Construction de BD

Définition de données avec le langage SQL

Dr N. BAME

PLAN

- **Présentation du langage SQL**
- **SQL : Langage de Définition de données**

Structure de la base de données

- Langage de définition de données
- Création de tables
- Contraintes d'intégrité
- Modification de la structure d'une table
- Suppression d'une table
- Création d'une table avec insertion de données

SQL : Structured Query Language

- Langage de **requêtes standard pour les BD relationnelles**
- Développé chez IBM (1970-80)
- Devenu une norme (ANSI/ISO) en 1986
- **Implantations légèrement différentes** selon SGBD
- Langage **déclaratif** : basé sur **l'algèbre relationnelle**

L'évolution des standards SQL

- Début : SQL86
- SQL89 ou **SQL1**
- SQL92 ou **SQL2**
- SQL99 ou **SQL3** (ajout récursivité, triggers, fonctions OO, types binaires,...)
- SQL2003 (ajout manipulation XML, auto-incrément...)
- SQL2008 (ajout des fonctions de fenêtrage, limite du nombre de résultats, ...)

Principaux rôles de SQL

SQL est composé de cinq **grands ensembles d'opérations**:

1) **DDL (LDD)** : Définir et modifier le schéma d'une BD

   CREATE, DROP ALTER

2) **DML (LMD):** Manipuler les données (ajout, suppression, modification)

   INSERT, DELETE, UPDATE

3) **DRL (LID):** Interroger les données

   SELECT

4) **DCL (LCD):** contrôle des accès aux données

   GRANT, REVOKE

5) **TCL (LCT):** contrôler les transactions

   COMMIT, ROLLBACK

SQL : Langage de Définition de données

Structure de la base de données

- Langage de définition de données
- Création de tables
- Contraintes d'intégrité
- Modification de la structure d'une table
- Suppression d'une table
- Création d'une table avec insertion de données

Définition de données

- SQL est un langage permettant de **créer ou de modifier** le ***schéma*** d'une relation ou ***table***.
  - Il permet de **créer, de modifier et de supprimer** les **tables**
  - La base de données doit être créée avant de créer ses relations (tables)
- **SQL est un LDD**
- Création de la base de données

  CREATE DATABASE databaseName;
- Se connecter à la base de données

  USE databaseName;
- Suppression de la base de données

  DROP DATABASE databaseName;

Création de Table : syntaxe simple

- La table est la structure de base contenant les données des utilisateurs.
- L'ordre **CREATE TABLE** permet de créer une table en définissant le nom et le type de chacune des colonnes de la table

```
CREATE TABLE table_name
(attribut1 type1,
attribut2 type2,
…
attributn typen);
```

- ***Table_name*** est le nom que l'on donne à la table,
- ***attribut$_1$, attribut$_2$,.. , attribut$_n$*** sont les noms des attributs (colonnes),
- ***type$_1$, type$_2$,.. , type$_n$*** sont les types des données qui seront contenues dans les colonnes.

Types de données : numériques

Nombres entiers :

- ***BIT*** *: permet de ranger une valeur booléenne (un bit) ; soit 0 ou 1*
- ***TINYINT*** : de 0 à 255 ; soit 2^8
- ***SMALLINT*** : de -32.768 à 32.767 ; 2^15
- **INT (INTEGER):** de -2,147,483,648 à 2,147,483,647; soit 2^31
- ***BIGINT :*** *de* -9,223,372,036,854,775,808 à 9,223,372,036,854,775,807 ; soit 2^63

Nombres décimaux avec un nombre fixe de décimales : **NUMERIC**, **DECIMAL**

- la norme impose à NUMERIC d'être implanté avec exactement le nombre de décimales indiqué alors que l'implantation de DECIMAL peut avoir plus de décimales) : *DECIMAL(p, d)* correspond à des nombres décimaux qui ont *p* chiffres significatifs et *d* chiffres après la virgule ; NUMERIC a la même syntaxe.
- **Money :** *de* -9,223,372,036,854,775,808 à 9,223,372,036,854,775,807 ; soit 2^63
- **Smallmoney :** de (-214,748.3648) à (214,748.3647) , 2^31
- **REAL** (simple précision, avec au moins *7 chiffres* significatifs),
- **DOUBLE** ou **FLOAT** (double précision, avec au moins *15 chiffres* significatifs).

Types de données

- **chaînes de caractères**

  CHAR(n) longueur fixée à n,

  VARCHAR(n) longueur maximale fixée à n

  text : longueur variable illimitée

  Les constantes chaînes de caractères sont entourées par des apostrophes ('). Si la chaîne contient une apostrophe, celle-ci doit être doublée.

  Exemple : 'aujourd''hui'.

- **Dates et heures**

  DATE réserve 2 chiffres pour le mois et le jour et 4 pour l'année ;

  TIME pour les heures, minutes et secondes (les secondes peuvent comporter un certain nombre de décimales) ;

  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).

- **Type booléen (*BOOLEAN*)**

  Seules deux valeurs possible: **TRUE** ou **FALSE**

  Un troisième état, (inconnu), est représenté par la valeur NULL de SQL.

Exemples de création de table

```
CREATE TABLE Module
(
    code varchar(10),
    intitule varchar(20),
    niveau char(2),
    suit varchar(10)
)
```

```
CREATE TABLE Compte
(
    numero NUMERIC,
    titulaire VARCHAR (50),
    lieu VARCHAR(100),
    Ouverture DATE
)
```

Création de Table : syntaxe complète

- On peut ajouter après la description d'une colonne l'option **NOT NULL** qui **interdira** que cette colonne contienne la valeur **NULL**.
- On peut donner une **valeur par défaut** pour une colonne si la colonne n'est pas renseignée.
- On peut aussi ajouter des **contraintes d'intégrité** portant sur une ou plusieurs colonnes de la table

CREATE TABLE table_name
(attribut$_1$ *type*$_1$ **[DEFAULT** defaultVal1**]** **[NOT NULL]** [constraintDefinition$_1$],
attribut$_2$ *type*$_2$ **[DEFAULT** defaultVal$_2$**]** **[NOT NULL]** [constraintDefinition$_2$],
…
attribut$_n$ *type*$_n$ **[DEFAULT** defaultVal$_n$**]** **[NOT NULL]** [constraintDefinition$_n$],
[constraintDefinition]
);

Exemple

```
CREATE TABLE article
(
        ref             VARCHAR(10) primary key,
        nom             VARCHAR(30) NOT NULL,
        prix            DECIMAL(9,2),
        datemaj         DATE DEFAULT CURRENT_DATE
);
```

Les contraintes d'intégrité

- Dans la définition d'une table, on peut indiquer des **contraintes d'intégrité portant sur une ou plusieurs colonnes**.
- Les contraintes possibles sont :

  PRIMARY KEY, UNIQUE, FOREIGN KEY...REFERENCES, CHECK
- Toute définition de table doit comporter **au moins une contrainte de type PRIMARY KEY**.
- Chaque contrainte doit être nommée

  ce qui permettra de la désigner par un ordre **ALTER TABLE**

  Le nom d'une contrainte doit être unique parmi toutes les contraintes de toutes les tables de la base de données.

[CONSTRAINT constraint_name] constraint_definition

- Il existe des contraintes :
  - **sur une colonne :** la contrainte porte sur une seule colonne. Elle suit la définition de la colonne dans un ordre CREATE TABLE (pas possible dans un ordre ALTER TABLE).
  - **sur une table :** la contrainte porte sur une ou plusieurs colonnes. Elles se place au même niveau que les définitions des colonnes dans un ordre CREATE TABLE ou ALTER TABLE.

Types de contraintes

- **Contrainte de clé primaire**

définit la **clé primaire** de la table. Aucune des colonnes qui forment cette clé ne doit avoir une valeur NULL.

- Pour une contrainte sur une **table**:
  [CONSTRAINT constraint_name] **PRIMARY KEY (colonne1, colonne2,…)**
- Pour une contrainte sur une **colonne**:
  [CONSTRAINT constraint_name] **PRIMARY KEY**

- **Contrainte d'unicité**

interdit qu'une colonne (ou la concaténation de plusieurs colonnes) contienne deux **valeurs identiques.**

- Pour une contrainte sur une **table**:
  [CONSTRAINT constraint_name] **UNIQUE (colonne1, colonne2,…)**
- Pour une contrainte sur une **colonne**:
  [CONSTRAINT constraint_name] **UNIQUE**

Cette contrainte UNIQUE convient à des clés candidates. Cependant une colonne UNIQUE peut avoir des valeurs NULL et une contrainte UNIQUE ne correspond donc pas toujours à un identificateur.

Types de contraintes

Contrainte de clé étrangère

- Pour une contrainte sur une table:
  [CONSTRAINT constraint_name] ***FOREIGN KEY (colonne1, colonne2,…)***
  REFERENCES tableref [(col1, col2,…)]
  [ON DELETE CASCASDE]
- Pour une contrainte sur une colonne:
  [CONSTRAINT constraint_name] **REFERENCES *tableref [(col1)]***
  [ON DELETE CASCASDE]

indique que la concaténation de *colonne1, colonne2,…* (ou la colonne que l'on définit pour une contrainte sur une colonne) est une clé étrangère qui fait référence à la concaténation des colonnes *col1, col2,…* de la table *tableref* (contrainte d'intégrité référentielle). Si aucune colonne de *tableref* n'est indiquée, c'est la clé primaire de *tableref* qui est prise par défaut.

Cette contrainte ne permettra pas d'insérer une ligne de la table si la table *tableref* ne contient aucune ligne dont la concaténation des valeurs de *col1, col2,…* est égale à la concaténation des valeurs de *colonne1, colonne2,… col1, col2,…* doivent avoir la contrainte PRIMARY KEY ou UNIQUE.

Ceci implique qu'une valeur de colonne1, colonne2,… va référencer une et une seule ligne de tableref.

Types de contraintes

Contrainte de clé étrangère

- L'option « ON DELETE CASCADE » indique que la suppression d'une ligne de *tableref* va entraîner automatiquement la suppression des lignes qui la référencent dans la table. Si cette option n'est pas indiquée, il est impossible de supprimer des lignes de *tableref* qui sont référencées par des lignes de la table.

- A la place de «ON DELETE CASCADE» on peut donner l'option «ON DELETE SET NULL». Dans ce cas, la clé étrangère sera mise à NULL si la ligne qu'elle référence dans *tableref* est supprimée.

- Ils existe d'autres options

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

  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 modifie.

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

Types de contraintes

- **Contrainte « CHECK »**

[CONSTRAINT constraint_name] **CHECK** ***condition***

  - donne une condition que les colonnes de chaque ligne devront vérifier. On peut ainsi indiquer des **contraintes d'intégrité de domaines**.

  - Cette contrainte peut être une contrainte de colonne ou de table.
    - Si c'est une contrainte de colonne, elle ne doit porter que sur la colonne en question.

Ajouter, supprimer ou renommer une contrainte

- Des contraintes d'intégrité peuvent être ajoutées ou supprimées par la commande ***ALTER TABLE***.

- On peut aussi modifier l'état de contraintes par ***MODIFY CONSTRAINT***.

- On ne peut ajouter que des contraintes de table.
  - Si on veut ajouter (ou modifier) une contrainte de colonne, il faut modifier la colonne

```
ALTER TABLE EMP
DROP CONSTRAINT NOM_UNIQUE
ADD CONSTRAINT SAL_MIN CHECK (SAL > 1000)
RENAME CONSTRAINT NOM1 TO NOM2
MODIFY CONSTRAINT SAL_MIN DISABLE
```

Désactiver des contraintes

- Les contraintes d'intégrité sont parfois gênantes. On peut vouloir les enlever pour améliorer les performances durant le chargement d'une grande quantité de données dans la base.
- Pour cela, certains SGBD comme oracle fournit la commande ***ALTER TABLE ... DISABLE/ENABLE***.

```
ALTER TABLE EMP
DISABLE CONSTRAINT NOM_UNIQUE
```

Modifier la structure d’une table

- On peut modifier dynamiquement la définition d'une table grâce a la commande **ALTER TABLE**.
- Les modifications sont possibles :
  - ajout d'une colonne et
  - modification d'une colonne existante.
  - Suppression d'une colonne existante.

Ajout d'une colonne - ADD

ALTER TABLE *table_name*
ADD ***(col_1 $type_1$, col_2 $type_2$, ...)***

- permet d'ajouter une ou plusieurs colonnes à une ***table existante***.
  - Les types possibles sont les mêmes que ceux décrits avec la commande CREATE TABLE.
  - Les parenthèses ne sont pas nécessaires si on n'a joute qu'une seule colonne.
  - L'attribut 'NOT NULL' peut être spécifié seulement si la table est vide
    - si la table contient déjà des lignes, la nouvelle colonne sera nulle dans ces lignes existantes et donc la condition 'NOT NULL' ne pourra être satisfaite.
  - Il est possible de définir des contraintes de colonne.

```
ALTER TABLE personne
ADD  (email_valide  CHAR(1)
CONSTRAINT personne_email_valide  CHECK(email_valide  in  ('o',  'n')))
```

Modification d'une colonne - MODIFY

ALTER TABLE *table_name*
MODIFY ***(col$_1$ type$_1$, col$_2$ type$_2$, ...)***

- *col$_1$, col$_2$...* sont les noms des colonnes que l'on veut modifier.
  - Elles doivent bien sûr déjà exister dans la table.
- *type1, type2,...* sont les nouveaux types que l'on désire attribuer aux colonnes.
- Il est possible de modifier la définition d'une colonne, à condition que
  - la colonne ne contiennent que des valeurs NULL ou que
  - la nouvelle définition soit compatible avec le contenu de la colonne :
    - on ne peut pas diminuer la taille maximale d'une colonne.
    - on ne peut spécifier 'NOT NULL' que si la colonne ne contient pas de valeur nulle.
    - Mais il est toujours possible d'augmenter la taille maximale d'une colonne, tant qu'on ne dépasse pas les limites propres à SQL, et on p eut dans tous les cas spécifier 'NULL' pour autoriser les valeurs nulles.

Modification d'une colonne - MODIFY

- On peut donner une contrainte de colonne dans la nouvelle définition de la colonne.

```
ALTER TABLE personne
MODIFY (sexe  char(1) CONSTRAINT  personne_sexe_ck  CHECK (sexe  in  ('m',  'f')))
```

Renommer une colonne - RENAME COLUMN

```
ALTER TABLE table
RENAME COLUMN ancien_nom TO nouveau_nom
```

Suppression d'une colonne - DROP COLUMN

```
ALTER TABLE table
DROP COLUMN col_name;
```

Ou

```
ALTER TABLE table
DROP col_name;
```

La colonne supprimée ne doit pas être référencée par une clé étrangère ou être utilisée par un index.

Renommer, supprimer une table

- Renommer une table

```
ALTER TABLE table_name
RENAME TO new_name
```

- Supprimer une table

```
DROP TABLE table_name
```

  - permet de supprimer une table : les données (lignes) de la table et la définition elle-même (structure) de la table sont détruites. L'espace occupé par la table est libéré.

```
TRUNCATE TABLE table_name
```

  - permet de supprimer toutes les données d'une table sans supprimer la table en elle-même. En d'autres mots, cela permet de purger la tablesi la table est référencée par une contrainte d'intégrité référentielle.

```
DROP TABLE table_name CASCADE CONSTRAINTS;
```

Création d'une table avec Insertion de données

- **CREATE TABLE AS** permet d'insérer pendant la création de la table des lignes venant d'autres tables :

```
CREATE TABLE table (col type.....)
AS SELECT .....
```

- On peut aussi spécifier des contraintes d'intégrité de colonne ou de table.

```
CREATE TABLE EtudiantMaster
(matricule INTEGER,
Nom VARCHAR(20),
Prenom VARCHAR(20))
AS SELECT matricule, nom, prenom FROM Etudiant WHERE
CodeClasse LIKE 'M%';
```