Langages de requêtes

M1 Informatique
Année 2010-2011
Jérôme Darmont

http://eric.univ-lyon2.fr/~jdarmont/

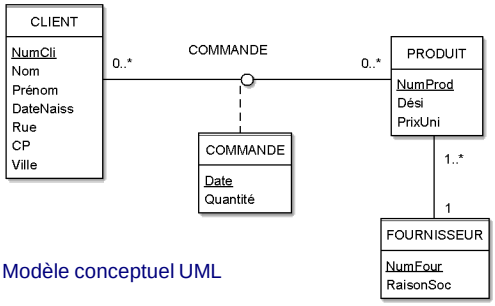
Plan du cours

- Introduction
- Algèbre relationnelle
- Langage SQL
- Langage XQuery

Objectifs du cours

- Logique de l'interrogation de bases de données relationnelles
  - ➲ Algèbre relationnelle
- Mise en œuvre (*création*, *mise à jour* et *interrogation*) de BD relationnelles
  - ➲ Langage SQL
- Interrogation de bases de données XML
  - ➲ Langage XQuery

Base de données exemple



- Modèle conceptuel UML

Base de données exemple

- Modèle logique relationnel

CLIENT (NumCli, Nom, Prénom, DateNaiss, Rue, CP, Ville)
PRODUIT (NumProd, Désí, PrixUni, NumFour#)
FOURNISSEUR (NumFour, RaisonSoc)

COMMANDE (NumCli#, NumProd#, Date, Quantité)

Clés primaires Clés étrangères#

Plan du cours

- ✓ Introduction
- ✓ Algèbre relationnelle
- Langage SQL
- Langage XQuery

Présentation

- SQL : *Structured Query Language,* issu de SEQUEL (*Structured English as a QUery Language*)
- SQL permet la définition, la manipulation et le contrôle d'une base de données relationnelle. Il se base sur l'algèbre relationnelle.
- SQL est un standard ANSI depuis 1986.
- Nous adoptons dans ce chapitre la syntaxe du SQL d'Oracle (très proche de la norme).

Présentation

- SQL se subdivise en trois sous-langages :
  - LDD (*Langage de Définition de Données*) : création, modification et suppression des définitions des tables
  - LMD (*Langage de Manipulation de Données*) : ajout, suppression, modification et interrogation des données
  - LCD (*Langage de Contrôle de Données*) : gestion des protections d'accès

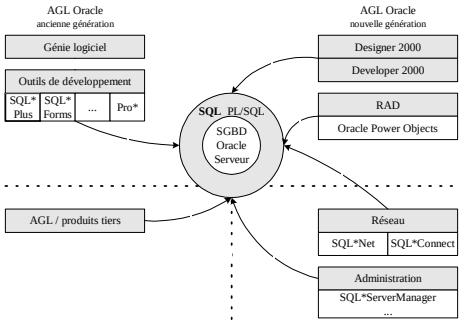
Qu'est-ce qu'Oracle ?

- Oracle : Système de Gestion de Bases de Données (SGBD) relationnel (SGBDR) édité par *Oracle Corporation* (http://www.oracle.com)
- Oracle est basé sur SQL.
- Première version d'Oracle : 1981
  Version actuelle : Oracle 11*g*

Fonctionnalités d'Oracle

- Aide à la décision (*data warehouses*, Oracle Express, Oracle Discoverer)
- Gestion de grands volumes de données (partitionnement des données)
- Mécanismes de sécurité
- Sauvegarde et récupération des données
- Gestion souple de l'espace disque
- Connectivité ouverte sur différentes plateformes

Architecture d'Oracle



Généralités

- Caractère de fin d'instruction : **;**
- Commentaires :
  - -- Ligne commentée
  - /* Bloc de texte commenté */
- Clauses optionnelles : Notées entre [ ] dans ce support de cours

Tables

- CREATE TABLE nom_table (Attribut1 TYPE,
  Attribut2 TYPE, …,
  contrainte_integrité1,
  contrainte_integrité2,
  …);
- Principaux type de données :
  - NUMBER(n) : Entier à n chiffres
  - NUMBER(n, m) : Réel à n chiffres au total (virgule comprise), m après la virgule
  - VARCHAR(n) : Chaîne de n caractères (entre ‘ ’)
  - DATE : Date au format ‘JJ-MM-AAAA’
  - BLOB : *Binary Large Object*

Contraintes d’intégrité

- Clé primaire :
  CONSTRAINT nom_contrainte PRIMARY KEY (attribut_clé [, attribut_clé2, …])
- Clé étrangère :
  CONSTRAINT nom_contrainte FOREIGN KEY (attribut_clé_ét) REFERENCES table(attribut)
- Contrainte de domaine :
  CONSTRAINT nom_contrainte CHECK (condition)

Exemple

```
CREATE TABLE Produit (      NumProd NUMBER(3),
                            Dési VARCHAR(30),
                            PrixUni NUMBER(8,2),
                            NumFour NUMBER(3),

CONSTRAINT produit_cle_pri PRIMARY KEY (NumProd),

CONSTRAINT produit_cle_etr FOREIGN KEY (NumFour)
        REFERENCES Fournisseur(NumFour),

CONSTRAINT prix_ok CHECK (PrixUni > 0)      );
```


Modifications structurelles

- Ajout d’attributs
  ALTER TABLE nom_table ADD (attribut TYPE, …);
  ex. ALTER TABLE Client ADD (tel NUMBER(8));
- Modifications d’attributs
  ALTER TABLE nom_table MODIFY (attribut TYPE, …);
  ex. ALTER TABLE Client MODIFY (tel NUMBER(10));
- Suppression d'attributs
  ALTER TABLE nom_table DROP COLUMN attribut, ...;
  ex. ALTER TABLE Client DROP COLUMN tel;

Modifications structurelles

- Ajout de contrainte
  ALTER TABLE nom_table ADD CONSTRAINT nom_contrainte définition_contrainte;
  ex. ALTER TABLE Client
  ADD CONSTRAINT sal_ok CHECK (salaire > 0);
- Suppression de contrainte
  ALTER TABLE nom_table DROP CONSTRAINT nom_contrainte;
  ex. ALTER TABLE Client
  DROP CONSTRAINT sal_ok;

Copie et destruction de tables

- Destruction
  DROP TABLE nom_table;
  ex. DROP TABLE Client;
- Copie
  CREATE TABLE copie AS requête;
  ex. CREATE TABLE Client_Copie AS
  SELECT * FROM Client;

Index

- Création d'index (accélération des accès)

  CREATE [UNIQUE] INDEX nom_index ON nom_table (attribut [ASC|DESC], …);

  UNIQUE ⇒ pas de double
  ASC/DESC ⇒ ordre croissant ou décroissant

  ex. CREATE INDEX Idx_cli ON Client (Nom);

- Destruction

  DROP INDEX nom_index;

  ex. DROP INDEX Idx_cli;

Insertion et mise à jour

- Ajout d'un n-uplet

  INSERT INTO nom_table
  VALUES (val_att1, val_att2, …);

  ex. INSERT INTO Produit
  VALUES (400, 'Nouveau produit', 78.90, 30);

- Mise à jour d'un attribut

  UPDATE nom_table SET attribut = valeur
  [WHERE condition];

  ex. UPDATE Client SET Nom = 'Dudule'
  WHERE NumCli = 3;

Suppression

- Suppression de n-uplets

  DELETE FROM nom_table [WHERE condition];

  ex. DELETE FROM Produit;

  DELETE FROM Client
  WHERE Ville = 'Lyon';

Requêtes simples

- Forme générale d'une requête

  SELECT [ALL|DISTINCT] attribut(s) FROM table(s)
  [WHERE condition]
  [GROUP BY attribut(s) [HAVING condition]]
  [ORDER BY attribut(s) [ASC|DESC]];

- Tous les n-uplets d'une table
  ex. SELECT * FROM Client;
- Tri du résultat
  ex. *Par ordre alphabétique inverse de nom*
  SELECT * FROM Client
  ORDER BY Nom DESC;

Requêtes simples

- Calcul
  ex. *Calcul de prix TTC*
  SELECT PrixUni + PrixUni * 0.196 FROM Produit;
- Projection
  ex. *Noms et Prénoms des clients, uniquement*
  SELECT Nom, Prenom FROM Client;
- Suppression des doublons
  ex. SELECT DISTINCT Nom FROM Client;
- Restriction
  ex. *Clients qui habitent à Lyon*
  SELECT * FROM Client
  WHERE Ville = 'Lyon';

Requêtes simples

ex. *Commandes en quantité au moins égale à 3*
SELECT * FROM Commande
WHERE Quantite >= 3;

ex. *Produits dont le prix est compris entre 50 et 100 €*
SELECT * FROM Produit
WHERE PrixUni BETWEEN 50 AND 100;

ex. *Commandes en quantité indéterminée*
SELECT * FROM Commande
WHERE Quantite IS NULL;

Requêtes simples

ex. *Clients habitant une ville dont le nom se termine par* « sur-Saône »

```
SELECT * FROM Client
WHERE Ville LIKE '%sur-Saône';
```

'sur-Saône%' ➲ commence par « *sur-Saône* »
'%sur%' ➲ contient le mot « *sur* »

Prédicat ensembliste IN

ex. *Prénoms des clients dont le nom est Dupont, Durand ou Martin*

```
SELECT Prénom FROM Client
WHERE Nom IN ('Dupont', 'Durand', 'Martin');
```

NB : Possibilité d'utiliser la négation pour tous ces prédicats : NOT BETWEEN, NOT NULL, NOT LIKE, NOT IN.

Fonctions d'agrégat

- Elles opèrent sur un ensemble de valeurs.
- AVG(), VARIANCE(), STDDEV() : moyenne, variance et écart-type des valeurs
- SUM() : somme des valeurs
- MIN(), MAX() : valeur minimum, valeur maximum
- COUNT() : nombre de valeurs
- ex. *Moyenne des prix des produits*
  SELECT AVG(PrixUni) FROM Produit;

Fonctions d'agrégat

Fonction COUNT et opérateur DISTINCT

ex. *Nombre total de commandes*

```
SELECT COUNT(*) FROM Commande;
SELECT COUNT(NumCli) FROM Commande;
```

ex. *Nombre de clients ayant passé commande*

```
SELECT COUNT(DISTINCT NumCli)
FROM Commande;
```


Fonctions d'agrégat

Table COMMANDE (simplifiée)

NumCli	*Date*	*Quantite*
1	22/09/99	1
3	22/09/99	5
3	22/09/99	2

COUNT(NumCli) ➲ Résultat = 3

COUNT(DISTINCT NumCli) ➲ Résultat = 2

Jointures

ex. *Liste des commandes avec le nom des clients*

```
SELECT Nom, Date, Quantite
FROM Client, Commande
WHERE       Client.NumCli =
            Commande.NumCli;
```

Jointures

ex. *Idem avec le numéro de client en plus*

```
SELECT C1.NumCli, Nom, Date, Quantite
FROM Client C1, Commande C2
WHERE C1.NumCli = C2.NumCli
ORDER BY Nom;
```

NB : Utilisation d'alias (C1 et C2) pour alléger l'écriture + tri par nom.

Jointures

Jointure exprimée avec le prédicat IN

ex. *Nom des clients qui ont commandé le 23/09*

```
SELECT Nom FROM Client
WHERE NumCli IN (
      SELECT NumCli FROM Commande
      WHERE Date = '23-09-1999'     );
```

NB : Il est possible d'imbriquer des requêtes.

Prédicats d'existence

- Prédicats EXISTS / NOT EXISTS

ex. *Clients qui ont passé au moins une commande [n'ont passé aucune commande]*

```
SELECT * FROM Client C1
WHERE [NOT] EXISTS (
      SELECT * FROM Commande C2
      WHERE C1.NumCli = C2.NumCli          );
```


Prédicats de dénombrement

- Prédicats ALL / ANY

ex. *Numéros des clients qui ont commandé au moins un produit en quantité supérieure à chacune [à au moins une] des quantités commandées par le client n°1.*

```
SELECT DISTINCT NumCli FROM Commande
WHERE Quantite > ALL [ANY] (
      SELECT Quantite FROM Commande
      WHERE NumCli = 1      );
```


Groupement

ex. *Quantité totale commandée par chaque client*

```
SELECT NumCli, SUM(Quantite)
FROM Commande
GROUP BY NumCli;
```

ex. *Nombre de produits différents commandés...*

```
SELECT NumCli, COUNT(DISTINCT NumProd)
FROM Commande
GROUP BY NumCli;
```


Groupement

ex. *Quantité moyenne commandée pour les produits faisant l'objet de plus de 3 commandes*

```
SELECT NumProd, AVG(Quantite)
FROM Commande
GROUP BY NumProd
HAVING COUNT(*) > 3;
```

Attention : La clause HAVING ne s'utilise qu'avec GROUP BY.

Groupement

- Différence entre HAVING et WHERE
  - HAVING : évaluation de condition sur un résultat de groupement (*a posteriori*)
  - WHERE : évaluation de condition *a priori* (avant GROUP BY)

Division

- Ex. Numéros des clients qui ont commandé tous les produits.
- Problème : Il n'existe pas d'opérateur de division en SQL !
- Deux stratégies :
  - Clients tels qu'il n'existe pas de produit tel qu'il n'existe pas de commande pour ce client et ce produit.
  - Clients qui ont commandé un nombre distinct de produits égal au nombre total de produits.

Division : solution « logique »

```
SELECT NumCli FROM Client Cl
WHERE NOT EXISTS (
      SELECT NumProd FROM Produit P
      WHERE NOT EXISTS (
                  SELECT * FROM Commande Co
                  WHERE Cl.NumCli = Co.NumCli
                  AND P.NumProd = Co.NumProd )
      );
```


Division : solution par comptage

- ```
  SELECT NumCli FROM Client Cl
  WHERE (     SELECT COUNT(DISTINCT NumProd)
              FROM Commande Co
              WHERE Co.NumCli = Cl.NumCli )
  = ( SELECT COUNT(*) FROM Produit );
  ```

ou

- ```
  SELECT NumCli FROM Commande
  GROUP BY NumCli
  HAVING COUNT(DISTINCT NumProd) =
        ( SELECT COUNT(*) FROM Produit );
  ```


Opérations ensemblistes

Opérateurs ensemblistes :

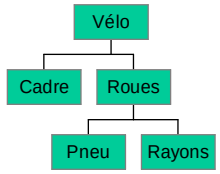
INTERSECT, MINUS, UNION

ex. *Numéro des produits qui soit ont un prix inférieur à 100 €, soit ont été commandés par le client n°2*

```
SELECT NumProd FROM Produit WHERE PrixUni < 100
UNION
SELECT NumProd FROM Commande WHERE NumCli = 2;
```


Requêtes hiérarchiques

Exemple de hiérarchie (nomenclature) :



Relation associée :

ELEMENT (No_Elt, Dési, Parent#)

0	Vélo	NULL
1	Cadre	0
2	Roue1	0
3	Roue2	0
4	Pneu1	2
5	Pneu2	3
6	Rayon11	2
7	Rayon12	2
8	Rayon13	2
9	Rayon21	3

Requêtes hiérarchiques

ex. *Structure hiérarchique des éléments à partir de la racine*

```
SELECT Dési FROM Element
CONNECT BY Parent = PRIOR No_Elt
START WITH Parent IS NULL;
```

Racine de la hiérarchie

Ordre de parcours de la hiérachie

Requêtes hiérarchiques

ex. *Idem avec indication du niveau dans la hiérarchie*

```
SELECT LEVEL, Dési FROM Element
CONNECT BY Parent = PRIOR No_Elt
START WITH Parent IS NULL;
```

Résultat :

1 Velo	3 Rayon12
2 Cadre	3 Rayon13
2 Roue1	2 Roue2
3 Pneu1	3 Pneu2
3 Rayon11	3 Rayon21

Requêtes hiérarchiques

ex. *Idem avec élagage d'une branche de la hiérarchie*

```
SELECT LEVEL, Dési FROM Element
CONNECT BY Parent = PRIOR No_Elt
           AND Dési <> 'Roue2'
START WITH Parent IS NULL;
```

Résultat :

1 Velo	3 Rayon12
2 Cadre	3 Rayon13
2 Roue1	2 Roue2
3 Pneu1	
3 Rayon11	

Requêtes hiérarchiques

ex. *Nombre d'éléments dans chaque niveau*

Il est possible d'utiliser le groupement.

```
SELECT LEVEL, COUNT(No_Elt)
FROM Element
CONNECT BY Parent = PRIOR No_Elt
START WITH Parent IS NULL
GROUP BY LEVEL;
```


Fonctions SQL

- ABS(n) : Valeur absolue de n
- CEIL(n) : Plus petit entier ≥ n
- FLOOR(n) : Plus grand entier ≤ n
- MOD(m, n) : Reste de m/n
- POWER(m, n) : m^n
- SIGN(n) : Signe de n
- SQRT(n) : Racine carrée de n
- ROUND(n, m) : Arrondi à 10^{-m}
- TRUNC(n, m) : Troncature à 10^{-m}
- CHR(n) : Caractère ASCII n°n
- INITCAP(ch) : 1ère lettre en maj.
- LOWER(ch) : c en minuscules
- UPPER(ch) : c en majuscules
- LTRIM(ch, n) : Troncature à gauche
- RTRIM(ch, n) : Troncature à droite
- REPLACE(ch, car) : Remplacement de caractère
- SUBSTR(ch, pos, lg) : Extraction de chaîne
- SOUNDEX(ch) : Cp. Phonétique
- LPAD(ch, lg, car) : Compléter à gauche
- RPAD(ch, lg, car) : Compléter à droite

Fonctions SQL

- ASCII(ch) : Valeur ASCII de ch
- INSTR(ch, ssch) : Recherche de ssch dans ch
- LENGTH(ch) : Longueur de ch
- ADD_MONTHS(dte, n) : Ajout de n mois à dte
- LAST_DAY(dte) : Dernier jour du mois
- MONTHS_BETWEEN(dt1, dt2) : Nombre de mois entre dt1 et dt2
- NEXT_DAY(dte) : Date du lendemain
- SYSDATE : Date/heure système
- TO_NUMBER(ch) : Conversion de ch en nombre
- TO_CHAR(x) : Conversion de x en chaîne
- TO_DATE(ch) : Conversion de ch en date
- NVL(x, val) : Remplace par val si x a la valeur NULL
- GREATEST(n1, n2…) : + grand
- LEAST (n1, n2…) : + petit
- UID : Identifiant numérique de l'utilisateur
- USER : Nom de l'utilisateur

Exemples

- SELECT UID, USER FROM DUAL;
- SELECT GREATEST(1, 2, 3) FROM DUAL;
- SELECT Nom, Prenom,
  FLOOR(MONTHS_BETWEEN(SYSDATE, DateNaiss) / 12) Age
  FROM Client;
- UPDATE Produit SET PrixUni = NVL(PrixUni, 15);

Vues

- Définition : Une vue est une table *virtuelle* calculée à partir d'autres tables grâce à une requête.
- Création d'une vue
  CREATE VIEW nom_vue AS requête;
  ex. CREATE VIEW Noms AS
  SELECT Nom, Prenom FROM Client;

Intérêt des vues

- Simplification de l'accès aux données en masquant les opérations de jointure

  ex. CREATE VIEW Prod_com AS
  SELECT P.NumProd, Dési, PrixUni, Date, Quantite
  FROM Produit P, Commande C
  WHERE P.NumProd = C.NumProd;

  SELECT NumProd, Dési FROM Prod_com
  WHERE Quantite > 10;

Intérêt des vues

- Sauvegarde indirecte de requêtes complexes
- Présentation de mêmes données sous différentes formes adaptées aux différents usagers particuliers
- Support de l'indépendance logique
  ex. Si la table Produit est remaniée, la vue Prod_com doit être refaite, mais les requêtes qui utilisent cette vue n'ont pas à être remaniées.
- Renforcement de la sécurité des données par masquage des lignes et des colonnes sensibles aux usagers non habilités

Restriction des vues (mise à jour)

- Pour que la mise à jour de données à travers une vue soit possible :
  - Le mot clé DISTINCT doit être absent de la requête.
  - La clause FROM doit faire référence à une seule table.
  - La clause SELECT doit faire référence directement aux attributs de la table concernée (pas d'attribut dérivé).
  - Les clauses GROUP BY et HAVING sont interdites.

Catalogue du système

- Définition : Ensemble de vues maintenues automatiquement par le système et contenant sous forme relationnelle la définition de tous les objets créés par le système et les usagers.
- Ces vues sont accessibles avec SQL (en mode consultation uniquement).

Vues systèmes

- ALL_TABLES (OWNER, TABLE_NAME, …)
- ALL_VIEWS (OWNER, VIEW_NAME, …)
- ALL_CONSTRAINTS (OWNER, TABLE_NAME, CONSTRAINT_NAME, CONSTRAINT_TYPE, SEARCH_CONDITION, ...)
- ALL_CONS_COLUMNS (OWNER, TABLE_NAME, CONSTRAINT_NAME, COLUMN_NAME, ...)
- USER_CATALOG (TABLE_NAME, TABLE_TYPE)
- USER_TAB_COLUMNS (TABLE_NAME, COLUMN_NAME, …)
- USER_IND_COLUMNS (INDEX_NAME, TABLE_NAME, COLUMN_NAME, …)
- USER_CONSTRAINTS (TABLE_NAME, CONSTRAINT_NAME, CONSTRAINT_TYPE, SEARCH_CONDITION, ...)

Exemples

- Tables qui contiennent un attribut *Intitulé*

```
SELECT TABLE_NAME FROM USER_TAB_COLUMNS
WHERE COLUMN_NAME = 'INTITULE';
```

- Attributs de la table *Client*

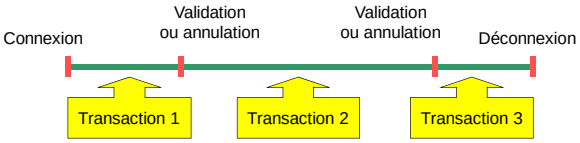
```
SELECT COLUMN_NAME FROM USER_TAB_COLUMNS
WHERE TABLE_NAME = 'CLIENT';
```

- Contraintes des tables de l'utilisateur courant

```
SELECT TABLE_NAME, CONSTRAINT_NAME
FROM ALL_CONSTRAINTS
WHERE OWNER = USER;
```


Gestion des transactions

- Transaction : ensemble de mises à jour des données (≠ modifications structurelles)
- Début de transaction : début de la session de travail ou fin de la transaction précédente



Contrôle des transactions

- Validation (et fin) d'une transaction :

```
COMMIT;
```

- Annulation (et fin) d'une transaction :

```
ROLLBACK;
```

- Fin de session de travail (avec la commande EXIT ou QUIT) ➲ validation automatique

Sécurité : utilisateurs

- Création
  - Ex. CREATE USER moi_meme IDENTIFIED BY mon_mot_de_passe;
- Suppression
  - Ex. DROP USER moi_meme CASCADE;
- Modification
  - Ex. ALTER USER moi_meme IDENTIFIED BY a;

Sécurité : privilèges globaux

- Droit d'effectuer une action sur les objets de l'utilisateur seulement
  - Ex. CREATE TABLE
    ALTER INDEX
    DROP VIEW
- Droit d'effectuer une action dans tous les schémas de la base de données
  - Ex. CREATE ANY TABLE
    ALTER ANY INDEX
    DROP ANY VIEW

Sécurité : privilèges sur les objets

Privilège	Signification	Tables	Vues
ALTER	Destruction	X	
DELETE	Suppression	X	X
INDEX	Construction	X	
INSERT	Insertion	X	X
REFERENCES	Clé étrangère	X	
SELECT	Lecture	X	X
UPDATE	Mise à jour	X	X
ALL	Tous	X	X

Sécurité : rôles

- Rôles prédéfinis
  - CONNECT : droit de création de tables, vues, synonymes, etc.
  - RESOURCE : droit de création de procédures stockées, déclencheurs, etc.
  - DBA : administrateur de la BD
- Création de nouveaux rôles
  - Ex. CREATE ROLE role1;

Sécurité : attribution de privilèges

- Transmission de privilèges

  GRANT privilège ON table|vue
  TO user|PUBLIC [WITH GRANT OPTION];
- Privilèges sur des objets
  - Ex. GRANT SELECT ON ma_table TO toto;
  - Ex. GRANT SELECT ON ma_table TO PUBLIC;
  - Ex. GRANT SELECT ON ma_table TO role1;
- Privilèges globaux et rôles
  - Ex. GRANT CREATE ANY TABLE TO toto;
  - Ex. GRANT CONNECT, RESOURCE TO toto;
  - Ex. GRANT role1 TO toto;

Sécurité : révocation de privilèges

- Suppression de privilèges

  REVOKE privilège ON table|vue FROM user|PUBLIC;
- Privilèges sur des objets
  - Ex. REVOKE SELECT ON ma_table FROM toto;
  - Ex. REVOKE SELECT ON ma_table FROM PUBLIC;
  - Ex. REVOKE SELECT ON ma_table FROM role1;
- Privilèges globaux et rôles
  - Ex. REVOKE CREATE ANY TABLE FROM toto;
  - Ex. REVOKE CONNECT, RESOURCE FROM toto;
  - Ex. REVOKE role1 FROM toto;

Tutoriel SQL

Pour approfondir SQL en ligne…

http://eric.univ-lyon2.fr/~jdarmont/tutoriel-sql/