Structured Query Language

Création : table

Création de table : **CREATE TABLE**, nom de la table, intitulés des attributs avec leur type.

Création : table

Création de table : **CREATE TABLE**, nom de la table, intitulés des attributs avec leur type.

Exemple

```
CREATE TABLE Figures (
id_f INT,
nom VARCHAR(255),
prénom VARCHAR(255),
naissance YEAR,
nationalité VARCHAR(255),
);
```

Création : table

Création de table : **CREATE TABLE**, nom de la table, intitulés des attributs avec leur type.

Exemple

```
CREATE TABLE Figures (
id_f INT,
nom VARCHAR(255),
prénom VARCHAR(255),
naissance YEAR,
nationalité VARCHAR(255),
);
```

Il existe un grand nombre de types (int, varchar (chaînes de caractères), date, etc...), ne pas hésiter à aller lire la documentation !

Et les clés dans tout ça?

Clé Primaire : **PRIMARY KEY**

id_f **INT**,
PRIMARY KEY (id_f)

Et les clés dans tout ça?

Clé Primaire : **PRIMARY KEY**

id_f **INT**,
PRIMARY KEY (id_f)

Clé Étrangère : **FOREIGN KEY REFERENCES**

(dans une autre table)
id_f **INT**,
FOREIGN KEY (ID_Cel) **REFERENCES** Figures(ID_f)

Insertion

Ajouter une nouvelle donnée dans une table se fait avec les mots-clés **INSERT INTO**, le nom de la table, et le mot-clé **VALUES** suivi des données à insérer.

Insertion

Ajouter une nouvelle donnée dans une table se fait avec les mots-clés **INSERT INTO**, le nom de la table, et le mot-clé **VALUES** suivi des données à insérer.

Exemple

INSERT INTO Figures **VALUES** (1, “Lovelace”, “Ada”, 1815, “britannique”);

Insertion

Ajouter une nouvelle donnée dans une table se fait avec les mots-clés **INSERT INTO**, le nom de la table, et le mot-clé **VALUES** suivi des données à insérer.

Exemple

INSERT INTO Figures **VALUES** (1, “Lovelace”, “Ada”, 1815, “britannique”);

Attention !!

Les données doivent être du bon type, et entrées dans le même ordre que les attributs de la table !

Exercice

- Rajouter les figures suivantes à notre table : George Boole, logicien britannique né en 1815, et Grace Hopper, informaticienne américaine née en 1906.

Projection/Sélection

Nous allons chercher à accéder aux éléments de la base de données, en les sélectionnant.
Le mot-clé est **SELECT**, suivi des attributs qui nous intéressent, puis de **FROM** et le nom de la table. On peut sélectionner sur les colonnes (**projection**) ou sur les lignes (**sélection**).

Projection/Sélection

Nous allons chercher à accéder aux éléments de la base de données, en les sélectionnant.
Le mot-clé est **SELECT**, suivi des attributs qui nous intéressent, puis de **FROM** et le nom de la table. On peut sélectionner sur les colonnes (**projection**) ou sur les lignes (**sélection**).

Exemple

SELECT nom **FROM** Figures;

Projection/Sélection

Nous allons chercher à accéder aux éléments de la base de données, en les sélectionnant.
Le mot-clé est **SELECT**, suivi des attributs qui nous intéressent, puis de **FROM** et le nom de la table. On peut sélectionner sur les colonnes (**projection**) ou sur les lignes (**sélection**).

Exemple

SELECT nom **FROM** Figures;

Choisir ses colonnes (projection)

- **SELECT** * permet de prendre toutes les colonnes
- Plusieurs colonnes : **SELECT** Col1, Col2, ... **FROM** ...;
- Pas de doublons : **DISTINCT** (**SELECT DISTINCT** nationalité **FROM** Célébrités;)

Projection/Sélection

Choisir ses lignes : le mot-clé **WHERE** (sélection)

- Le mot-clé **WHERE** permet d'ajouter des conditions sur les données
- Exemple d'utilisation : **SELECT** * **FROM** Figures **WHERE** naissance $>$ 1900;
- Plusieurs conditions avec les mots-clés **AND**, **OR**, et **NOT**

Projection/Sélection

Choisir ses lignes : le mot-clé **WHERE** (sélection)

- Le mot-clé **WHERE** permet d'ajouter des conditions sur les données
- Exemple d'utilisation : **SELECT** * **FROM** Figures **WHERE** naissance $>$ 1900;
- Plusieurs conditions avec les mots-clés **AND**, **OR**, et **NOT**

Ordonner ses données : **ORDER BY**

- Ordre croissant : **SELECT** * **FROM** Figures **ORDER BY** naissance;
- Ordre décroissant : **SELECT** * **FROM** Figures **ORDER BY** naissance **DESC**;

Projection/Sélection

Des fonctions d'agrégation permettent d'effectuer des calculs sur les données. Par exemple : **MIN**, **MAX**, **SUM**, et **COUNT**.

Retourner la date de naissance la plus ancienne

SELECT MIN(naissance) **FROM** Figures

Projection/Sélection

Des fonctions d'agrégation permettent d'effectuer des calculs sur les données. Par exemple : **MIN**, **MAX**, **SUM**, et **COUNT**.

Retourner la date de naissance la plus ancienne

SELECT MIN(naissance) **FROM** Figures

Exercice

- Écrire la requête sélectionnant tous les *noms* et *prénoms* des figures de l'informatique.
- Écrire la requête sélectionnant toutes les *Figures* américaines.
- Écrire la requête sélectionnant tous les noms et prénoms des figures classées selon leur date de naissance : de la plus actuelle à la plus ancienne.
- Compter le nombre de figures présentes dans la base de données.

Nouvelle table, pour la suite

Pour les besoins de démonstration, créons la nouvelle table Avancées :

Avancées

```
CREATE TABLE Avancées (
id_a INTEGER,
date_Inv YEAR,
nom VARCHAR(255),
id_f VARCHAR(255),
PRIMARY KEY (id_a),
FOREIGN KEY (id_f) REFERENCES Figures(id_f)
);
```

Jointure

Il est possible de recouper des données entre plusieurs tables différentes, au moyen d'une *jointure*.

Jointure

Il est possible de recouper des données entre plusieurs tables différentes, au moyen d'une *jointure*.

Syntaxe d'une jointure

SELECT * **FROM** Figures **JOIN** Avancées **ON** Avancées.id_f = Figures.id_f

Jointure

Il est possible de recouper des données entre plusieurs tables différentes, au moyen d'une *jointure*.

Syntaxe d'une jointure

SELECT * **FROM** Figures **JOIN** Avancées **ON** Avancées.id_f = Figures.id_f

Nota Bene

Mettre le nom de la table devant l'attribut (cf POO) est une bonne pratique, nécessaire lorsque le même nom d'attribut est utilisé dans les différentes tables.

Jointure

Il est possible de recouper des données entre plusieurs tables différentes, au moyen d'une *jointure*.

Syntaxe d'une jointure

SELECT * **FROM** Figures **JOIN** Avancées **ON** Avancées.id_f = Figures.id_f

Nota Bene

Mettre le nom de la table devant l'attribut (cf POO) est une bonne pratique, nécessaire lorsque le même nom d'attribut est utilisé dans les différentes tables.

Autres types de jointures

D'autres jointures sont possibles, mais dépassent le cadre du programme. Celle-ci est une jointure interne (**INNER JOIN**).

Jointure

Autre exemple : liste des avancées associées à des figures nées après 1900

SELECT Avancées.Nom **FROM** Inventions **JOIN** Figures **ON** Avancées.id_f = Figures.id_f **AND** Figures.naissance > 1900

Jointure

Il est possible d'utiliser des **alias** qui rendent l'écriture des jointures plus facile.

Jointure

Il est possible d'utiliser des **alias** qui rendent l'écriture des jointures plus facile.

Liste des avancées associées à des figures nées après 1900

SELECT Avancées.nom **FROM** Avancées as A **JOIN** Figures as F **ON** A.id_f = F.id_f **AND** F.naissance > 1900

Exercice

- Écrire la requête renvoyant les noms des figures associées à une avancée d'après 1850 ainsi que le nom de cette avancée.
- Écrire la requête renvoyant la date des avancées des figures britanniques.

Mise à jour/suppression

Mettre à jour une donnée : **UPDATE SET**

UPDATE Figures **SET** prenom = “Alan Mathison ” **WHERE** nom = “Turing”;

Mise à jour/suppression

Mettre à jour une donnée : **UPDATE SET**

UPDATE Figures **SET** prenom = “Alan Mathison ” **WHERE** nom = “Turing”;

Supprimer une donnée : **DELETE**

DELETE FROM Figures **WHERE** Prénom = “George”;

Exercice

- La recherche est souvent plus un travail d'équipe qu'individuel. On se propose de le préciser en changeant le nom de l'avancée "Déchiffrement d'Enigma" en "Déchiffrement d'Enigma (avec son équipe)".
- Supprimer la figure dont la clé primaire est 4.