Licence 3 - 2016–2017
Projet technologique
Université Bordeaux

Notes de cours
Bases de données
Lionel Clément

Plan

Table des matières

A quoi servent les bases de données ?

Plusieurs motivations historiques

— Stocker de l'information en grosse quantité
— Que cette information soit accessible facilement et rapidement
— Que cette information ne soit pas inutilement redondante
— Que cette information soit cohérente et non contradictoire
— Limiter le nombre d'outils permettant d'avoir accès à cette informations
— Limiter la dépendance entre l'information stockée et les logiciels pour y avoir accès

Les défis actuels

— Multiplier les types de données : Vidéos, textes et hypertextes, fichiers audio, etc.
— Gérer de gros volumes.
— Accès simple pour les utilisateurs finaux
Aucune formation préalable ne devrait être supposée pour qu'un utilisateur final puisse accéder aux données enregistrées.
— Maintenance, évolution aisée
Les bases de données évoluent rapidement dans le temps ; des données sont ajoutées, sont supprimées ou modifiées. Il faut que le système puisse faire ces transformations sans produire de panne ni d'incohérence. Pensez à un système de réservation de train : dans toutes les gares ou depuis chez eux, des milliers d'utilisateurs peuvent modifier la même base de réservation de billets sans incohérence.
— Accès rapide à l'information
Quelle que soit la complexité du système d'information, la dispersion des données et leur volume, il faut pouvoir accéder en une fraction de seconde à son compte en banque, à la réservation de billetterie, à une référence dans une bibliothèque, etc.
— Accès sécurisé aux informations sensibles
Certaines données sont sensibles (comptes en banque, vie privée, données sensibles de la police ou de l'armée) ; d'autres sont précieuses (données judiciaires, données bancaires, archives). Il faut pouvoir s'assurer qu'elles ne seront accessibles ou modifiables qu'avec une authentification.
— Accès concurrent
La même base de donnée peut être utilisée par plusieurs systèmes ou par plusieurs personnes en même temps. Il ne faut pas que des requêtes contradictoires puissent corrompre les données. Par exemple, un système où l'on pourrait acheter un billet d'avion alors qu'il est réservé par une autre personne serait inadapté.

Exemples de bases de données :

— Comptes bancaires.
— Bibliothèque.

— Ventes, achats, dépôts, livraisons d'un groupe de distribution.
— Réservations de billetterie de spectacles, de voyages, d'hôtels, de restaurants, etc.
— Jeux en ligne massivement multi-joueur (MMO ou MMOG, *Massively Multiplayer Online Game*)

Les Système de Gestion de Bases de Données (SGBD) [1]

Il s'agit d'un programme destiné à stocker et à rendre disponible des informations en respectant un cahier des charges rigoureux sur la sécurité, la pérennité, la cohérence et la disponibilité des données.

En voici une liste parmi les plus courants avec la licence d'utilisation associée.

— Mysql (GPL)
— Postgres (BSD)
— Oracle (Propriétaire)
— Microsoft Access (Propriétaire)
— SQLite (Domaine public)
— OpenOffice Base (LGPL)
— ...

Nous utiliserons `MySQL` en raison de sa licence non propriétaire et de l'usage important qui est fait de se programme dans le monde.

Cahier des charges [2] pour un SGBD

1. Organisation des données

   Le système doit permettre d'indiquer la façon dont les données sont organisées. On doit par exemple pouvoir choisir comment identifier un individu parmi d'autres : inscrit-on son prénom, son nom, un numéro d'identifiant ?

   Des **contraintes d'intégrité** peuvent être définies. Elles permettront de préciser les conditions pour qu'une valeur soit correcte, si deux éléments sont incompatibles entre eux, etc.

2. Gestion des données

   Il doit être possible d'ajouter, de supprimer et de modifier des éléments de la base de donnée sans remettre en cause l'intégrité de la base.

3. Accès aux données

   Il est possible de retrouver une information rapidement dans un grand ensemble de données.

4. Personnalisation des données

   Une partie seulement de l'ensemble des informations qui sont contenues dans la base de données sont accessibles par chaque individu.

5. Contrôles des accès

1. *DataBase Management System*
2. *Scope statement*

Certaines données sont sensibles ou confidentielles. Chaque utilisateur a des droits spécifiques qui limitent l'accès à une partie seulement de la base de données. Cela est vrai indépendamment pour la consultation et pour la modification de la base. On peut par exemple vouloir laisser l'accès à la lecture des données mais pas à leur modification.

6. Protection contre les incidents.

   Un disque dur qui tombe en panne, un ordinateur qui s'interrompt pendant la modification d'une donnée ne doit pas mettre en péril l'ensemble de la base de données.

7. Accès concurrents

   Une même base de données peut être utilisée en même temps par plusieurs personnes à la fois. L'intégrité de la base de données et la validité des résultats ne doit pas être remise en cause.

1 Le modèle relationnel

La relation est le seul concept qui sera utilisé ici. Il s'agit de la relation au sens mathématique du terme, c'est-à-dire un ensemble de *tuples*.

1.1 Relation

Exemples de relations :

- Une fonction $f : E \to F$ est une relation sur deux ensembles.
  Par exemple $\{(0,1),(1,2),(2,3),\dots,\} \subset \mathbb{N} \times \mathbb{N}$, la fonction successeur est une relation.
- Un opérateur est une fonction sur trois ensembles. Par exemple l'addition sur les entiers peut se représenter par un ensemble de triplets (x, y, z) où z est la somme de x et de y. $\{(0,0,0),(0,1,1),(1,0,1),(2,0,2),(1,1,2),(0,2,2),\dots,\} \subset \mathbb{N}^3$
- Un prédicat est une relation. Par exemple l'ensemble $\{(1),(3),(5),(7),(11),\dots\}$ représente la relation "est premier", à ne pas confondre avec l'ensemble des nombres premiers.

La relation servira à représenter à la fois les entités et les associations.

Par exemple, l'entité "individu" qui permet de distinguer un individu d'un autre sera représenté par une relation sur l'ensemble des noms, des prénoms, des adresses, des numéros de téléphones, etc.

Nous aurons quelque chose comme = {("Durant", "Christine", "3, rue du Loup, Bordeaux", "05 54 56 87 56"), ("Dupont", "Jean", "6, cours Pasteur, Bordeaux", "05 47 58 25 14"), ("Marceau", "Marie", "7, cours de la Libération, Talence", "05 78 65 12 32"), ("Jules", "Jean", "8, cours Victor Hugo, Bordeaux", "05 84 59 62 48")}.

Une façon plus lisible et pratique d'écrire cet ensemble de tuples consiste à faire un tableau où chaque colonne correspond à un argument donné et chaque ligne correspond à un tuple.

Voici la même donnée écrite sous la forme d'un tableau :

Nom	Prénom	Adresse	Téléphone
"Durant"	"Christine"	"3, rue du Loup, Bordeaux"	"05 54 56 87 56"
"Dupont"	"Jean"	"6, cours Pasteur, Bordeaux"	"05 47 58 25 14"
"Marceau"	"Marie"	"7, cours de la Libération, Talence"	"05 78 65 12 32"
"Jules"	"Jean"	"8, cours Victor Hugo, Bordeaux"	"05 84 59 62 48"

1.2 Quelques définitions qui nous seront utiles

Un **attribut** pour une relation donnée, représente la valeur d'un de ses arguments. Nous avons matérialisé l'attribut par une colonne du tableau.

Un **domaine** est un ensemble de valeurs possibles pour une colonne. Par exemple des nombres, des chaînes de caractères, des couleurs, etc. Le type d'un attribut est l'implémentation de son domaine. Par exemple, les nombres entiers seront représentés par un type `integer` correspondant à un ensemble fini de nombres entiers (de 0 à $2^{16} - 1$ par exemple).

Chaque attribut A possède un domaine noté $Dom(A)$

Une **relation** $R(A_1, A2, \ldots, A_k)$ sur les attributs $A_1, A_2, \ldots A_k$ est une partie du produit cartésien des domaines de $A_1, A_2, \ldots, A_k$. On écrit $R \subseteq \prod_{i \in [1,k]} Dom(A_i)$

1.3 Exemple de relation

Une liste de produits peut être proposée sous la forme d'une relation `Produit` = R(Libellé, Prix, Couleur, Poids). Cette relation possède 4 attributs, c'est donc un ensemble de quadruplets (4-uplets) dont les domaines sont :

- Dom(Libellé) = chaine de caractère (*string*)
- Dom(Prix) = $\mathbb{D}$ (nombre décimal)
- Dom(Poids) = $\mathbb{N}$
- Dom(Couleur) = {rouge, jaune, bleu, vert, orange, violet}

Nous allons représenter cette relation sous la forme d'un tableau :

1. Les colonnes représentent les attributs
2. Les lignes représentent les tuples

Libellé	**Prix**	**Poids**	**Couleur**
"Chaise"	45	4	blanc
"Table"	98,5	12	blanc
"Bureau"	150	25	gris
"Armoire"	98,5	24	bleu

1.4 Clef (ou identifiant) d'une relation

Une *clef*[3] est un ensemble d'attributs suffisant pour identifier un tuple dans l'ensemble de la relation. Plus formellement, si deux tuples sont différents, alors les valeurs sur la clef seront différentes.

Soit $R(A_1, \ldots, A_k)$ une relation, soit $C = \{C_1, \ldots, C_l\} \subseteq \{A_1, \ldots, A_k\}$. C est une clef si et seulement si $\forall v = (v_1, \ldots, v_k), w = (w_1, \ldots, w_k) \in R$, $(v_i = v_j) \Rightarrow (w_i = w_j)$

Nous ajouterons que la clef doit être minimale, c'est-à-dire qu'elle ne contient elle-même pas d'autres clefs. Autrement dit, une clef est le plus petit sous-ensemble d'attributs qui suffit à distinguer tout tuple des autres.

Remarques :

1. Une relation peut avoir plusieurs clefs.
2. Toute relation possède au moins une clef. En effet, l'ensemble des attributs suffit à distinguer un tuple des autres.

1.5 Clef externe

Une clef externe[4] est un ensemble d'attributs d'une relation qui fait référence à une clef d'une autre relation.

3. Key
4. Foreign key

Soit deux relations $R(A_1, \ldots A_k)$ et $S(B_1, \ldots B_l)$. $C \in \{A_1, \ldots A_k\}$, est une clef externe de la relation R vers la relation S si et seulement si.

1. C est une clef dans S
2. Pour tout tuple dans R, il existe un tuple dans S qui a les mêmes valeurs sur les attributs C.

Prenons par exemple deux relations, l'une concernant des clients, l'autre concernant les commandes que passent ces clients. Nous allons utiliser une clef de la relation client pour l'utiliser dans la relation commande.

Client

IDClient	Nom	Prénom	Adresse	Téléphone
1	"Durant"	"Christine"	"3, rue du Loup, Bordeaux"	"05 54 56 87 56"
2	"Dupont"	"Jean"	"6, cours Pasteur, Bordeaux"	"05 47 58 25 14"
3	"Marceau"	"Marie"	"7, cours de la Libération, Talence"	"05 78 65 12 32"
4	"Jules"	"Jean"	"8, cours Victor Hugo, Bordeaux"	"05 84 59 62 48"

Commande

IDCommande	**IDClient**	Date
1000	**1**	2010-04-25
1001	**1**	2010-04-26
1002	**3**	2010-05-04

Dans cet exemple, les valeurs **1** et **3** marquée en **gras** de la relation *Commande* correspondent à des valeurs de l'attribut IDClient de la relation *Client*. Par ailleurs, l'attribut IDClient [5] constitue une **clef** de la relation *Client*. L'attribut IDClient est donc une **clef externe** de la relation *Commande* vers la relation *Client*.

1.6 Schéma d'une relation

Nous verrons plus tard qu'il est possible d'intégrer un ensemble de contraintes permettant de valider les relations. Ces contraintes servent à assurer la correction des données. Il est par exemple possible de limiter les valeurs possibles d'un attribut aux seules valeurs utiles. Un prix, un poids ou un volume doivent être positifs, les dates doivent correspondre à des dates valides (par exemple pas un 30 février ou un 31 septembre).

La définition des domaines des attributs, des clefs, et des clefs externes constitue déjà un ensemble de contraintes auxquelles vont s'ajouter des contraintes explicites.

Nous appelons *schéma* d'une relation, une définition de cette relation comprenant
- Son nom
- Ses Attributs
- La définition des domaines de chaque attribut
- Les clefs et les clefs externes
- Les contraintes supplémentaires

5. L'attribut *IDClient* seul constitue une clef. Dans ce cas, nous l'identifierons par abus d'écriture à l'ensemble {*IDClient*}.

1.6.1 Exemple

Le schéma de la relation *Commande* sera le suivant :

$Commande(IDCommande, IDClient, Date)$

$Dom(IDCommande) = \mathbb{N}$

$Dom(IDClient) = \mathbb{N}$

$Dom(Date) = ensembledesdates$

Avec :

IDCommande, Clef de la relation

IDClient, clef externe vers la relation Client

Date, ensemble des dates valides

2 Algèbre relationnel

Nous rappelons que la seule notion dont nous aurons besoin est celle de *relation*, que nous avons défini par un ensemble de tuples représentant les entités et parfois les associations de nos bases de données. Nous représenterons les relations sous la forme de tables où les colonnes représentent les attributs et les lignes les tuples.

Les relations sont des objets mathématiques, et nous pouvons appliquer des opérations sur ces objets comme nous le faisons avec les nombres. Les opérations sont, soit unaires (elles ne prennent qu'un seul argument), soit binaires (elles prennent deux arguments).

Ces opérations visent à obtenir un résultat à partir d'une ou plusieurs relations. Par exemple, on utilise une opération pour extraire les tuples d'une relation ou encore pour obtenir les valeurs d'une relation sur seulement quelques attributs. Encore un exemple : on peut utiliser une opération entre deux relations R_1 et R_2 pour extraire les tuples de R_1 qui ne seraient pas dans R_2.

Les opérateurs classiques des ensembles sont utilisés (l'union, l'intersection, la différence) [6].

2.1 Sélection σ

Soit une relation $R(A_1, A_2, \ldots, A_k)$

Soit P un prédicat sur les attributs $A_1, A_2, \ldots, A_k$.

La sélection notée $\sigma[P]$, est un opérateur sur une relation qui renvoie les tuples tels que le prédicat donné en paramètre est vérifié.

Exemple : Soit la relation *Livre* suivante :

Titre	Auteur	Année
"Madame Bovary"	Gustave Flaubert	1856
"Le père Goriot"	Honoré de Balzac	1835
"Notre-Dame de Paris"	Victor Hugo	1831
"Les misérables"	Victor Hugo	1862

La sélection $\sigma[Date \leq 1835](Livre)$ donne :

Titre	Auteur	Année
"Le père Goriot"	Honoré de Balzac	1835
"Notre-Dame de Paris"	Victor Hugo	1831

La sélection permet de filtrer une liste de lignes d'une table.

2.2 Projection π

Soit une relation $R(A_1, A_2, \ldots, A_k)$

Soit $B_1, B_2, \ldots, B_l$ un sous ensemble des attributs de R.

6. Nous n'utilisons pas le complémentaire qui implique un contexte que nous n'avons pas ici ; car le complémentaire d'un ensemble est l'ensemble des éléments qui ne sont pas dans cet ensemble.

La projection notée π, est un opérateur sur une relation qui renvoie la même relation en ne conservant qu'une partie des attributs.

La sélection $\pi[\{Auteur\}](Livre)$ donne :

Auteur
Gustave Flaubert
Honoré de Balzac
Victor Hugo

La projection permet de filtrer les valeurs d'une partie des attributs d'une relation.

Remarque :

La ligne *Victor Hugo* n'est présente qu'une et une seule fois. En effet, n'oublions pas que nous manipulons des ensembles et non des listes de tuples.

2.3 Renommage α

Le renommage est une opération qui permet de donner un autre nom à un attribut sans rien changer d'autre. Cette opération est utile pour éviter des confusions de nom ou au contraire pour donner le même nom à deux attributs différents.

On note $\alpha[A : B]R$ la relation R où l'attribut A est renommé B.

2.4 Union $\cup$

Soit $R(A_1, A_2, \ldots, A_K)$ et $S(A_1, A_2, \ldots, A_k)$ deux relations ayant les mêmes attributs.

L'union $R \cup S$ est l'ensemble des tuples qui sont dans R **ou** qui sont dans S (ou qui sont dans les deux).

L'union sert à additionner deux ensemble de tuples.

2.5 Intersection $\cap$

Soit $R(A_1, A_2, \ldots, A_K)$ et $S(A_1, A_2, \ldots, A_k)$ deux relations ayant les mêmes attributs.

L'intersection $R \cap S$ est l'ensemble des tuples qui sont dans R **et** qui sont dans S.

L'intersection permet de filtrer un ensemble de tuples à partir d'un autre par exemple.

2.6 Produit cartésien

Soit $R(A_1, A_2, \ldots, A_K)$ et $S(B_1, B_2, \ldots, B_l)$ deux relations *dont les attributs ne sont pas les mêmes*.

Le produit $R \times S$ est l'ensemble des tuples qui combinent les tuples de R à tous ceux de S.

Exemple :

Soit *Oeuvre*, la relation suivante :

Titre	Auteur	Année
"Madame Bovary"	Gustave Flaubert	1856
"Le père Goriot"	Honoré de Balzac	1835
"Notre-Dame de Paris"	Victor Hugo	1831
"Les misérables"	Victor Hugo	1862

Soit *Livre*, la relation suivante :

Etat	Prix
"Neuf"	15
"Bon état"	7
"Abimé"	3

$Oeuvre \times Livre =$

Titre	Auteur	Année	Etat	Prix
"Madame Bovary"	Gustave Flaubert	1856	" Neuf"	15
"Le père Goriot"	Honoré de Balzac	1835	"Neuf"	15
"Notre-Dame de Paris"	Victor Hugo	1831	"Neuf"	15
"Les misérables"	Victor Hugo	1862	"Neuf"	15
"Madame Bovary"	Gustave Flaubert	1856	"Bon état"	7
"Le père Goriot"	Honoré de Balzac	1835	"Bon état"	7
"Notre-Dame de Paris"	Victor Hugo	1831	"Bon état"	7
"Les misérables"	Victor Hugo	1862	"Bon état"	7
"Madame Bovary"	Gustave Flaubert	1856	"Abimé"	3
"Le père Goriot"	Honoré de Balzac	1835	"Abimé"	3
"Notre-Dame de Paris"	Victor Hugo	1831	"Abimé"	3
"Les misérables"	Victor Hugo	1862	"Abimé"	3

2.7 Jointure $\bowtie$

Nous voulons maintenant avoir une opération entre deux relations, non pas pour lister tous les cas possibles, mais pour mettre en correspondance certains tuples d'une relation avec certains tuples d'une autre relation. Par exemple, les *Livres* qui correspondent à certaines *Oeuvres* et non à toutes.

Pour faire cette correspondance, il faut que les deux relations aient des attributs communs.

La jointure entre R et S notée $R \bowtie S$ construit une relation constituée des tuples de R et des tuples de S où les valeurs sur les attributs en commun sont les mêmes.

Soit $R(A_1, \ldots, A_k, C_1, \ldots C_n)$ et $S(B_1, \ldots, B_m, C_1, \ldots C_n)$,

$R \bowtie S = \pi[\{A_i, B_j, C_k\}]\sigma[A_i = A'_i](\alpha[A_i : A'_i]R \times S)$

Exemple :

Soit *Oeuvre*, la relation suivante :

ID	Titre	Auteur	Année
1	"Madame Bovary"	Gustave Flaubert	1856
2	"Le père Goriot"	Honoré de Balzac	1835
3	"Notre-Dame de Paris"	Victor Hugo	1831
4	"Les misérables"	Victor Hugo	1862

Soit *Exemplaire*, la relation suivante :

ID	Etat	Prix
3	"Neuf"	15
3	"Bon état"	7
4	"Abimé"	3
1	"Neuf"	15

$Oeuvre \bowtie Livre =$

ID	Etat	Prix	Titre	Auteur	Année
1	"Neuf"	15	"Madame Bovary"	Gustave Flaubert	1856
3	"Neuf"	15	"Notre-Dame de Paris"	Victor Hugo	1831
3	"Bon état"	7	"Notre-Dame de Paris"	Victor Hugo	1831
4	"Abimée	3	"Les misérables"	Victor Hugo	1862

2.8 Propriétés algébriques

L'union, l'intersection, et le produit sont associatifs et commutatifs.

R op $S = S$ op R
R op $(S$ op $T) = (R$ op $S)$ op T

La jointure n'est pas associative.

$\pi[A](\pi[B]R) = \pi[A \cap B]R$
$\sigma[p](\sigma[q]R) = \sigma[p$ ET $q]R$

3 SQL (Structured Query Language)

SQL (Structured Query Language) est un langage permettant de décrire les schémas relationnels, et de manipuler les données enregistrées sous forme de relation.

Nous utiliserons le logiciel MySQL pour expérimenter nos travaux. Il est possible de l'installer sous Linux, et je vous invite à le faire sur votre propre ordinateur. MySQL est gratuit et sous licence GPL (les sources du logiciel sont disponibles publiquement et ne peuvent pas être versés dans un logiciel commercial sous une autre licence).

Documentation : *http ://dev.mysql.com/doc/*

Au CREMI, vous pourrez utiliser le serveur MySQL. En revanche, vous ne pourrez pas l'administrer.

Connexion au serveur

```
mysql -h <hote> -u <utilisateur> -p
```

Création d'une base de donnée

Vous pouvez créer votre propre base de données sur un ordinateur que vous administrez vous-même. Cette situation est cependant exceptionnelle dans le monde industriel ; seul l'administrateur système fait ce genre de choses. En administrant sa base de données, il est possible de créer des utilisateurs, de leur donner des droits particuliers, etc.

Au Cremi, vous aurez la possibilité de créer des nouvelles tables, mais pas des utilisateurs ni des bases de données.

```
CREATE DATABASE <nom de la base>;
```

Ensuite, la connexion au serveur se fait en utilisant une base de donnée :

mysql -h <hote> -u <utilisateur> -p <nom de la base> ;

3.1 SQL, langage de description de schémas relationnels *(DDL pour Data Definition Language)*

3.1.1 Création de tables

```
CREATE TABLE <nomDeLaTable> <attributs>;
```

Les attributs sont listés avec leur type et leurs éventuelles contraintes.

```
CREATE TABLE Oeuvre (
        ID INTEGER PRIMARY KEY,
        Titre VARCHAR NOT NULL,
        Auteur VARCHAR NOT NULL,
        Annee YEAR,
        ISBN CHAR(13) UNIQUE
);
```

Dans cette déclaration, nous créons une table *Oeuvres* contenant les attributs *ID, Titre, Auteur, AnnéeEdition, ISBN*.

Nous avons ajouté ces contraintes à la table :

- {ID} est la clef primaire, celle qui sera utilisée pour faire référence à une œuvre en particulier.
- Tous ces attributs doivent avoir des valeurs non nulles, sauf *Annee* et *ISBN*.
- ISBN est une clef (nous dirons une clef *secondaire*)

Voyons maintenant une table qui fait référence à la table Ouvrage

```
CREATE TABLE Exemplaire (
        IDExemplaire INTEGER PRIMARY KEY,
        IDOuvrage INTEGER NOT NULL,
        FOREIGN KEY (IDOuvrage) REFERENCES Oeuvre(ID) );
```

Dans cette dernière déclaration, nous indiquons que IDOuvrage est une clef externe qui fait référence à la clef ID de la table Ouvrage.

3.1.2 Destruction de tables

Voici la commande pour supprimer la table Ouvrages :

```
DROP TABLE Ouvrages;
```

Il se peut cependant que cette commande provoque une erreur de cohérence de la base de données. En effet, si une table *Exemplaire* existe et fait référence à la table *œuvre*, celle-ci ne peut pas être supprimée en premier. Il faut donc veiller à respecter un ordre de suppression : les tables qui contiennent des clefs externes doivent être supprimées après celles qui leur font référence. Ceci se fait automatiquement avec le mot clef *CASCADE*.

3.1.3 Modification de tables

La commande ALTER TABLE permet de modifier une table pour

- Ajouter, supprimer ou modifier des attributs.

  ```
  ALTER TABLE Ouvrages ADD COLUMN Publisher VARCHAR AFTER
     Auteur;
  ```

  Ajoute l'attribut *Edit* dont le type est une chaine de caractères après l'attribut *Auteur*.

  ```
  ALTER TABLE Ouvrages CHANGE COLUMN Publisher Editeur VARCHAR
     ;
  ```

  Modifie le nom de l'attribut *Edit* pour *Editeur*

  ```
  ALTER TABLE Ouvrages DROP COLUMN Editeur;
  ```

  Supprime la colonne *Editeur*

3.2 SQL, Langage de manipulation des données (DML pour Data Manipulation Language)

3.2.1 Insertion de nouvelles lignes dans une table

```
INSERT INTO Ouvrages
(Titre, Auteur, AnneeEdition, AnneeImpression)
VALUES ('Mme␣Bovary', 'Gustave␣Flaubert', 1857, 2005);

INSERT INTO Exemplaires
VALUES (1001, 256);
```

3.2.2 Mise à jour de certains champs d'une ligne dans une table

```
UPDATE Ouvrages
SET Titre = 'Madame␣Bovary'
WHERE IdExemplaire=1001;
```

3.2.3 Suppression des lignes dans une table

```
DELETE FROM Exemplaires
WHERE IDExemplaire=1001;
```

3.3 SQL, Langage d'interrogation des données (DQL pour Data Query Language)

3.3.1 Une seule commande : SELECT

```
SELECT ...
FROM ...
WHERE ...
```

Pour projeter sur tous les attributs

```
SELECT *
FROM ...
WHERE ...
```

Pour obtenir une relation au sens mathématique, c'est-à-dire pour éviter les doublons

```
SELECT DISTINCT ...
FROM ...
WHERE ...
```

Voici la forme générale d'une requête SQL

```
SELECT [ALL]|[DISTINCT] {*|colonne [, ...]}
FROM table [, ...]
[WHERE conditions]
```

GROUP pour regrouper les résultats en fonction de la valeur sur certaines colonnes

```
SELECT {*|colonne [, ...]}
FROM table [, ...]
[WHERE conditions]
GROUP BY {colonne [ASC|DESC] [, ...]}
[HAVING conditions]
```

ORDER pour trier les résultats

```
SELECT {*|colonne [, ...]}
FROM table [, ...]
[WHERE conditions]
ORDER BY {colonne [ASC|DESC] [, ...]}
```

Quelques fonctions utiles pour quantifier les résultats

AVG	Calcule la valeur moyenne d'une colonne
COUNT	Compter le nombre d ?occurrences d'une ligne
MAX	Extrait la valeur maximale d'une colonne
MIN	Extrait la valeur minimale d'une colonne
SUM	Additionne toutes les valeurs d'une colonne