IUT d'Aix-en-Provence
Département Informatique

Mémento des ordres SQL*PLUS

SQL comme Langage de Définition des Données

Types syntaxiques des attributs : **VARCHAR2(n) CHAR[(n)] NUMBER[(n[, m)]] DATE LONG**

Création de relation

```
CREATE TABLE   <nom_table>
(<nom_colonne1> <type1> [DEFAULT <expression1>]
[,<nom_colonne2> <type2> [DEFAULT <expression2>]
[, <contrainte1>[, <contrainte2>… ]])
```

où :

```
<contraintei> := CONSTRAINT <nom_contrainte> <spec_contrainte> [<etat>]
<spec_contrainte> :=  PRIMARY KEY(<attribut1>[, <attribut2>, …])
                    | FOREIGN KEY(<attribut1>[, <attribut2>, …])
                      REFERENCES <nom_relation_associée>(<att1>[, <att2>, …])
                      [ON DELETE CASCADE | ON DELETE SET NULL]
                    | CHECK(<nom_attribut | expression> <condition>)
<etat> := ENABLE | DISABLE
```

```
CREATE TABLE <nom_relation> [(<liste_attributs>, <liste_contraintes>)]
AS <requete>
```

Ajout d'attributs et de contraintes dans une relation

```
ALTER TABLE <nom_table>
ADD ([<nom_colonne1> <type1>] [DEFAULT <expr1>] [NOT NULL] [UNIQUE]
[,<nom_colonne2> <type2> [DEFAULT <expr2>] [NOT NULL] [UNIQUE]…]
[,<contraintei> …])
```

où :

<contraintei> est identique à la spécification de contraintes lors de la création de relation.

```
<etat> := ENABLE | DISABLE | VALIDATE | NOVALIDATE | ENABLE VALIDATE| ENABLE
NOVALIDATE | DISABLE VALIDATE | DISABLE NOVALIDATE
```

Modification de la définition d'un attribut

```
ALTER TABLE <nom_table>
MODIFY [()]<nom_colonne1> [<nouveau_type1>] [DEFAULT <expr1>] [NOT NULL]
      [,<nom_colonne2>[<nouveau_type2>] [DEFAULT <expr2>] [NOT NULL]…] [()]
```

Modification de l'état d'une contrainte

```
ALTER TABLE <nom_table>
MODIFY CONSTRAINT <nom_contrainte> <etat_contrainte>
```

Suppression de contrainte dans une relation

```
ALTER TABLE <nom_table> DROP CONSTRAINT <nom_contrainte> [CASCADE]

ALTER TABLE <nom_table> DROP UNIQUE(<nom_attribut>) [CASCADE]

ALTER TABLE <nom_table> DROP PRIMARY KEY [CASCADE]
```

Suppression d'attribut dans une relation

```
ALTER TABLE <nom_table> SET UNUSED COLUMN <nom_attribut>

ALTER TABLE <nom_table> SET UNUSED (<nom_attribut1>[, <nom_attribut2> …])

ALTER TABLE <nom_table> DROP COLUMN <nom_attribut> [CASCADE CONSTRAINTS]

ALTER TABLE <nom_table> DROP (<nom_attribut1>[, <nom_attribut2> …])
[CASCADE CONSTRAINTS]

ALTER TABLE <nom_table> DROP UNUSED COLUMNS
```

Suppression de relation

```
DROP TABLE <nom_table> [CASCADE CONSTRAINTS]
```

Création/suppression de synonyme et changement du nom d'une relation

```
CREATE [PUBLIC] SYNONYM <nom_synonyme> FOR <nom_objet>

DROP SYNONYM <nom_synonyme>

RENAME <ancien_nom> TO <nouveau_nom>
```

Gestion de séquences

```
CREATE SEQUENCE <nom_sequence>
    [START WITH  <valeur_initiale>]
    [INCREMENT BY  <valeur_increment>]
    [MAXVALUE  <valeur_maximale> | NOMAXVALUE]
    [MINVALUE  <valeur_minimale> | NOMINVALUE]
    [CYCLE  | NOCYCLE]

DROP SEQUENCE <nom_sequence>
```

```
ALTER SEQUENCE <nom_sequence>
     [INCREMENT BY  <valeur_increment>]
     [MAXVALUE  <valeur_maximale> | NOMAXVALUE]
     [MINVALUE  <valeur_minimale> | NOMINVALUE]
     [CYCLE  | NOCYCLE]
```

Index sur les relations

```
CREATE [UNIQUE | BITMAP] INDEX  <nom_index>
ON <nom_table> (<nom_colonne1>[, <nom_colonne2> …])

ALTER INDEX <nom_index> RENAME TO <nouveau_nom>

DROP INDEX <nom_index>
```

Principales tables systèmes ORACLE

```
ALL_CONS_COLUMNS  (OWNER, CONSTRAINT_NAME, TABLE_NAME, COLUMN_NAME … )
ALL_CONSTRAINTS   (OWNER,CONSTRAINT_NAME,CONSTRAINT_TYPE,TABLE_NAME,
                  SEARCH_CONDITION …)
ALL_INDEXES       (OWNER,  INDEX_NAME,  INDEX_TYPE,  TABLE_OWNER,  TABLE_NAME
                  TABLE_TYPE, UNIQUENESS, COMPRESSION)
ALL_OBJECTS       (OWNER, OBJECT_NAME, OBJECT_ID, DATA_OBJECT_ID, OBJECT_TYPE
                  CREATED …)
ALL_SEQUENCES     (SEQUENCE_OWNER, SEQUENCE_NAME, MIN_VALUE, MAX_VALUE,
                  INCREMENT,  CYCLE_FLAG)
ALL_SYNONYMS      (OWNER, SYNONYM_NAME, TABLE_OWNER, TABLE_NAME …)
ALL_TAB_COLUMNS   (OWNER, TABLE_NAME, COLUMN_NAME, DATA_TYPE, DATA_LENGTH …)
ALL_TABLES        (OWNER, TABLE_NAME, TABLESPACE_NAME …)
ALL_VIEWS         (OWNER, VIEW_NAME, TEXT_LENGTH, TEXT …)
```

Les tables de même nom préfixées par `USER_` ont la même structure hormis l'attribut `OWNER` et décrivent seulement les composants du schéma de l'utilisateur.

Pseudo-colonnes : `<nom_sequence>.CURRVAL`, `<nom_sequence>.NEXTVAL`, **LEVEL**, **ROWID**, **ROWNUM**, **USER**.

SQL comme Langage de Manipulation des Données

```
<requete> :=     SELECT  <liste_resultat | * >
                 FROM    <liste_relations>
                 [WHERE  <liste_conditions>]
                 [GROUP  BY <liste_attributs_de_partitionnement>
                 [HAVING <liste_conditions_de_partitionnement>]]
                 [ORDER  BY <liste_attributs_a_trier>]
```

où :

```
liste_resultat  :=  [DISTINCT] <attribut1  |  expr1|  requete1>  [<alias1>]  [,
      <attribut2 | expr2 | requete2> [<alias2> …]
```

```
<liste_relations> := <relation1 | vue1 | requete1> [alias1]
      [, <relation2 | vue2 | requete2> [alias2] … ]
<liste_conditions> := [NOT] <condition1> [AND | OR <condition2> …]
```

condition de sélection :

```
<conditioni>:= <attribut [(+)] | expression> <comparateur | predicat_cond>
      <constante>
<predicat_cond> := IS NULL | IN | BETWEEN … AND | LIKE | IS NOT NULL | NOT IN
      | NOT BETWEEN | NOT LIKE
```

condition de jointure prédicative :

```
<conditionj>:= <attribut1[(+)]|expr1> <comparateur> <attribut2[(+)]|expr2>
```

condition de jointure imbriquée :

```
<conditionji>:= <expression1>[, <expression2>, <expression3>…] θ (<requete>)
      | <expression1>[, <expr2>, <expr3>…]  θ ANY | IN (<requete>)
      | <expression1>[, <expression2>, <expression3>…]  θ ALL (<requete>)
```

Calculs verticaux (fonctions agrégatives)

```
      <nom_fonction> ([DISTINCT]<nom_colonne>)
```

où :

```
<nom_fonction> := SUM | AVG | COUNT | MAX | MIN | STDDEV | VARIANCE
```

Tri des résultats

```
      ORDER BY <expression1> [ASC | DESC] [NULLS FIRST | NULLS LAST]
      [, <expression2> [ASC | DESC] [NULLS FIRST | NULLS LAST]…]
```

Jointure algébrique

```
      SELECT <nom_colonne1> [, <nom_colonne2>…]
      FROM  <nom_table1>
      [INNER] JOIN <nom_table2> …
      ON    < nom_colonne1> θ <nom_colonne2> [AND <condition>…]
      [[INNER] JOIN <nom_table3> …]
      [WHERE <condition>]
```

Jointures externes

```
      SELECT <nom_colonne1> [, <nom_colonne2>…]
      FROM  <nom_table1>, <nom_table2> …
      WHERE <nom_colonne1> θ < nom_colonne2> (+)
      AND <nom_colonne3> (+) θ <constante | expression> ..

      SELECT <nom_colonne1> [, <nom_colonne2>…]
      FROM  <nom_table1>
      LEFT[1] [OUTER JOIN] <nom_table2> …
      ON    < nom_colonne1> θ <nom_colonne2> [AND <condition>…]
      [LEFT [OUTER] JOIN <nom_table3> …]
      [WHERE <condition>]
```

1) RIGHT ou FULL

Séquence

CREATE SEQUENCE <nom_sequence> **INCREMENT BY** <entier1> **START WITH** <entier2>

Deux pseudo-colonnes :

- **nom_seq.CURVAL** : valeur courante de la séquence.
- **nom_sequence.NEXTVAL** : valeur suivante de la séquence.

Opérateurs ensemblistes

```
<requete1>
UNION | INTERSECT | MINUS
<requete2>
```

Test d'absence ou d'existence de données

```
SELECT      <liste_attributs>
FROM        <relation1> [<alias1>][, <relation2> [<alias2] …]
WHERE       [<liste_conditions> AND | OR] [NOT] EXISTS
            (<sous_requete>)
```

Classification ou partitionnement

```
GROUP BY <colonne1> [, <colonne2>,…]
HAVING <liste_condition_classe>
```

Recherche dans une arborescence

```
SELECT <colonne1> [, <colonne2> …]
FROM  <table> [<alias>]
[WHERE                  <liste_conditions>]
CONNECT BY [PRIOR]      <colonne1> = [PRIOR] <colonne2>
[AND                    <condition_hierarchique>]
[START WITH             <condition_depart>]
[ORDER  BY  LEVEL]
```

Mises à jour des données

```
UPDATE <nom_table>
SET    <nom_colonne1> = <expression1>[, <nom_colonne2> = <expr2> …]
[WHERE <condition_selection>]

UPDATE <nom_table>
SET          (<nom_colonne1>[, <nom_colonne2>, nom_colonne3> …]) =
             (SELECT <col1>[, <col2>, col3> …] FROM … WHERE …)
WHERE  <condition_selection>


INSERT INTO <nom_table> [(<liste_attributs>)]
VALUES (<valeur1>[, <valeur2> …])


INSERT INTO <nom_table> [(<liste_attributs>,…)] <requête>


DELETE FROM <nom_table> WHERE <condition>
```

SQL comme Langage de Contrôle des Données

Gestion des transactions *:* **COMMIT** , **ROLLBACK**.

Création et suppression de rôles et d'utilisateurs

```
CREATE ROLE <nom_role> [IDENTIFIED BY <mot_de_passe>]

ALTER ROLE <nom_role> [IDENTIFIED BY <nouveau_mot_de_passe>]

DROP ROLE <nom_role>

CREATE USER <nom_utilisateur> [IDENTIFIED BY <mot_de_passe>]
DEFAULT TABLESPACE <nom_table_space>
QUOTA <taille> PROFILE <nom_profil>

ALTER USER <nom_utilisateur> [IDENTIFIED BY mot_de_passe>]

DROP USER <nom_utilisateur>
```

Attribution et suppression de privilèges

```
GRANT       <systeme_privileges | ALL [PRIVILEGES]>
TO          <liste_role_utilisateur | PUBLIC>
[WITH ADMIN OPTION]
```

où :

<systeme_privileges> **CREATE ROLE** | **CREATE SEQUENCE** | **CREATE SESSION** |**CREATE SYNONYM** | **CREATE PUBLIC** | **CREATE TABLE** | **CREATE USER** |**CREATE VIEW**

```
GRANT       <liste_droits>
ON          <nom_composant>
TO          <liste_roles_utilisateurs>
[WITH GRANT OPTION]
```

<liste_droits> := **SELECT** | **INSERT** | **UPDATE** [<nom_colonne1, <nom_colonne2>…] | **DELETE** | **ALTER** | **INDEX** | **REFERENCES** | **ALL** [**PRIVILEGES**]

```
GRANT   <liste_roles_attribues>
TO      <liste_roles_utilisateurs>
[WITH ADMIN OPTION]
```

Gestion de vues

```
CREATE [OR REPLACE] [FORCE | NO FORCE]
VIEW <nom_vue> [(alias1, alias2…)]
AS
<requete >
[WITH CHECK OPTION | WITH READ ONLY]

ALTER VIEW <nom_vue> COMPILE

DROP VIEW <nom_vue>
```