Cinquième partie V

Le langage SQL DDL

Plan du cours

Qu'allons nous aborder dans ce cours ?

1. La création d'un schéma relationnel
2. La création d'une table
3. La suppression d'une table
4. L'ajout, le retrait et la modification d'une colonne
5. L'ajout, le retrait d'une contrainte
6. La modification des données

Création d'un schéma relationnel

- Une base de données est définie par son schéma.
- SQL propose de créer ce schéma avant de définir ses composants :

```
create schema CLICOM ;
```

- Cette instruction pourra être accompagné de divers paramètres spécifiant notamment les conditions d'autorisation d'accès
- Les schémas sont rassemblés dans un catalogue qui représente un ensemble de bases de données

Création d'un schéma relationnel

- L'approche client-serveur des SGBD SQL implique quelques démarches administratives préalables au travail sur une base de données
- L'utilisateur doit établir tout d'abord une connexion avec le serveur sur lequel elle réside :

```
connect to ADMIN\SERV01 as CON_14Juil2013_001
user 'dp' passeword 'dpclicom' ;
```

- À la fin du travail, on doit demander la déconnexion :

```
disconnect CON_14Juil2013_001 ;
```

- Remarque : souvent la connexion à la base se fait par programme via un connecteur

Création d'une table

- L'opération ci-dessous produit une table (vide, i.e., sans lignes) dont le schéma est conforme aux indications données par la requête **create table**
- On y spécifie le nom de la table et la description de ses colonnes
- On spécifiera pour chaque colonne son nom et le type de ses valeurs

```
create table CLIENT (NCLI      char(10),
                     NOM       char(32),
                     ADRESSE   char(60),
                     LOCALITE  char(30),
                     CAT       char(2),
                     COMPTE    decimal(9,2)) ;
```

Création d'une table

- SQL offre divers types de données dans la déclaration d'une colonne d'une table :
  - **SMALLINT** : entiers signés courts (16 bits)
  - **INTEGER** : entiers signés longs (32 bits)
  - **NUMERIC(p,q)** : nombres décimaux de p chiffres dont q après le point décimal ; si elle n'est pas mentionnée, la valeur de q est 0
  - **FLOAT(p)** : nombres en virgule flottante d'au moins p bits significatifs (p est optionnel)
  - **CHARACTER(p)** ou **CHAR** : chaînes de caractères de longueur fixe de p caractères
  - **CHARACTER VARYING** ou **VARCHAR(p)** : chaînes des caractères de longueur variable d'un plus p caractères
  - **BIT(p)** : chaînes de longueur fixe de p bits
  - **BIT VARYING (p)** : chaînes de longueur variable ou d'au plus p bits
  - **DATE** : dates (années, mois, jours)
  - **TIME** : instants (heures, minutes, secondes, éventuellement 1000^e de seconde)
  - **TIMESTAMP** : dates + temps
  - **INTERVAL** : intervalles en années/mois/jours entre dates ou en heures/minutes/secondes entre instants

Création d'une table

- Chaque SGBD ajoute de sa propre initiative d'autres types de données
- Citons-en trois généralement disponibles sous une forme ou sous une autre dans tous les moteurs SQL :
  1. Les Binary Large Objects (BLOB), sorte de contenants génériques pouvant acceuillir des chaînes de bit de longueur illimitée telles que des images, des vidéos, etc.
  2. Les Character Large Objects (CLOB) sont similaires mais constitués de caractères
  3. Les générateurs de valeurs, qui attribuent automatiquement la valeur suivante à la colonne lors de l'insertion d'une ligne ; ils permettent d'assigner des valeurs uniques à des colonnes servant d'identifiant technique (tel le type **sequence** d'Oracle)

Création d'une table

- Outre les types de base abstraits et peu informatif, il est possible de définir des domaines de valeurs qui rendront le schéma plus lisible et plus facile à modifier

```
create domain MONTANT decimal(9,2) ;
create domain MATRICUL char(10) ;
create domain LIBELLE char(32) ;

create table CLIENT (NCLI      MATRICULE,
                     NOM       LIBELLE,
                     ADRESSE   char(60),
                     LOCALITE  LIBELLE,
                     CAT       char(2),
                     COMPTE    MONTANT) ;
```

Création d'une table

- On peut également définir une valeur par défaut que le SGBD assignera automatiquement à une colonne (éventuellement via son domaine) lorsque l'utilisateur omettra d'en fournir une lors de la création d'une ligne

```
create domain MONTANT decimal(9,2) default 0.0 ;
. . .  CAT char(2) default 'AA' ;
. . .  DATECOM date not null default current_date ;
```

Les indentifiants

- On complétera la déclaration de la table par la clause primary key

```
create table CLIENT ( NCLI      char(10),
                      NOM       char(32),
                      ADRESSE   char(60),
                      LOCALITE  char(30),
                      CAT       char(2),
                      COMPTE    decimal(9,2)
                      primary key (NCLI)) ;

create table DETAIL ( NCOM      char(12),
                      NPRO      char(15),
                      QCOM      decimal(8),
                      primary key (NCOM, NPRO));
```

Les indentifiants

- Les autres indentifiants sont déclarés via le prédicat unique

```
create table ASSURE ( NUM_AFFIL char(10),
                      NUM_IDENT char(15),
                      NOM       char(35),
                      primary key (NUM_AFFIL),
                      unique (NUM_IDENT));
```

Les clés étrangères

- On déclare la clé étrangère et la table référencée par la clause foreign key

```
create table DETAIL (
    NCOM      char(12),
    NPRO      char(15),
    QCOM      decimal(8),
    primary key (NCOM, NPRO),
    foreign key (NCOM) references COMMANDE,
    foreign key (NPRO) references PRODUIT);
```

- Par défaut, l'identifiant visé par la clé étrangère dans la table cible est l'identifiant primaire de celle-ci. Il est toujours possible de faire viser un identifiant secondaire de la table même si cela est déconseillé

```
foreign key (PROPRIO) references
    PERSONNE(NUM_REGISTRE);
```

Les clés étrangères

- Une clé étrangère comporte autant de colonnes que l'identifiant cible et ses colonnes sont de même type
- Si au moins une des colonnes est facultative, alors il est possible de préciser via la clause match, la manière dont l'intégrité référentielle sera vérifiée en présence de valeurs nulles
  1. **match simple** : si toute les colonnes de la clé étrangère possèdent une valeur, la contrainte référentielle est évaluée, sinon elle est ignorée
  2. **match full** : si les colonnes sont toutes nulles, la contrainte est ignorée. Si elles sont toutes non nulles, elle est évaluée. Dans les autres cas, elle n'est pas satisfaite
  3. **match partial** : la contraite est évaluée pour les colonnes non nulles. La table cible doit contenir au moins une ligne dont l'identifiant comporte les valeurs non nulles de la clé étrangère

```
foreign key (NCLI, NFOURN) references
   ACHAT(NCLI, NFOURN) match full
```

Caractère obligatoire/facultatif d'une colonne

- Par défaut, i.e., si l'on ne spécifie rien, toute colonne est facultative
- Le caractère obligatoire d'une colonne se déclare avec la clause not null

```
create table OFFRE (
    NUMFL char(10) not null,
    NUMPL char(15) not null,
    PRIX  decimal(8),
    primary key (NUMFL, NUMPL),
    foreign key (NUMFL) references FOURNISSEUR,
    foreign key (NUMPL) references PIECE) ;
```

Exemple complet

```
create table CLIENT    ( NCLI      char(10) not null,
                         NOM       char(32) not null,
                         ADRESSE   char(60) not null,
                         LOCALITE  char(30) not null,
                         CAT       char(2),
                         COMPTE    decimal(9,2) not null,
                         primary key (NCLI));
create table PRODUIT   ( NPRO      char(15) not null,
                         LIBELLE   char(60) not null,
                         PRIX      decimal(6) not null,
                         QSTOCK    decimal(8) not null,
                       primary key (NPRO));
create table COMMANDE  ( NCOM      char(12) not null,
                         NCLI      char(10) not null,
                         DATECOM   date not null,
                       primary key (NCOM),
                       foreign key (NCLI) references CLIENT);
create table DETAIL    ( NCOM      char(12) not null,
                         NPRO      char(15) not null,
                         QCOM      decimal(8) not null,
                       primary key (NCOM, NPRO),
                       foreign key (NCOM) references COMMANDE,
                       foreign key (NPRO) references PRODUITS);
```

Forme synthétique des contraintes

- Il existe aussi une forme synthétique pour spécifier les contraintes

```
create table COMMANDE (
    NCOM     char(12) not null primary key
    NCLI     char(10) not null references CLIENT,
    DATECOM  date not null);
```

Suppression d'une table

- Toute table peut être supprimée : elle sera inconnue du SGBD et son contenu sera perdu

```
drop table DETAIL ;
```

- Les lignes d'une table sont préalablement supprimées avant sa suppression
- La suppression des lignes d'une table est soumise aux contraintes référentielles
- Par exemple, la suppression de la table COMMANDE pourrait entraîner la supression de toutes les lignes de la table DETAIL (mais pas de la table)

Ajout, retrait et modification d'une colonne

- La commande suivante ajoute la colonne POIDS à la table PRODUIT :

```
alter table PRODUIT add column POIDS smallint ;
```

  - la table possède une nouvelle colonne qui ne contient que des valeurs nulles
  - On ne peut ajouter une colonne obligatoire que si la table est vide, ou si celle-ci possède une valeur par défaut

- La commande suivante supprime la colonne PRIX de la table PRODUIT :

```
alter table PRODUIT drop column PRIX ;
```

- Il est également possible de modifier une caractéristique d'une colonne :

```
alter table CLIENT alter column CAT set default '00' ;
```

- Un domaine peut être modifié et supprimé lorsqu'il n'est plus utilisé :

```
alter domain MONTANT set default -1.0 ;
drop domain MATRICULE ;
```

Ajout, retrait d'une contrainte

- Il est possible d'ajouter ou de retirer des contraintes d'intégrité a posteriori en utilisant la commande alter table

```
alter table CLIENT add primary key (NCLI) ;
alter table CLIENT add unique (NOM, ADRESSE, LOCALITE) ;
```

  - la demande d'ajout d'un identifiant sera refusée si les données que la table contient déjà violent cette propriété

- Une colonne obligatoire peut être déclarée facultative et inversement si les données le permettent

```
alter table CLIENT alter CAT not null ;
alter table CLIENT alter ADRESSE null ;
```

Ajout, retrait d'une contrainte

- Une clé étrangère peut être définie ou retirée après création de la table source

```
alter table COMMANDE
add foreign key (NCLI) references CLIENT ;
```

  - Cette possibilité est indispensable lorsque le SGBD n'accepte pas la déclaration d'une clé étrangère vers une table non encore définie (ce qu'on appelle une référence en avant)

Ajout, retrait d'une contrainte

- Lors de la déclaration d'une contrainte, il est possible de donnéer un nom à celle-ci avec la clause constraint<nom>

```
create table DETAIL (NCOM char(12) not null,
                     NPRO char(15) constraint C1 not null,
                     QCOM decimal(8),
       constraint C2 primary key (NCOM, NPRO),
       constraint C3 foreign key (NPRO) references PRODUIT) ;
alter table CLIENT
add constraint C_CLI_U unique (NOM, ADRESSE, LOCALITE) ;
alter table COMMANDE
add constraint C5 foreign key (NCLI) references CLIENT ;
```

- La suppression d'une contrainte s'effectue de la manière suivante :

```
alter table DETAIL
drop constraint C2 ;
```

Modifications des données

- On va présenter les opérateurs qui permettent
  - L'introduction de données (instruction insert)
  - La suppression de données (instruction delete)
  - La modification de données (instruction update)
- On va regarder aussi le comportement du SGBD lors des mises à jour en présence de contraintes référentielles

Modifications des données

- L'ajout d'une ligne s'effectue avec l'instruction insert values

```
insert into DETAIL values ('30185', 'PA45', 12) ;
```

- L'ordre des valeurs est celui de la déclaration des colonnes lors de la création de la table
- Lorsque toutes les valeurs ne sont pas introduites, il faut préciser le nom et l'ordre des colonnes :

```
insert into CLIENT (NCLI, NOM, ADRESSE, COMPTE,
LOCALITE) values ('C402', 'BERNIER',
'28 avenue de France', -2500, 'Lausanne') ;
```

- Toute colonne non spécifiée, telle que CAT, prend la valeur nulle, ou la valeur par défaut
- Toute colonne obligatoire (not null) doit recevoir une valeur sauf si on lui a assigné une valeur par défaut

Modifications des données

- Il est possible d'insérer dans une table existante des données extraites d'une autre table
- Ces données sont obtenues par une expression SFW attachée à l'instruction insert
- Exemple :

```
INSERT INTO CLIENT_TOULOUSE
SELECT NCLI, NOM, ADRESSE
FROM CLIENT
WHERE LOCALITE = 'Toulouse' ;
```

  - La table CLIENT_TOULOUSE est indépendante de la table CLIENT
  - On appelle parfois ses tables *snapshot*. Elles sont utilisées pour distribuer l'information à partir d'une base de données centrale et pour constituer des systèmes d'aide à la décision (les entreprôts de données)

Modifications des données

- L'ordre de suppression porte sur un sous-ensemble des lignes d'une table
- Les lignes à supprimer sont désignées par la clause WHERE dont le format est le même que celui de l'instruction SWF :

```
DELETE FROM CLIENT
WHERE NCLI = 'K111' ;
```

  - La commande supprime de CLIENT la ligne correspondant au client K111
- L'instruction ci-dessous supprime de la table DETAIL les lignes, quel qu'en soit le nombre, qui spécifient un produit en rupture de stock

```
DELETE FROM DETAIL WHERE NPRO IN
(SELECT NPRO FROM PRODUIT WHERE QSTOCK <= 0) ;
```

- Remarque : Après toute opération de suppression, la base de données doit être dans un état qui respecte les contraintes d'intégrité

Modifications des données

- Comme la suppression, la modification est effectuée sur toutes lignes qui vérifient une condition de sélection :

```
UPDATE CLIENT
SET ADRESSE = '29, av. de la Magne',
    LOCALITE = 'Niort'
WHERE NCLI = 'F011' ;
```

- Les nouvelles valeurs peuvent être obtenues par une expression arithmétique

```
UPDATE PRODUIT
SET PRIX = PRIX * 1.05
WHERE LIBELLE like '%SAPIN%' ;
```

Modifications des données

- Les données modifiées peuvent provenir également de la base de données elle-même

```
UPDATE PRODUIT P
SET QSTOCK = QSTOCK - (SELECT SUM(QCOM)
  FROM DETAIL WHERE NPRO = P.NPRO)
WHERE EXISTS (SELECT * FROM DETAIL
              WHERE NPRO = P.NPRO) ;
```

- La requête déduit de la quantité en stock de chaque produit les quantités actuellement en commande

Modifications des données

- Une requête de modification du contenu de la base de données (INSERT, DELETE, UPDATE, etc.) n'est exécutée que si le résultat respecte toutes les contraintes d'intégrité définies sur cette base
- Une opération dont l'exécution laisse les données dans un état invalide est refusée
- Considérons les tables CLIENT et COMMANDE :

```
create table CLIENT   ( NCLI     char(10) not null, . . .
                        primary key (NCLI));
create table COMMANDE ( NCOM     char(12) not null,
                        NCLI     char(10) not null, . . .
                      primary key (NCOM),
                      foreign key (NCLI) references CLIENT);
```

Lors de la suppression d'une ligne de la table CLIENT, trois types de comportements sont possibles :

1. Blocage
2. Propagation
3. Découplage

Modifications des données

- Lors qu'un blocage se produit la suppression est refusée
- Un blocage se produit s'il existe une ou plusieurs lignes de COMMANDE dépendantes, i.e., dont COMMANDE.NCLI = CLIENT.NCLI
- On définit se comportement comme suit :

```
create table COMMANDE ( NCOM     char(12) not null,
                        NCLI     char(10) not null, . . .
                      primary key (NCOM),
                      foreign key (NCLI) references CLIENT
                                         on delete no action) ;
```

Modifications des données

- Lors qu'une propagation se produit la suppression est acceptée
- Une propagation entraîne la supression conjointe des lignes de COMMANDE dépendantes
- On définit se comportement comme suit :

```
create table COMMANDE ( NCOM     char(12) not null,
                        NCLI     char(10) not null, . . .
                      primary key (NCOM),
                      foreign key (NCLI) references CLIENT
                                         on delete cascade) ;
```

Modifications des données

- Lors qu'un découplage se produit la suppression est acceptée
- Un découplage entraîne l'attribution de la valeur nulle à la colonne NCLI, i.e., les données sont rendues indépendantes
- On définit se comportement comme suit :

```
create table COMMANDE ( NCOM     char(12) not null,
                        NCLI     char(10) not null, . . .
                      primary key (NCOM),
                      foreign key (NCLI) references CLIENT
                                         on delete set null) ;
```

- Remarque : ce mode sera notamment utilisé pour les clés étrangères cycliques

Modifications des données

- La mise à jour de valeur (UPDATE) est régit par les mêmes principes
- Par exemple :

```
create table COMMANDE ( NCOM     char(12) not null,
                        NCLI     char(10) not null, . . .
                      primary key (NCOM),
                      foreign key (NCLI) references CLIENT
                                         on delete set no action
                                         on update cascade) ;
```

Modifications des données

- Le mode d'une clé étrangère peut influencer celui d'une autre clé étrangère

```
create table CLIENT    ( NCLI     char(10) not null, . . .
                                  primary key (NCLI)) ;
create table COMMANDE ( NCOM     char(12) not null,
                        NCLI     char(10) not null, . . .
                      primary key (NCOM),
                      foreign key (NCLI) references CLIENT
                                         on delete cascade) ;
create table DETAIL    ( NCOM     char(12) not null,
                        NPRO     char(15) not null, . . .
                      primary key (NCOM, NPRO),
                      foreign key (NCOM) references COMMANDE,
                                         on delete no action) ;
```

- Lors de la suppression d'une ligne CLIENT, le SGBD tente de supprimer en cascade les lignes de COMMANDE qui en dépendent
- Si une des lignes COMMANDE est associée à une ligne DETAIL, sa supression est refusée et donc la suppression de CLIENT
- De ce schéma, on déduit qu'on ne peut supprimer un client que s'il n'a aucune commande qui possède des détails

Modifications des données

- Une même requête exécutée sur une même base de données produit elle toujours le même résultat ?
- La réponse devrait être résolument positive
- La réalité oblige à nuancer cette certitude
- Les comportements liés à la modifciation des données posent des problèmes délicats parmi lesquels
  1. L'interaction entre colonnes lors d'une modification
  2. La suppression dans une structure cyclique
  3. L'interférence entre les modes *on delete* qui s'applique sur les clés étrangères

Modifications des données

- L'effet d'une modification faisant intervenir les mêmes colonnes à gauche et à droite des clauses set dépend malheureusement du SGBD

```
UPDATE CLIENT
SET ADRESSE = LOCALITE, LOCALITE = ADRESSE
```

- Avec MySQL ou InterBase, cette requête copiera, pour chaque ligne la valeur de LOCALITE dans la colonne ADRESSE, détruisant les valeurs de cette dernière colonne
- Avec Oracle ou PostgreSQL, elle échangera les valeurs de ces deux colonnes pour chaque ligne
- La différence est de taille !

Modifications des données

- Considérons le cas de la table PERSONNE

```
create table PERSONNE (NPERS       char(4) not null,
                       NOM         char(25) not null,
                       RESPONSABLE char(4)
            primary key (NPERS),
            foreign key (RESPONSABLE) references PERSONNE
                                      on delete no actions) ;
```

- Imaginons 2 requêtes excutées dans l'ordre suivant avec p2 le responsable de p8 et p2 sans responsable :

```
DELETE FROM PERSONNE WHERE NPERS = 'p2' ;
DELETE FROM PERSONNE WHERE NPERS = 'p8' ;
```

- La première requête va échoué tandis que la seconde va réussir
- Si l'on permute l'ordre maintenant les deux requêtes vont réussir

Modifications des données

- Considérons maintenant la requête suivante

```
DELETE FROM PERSONNE WHERE NPERS IN ('p2', 'p8') ;
```

- Tout dépendant de l'ordre dans lequel les suppressions ont lieu
- Il existe 3 stratégies d'exécution de cette requête
  1. La stratégie séquentielle : Le SGBD parcourt les lignes vérifiants la condition et les supprime au fur et à mesure. Si une contrainte est violé, la suppression s'arrête. Etant donné que le stockage est aléatoire, la requête a un comportement non déterminsite. Ce comportement est implanté dans MySQL, Intervase, etc.
  2. La stratégie snapshot : Le SGBD construit l'ensembles des lignes vérifiant la condition. Si une ligne est dépendante, la suppression est annulée. Le comportement est déterministe
  3. La stratégie point fixe : Le SGBD construit l'ensembles des lignes vérifiant la condition puis parcourt autant de fois qu'il est nécessaire cet ensemble afin d'en supprimer toutes les lignes dépendantes. Ce comportement déterministe est implanté dans Oracle, DB2, PostgreSQL, etc.

Modifications des données

- Considérons le cas suivant :

```
create table A (IA primary key, . . .) ;
create table B (IB primary key, RA, . . .
       foreign key (RA) references A on delete cascade) ;
create table C (IC primary key, RA, . . .
       foreign key (RA) references A on delete cascade) ;
create table C (IC primary key, RB, RC, . . .
       foreign key (RB) references B on delete cascade,
       foreign key (RC) references C on delete no action) ;
```

- Supposons que l'on supprime une ligne de A dépendante de B et de C et qu'une ligne de C dépendante soit elle-même dépendante de D
- Si la suppression se fait dans l'ordre A → B → C → D toutes les lignes seront supprimées
- Si la suppression se fait dans l'ordre A → C → D → B aucune ligne ne sera supprimée

Exercice

- Exprimer en SQL les requêtes suivantes :
  1. Créer une table temporaire DETAIL_COM et y ranger les données suivantes relatives aux détails de commande : numéro et date de commande, quantité commandée, numéro et prix du produit, montant du détail
  2. Annuler les comptes négatifs des clients de catégorie C1

Exercice

1. Créer une table temporaire DETAIL_COM et y ranger les données suivantes relatives aux détails de commande : numéro et date de commande, qunatité commandée, numéro et prix du produit, montant du détail

```sql
CREATE TABLE DETAIL_COM (NCOM ..., DATECOM ..., ..., MONTANT ...) ;
INSERT INTO DETAIL_COM (NCOM, DATECOM, QCOM, NPRO, PRIX, MONTANT)
SELECT M.NCOM, DATECOM, QCOM, P.NPRO, PRIX, QCOM*PRIX
FROM COMMANDE M, DETAIL D, PRODUIT P
WHERE M.NCOM = D.NCOM AND D.NPRO = P.NPRO ;
```

2. Annuler les comptes négatifs des clients de catégorie C1

```sql
UPDATE CLIENT SET COMTE = 0
WHERE COMPTE < 0 AND CAT = 'C1' ;
```