Le langage SQL

Introduction

- SQL : Structured Query Language
- accéder aux données de la base de données,
- langage adapté aux bases de données relationnelles,
- existe sur tous les SGBD relationnels (Oracle, Access,...),
- défini par une norme ISO,
- est utilisé dans les bases de données Oracles depuis 1979.

Caractéristiques de SQL

SQL est un langage de définition de données

SQL est un langage de définition de données (LDD), c'est-à-dire qu'il permet de créer des tables dans une base de données relationnelle, ainsi que d'en modifier ou en supprimer.

Caractéristiques de SQL

SQL est un langage de définition de données

SQL est un langage de définition de données (LDD), c'est-à-dire qu'il permet de créer des tables dans une base de données relationnelle, ainsi que d'en modifier ou en supprimer.

SQL est un langage de manipulation de données

SQL est un langage de manipulation de données (LMD), cela signifie qu'il permet de sélectionner, insérer, modifier ou supprimer des données dans une table d'une base de données relationnelle.

Caractéristiques de SQL

SQL est un langage de définition de données

SQL est un langage de définition de données (LDD), c'est-à-dire qu'il permet de créer des tables dans une base de données relationnelle, ainsi que d'en modifier ou en supprimer.

SQL est un langage de manipulation de données

SQL est un langage de manipulation de données (LMD), cela signifie qu'il permet de sélectionner, insérer, modifier ou supprimer des données dans une table d'une base de données relationnelle.

SQL est un langage de protections d'accès

Il est possible avec SQL de définir des permissions au niveau des utilisateurs d'une base de données. On parle de DCL (Data Control Language).

Définition de données

Création de table

Pour créer une table, on utilise le couple de mots-clés **CREATE TABLE**. La syntaxe est la suivante :

```
CREATE TABLE NomTable (
NomColonne1 TypeDonnée1,
NomColonne2 TypeDonnée2,
...
);
```

Insertion de lignes à la création

Il est possible des créer une table un insérant directement des lignes lors de la création. On récupère les lignes à insérer avec **AS SELECT**.

```
CREATE TABLE NomTable (
Nom_de_colonne1 Type_de_donnée,
Nom_de_colonne2 Type_de_donnée,
...
)

AS SELECT NomChamp1,
  NomChamp2,
  ...

FROM NomTable2

WHERE ...;
```

Les type de données

- numériques : **SMALLINT** (16 bits), **INTEGER** (32 bits), **FLOAT**
- chaînes de caractères : **CHAR(n)** (longueur fixée), **VARCHAR(n)** (longueur maximale fixée)
- date : **DATE**, **TIME**, **TIMESTAMP**

Les contraintes d'intégrité

- pour nommer une contrainte : **CONSTRAINT**
- définir une valeur par défaut : **DEFAULT**
- préciser que la saisie est obligatoire : **NOT NULL**
- vérifier qu'une valeur saisie pour un champ n'existe pas déjà dans la table : **UNIQUE**
- test sur un champ : **CHECK**

Les contraintes d'intégrité

- pour nommer une contrainte : **CONSTRAINT**
- définir une valeur par défaut : **DEFAULT**
- préciser que la saisie est obligatoire : **NOT NULL**
- vérifier qu'une valeur saisie pour un champ n'existe pas déjà dans la table : **UNIQUE**
- test sur un champ : **CHECK**

Exemple :

```
CREATE TABLE Clients(
Nom char(30) NOT NULL,
Prenom char(30) NOT NULL,
Age integer, check (Age < 100),
Email char(50) NOT NULL, check (Email LIKE "%@%")
)
```

Définition des clés

Rappel : une clé est une (ou plusieurs) colonne(s) dont la connaissance permet de préciser un et un seul n-uplet.

Définition des clés

Rappel : une clé est une (ou plusieurs) colonne(s) dont la connaissance permet de préciser un et un seul n-uplet.

- clé primaire : **PRIMARY KEY**

  ```
  PRIMARY KEY (colonne1, colonne2, ...)
  ```

Définition des clés

Rappel : une clé est une (ou plusieurs) colonne(s) dont la connaissance permet de préciser un et un seul n-uplet.

- clé primaire : **PRIMARY KEY**

  ```
  PRIMARY KEY (colonne1, colonne2, ...)
  ```

- clé étrangère : **FOREIGN KEY...REFERENCES...**

  ```
  FOREIGN KEY (colonne1, colonne2, ...)
  REFERENCES NomTableEtrangere(colonne1,colonne2,...)
  ```

Mise à jour d'informations

- ajouter des n-uplets : **INSERT INTO**

Mise à jour d'informations

- ajouter des n-uplets : **INSERT INTO**
  - insérer une seule ligne :

```
INSERT INTO NomTable(colonne1,colonne2,colonne3,...)
VALUES (Valeur1,Valeur2,Valeur3,...)
```

Mise à jour d'informations

- ajouter des n-uplets : **INSERT INTO**
  - insérer une seule ligne :

    ```
    INSERT INTO NomTable(colonne1,colonne2,colonne3,...)
    VALUES (Valeur1,Valeur2,Valeur3,...)
    ```

  - insérer plusieurs lignes :

    ```
    INSERT INTO NomTable(colonne1,colonne2,...)
    SELECT colonne1,colonne2,... FROM NomTable2
    WHERE qualification
    ```

  (les valeurs inconnues prennent la valeur **NULL**)

Mise à jour d'informations

- ajouter des n-uplets : **INSERT INTO**
  - insérer une seule ligne :

    ```
    INSERT INTO NomTable(colonne1,colonne2,colonne3,...)
    VALUES (Valeur1,Valeur2,Valeur3,...)
    ```

  - insérer plusieurs lignes :

    ```
    INSERT INTO NomTable(colonne1,colonne2,...)
    SELECT colonne1,colonne2,... FROM NomTable2
    WHERE qualification
    ```

  (les valeurs inconnues prennent la valeur **NULL**)
- modifier des n-uplets existants : **UPDATE...SET...WHERE...**

```
UPDATE NomTable
SET Colonne = ValeurOuExpression
WHERE qualification
```

Mise à jour d'informations

- ajouter des n-uplets : **INSERT INTO**
  - insérer une seule ligne :

    ```
    INSERT INTO NomTable(colonne1,colonne2,colonne3,...)
    VALUES (Valeur1,Valeur2,Valeur3,...)
    ```

  - insérer plusieurs lignes :

    ```
    INSERT INTO NomTable(colonne1,colonne2,...)
    SELECT colonne1,colonne2,... FROM NomTable2
    WHERE qualification
    ```

  (les valeurs inconnues prennent la valeur **NULL**)
- modifier des n-uplets existants : **UPDATE...SET...WHERE...**

  ```
  UPDATE NomTable
  SET Colonne = ValeurOuExpression
  WHERE qualification
  ```

- supprimer des n-uplets : **DELETE FROM...WHERE...**

  ```
  DELETE FROM NomTable
  WHERE qualification
  ```

Manipulation de données

Modification de la table

- suppression d'éléments : **DROP** (VIEW, INDEX, TABLE)

  `DROP TABLE NomTable`

  (supprime les données et la structure de la table)

Modification de la table

- suppression d'éléments : **DROP** (VIEW, INDEX, TABLE)

  `DROP TABLE NomTable`

  (supprime les données et la structure de la table)
- suppression de données uniquement : **TRUNCATE**

  `TRUNCATE TABLE NomTable`

Modification de la table

- suppression d'éléments : **DROP** (VIEW, INDEX, TABLE)

  `DROP TABLE NomTable`

  (supprime les données et la structure de la table)
- suppression de données uniquement : **TRUNCATE**

  `TRUNCATE TABLE NomTable`
- renommer une table : **RENAME**

  `RENAME TABLE AncienNom TO NouveauNom`

Modification de la table

- suppression d'éléments : **DROP** (VIEW, INDEX, TABLE)

  ```
  DROP TABLE NomTable
  ```

  (supprime les données et la structure de la table)
- suppression de données uniquement : **TRUNCATE**

  ```
  TRUNCATE TABLE NomTable
  ```
- renommer une table : **RENAME**

  ```
  RENAME TABLE AncienNom TO NouveauNom
  ```
- ajouter des commentaires à une table (ou une vue ou certaines colonnes) : **COMMENT**

  ```
  COMMENT NomTable IS 'Commentaires';
  COMMENT NomVue IS 'Commentaires';
  COMMENT NomTable.NomColonne IS 'Commentaires';
  ```

Modifications de schémas

Avec **ALTER** on peut modifier les colonnes d'une table :

- Modifier le type d'une colonne

```
ALTER TABLE NomTable
MODIFY NomColonne TypeDonnees
```

Modifications de schémas

Avec **ALTER** on peut modifier les colonnes d'une table :

- Modifier le type d'une colonne

```
ALTER TABLE NomTable
MODIFY NomColonne TypeDonnees
```

- ajouter de nouvelles colonnes

```
ALTER TABLE NomTable
ADD NomColonne TypeDonnees
```

Modifications de schémas

Avec **ALTER** on peut modifier les colonnes d'une table :

- Modifier le type d'une colonne

  ```
  ALTER TABLE NomTable
  MODIFY NomColonne TypeDonnees
  ```

- ajouter de nouvelles colonnes

  ```
  ALTER TABLE NomTable
  ADD NomColonne TypeDonnees
  ```

- supprimer des colonnes

  ```
  ALTER TABLE NomTable
  DROP COLUMN NomCcolonne
  ```

  (possible que si la colonne ne fait pas partie d'une vue, ne fait pas l'objet d'une contrainte d'intégrité)

Modifications de schémas (2)

- ajouter de nouvelles contraintes

```
ALTER TABLE NomTable
ADD CONSTRAINT NomContrainte
```

Modifications de schémas (2)

- ajouter de nouvelles contraintes

```
ALTER TABLE NomTable
ADD CONSTRAINT NomContrainte
```

- supprimer des contraintes

```
ALTER TABLE NomTable
DROP CONSTRAINT NomContrainte
```

Modifications de schémas (2)

- ajouter de nouvelles contraintes

```
ALTER TABLE NomTable
ADD CONSTRAINT NomContrainte
```

- supprimer des contraintes

```
ALTER TABLE NomTable
DROP CONSTRAINT NomContrainte
```

- activer ou désactiver toutes les contraintes

```
ALTER TABLE NomTable
CHECK CONSTRAINT NomContrainte

ALTER TABLE NomTable
NOCHECK CONSTRAINT ALL
```

Création de vues

Une vue est une table virtuelle évaluée à chaque consultation (les données ne sont pas stockée dans une table de la BD). Une vue est définie par une close **SELECT**.

Création de vues

Une vue est une table virtuelle évaluée à chaque consultation (les données ne sont pas stockée dans une table de la BD). Une vue est définie par une close **SELECT**.

- obtenir des informations synthétiques
- éviter de divulguer certaines informations
- assurer l'indépendance du schéma externe

Création de vues (2)

La syntaxe pour définir une vue est la suivante :

```
CREATE VIEW NomVue(NomColonne1,...)
AS SELECT NomColonne1,..
FROM NomTable
WHERE Condition
```

Création de vues (2)

La syntaxe pour définir une vue est la suivante :

```
CREATE VIEW NomVue(NomColonne1,...)
AS SELECT NomColonne1,..
FROM NomTable
WHERE Condition
```

Exemple :

```
CREATE VIEW EtudiantsSrc(Nom,Prenom)
AS SELECT Nom,Prenom
FROM Etudiants
WHERE n_formation=12
```

Interrogation d'un BD

La commande **SELECT** permet d'interroger une BD. La syntaxe est la suivante :

```
SELECT [ALL|DISTINCT] NomColonne1,... | *
FROM NomTable1,...
WHERE Condition
```

Interrogation d'un BD

La commande **SELECT** permet d'interroger une BD. La syntaxe est la suivante :

```
SELECT [ALL|DISTINCT] NomColonne1,... | *
FROM NomTable1,...
WHERE Condition
```

- **ALL** : option par défaut, sélectionne toutes les lignes qui satisfont la condition
- **DISTINCT** : permet d'éliminer les doublons

La clause **AS**

La clause **AS** permet de les champs dans une requête définie par **SELECT**.

La clause **AS**

La clause **AS** permet de les champs dans une requête définie par **SELECT**.
Exemple :

```
SELECT Compteur AS Ctp
FROM Vehicule
```

Restriction

Les conditions peuvent faire appel aux opérateurs suivants :

- opérateurs logiques : **AND**, **OR**, **NOT**
- comparateurs de chaînes : **IN**, **BETWEEN**, **LIKE**
- opérateurs arithmétiques : $+, -, /, \%$
- comparateurs arithmétiques : $=, \neq, <, >, \leq, \geq, <>$

Restriction

Les conditions peuvent faire appel aux opérateurs suivants :

- opérateurs logiques : **AND**, **OR**, **NOT**
- comparateurs de chaînes : **IN**, **BETWEEN**, **LIKE**
- opérateurs arithmétiques : $+, -, /, \%$
- comparateurs arithmétiques : $=, \neq, <, >, \leq, \geq, <>$

Exemple :

```
SELECT *
FROM Vehicules
WHERE (Compteur>10000) AND (Compteur<=30000)
```

Restriction sur une comparaison de chaînes

Le prédicat **LIKE** permet de faire des restrictions sur des chaînes grace à des caractères appelés caractères joker :

- le caractère % remplace une séquence (éventuellement vide) de caractères
- le caractère _ remplace un caractère
- les caractères [−] définissent un intervalle de caractères (par exemple $[A - E]$)

Restriction sur une comparaison de chaînes

Le prédicat **LIKE** permet de faire des restrictions sur des chaînes grace à des caractères appelés caractères joker :

- le caractère % remplace une séquence (éventuellement vide) de caractères
- le caractère _ remplace un caractère
- les caractères [−] définissent un intervalle de caractères (par exemple $[A - E]$)

Exemple :

```
SELECT *
FROM Vehicules
WHERE Marque LIKE "_e%"
```

Autres fonctions

Autres fonctions sur les chaînes de caractères :

- **CONCAT**
- **TRANSLATE**
- **REPLACE**
- **LENGTH**,...

Restriction sur un ensemble

Les prédicats **BETWEEN** et **IN** vérifient

- qu'une valeur se trouve dans un intervalle
- qu'une valeur appartient à une liste de valeurs

Restriction sur un ensemble

Les prédicats **BETWEEN** et **IN** vérifient

- qu'une valeur se trouve dans un intervalle
- qu'une valeur appartient à une liste de valeurs

Exemple :

```
SELECT *
FROM Vehicules
WHERE Compteur BETWEEN 10000 AND 30000

SELECT *
FROM Vehicules
WHERE Marque IN ("Peugeot","Citroën")
```

Les dates

Différents formats de date :

- Day, Month DD,YYYY HH24 :MI :SS
  Friday, December 28,2001 23 :16 :41
- DD/MM/YYYY HH :MI :SS AM
  28/12/2001 11 :16 :41 AM

Les formats de date et heure peuvent être utilisés en argument des fonctions de formatage.

Les formats de dates

- **YEAR|SYEAR** représente une année
- **YYYY|SYYYY** représente une année sur 4 chiffres
- **YYY|YY|Y** représente les trois, deux ou un chiffre(s) d'une année
- **q** représente le trimestre d'une année (1 à 4)
- **MM** représente le mois sur deux chiffres (1 à 12)
- **MONTH** représente le littéral du mois (JANUARY à DECEMBER)
- **MON** représente une abbréviation du littéral du mois (JAN à DEC)
- **WW** représente le numéro de la semaine sur une année (1 à 53)
- **W** représente le numéro de la semained d'un mois (1 à 5)
- **...**

Les fonctions sur les dates

- addition $+$, soustraction $-$
- **ROUND(date,format)** renvoie la date spécifiée selon le format choisi
- **ADD_MONTHS(date,nb_mois)** ajoute un nombre de mois spécifiés à une date
- **CURRENT_DATE** retourne la date courante
- **CURRENT_TIME** retourne l'heure courante
- **CURRENT_TIMESTAMP** retourne la date et l'heure courantes
- **LAST_DAY(date)** retourne une date qui représente le dernier jour du mois dans lequel la date s'est produite
- **MONTHS_BETWEEN(d1,d2)** retourne le nombre de mois entre les deux valeurs de date
- **NEXT_DAY(date,chaîne)** retourne la date du premier jour de la semaine correspondant à la chaîne de caractères spécifiée
- **TO_DATE(chaîne)** convertit une chaîne de caractères en une valeur de date et d'heure valide
- **TRUNC(date)** retourne le résultat d'une troncature de date
- **...**

La fonction **INTERVAL**

p représente la précision (p=3 permet de stocker 999 jours) :

- **INTERVAL MONTH(p)** nombre de mois écoulés entre deux dates
- **INTERVAL YEAR(p)** nombre d'années
- **INTERVAL YEAR(p) TO MONTH** nombres d'années et de mois
- **INTERVAL DAY(p)** nombre de jours
- **INTERVAL DAY(p) TO HOUR** nombre de jours et d'heures
- **INTERVAL DAY(p1) TO SECOND(p2)** nombre de jours, minutes et secondes

Exemples

Nombre de mois entre la date de la commande et la date de la livraison :

```
SELECT MONTHS_BETWEEN(date_cmd,date_liv)
FROM commande
```

Exemples

Nombre de mois entre la date de la commande et la date de la livraison :

```
SELECT MONTHS_BETWEEN(date_cmd,date_liv)
FROM commande
```

Ajout de 30 jours à la date de commande :

```
SELECT id_cmd,date_cmd+INTERVAL '30' DAY
FROM commande
```

Tri et autres opérations

- le tri : **ORDER BY...DESC** ou **ORDER BY...ASC**

```
SELECT *
FROM Vehicules
ORDER BY Marque ASC, Compteur DESC
```

Tri et autres opérations

- le tri : **ORDER BY...DESC** ou **ORDER BY...ASC**

```
SELECT *
FROM Vehicules
ORDER BY Marque ASC, Compteur DESC
```

- la moyenne d'une colonne : **AVG**

Tri et autres opérations

- le tri : **ORDER BY...DESC** ou **ORDER BY...ASC**

```
SELECT *
FROM Vehicules
ORDER BY Marque ASC, Compteur DESC
```

- la moyenne d'une colonne : **AVG**
- le nombre de lignes d'une table : **COUNT**

Tri et autres opérations

- le tri : **ORDER BY...DESC** ou **ORDER BY...ASC**

```
SELECT *
FROM Vehicules
ORDER BY Marque ASC, Compteur DESC
```

- la moyenne d'une colonne : **AVG**
- le nombre de lignes d'une table : **COUNT**
- la valeur maximale d'une colonne : **MAX**

Tri et autres opérations

- le tri : **ORDER BY...DESC** ou **ORDER BY...ASC**

```
SELECT *
FROM Vehicules
ORDER BY Marque ASC, Compteur DESC
```

- la moyenne d'une colonne : **AVG**
- le nombre de lignes d'une table : **COUNT**
- la valeur maximale d'une colonne : **MAX**
- la valeur minimale d'une colonne : **MIN**

Tri et autres opérations

- le tri : **ORDER BY...DESC** ou **ORDER BY...ASC**

```
SELECT *
FROM Vehicules
ORDER BY Marque ASC, Compteur DESC
```

- la moyenne d'une colonne : **AVG**
- le nombre de lignes d'une table : **COUNT**
- la valeur maximale d'une colonne : **MAX**
- la valeur minimale d'une colonne : **MIN**
- la somme des valeurs d'une colonne : **SUM**
- ...

Opérations ensemblistes

Le produit cartésien :

```
SELECT *
FROM NomTable1,NomTable2
WHERE ...
```

Opérations ensemblistes

Le produit cartésien :

```
SELECT *
FROM NomTable1,NomTable2
WHERE ...
```

Les deux tables sur lesquelles on travaille doivent avoir le même schéma !

- l'union : **UNION** (pour conserver les doublons : **UNION ALL**)

```
SELECT Nom,Prenom
FROM Table_Employes
UNION
SELECT Nom,Prenom
FROM Table_Clients
```

Opérations ensemblistes

Le produit cartésien :

```
SELECT *
FROM NomTable1,NomTable2
WHERE ...
```

Les deux tables sur lesquelles on travaille doivent avoir le même schéma !

- l'union : **UNION** (pour conserver les doublons : **UNION ALL**)

  ```
  SELECT Nom,Prenom
  FROM Table_Employes
  UNION
  SELECT Nom,Prenom
  FROM Table_Clients
  ```

- l'intersection : **INTERSECT**
- la différence ensembliste : **MINUS**

Optimisations

- Evitez d'employer $*$ dans la clause **SELECT**
  Préférez nommer les colonnes une à une
- Evitez d'utiliser les **LIKE**
  Si une fourchette de recherche le permet
- Evitez les fourchettes $<$ et $>$ pour des valeurs discrètes
  Préférez le **BETWEEN**
- Evitez le **IN** avec des valeurs discrètes recouvrantes
  Préférez le **BETWEEN**

Protections d'accès

La gestion utilisateurs

Plusieurs personnes peuvent travailler simultanément sur la même BD, mais elles n'ont pas les mêmes besoin. L'administrateur de la BD définit les permissions qu'il accorde aux utilisateurs :

- accorder des droits : **GRANT**
- retirer des droits : **REVOKE**

La gestion utilisateurs

Plusieurs personnes peuvent travailler simultanément sur la même BD, mais elles n'ont pas les mêmes besoin. L'administrateur de la BD définit les permissions qu'il accorde aux utilisateurs :

- accorder des droits : **GRANT**
- retirer des droits : **REVOKE**

Le créateur de la table obtient tous les droits (**INSERT**,**DELETE**,**UPDATE**,...)

La gestion utilisateurs

Plusieurs personnes peuvent travailler simultanément sur la même BD, mais elles n'ont pas les mêmes besoin. L'administrateur de la BD définit les permissions qu'il accorde aux utilisateurs :

- accorder des droits : **GRANT**
- retirer des droits : **REVOKE**

Le créateur de la table obtient tous les droits (**INSERT**,**DELETE**,**UPDATE**,...)

- la liste d'utilisateurs peut être **PUBLIC**
- la liste des permissions peut être **ALL**

Accorder des droits

- accorder des droits : **GRANT...ON...TO...**
- utilisateur ayant des droits peut accorder ces mêmes droits à d'autres utilisateurs : **WITH GRANT OPTION**

Accorder des droits

- accorder des droits : **GRANT...ON...TO...**
- utilisateur ayant des droits peut accorder ces mêmes droits à d'autres utilisateurs : **WITH GRANT OPTION** (donc plusieurs utilisateurs peuvent accorder des droits au même utilisateur...)

Accorder des droits

- accorder des droits : **GRANT...ON...TO...**
- utilisateur ayant des droits peut accorder ces mêmes droits à d'autres utilisateurs : **WITH GRANT OPTION** (donc plusieurs utilisateurs peuvent accorder des droits au même utilisateur...)

Exemple :

```
GRANT UPDATE(Nom,Prenom)
ON Etudiants
TO Jerome,Francois,Georges
WITH GRANT OPTION;
```

Retirer des droits

- retirer des droits : **REVOKE...ON...FROM...**
- supprimer le droit d'un utilisateur à accorder des permissions à un autre utilisateur : **GRANT OPTION FOR**

Retirer des droits

- retirer des droits : **REVOKE...ON...FROM...**
- supprimer le droit d'un utilisateur à accorder des permissions à un autre utilisateur : **GRANT OPTION FOR**

```
REVOKE
[GRANT OPTION FOR] ListePermissions
ON ListeObjets
FROM ListeUtilisateurs;
```