R1.05 — Introduction aux bases de données

Chapitre 1 – Lire et écrire dans une base de données

Clément Pagès
clement.pages@u-pec.fr

Université Paris-Est Créteil
BUT informatique 1re année

2025-2026

Sommaire

Structured Query Langage (Langage de requêtes structurées)

Rapide historique

- 1971 : Langage `ALPHA`, balbutiements.
- 1974 : Ingres (*Interactive graphics retrieval system*) et Quel (*Query English Language*).
- 1979 : première commercialisation d'un SGBDD : Oracle v2.
- 1986 : première normalisation.
- 1986-2011 : principales évolutions.
- Depuis : SQL3, les technologies sont globalement stables.

Structured Query Language (Langage de requêtes structurées)

Spécifications

- Création d'une base de données
- Définition des tables et des colonnes
- Définition des types : numériques, logiques, textuels et temporels
- Définition des contraintes
- Définition des vues
- Définition des droits d'accès
- Opérations sur la strcuture de la base
- Manipulation des données

Definition

Les instructions données à un SGBDD s'appellent des *requêtes*.

Sommaire

Création de base de données et de tables

```SQL
CREATE DATABASE cooperative; -- Crée une base de données

CREATE TABLE personnes( -- Crée une table dans la base
    numero INTEGER NOT NULL,
    nom VARCHAR(40) NOT NULL,
    ville VARCHAR(40) NOT NULL,
    telephone VARCHAR(14)
); -- Point-virgule en fin requête

DROP TABLE IF EXISTS personnes; -- Suppression de la table
```

Création de base de données et de tables

```sql
CREATE DATABASE cooperative; -- Crée une base de données

CREATE TABLE personnes( -- Crée une table dans la base
    numero INTEGER NOT NULL,
    nom VARCHAR(40) NOT NULL,
    ville VARCHAR(40) NOT NULL,
    telephone VARCHAR(14)
); -- Point-virgule en fin requête

DROP TABLE IF EXISTS personnes; -- Suppression de la table
```

SQL

Definition

Les champs présents dans une table s'appellent des attributs. Les attributs ont un type et éventuellement une taille.

Écriture de données

Insertion de données

```SQL
INSERT INTO personnes(numero, nom, ville)
VALUES
    (1, "Ada LOVELACE", "Londres"),
    (2, "Alan TURING", "Wilmslow");
```

Écriture de données

Insertion de données

```SQL
INSERT INTO personnes(numero, nom, ville)
VALUES
    (1, "Ada LOVELACE", "Londres"),
    (2, "Alan TURING", "Wilmslow");
```

Modification de données

```SQL
UPDATE personnes
SET ville = "Londres"
WHERE nom = "Alan TURING"; -- Condition sur les lignes à modifier
```

Écriture de données

Insertion de données

```SQL
INSERT INTO personnes(numero, nom, ville)
VALUES
    (1, "Ada LOVELACE", "Londres"),
    (2, "Alan TURING", "Wilmslow");
```

Modification de données

```SQL
UPDATE personnes
SET ville = "Londres"
WHERE nom = "Alan TURING"; -- Condition sur les lignes à modifier
```

Suppression de données

```SQL
DELETE FROM personnes
WHERE numero = 2;
```

Sommaire

Principe

- On peut ajouter des contraintes sur les tables ou attributs, pour garantir l'intégrité des données.
- Exemple attribut obligatoire : `ville VARCHAR(40) NOT NULL`.
- Si la contrainte n'est pas vérifiée, la requête échoue.

Principe

- On peut ajouter des contraintes sur les tables ou attributs, pour garantir l'intégrité des données.
- Exemple attribut obligatoire : `ville VARCHAR(40) NOT NULL`.
- Si la contrainte n'est pas vérifiée, la requête échoue.

SQL
```sql
INSERT INTO personnes(numero, nom)
VALUES (3, "Al-Khwârizmî",);
```

Console
```
SQL Error [19]: [SQLITE_CONSTRAINT_NOTNULL] A NOT NULL
constraint failed (NOT NULL constraint failed: personnes.ville)
```

Les clés primaires

- Une clé primaire permet d'identifier chaque ligne d'une table de manière unique.

Les clés primaires

- Une clé primaire permet d'identifier chaque ligne d'une table de manière unique.
- Exemples : numéro de Sécurité sociale, ISBN (livres), numéro de commande, etc.

Les clés primaires

- Une clé primaire permet d'identifier chaque ligne d'une table de manière unique.
- Exemples : numéro de Sécurité sociale, ISBN (livres), numéro de commande, etc.
- Une clé primaire est identifiée avec la contrainte `PRIMARY KEY`.
- Il ne peut pas y avoir deux lignes ayant la même clé primaire.

Les clés primaires

- Une clé primaire permet d'identifier chaque ligne d'une table de manière unique.
- Exemples : numéro de Sécurité sociale, ISBN (livres), numéro de commande, etc.
- Une clé primaire est identifiée avec la contrainte `PRIMARY KEY`.
- Il ne peut pas y avoir deux lignes ayant la même clé primaire.

Exemple

SQL

```sql
CREATE TABLE personnes(
    numero INTEGER NOT NULL PRIMARY KEY, -- Clé primaire
    nom VARCHAR(40) NOT NULL,
    ville VARCHAR(40) NOT NULL,
    telephone VARCHAR(14)
);
```

Différentes sortes de clés primaires

Clés primaires attributs

Clé primaire naturelle

Un attribut apparait spontanément pour être la clé primaire (ISBN, numéro de Sécurité sociale, code postal, etc.)

Différentes sortes de clés primaires

Clés primaires attributs

Clé primaire naturelle

Un attribut apparait spontanément pour être la clé primaire (ISBN, numéro de Sécurité sociale, code postal, etc.)

Clé primaire autoincrement

Il n'y a pas de clé primaire naturelle.

- On prévoit un attribut pour jouer le rôle d'identifiant.
- La clé sera alors en incrément automatique.

Différentes sortes de clés primaires

Clés primaires attributs

Clé primaire naturelle

Un attribut apparait spontanément pour être la clé primaire (ISBN, numéro de Sécurité sociale, code postal, etc.)

Clé primaire autoincrement

Il n'y a pas de clé primaire naturelle.

- On prévoit un attribut pour jouer le rôle d'identifiant.
- La clé sera alors en incrément automatique.

Exemple

```sql
CREATE TABLE personnes(
    numero INTEGER AUTOINCREMENT PRIMARY KEY, -- Clé primaire
    nom VARCHAR(40) NOT NULL,
    ville VARCHAR(40) NOT NULL
);
```

Différentes sortes de clés primaires

Clés primaires composites

- On a besoin de plusieurs attributs pour former la clé primaire.
- Il n'y a pas d'incrément automatique.

Différentes sortes de clés primaires

Clés primaires composites

- On a besoin de plusieurs attributs pour former la clé primaire.
- Il n'y a pas d'incrément automatique.

Exemple : notes des étudiant·es

```SQL
CREATE TABLE etudiants_notes(
    etudiant_numero INTEGER NOT NULL,
    ressource VARCHAR(4) NOT NULL, -- Code de la ressource
    note DECIMAL (3,1) -- 3 chiffres dont 1 après la virgule,
    PRIMARY KEY (etudiant_numero, etudiants_notes)
);
```

Différentes sortes de clés primaires

Clés primaires composites

- On a besoin de plusieurs attributs pour former la clé primaire.
- Il n'y a pas d'incrément automatique.

Exemple : notes des étudiant·es

```SQL
CREATE TABLE etudiants_notes(
    etudiant_numero INTEGER NOT NULL,
    ressource VARCHAR(4) NOT NULL, -- Code de la ressource
    note DECIMAL (3,1) -- 3 chiffres dont 1 après la virgule,
    PRIMARY KEY (etudiant_numero, etudiants_notes)
);
```

- Chaque étudiant·e apparait une fois par ressource.
- Chaque ressource apparait une fois par étudiant·e.
- Chaque couple (`etudiant`, `ressource`) est unique.

Sommaire

La directive SELECT

Afficher toutes les lignes

```sql
/* Renvoie toutes les lignes de la table personnes */
SELECT * -- Tous les attributs
FROM personnnes;
```

La directive SELECT

Afficher toutes les lignes

```sql
/* Renvoie toutes les lignes de la table personnes */
SELECT * -- Tous les attributs
FROM personnnes;
```

Filtrer les données avec WHERE

```sql
SELECT numero, nom, telephone -- Choix des attributs
FROM personnnes
WHERE ville="Vitry-Sur-Seine";

SELECT * -- Tous les attributs
FROM etudiants_notes
WHERE note>=10 AND ressource="R105"; -- Filtres sur les valeurs
```

Groupement des données

```sql
SELECT etudiant_numero, AVG(notes) -- Moyenne des notes…
FROM etudiants_notes
GROUP BY etudiant_numero; -- … par étudiant

SELECT etudiant_numero, AVG(note) -- Moyenne des notes…
FROM etudiants_notes
WHERE note!=0 -- … autres que zéro …
GROUP BY ressource; -- … par ressource
```