



Chapitre 4

SQL

Data Definition Language
Data Manipulation Language

Plan

- Les langages informatiques
- Le langage SQL

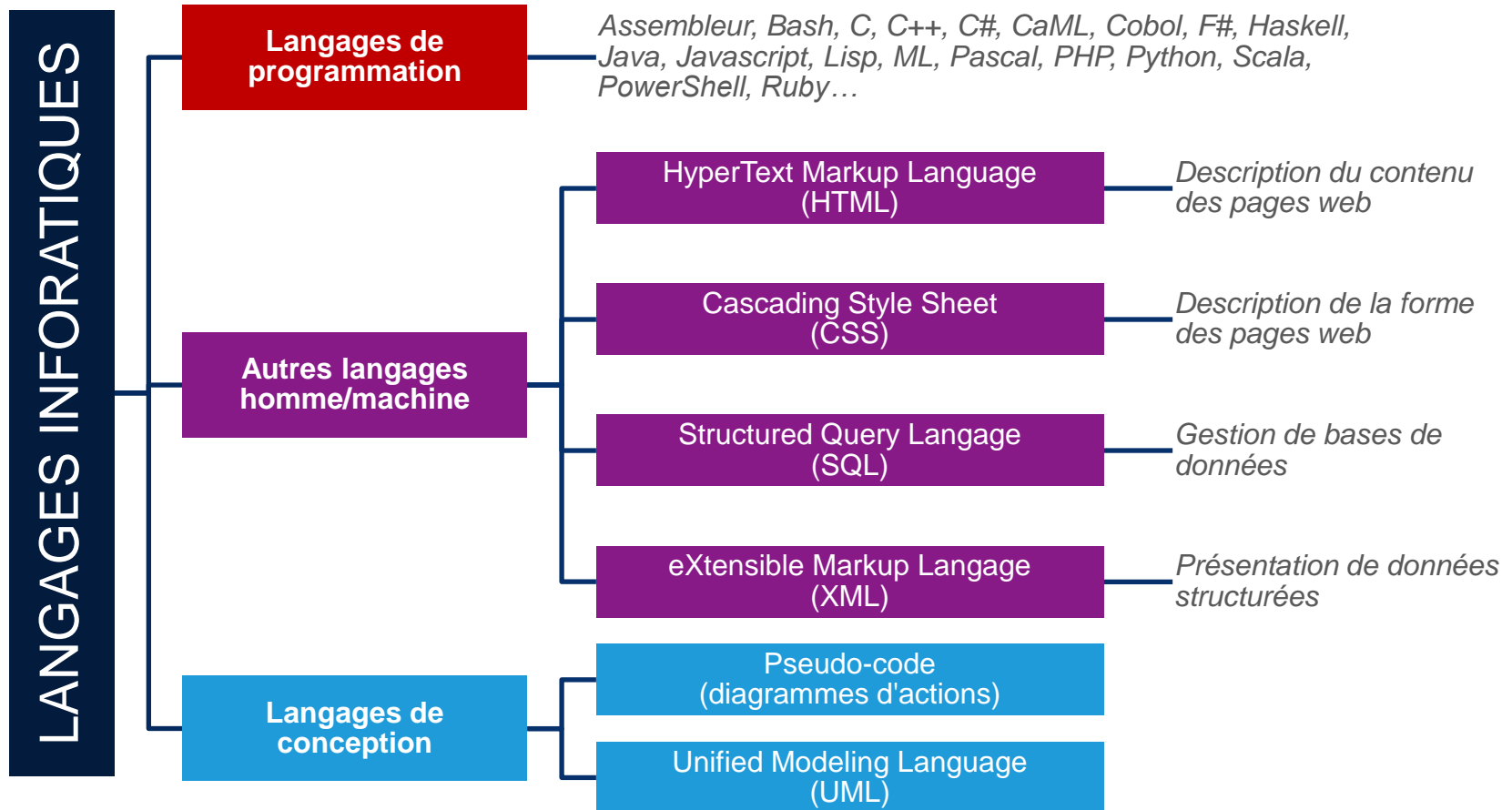
4.1. Data Definition Language

- Contraintes et conventions de nommage
- Créer une base de données
- Créer une table
- Valeur par défaut pour une colonne
- Contrainte de colonne
- Contrainte de table
- Check
- Clé étrangère
- Modifier la structure d'une table
- Supprimer une table

4.2. Data Manipulation Language

- Insérer des données
- Modifier des données
- Supprimer des données

Les langages informatiques



Le langage SQL

Structured Query Language

Langage d'exploitation des bases de données relationnelles

Le langage SQL

Sous-langages SQL

- **DDL** (Data Definition Language)
 - Définition des tables
- **DML** (Data Manipulation Language)
 - Insertion, suppression et modification des données
- **DQL** (Data Query Language)
 - Sélection des données (requêtes)
- **DCL** (Data Control Language)
 - Gestion des accès aux tables
- **TCL** (Transactional Control Language)
 - Gestion des transactions dans les bases de données

Le langage SQL

Instructions SQL

- **Insensibles à la casse** (pas de différence entre majuscules et minuscules)
- **se terminent par ;**
- Une instruction peut s'écrire sur plusieurs lignes



4.1. Data Definition Language

Création, modification et suppression de tables

Contraintes et conventions de nommage

Les noms (de bases de données, tables, colonnes...) doivent :

- commencer par une lettre
- contenir entre 1 et 30 caractères
- contenir seulement **A–Z, a–z, 0–9, _, \$, et #**
- être différents (pour un même utilisateur)
- ne pas être des mots réservés



Comme SQL n'est pas sensible à la casse, nous proposons de ne pas utiliser la notation camel-case mais plutôt la notation **snake-case** pour mettre en évidence le début de chaque mot composant un nom.

Exemple : `date_de_naissance` >< `DateDeNaissance`

On évitera aussi les caractères spéciaux comme les accents. Ce qui aura pour avantage la portabilité du code.

Créer une base de données

```
CREATE DATABASE <nom_base_de_données>;
```

Pour spécifier la syntaxe des instructions SQL, nous allons utiliser des symboles particuliers et appliquer des conventions :

Les **mots réservés** seront proposés en **majuscules**. Mais comme SQL n'est pas sensible à la casse, tu peux aussi les écrire en minuscules.

Tu dois remplacer toute expression <...> par tes **propres valeurs**.



Exemple

```
create database henallux ;
```

Créer une table

```
CREATE TABLE <nom_table> (  
  <nom_colonne> <type_colonne> ,  
  ... );
```

Définition de colonne :

À répéter autant de fois qu'il y a de colonnes en séparant par des virgules

Quelques types de colonnes :

- **char** : 1 seul caractère
- **varchar(longueur)** : chaîne de caractères (nombre maximal de caractères)
- **numeric(longueur)** : entier (nombre maximal de chiffres)
- **decimal(précision,échelle)** : nombre réel avec :
 - précision : nombre maximal de chiffres (ceux avant la virgule + ceux après la virgule)
 - échelle : nombre de chiffres après la virgule
- **date**

Créer une table

Exemple

Livre
Titre
NombrePages
DateParution
Prix

```
create table livre (  
  titre varchar(100),  
  nombre_pages numeric(4),  
  date_parution date,  
  prix decimal(6,2) );
```



prix decimal(6,2) !?!

La colonne prix est de type réel de maximum 6 chiffres au total, dont 2 après la virgule!




Valeur par défaut pour une colonne

```
CREATE TABLE <nom_table> (  
  <nom_colonne> <type_colonne> [ DEFAULT <valeur_par_défaut> ],  
  ... );
```

Valeur par défaut de la colonne

Dans la syntaxe des instructions SQL, une expression entre [et] signifie qu'elle est **optionnelle** : tu peux ne pas utiliser cette partie de l'instruction. Attention, si tu l'utilises, tu ne dois pas écrire les [] !

Exemple



Article
Nom
Poids
Unité

```
create table article (  
  nom varchar(30),  
  poids decimal(6,2),  
  unite varchar(15) default 'kilo' );
```

Contraintes de colonne et de table

Des contraintes d'intégrité peuvent être ajoutées lors de la création de la table via les mots réservés suivants :

- **NOT NULL** : obligatoire
- **PRIMARY KEY** : identifiant principal (clé primaire)
- **UNIQUE** : valeur unique (appelé aussi clé secondaire)
- **FOREIGN KEY** : clé étrangère
- **CHECK** : contraintes additionnelles

Contraintes de colonne et de table

```
CREATE TABLE <nom_table> (  
    <nom_colonne> <type> [ DEFAULT <valeur> ] [<contrainte_de_colonne>] ,  
    ... ,  
    [<contrainte_de_table> , ←  
    ... ] ) ;
```

*À répéter autant de fois qu'il y a de contraintes de table
en séparant par des virgules*

Contrainte de colonne

Porte sur une seule colonne et se note à la fin de la définition de la colonne :

- **NOT NULL** : rend la colonne obligatoire
- **PRIMARY KEY** : définit la colonne comme identifiant (clé primaire)
 - Contient implicitement NOT NULL ⇒ colonne obligatoire
- **UNIQUE** : interdit à 2 valeurs de la colonne d'être les mêmes
 - La colonne peut être facultative
- **FOREIGN KEY** : définit la colonne comme clé étrangère
[FOREIGN KEY] REFERENCES <table>(<colonne_identifiante>)
- **CHECK** : pour définir des contraintes additionnelles
CHECK <conditions>

Contrainte de colonne

Exemple

Personne
<u>Matricule</u>
Nom
Prénom
DateNaissance
Gsm[0..1]
Email[0..1]

```
create table personne (  
    matricule varchar(6) primary key,  
    nom varchar(50) not null,  
    prenom varchar(25) not null,  
    date_naissance date not null,  
    gsm varchar(15),  
    email varchar(50) unique);
```

Colonne facultative
+ s'il y a des emails, ils sont tous différents

Contrainte de table

Peut porter sur une combinaison de colonnes

Se note en dehors des définitions de colonnes (par exemple, à la fin de la création de la table) :

- **PRIMARY KEY** : définit une ou plusieurs colonnes comme identifiant
PRIMARY KEY (<colonne>, ...)
- **UNIQUE** : interdit que 2 lignes aient les mêmes valeurs pour une ou plusieurs colonnes
UNIQUE (<colonne>, ...)
- **FOREIGN KEY** : définit une ou plusieurs colonnes comme clé étrangère référençant une autre table
FOREIGN KEY (<colonne>, ...) REFERENCES <table>(<colonne>, ...)
- **CHECK** : pour définir des contraintes additionnelles
CHECK (<conditions>)

Contrainte de table

Exemple

Localité
<u>Numéro</u>
Nom
CodePostal

```
create table Localite (  
    numero numeric(6),  
    nom varchar(50),  
    code_postal numeric(4),  
    primary key (numero),  
    unique(nom, code_postal) );
```

Il n'y a pas deux localités avec la même combinaison nom + code postal

Notons qu'il est possible de déclarer une clé primaire composée de plusieurs attributs, mais nous ne le verrons pas dans le cadre de ce cours.



Check

Permet de restreindre les valeurs permises dans les colonnes en précisant des conditions sur ces colonnes

Exemple

Article
Nom
Poids

Avec comme contrainte : le poids doit être > 0

Soit via une contrainte de colonne :

```
create table article (  
  nom varchar(30),  
  poids decimal(6,2) check (poids > 0) );
```

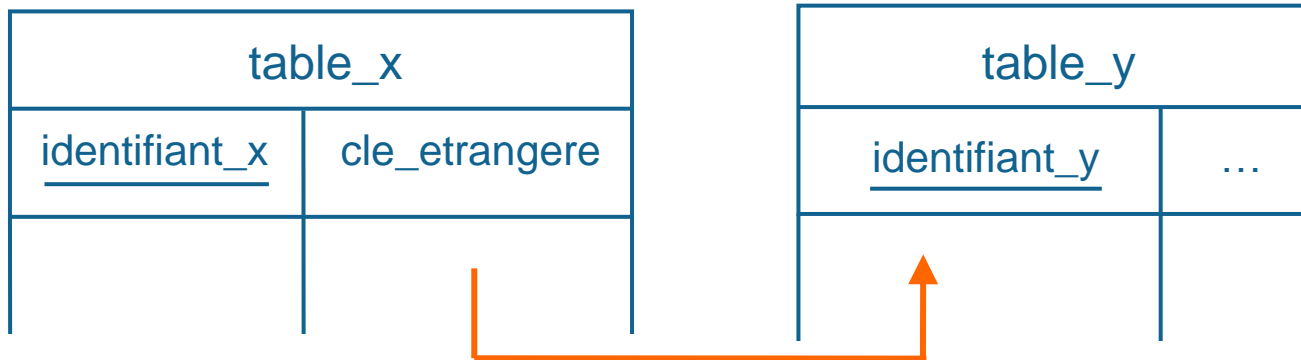
Soit via une contrainte de table :

```
create table article (  
  nom varchar(30),  
  poids decimal(6,2) ,  
  check (poids > 0) );
```

N.B. Syntaxe des conditions du check :

⇒ voir syntaxe des conditions des requêtes (cf. chapitre 5.Requêtes simples)

Clé étrangère



Soit via une contrainte de colonne

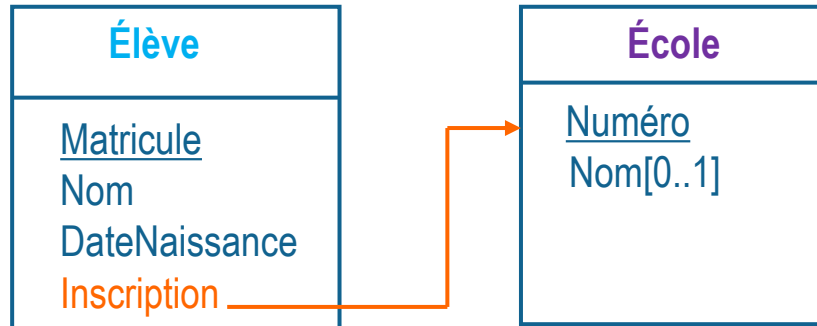
```
create table table_x (  
  identifiant_x <type1> primary key,  
  cle_etrangere <type2> references table_y (identifiant_y) );
```

Soit via une contrainte de table

```
create table table_x (  
  identifiant_x <type1> primary key,  
  cle_etrangere <type2>,  
  foreign key (cle_etrangere) references table_y (identifiant_y) );
```

Clé étrangère

Exemple



```
create table ecole (  
    numero numeric(3) primary key,  
    nom varchar(30) );
```

```
create table eleve (  
    matricule varchar(5) primary key,  
    nom varchar(40) not null,  
    date_naissance date not null,  
    inscription numeric(3),  
    foreign key (inscription) references ecole (numero) );
```

Modifier la structure d'une table

ALTER TABLE <nom_table>

- **ADD** <définition_colonne>, ←
- ...
- **ALTER COLUMN** <définition_colonne>
- **DROP COLUMN** <nom_colonne>

Définition de colonne :
À répéter autant de fois qu'il y a
de colonnes à ajouter en
séparant par des virgules

Exemple

```
alter table eleve  
  add nom_rue varchar(80),  
      numero_dans_rue numeric(4) ;
```

```
alter table ecole  
  alter column nom varchar(100) not null;
```

```
alter table eleve  
  drop column date_naissance ;
```

Supprimer une table

Supprimer la structure d'une table (et son contenu)

```
DROP TABLE <nom_table> ;
```

Exemple

```
drop table eleve;
```

Supprimer toutes les lignes d'une table (en gardant la structure de la table)

```
TRUNCATE TABLE <nom_table> ;
```




4.2. Data Manipulation Language

Insertion, modification et suppression de données

Insérer des données

```
INSERT INTO <nom_table> [ ( <nom_colonne>, ... ) ]  
VALUES      ( <valeur>, ... );
```



On peut ne pas préciser les noms des colonnes lors d'une insertion, mais il faut alors impérativement écrire les valeurs à insérer dans l'ordre correspondant à l'ordre de création des colonnes. Ce qui peut être source d'erreurs.

Insérer des données

Personne
<u>Matricule</u>
Nom
Prénom
DateNaissance
Gsm[0..1]
Email[0..1]

```
create table personne (  
    matricule varchar(6) primary key,  
    nom varchar(50) not null,  
    prenom varchar(25) not null,  
    date_naissance date not null,  
    gsm varchar(15),  
    email varchar(50));
```

Exemples d'instruction d'insertion dans la table personne

```
insert into personne  
values ('abc123','Dupond','Jean','01/01/1980','0497562314','j.dup@gmail.com');  
  
insert into personne (matricule,nom,prenom,date_naissance,gsm,email)  
values ('def456','Ledux','Marie','03/23/1996',null,'mledux@gmail.com');  
  
insert into personne (matricule,nom,prenom,date_naissance,email)  
values ('ghi789','Grant','Jason','10/08/1996','jasgrant@gmail.com');  
  
insert into personne  
values ('jkl123','Petit','Axel','11/10/1976',null,null);
```

Valeur
inconnue

Pas de
gsm

Insérer des données

Les valeurs de type chaîne de caractères sont placées entre quotes (' et ').
Attention, les dates introduites via des chaînes de caractères doivent l'être avec précaution, car elles dépendent du format par défaut de la date (cf. inversion des mois et des jours en anglais par rapport au français)



Insérer des données

Le mot clé DEFAULT peut être utilisé pour insérer la valeur par défaut d'une colonne

Exemple

Article
Nom
Poids
Unité

```
create table article (  
  nom varchar(30),  
  poids decimal(6,2),  
  unite varchar(15) default 'kilo' );
```

```
insert into article (nom, poids, unite)  
values ('Éolienne XT300',1520,'tonne');
```

—————→ unite = tonne

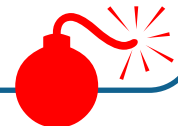
```
insert into article (nom, poids, unite)  
values ('Senseo Philipps',1.69,default);
```

—————→ unite = kilo

Modifier des données

```
UPDATE      <nom_table>
SET          <nom_colonne> = <valeur>,
            ...
[ WHERE      <conditions> ] ;
```

La clause WHERE permet de préciser quelles sont les lignes à modifier.
Attention, cette clause est optionnelle, **mais si on ne l'utilise pas,**
TOUTES LES LIGNES de la table seront **MODIFIÉES**.



Modifier des données

Suggestion :
Modifier une ligne à la fois en la sélectionnant sur base de sa clé primaire



Exemples

```
update personne  
set gsm = '0495124578', email = 'petiax@hotmail.com'  
where matricule = 'jkl123';
```



Une seule ligne modifiée

```
update personne  
set gsm = '0495124578', email = 'petiax@hotmail.com'
```



Toutes les lignes sont
modifiées

Supprimer des données

```
DELETE  
[ FROM ] <nom_table>  
[ WHERE <conditions> ] ;
```

Ici aussi, si on n'utilise pas la clause WHERE,
TOUTES LES LIGNES de la table seront **SUPPRIMÉES**.



Supprimer des données

Suggestion :

Supprimer une ligne à la fois en la sélectionnant sur base de sa clé primaire



Exemples

```
delete from personne  
where matricule ='jkl123';
```



Une seule ligne supprimée

```
delete from personne;
```



Toutes les lignes sont supprimées