LMD SQL PostgreSQL

```
SELECT [ALL | DISTINCT] < projection>
FROM <objets>
[WHERE <clause de restriction> ]
[GROUP BY <clause de groupage> [HAVING <condition>]]
[ORDER BY <clause d'ordonnancement>] ;
```

PRODUIT CARTESIEN : utilisation dans la clause FROM

- <nom_table1> [alias] CROSS JOIN <nom_table2> [alias]

JOINTURE: utilisation dans la clause FROM de l'opérateur JOIN avec l'une des syntaxes suivantes :

- <nom_table1> [alias] INNER JOIN <nom_table2> [alias] ON <condition> ou
- <nom_table1> [alias] {LEFT | RIGHT | FULL} OUTER JOIN <nom_table2> [alias] ON <condition>
- <nom_table1> [alias] NATURAL JOIN <nom_table2> [alias]

OPERATEURS ENSEMBLISTES : UNION | INTERSECT | EXCEPT entre deux requêtes SELECT

INSERT :
INSERT INTO <nom_table> [(<liste de nom_colonne>)] VALUES(<liste de valeurs>) ;
INSERT INTO <nom_table> [(<liste de nom_colonne>)] requête ;

DELETE : destruction de n-uplets dans **une** table
DELETE [FROM] <nom_table> : efface tous les n-uplets de la table.
DELETE [FROM] <nom_table> WHERE <condition> : efface tous les n-uplets sélectionnés par la condition.

UPDATE : permet de modifier la valeur de colonnes de n-uplets d'**une** table.
UPDATE <table> SET nom_colonne = <expression> | (requête) [WHERE <condition>] ;

WITH [RECURSIVE] *requête_with* [, ...] SELECT ...
La clause WITH vous permet de spécifier une ou plusieurs sous-requêtes qui peuvent être utilisées par leur nom dans la requête principale. Les sous-requêtes se comportent comme des tables temporaires ou des vues pendant la durée d'exécution de la requête principale. Toutes les requêtes dans la liste WITH sont évaluées. Elles jouent le rôle de tables temporaires qui peuvent être référencées dans la liste FROM de la requête principale.

LDD LDC SQL PostgreSQL

Création d'un schéma de table
CREATE TABLE <nom_table> (<déclaration colonne>[, {<déclaration colonne> | <contrainte table>}]^n) ;

<déclaration colonne> : <nom-colonne> <type> [<contrainte colonne> [<contrainte colonne>]^n]
Principaux types de données
<type> : CHAR(n) | DATE | NUMERIC[(n[,p])] | VARCHAR(n)
<contrainte colonne> : CONSTRAINT <nom_contrainte>
{PRIMARY KEY | [NOT] NULL |
UNIQUE | REFERENCES <nom_table> [(<nom_colonne>)] |
CHECK (<condition>)}
<contrainte table> : CONSTRAINT <nom_contrainte>
{ PRIMARY KEY(<nom_colonne>[,<nom_colonne>]^n)
| [NOT] NULL |
UNIQUE (<nom_colonne>[,<nom_colonne>]^n) |
FOREIGN KEY (<nom_colonne>[,<nom_colonne>]^n)
REFERENCES <nom_table> [(<nom_colonne>[,<nom_colonne>]^n)] |
CHECK (<condition>) }

Création avec peuplement: CREATE est suivi d'une requête dont le résultat est chargé dans la table déclarée CREATE TABLE <nom_table> (…) AS (requête).

Destruction d'un schéma : DROP TABLE <nom_table> (détruit le contenu éventuel **et** le schéma de la table).

Modification d'un schéma :
ALTER TABLE <nom_table> ADD (déclaration de colonne ou de contrainte) ;
ALTER TABLE <nom_table> DROP (colonne ou contrainte)
ALTER TABLE <nom_table> ALTER COLUMN (nom_de colonne , avec nouveau type éventuel et/ou une nouvelle Contrainte).