SQL

Introduction

SQL - Structured Query Language

- Langage informatique normalisé
- N'est pas un langage procédural
- Créé par IBM en 1974
- Normalisé depuis 1986
  - Recommandation de l'ANSI
  - Norme ISO/CEI 9075 - Technologies de l'information - Langages de base de données – SQL

Base De Données Relationnelle

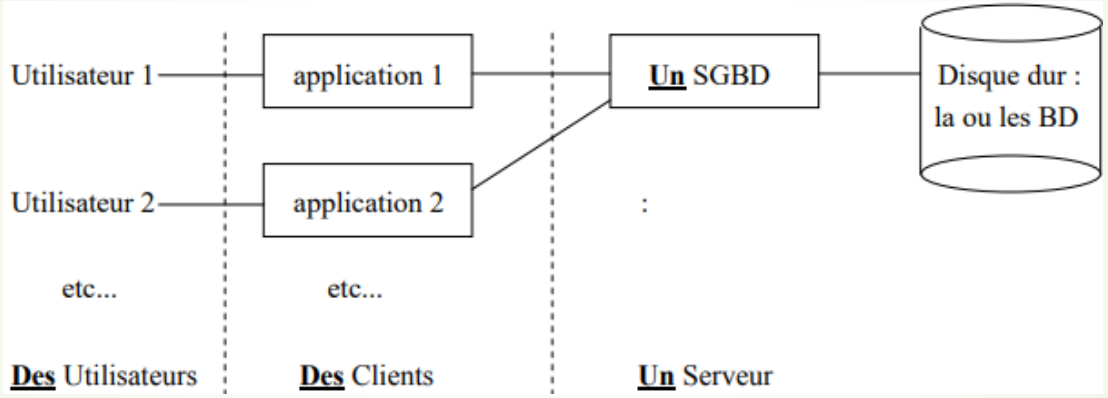
- 3 objectifs :
  - Garantir l'intégrité des données
    - Eviter l'altération des données (usure, pannes, erreurs, malveillance)
    - Eviter l'incohérence des données
      - La duplication des données : exemple 2 adresses différentes pour un utilisateur
      - Les valeurs aberrantes : exemple Age < 0
  - Garantir l'indépendance entre données et traitements
    - L'information correspondant à une donnée doit être compréhensible indépendamment
    - Si une donnée est calculée, on évitera de la stocker en BDD
  - Traitements rationalisés
    - Une fois les données définies, les traitements consistent principalement à ajouter, modifier, supprimer et consulter des données

SGBD – Système de Gestion de Base de Données

- Ensemble de logiciels faisant l'interface entre l'utilisateur et la BDD
- Objectifs :
  - Stockage et requêtage des données
  - Multi-utilisateurs => Gestion des droits d'accès, des collisions
  - Limiter la redondance (duplication de données) => Intégrité des données

SGBD – Système de Gestion de Base de Données

- Architecture
  - Le plus souvent on utilise une architecture client/serveur



  - Une BDD est stockée sur 1 ou plusieurs disques durs
  - 1 seul SGBD par BDD
  - Plusieurs applications / API (clients) peuvent communiquer avec le SGBD

SGBD – Système de Gestion de Base de Données

- L'objectif majeur du SGBD étant d'assurer l'intégrité, voici quelques moyens mis en œuvre
  - Conformité au modèle : Vérifier que les données saisies soient cohérentes
  - Accès concurrents : Eviter les incohérences dues à des modifications multiples au même moment (transactions)
  - Gestion sur panne : Garantir le maintien de la cohérence même quand une panne intervient
  - Autorisations d'accès

Modèle relationnel

- Généralités
  - Inventé par Codd (IBM) en 1970
  - Permet de fabriquer la BDD et de l'interroger
  - Enregistrement des données sous forme de tableau à 2 dimensions
- Vocabulaire
  - Relation = table = classe = ensemble = collection
  - Tuple = ligne du tableau = enregistrement = n-uplet
  - Attribut = colonne du tableau = propriété

SQL

Administration d'une BDD

Création de BDD

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name [create_option]…
create_option: [DEFAULT] {
    CHARACTER SET [=] charset_name |
    COLLATE [=] collation_name |
    ENCRYPTION [=] {'Y' | 'N'}
}
```

IF NOT EXISTS : Si cette option est présente, cette commande ne lèvera pas d'exception si la BDD existe déjà. Il n'y fera aucune modification.

create_option permet de définir des paramètres de la BDD.

Pour plus d'informations : https://dev.mysql.com/doc/refman/8.0/en/create-database.html

Suppression et modification de BDD

- SUPPRESSION

```
DROP {DATABASE | SCHEMA} [IF EXISTS] db_name;
```

- MODIFICATION

```
ALTER {DATABASE | SCHEMA} [db_name] alter_option ...
alter_option: { [DEFAULT] CHARACTER SET [=] charset_name |
    [DEFAULT] COLLATE [=] collation_name |
    [DEFAULT] ENCRYPTION [=] {'Y' | 'N'} |
    READ ONLY [=] {DEFAULT | 0 | 1} }
```

- SELECTIONNER UNE BDD : Nécessaire pour pouvoir effectuer des opérations (création ou consultation)

```
USE db_name;
```

- DIVERS

```
SHOW DATABASES;
```

Gestion des droits et utilisateurs

- Sécurité
  - Limiter au strict nécessaire les droits des utilisateurs exposés (application web)
  - Utiliser des mots de passe forts et uniques
  - Maintenir ses logiciels à jour

Gestion des droits et utilisateurs

- Création d'un utilisateur

  CREATE USER 'alice'@'localhost' IDENTIFIED BY 'P@ssw0rd';

- Assigner les privileges

  GRANT type_of_permission ON database_name.table_name TO 'username'@'host';

  Example :

  GRANT CREATE, ALTER, DROP, INSERT, UPDATE, DELETE, SELECT, REFERENCES, RELOAD

  ON *.* TO 'alice'@'localhost' WITH GRANT OPTION;

- Révoquer un privilege

  REVOKE type_of_permission ON database_name.table_name FROM 'username'@'host'

Gestion des droits et utilisateurs

- Appliquer les modifications:

  FLUSH PRIVILEGES;

- Afficher les privileges d'un utilisateur

  SHOW GRANTS FOR 'username'@'host';

- Supprimer un utilisateur

  DROP USER 'username'@'localhost';

Exercice

- Coder un script pour :

  Créer un autre utilisateur ‘bob’

  Créer une BDD pour bob lui assignant tous les privileges

- Ressource :

  https://www.digitalocean.com/community/tutorials/how-to-create-a-new-user-and-grant-permissions-in-mysql

SQL

Structure des données

Les Tables

- Les tables sont les objets qui contiennent toutes les données d'une BDD.

- Données organisées dans un tableau à deux dimensions
  - Les lignes sont des tuples ou n-uplets
  - Les colonnes sont des attributs

Conception : Les Formes Normales

- Il existe 9 formes normales mais les 3 premières sont les plus fondamentales

- Forme Normale 1 (1FN)

  Conditions :

  Tous les attributs doivent posséder une valeur atomique (Valeur non subdivisible)

  Par exemple, un n° de sécu est non atomique (composé de parties distinctes : sexe, date de naissance, identifiant INSEE de la commune, …)

  En revanche une date est atomique

Conception : Les Formes Normales

- Forme Normale 2 (2FN)

  Conditions :

  Respecter la 1FN

  Un attribut non identifiant ne dépend pas d'une partie de l'identifiant mais de tout l'identifiant.

- Forme Normale 3 (3FN)

  Conditions :

  Respecter la 2FN

  Un attribut non identifiant ne dépend pas d'un ou plusieurs attributs ne participant pas à l'identifiant.

Les types de données

- Numériques exacts :
  - Entiers signés :
    - INTEGER 4 octets => -2147483648 à +2147483647
    - SMALLINT (2 octets), BIGINT (8 octets)
  - Décimaux signés :
    - NUMERIC, DECIMAL
- Numériques approximatifs
  - REAL, DOUBLE PRECISION, FLOAT
- Chaînes de caractères
  - Longueur fixe : CHAR
  - Longueur variable : VARCHAR
  - Objets de type caractère : CLOB

Les types de données

- Chaînes binaires
  - Nombre d'octets fixe : BINARY
  - Longueur variable : VARBINARY
  - Binary Large Object : BLOB
- Booléens
  - 3 valeurs : true, false, unknown
- Dates et heures
  - DATE
  - TIME/TIMESTAMP WITH/WITHOUT TIMEZONE
- DIVERS
  - XML, ARRAY, INTERVAL

Opérations sur les tables

- Création

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
        (create_definition,...)
        [table_options]
        [partition_options]
```

Exemple :

```
CREATE TABLE users (
        id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
        name VARCHAR(100),
        email VARCHAR(255) NOT NULL,
        birthdate DATE
)
```

https://dev.mysql.com/doc/refman/8.0/en/create-table.html

Opérations sur les tables

- Modification :
  - Ajouter/Modifier/Supprimer un attribut
  - Ajouter/Modifier/Supprimer une contrainte
  - Modifier des propriétés de la table

```
ALTER TABLE tbl_name
    [alter_option [, alter_option] ...]
    [partition_options]
```

Exemple :

```
ALTER TABLE users
    ADD firstname VARCHAR(100) AFTER name;
```

https://dev.mysql.com/doc/refman/8.0/en/alter-table.html

Indexes

- Dans chaque table, on peut définir un ou plusieurs indexes.
- Un ROWID est créé pour chaque enregistrement
- Avantage : Requêtes plus rapides
- Inconvénient : Peuvent surcharger le système donc à utiliser avec précaution
- Exemple :

  Création d'un index : Notez que l'on peut le définir sur une ou plusieurs colonnes

  ```
  CREATE INDEX index_name ON table_name ( column1, column2, ...);
  ```

  Suppression d'un index

  ```
  ALTER TABLE table_name DROP INDEX index_name;
  ```

Contraintes

Les contraintes sont des règles appliquées aux attributs d'une table.

Elles permettent d'améliorer l'intégrité des données de la base.

- Les contraintes les plus communes sont :
  - **NOT NULL** : Contraint à avoir une valeur différente de NULL.
  - **DEFAULT** : Fournit une valeur par défaut à la colonne.
  - **UNIQUE** : Ne permet pas que 2 enregistrements aient la même valeur pour une colonne (email)
  - **CHECK** : Active une condition pour qu'un enregistrement soit valide.
  - **INDEX**
  - **PRIMARY KEY** : La clé primaire permet d'identifier de manière unique chaque enregistrement. Une clé primaire peut être composée, elle fait dans ce cas référence à plusieurs attributs de la table.
  - **FOREIGN KEY** : Une clé étrangère permet de relier 2 tables.

Contraintes - Clés

- Clé primaire :

  Une clé primaire permet d'identifier chaque enregistrement d'une table.

  Par définition, elle doit être UNIQUE et NOT NULL.

  Une clé primaire peut être simple (définie par un seul attribut) ou composée de plusieurs attributs.

- Clé étrangère :

  Une clé étrangère est une contrainte qui permet d'associer plusieurs tables entre elles tout en assurant l'intégrité des données.

  Une clé étrangère fait référence à une clé primaire d'une autre table.

  A noter :

  - Il est impossible de faire référence à une clé primaire si celle-ci n'existe pas dans la BDD.
  - Il n'est par défaut pas possible de supprimer un enregistrement si sa clé primaire est référencée comme clé étrangère dans une autre table.

Contraintes – ON DELETE / ON UPDATE

- Clé étrangère :

  Si un enregistrement est référencé dans une autre table, le SGBD empêchera la modification ou suppression de la clé primaire.

  Si l'on souhaite forcer la modification / suppression, il est nécessaire de le renseigner lors de la création de la clé secondaire.

  Pour cela on doit utiliser les instructions ON DELETE et ON UPDATE

  - ON DELETE : A la suppression, effectue une action sur tous les enregistrements enfants.
  - ON UPDATE : A la modification, reporte le changement sur tous les enregistrements enfants.

  L'usage le plus courant est d'utiliser ON DELETE CASCADE ou ON UPDATE CASCADE.

  Toutefois, c'est un comportement assez destructeur ce qui n'est pas toujours souhaitable.

  Dans ce cas, on peut utiliser des alternatives, telles que :

  ON DELETE | UPDATE [SET NULL | SET DEFAULT | NO ACTION | CASCADE]

Contraintes - Exemple

```
CREATE TABLE employes(
        id      INT     NOT NULL,
        nom     VARCHAR (100)   NOT NULL,
        email   VARCHAR (100)   NOT NULL UNIQUE,
        age     INT  NOT NULL CHECK (age >= 18),
        salaire  DECIMAL (18, 2) DEFAULT 3000.00,
        PRIMARY KEY (id)
);
CREATE TABLE conges (
        id      INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
        date_debut  DATE    NOT NULL,
        date_fin    DATE    NOT NULL,
        id_employe INT,
        FOREIGN KEY (id_employe) REFERENCES employes(id) ON DELETE CASCADE
        -- Pour pouvoir nommer la contrainte, utiliser la syntaxe suivante
        CONSTRAINT fk_id_employe FOREIGN KEY (id_employe) REFERENCES employes(id)
);
```

Contraintes - Suppression

- En ciblant la contrainte :

  ALTER TABLE table_name

  ALTER COLUMN column_name DROP CONSTRAINT;

  Ex : ALTER TABLE employe

  ALTER COLUMN salaire DROP DEFAULT;

- Si la contrainte a un alias

  ALTER TABLE table_name DROP CONSTRAINT constraint_alias;

Associations

Lorsque l'on modélise notre BDD, on va identifier différents types d'associations qui vont lier les tables entre elles.

Bien concevoir ses associations est indispensable pour satisfaire les conditions des formes normales et assurer l'intégrité des données.

On dénombre 5 types de relations :

- One-to-One
- One-to-Many / Many-to-One
- Many-to-Many
- Self-Reference (association récursive)

Associations – One-to-One

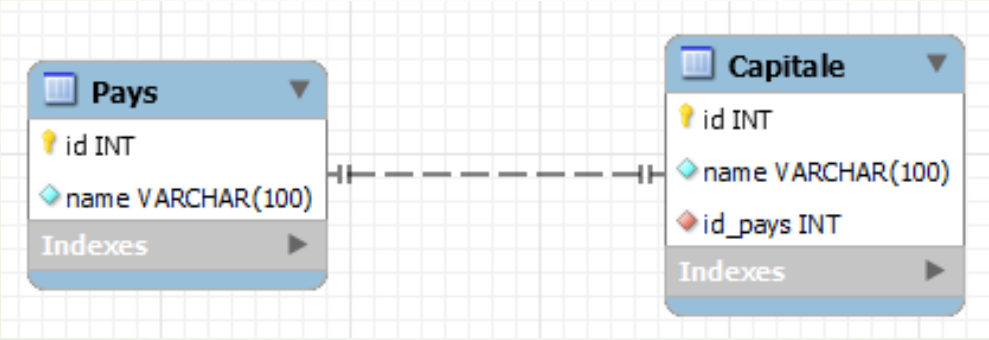
Permet d'associer un et un seul enregistrement d'une table T1 à un et un seul enregistrement d'une table T2.

Implémentée par l'utilisation d'une clé étrangère.

- Veiller à ajouter des contrôles (code et/ou BDD) pour assurer l'intégrité des données (ex : contrainte d'unicité sur la clé étrangère)

Exemples :

- Citoyen <-> Passeport
- Capitale <-> Pays



Associations – One-to-Many et Many-to-One

- One-to-Many :

  Associe un enregistrement d'une table T1 à de multiples enregistrements d'une table T2

- Many-to-One :

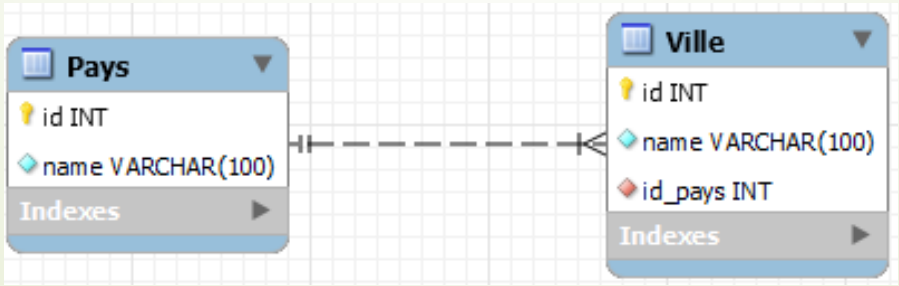
  Associe de multiples enregistrements d'une table T1 à un unique enregistrement d'une table T2

Implémentée par l'utilisation d'une clé étrangère.

- Veiller à ajouter des contrôles (code et/ou BDD) pour assurer l'intégrité des données.

Exemple :

- Pays <-> Ville : Un pays possède plusieurs villes, une ville n'appartient qu'à un seul pays



Associations – Many-to-Many

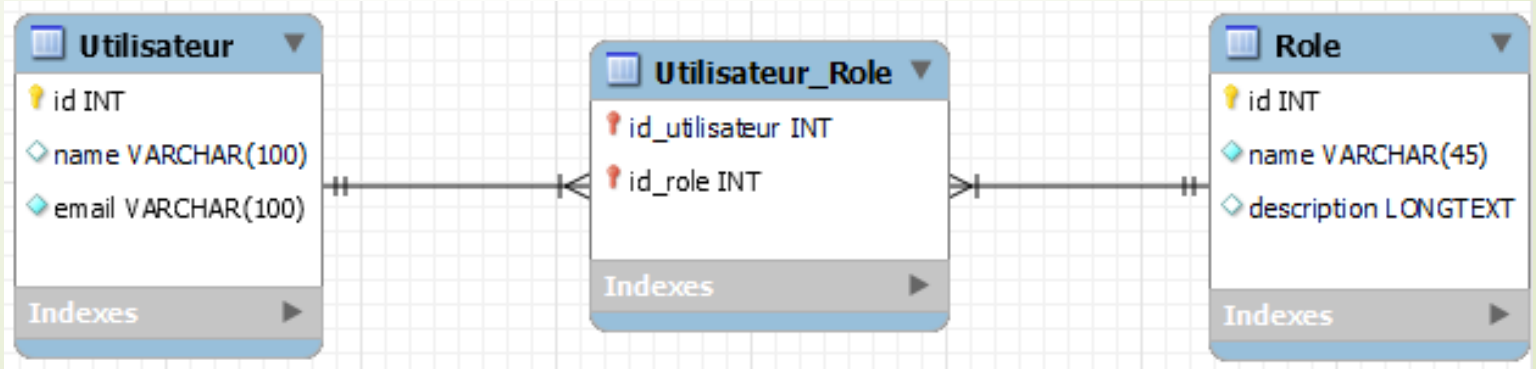
Associe un enregistrement d'une table T1 à de multiples enregistrements de la table T2 et vice versa.

Pour implémenter une association Many-to-Many, il est nécessaire de passer par une table d'association.

La table d'association peut aussi être porteuse de donnée, par exemple ci-dessous, la table Utilisateur_Role pourrait indiquer la date d'expiration du rôle.

Exemple :

- Utilisateur <-> Rôle : un utilisateur peut avoir plusieurs rôles et un rôle peut être possédé par plusieurs utilisateurs.



Associations – Self-Reference

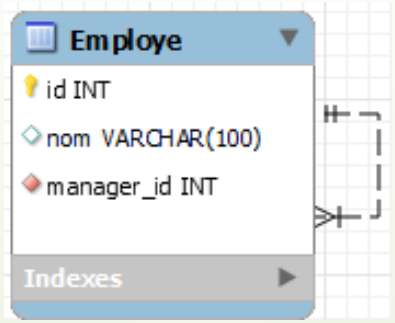
Une association Self-Reference permet d'associer un enregistrement d'une table T à un autre enregistrement d'une table T.

L'association sera souvent une association One-to-Many ou Many-to-One.

C'est notamment utilisé lorsque l'on souhaite gérer une arborescence.

Exemples :

- Catégories <-> Catégorie enfant
- Employé <-> Manager



SQL

Ressources

Documentation

- Site officiel :

  https://www.mysql.com/
- Documentation officielle :

  https://dev.mysql.com/doc/refman/8.0/en/
- Tuto W3Schools :

  https://www.w3schools.com/sql/default.asp
- MySQL Tutorial :

  https://www.mysqltutorial.org/
- Slide sur l'optimisation :

  https://www.slideshare.net/ZendCon/joinfu-the-art-of-sql-tuning-for-mysql-presentation

Outils

- MySQL Workbench :

  https://www.mysql.com/products/workbench/index.html

- HeidiSQL :

  https://www.heidisql.com/

SQL

Travaux Pratiques