Génie Logiciel

UE1

Bases de données

TRAORE Aboudou

Ingénieur de conception des techniques informatiques

Développement et intégration d'applications

Consultation et formation en informatique

traobou12@yahoo.fr

Ouagadougou, Octobre 2015

LEÇON IV

Le langage SQL

Sommaire

Présentation de la leçon

Dans cette leçon, nous nous limiterons seulement à l'étude du langage SQL (SQL2/SQL3). Nous parlerons du langage de définition de données, de manipulation et contrôle de donnée. Nous utiliserons Oracle comme système de gestion de bases de données relationnelles (SGBDR) et l'utilitaire SQL-PLUS fourni par Oracle pour taper et exécuter nos commandes. Cependant il n'y a pas d'imposition de SGBDR ni d'utilitaire.

1. Objectif général

L'objectif général est de comprendre et de pouvoir faire des requêtes SQL.

2. Objectifs spécifiques

Nous étudierons ici un ensemble de commandes disponibles dans la plus part des systèmes de gestion de bases de données relationnelles (SGBDR). A la fin de la leçon, l'apprenant sera à mesure de pouvoir définir et créer des tables, d'enregistrer des informations dans ces tables et de pouvoir les manipuler. Il devra être à mesure de contrôler l'accès à ces informations du SGBD.

3. Présentation de SQL

SQL signifie Structured Query Language en français Langage d'interrogation structurée. Il est une interface de communication avec les SGBD relationnels. C'est un langage simple mais complet de gestion de bases de données relationnelles. Il s'appuie sur le schéma conceptuel pour exprimer des requêtes tout en laissant la stratégie d'exécution au SGBD.

Conçu par IBM dans les années 70, il devient le standard des SGBDR.

Il existe plusieurs normes SQL, mais la plus rependu aujourd'hui est la norme SQL2 qui a été définie en 1992. SQL3 (appelée aussi SQL99) est la nouvelle norme SQL qui intègre la notion de tables imbriquées.

4. Niveaux de gestion de données

Le langage SQL comporte trois niveaux de gestion des données:

- ✓ le langage de définition des données (LDD) qui permet la description et la création de la structure de la base de données (tables, relations, vues, attributs, contraintes d'intégrité, index) avec les commandes comme CREATE, ALTER ;
- ✓ le langage de manipulation des données (LMD) qui permet la manipulation des tables et des vues avec les quatre commandes : SELECT, INSERT, DELETE, UPDATE sans se préoccuper de l'organisation physique des données;

- ✓ le langage de contrôle des données (LCD) qui permet de contrôler la sécurité et les accès aux données avec les primitives de gestion des transactions : COMMIT, ROLLback et de privilèges d'accès aux données : GRANT et REVOKE.

5. Les types de données

Les types de données disponibles dépendent d'un SGBD à un autre. Les principaux types de données disponibles en SQL sont :

Libellé	Signification	Détails
Types numériques SQL		
SMALLINT	Nombre entier sur 2 octets	
INTEGER	Nombre entier sur 4 octets	
DECIMAL	Nombres décimaux avec un nombre fixe de décimales	DECIMAL (p, d) correspond à des nombres décimaux qui ont p chiffres significatifs et d chiffres après la virgule **ex. :** SALAIRE DECIMAL(8,2)
NUMERIC		même syntaxe que DECIMAL
REAL	Numériques non exacts à virgule flottante	simple précision, avec au moins 7 chiffres significatifs
DOUBLE ou FLOAT		double précision, avec au moins 15 chiffres significatifs
BIT	Le type BIT permet de ranger une valeur booléenne (un bit) en SQL-2	
NUMBER*	type numérique d'Oracle.	NUMBER(p) NUMBER(p, d) p et d représentent le nombre maximum de chiffres et de décimales
Types chaîne de caractères		
CHAR	pour les colonnes qui	CHAR(longueur) où longueur

	contiennent des chaînes de longueur constante.	est la longueur maximale (en nombre de caractères) pouvant être stocker dans le champ ; par défaut, longueur est égale à 1.
VARCHAR	pour les colonnes qui contiennent des chaînes de longueurs variables	VARCHAR(longueur) longueur indique la longueur maximale des chaînes pouvant contenir dans la colonne
VARCHAR2*	Type d'oracle, équivalent à VARCHAR	
Types temporels		
DOUBLE		
DATE	réserve 2 chiffres pour le mois et le jour et 4 pour l'année. Pour Oracle une donnée de type DATE inclut un temps en heures, minutes et secondes.	
TIME	pour les heures, minutes et secondes.	
TIMESTAMP	permet d'indiquer un moment précis par une date avec heures, minutes et secondes (6 chiffres après la virgule ; c'est-à-dire en microsecondes).	
TIMESTAMP INTERVAL	permet d'indiquer un intervalle de temps.	
Types binaires		
BIT et BIT VARYING	Longueur variable ou pas, permet d'enregistrer des données telles que les images et les sons.	Ce type de données varie d'un SGBD à un autre : **LONG RAW** pour Oracle, mais **IMAGE** pour Sybase, **BYTE**

		pour Informix, etc. Ce type de données sera pas n'utilisé dans cette leçon.
Valeur NULL		
NULL	La valeur NULL pour une colonne signifie qu'elle n'est pas renseignée, et donc vide. Cette valeur n'est pas zéro, c'est une absence de valeur	

Remarque : Les constantes chaînes de caractères doivent être entre des apostrophes ('). Si la chaîne contient déjà une apostrophe, celle-ci doit être doublée. Exemple : 'Aujourd''hui'

6. Contraintes

6.1 Notion de contrainte d'intégrité

Les tables d'un SGBD obéissent à une certaine logique de création qui doit respecter des règles strictes appelées contraintes d'intégrité.

Une contrainte d'intégrité est un automatisme du SGBD qui garantit que les valeurs d'une colonne ou d'un groupe de colonnes satisfont à une condition déclarée.

Les contraintes assurent la cohérence des données (concept d'intégrité). Elles sont décrites lors de la définition des tables de la base.

6.2 Types de contraintes

Il existe plusieurs types de contraintes :

- contrainte de valeurs (contrainte de domaine) ;
- unicité d'occurrences (unicité de lignes) ;
- clé primaire (unicité et indexation pour l'intégrité d'entité) ;
- clé étrangère (intégrité référentielle) ;
- caractère obligatoire ou facultatif des valeurs.

Les contraintes peuvent s'exprimer :

- soit au niveau colonne (contraintes de colonnes) : valables uniquement pour une colonne ;

- NULL ou NOT NULL (caractère facultatif/ obligatoire) ;
- UNIQUE (unicité des valeurs) ;
- PRIMARY KEY (clé primaire élémentaire) ;
- REFERENCES (intégrité de référence) ;
- CHECK (contraintes de valeurs).

- soit au niveau table (contraintes de tables) : valables pour un ensemble de colonnes d'une table.
  - UNIQUE (composition des colonnes) ;
  - PRIMARY KEY (clé primaire composée) ;
  - FOREIGN KEY… REFERENCES… (Composition de colonnes) ;
  - CHECK (condition particulière sur les valeurs).

Les contraintes se définissent :

- soit lors de la création des tables, dans l'ordre CREATE TABLE ;
- soit après la création des tables, par l'ordre ALTER TABLE permettant certaines modifications de la structure des tables.

Chaque contrainte définie est affectée d'un nom propre (nom de contrainte) stockée dans le dictionnaire des données de la base, implicitement par le SGBD (clé primaire) ou explicitement par la clause CONSTRAINT.

Les contraintes peuvent être activées (ENABLE) ou désactivées (DISABLE).

7. Langage de définition de données (LDD)

7.1 Création de table

Une table ou un tableau est une entité qui est contenu dans une base de données permettant de stocker des données ordonnées dans des colonnes.

La création d'une table sert à définir les colonnes et le type de données (entier, chaîne de caractères, date, valeur binaire …) qui seront contenus dans chacun des colonnes.

La création d'une table en SQL se fait par la commande CREATE TABLE.

Syntaxe

La syntaxe générale pour créer une table est la suivante :

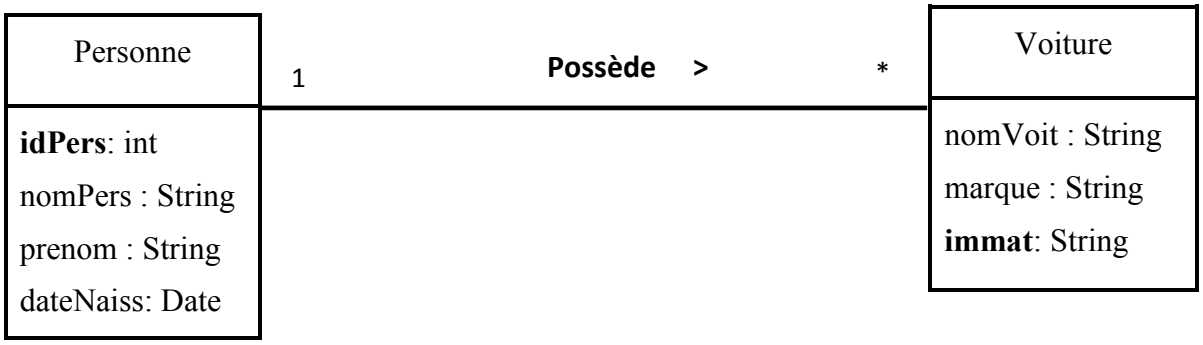
CREATE TABLE *[propriétaire_de_la_table.]nom_de_la_table*
(
col1 TYPE_DONNEES (taille) [DEFAULT expression] [NOT NULL],
col2 TYPE_DONNEES(taille) [DEFAULT expression] [NOT NULLl],
col3 TYPE_DONNEES (taille) [DEFAULT expression] [NOT NULL] ,
...
[PRIMARY KEY (col N, col M...)]
[FOREIGN KEY (col N, col M...) REFERENCES nom_table2 (col I)]
[ON DELETE CASCADE]
[ON DELETE SET NULL]
);

Le mot-clé « TYPE_DONNEES» définie le type de données (INT, DATE, TEXT, VARCHAR …). Pour chaque colonne, il est également possible de définir des options telles que :

- L'indicateur **NOT NULL :** empêche d'enregistrer une valeur nulle pour une colonne ;
- **DEFAULT :** attribuer une valeur par défaut si aucune donnée n'est indiquée pour cette colonne lors de l'ajout d'une ligne dans la table ;
- La clause **PRIMARY KEY :** indique si cette colonne est considérée comme clé primaire de la table pour un index ;
- La clause **FOREIGN KEY** : permet de définir une contrainte d'intégrité référentielle avec la clef primaire d'une autre table ;
- Les éléments entre crochets « [ ] » : indique qu'ils ne sont pas obligatoires.

Remarque : une clé étrangère ne peut pas faire référence à la clé primaire d'une autre table qui n'existe pas.
Si on veut utiliser la table créée par un autre utilisateur, il faut spécifier le nom de cet utilisateur devant le nom de la table : atraore.Voiture

Exemple : soit le modèle conceptuel suivant :



On détermine le modèle E/A correspondant :

*Personne (**idPers**,nomPers,prenom,dateNaiss) et Voiture(**immat**, nomVoit, marque, **idPers**)*

La requête SQL correspondent:

```
CREATE TABLE personne
(
    idPers INT PRIMARY KEY,
    nomPers VARCHAR(80),
    prenom VARCHAR(20),
    date_naissance DATE
);

CREATE TABLE voiture
(
    immat VARCHAR(25) PRIMARY KEY,
    nomVoit VARCHAR(50),
    marque VARCHAR(50),
    idPerso INT,
    FOREIGN KEY (idPerso) REFERENCES personne (idPers)
);
```

7.2 Modification et suppression de schéma d'une table

La partie de SQL qui permet de décrire les tables et autres objets manipulés par le SGBD est le langage de définition des données.

Le tableau ci-dessous résume l'ensemble des commandes utiles pour la description des données

Nom	Rôle et syntaxe	Exemple
ALTER TABLE	**Rôle** : Modifie le schéma de relation. **Syntaxe générale:** ALTER TABLE nom_table Instruction	
	Ajouter une colonne ALTER TABLE nom_table ADD nom_colonne type_donnees	ALTER TABLE Personne ADD sex VARCHR(8)
	Modifier une colonne ALTER TABLE nom_table MODIFY nom_colonne type_donnees ***Ou*** ALTER TABLE nom_table MODIFY COLUMN nom_colonne type_donnees	ALTER TABLE Personne MODIFY sex VARCHR(1)

	Renommer une colonne ALTER TABLE nom_table CHANGE colonne_ancien_nom colonne_nouveau_nom type_donnees ***Ou*** ALTER TABLE nom_table RENAME COLUMN colonne_ancien_nom TO colonne_nouveau_nom	ALTER TABLE Personne RENAME sex TO Sexe
	Supprimer une colonne ALTER TABLE nom_table DROP nom_colonne ***Ou*** ALTER TABLE nom_table DROP COLUMN nom_colonne	ALTER TABLE Personne DROP Sexe
DROP TABLE	**Rôle**: Permet de supprimer une relation ou une table d'un schéma de base de données. **Syntaxe** : DROP TABLE nom_table	DROP TABLE Personne
CONTRAINTES		
ALTER TABLE …CONSTRAI NT…	**Rôle**: Permet d'ajouter, modifier une contrainte **Syntaxe** : ALTER TABLE nom_table ADD CONSTRAINT nom_contrainte definition_de_la_contrainte	ALTER TABLE Personne ADD CONSTRAINT mineur_majeure CHECK(DateNaiss > '01/01/1997)
	Modifier d'une contrainte ALTER TABLE nom_table MODIFY CONSTRAINT nom_contrainte definition_de_la_contrainte	ALTER TABLE Personne MODIFY CONSTRAINT mineur_majeure DISABLE
	Supprimer une contrainte ALTER TABLE nom_table DROP CONSTRAINT nom_contrainte definition_de_la_contrainte	ALTER TABLE Personne DROP CONSTRAINT mineur_majeure

	Renommer une contrainte ALTER TABLE nom_table RENAME CONSTRAINT nom1 TO nom2	ALTER TABLE Personne RENAME CONSTRAINT prenom TO prenomPers
ALTER TABLE ... DISABLE/ENABLE	**Rôle**: Commande Oracle permettant d'enlever, différer des contraintes. **Syntaxe** : ALTER TABLE nom_table DISABLE/ENABLE CONSTRAINT nom_contrainte	ALTER TABLE Personne ENABLE CONSTRAINT mineur_majeure
Clause		
CASCADE CONSTRAINTS	**Rôle**: Permet de supprimer toutes les contraintes d'intégrités référentielles qui réfèrent aux clés (PRIMARY KEY et unique) de la relation supprimée. Cette clause concerne les relations où des contraintes de clés étrangères ont été spécifiées. Par exemple, la suppression de la table PERSONNE remet en cause la contrainte référentielle sur VOITURE spécifiée dans la relation POSSEDE. **Syntaxe** : DROP TABLE <nom_de_relation> [CASCADE CONSTRAINTS]	DROP TABLE Personne CASCADE CONSTRAINTS

7.3 Les index

7.3.1 Définition

Les index sont des ressources très utiles en SQL. Ils permettent d'accéder plus rapidement aux données.

La création d'un index permet de réduire les temps de recherche. Un index, dans une base de données se base sur le même principe qu'un index dans un livre.

La création d'index ne fait pas partie de la norme ANSI SQL. Néanmoins, la plus part des produits SQL supportent la création d'index. Si le mot clé UNIQUE est utilisé, le SGBD interdit les doublons.

7.3.2 Création et suppression d'index

Index		
Nom	**Rôle et syntaxe**	**Exemple**
CREATE INDEX	**Rôle**: Crée un index. **Syntaxe** : CREATE [UNIQUE] INDEX <nom_de_l'index> ON <nom_de_relation> (<nom_d'attribut> [{ASC | DESC}] [, ...])	
	Index sur une table `CREATE INDEX index_nom ON table;`	CREATE INDEX ix_Pers ON Personne
	Index sur une colonne `CREATE INDEX index_nom ON table (colonne1);`	CREATE INDEX ix_Datenais ON Personne (DateNaiss)
	Index sur plusieurs colonnes `CREATE INDEX index_nom ON table (colonne1, colonne2);`	CREATE INDEX ix_Nom ON Personne (nom, prenom)
	un index d'unicité `CREATE UNIQUE INDEX index_nom ON table (colonne1, colonne2);`	CREATE UNIQUE INDEX ix_PersUnique ON Personne (nom, prenom,DateNaiss)
DROP INDEX	**Rôle**: Permet de supprimer un index. Comme pour la commande DROP TABLE, le contenu de l'index et la définition de l'index elle-même sont supprimés. **Syntaxe** : DROP INDEX <nom_de_l'index>	DROP INDEX ix_PersUnique

7.4 Les vues

7.4.1 Définition

Une vue consiste en un ensemble d'enregistrements en provenance d'une ou plusieurs tables de la base de données. C'est une table virtuelle contenant les résultats d'un select. Cependant seule la définition de la vue est enregistrée dans la base, et pas les données.

7.4.2 Création et suppression

Vues		
Nom	Commentaires	Exemple
CREATE VIEW	**Rôle** : permet de Créer une vue. **Syntaxe** : CREATE VIEW <nom_de_la_vue> AS <commande_SELECT> [WITH CHECK OPTION]	CREATE VIEW Personne2 (nomP, prenomP) AS SELECT nomPers, prenom FROM Personne
DROP VIEW	**Rôle** : Permet de supprimer une vue. Contrairement à la commande DROP TABLE, les données incluses dans la vue ne seront pas détruites. Seule la définition de la vue est supprimée du catalogue. **Syntaxe** : DROP VIEW <nom_de_la_vue>	DROP VIEW Personne2

Remarque : La clause WITH CHECK OPTION garantit que toute opération de mise à jour (INSERT, UPDATE, DELETE) exécutée sur la vue sera contrôlée pour voir si le changement est conforme à la définition de la vue. Un changement non conforme sera rejeté. Si cette clause n'est pas indiquée, les changements pourront être acceptés par les relations sur lesquelles est définie la vue.

7.5 Quelques nommages

Il n'existe pas de convention de nommage spécifique. Mais quelques développeurs et administrateurs de bases de données suggèrent dans le nommage de préfixer:

- Préfixe « PK_ » pour **P**rimary **K**ey (traduction : clé primaire)
- Préfixe « FK_ » pour **F**oreign **K**ey (traduction : clé étrangère)
- Préfixe « UK_ » pour **U**nique **K**ey (traduction : clé unique)
- Préfixe « UX_ » pour **U**nique **I**ndex (traduction : index unique)
- Préfixe « IX_ » pour chaque autre **I**nde**X**

8. Le langage de manipulation des données (LMD)

Les commandes du LMD (SELECT, INSERT, UPDATE, DELETE) permettent de manipuler des enregistrements d'une relation. Ces opérations n'affectent pas le schéma de la relation et ceci même dans le cas où la commande a pour effet de supprimer tous les enregistrements de la relation.

8.1 Expression générale d'une requête

8.1.1 Présentation d'une requête

Une requête se présente généralement sous la forme " **SELECT** exp1, exp2,... **FROM** table **WHERE** prédicat " :

- la clause **SELECT** exprime le résultat attendu sous la forme d'une liste d'attributs auxquels il est possible d'appliquer différents opérateurs et fonctions ;
- la clause **FROM** liste les relations utilisées pour évaluer les requêtes;
- la clause **WHERE** qui est facultative énonce une condition que doivent respecter les enregistrements sélectionnés ;
- **table** est le nom de la table sur laquelle porte la sélection ;
- **exp1, exp2,... est** la liste des expressions (colonnes, constantes,…) que l'on veut obtenir. Cette liste peut être *, auquel cas toutes les colonnes de la table sont sélectionnées.

8.1.2 Syntaxe

SELECT [DISTINCT] { * | <nom_de_relation>,<nom_d'attribut> [alias] | <nom_d'attribut>[alias] [, ...]
FROM [<nom_d'utilisateur>]<nom_de_relation> [alias] [, ...]
[WHERE <condition>]
[GROUP BY <nom_d'attribut> [, ...] [HAVING <condition>]]
[{UNION | INTERSECT | MINUS [ALL]} <commande_SELECT>
[ORDER BY {<nom_d'attribut> | <numéro_de_colonne>} [{ASC | DESC}] [, ...]

8.2 Lien entre algèbre relationnelle et SQL

Avant d'introduire l'utilité des différentes clauses facultatives ainsi que les différentes possibilités offertes par SQL, nous montrons le lien entre algèbre relationnel et SQL.

Soient les schémas relationnels R1, R2, R3 et R4 définis comme suit :

R1 (A :D1, B :D2)
R2 (C :D1, D :D2)
R3 (A :D1, E :D3)
R4 (B :D2)

Opération	Expression	Expression SQL équivalente

	algébrique	
Projection	Π_A (R1)	SELECT A FROM R1
Sélection	σ < condition > (R1)	SELECT * FROM R1 WHERE <condition >
Produit cartésien	R1 X R2	SELECT * FROM R1, R2
Jointure	R1 * R3	SELECT * FROM R1, R3 WHERE R1.A = R3.A
Union	R1 U R2	SELECT * FROM R1 UNION SELECT * FROM R2
Intersection	R1 $\cap$ R2	SELECT * FROM R1, R2 WHERE R1.A = R2.C and R1.B = R2.D
Différence	R1 / R2	SELECT * FROM R1 WHERE not exists (SELECT * FROM R2 WHERE R2.C = R1.A and R2.D = R1.B)
Division	R1 : R2	SELECT A FROM R1 GROUP BY A HAVING COUNT (distinct B) = (SELECT count (distinct B) FROM R2)

8.3 Les requêtes imbriquées

Requêtes imbriquées		
Nom	Commentaires	Exemple
IN	Permet de tester la présence d'une valeur particulière dans un ensemble.	
NOT IN	Permet de tester l'absence d'une valeur particulière dans un ensemble.	
ALL	Compare chacune des valeurs de l'ensemble à une valeur particulière et retourne "VRAI" si la comparaison est évaluée pour chacun des éléments. Les comparateurs sont: <, <=, >, >=, =, != .	
ANY	Compare chacune des valeurs de l'ensemble à une valeur particulière et retourne "VRAI" si la comparaison est évaluée à "VRAI" pour au moins un des éléments. Les comparateurs sont les même que ceux cités précédement.	
EXISTS	Retourne "VRAI" si une requête imbriquée retourne au moins une ligne.	

8.4 Les prédicats

Prédicats

Nom	Commentaires	Exemple
BETWEEN	Teste l'appartenance d'une valeur à un intervalle.	
LIKE	Permet de faire une recherche approximative.	
IS NULL	Permet de tester si un champ a été affecté.	

8.5 Les clauses

Clauses		
Nom	Commentaires	Exemple
GROUP BY	Application de fonction agrégats à des collections d'enregistrements reliées sémantiquement.	
HAVING	Cette clause ne s'emploie qu'avec un "GROUP BY". Exprime une condition sur le groupe d'enregistrement associé à chaque valeur du groupage.	
ORDER BY	Permet l'ordonnancement du résultat avant l'affichage.	
DISTINCT	Elimine les doublons avant d'utiliser une fonction agrégat.	

8.6 Les fonctions agrégats

Ces fonctions ne peuvent être utilisées que dans une clause SELECT ou dans une clause HAVING .

Fonctions agrégats		
Nom	**Commentaires**	**Exemple**
COUNT	Rôle : Dénombre les lignes sélectionnées. Syntaxe : COUNT (expr)	
SUM	Rôle : Additionne les valeurs de type numérique.	

	Syntaxe : SUM (expr)	
MIN	Rôle : Retourne la valeur minimale d'une colonne de type caractère ou numérique. Syntaxe : MIN (expr)	
MAX	Rôle : Retourne la valeur maximale d'une colonne de type caractère ou numérique. Syntaxe : MAX (expr)	
AVG	Rôle : Calcule la moyenne d'une colonne de type numérique. Syntaxe : AVG (expr)	

Remarque : On peut préfixer expression (expr) par les mots clés [DISTINCT | ALL].

8.7 La mise à jour de données dans une relation

Instructions de mises à jour		
Nom	**Commentaires**	**Exemple**
INSERT	**Rôle** : Permet d'ajouter un ou plusieurs éléments (enregistrements) dans une relation en précisant toutes les valeurs ou uniquement les non-nulles. **Syntaxe** : INSERT INTO <nom_de_relation > [(<liste_d'attributs>)] VALUES (<liste_de_valeurs >)]	
UPDATE	Rôle : Permet de modifier un ou plusieurs individus. Syntaxe :	

	UPDATE <nom_de_relation> SET < champ >	
DELETE	**Rôle** : Supprime un ou plusieurs éléments dans une relation. Si la relation devient vide après l'action de DELETE, la relation n'est pas supprimée. **Syntaxe** : DELETE FROM < nom_de_relation > [WHERE < expression_de_selection >]	

Remarque : La norme SQL2 offre d'autres options qui ne sont pas forcément implémentée dans Oracle :

ON DELETE SET DEFAULT met une valeur par défaut dans la clé étrangère quand la clé primaire référencée est supprimée.

ON DELETE CASCADE supprime la clé étrangère si on supprime la clé primaire.

ON DELETE SET NULL met NULL dans la clé étrangère quand la clé primaire référencée est supprimée.

ON DELETE SET DEFAULT met une valeur par défaut dans la clé étrangère quand la clé primaire référencée est supprimée.

ON DELETE RESTRICT Permet d'interdire la suppression d'un enregistrement référencé par un enregistrement d'une autre relation. On trouve également l'expression NO ACTION à la place du mot clé RESTRICT dans certains SGBD

ON UPDATE CASCADE modifie la clé étrangère si on modifie la clé primaire (ce qui est à éviter).

ON UPDATE SET NULL met NULL dans la clé étrangère quand la clé primaire référencée est modifiée.

ON UPDATE SET DEFAULT met une valeur par défaut dans la clé étrangère quand la clé primaire référencée est modifiée.

9. Le langage de contrôle des données

Cette partie du langage SQL comporte deux commandes, la commande GRANT pour donner des privilèges ou droits à des utilisateurs et la commande REVOKE pour supprimer des droits préalablement accordés.

Un privilège est une autorisation d'exécuter un acte. Les privilèges les plus courants sont :

Permission	Signification
SELECT	droit de lecture
INSERT	droit d'insertion de lignes
UPDATE	droit de modification de lignes
UPDATE (col1, col2, ...)	droit de modification de lignes limité à certaines colonnes
DELETE	droit de suppression de lignes
ALTER	droit de modification de la définition de la table
INDEX	droit de création d'index
ALL	tous les droits ci-dessus

Les privilèges SELECT, INSERT et UPDATE s'appliquent aux tables et aux vues. Les autres s'appliquent uniquement aux tables

Un utilisateur ne peut exécuter que les commandes SQL pour lesquelles les droits lui ont été explicitement attribués ou qu'il possède implicitement car l'objet accédé a été créé par lui-même.

Instructions de mises à jour		
Nom	**Commentaires**	**Exemple**
GRANT	**Rôle** : La commande GRANT permet d'attribuer des privilèges **Syntaxe** : GRANT < privilège > [, ...] ON [TABLE] <nom_de_relation> TO {<nom_d'utilisateur > [, ...] | PUBLIC} **Ou** GRANT privilège ON table /vue TO utilisateur [WITH GRANT OPTION]	- GRANT SELECT ON Personne TO traore - GRANT SELECT ON Personne2 TO PUBLIC - GRANT SELECT, UPDATE ON Personne TO traore, hien
REVOKE	**Rôle** : permet de supprimer des privilèges **Syntaxe** : REVOKE <privilège > [, ...] ON [TABLE]	REVOKE SELECT ON Personne FROM traore

	<nom_de_relation> FROM {<nom_d'utilisateur> [, ...] | PUBLIC} **Ou** REVOKE privilège ON table /vue FROM utilisateur	

Remarque : Un utilisateur ayant reçu un privilège avec l'option facultative [WITH GRANT OPTION] peut le transmettre à son tour.

Si on enlève un privilège à un utilisateur, ce privilège est automatiquement retiré à tout autre utilisateur à qui il aurait accordé ce privilège.

10. Autres

10.1 Connexion et déconnexion

On entre dans SQL*PLUS par la commande :

SQLPLUS nom /mot-de-passe

Pour se déconnecter l'ordre SQL à taper est *EXIT*

10.2 Changement de mot de passe

Tout utilisateur peut modifier son mot de passe par l'ordre GRANT CONNECT :

GRANT CONNECT TO utilisateur IDENTIFIED BY mot-de-passe

10.3 Synonyme

Oracle (et d'autres SGBD comme SQL Server) permet de donner des synonymes aux tables et vues.

Les synonymes permettent le plus souvent de donner des noms simplifiés aux tables et vues.

Ils peuvent être publics ou privés. Publics s'ils sont connus de tous les utilisateurs de la base, et privés s'ils ne sont connus que de celui qui les a créé.

Les synonymes publics sont particulièrement intéressants quand ils permettent de ne pas préfixer une table ou une vue par le nom du propriétaire lorsque l'utilisateur du synonyme n'est pas le propriétaire de la table ou de la vue.

Création : *CREATE [PUBLIC | PRIVATE] SYNONYM nom_synonyme FOR objet*

Suppression : *DROP SYNONYM nom_synonyme*

Exemple

CREATE PUBLIC SYNONYM departement FOR toto.dept

CREATE PUBLIC SYNONYM personne FOR traore.personne

La table du dictionnaire all_synonym contient les définitions des synonymes.

10.4 Création d'utilisateur

Etant donné que nous ne faisons pas de cours d'administration de base de données, nous n'allons pas ns appesantir sur ces notions. On peut créer un utilisateur par la commande :

CREATE USER nom_user IDENTIFIED BY password_user

Exemple sous Oracle:

```
CREATE USER atraore IDENTIFIED BY 123adc
DEFAULT TABLESPACE t2
TEMPORARY TABLESPACE temp_t2
QUOTA 1M ON t2;
GRANT R_etudiant TO atraore;
```

R_etudiant est un rôle, c'est-à-dire un ensemble de droits que l'on peut donner par GRANT

10.5 Création de rôle

La commande *CREATE ROLE nom_rôle* permet de créer un rôle.

On peut attribuer à ce rôle des droits comme on le fait avec un utilisateur par la commande GRANT.

Exemple

GRANT CONNECT, RESOURCE, SELECT, UPDATE ON Personne TO R_etudiant

Des rôles sont prédéfinis par Oracle : CONNECT permet à un utilisateur de se connecter et de créer des tables. RESOURCE permet en plus à un utilisateur de créer des procédures stockées, des types, des séquences, etc.

Dans les leçons suivantes nous aborderons la notion de procédures stockées, de séquences et de déclencheurs (Triggers).

Activités d'apprentissage

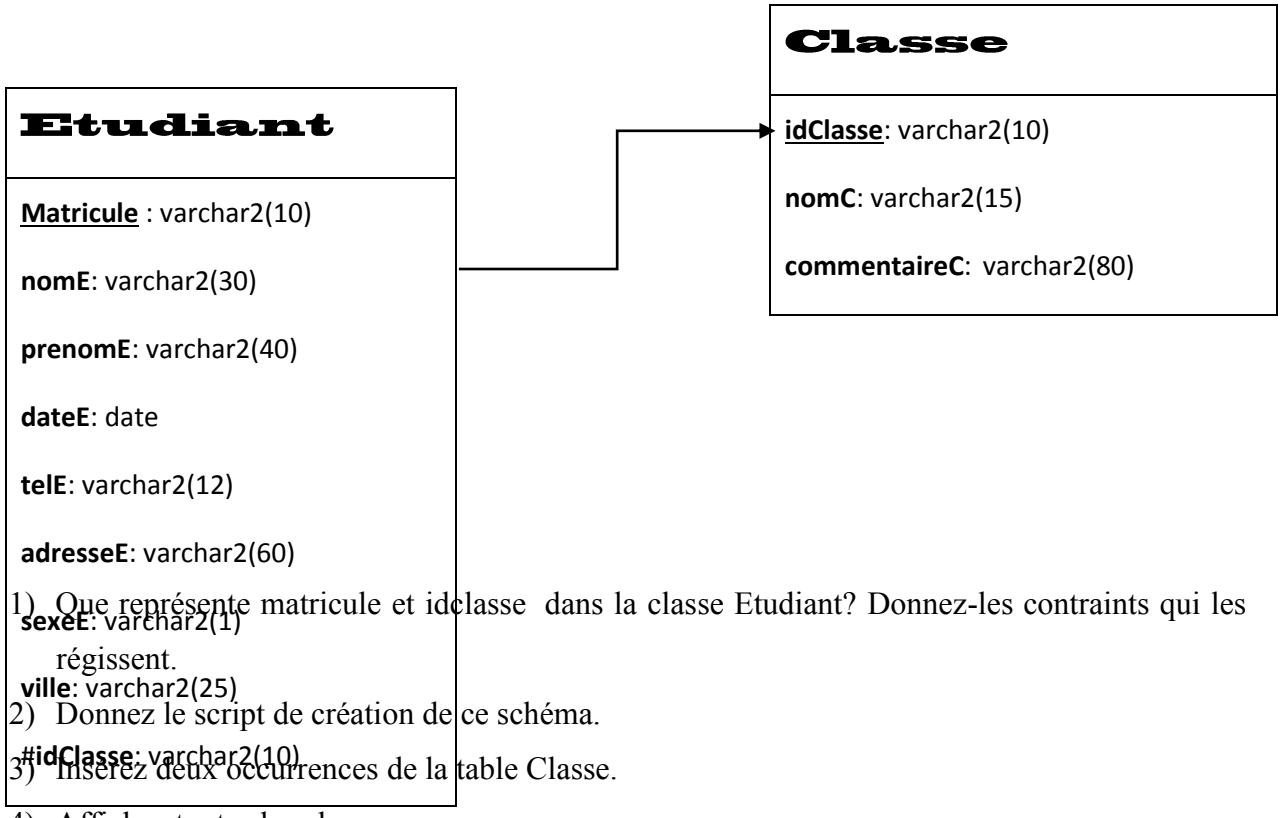
Voir en plus des deux exercices la fiche de TD et TP

Exercice 1 :

1. Que signifie SQL
2. Que renvoyer la requête suivante si taux est null : Select salaire*taux from Emp ; comment corrige-t-on
3. Les différents types de langage SQL et citez un ordre dans chaque cas.
4. Donnez le rôle des ordres GRANT et REVOKE
5. Supposons que vous avez un utilisateur Issouf dans votre BD. Attribuer tous les droits à cet utilisateur Issouf.

Exercice 2 :

On a le schéma relationnel suivant :



1) Que représente matricule et idclasse dans la classe Etudiant? Donnez-les contraints qui les régissent.
2) Donnez le script de création de ce schéma.
3) Insérez deux occurrences de la table Classe.
4) Affichez toutes les classes.
5) Donnez la liste des étudiants par ville.
6) Quel est le nombre d'étudiants de la classe info2.
7) Renommez la colonne dateE en dateNaisE.
8) Donnez la requête de suppression de la colonne commentaireC.
9) Quels sont les noms des étudiants qui habitent Paris ?

10) Créez une vue (view) pour obtenir les noms et les prénoms des étudiants de la classe info3
11) Affichez la liste des 10 étudiants les plus âgés.