Bases de données relationnelles

Le modèle relationnel est mis en oeuvre par un logiciel particulier, *le système de gestion de bases de données* relationnel (SGBD relationnel ou SGBDR). La très grande majorité des SGBD relationnelles utilisent le langage SQL (*Structured Query Language*) qui permet d'envoyer des *ordres* au SGBD. Ces ordres peuvent être de deux natures :

- Les *mises à jour* permettent la création de relations, l'ajout de données dans ces dernières, leur modification et leur suppression.
- Les *requêtes* permettent de récupérer les données répondant à des critères particuliers.

1 Mises à jour de tables

1.1 Création de tables

La commande `CREATE TABLE` permet de créer une table (relation) en définissant le nom et le type des attributs.
Pour créer les tables correspondants aux relations de la modélisation du CDI, on peut saisir les ordres suivants :

```
CREATE TABLE eleve (num_etudiant INT PRIMARY KEY,
                    nom VARCHAR(90), prenom VARCHAR(90)
                    classe VARCHAR(30));

CREATE TABLE livre (isbn CHAR(14) PRIMARY KEY,
                    siret INT REFERENCES editeur(siret),
                    titre VARCHAR(300), annee INT);

CREATE TABLE auteur (a_id INT PRIMARY KEY,
                     nom VARCHAR(90),
                     prenom VARCHAR(90));

CREATE TABLE ecrire (a_id REFERENCES auteur(a_id),
                     isbn CHAR(14) REFERENCES livre(isbn),
                     PRIMARY KEY(a_id,isbn));

CREATE TABLE emprunt (num_etudiant INT REFERENCES eleve(num_etudiant),
                      isbn CHAR(14) REFERENCES livre(isbn),
                      retour DATE,
                      PRIMARY KEY(isbn));

CREATE TABLE editeur (siret INT PRIMARY KEY,
                      nom VARCHAR(80));
```

1.2 Les différents types de données

Types numériques

nom du type	exact/approché	description
SMALLINT	exact	entier 16 bits signé
INTEGER	exact	entier 32 bits signé
INT	exact	alias pour INTEGER
BIGINT	exact	entier 64 bits signé
DECIMAL(t, f)	exact	décimal signé de t chiffres dont f après la virgule
REAL	approché	flottant 32 bits
DOUBLE PRECISION	approché	flottant 64 bits

Types textes

nom du type	description
CHAR(n)	chaîne d'exactement n caractères, les caractères manquant sont complétés par des espaces
VARCHAR(n)	chaîne d'au plus n caractères
TEXT	chaîne de taille quelconque

Types dates

nom du type	description
DATE	une date au format 'AAAA-MM-JJ'
TIME	une heure au format 'hh :mm :ss'
TIMESTAMP	un instant (date et heure) au format 'AAAA-MM-JJ hh :mm :ss'

Une fonctionnalité intéresssante des types dates est la possibilité d'ajouter ou soustraire une durée. Par exemple si `d` est de type `DATE` alors `d+10` est de type `DATE` et représente la date 10 jours après `d`.

1.3 Les contraintes d'intégrité

Clé primaire. Les mots clés `PRIMARY KEY` permettent d'indiquer qu'un attribut est clé primaire :

```
CREATE TABLE eleve (num_etudiant INT PRIMARY KEY,
                    nom VARCHAR(90), prenom VARCHAR(90)
                    classe VARCHAR(30));
```

Si plusieurs attributs sont forment la clé primaire, on peut spéfier la contrainte après les attributs :

```
CREATE TABLE ecrire (a_id REFERENCES auteur(a_id),
                     isbn CHAR(14) REFERENCES livre(isbn),
                     PRIMARY KEY(a_id,isbn));
```

Clé étrangère. Un attribut peut être qualifié de clé étrangère avec le mot clé `REFERENCES` suivi du nom de la table où se trouve la clé primaire et de son nom.

```
CREATE TABLE livre (isbn CHAR(14) PRIMARY KEY,
                    siret INT REFERENCES editeur(siret),
                    titre VARCHAR(300), annee INT);

CREATE TABLE ecrire (a_id REFERENCES auteur(a_id),
                     isbn CHAR(14) REFERENCES livre(isbn),
                     PRIMARY KEY(a_id,isbn));
```

Unicité, non nullité. Il peut être intéressant de spécifier qu'un attribut est unique, sans pour autant en faire une clé primaire. Cela peut être spécifié à l'aide du mot clé `UNIQUE`.

```
CREATE TABLE eleve (num_etudiant INT PRIMARY KEY,
                    nom VARCHAR(90), prenom VARCHAR(90),
                    classe VARCHAR(30),
                    email VARCHAR(60) UNIQUE);
```

Une autre pratique consiste à déclarer qu'un attribut ne peut être vide (`NULL`) à laide des mots clés `NOT NULL`.

```
CREATE TABLE eleve (num_etudiant INT PRIMARY KEY,
                    nom VARCHAR(90) NOT NULL,
                    prenom VARCHAR(90) NOT NULL,
                    classe VARCHAR(30) NOT NULL,
                    email VARCHAR(60) NOT NULL UNIQUE);
```

Contraintes utilisateurs. Il est possible de spécifier des contraintes arbitraires sur les attributs d'une même ligne au moyen du mot clé `CHECK`.

```
CREATE TABLE produit (pid INT PRIMARY KEY,
                      nom VARCHAR(100) NOT NULL,
                      quantite INT NOT NULL,
                      prix DECIMAL(10,2) NOT NULL,
                      CHECK (quantite >= 0 AND prix >= 0));
```

2 Modification de tables

2.1 Suppression de tables

Pour recréer une table, par exemple avec un schéma différent, il faut d'abord supprimer celle portant le même nom à l'aide de l'instruction `DROP TABLE`.

```
DROP TABLE ecrire;
```

Il n'est pas possible de supprimer une table si elle sert de référence pour une clé étrangère d'une autre table, car cela violerait une contrainte de référence :

```
DROP TABLE auteur;
```

```
constraint failed
```

Il convient donc de supprimer les tables dans le bon ordre : d'abord les tables contenant les clé étrangères puis celles contenant les clés primaires référencées.

2.2 Insertion dans une table

Insérer une valeur dans une table s'effectue grâce à l'instruction `INSERT INTO`.

```
INSERT INTO auteur VALUES (1,'Kundera','Milan'),
                          (2,'Philip','K.Dick'),
                          (3,'Gingras','Yves');
```

Si on souhaite passer les valeurs dans un autre ordre, il faut le spécifier.

```
INSERT INTO auteur VALUES(prenom, a_id, nom) VALUES
                        ('Jean-Jaques',4,'Rousseau');
```

Les contraintes d'intégrités sont vérifiées au moment de l'insertion.

```
INSERT INTO auteur VALUES (1,'Dumas','Alexandre');
```

```
UNIQUE constraint failed: auteur.a_id
```