Chapitre 15

Bases de données

Une base de données (BD) est un ensemble structuré d'informations. Dans le langage courant, le terme peut désigner toute source importante de données telle qu'une encyclopédie. Dans le domaine de l'informatique, c'est un ensemble d'informations regroupées sous la forme d'enregistrements stockés sur un système de fichiers structurés et organisées de manière à pouvoir être facilement manipulées.

L'organisation logique des données se fait selon un modèle de données et la structure physique des fichiers comporte des index destinés à accélérer les opérations de recherche et de tri. Le modèle de données relationnel est aujourd'hui le plus utilisé parce qu'il permet l'indépendance entre la structure physique et l'organisation logique des données.

Le logiciel qui manipule les bases de données est appelé système de gestion de base de données (SGBD). Il permet d'organiser, de contrôler, de consulter et de modifier la base de données. Les opérations sont parfois formulées dans un langage de requête tel que SQL, qui est le plus connu et employé pour les modèles relationnels.

Quatre exemples de bases de données

Tous les exemples ci-dessous proviennent du site www.apprendre-en-ligne.net.

Seshat

On peut utiliser des bases de données depuis des pages web grâce au langage PHP, couplé avec MySQL.

Seshat est une base de données de mathématiciens. Comme vous pourrez le constater, cette base de données permet de retrouver rapidement un mathématicien selon divers critères : nom, année de naissance, année de mort, jour de naissance, ville de naissance ou de mort, signe astrologique, pays, etc. Elle permet aussi de trouver tous les contemporains d'un mathématicien. La mise en page peut aussi se changer aisément et est la même pour tous les mathématiciens.

En fait, les pages que vous verrez n'existent pas réellement : elles sont construites à chaque requête d'après les informations figurant dans la base de données. De même, la chronologie des mathématiciens est construite en visitant toute la base de données. Cela signifie que si l'on ajoute ou enlève un mathématicien à la base de données, le tableau chronologique sera automatiquement modifié lors de la prochaine requête.

Les blogs

Les blogs sont aussi basés sur une base de données. Les contenus des billets sont stockés dans un champ, ainsi que des informations comme le sujet, le nombre de lectures, la date, etc. Quand on ouvre le blog, le système affiche les *n* derniers billets, par ordre antéchronologique.

Les quiz interactifs

Pour les quiz aussi, les questions sont stockées dans une base de données, avec les réponses, la difficulté, le chapitre concerné, etc. Les questions sont ensuite choisies par le système au hasard ou selon certains critères.

Le défi Turing

Le défi Turing est une base de données d'exercices de programmation. Les questions sont contenues dans une base de données, ainsi que les membres du défi. On garde en mémoire les problèmes résolus par chaque membre, ce qui permet d'établir un classement. Enfin, un forum permet aux membres de partager leurs solutions : c'est aussi géré avec une base de données.

15.1. Un exemple qui explique beaucoup de choses

Replongeons-nous avant les années 1980, lorsque la téléphonie mobile n'existait pas. Nous allons imaginer un opérateur de téléphonie qui veut créer une base de données pour gérer ses clients. Les clients peuvent obtenir plusieurs numéros de téléphone auprès de cet opérateur. Pour chacun de ces numéros, l'opérateur enregistrera les informations nécessaires pour établir une facture mensuelle détaillée.

Quelles sont ces données nécessaires ?

Il faudra tout d'abord identifier de manière non équivoque le client : on enregistrera donc son nom, son prénom et son adresse. On stockera aussi le numéro de téléphone qu'il a utilisé (rappelons qu'il peut en avoir plusieurs ; on fera une facture par numéro).

Il faudra aussi connaître, pour calculer les prix de la communication, le numéro appelé, la date et l'heure de l'appel (pour établir la facture détaillée) ainsi que la durée de l'appel. Enfin, comme le prix de l'appel peut varier selon l'heure et le numéro appelé, on mentionnera le tarif appliqué.

15.1.1. Première approche : un tableur

Une première idée pour implémenter une base de données consiste à utiliser un tableur. En effet, une base de données « plate » peut se voir comme un simple tableau : les colonnes représenteront des champs (nom, prénom, numéro de téléphone, etc.), et les lignes des enregistrements (ici des appels). Par exemple, voici un extrait de cette base de données :

Nom	Prénom	Adresse	Numéro de tél.	No appelé	Date et heure de l'appel	Durée de l'appel (sec)	Tarif (CHF/min)
Camé	Léon	Petit-Chêne 3 2900 Porrentruy	0324660010	00333246634217	10.2.2010 14:32	455	0.70
Camé	Léon	Petit-Chêne 3 2900 Porrentruy	0324660010	0324711230	17.2.2010 18:37	62	0.20
Camé	Léon	Petit-Chêne 3 2900 Porrentruy	0324669987	0324711230	23.2.2010 9:21	145	0.20
Camé	Léon	Grand-Rue 49 2800 Delémont	0324221022	0214537124	12.2.2010 19:01	302	0.10
...	...	...	...	...	...	...	

L'avantage de cette méthode est sa simplicité : tout ce qui concerne un appel tient sur une ligne, et tous les appels sont répertoriés dans une table.

Cette approche a aussi des inconvénients :

1. Les informations sont redondantes : l'adresse du client apparaît plusieurs fois, encombrant inutilement la mémoire. Il aurait été plus judicieux d'avoir une autre table où l'on aurait enregistré les coordonnées des clients une fois pour toutes.
2. Pour identifier un client, on a besoin de trois champs. Il aurait été plus simple d'associer à chaque client un numéro de client unique (ce que toutes les entreprises font).
3. Si un utilisateur change d'adresse, il faut reporter ce changement sur chaque ligne où il apparaît, sous peine d'avoir une table incohérente.

En fait, un tableur n'est tout simplement pas fait pour gérer une grande base de données...

15.1.2. Deuxième approche : base de données relationnelle

Une autre solution est d'utiliser plusieurs tables « reliées » entre elles. On parle alors de base de données relationnelle. La liaison de différentes tables se fera en utilisant un logiciel de gestion de bases de données (SGBD).

Reprenons l'exemple de l'entreprise de téléphonie. On va utiliser trois tables au lieu d'une : la première contiendra les coordonnées des clients, la seconde les numéros de téléphone des clients, et la troisième la liste des appels.

Id_client	Nom	Prénom	Adresse
156	Camé	Léon	Grand-Rue 49 2800 Delémont
10234	Camé	Léon	Petit-Chêne 3 2900 Porrentruy
...	...	...	...

Table des clients

Numéro	Id_client
0324221022	156
0324660010	10234
0324669987	10234
...	...

Table des numéros

La table des numéros permet de faire la liaison entre un abonné et son ou ses numéros de téléphone. Elle est indispensable du fait qu'un abonné peut avoir plusieurs numéros de téléphone différents.

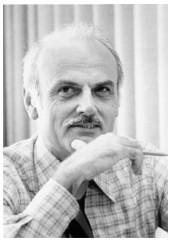
Si cela n'avait pas été le cas, on aurait simplement pu ajouter un champ « numéro » à la table des clients. Ce champ aurait alors pu être utilisé comme identifiant et remplacer le champ « Id_Client ».

Id_appel	Numéro appelant	No appelé	Date et heure de l'appel	Durée de l'appel (sec)	Tarif (CHF / min)
123465	0324660010	00333246634217	10.2.2010 14:32	455	1.00
145674	0324660010	0324711230	17.2.2010 18:37	62	0.20
162346	0324669987	0324711230	23.2.2010 9:21	145	0.20
124369	0324221022	0214537124	12.2.2010 19:01	302	0.10
	...	...	...	...	...

Table des appels

Cette manière de faire est moins lisible au premier coup d'œil : il faut par exemple consulter deux tables pour connaître le propriétaire d'un numéro de téléphone. Mais cet inconvénient est vite balayé par les avantages :

1. Les informations ne sont plus redondantes : par exemple, l'adresse d'un client apparaît une seule fois et est donc facilement modifiable.
2. Un client est identifié par un numéro, ce qui évite des confusions.
3. Il est très facile de tirer la liste des clients, puisqu'ils sont tous rassemblés dans une table.



Edgar Frank **Codd**
(1923-2003)

15.2. Les bases de données relationnelles

C'est l'article d'Edgar Frank **Codd** *« A Relational Model of Data for Large Shared Data Banks », CACM 13, No. 6, June 1970* qui fonde le modèle relationnel, mais une première description de ce modèle avait déjà été publiée l'année précédente dans un rapport technique d'IBM : *« Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks », IBM Research Report RJ599*.

Tables

Dans les bases de données relationnelles, une table est un ensemble de données organisées sous forme d'un tableau où les colonnes correspondent à des champs et les lignes à des enregistrements, également appelés entrées.

La notion de table est apparue dans les années 1970 chez IBM avec l'algèbre relationnelle qui est une théorie mathématique en relation avec la théorie des ensembles. Cette théorie a pour but de clarifier et de faciliter l'utilisation d'une base de données.

Conception d'une table

Lors de la conception d'une base de données relationnelle, il est important de clairement définir toutes les tables qui la composeront et les différentes relations qui les lient, de manière à pouvoir dresser le schéma conceptuel qui permettra de décrire le fonctionnement de la base de données avant de la mettre « physiquement » en place.

On distinguera les tables courantes, qui contiendront divers champs contenant des informations, et les tables de liaison, qui feront les liens entre deux tables courantes.

Contenu d'une table

Par nature, chaque colonne d'une table, également nommé « champ », doit contenir des données d'un même type et ce champ doit être nommé. Chaque table est l'implémentation physique d'une relation entre les différents champs. Chaque correspondance est définie par une ligne de la table. Il y a certaines règles à respecter, notamment le fait qu'il faille mettre un identifiant pour chaque enregistrement dans la table. Il y a deux possibilités :

- Mettre un identifiant qui s'auto-incrémente au fur et à mesure des données entrées (par exemple « Id_appel » dans la « table des appels »)
- Choisir un identifiant unique (par exemple « Id_client » dans la « table des clients »).

Travail sur une table

Il y a deux niveaux de travail sur une table :

- un niveau de définition des données d'une table, qui permet de définir, lier, et contraindre les données via un langage de définition de données ;
- un niveau de manipulation des données d'une table, qui permet d'ajouter, supprimer, rechercher des données via un langage de manipulation de données

Actuellement, le langage standardisé pour travailler sur les tables est le SQL. Il est utilisé avec quelques variantes sur la plupart des systèmes de gestion de bases de données. Nous en reparlerons au § 15.5.

Peter Pin-Shan
Chen
(né en 1947)

15.3. Modèle entité-association

Le modèle entité-association (aussi appelé « modèle entité-relation ») est un type de schéma conceptuel très utilisé pour les bases de données, notamment les bases de données relationnelles. Il a été inventé par Peter Pin-Shan **Chen** en 1975 et est destiné à clarifier l'organisation des données dans les bases de données relationnelles en notifiant :

- les entités

- Ce sont des objets concrets que l'on peut identifier (client, livre, individu, voiture, etc.).
- On peut représenter un ensemble d'entités de la réalité par une entité type (un client pour l'ensemble des clients).

- Ces entités sont caractérisées par leurs attributs (pour un client : nom, prénom, adresse, ...). Parmi ces attributs, on définit un identifiant qui va permettre de caractériser de façon unique l'entité dans l'ensemble (un numéro de client).

- les relations entre les entités

- Elles représentent les liens existant entre une ou plusieurs entités.
- Elles sont caractérisées par un nom, une propriété d'association et éventuellement des attributs.

- le degré de relation et cardinalités

- Le degré de la relation (ou dimension de la relation) est le nombre d'entités qui sont impliquées dans cette relation.
- La cardinalité (d'une entité par rapport à une relation) exprime le nombre de participations possibles d'une entité à une relation. Comme c'est un nombre variable, on note la cardinalité minimum (0 ou 1) et maximum pour chaque entité.
  Dans notre exemple, un abonné peut avoir de 1 à n numéros de téléphone, mais le numéro de téléphone n'appartient qu'à un client ou aucun client (numéro pas encore attribué ou plus attribué). Il ne peut pas avoir 0 numéro, car dans ce cas il ne serait plus client de l'opérateur.

15.3.1. Représentation graphique

- Les entités sont représentées dans des rectangles et s'écrivent en lettres majuscules.
- L'identifiant d'une entité (aussi appelé « clé primaire ») est le premier attribut cité et est souligné. Les autres attributs sont placés à la suite.
- Les relations sont placées dans des losanges avec leurs attributs éventuels.
- Les cardinalités sont placées à côté de l'entité qu'elles caractérisent.

15.3.2. Démarche de conception

Voici une méthode possible pour réaliser un schéma entité-association :

1. établir la liste des entités ;
2. déterminer les attributs de chaque entité en choisissant un identifiant ;
3. établir les relations entre les différentes entités et définir les cardinalités.

15.3.3. Exemple

Reprenons l'exemple de notre opérateur de téléphonie fixe.

1. Il y a 3 entités : les clients, les numéros de téléphone et les appels.

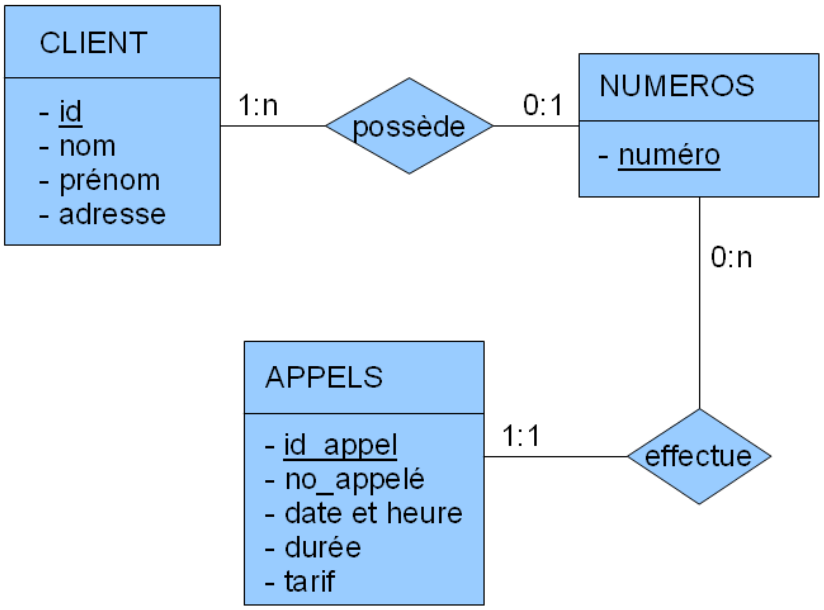
2. Les attributs :

   - Pour les clients : id, nom, prénom, adresse.
   - Pour les numéros de téléphone : le numéro.
   - Pour les appels : le numéro appelé, la date, la durée et le tarif.

3. Les relations et les cardinalités :

   - Un client possède 1 ou plusieurs numéros.
   - Un numéro appartient à 0 ou 1 client.
   - Un numéro peut faire 0 ou plusieurs appels.
   - Un appel est fait depuis 1 et un seul numéro.

Représentation graphique



On peut classer les relations en 3 types selon leur cardinalité maximale :

- les relations un à un (par exemple 1:1 – 0:1)
- les relations un à plusieurs (c'est le cas des deux relations ci-dessus)
- les relations plusieurs à plusieurs (par exemple 0:n – 1:n)

Exercice 15.1

On veut créer une petite base de données permettant de garder le contact avec nos copains de classe. On supposera qu'ils sont tous domiciliés en Suisse, qu'ils n'ont qu'un numéro de téléphone, mais éventuellement plusieurs adresses. On veut stocker les renseignements suivants : nom, prénom, sexe, date de naissance, numéro de téléphone, rue, numéro postal, ville et canton.

Réalisez un schéma entité-association en suivant la démarche décrite au § 15.3.2.

Exercice 15.2

On veut créer une base de données permettant de gérer les clients étrangers d'une entreprise et les pays de ces clients. La table des clients sera simplifiée et comportera leur nom, leur prénom, le pays de résidence et le solde de leur compte, dans la monnaie du pays.

On veut aussi que les clients aient la possibilité de faire des réclamations et que la base de données gère toutes ces réclamations pour chacun des clients.

Réalisez un schéma entité-association en suivant la démarche décrite au § 15.3.2.

Exercice 15.3

Une école veut informatiser ses listes de classe. Une classe est formée d'élèves (toujours les mêmes et un élève n'appartient qu'à une seule classe). Il y a un maître de classe par classe. Un prof peut enseigner plusieurs disciplines.

Réalisez un schéma entité-association en suivant la démarche décrite au § 15.3.2.

15.4. Traduction du schéma conceptuel en un modèle relationnel

Les règles principales de transformation d'un schéma conceptuel entité-association en un schéma relationnel sont :

Règle I

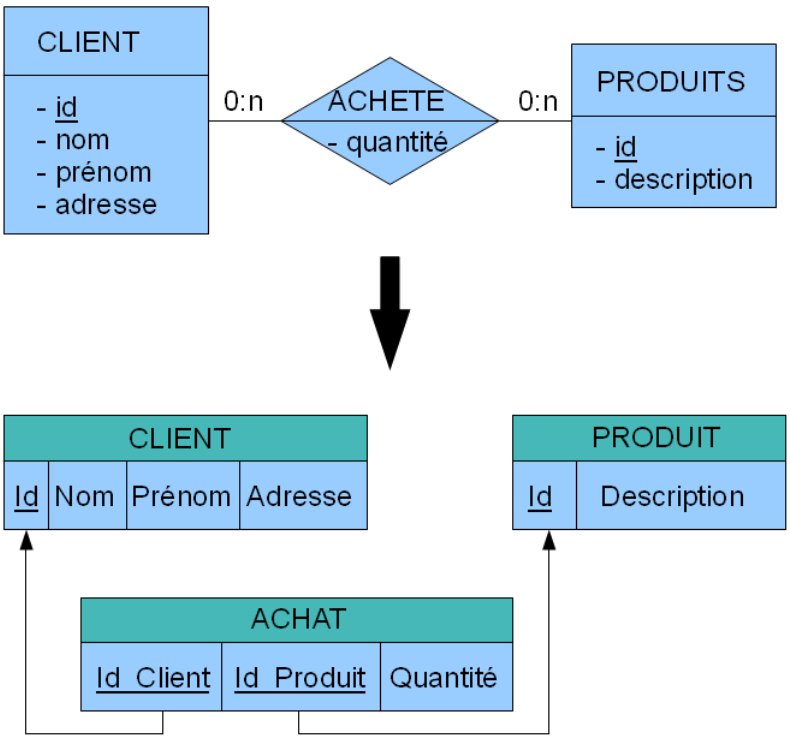
Toute entité est traduite en une table relationnelle dont les caractéristiques sont les suivantes :

- le nom de la table est le nom de l'entité ;
- la clé de la table est l'identifiant de l'entité ;
- les autres attributs de la table forment les autres colonnes de la table.

Règle II

Toute relation binaire plusieurs à plusieurs est traduite en une table relationnelle dont les caractéristiques sont les suivantes :

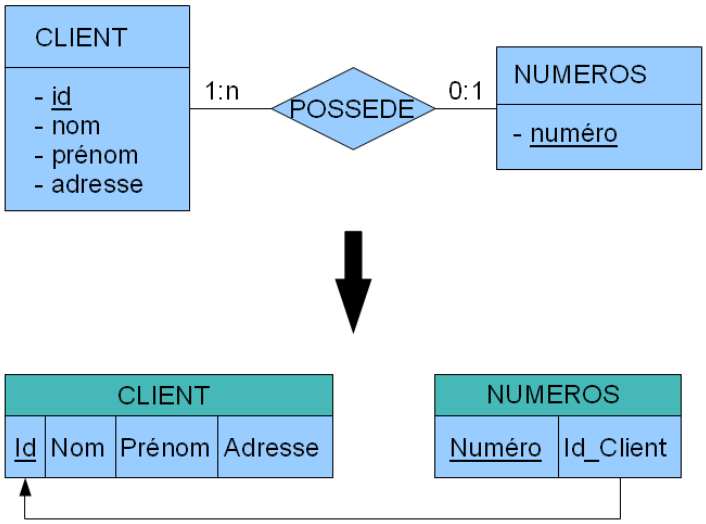
- le nom de la table est le nom de la relation ;
- la clé de la table est formée par « l'addition » des identifiants des entités participant à la relation ;
- les attributs spécifiques de la relation forment les autres colonnes de la table.



Règle III

Toute relation binaire un à plusieurs est traduite :

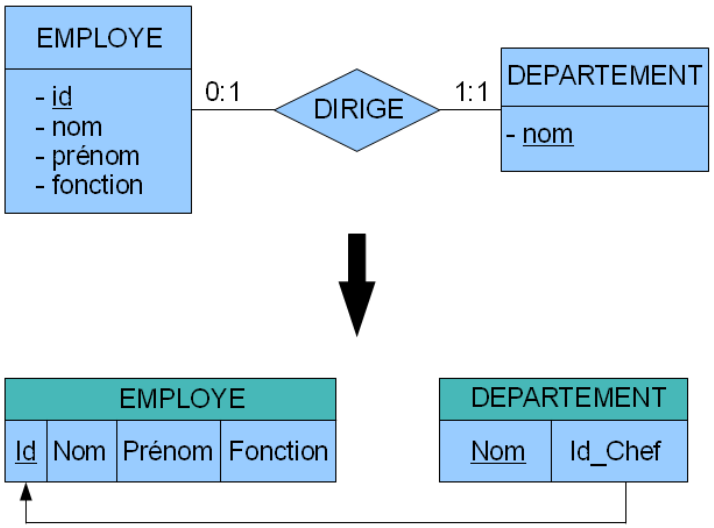
1. soit par un report de clé (voir schéma ci-dessous) : l'identifiant de l'entité participant à la relation côté *N* est ajoutée comme colonne supplémentaire à la table représentant l'autre entité. Cette colonne est parfois appelée clé étrangère. Le cas échéant, les attributs spécifiques à la relation sont eux aussi ajoutés à la même table ;



2. soit par une table spécifique dont les caractéristiques sont les suivantes :
   - le nom de la table est le nom de la relation ;
   - la clé de la table est l'identifiant de l'entité participant à la relation côté 1 ;
   - les attributs spécifiques de la relation forment les autres colonnes de la table.

Règle IV

Toute relation binaire un à un est traduite, au choix, par l'une des trois solutions suivantes :
- choix 1 : fusion des tables des entités qu'elle relie ;
- choix 2 : report de clé d'une table dans l'autre (voir ci-dessous) ;
- choix 3 : création d'une table spécifique reliant les clés des deux entités.



Exercice 15.4

Transformez le schéma entité-association de l'exercice 15.1 en un modèle relationnel.

Exercice 15.5

Transformez le schéma entité-association de l'exercice 15.2 en un modèle relationnel.

Exercice 15.6

Transformez le schéma entité-association de l'exercice 15.3 en un modèle relationnel.

15.5. Le langage SQL

Le modèle relationnel de **Codd** a été rapidement admis comme modèle définitif pour les bases de données. Un langage, *Structured English Query Language* (langage d'interrogation structuré en anglais) a été développé par IBM pour le mettre en œuvre. La première version de SQL a été développée chez IBM en 1970 par Donald **Chamberlain** et Raymond **Boyce**.

En 1979, Relational Software, Inc. (actuellement Oracle Corporation) présenta la première version commercialement disponible de SQL (*Structured Query Language*), rapidement imité par d'autres fournisseurs.

SQL se décompose en 5 parties, à savoir :

- Ordres LDD (langage de définition des données, ou DDL, *Data Definition Language*) : permet de modifier la structure de la base de données.
- Ordres LMD (langage de manipulation des données, ou DML, *Data Manipulation Language*) : permet de consulter / modifier le contenu de la base de données.
- Ordres LCD (langage de contrôle des données, ou DCL, *Data Control Language*) : permet de gérer les privilèges, c'est-à-dire les utilisateurs et les actions qu'ils peuvent entreprendre.
- Ordres LCT (langage de contrôle des transactions, ou TCL, *Transaction Control Language*) : permet de gérer les transactions, c'est-à-dire rendre atomique divers ordres enchaînés en séquence.
- *SQL procedural* : PSM (Persistent Stored Module), CLI (Call Level Interface), Embedded SQL, … qui est un ensemble d'outils pour que SQL s'interface avec des langages hôtes.

Nous ne verrons ici que des exemples simples pour les deux premières parties.

15.5.1. Exemples d'ordres LDD

Création d'une table : `CREATE TABLE`

```
CREATE TABLE table1 (colonne1 INTEGER,
                     colonne2 INTEGER,
                     colonne3 DATE,
                     colonne4 DATE);
```

Modification d'une table : `ALTER TABLE`

Ajouter une colonne

```
ALTER TABLE table1 ADD COLUMN colonne5 INTEGER NULL;
```

Supprimer une colonne

```
ALTER TABLE table1 DROP COLUMN colonne5;
```

Suppression d'une table : `DROP TABLE`

```
DROP TABLE table1;
```

15.5.2. Exemples d'ordres LMD

Recherche simple de lignes dans une table

Il existe d'autres commandes comme « GROUP BY » ou « HAVING ».

```
SELECT {liste_colonnes}
FROM {TABLES}
WHERE {conditions}
ORDER BY {expressions}
```

Comme souvent en informatique, il mieux vaut éviter les caractères accentués...

Afin de rendre les choses plus concrètes, nous utiliserons les deux tables « employes » et « departements » ci-dessous :

employes						
id	nom	prenom	telephone	id_departement	statut	salaire

departements		
id	nom	id_chef

Trouver le prénom et le numéro de téléphone de tous les « Dupont » dans la table « employes » :

Pour supprimer les doublons dans les résultats : SELECT DISTINCT attributs FROM ...

```
SELECT prenom, telephone
FROM   employes
WHERE  nom = "Dupont";
```

Trouver le prénom et le téléphone des « Dupont » ou « Dupond » dans la table « employes » :

```
SELECT prenom, telephone                SELECT prenom, telephone
FROM   employes                    ou   FROM   employes
WHERE  nom in ("Dupont","Dupond");      WHERE  nom = "Dupont"
                                                OR nom = "Dupond";
```

Compter le nombre de « Dupont » dans la table « employes » :

Il existe cinq fonctions utilisables avec SELECT : COUNT, SUM, AVG, MIN et MAX

```
SELECT COUNT(*)
FROM   employes
WHERE  nom = "Dupont";
```

Compter le nombre d'employés appartenant au service « comptabilité » :

```
SELECT COUNT(*)
FROM   employes, departements
WHERE  departements.nom = "comptabilité"
         AND departements.id = employes.id_departement;
```

Trouver le nom de tous les stagiaires dans la table « employes » et les trier par nom.

Ordre décroissant : ORDER BY nom DESC ;

```
SELECT   nom
FROM     employes
WHERE    statut="stagiaire"
ORDER BY nom;
```

Trouver tous les employés touchant plus de 1500 CHF par mois classés par salaire et afficher tous les champs (*) :

```
SELECT   *
FROM     employes
WHERE    salaire > 1500
ORDER BY salaire;
```

Afficher les statuts et le salaire moyen des employés par statut, si le salaire moyen dépasse 5000 CHF (en omettant le dernière ligne du code ci-dessous, on affichera tous les statuts).

HAVING a le même rôle que WHERE, mais est toujours couplé avec GROUP BY.

```
SELECT   statut, AVG(salaire)
FROM     employes
GROUP BY statut
HAVING   AVG(salaire) > 5000;
```

Ajout d'une ligne

Ajouter un dénommé « Martin » comme chef de la comptabilité.

```
INSERT INTO departements (id, nom, id_chef)
VALUES ("compta", "comptabilité", "Martin");
```

Ajout de données à partir des lignes d'une autre table :

```
INSERT INTO TABLE1 (colonne1, colonne2, colonne3)
       SELECT colonne10, colonne20, (colonne30 + colonne40) / 2
       FROM   TABLE2;
```

Modification de lignes

Changer le statut de l'employé « `Durand` » :

```
UPDATE employes
SET    statut="collaborateur"
WHERE  nom = "Durand";
```

Suppression d'une ligne

Supprimer tous les employés nommés « Alpha » :

```
DELETE FROM employes
WHERE nom = "Alpha";
```

Exercice 15.7

Avec les tables « employes » et « departements » ci-dessus, faites les opérations suivantes :

1. supprimer le département « culture » ;
2. trouver le statut de « Jean Némare » ;
3. modifier le numéro de téléphone de « Jean Némare » : le nouveau numéro est 0325641353 ;
4. ajouter un département « réclamations », avec pour chef de département « Jean Némare » ;
5. supprimer le département dont le chef s'appelle « Haenni » ;
6. trouver la somme (SUM) des salaires des employés pour chaque département ;
7. lister les noms de tous les chefs de départements, triés par nom de famille.

Exercice 15.8 (en ligne)

Vous trouverez sur le site web compagnon un exercice en ligne (avec un corrigé) qui vous permettra de manipuler une base de données réaliste avec le langage SQL.

http://www.apprendre-en-ligne.net/info/database

15.6. Outils de gestion pour MySQL

MySQL est un système de gestion de base de données relationnelle (SGBDR). Il fait partie des logiciels de gestion de base de données les plus utilisés au monde. Il supporte le langage de requête SQL.

On trouve plusieurs outils permettant de gérer des bases de données :

- phpMyAdmin, outil de gestion de bases de données MySQL écrit en PHP
- Adminer, un autre outil de gestion lui aussi écrit en PHP
- MySQL Workbench
- …

Exercice 15.9 : Gestion d'une bibliothèque publique

Le cahier des charges

Une école veut développer une base de données pour sa bibliothèque. Voici les détails à prendre en compte pour la création de la base de données qui permettra la gestion de la bibliothèque :

Faites attention à la différence entre un *livre* et un *exemplaire* d'un livre...

- Grâce au système informatique, un **abonné** devra pouvoir retrouver un **exemplaire** d'un **livre** dans les rayons en connaissant son titre. L'abonné devra aussi pouvoir connaître la liste des livres d'un **auteur** ou la liste par **éditeur**.
- Un abonné pourra réserver un livre emprunté.
- Les livres sont identifiés par un code catalogue affecté à l'achat et par un code rayon qui permet de les situer dans la bibliothèque. Chaque livre est acheté en un ou plusieurs exemplaires (on stocke la date d'acquisition). Tous les **exemplaires** d'un même livre ont un code rayon différent mais le même code catalogue. Les différents exemplaires d'un même livre peuvent éventuellement provenir de différents éditeurs.
- La bibliothèque gère un fichier des abonnés organisé par numéro de matricule qui contient les coordonnées (nom, adresse et téléphone) de l'abonné.
- La gestion des prêts implique la possibilité de connaître à tout moment la liste des livres détenus par un abonné, et inversement, qu'on puisse retrouver le nom des abonnés détenant un livre non présent dans les rayons.
- Les prêts sont accordés pour une durée de quinze jours, éventuellement renouvelable, si aucune demande de ce livre n'a eu lieu entre-temps. Il faudrait donc connaître pour chaque livre emprunté, la date du prêt et la date de retour.

Les fonctions que le système devra effectuer sont :

- la gestion des prêts (prêts, réservations et retours) ;
- la mise à jour du fichier des livres (mises au rebut ou achats de livres) ;
- la mise à jour du fichier des abonnés (retraits et nouvelles inscriptions) ;
- la possibilité d'aide au choix d'un ouvrage grâce à une procédure de recherche par nom d'auteur ou par éditeur.

À faire

a) Établir un modèle entités-associations.
b) Transformer le modèle entités-associations en un modèle relationnel.
c) Écrire en SQL les requêtes 1 à 6 ci-dessous.

La base de données doit notamment permettre de répondre aux requêtes suivantes :

1. Qui a emprunté « Madame Bovary » ?
2. Quels sont les livres de Gustave Flaubert que la bibliothèque possède ?
3. Quand « Les 9 couronnes » a-t-il été emprunté (il n'y a qu'un exemplaire) ?
4. Quels sont tous les livres de l'éditeur « Gallimard » ?
5. Quel est le courriel de la personne qui a réservé « Mangeclous » (il n'y a qu'un exemplaire) ?
6. Quels sont les livres qui devront être restitués demain ?

Exercice 15.10 : Gestion d'une ludothèque privée

Le cahier des charges

Passionné de jeux de société, *Maximilien* souhaite créer une base de données de ses jeux, et garder une trace des parties qu'il a jouées.

Un **jeu** est défini par son nom. Il faudra que s'affichent une image de la boîte, le créateur, l'éditeur, l'année de sortie, le nombre de joueurs minimum et maximum, l'âge minimal conseillé et la

durée théorique d'une partie. À cela s'ajouteront encore le thème du jeu (animaux, vampires, etc.), les mécanismes (pari, course, affrontement, etc.), et si elle existe, une vidéo disponible sur le web qui explique les règles. Enfin, *Maximilien* aimerait pouvoir rédiger un avis et donner une note.

Certains jeux ont une ou plusieurs **extensions**, identifiées par leur nom. Celles-ci ne peuvent pas se jouer sans le jeu de base. Quand la page du jeu s'affiche, il faudra que l'on puisse voir les extensions que *Maximilien* possède. Comme pour les jeux, on veut pouvoir afficher une image de la boîte, l'année de sortie, le nombre de joueurs minimum et maximum, l'âge minimal conseillé et la durée théorique d'une partie

Une **partie**, identifiée par un numéro, a lieu à une date donnée entre certains **joueurs**, définis par leur pseudo. Pour chaque partie, on mémorisera le vainqueur. *Maximilien* veut aussi pouvoir commenter la partie jouée.

À faire

a) Établir un modèle entités-associations.
b) Transformer le modèle entités-associations en un modèle relationnel.
c) Écrire en SQL les requêtes 1 à 5 ci-dessous.

La base de données doit notamment permettre de répondre aux requêtes suivantes :

1. « Quels sont les jeux de *Maximilien* créés par *Reiner Knizia* ? »
2. « Combien de parties a jouées *Greg* en 2014 ? »
3. « Quelles parties a gagnées *Greg* en 2014 ? »
4. « Combien de fois *Maximilien* a-t-il joué à *Suburbia* » ?
5. « Qui a déjà joué à *Suburbia* ? »

Exercice 15.11 : Cycle de colloques

Le cahier des charges

On désire informatiser l'organisation d'un cycle de colloques universitaires.

- Les **colloques** se déroulent dans des universités différentes à des dates différentes et sont organisées par des personnes différentes. Chaque colloque a un nom spécifique et est constitué d'un ensemble d'exposés. L'université dans laquelle il a lieu et la date sont aussi fixées.
- Chaque **exposé** est identifié par un titre. Il est accompagné d'un résumé. Le même exposé peut être présenté dans plusieurs colloques.
- Un exposé est présenté par un seul **conférencier** dans un colloque. Par contre, un conférencier peut faire plusieurs exposés. Un conférencier peut aussi être un **organisateur** de colloque.
- On souhaite garder la trace des **participants** à ces colloques (qui peuvent aussi être conférencier et/ou organisateur). Chaque participant est identifié par un numéro et décrit par son nom, son prénom et son email.

À faire

a) Établir un modèle entités-associations.
b) Transformer le modèle entités-associations en un modèle relationnel.
c) Écrire en SQL les requêtes 1 à 4 ci-dessous.

La base de données doit notamment permettre de répondre aux requêtes suivantes :

1. « Qui (nom et prénom) organise le colloque *Mathématiques et Société* » ?
2. « Quels sont les titres des exposés du colloque *Mathématiques et Société* » ?
3. « Combien d'exposés sont présentés par *John Doe* » ?
4. « Combien de personnes sont inscrites au(x) colloque(s) de l'université de Neuchâtel » ?

Références

[1] « Conception des bases de données relationnelles », <http://tecfa.unige.ch/staf/staf-h/tassini/staf2x/Heidi/last_bd.htm>

[2] H. F. Korth, A. Silberschatz, *Systèmes de gestion des bases de données*, McGraw-Hill, 1988

[3] M. Takahashi, S. Azuma, Tred-Pro Co., *The manga guide to databases*, Ohmsha, 2009