Base de données

Cours 5 - Structures et intégrité des données

Table des matières

Les structures de données

La création et modification des structures de données constituent le fondement de toute base de données, permettant d'organiser et de stocker l'information de manière structurée et efficace.

Les structures de données

Les principales opérations pour gérer les structures de données dans une base de données relationnelle sont :

- Création de tables
- Modification de tables
- Suppression de table
- Contraintes

Les structures de données - La création de tables

Il est possible de créer une table en utilisant la commande SQL : **CREATE TABLE [Table name]**.

Une table est une collection de données organisées en lignes et en colonnes. Chaque table a un nom unique dans la base de données.

Les structures de données - La création de tables

```sql
CREATE TABLE course (
    course_id INT PRIMARY KEY,
    course_code VARCHAR(20) NOT NULL,
    course_name VARCHAR(100) NOT NULL,
    department VARCHAR(50),
    instructor VARCHAR(100) NOT NULL
);
```

Listing: https://www.postgresql.org/docs/current/sql-createtable.html

```sql
CREATE TABLE student (
    student_id INT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL
);
```

Listing: https://www.postgresql.org/docs/current/sql-createtable.html

Les structures de données - La modification de tables

Il est possible de modifier une table en utilisant la commande SQL : **ALTER TABLE [Table name]**.

Cette commande permet d'ajouter, de supprimer ou de modifier des colonnes dans une table existante.

Les structures de données - La modification de tables

Pour ajouter une colonne à une table, on utilise la syntaxe suivante :

```
1 -- Add a new column 'phone_number' to the 'student' table
2 ALTER TABLE student ADD COLUMN phone_number VARCHAR(15);
```

Listing: https://www.postgresql.org/docs/current/sql-altertable.html

Les structures de données - La modification de tables

Pour supprimer une colonne d'une table, on utilise la syntaxe suivante :

```
1 -- Delete the 'phone_number' column from the 'student' table
2 ALTER TABLE student DROP COLUMN phone_number;
```

Listing: `https://www.postgresql.org/docs/current/sql-altertable.html`

Les structures de données - La modification de tables

Pour modifier une colonne dans une table, on utilise la syntaxe suivante :

```sql
-- Modify the 'course_name' column in the 'course' table
-- to increase its length and set it to NOT NULL
ALTER TABLE course MODIFY COLUMN course_name VARCHAR(150) NOT NULL;
```

Listing: https://www.postgresql.org/docs/current/sql-altertable.html

Les structures de données - La modification de tables

Pour renommer une colonne d'une table, on utilise la syntaxe suivante:

```
1 -- Rename the 'instructor' column to 'professor' in the 'course' table
2 ALTER TABLE course RENAME COLUMN instructor TO professor;
```

Listing: `https://www.postgresql.org/docs/current/sql-altertable.html`

Les structures de données - La suppression de tables

Il est possible de supprimer une table en utilisant la commande SQL : **DROP TABLE [Table name]**.

Cette commande supprime la table **et toutes les données qu'elle contient** de la base de données.

Les structures de données - La suppression de tables

```sql
-- Delete the 'student' table from the database
DROP TABLE student;

-- Delete the 'student' table from the database if it exists
DROP TABLE IF EXISTS student;
```

Listing: https://www.postgresql.org/docs/current/sql-droptable.html

Les structures de données - Les contraintes

Les contraintes sont utilisées pour spécifier des règles concernant les données d'une table.

Elles servent **à limiter le type de données** qui peuvent être insérées dans une table. Cela garantit l'exactitude et la fiabilité des données de celle-ci. S'il y a **une violation** entre la contrainte et l'action sur les données, **l'action est abandonnée**.

Les structures de données - Les contraintes

Les types de contraintes les plus courants sont :

- **PRIMARY KEY** : Identifie de manière unique chaque enregistrement dans une table.
- **FOREIGN KEY** : Assure l'intégrité référentielle entre deux tables.
- **UNIQUE** : Garantit que toutes les valeurs d'une colonne sont différentes.
- **NOT NULL** : Empêche les valeurs NULL dans une colonne.
- **CHECK** : Assure que les valeurs dans une colonne respectent une condition spécifique.
- **DEFAULT** : Définit une valeur par défaut pour une colonne si aucune valeur n'est spécifiée lors de l'insertion.

Les structures de données - Les contraintes

```sql
CREATE TABLE student (
    student_id INT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL

    -- Advanced constraints
    UNIQUE (email) -- Make sure email is unique
);

CREATE TABLE student_course (
    student_id INT,
    course_id INT,
    enrollment_date DATE,

    -- Advanced constraints
    PRIMARY KEY (student_id, course_id),
    FOREIGN KEY (student_id) REFERENCES student(student_id),
    FOREIGN KEY (course_id) REFERENCES course(course_id)
);
```

Listing: https://www.postgresql.org/docs/current/sql-createtable.html

Intégrité référentielle

L'intégrité référentielle est un concept fondamental dans les bases de données relationnelles. Celle-ci garantit que les relations entre les tables restent cohérentes.

L'intégrité référentielle assure qu'une clé étrangère dans une table fait toujours référence à une clé primaire valide dans une autre table.

Intégrité référentielle

```sql
CREATE TABLE department (
    departmet_id INT PRIMARY KEY,
    department_name VARCHAR(50) NOT NULL,
);

CREATE TABLE professor (
    professor_id INT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL
    department_id INT,

    -- Referential integrity constraint
    FOREIGN KEY (department_id) REFERENCES department(departmet_id)
    ON DELETE CASCADE
);
 -- Other options for referential actions:
    -- ON DELETE SET NULL  -> Make NULL
    -- ON DELETE SET DEFAULT -> Default value
    -- ON DELETE RESTRICT  -> Restrict deletion if related records exist
```

Listing: https://www.postgresql.org/docs/current/sql-createtable.html

Cohérence des données

La cohérence des données fait référence à l'état dans lequel les données d'une base de données sont exactes, fiables et conformes aux règles définies.

Elle est essentielle pour assurer l'intégrité et la fiabilité des informations stockées.

Cohérence des Données - Problèmes de Cohérence Courants

La duplication de données :

```sql
CREATE TABLE student_course (
    id INT PRIMARY KEY,
    student_first_name VARCHAR(50), -- Duplication of the table 'student'
    student_last_name VARCHAR(50), -- Duplication of the table 'student'
    course_name VARCHAR(100), -- Duplication of the table 'course'
    course_instructor VARCHAR(100), -- Duplication of the table 'course'
    enrollment_date DATE
);
```

Listing: `https://www.postgresql.org/docs/current/sql-createtable.html`

Cohérence des Données - Problèmes de Cohérence Courants

À la place de dupliquer les colonnes des autres tables, il est préférable d'utiliser des références (clés étrangères) :

```sql
CREATE TABLE student_course (
    student_id INT,
    course_id INT,
    enrollment_date DATE,
    PRIMARY KEY (student_id, course_id),

    -- Reference to student and course tables
    FOREIGN KEY (student_id) REFERENCES student(student_id),
    FOREIGN KEY (course_id) REFERENCES course(course_id)
);
```

Listing: https://www.postgresql.org/docs/current/sql-createtable.html

Cohérence des Données - Problèmes de Cohérence Courants

Tables inutiles ou mal conçues :

```sql
-- Redundant structure (bad design)
CREATE TABLE student_full_time (...);
CREATE TABLE student_part_time (...);
```

Listing: https://www.postgresql.org/docs/current/sql-createtable.html

```sql
-- Normalized structure (optimized design)
CREATE TABLE student (
    student_id INT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL,
    enrollment_type VARCHAR(10), -- e.g., 'full-time', 'part-time'
);
```

Listing: https://www.postgresql.org/docs/current/sql-createtable.html

Cohérence des Données - Bonnes Pratiques

Il est important de suivre certaines bonnes pratiques pour maintenir la cohérence des données dans une base de données relationnelle :

- Utiliser des contraintes pour garantir l'intégrité des données.
- Normaliser les données pour réduire la redondance.
- Éviter la duplication de données et la creation de tables inutiles.
- Définir des valeurs par défaut appopriées.
- Utiliser des transactions pour garantir que les opérations sur les données sont complètes et cohérentes.

Bibliographie

- https://www.postgresql.org/docs/current/sql-createtable.html
- https://www.postgresql.org/docs/current/sql-altertable.html
- https://www.postgresql.org/docs/current/sql-droptable.html
- https://www.w3schools.com/sql/default.asp