SQL - Introduction

Ludovic Deneuville

Structured Query Language

> **Définition**
>
> Le SQL est un langage déclaratif et standardisé utilisé pour interagir avec des bases de données relationnelles.
>
> Il permet de créer, manipuler, interroger et gérer les données stockées dans des tables.

Norme SQL

SQL est un language normé dont la dernière version est **SQL:2023**.

> **Caution**
>
> Les SGBD ne respectent pas exactement la norme SQL.
>
> À un niveau avancé d'utilisation, il y a des différences entre les SGBD.

- PostgreSQL SQL
- DuckDB SQL

Composantes du langage

Objectifs

- Définition de Données
- Manipulation de Données
- Contrôle des Transactions
- Contrôle d'accès aux Données

Sous-langages

	Acronyme	Exemple
DDL	Data Definition Language	`CREATE TABLE`, `ALTER TABLE`, `DROP TABLE`
DML	Data Manipulation Language	`INSERT`, `UPDATE`, `DELETE`
DQL	Data Query Language	`SELECT`
DCL	Data Control Language	`GRANT`, `REVOKE`
TCL	Transaction Control Language	`COMMIT`, `ROLLBACK`

Définition de Données

Créer une table

```sql
CREATE TABLE joueuse (
    id_joueuse     INT,
    nom            TEXT,
    prenom         TEXT,
    date_naissance DATE,
    elo            INT,
    est_arbitre    BOOLEAN,
    code_pays      TEXT,
    taux_victoires FLOAT
);
```

CREATE avancé

```sql
CREATE TABLE echecs.joueuse (
    id_joueuse     SERIAL         PRIMARY KEY,
    nom            VARCHAR(50)    NOT NULL,
    prenom         VARCHAR(50)    NOT NULL,
    date_naissance TIMESTAMP,
    elo            INT            CHECK (elo > 0),
    mail           VARCHAR(100)   UNIQUE,
    est_arbitre    BOOLEAN        DEFAULT false,
    code_pays      VARCHAR(2)     REFERENCES echecs.pays(code),
    taux_victoires FLOAT          CHECK (taux_victoires BETWEEN 0 AND 1)
);
```

! Contraintes

- `NOT NULL` : l'attribut doit être renseigné
- `UNIQUE` : chaque valeur de l'attribut est unique (pas de doublons)
- `CHECK` : pour vérifier la cohérence d'un attribut

Schéma

Utile lorsque le nombre de tables devient grand.

Cela facilite :

- l'organisation des objets, la clarté
- la Gestion des permissions

```sql
CREATE SCHEMA echecs;
```

Table dans un schéma

```
CREATE TABLE echecs.club (
    id_club          SERIAL          PRIMARY KEY,
    nom              VARCHAR(100)    NOT NULL,
    ville            VARCHAR(100)    NOT NULL,
    date_creation    DATE
);
```

Déplacer la table *joueuse* dans le schéma *echecs* :

```
ALTER TABLE joueuse SET SCHEMA echecs;
```

> **i** Schéma par défaut
>
> Par défaut les objets sont créés dans le schéma `public`.
>
> Ça devient rapidement un joyeux bazar si on ne range pas ses tables.

Autres CREATE

```
CREATE TABLE top_joueuses AS
SELECT nom, prenom, elo
  FROM echecs.joueuse
 WHERE elo > 2200;


CREATE TABLE joueuse_vide AS
SELECT nom, prenom, elo
  FROM echecs.joueuse
 WHERE 0=1;


CREATE TABLE joueuse_backup (LIKE echecs.joueuse INCLUDING ALL);
```

Bonnes pratiques

- Nom de table représentatif et au singulier
- Pas de majuscules, pas d'accents
- Mots séparés par des underscores (snake_case)
- Ne pas utiliser de mots clés réservés (date, user…)
- Ne pas avoir de colonne qui porte le même nom que sa table

Modifier une colonne

Modifier le type :

```
ALTER TABLE joueuse
ALTER COLUMN nom TYPE VARCHAR(80);
```

Renommer :

```
ALTER TABLE echecs.joueuse
RENAME COLUMN date_naissance TO dnais;
```

Créer/Supprimer une colonne

Ajout :

```
ALTER TABLE echecs.joueuse
ADD COLUMN id_club INT;
```

Suppression :

```
ALTER TABLE echecs.joueuse
DROP COLUMN taux_victoires;
```

Définir une clé étrangère

- Nous venons de créer une colonne `id_club`
- Lions maintenant cette colonne à la table *club*

```
ALTER TABLE echecs.joueuse
ADD CONSTRAINT fk_id_club
FOREIGN KEY (id_club)
REFERENCES echecs.club(id_club);
```

Clé étrangère

Dans la table joueuse, l'attribut `id_club` a pour valeur soit :

- une valeur de `id_club` de la table *club*
- NULL

Autres possibilités

- Renommer une table
- Ajouter / Supprimer une contrainte
- Définir / Supprimer une valeur par défaut

Supprimer une table

```
DROP TABLE echecs.joueuse;
```

⚠️ Warning

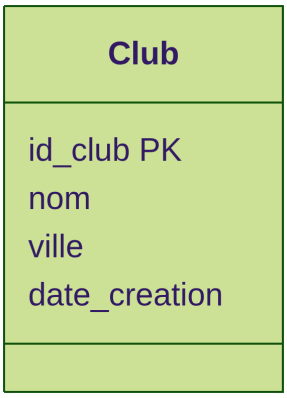
Cette opération est irréversible !

Elle supprime définitivement :

- la table
- toutes les données qu'elle contient

```
DROP TABLE IF EXISTS echecs.joueuse;      -- utile pour les scripts SQL
```

Insérer des données




```
-- Rappel du CREATE
CREATE TABLE echecs.club (
    id_club          SERIAL          PRIMARY KEY,
    nom              VARCHAR(100)    NOT NULL,
    ville            VARCHAR(100)    NOT NULL,
    date_creation    DATE
);
```

```
INSERT INTO echecs.club
(nom,                            ville,           date_creation)
VALUES
('C''Chartres Echecs',           'Chartres',      '1991-09-01'),
('Bischwiller',                  'Bischwiller',   '1976-09-01'),
('Asnieres - le Grand Echiquier', 'Asnieres',     '1987-03-20'),
('Tours des Hauts-De-France',    'Saint-Quentin', '1999-12-01');
```

Pourquoi l'on n'insère pas la colonne *id_club* ? 😕

Séquence

La colonne *id_club* est une séquence gérée par le SGBD.

Les valeurs sont générées automatiquement à l'insertion et la séquence est incrémentée (1, 2, 3…).

Séquences

Il est possible de créer et gérer ses propres séquences :

```
CREATE SEQUENCE nom_sequence START 1 INCREMENT BY 1;

SELECT currval('nom_sequence'); -- dernière valeur utilisée
SELECT nextval('nom_sequence'); -- prochaine valeur

ALTER SEQUENCE nom_sequence RESTART WITH nouvelle_valeur;

DROP SEQUENCE nom_sequence;
```

Lire des données

`SELECT * FROM <schema>.<table>` pour tout afficher.

```
SELECT *
  FROM echecs.club;
```

id_club	nom	ville	date_creation
1	C'Chartres Echecs	Chartres	1991-09-01
2	Bischwiller	Bischwiller	1976-09-01
3	Asnieres - le Grand Echiquier	Asnieres	1987-03-20
4	Tours des Hauts-De-France	Saint-Quentin	1999-12-01

Colonnes à afficher

Listez les colonnes à afficher après le `SELECT`

```
SELECT nom,
       prenom,
       elo
  FROM echecs.joueuse;
```

nom	prenom	elo
Guichard	Pauline	2357
Daulyte-Cornette	Deimante	2371
Navrotescu	Andreea	2338
Sebag	Marie	2448
Millet	Sophie	2366
Skrichempko	Almira	2361

Filtrer les lignes

Filtrez sur les lignes avec une clause `WHERE`.

```
SELECT nom,
       prenom,
       elo
  FROM echecs.joueuse
 WHERE elo > 2400;
```

nom	prenom	elo
Sebag	Marie	2448

Plusieurs filtres

Le mot clé `AND` permet d'ajouter d'autres filtres.

```
SELECT nom,
       prenom,
       elo
  FROM echecs.joueuse
 WHERE elo < 2400
   AND prenom LIKE 'A%';
```

nom	prenom	elo
Navrotescu	Andreea	2338
Skrichempko	Almira	2361

Vous pouvez autoriser plusieurs conditions avec `OR`.

```
SELECT nom,
       prenom,
       elo
  FROM echecs.joueuse
 WHERE elo > 2400
    OR prenom LIKE 'A%';
```

nom	prenom	elo
Sebag	Marie	2448
Navrotescu	Andreea	2338
Skrichempko	Almira	2361

Opérations booléennes

- Utilisation des mots-clés `AND`, `OR` et `NOT`
- *AND* prioritaire sur le *OR*

```sql
SELECT nom,
       prenom,
       elo
  FROM echecs.joueuse
 WHERE elo > 2400 OR (prenom LIKE 'A%' AND nom NOT LIKE 'N%') ;
```

LIKE

```sql
SELECT *
  FROM echecs.joueuse
 WHERE prenom LIKE 'A%'          -- commence par A
    OR prenom LIKE '%A'          -- termine par A
    OR prenom LIKE '%A%'         -- contient au moins un A
    OR UPPER(prenom) LIKE '%A%' -- contient au moins un A ou un a
    OR prenom LIKE 'A__'         -- commence par A suivi de 2 autres caractères
```

- `%` : représente entre 0 et une infinité de caractères
- `_` : représente exactement 1 caractère

> LIKE est sensible à la Casse
>
> MAJUSCULES ≠ minuscules

Plage de valeurs

`BETWEEN` permet de spécifier une plage de valeurs numériques.

```sql
SELECT nom,
       prenom,
       elo
  FROM echecs.joueuse
 WHERE elo BETWEEN 2350 AND 2400;
```

nom	prenom	elo
Guichard	Pauline	2357
Daulyte-Cornette	Deimante	2371
Millet	Sophie	2366
Skrichempko	Almira	2361

Valeurs dans une liste

Vous pouvez spécifier qu'un attribut doit appartenir à un liste avec `IN`.

Ou inversement avec `NOT IN`.

```sql
SELECT nom,
       prenom,
       elo
```

```
  FROM echecs.joueuse
 WHERE prenom IN ('Deimante', 'Sophie');
```

nom	prenom	elo
Daulyte-Cornette	Deimante	2371
Millet	Sophie	2366

Filtre sur les attributs booléens

```
SELECT *
  FROM echecs.joueuse
 WHERE est_arbitre IS TRUE;
```

```
SELECT *
  FROM echecs.joueuse
 WHERE [NOT] est_arbitre;
```

Ordonner le résultat

`ORDER BY` pour ordonner selon plusieurs colonnes.

```
SELECT nom,
       prenom,
       elo
  FROM echecs.joueuse
 ORDER BY elo,
          taux_victoires DESC;
```

nom	prenom	elo
Navrotescu	Andreea	2338
Guichard	Pauline	2357
Skrichempko	Almira	2361
Millet	Sophie	2366
Daulyte-Cornette	Deimante	2371
Sebag	Marie	2448

Afficher n lignes

```
SELECT nom,
       prenom,
       elo
  FROM echecs.joueuse
 LIMIT 3;
```

nom	prenom	elo
Guichard	Pauline	2357
Daulyte-Cornette	Deimante	2371
Navrotescu	Andreea	2338

Mise à jour de données

```
UPDATE echecs.joueuse
   SET elo = 2399,
       prenom = 'Popo'
 WHERE id_joueuse = 1;
```

```
SELECT id_joueuse,
       nom,
       prenom,
       elo
  FROM echecs.joueuse
 WHERE id_joueuse = 1;
```

id_joueuse	nom	prenom	elo
1	Guichard	Popo	2399

Màj globale

Imaginons que vous souhaitez modifier le `elo` de toutes les joueuses.

```
UPDATE echecs.joueuse
   SET elo = elo + 100;
-- RETURNING id_joueuse, nom, prenom, elo;
```

> ⚠ Warning
>
> Généralement un message d'avertissement vous demande si vous êtes certain de vouloir mettre à jour toutes les lignes.

Màj à partir d'une autre table

```
--ALTER TABLE echecs.joueuse
--ADD COLUMN club_commencant_par_a BOOLEAN DEFAULT false;

UPDATE echecs.joueuse j
   SET club_commencant_par_a = true
  FROM echecs.club c
 WHERE j.id_club = c.id_club
   AND c.nom LIKE 'A%';
```

Suppression de données

Supprimer une joueuse :

```
DELETE FROM echecs.joueuse
 WHERE id_joueuse = 1;
```

Vider la table :

```
DELETE FROM echecs.joueuse;
```

Validation des opérations

Et si ? Vous avez exécuté par erreur :

`DELETE FROM echecs.joueuse;` 😱

Il y a un système de transactions avec les mots clés :

- BEGIN
- COMMIT
- ROLLBACK

> ⚠️ Auto-commit
>
> Attention : il n'y a pas de garde-fou lorsque vous êtes en Auto-commit.

Dates et Heures

Type DATE

Quelques opérations avec le type `DATE`.

```
SELECT nom,
       prenom,
       EXTRACT(YEAR FROM date_naissance) AS annee_naissance,
       date_naissance + INTERVAL '10 years' AS anniv_dix_ans,
       AGE(CURRENT_DATE, date_naissance) AS age
  FROM echecs.joueuse
 WHERE date_naissance >= '1991-04-20'
    OR date_naissance BETWEEN '1991-04-20' AND '1992-12-31'
    OR EXTRACT(MONTH FROM date_naissance) = 4;
```

Type TIMESTAMP

Pour stocker la date et l'heure

```
CREATE TABLE echecs.club (
    id_club          SERIAL          PRIMARY KEY,
    nom              VARCHAR(100)    NOT NULL,
    ville            VARCHAR(100)    NOT NULL,
```

```
    date_creation    TIMESTAMP
);
```

Utilisations

```
INSERT INTO echecs.club (nom, ville, date_creation)
VALUES ('C''Chartres Echecs', 'Chartres', '1991-09-01 20:00:00');
```

```
INSERT INTO echecs.club (nom, ville, date_creation)
VALUES ('Bischwiller', 'Bischwiller', CURRENT_TIMESTAMP);
```

Vous pouvez utiliser également sur le type *TIMESTAMP* :

- `EXTRACT`,
- `BETWEEN`,
- les opérateurs de comparaison

Valeurs nulles

Tester si un attribut est NULL

Joueuses dont le mail n'est pas renseigné :

```
SELECT *
  FROM echecs.joueuse
 WHERE mail IS NULL;
```

Et inversemenent :

```
SELECT *
  FROM echecs.joueuse
 WHERE mail IS NOT NULL;
```

COALESCE

La fonction COALESCE

En PostgreSQL, la fonction COALESCE :

- prend un nombre quelconque de paramètres
- retourne le 1er paramètre dont la valeur est NON NULLE

COALESCE - Exemple

```
SELECT COALESCE(mail, tel, 'pas de contact')
  FROM echecs.joueuse
 WHERE id_joueuse = 1
```

La requête ci-dessus va afficher :

- Si mail est NON NULL ➡️ mail
- Sinon
  - Si tel NON NULL ➡️ tel
  - Sinon ➡️ `pas de contact`

NULLIF

```
SELECT NULLIF(tel, '00.00.00.00.00')
  FROM echecs.joueuse;
```

- Si tel = `00.00.00.00.00`
  - Renvoie *NULL*

Divers

Jouer avec les colonnes

```
SELECT CONCAT(nom, ' ', prenom) AS nom_complet,
       elo
  FROM echecs.joueuse;
```

nom_complet	elo
Guichard Pauline	2357
Daulyte-Cornette Deimante	2371
Navrotescu Andreea	2338
Sebag Marie	2448
Millet Sophie	2366
Skrichempko Almira	2361

Supprimer les doublons

Le mot clé `DISTINCT` permet de supprimer les doublons.

```
SELECT DISTINCT prenom
  FROM echecs.joueuse;
```

Choix

```
SELECT CASE WHEN elo > 2400 THEN 'Niveau MI'
            WHEN elo > 2200 THEN 'Joueuse très forte'
            ELSE 'Joueuse de niveau normal'
       END
  FROM echecs.joueuse;
```

Échantillon

Afficher un échantillon aléatoire de 10 % des lignes

```
SELECT *
  FROM echecs.joueuse
TABLESAMPLE BERNOULLI(10);
```

Et même !?

```
SELECT 1,
       TRUE,
       'a',
       current_date,
       NOW()
  FROM echecs.joueuse;
```

Résumé

- CREATE / ALTER / DELETE
- INSERT
- SELECT
- WHERE