420-C42

Langages d'exploitation des bases de données

Partie 6

DDL

DDL

- Le DDL (*Data Definition Language*) permet la définition de la structure de la base de données.
- Les clauses principales sont :
  - CREATE création d'objets
  - ALTER modification d'objets
  - DROP suppression d'objets
- Les clauses secondaires sont :
  - RENAME renommer un objet
  - TRUNCATE TABLE vider une table (seulement pour les objets tables)
  - COMMENT déterminer ou modifier le commentaire d'un objet

DDL structure lexicale

- Le nom d’un objet :
  - doit avoir un nom unique qui n’existe pas dans la BD pour le même groupe hiérarchique. Par exemple :
    - dans la BD *projets*, une table *pays*
    - dans la table *pays*, une colonne *nom*
  - au plus 63 caractères (possible d’en avoir plus mais le nom est tronqué)
  - débute par l’un de ces caractères : a-z, _
  - enchaîne par l’un de ces caractères : a-z, 0-9, _, $
  - n’est pas sensible à la casse
  - ne doit pas utiliser un mot réservé du langage (SELECT, CREATE, ...)
  - il est possible de créer un nom quelconque à l’aide des guillemets CREATE TABLE "select" ...

DDL CREATE TABLE

- La clause CREATE TABLE permet la création d'une table.
- Le synopsis simplifié de cette clause est :

CREATE TABLE *nom_table* (
nom_colonne *type_colonne* [*contrainte_colonne* [, …]]
[, ...]

[*contrainte_table* [, ...]]
);

DDL types de données

- Chaque SGBD propose plusieurs types de données similaires mais différents. On couvre ici les types principaux de PostgreSQL.
- Types numériques :

• SMALLINT	2 octets	
• INTEGER	4 octets	
• BIGINT	8 octets	
• DECIMAL(précision = limite impl. , échelle = 0)	variable	exact
NUMERIC(précision = limite impl. , échelle = 0)	variable	exact
• REAL	4 octets	inexact
• DOUBLE PRECISION	8 octets	inexact

DDL types de données

- Types numériques à incrément automatique :
  - SMALLSERIAL SMALLINT
  - SERIAL INTEGER
  - BIGSERIAL BIGINT
- En fait, ces types sont des alias de types mettant en place plusieurs mécanismes permettant cette automatisation. Nous verrons plus tard les objets SEQUENCE qui sont impliqués ici.
- Ils sont une substitution à la propriété AUTO_INCREMENT de MySQL et de son équivalent chez Oracle (GENERATE ... AS IDENTITY).

DDL types de données

- Types caractère :
  - CHARACTER VARYING | VARCHAR — longueur variable sans limite
    CHARACTER VARYING(*n*) | VARCHAR(*n*) — longueur variable avec limite
  - CHARACTER(n) | CHAR(n) — longueur fixe
  - TEXT — longueur variable sans limite
- Type binaire
  - BYTEA — données binaires
  - *Attention aux types CLOB et BLOB*

DDL types de données

- Types date et heure :
  - DATE — date (sans heure)
  - TIME — heure (sans date) (avec ou sans fuseau horaire)
  - TIMESTAMP — date et heure (avec ou sans fuseau horaire)
  - INTERVAL — intervalle de temps

- Les types dates doivent être utilisé avec attention. Leur manipulation requiert l'usage des fonctions spécifiques à leur usage. Voir la documentation pour tous les détails.

DDL types de données

- Type booléen
  - BOOLEAN booléen
- Type énuméré
  - PostgreSQL permet la création de types personnalisés énumérés (similaire à *enum* de plusieurs autres langages).
  - CREATE TYPE jour_semaine AS ENUM
    ('lundi', 'mardi', 'mercredi', 'jeudi', 'vendredi');

DDL types de données

- PostgreSQL possède plusieurs autres types utilitaires :
  - Types géométriques 2d :
    - POINT — point (x, y)
    - LINE — ligne $(ax + by + c = 0)$
    - LSEG — segment de ligne $((x_1, y_1), (x_2, y_2))$
    - BOX — rectangle $((x_1, y_1), (x_2, y_2))$
    - PATH — chemin fermé ou ouvert $((x_1, y_1), \ldots, (x_n, y_n))$
    - POLYGON — polygone $((x_1, y_1), \ldots, (x_n, y_n))$
    - CIRCLE — cercle $<(x_c, y_c), r>$
  - Types utilitaires:
    - MONEY — monétaire
    - BIT & BIT VARYING — représentation binaire (chaîne de bits)

DDL types de données

- PostgreSQL possède plusieurs autres types utilitaires :
  - Réseautique et matériel :
    - INET adresse IPv4 ou IPv6 + sous réseau
    - CIDR format des adresses IPV4 ou IPV6
    - MACADDR adresse MAC
    - UUID *Universally Unique Identifiers*
  - Fichiers texte structurés avec accès interne (limité) :
    - XML
    - JSON
  - Types composés (tableaux et structures)

DDL contraintes

- Le langage SQL propose 6 contraintes liées aux colonnes d'une table :
  - Permet de déléguer la gestion de ces contraintes au SGBD. Ainsi, une requête violant une de ces contraintes est systématiquement refusée. Cette approche présente plusieurs avantages.
  - Les contraintes :

Contrainte	Colonne	Colonnes	Par défaut	Inclus
Valeur nulle permise	oui	non	permise	-
Valeur par défaut	oui	non	NULL	-
Valeur unique	oui	oui	doublon	INDEX
Validation de la valeur	oui	oui	X	-
Clé primaire	oui	oui	X	NOT NULL UNIQUE
Clé étrangère	oui	oui	X	-

- **Colonne**
  Peut être définie sur une seule colonne à la fois
- **Colonnes**
  Peut être définie sur plusieurs colonnes à la fois (contrainte de table)
- **Par défaut**
  comportement si non spécifié
- **Inclus**
  L'application de cette contrainte inclus systémati-quement d'autres contraintes
- X => ne s'applique pas

DDL contraintes NULL / NOT NULL

- La contrainte de valeur nulle rend possible ou non qu'une colonne accepte une valeur nulle.
- Si la contrainte est non spécifiée, la valeur nulle est acceptée.
- Mots clés :
  - NULL permet les valeurs nulles
  - NOT NULL interdit les valeurs nulles

DDL

contraintes NULL / NOT NULL

```
CREATE TABLE employe (
    nom                 VARCHAR     NOT NULL,
    date_naissance      DATE        NULL,
    departement         INTEGER,
    commission          NUMERIC     CONSTRAINT nc_emp_com
                                        NOT NULL
);
```

DDL contrainte DEFAULT

- La contrainte de valeur par défaut permet de préciser la valeur d'une colonne si elle n'est pas déterminée lors de l'insertion d'une ligne.
- La contrainte de valeur par défaut à nulle existe si la colonne accepte les valeurs nulles et si aucune valeur par défaut n'est spécifiée.
- Elle peut être définie sur une colonne à la fois.
- Mots clés :
  - DEFAULT défini la contrainte de valeur par défaut

DDL contrainte DEFAULT

```
CREATE TABLE employe (
    nas             INTEGER     NOT NULL,    -- aucune valeur par défaut
    nom             VARCHAR,                 -- valeur par défaut : NULL
    prenom          VARCHAR     NULL,        -- valeur par défaut : NULL
    commission      NUMERIC     DEFAULT NULL, -- même que les 2+haut
    courriel        VARCHAR     DEFAULT 'information@abc_xyz.com’,
    date_embauche   DATE        CONSTRAINT dc_emp_emb
                                    DEFAULT CURRENT_DATE
);
```

DDL contrainte UNIQUE

- La contrainte d'unicité garantie que la colonne ne possède pas de doublon.
- Cette contrainte implique la création d'un index (on y reviendra).
- Elle peut être définie sur une ou plusieurs colonnes à la fois.
- Mots clés :
  - UNIQUE interdit les doublons

DDL

contrainte UNIQUE

```
CREATE TABLE employe (
    nas                     INTEGER         UNIQUE,
    nom                     VARCHAR,
    prenom                  VARCHAR,
    courriel_entreprise     VARCHAR         CONSTRAINT uc_emp_ce
                                                UNIQUE,
    courriel_personnel      VARCHAR,

    CONSTRAINT uc_emp_cp                    UNIQUE(courriel_personnel),
    CONSTRAINT uc_emp_nom_prenom            UNIQUE(nom, prenom)
);
```

DDL contrainte CHECK

- La contrainte de validation permet d'accepter ou non une valeur lors de son insertion ou de sa modification.
- Cette contrainte est très puissante (expressions) et en même temps limitée (ne peut effectuer de requête sur d'autres tables).
- Elle peut être définie sur une ou plusieurs colonnes à la fois.
- Mots clés :
  - CHECK défini la contrainte de validation

DDL contrainte CHECK

```
CREATE TABLE employe (
    nas                 INTEGER     CHECK( nas BETWEEN 100000000
                                                   AND 999999999),
    nom                 VARCHAR     CONSTRAINT cc_emp_nom
                                        CHECK(LENGTH(nom) > 1),
    date_naissance      DATE,
    date_embauche       DATE,
    ...
    CONSTRAINT cc_emp_date          CHECK (date_embauche >=
                                        date_naissance + INTERVAL'18 YEARS')
);
```

DDL contrainte PRIMARY KEY

- La contrainte de clé primaire représente la clé primaire choisie par le concepteur.
- Cette contrainte implique les contraintes NOT NULL et UNIQUE. Il est interdit de répéter l'existence de ces contraintes.
- Elle peut être définie sur une ou plusieurs colonnes à la fois.
- Mots clés :
  - PRIMARY KEY défini la clé primaire

DDL

contrainte PRIMARY KEY

```
CREATE TABLE employe (
    nas                 INTEGER         PRIMARY KEY,
    ...
);
CREATE TABLE employe (
    nas                 INTEGER         CONSTRAINT pk_emp
                                            PRIMARY KEY,
    ...
);
```

DDL

contrainte PRIMARY KEY

```
CREATE TABLE employe (
    nas                     INTEGER,
    ...
    CONSTRAINT pk_emp       PRIMARY KEY (nas)
);
CREATE TABLE employe (
    nom                     VARCHAR,
    prenom                  VARCHAR,
    adresse                 VARCHAR,
    ...
    CONSTRAINT pk_emp       PRIMARY KEY (nom, prenom, adresse)
);
```

DDL contrainte FOREIGN KEY … REFERENCES

- La contrainte de clé étrangère représente un lien entre deux colonnes choisie par le concepteur. Elle rend l'existence de la valeur liée obligatoire.
- Cette contrainte requiert que la colonne liée ou les colonnes liées soient de mêmes types et possèdent la contrainte UNIQUE. Il n'y a pas d'autres contraintes (NOT NULL ou PRIMARY KEY par exemple).
- Elle peut être définie sur une ou plusieurs colonnes à la fois.
- Mots clés :
  - REFERENCES défini la clé étrangère sur une colonne
  - FOREIGN KEY … REFERENCES défini la clé étrangère en fin de table

DDL contrainte FOREIGN KEY … REFERENCES

- Il est possible de déterminer le comportement du SGBD en cas de dépendance.
- Mots clés :
  - ON DELETE à la suppression
  - ON UPDATE à la modification
  - -
  - NO ACTION -> erreur
  - RESTRICT -> erreur (comportement par défaut)
  - CASCADE -> propage l'action (suppression ou modif.)
  - SET NULL -> met la valeur nulle (si la colonne l'accepte)
  - SET DEFAULT -> met la valeur par défaut (si définie)

DDL contrainte FOREIGN KEY … REFERENCES

```
CREATE TABLE departement (
    id                  INTEGER         PRIMARY KEY,
    nom                 VARCHAR,
    ...
);
CREATE TABLE employe (
    nas                 INTEGER         PRIMARY KEY,
    departement         INTEGER         REFERENCES departement(id),
    superviseur         INTEGER         CONSTRAINT fk_emp_sup
                                            REFERENCES employe(nas),
    ...
);
```

DDL contrainte FOREIGN KEY … REFERENCES

```
CREATE TABLE employe_projet (
    employe                 INTEGER,
    projet                  INTEGER,
    CONSTRAINT pk_emp_pro   PRIMARY KEY (employe, projet),
    CONSTRAINT fk_emp_pro_emp   FOREIGN KEY (employe) REFERENCES employe(nas),
    CONSTRAINT fk_emp_pro_pro   FOREIGN KEY (projet) REFERENCES projet(id)
                                    ON DELETE CASCADE ON UPDATE SET NULL
);
CREATE TABLE description_tache (
    employe                 INTEGER,
    projet                  INTEGER,
    CONSTRAINT fk_desc_tache    FOREIGN KEY (employe, projet)
                                    REFERENCES employe_projet(employe, projet)
);
```

DDL

DROP TABLE

- La clause DROP TABLE supprime la table et ses données.
- Le synopsis est :

DROP TABLE [IF EXISTS] name [, ...] [CASCADE | **RESTRICT**];

DROP TABLE IF EXISTS employe CASCADE;

DDL

ALTER TABLE

- La clause ALTER TABLE permet la modification d'une table.
- Un extrait du synopsis est :

ALTER TABLE *nom_table*

[[ADD COLUMN ...] |

[DROP COLUMN ...] |

[ADD CONSTRAINT …] |

[DROP CONSTRAINT …] |

[…]];

DDL

ALTER TABLE

```
ALTER TABLE employe
        ADD CONSTRAINT fk_emp_dep
                FOREIGN KEY (dep) REFERENCES departement(id);
```

DDL script de création

- Attention à la création de tables avec dépendances circulaires.
- Un script de création de tables est généralement construit de cette façon :
  - suppression de tous les objets en ordre inverse de création
  - création des objets dont les tables dépendent (séquences, types, ...)
  - création des tables sans les contraintes de clé étrangère
  - modification des tables et ajout des contraintes de clé étrangère
  - ajouts des objets supplémentaires