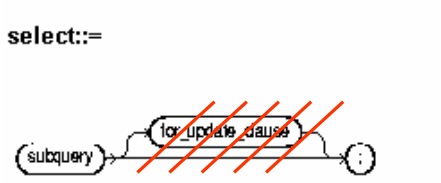
Chapitre II

Second noyau de SQL

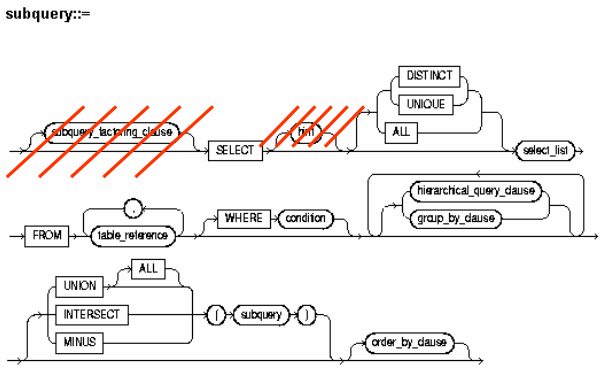
2. Second noyau de SQL

- 2.1 – Requêtes et sous-requêtes
- 2.2 – Création des tables
- 2.3 – Modifications des contenus des tables
- 2.4 – Création des vues
- 2.5 – Fonctions
- 2.6 – Conclusions

2.1- Requêtes et sous-requêtes



Sous-requête



Equijointure

```
SELECT ename, job, dept.deptno, dname
     FROM emp, dept
     WHERE emp.deptno = dept.deptno;

ENAME      JOB       DEPTNO     DNAME
---------- --------- ---------- --------------
CLARK      MANAGER           10 ACCOUNTING
KING       PRESIDENT         10 ACCOUNTING
MILLER     CLERK             10 ACCOUNTING
SMITH      CLERK             20 RESEARCH
ADAMS      CLERK             20 RESEARCH
FORD       ANALYST           20 RESEARCH
SCOTT      ANALYST           20 RESEARCH
JONES      MANAGER           20 RESEARCH
ALLEN      SALESMAN          30 SALES
BLAKE      MANAGER           30 SALES
MARTIN     SALESMAN          30 SALES
JAMES      CLERK             30 SALES
TURNER     SALESMAN          30 SALES
WARD       SALESMAN          30 SALES
```

Sous-requête

```
SELECT ename, deptno
    FROM emp
    WHERE deptno =
        (SELECT deptno
            FROM emp
            WHERE ename = 'TAYLOR');
```

Auto-jointure

```
SELECT e1.ename||' works for '||e2.ename
"Employees and their Managers"
    FROM emp e1, emp e2   WHERE e1.mgr = e2.empno;

Employees and their Managers
--------------------------------
BLAKE works for KING
CLARK works for KING
JONES works for KING
FORD works for JONES
SMITH works for FORD
ALLEN works for BLAKE
WARD works for BLAKE
MARTIN works for BLAKE
SCOTT works for JONES
TURNER works for BLAKE
ADAMS works for SCOTT
JAMES works for BLAKE
MILLER works for CLARK
```

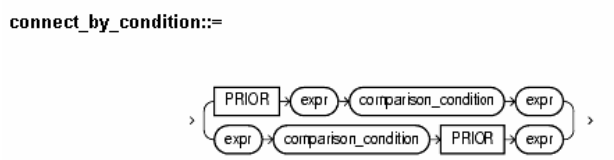
Jointure externe

```
SELECT ename, job, dept.deptno, dname
    FROM emp, dept
    WHERE emp.deptno (+) = dept.deptno;

ENAME      JOB       DEPTNO     DNAME
---------- --------- ---------- --------------
CLARK      MANAGER           10 ACCOUNTING
KING       PRESIDENT         10 ACCOUNTING
MILLER     CLERK             10 ACCOUNTING
SMITH      CLERK             20 RESEARCH
ADAMS      CLERK             20 RESEARCH
FORD       ANALYST           20 RESEARCH
SCOTT      ANALYST           20 RESEARCH
JONES      MANAGER           20 RESEARCH
ALLEN      SALESMAN          30 SALES
BLAKE      MANAGER           30 SALES
MARTIN     SALESMAN          30 SALES
JAMES      CLERK             30 SALES
TURNER     SALESMAN          30 SALES
WARD       SALESMAN          30 SALES
                             40 OPERATIONS
```

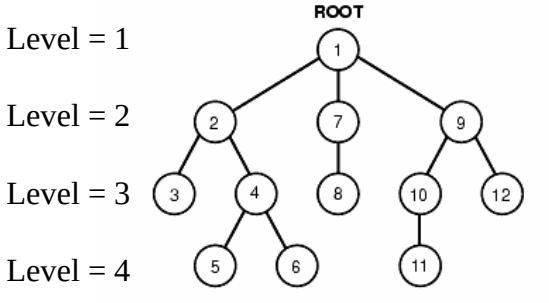
Requête hiérarchique

hierarchical_query_clause::=

connect_by_condition::=



Arbre



Exemple de requête hiérarchique

```
SELECT employee_id, last_name, manager_id
   FROM employees
   CONNECT BY PRIOR employee_id = manager_id;

EMPLOYEE_ID LAST_NAME                 MANAGER_ID
----------- ------------------------- ----------
        101 Kochhar                           100
        108 Greenberg                         101
        109 Faviet                            108
        110 Chen                              108
        111 Sciarra                           108
        112 Urman                             108
        113 Popp                              108
        200 Whalen                            101
```

Exemple avec niveau

```
SELECT employee_id, last_name, manager_id, LEVEL
   FROM employees
   CONNECT BY PRIOR employee_id = manager_id;

EMPLOYEE_ID LAST_NAME                 MANAGER_ID      LEVEL
----------- ------------------------- ---------- ----------
        101 Kochhar                           100          1
        108 Greenberg                         101          2
        109 Faviet                            108          3
        110 Chen                              108          3
        111 Sciarra                           108          3
        112 Urman                             108          3
        113 Popp                              108          3
```

UNION

```
SELECT location_id, department_name "Department",
   TO_CHAR(NULL) "Warehouse"  FROM departments
   UNION
   SELECT location_id, TO_CHAR(NULL) "Department", warehouse_name
   FROM warehouses;

LOCATION_ID Department           Warehouse
----------- -------------------- -------------------------
       1400 IT
       1400                     Southlake, Texas
       1500 Shipping
       1500                     San Francisco
       1600                     New Jersey
       1700 Accounting
       1700 Administration
       1700 Benefits
       1700 Construction
```

UNION et UNION ALL

```
SELECT product_id
FROM order_items
UNION
SELECT product_id
FROM inventories;
```

```
SELECT location_id
FROM locations
UNION ALL
SELECT location_id
FROM departments;
```

INTERSECT et MINUS

```
SELECT product_id
FROM inventories
INTERSECT
SELECT product_id
FROM order_items;
```

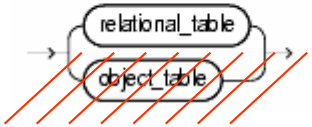
```
SELECT product_id
FROM inventories
MINUS
SELECT product_id
FROM order_items;
```

Requêtes distribuées

```
SELECT employees_ny.*
FROM employees_ny@ny, departments
WHERE
employees_ny.department_id = departments.department_id
AND
departments.department_name = 'ACCOUNTING';
```

2.2 - Création de tables

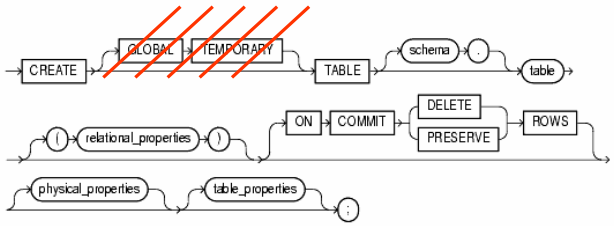
create_table::=




```
CREATE OR REPLACE TABLE….
```

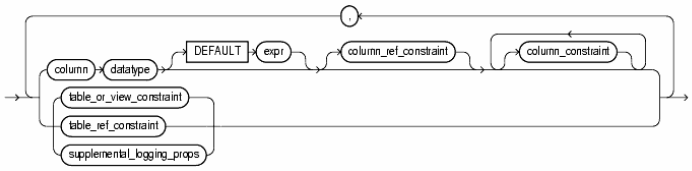
Relational table

relational_table::=



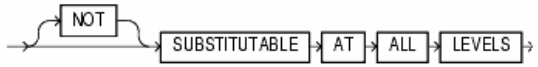
Propriétés relationnelles

relational_properties::=



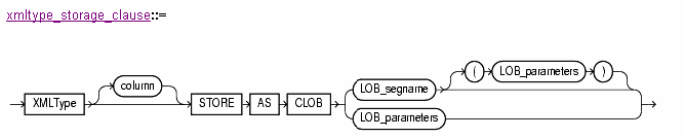
Substitution de tables objets

object_table_substitution::=



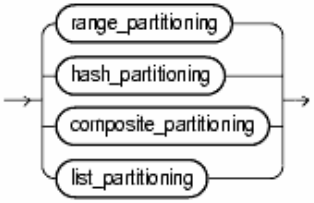
Stockage en XML

xmltype_storage_clause::=



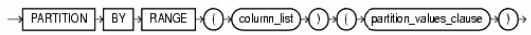
Partitionnement de table

table_partitioning_clauses::=



Partitionnement

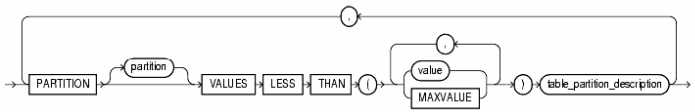
range_partitioning::=



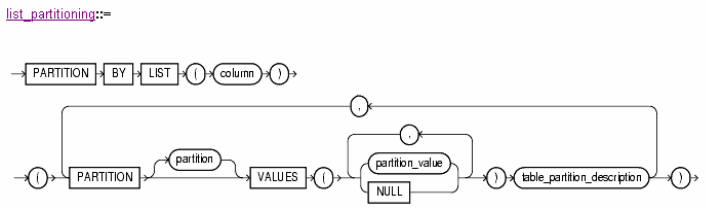
composite_partitioning::=

Partitionnement par valeur

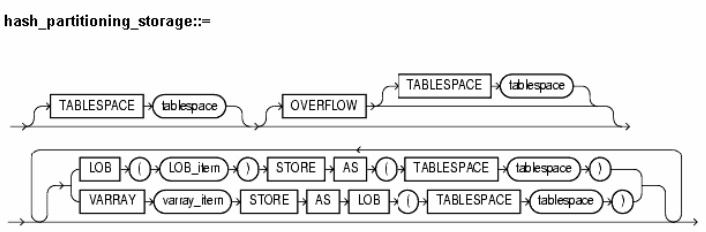
partition_values_clause::=



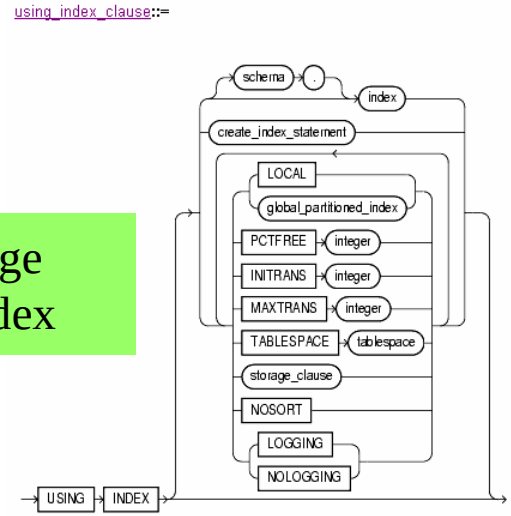
Partitionnement de listes



Partitionnement par hashing

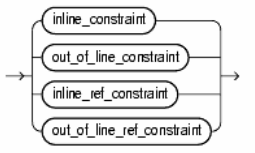


Usage d'index

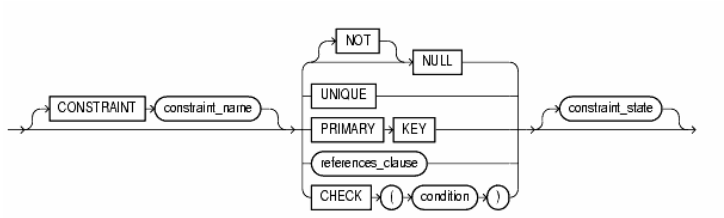


Contraintes d'intégrité déclaratives

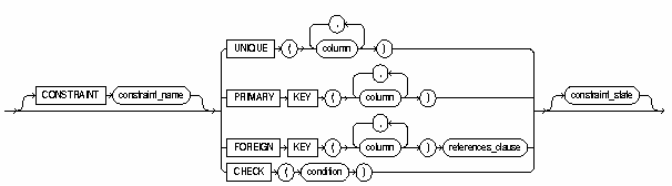
- Type de contraintes déclaratives
  - NOT NULL
  - Unicité (null acceptés)
  - Clé primaire
  - Clé étrangère et références
  - Petites vérifications
- Localisation
  - Dans une colonne
  - Dans la table
- Activation / désactivation



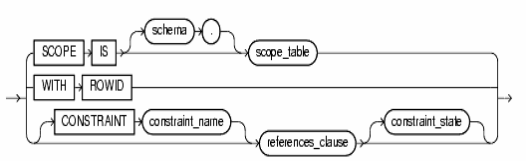
In_line constraint



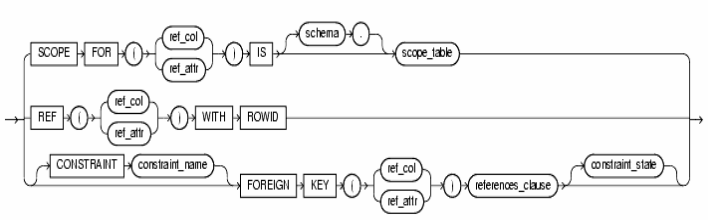
Out_of_line constraint



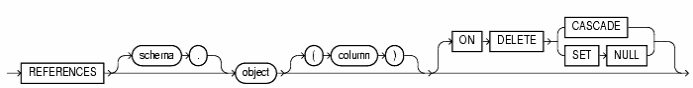
inline_ref_constraint::=



out_of_line_ref_constraint::=



REFERENCES



Exemple avec contraintes

```
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  DEFAULT SYSDATE
        CONSTRAINT emp_hire_date_nn  NOT NULL
    , job_id          VARCHAR2(10)
      CONSTRAINT     emp_job_nn  NOT NULL
    , salary          NUMBER(8,2)
      CONSTRAINT     emp_salary_nn  NOT NULL
    , commission_pct NUMBER(2,2)
    , manager_id      NUMBER(6)
    , department_id   NUMBER(4)
    , dn              VARCHAR2(300)
    , CONSTRAINT      emp_salary_min
                      CHECK (salary > 0)
    , CONSTRAINT      emp_email_uk
                      UNIQUE (email)
    ) ;
```

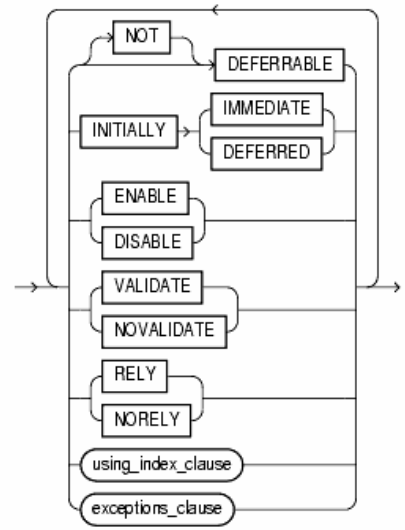
```
CREATE TABLE divisions
   (div_no    NUMBER  CONSTRAINT check_divno
              CHECK (div_no BETWEEN 10 AND 99)
              DISABLE,
    div_name  VARCHAR2(9)  CONSTRAINT check_divname
              CHECK (div_name = UPPER(div_name))
              DISABLE,
    office    VARCHAR2(10)  CONSTRAINT check_office
              CHECK (office IN ('DALLAS','BOSTON',
              'PARIS','TOKYO'))
              DISABLE);
```

```
CREATE TABLE dept_20
   (employee_id     NUMBER(4) PRIMARY KEY,
    last_name       VARCHAR2(10),
    job_id          VARCHAR2(9),
    manager_id      NUMBER(4),
    salary          NUMBER(7,2),
    commission_pct  NUMBER(7,2),
    department_id   NUMBER(2),
    CONSTRAINT check_sal CHECK (salary * commission_pct <= 5000));
```

```
CREATE TABLE order_detail
  (CONSTRAINT pk_od PRIMARY KEY (order_id, part_no),
   order_id    NUMBER
      CONSTRAINT fk_oid
         REFERENCES oe.orders(order_id),
   part_no     NUMBER
      CONSTRAINT fk_pno
         REFERENCES oe.product_information(product_id),
   quantity    NUMBER
      CONSTRAINT nn_qty NOT NULL
      CONSTRAINT check_qty CHECK (quantity > 0),
   cost        NUMBER
      CONSTRAINT check_cost CHECK (cost > 0) );
```

```
ALTER TABLE dept_20
   ADD CONSTRAINT fk_empid_hiredate
   FOREIGN KEY (employee_id, hire_date)
   REFERENCES hr.job_history(employee_id, start_date)
   EXCEPTIONS INTO wrong_emp;
```

Etat des contraintes



Exemple de désactivation de contraintes

```
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)
    , dn               VARCHAR2(300)
    ) ;

CREATE TABLE departments
    ( department_id    NUMBER(4)   PRIMARY KEY DISABLE
    , department_name  VARCHAR2(30)
          CONSTRAINT  dept_name_nn  NOT NULL
    , manager_id       NUMBER(6)
    , location_id      NUMBER(4)
    , dn               VARCHAR2(300)
    ) ;
```

Désactivation de contrainte

Exemple avec description du stockage

```
CREATE TABLE salgrade
   ( grade  NUMBER  CONSTRAINT pk_salgrade
                    PRIMARY KEY
                    USING INDEX TABLESPACE users_a,
     losal  NUMBER,
     hisal  NUMBER )
   TABLESPACE human_resource
   STORAGE (INITIAL     6144
            NEXT        6144
            MINEXTENTS     1
            MAXEXTENTS     5 );
```

Large Objects (LOB)

- LOB interne
  - CLOB (Character Large Object) : chaînes de caractères.
  - BLOB (Binary Large Object): données binaires
  - NCLOB (National Character Large Object) : les chaînes de caractères Unicode.
- LOB externe
  - BFILE (Binary File): pour les données stockées dans le système de fichier du système d'exploitation.

Tables avec LOB

```
CREATE TABLE print_media
    ( product_id        NUMBER(6)
    , ad_id             NUMBER(6)
    , ad_composite      BLOB
    , ad_sourcetext     CLOB
    , ad_finaltext      CLOB
    , ad_fltextn        NCLOB
    , ad_textdocs_ntab  textdoc_tab
    , ad_photo          BLOB
    , ad_graphic        BFILE
    , ad_header         adheader_typ
    , press_release     LONG
    ) NESTED TABLE ad_textdocs_ntab STORE AS textdocs_nestedtab
    LOB (ad_sourcetext, ad_finaltext), STORE AS
      (TABLESPACE lob_seg_ts
       STORAGE (INITIAL 6144 NEXT 6144)
       CHUNK 4000
       NOCACHE LOGGING);
```

Tables emboîtées

```
CREATE TABLE print_media
    ( product_id        NUMBER(6)
    , ad_id             NUMBER(6)
    , ad_composite      BLOB
    , ad_sourcetext     CLOB
    , ad_finaltext      CLOB
    , ad_fltextn        NCLOB
    , ad_textdocs_ntab  textdoc_tab
    , ad_photo          BLOB
    , ad_graphic        BFILE
    , ad_header         adheader_typ
    , press_release     LONG
    ) NESTED TABLE ad_textdocs_ntab STORE AS textdocs_nestedtab;
```

Table avec partionnement en intervalle

```
CREATE TABLE range_sales
    ( prod_id        NUMBER(6)
    , cust_id        NUMBER
    , time_id        DATE
    , channel_id     CHAR(1)
    , promo_id       NUMBER(6)
    , quantity_sold  NUMBER(3)
    , amount_sold          NUMBER(10,2)
    )
PARTITION BY RANGE (time_id)
  (PARTITION SALES_Q1_1998 VALUES LESS THAN (TO_DATE('01-APR-1998','DD-MON-YYYY')),
   PARTITION SALES_Q2_1998 VALUES LESS THAN (TO_DATE('01-JUL-1998','DD-MON-YYYY')),
   PARTITION SALES_Q3_1998 VALUES LESS THAN (TO_DATE('01-OCT-1998','DD-MON-YYYY')),
   PARTITION SALES_Q4_1998 VALUES LESS THAN (TO_DATE('01-JAN-1999','DD-MON-YYYY')),
   PARTITION SALES_Q1_1999 VALUES LESS THAN (TO_DATE('01-APR-1999','DD-MON-YYYY')),
   PARTITION SALES_Q2_1999 VALUES LESS THAN (TO_DATE('01-JUL-1999','DD-MON-YYYY')),
   PARTITION SALES_Q3_1999 VALUES LESS THAN (TO_DATE('01-OCT-1999','DD-MON-YYYY')),
   PARTITION SALES_Q4_1999 VALUES LESS THAN (TO_DATE('01-JAN-2000','DD-MON-YYYY')),
   PARTITION SALES_Q1_2000 VALUES LESS THAN (TO_DATE('01-APR-2000','DD-MON-YYYY')),
   PARTITION SALES_Q2_2000 VALUES LESS THAN (TO_DATE('01-JUL-2000','DD-MON-YYYY')),
   PARTITION SALES_Q3_2000 VALUES LESS THAN (TO_DATE('01-OCT-2000','DD-MON-YYYY')),
   PARTITION SALES_Q4_2000 VALUES LESS THAN (MAXVALUE))
;
```

Table avec partitionnement de type Hashing

```
CREATE TABLE product_information
    ( product_id          NUMBER(6)
    , product_name        VARCHAR2(50)
    , product_description VARCHAR2(2000)
    , category_id         NUMBER(2)
    , weight_class        NUMBER(1)
    , warranty_period     INTERVAL YEAR TO MONTH
    , supplier_id         NUMBER(6)
    , product_status      VARCHAR2(20)
    , list_price          NUMBER(8,2)
    , min_price           NUMBER(8,2)
    , catalog_url         VARCHAR2(50)
    , CONSTRAINT          product_status_lov
                          CHECK (product_status in ('orderable'
                                                    ,'planned'
                                                    ,'under development'
                                                    ,'obsolete')
    ) )
    PARTITION BY HASH (product_id)
    PARTITIONS 5
    STORE IN (prod_ts1, prod_ts2, prod_ts3, prod_ts4, prod_ts5);
```

Tables avec objets

```
CREATE TYPE dept_t AS OBJECT
   ( dname   VARCHAR2(100),
     address VARCHAR2(200) );
```

```
CREATE OR REPLACE TYPE salesrep_t AS OBJECT
  ( repId NUMBER,
    repName VARCHAR2(64));
CREATE TABLE salesreps OF salesrep_t;
```

Usage de type utilisateur

```
CREATE TABLE emp
   ( ename   VARCHAR2(100),
     enumber NUMBER,
     edept   REF dept_t SCOPE IS dept );
```

Référence simple

```
CREATE TABLE emp
   ( ename   VARCHAR2(100),
     enumber NUMBER,
     edept   REF dept_t REFERENCES dept);
```

Référence avec contrainte référentielle

Objets avec contraintes

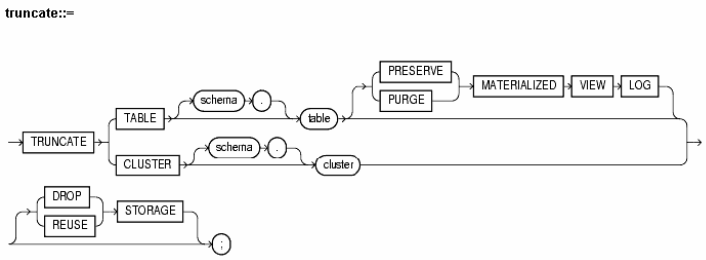
```
CREATE TYPE address_t AS OBJECT
  ( hno    NUMBER,
    street VARCHAR2(40),
    city   VARCHAR2(20),
    zip    VARCHAR2(5),
    phone  VARCHAR2(10) );

CREATE TYPE person AS OBJECT
  ( name        VARCHAR2(40),
    dateofbirth DATE,
    homeaddress address,
    manager     REF person );

CREATE TABLE persons OF person
  ( homeaddress NOT NULL,
      UNIQUE (homeaddress.phone),
      CHECK (homeaddress.zip IS NOT NULL),
      CHECK (homeaddress.city <> 'San Francisco') );
```

DROP TABLE

drop_table::=

TRUNCATE

truncate::=

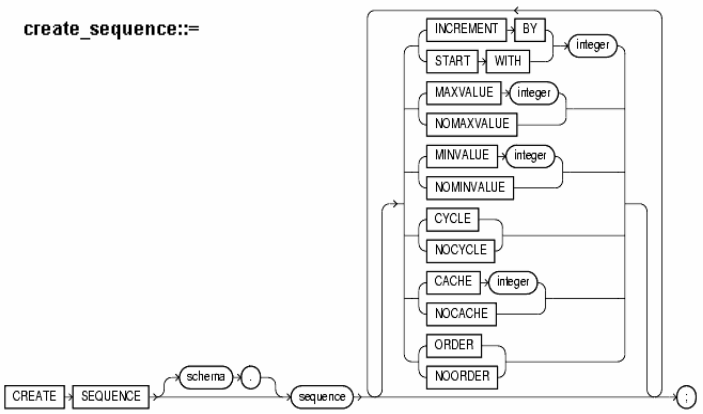


Exemples

```
TRUNCATE TABLE employees;
```

```
TRUNCATE CLUSTER personnel REUSE STORAGE
```

Sequence



Exemple

```
CREATE SEQUENCE orders_seq
 START WITH     1000
 INCREMENT BY   1
 NOCACHE
 NOCYCLE;
```

CURRVAL = valeur en cours
NEXTVAL = incrémente et donne valeur suivante

Exemples d'utilisation

```
orders_seq.nextval
ordres_seq.currval
```

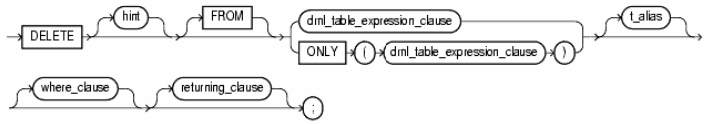
2.3 - Modifications des contenus

- DELETE
- INSERT
- UPDATE

- COMMIT et ROLLBACK

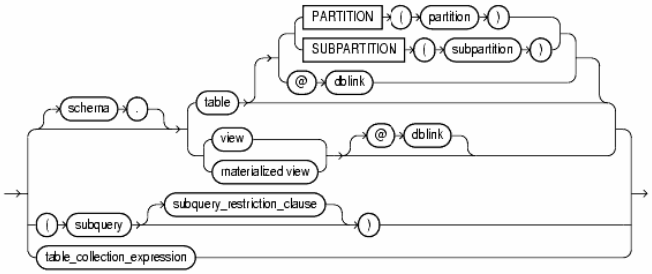
2.3.1 - DELETE

delete::=

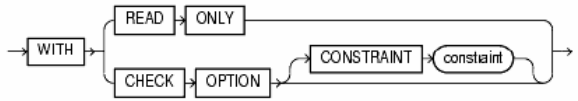


Remarque : hint = suggestion pour l'optimisation

dml_table_expression_clause::=



subquery_restriction_clause::=



table_collection_expression::=

where_clause::=

returning_clause::=

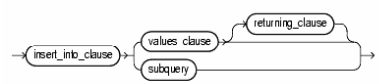
Exemples

```
DELETE FROM product_descriptions;


DELETE FROM employees
    WHERE job_id = 'PU_CLERK'
    AND commission_pct < .1;


DELETE FROM blake.accounts@dallas;

DELETE FROM sales PARTITION (sales_q1_1998)
   WHERE amount_sold != 0;
```

2.3.2 - INSERT

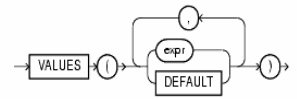
insert::=

single_table_insert::=



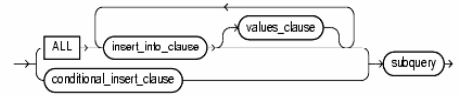
insert_into_clause::=

values_clause::=

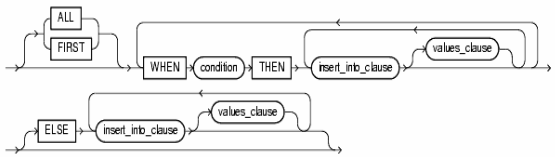


returning_clause::=

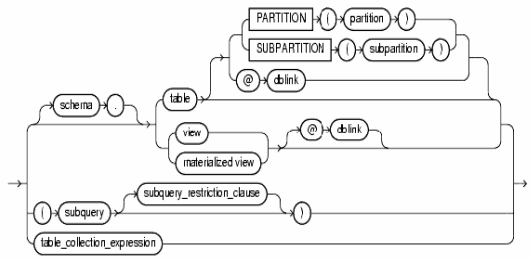
multi_table_insert::=



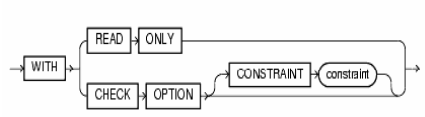
conditional_insert_clause::=



dml_table_expression_clause::=



subquery_restriction_clause::=



table_collection_expression::=

Exemples

```
INSERT INTO departments
   VALUES (280, 'Recreation', 121, 1700);

INSERT INTO departments
   VALUES (280, 'Recreation', DEFAULT, 1700);

INSERT INTO employees (employee_id, last_name, email,
      hire_date, job_id, salary, commission_pct)
   VALUES (207, 'Gregory', 'pgregory@oracle.com',
      sysdate, 'PU_CLERK', 1.2E3, NULL);

INSERT INTO
   (SELECT employee_id, last_name, email, hire_date, job_id,
      salary, commission_pct FROM employees)
   VALUES (207, 'Gregory', 'pgregory@oracle.com',
      sysdate, 'PU_CLERK', 1.2E3, NULL);
```

Autres exemples

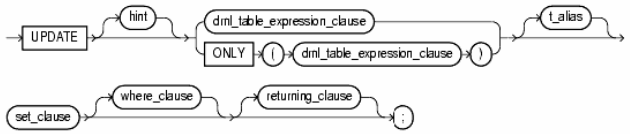
```
INSERT INTO bonuses
   SELECT employee_id, salary*1.1
   FROM employees
   WHERE commission_pct > 0.25 * salary;

INSERT INTO scott.accounts@sales (acc_no, acc_name)
   VALUES (5001, 'BOWER');

INSERT INTO departments
   VALUES  (departments_seq.nextval, 'Entertainment', 162, 1400);

INSERT INTO sales PARTITION (oct98)
   SELECT * FROM latest_data;
```

Exemple LOB

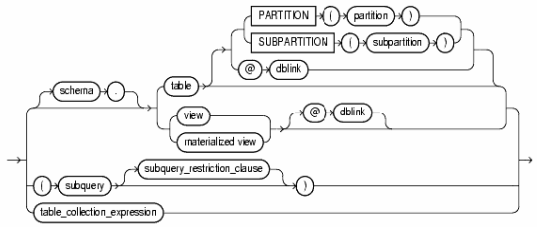
```
CREATE TABLE long_tab (long_pics LONG RAW);

CREATE TABLE lob_tab (lob_pics BLOB);

INSERT INTO lob_tab (lob_pics)
   SELECT TO_LOB(long_pics) FROM long_tab;
```

2.3.3 - UPDATE

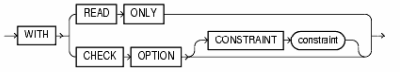
update::=



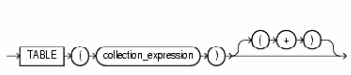
dml_table_expression_clause::=



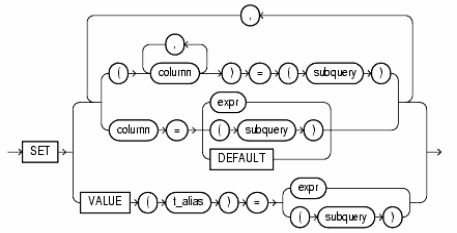
subquery_restriction_clause::=



table_collection_expression::=



update_set_clause::=



where_clause::=

returning_clause::=

Exemples

```
UPDATE employees
    SET commission_pct = NULL
    WHERE job = 'SA_CLERK';
```

```
UPDATE employees SET
    job_id = 'SA_MAN', salary = salary + 1000, department_id = 120
    WHERE first_name||' '||last_name = 'Douglas Grant';
```

```
UPDATE accounts@boston
    SET balance = balance + 500
    WHERE acc_no = 5001;
```

```
UPDATE sales PARTITION (sales_q1_1999) s
   SET s.promo_id = 494;
```

```
UPDATE employees a
    SET department_id =
        (SELECT department_id
            FROM departments
            WHERE location_id = '2100'),
        (salary, commission_pct) =
        (SELECT 1.1*AVG(salary), 1.5*AVG(commission_pct)
          FROM employees b
          WHERE a.department_id = b.department_id)
    WHERE department_id IN
        (SELECT department_id
          FROM departments
          WHERE location_id = 2900
              OR location_id = 2700);
```

```
UPDATE table1 p SET VALUE(p) =
   (SELECT VALUE(q) FROM table2 q WHERE p.id = q.id)
   WHERE p.id = 10;
```

```
UPDATE TABLE(SELECT projs
           FROM dept d WHERE d.dno = 123)  p
  SET p.budgets = p.budgets + 1
  WHERE p.pno IN (123, 456);
```

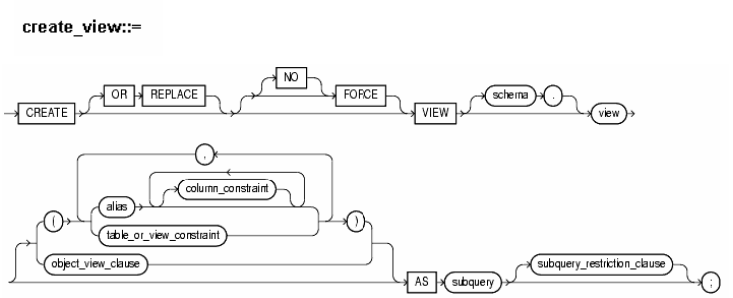
2.3.4 COMMIT ET ROLLBACK

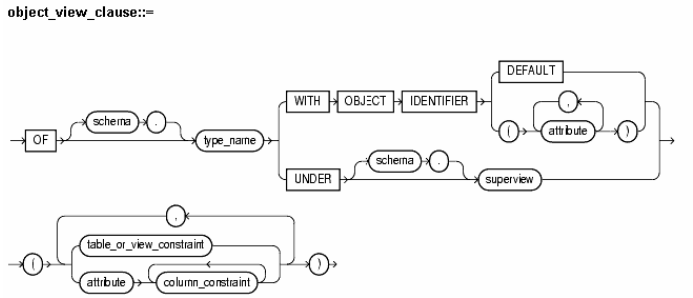
- Ordres qui permettent de valider ou d'invalider une ou une série de transactions
- Si invalidation, possibilité de préciser le point de reprise
- Détails dans le chapitre suivant

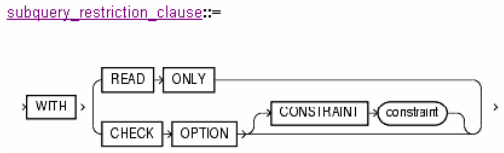
2.4 - Création des vues

- Vue = pseudo-table non matérialisée créée à partir de tables pré-existantes ou d'autres vues
- Utile
  - confidentialité
  - certaines applications
  - renommage (langues)

CREATE VIEW







Exemples

```
CREATE VIEW emp_view AS
   SELECT last_name, salary*12 annual_salary
   FROM employees
   WHERE department_id = 20;

CREATE VIEW emp_sal (emp_id, last_name,
      email UNIQUE RELY DISABLE NOVALIDATE,
   CONSTRAINT id_pk PRIMARY KEY (emp_id) RELY DISABLE NOVALIDATE)
   AS SELECT employee_id, last_name, email FROM employees;

CREATE VIEW customer (name, language, credit)
      AS SELECT cust_last_name, nls_language, credit_limit
      FROM customers
      WITH READ ONLY;
```

Mise à jour par l'intermédiaire d'une vue

```
CREATE VIEW clerk AS
   SELECT employee_id, last_name, department_id, job_id
   FROM employees
   WHERE job_id = 'PU_CLERK'
      or job_id = 'SH_CLERK'
      or job_id = 'ST_CLERK';
```

```
UPDATE clerk SET job_id = 'PU_MAN' WHERE employee_id = 118;
```

```
CREATE VIEW locations_view AS
   SELECT d.department_id, d.department_name, l.location_id, l.city
   FROM departments d, locations l
   WHERE d.location_id = l.location_id;

SELECT column_name, updatable
  FROM user_updatable_columns
  WHERE table_name = 'LOCATIONS_VIEW';

COLUMN_NAME                    UPD
------------------------------ ---
DEPARTMENT_ID                  YES
DEPARTMENT_NAME                YES
LOCATION_ID                    NO
CITY                           NO
```

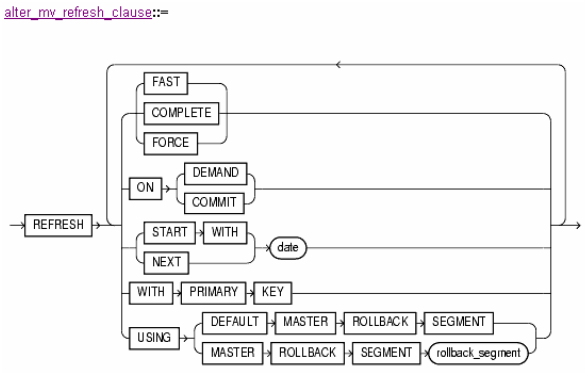
Tentative de mise à jour

```
INSERT INTO locations_view VALUES
   (999, 'Entertainment', 87, 'Roma');

INSERT INTO locations_view VALUES
*
ERROR at line 1:
ORA-01776: cannot modify more than one base table through a join
view
```

Vues matérialisées

- Il est possible de matérialiser le résultat d'une requête (Materialized View /snapshot)
- Master Table -> Detail Table
- Permet des recopies automatiques régulières (REFRESH)
- CREATE MATERIALIZED VIEW …
- DROP MATERIALIZED VIEW …
- ALTER MATERIALIZED VIEW …

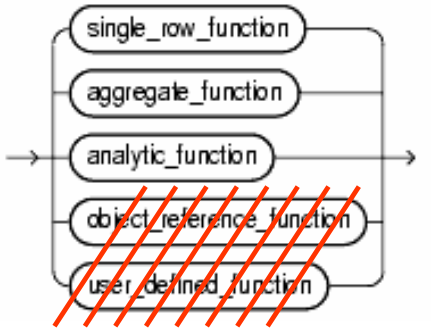
Rafraîchissement régulier



Intérêt des vues

- Confidentialité et protection
- Autres visions du contenu
- Multi-lingue
- Vues sur tables distantes

2. 5 – Fonctions



Single_row function

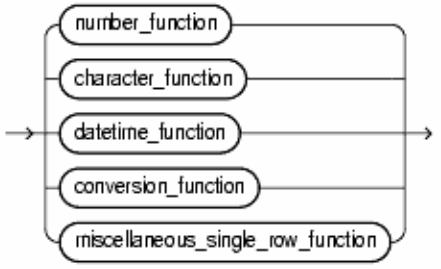
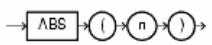


Table 6-1 Number Functions

ABS	EXP	SIN
ACOS	FLOOR	SINH
ASIN	LN	SQRT
ATAN	LOG	TAN
ATAN2	MOD	TANH
BITAND	POWER	TRUNC (number)
CEIL	ROUND (number)	WIDTH_BUCKET
COS	SIGN	
COSH		

Exemple ABS

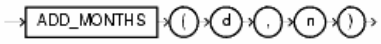
abs::=




```
SELECT ABS(-15) "Absolute" FROM DUAL;

  Absolute
----------
        15
```

ADD_MONTHS

add_months::=




```
SELECT TO_CHAR(
     ADD_MONTHS(hire_date,1),
     'DD-MON-YYYY') "Next month"
     FROM employees
     WHERE last_name = 'Baer';

Next Month
-----------
07-JUL-1994
```

Arc tangente


```
SELECT ATAN(.3) "Arc_Tangent" FROM DUAL;

Arc_Tangent
----------
.291456794
```

Average (1/2)


```
SELECT AVG(salary) "Average" FROM employees;

 Average
--------
    6425
```

Average (2/2)

```
select manager_id, AVG(salary) as c_mavg
from employees
group by manager_id
order by manager_id
```

Table 6-2 Character Functions Returning Character Values

CHR	NLS_LOWER	SUBSTR
CONCAT	NLSSORT	TRANSLATE
INITCAP	NLS_UPPER	TREAT
LOWER	REPLACE	TRIM
LPAD	RPAD	UPPER
LTRIM	RTRIM	
NLS_INITCAP	SOUNDEX	

Table 6-3 Character Functions Returning Number Values

ASCII	INSTR	LENGTH

Table 6-4 Datetime Functions

ADD_MONTHS	MONTHS_BETWEEN	SYSTIMESTAMP
CURRENT_DATE	NEW_TIME	SYSDATE
CURRENT_TIMESTAMP	NEXT_DAY	TO_DSINTERVAL
DBTIMEZONE	NUMTODSINTERVAL	TO_TIMESTAMP
EXTRACT (datetime)	NUMTOYMINTERVAL	TO_TIMESTAMP_TZ
FROM_TZ	ROUND (date)	TO_YMINTERVAL
LAST_DAY	SESSIONTIMEZONE	TRUNC (date)
LOCALTIMESTAMP	SYS_EXTRACT_UTC	TZ_OFFSET

Table 6-5 Conversion Functions

ASCIISTR	RAWTONHEX	TO_NCHAR (character)
BIN_TO_NUM	ROWIDTOCHAR	TO_NCHAR (datetime)
CAST	ROWIDTONCHAR	TO_NCHAR (number)
CHARTOROWID	TO_CHAR (character)	TO_NCLOB
COMPOSE	TO_CHAR (datetime)	TO_NUMBER
CONVERT	TO_CHAR (number)	TO_SINGLE_BYTE
DECOMPOSE	TO_CLOB	TO_YMINTERVAL
HEXTORAW	TO_DATE	TRANSLATE ... USING
NUMTODSINTERVAL	TO_DSINTERVAL	UNISTR
NUMTOYMINTERVAL	TO_LOB	
RAWTOHEX	TO_MULTI_BYTE	

Table 6-6 Miscellaneous Single-Row Functions

BFILENAME	NLS_CHARSET_DECL_LEN	SYS_GUID
COALESCE	NLS_CHARSET_ID	SYS_TYPEID
DECODE	NLS_CHARSET_NAME	SYS_XMLAGG
DUMP	NULLIF	SYS_XMLGEN
EMPTY_BLOB, EMPTY_CLOB	NVL	UID
EXISTSNODE	NVL2	USER
EXTRACT (XML)	SYS_CONNECT_BY_PATH	USERENV
GREATEST	SYS_CONTEXT	VSIZE
LEAST	SYS_DBURIGEN	
	SYS_EXTRACT_UTC	

Fonctions d'agrégation

- Liées au "Group by"
- Possible simplement sur des paquets de lignes
- Ex : moyenne

Table 6-7 Aggregate Functions

AVG	GROUPING_ID	STDDEV
CORR	LAST	STDDEV_POP
COUNT	MAX	STDDEV_SAMP
COVAR_POP	MIN	SUM
COVAR_SAMP	PERCENTILE_CONT	VAR_POP
CUME_DIST	PERCENTILE_DISC	VAR_SAMP
DENSE_RANK	PERCENT_RANK	VARIANCE
FIRST	RANK	
GROUP_ID	REGR_ (linear regression) functions	
GROUPING		

Fonction analytique

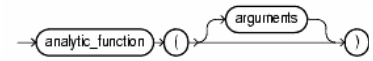


Table 6-8 Analytic Functions

AVG *	LEAD	ROW_NUMBER
CORR *	MAX *	STDDEV *
COVAR_POP *	MIN *	STDDEV_POP *
COVAR_SAMP *	NTILE	STDDEV_SAMP *
COUNT *	PERCENT_RANK	SUM *
CUME_DIST	PERCENTILE_CONT	VAR_POP *
DENSE_RANK	PERCENTILE_DISC	VAR_SAMP *
FIRST	RANK	VARIANCE *
FIRST_VALUE *	RATIO_TO_REPORT	
LAG	REGR_ (linear regression) functions *	
LAST		
LAST_VALUE *		

Object Reference Functions

DEREF	REF	VALUE
MAKE_REF	REFTOHEX	

Conclusion

- Il s'agit de fonctions propres à SQL
- Possibilité de rédiger d'autres fonctions à la demande

2.6 – Conclusion

- Consultations puissantes
- Ordres de manipulation
  - attention `COMMIT` et `Rollback`
- Importance des vues