Sommaire

1 Introduction

Les cours précédents nous ont montré les moyens d'accéder aux données d'une base existante, et de mise à jour de ces données. Il nous reste à voir un élément important, à savoir la création et la modification de la structure de la base de donnée. Cette création fait également appel à des commandes SQL regroupées sous l'acronyme de LDD(Langage de Définition des Données).

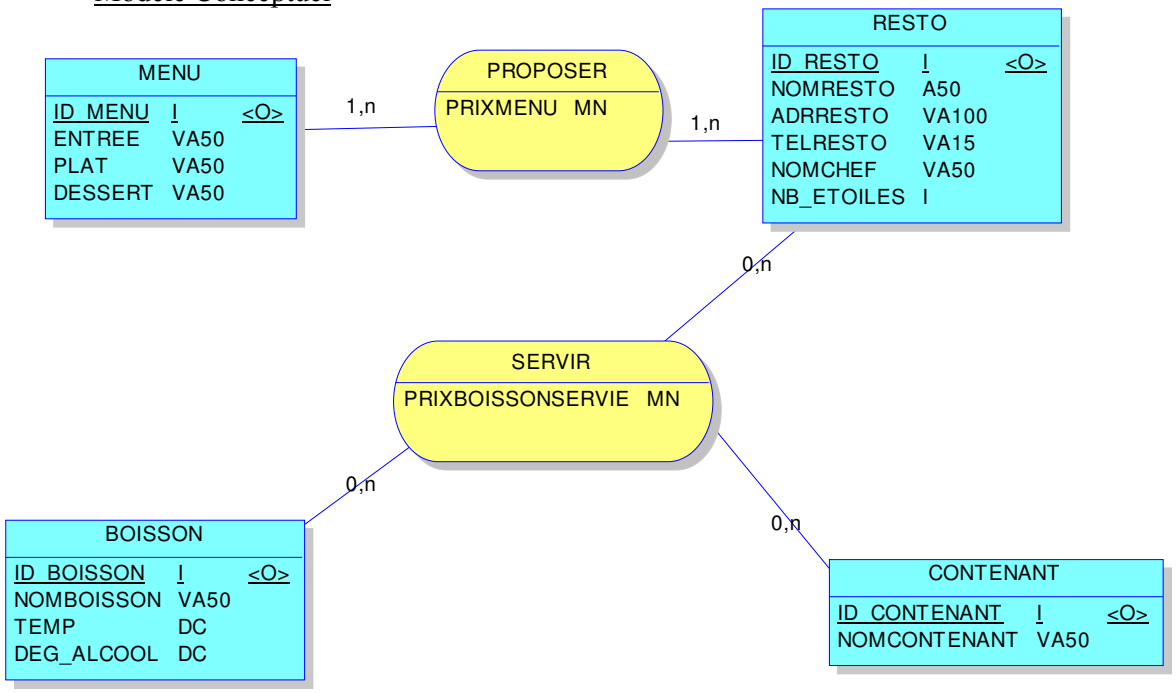
La définition des données va permettre de définir des bases de données, des tables, des contraintes et des index, d'en modifier ou supprimer tout ou partie.

Cette partie s'appuie sur le modèle relationnel. Tous les attributs, et plus particulièrement leur type (entier, chaîne…) doivent être définis.

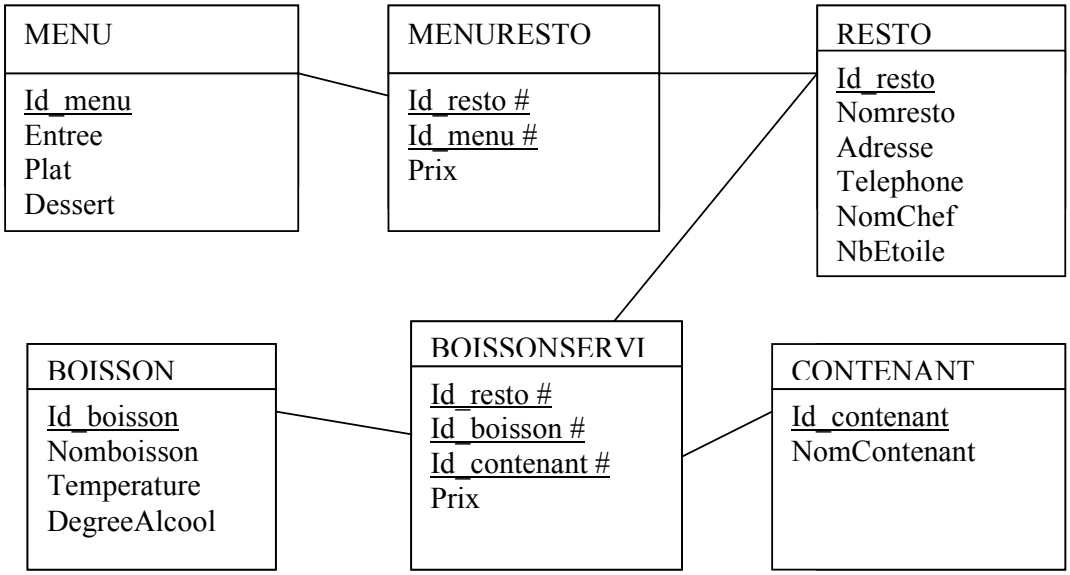
2 Présentation du modèle du cours

Pour les exemples de ce cours, nous nous appuierons sur la base de donnée RESTO vue lors de l'évaluation. Cette base est plus riche en terme d'organisation et permet de voir les différents cas de figure.

Modèle Conceptuel



Modèle relationnel



3 Les types de données

Les types de données sont directement dépendant des bases utilisées. Certains termes ont également évolués avec le temps (NUMERIC → NUMBER), néanmoins on peut retenir les types les plus standards à savoir :

3.1 Les types numériques exacts

- INTEGER (ou INT) : Type entier, la taille (8, 16 ou 32 bits) dépendant de l'implémentation
- SMALLINT : Valeur entière de taille inférieure à INTEGER
- NUMERIC (p, s) : Définit un nombre non entier. p définit le nombre de caractères, s le nombre de chiffres après la virgule.
- DECIMAL (p,s) : Idem
- NUMBER (t,d) : Idem NUMERIC

3.2 Les types numériques non exacts

- REAL : Définit un type réel, la précision dépendant de l'implémentation
- DOUBLE PRECISION : Idem précédemment mais avec une précision double.
- FLOAT (p) : Idem avec un paramètre spécifiant la précision.

3.3 Les types chaîne de caractère

- CHAR (x) : Définit une chaîne de x caractère
- VARCHAR (x) : Idem mais la longueur stockée dans la base est variable et limitée à x. Cela permet d'optimiser la taille de la base.
- NCHAR ou NVARCHAR : Idem précédemment mais les caractères sont codés en unicode.

3.4 Les types bits :

Ce type de données permet de stocker des chaînes de bits ne répondant pas à un codage standard (image, son numérisé…). La taille peut aller à plusieurs giga-bits

- BIT(x)
- BIT VARYING(x) : Idem mais stockage variable en fonction de la taille.

3.5 Les types temporels

Les types temporels représentent tout ce qui a trait aux dates et heures. Attention, le format de stockage diffère en fonction des SGBD, et plus particulièrement en ce qui concerne l'ordre.

- DATE : Type date : 2 caractères pour le jour, 2 pour le mois, 4 pour l'année
- TIME (x) : Type heure : 2 caractères pour l'heure (0 → 23), 2 pour les minutes (0 → 59) et 2 pour les secondes. Le paramètre x permet de définir le nombre de chiffres après la virgule pour les secondes.
- TIMESTAMP (x) : Type complet date et heure. Par défaut, x = 6.

3.6 Les conversions de type

Comme la plupart des langages, SQL effectue la conversion de certain type de façon automatique (addition d'un entier et d'un réel). D'autres conversions sont obtenues à l'aide de fonctions. Ce point sera vu ultérieurement.

4 La création des tables

Une table est contenue dans une base de donnée. L'opération initiale est donc la création de la base de donnée. Dans tous les cas, une création ne peut se faire que si l'objet est inexistant.

4.1 Création de la base de donnée

Cette opération s'effectue à l'aide de la commande SQL CREATE. La syntaxe est la suivante :

```
CREATE DATABASE <nom de la base>
  [ [ WITH ] [ OWNER [=] <propriétaire de la base>
       [ TEMPLATE [=] <base modèle>]
       [ ENCODING [=] <type de codage de caractère>]
       [ TABLESPACE [=] tablespace ] ];
```

Les différents paramètres, excepté le nom de la base sont optionnels et peuvent être définis au sein de la base.

- OWNER est l'utilisateur propriétaire de la base (super utilisateur), par défaut, c'est celui qui crée la base.
- TEMPLATE est le nom d'une base de données existante dont on reprend les paramètres.
- ENCODING permet de définir l'encodage des caractères pour la base.
- TABLESPACE : Le tablespace permet à l'administrateur de la base de données d'indiquer l'endroit dans le système de fichier dans lequel sont stockés les fichiers représentant la base de données. Un tablespace est représenté par un nom et peut être utilisé pour plusieurs bases.

Exemple : Création de la base resto :

```
CREATE DATABASE RESTO
  WITH OWNER = postgres
      ENCODING = 'SQL_ASCII'
      TABLESPACE = pg_default;
GRANT ALL ON DATABASE test TO public;
```

La commande GRANT permet d'allouer des droits sur la base. Cette commande sera vue en 2ème année. Ici, tout le monde (public) dispose de tous les droits (all) sur la base.

4.2 Création des tables

Cette opération s'effectue à l'aide de la commande SQL CREATE. La syntaxe simplifiée est la suivante :

```
CREATE TABLE [nom de base.] nom de table
        (colonne1 type1 [default valeur1] [NOT NULL] ,
          colonne2 type2 [default valeur] [NOT NULL] ,
        ….
        [CONSTRAINT nom contrainte1 type contrainte1] …
        [ON DELETE action delete] [ON UPDATE actions update]
        );
```

- nom de base : Le nom de la base est optionnel. Dans ce cas, la table sera créée au niveau de la base courante.
- nom de table : Attention, les règles de nommage s'appliquent (pas de caractère particulier, pas d'accent …).
- colonne : Spécifie le nom de la colonne (champ) à créer
- type se réfère aux types de données existant dans la base (voir paragraphe sur les types de données)
- CONSTRAINT est un mot clé listant les contraintes à appliquer sur la table. Ces contraintes peuvent être :
    - PRIMARY KEY : déclaration d'une clé primaire (un index sur cette clé est automatiquement créé.
    - FOREIGN KEY : déclaration d'une clé étrangère. Dans ce cas, le mot clé REFERENCES sera utilisé pour dire à quel champ de quelle table cette clé est référencée. Cette contrainte peut être accompagnée d'une directive ON DELETE, ON UPDATE afin de respecter l'intégrité référentielle en cas d'effacement ou de mise à jour de clé primaire. Il faut définir alors l'action à effectuer.
    - CHECK permet de définir un domaine de valeur autorisé
    - UNIQUE impose une valeur unique pour une colonne au niveau de la table

Concernant les contraintes, il est conseillé de leur donner un nom explicite qui reprend le nom de la table afin de pouvoir les modifier par la suite ou les identifier en cas de message d'erreur.

Du fait de l'intégrité référentielle, les tables doivent être créées dans l'ordre à savoir :

- Les tables ne disposant que de clé primaire en premier
- Les tables avec clé étrangère ensuite.

Si ces points ne sont pas respectés, une erreur sera générée au niveau de la base et la table correspondante ne sera pas créée.

Afin de voir la syntaxe de création de table, nous allons créer l'ensemble des tables de la base resto.

Création de la table Menu

```
CREATE TABLE MENU
        (
          Id_Menu integer NOT NULL,
          entree varchar(50),
          plat varchar(50),
          dessert varchar(50),
          CONSTRAINT menu_pkey PRIMARY KEY (Id_Menu)
        );
```

Autre solution sans nommer la contrainte de clé primaire :

```
CREATE TABLE MENU
        (Id_Menu Integer NOT NULL PRIMARY KEY,
        Entree Varchar(50),
        Plat Varchar(50),
        Dessert Varchar(50)
        );
```

Création de la table RESTO avec une contrainte de domaine sur les étoiles :

```
CREATE TABLE resto
        (
          Id_resto integer NOT NULL,
          Nomresto char(50),
          Adresse varchar(100),
          Telephone varchar(15),
          NomChef varchar(50),
          NbEtoile integer,
          CONSTRAINT resto_pkey PRIMARY KEY (Id_resto),
          CONSTRAINT Resto_nbetoile CHECK (nbetoile IN (0,1,2,3))
        );
```

la syntaxe sur nbetoile pourrait aussi être :

- <=3
- BETWEEN 0 AND 3

Création de la table BOISSON:

```
CREATE TABLE BOISSON
        (Id_Boisson Integer,
        NomBoisson Varchar(50),
        Temperature NUMERIC (4,2),
        DegreeAlcool NUMERIC (2,1),
        CONSTRAINT Boisson_pkey PRIMARY KEY (Id_Boisson)
        );
```

Création de la table CONTENANT avec nom de contenant unique:

```
CREATE TABLE CONTENANT
        (Id_Contenant Integer,
        NomContenant Varchar(50),
        CONSTRAINT Contenant_pkey PRIMARY KEY (Id_Contenant),
        CONSTRAINT contenant_nom UNIQUE (NomContenant)
        );
```

Création de la table MENURESTO

```
CREATE TABLE MENURESTO
        (Id_Resto Integer,
         Id_Menu Integer,
         Prix Numeric(6,2),

        CONSTRAINT MenuResto_pkey PRIMARY KEY (Id_Resto, Id_Menu),
        CONSTRAINT MenuResto_Idresto_Fkey FOREIGN KEY (Id_Resto)
        REFERENCES RESTO (Id_Resto),
        CONSTRAINT MenuResto_IDmenu_Fkey FOREIGN KEY (Id_Menu)
        REFERENCES MENU (Id_Menu)
        );
```

Création de la table BOISSONSERVI

```
CREATE TABLE BOISSONSERVI
        (Id_Resto Integer,
         Id_Boisson Integer,
         Id_Contenant Integer,
         Prix Numeric(6,2),
        CONSTRAINT MenuBoisson_pkey PRIMARY KEY (Id_Resto, Id_Boisson,
        Id_Contenant),
        CONSTRAINT MenuBoisson_IdResto_Fkey FOREIGN KEY (Id_Resto)
        REFERENCES RESTO (Id_Resto),
        CONSTRAINT MenuResto_IDBoisson_Fkey FOREIGN KEY (Id_Boisson)
        REFERENCES BOISSON (Id_Boisson),
        CONSTRAINT MenuResto_IDContenant_Fkey FOREIGN KEY
        (Id_Contenant) REFERENCES CONTENANT (Id_Contenant)
        );
```

Autre écriture sans nommer toutes les contraintes :

```
CREATE TABLE BOISSONSERVI
        (Id_Resto Integer REFERENCES RESTO (Id_Resto),
         Id_Boisson Integer REFERENCES BOISSON (Id_BOISSON),
         Id_Contenant Integer REFERENCES CONTENANT (Id_Contenant),
         Prix Numeric(6,2),
        CONSTRAINT MenuBoisson_pkey PRIMARY KEY (Id_Resto,
        Id_Boisson,Id_Contenant)
        );
```

4.3 Assurer la cohérence père - fils

Les instructions ON DELETE et ON UPDATE ont pour rôle d'assurer la cohérence des données et surtout de mettre en place des mécanismes permettant de garder cette cohérence.

Par exemple, on veut pouvoir effacer un resto, mais dans ce cas pour assurer l'intégrité référentielle, il est indispensable d'effacer les enregistrements correspondant à cet Idresto dans les tables MENURESTO et BOISSON SERVI

- Par défaut, le ON DELETE est restrictif, c'est à dire qu'il va empêcher l'effacement du père (idresto).
- ON DELETE CASCADE propage la suppression des enregistrements du père vers le fils
- ON DELETE SET NULL propage l'affectation de la valeur nulle aux clés étrangères. Attention, dans le cas ou une clause NOT NULL est sur la clé étrangère, cela ne sera pas possible, ce qui est le cas de notre exemple.

Attention, les expressions ON DELETE et ON UPDATE se positionnent au niveau des clés étrangères.

Le mécanisme est identique pour la mise à jour.

Les actions possibles sont :

- NO ACTION
- CASCADE
- SET NULL
- SET DEFAULT

```
CREATE TABLE BOISSONSERVI
(
        id_resto INTEGER NOT NULL,
        id_boisson INTEGER NOT NULL,
        id_contenant INTEGER NOT NULL,
        prix NUMERIC (6,2),
        CONSTRAINT menuboisson_pkey PRIMARY KEY (id_resto, id_boisson,
      id_contenant),
        CONSTRAINT boissonservi_id_boisson_fkey FOREIGN KEY (id_boisson)
          REFERENCES boisson (id_boisson) MATCH SIMPLE
          ON UPDATE NO ACTION ON DELETE NO ACTION,
        CONSTRAINT boissonservi_id_contenant_fkey FOREIGN KEY
      (id_contenant)
          REFERENCES contenant (id_contenant) MATCH SIMPLE
          ON UPDATE NO ACTION ON DELETE NO ACTION
);
```

5 Les commandes d'effacement

5.1 Effacement de table

La commande d'effacement est une instruction SQL dont la syntaxe est :

DROP TABLE nom de la table [CASCADE]

Un effacement de table consiste à effacer tous les éléments concernant la table à savoir

- Les données
- La structure
- Les contraintes attachées (index).

Un effacement de table est soumis à la contrainte d'intégrité référentielle. Le mot clé CASCADE, supprime cette contrainte avec toutes les conséquences que cela comporte.

5.2 Effacement de base

De la même manière, il est possible d'effacer une base de donnée. La syntaxe est :

DROP DATABASE nom de la base

Cette commande ne peut se faire si l'on est connecté à la base.

6 La modification des tables

La modification consiste à tout ce qui peut être changé dans la structure d'une table :

- Ajout ou suppression d'un champ
- Modification d'un type de champs
- Ajout ou suppression d'une clé primaire ou étrangère
- Ajout ou suppression d'une contrainte de domaine
- Changement du nom d'un champ
- …

Ces modifications sont d'autant plus importante que l'on peut être amené à les effectuer sans détruire les données existantes. Il faudra donc être vigilant afin de vérifier si la modification peut effectivement être réalisée qu vu des données contenues dans la table.

La syntaxe de la commande est :

ALTER TABLE <nom table> action à réaliser

L'action à réaliser peut être :

- ADD [COLUMN]
- DROP [COLUMN]
- ALTER [COLUMN]
- ADD <contrainte>
- DROP CONSTRAINT <nom de la contrainte>[RESTRICT | CASCADE]
- …

6.1 Modification sur la table

Modification du nom d'une table :

```
ALTER TABLE RESTO
    RENAME TO RESTAURANT;
```

6.2 Modification sur les champs

Ajout du champs e-mail dans la Table RESTO

```
ALTER TABLE RESTO
    ADD E_mail VARCHAR(30) DEFAULT '';
```

Suite au ADD, plusieurs champs peuvent être rajoutés :

```
ADD (E-mail VARCHAR(30), Directeur VARCHAR(30))
```

Suppression d'un champs :

```
ALTER TABLE RESTO
    DROP E_mail ;
```

Modification de la taille ou du type d'un champs

```
ALTER TABLE RESTO
    MODIFY E_mail VARCHAR(50);
```

Attention, la syntaxe MODUIFY ne fonctionne pas sur toute les tables.

6.3 Modification sur les contraintes

Suppression d'une contrainte

```
ALTER TABLE RESTO
    DROP CONSTRAINT resto_nbetoile;    (suppression de la contrainte sur le nb d'étoiles)
```

Ajout d'une contrainte

```
ALTER TABLE RESTO
    ADD CONSTRAINT resto_nbetoile CHECK (nbetoile IN (0,1,2,3));
```

Suppression d'une clé primaire

```
ALTER TABLE RESTO
    DROP CONSTRAINT resto_Pkey CASCADE;
```

De la même façon, du fait de l'intégrité référentielle, la suppression de la clé primaire n'est pas possible. le mot clé CASCADE force cette suppression avec tous les risques que cela comporte.
Attention, la suppression d'une contrainte ne supprime pas la colonne correspondante. Ici, le champ correspondant à la clé primaire est conservé.

ajout d'une clé primaire :

```
ALTER TABLE RESTO
    ADD CONSTRAINT resto_pkey PRIMARY KEY (Id_resto);
```

Ce travail sur les clés peut être nécessaire dans certain cas : Imaginez que vous vouliez agrandir un champ qui est clé primaire et clé étrangère dans une autre table. Les opérations à réaliser dans l'ordre seront :

- Suppression de la contrainte de clé étrangère (avec l'option CASCADE)
- Suppression de la contrainte de clé primaire (avec l'option CASCADE)
- modification des tailles de champs
- rétablissement des contraintes