

Introduction générale aux bases de données relationnelles

Table des matières

I. Contexte	3
II. Base de données	3
III. Exercice : Appliquer la notion	5
IV. Système de gestion de bases de données (SGBD)	5
V. Exercice : Appliquer la notion : Enquête Wikipédia	7
VI. Introduction à DB Fiddle	7
VII. Exercice : Appliquer la notion	11
VIII. Application de base de données	11
IX. Exercice : Appliquer la notion	12
X. Donnée (en relationnel) : table, objet, propriété, domaine, atomicité	13
XI. Exercice : Appliquer la notion	14
XII. Langage de données : l'exemple du langage SQL	15
XIII. Exercice : Appliquer la notion	16
XIV. Essentiel	17
XV. Auto-évaluation	17
A. Exercice final	17
B. Défi	21
1. Exercice : Créer une table.....	21
2. Exercice : Alimenter et interroger une la table.....	22
3. Exercice : Notion de contraintes	23
4. Exercice : Notion de références.....	24
5. Exercice : Projection, restriction et jointure.....	25
6. Exercice : Fonctions et agrégats.....	26
Contenus annexes	28
Solutions des exercices	28

I. Contexte

Durée : 3h

Environnement de travail : DB Fiddle

Pré-requis : Aucun

Contexte

Les *BD* sont nées à la fin des années 1960 pour combler les lacunes des systèmes de fichiers et faciliter la gestion qualitative et quantitative des données informatiques. Les *SGBD* sont des applications informatiques permettant de créer et de gérer des *BD* (comme Oracle ou PostgreSQL par exemple).

Les *BD* relationnelles, issues de la recherche de Codd chez IBM, sont celles qui ont connu le plus grand essor depuis les années 1970, et qui reste encore aujourd'hui les plus utilisées. On utilise des *SGBDR* pour les implémenter. Le langage *SQL* est le langage commun à tous les *SGBDR*, ce qui permet de concevoir des *BD* relativement indépendamment des systèmes utilisés.

Les usages de *BD* se sont aujourd'hui généralisés pour entrer dans tous les secteurs de l'entreprise, depuis les petites bases utilisées par quelques personnes dans un service pour des besoins de gestion de données locales, jusqu'aux bases qui gèrent de façon centralisée des données partagées par tous les acteurs de l'entreprise.

L'accroissement de l'utilisation du numérique comme outil de manipulation de toutes données (bureautique, informatique applicative, etc.) et comme outil d'extension des moyens de communication (réseaux), ainsi que les évolutions technologiques (puissance des PC, Internet, etc.) ont rendu indispensable, mais aussi complexifié la problématique des *BD*.

Les conséquences de cette généralisation et de cette diversification des usages se retrouvent dans l'émergence de solutions conceptuelles et technologiques nouvelles, notamment les bases de données du mouvement NoSQL, particulièrement utilisées par les grands acteurs du web.

Plan du cours

Ce module introduit les concepts fondamentaux des bases de données relationnelles. Qu'est-ce qu'une base de données ? un système de gestion de bases de données ? une donnée relationnelle ? une application de base de données ?

Il permet également de découvrir l'*IDE* web DB Fiddle¹ permettant d'écrire et exécuter du code *SQL* afin de créer des bases de données relationnelles.

II. Base de données

Objectifs

- Connaître le concept de données ;
- Connaître le concept de bases de données.

¹ <https://www.db-fiddle.com/>

Mise en situation

La base de l'informatique, c'est de travailler sur des données. Tout est donnée : des caractères que vous entrez au clavier aux photos que vous prenez avec votre téléphone, en passant par les couleurs des pixels de votre écran.

Les données sont donc aussi la base des applications web : un message, un contact, un profil, une commande, etc. Chacun de ces exemples a un point commun : une structure pré-définie. Un contact aura un nom et un prénom, une commande aura un prix et une adresse d'expédition, un message aura une date d'envoi et un auteur, etc. Comment les stocker efficacement ?

C'est précisément dans ce but que l'on utilise des bases de données : elles facilitent grandement le stockage d'un ensemble de données qui répondent à une structure logique.

Logiciel et données

Un logiciel informatique est composé de programmes, c'est à dire :

- d'instructions,
- et de données auxquelles s'appliquent ces instructions.

Exemple

Par exemple un logiciel de traitement de texte est composé de fonctions :

- ouvrir, copier, coller, insérer une image, changer la police, enregistrer, etc.,
- et de fichiers sur lesquels elles s'appliquent. Dans ce cas les fichiers de traitement de texte sont les données.

Définition

Définition lâche de base de données : un ensemble de données

On appelle parfois base de données tout ensemble de données stocké numériquement et pouvant servir à un ou plusieurs programmes. De ce point de vue des fichiers sur un disque dur, un fichier de tableur, voire un fichier de traitement de texte peuvent constituer des bases de données.

Définition

Définition restreinte de base de données : un ensemble de données structuré

On appellera base de données un ensemble de données numériques qui possède une structure ; c'est à dire dont l'organisation répond à une **logique** systématique.

On parlera de modèle logique de données pour décrire cette structure.

Exemple

Base de données relationnelle

Une base de données relationnelle permet d'organiser les données en tableaux (appelés relations).

Base de données de classification classique des espèces animales

espèce	eucaryote	multicellulaire	propriété
bactéries	false	false	
archées	false	false	
protistes	true	false	
champignons	true	true	décompose
végétaux	true	true	photosynthétise
animaux	true	true	ingère

Fondamental Fonctions d'une base de données

Une base de données est structurée afin de pouvoir mieux répondre à des fonctions fondamentales en informatique, telles que :

- Stocker l'information de façon fiable (c'est à dire être capable de restituer l'information entrée dans le système).
- Traiter de grands volumes de données (massification).
- Traiter rapidement les données (optimisation).
- Sécuriser les accès aux données (gérer les autorisations selon les utilisateurs).
- Contrôler la qualité des données (par exemple la cohérence par rapport à un modèle pré-établi).
- Partager les données (entre plusieurs applications dédiées à plusieurs métiers).
- Rendre accessible les données en réseau (gérer la concurrence des accès parallèles).
- Etc.

Exercice : Appliquer la notion

Parmi les raisons suivantes, lesquelles justifient l'utilisation d'une base de données ?

- ☐ S'abstraire des modalités de stockage physique de l'information.
- ☐ Minimiser la redondance d'information.
- ☐ Faciliter la mise à jour de données.
- ☐ Contrôler la cohérence des données.
- ☐ Sécuriser l'accès aux données.
- ☐ Mieux gérer l'accès concurrent aux données.

III. Système de gestion de bases de données (SGBD)**Objectifs**

- Connaître le concept de *SGBD*;
- Connaître les SGBD les plus utilisés ;
- Connaître la différence entre SGBD relationnel et SGBD non-relationnel.

Mise en situation

Lorsque l'on développe une application, on a souvent besoin de stocker des données, qui ont, la plupart du temps, des liens entre elles. Sur un réseau social, il y a des personnes qui peuvent interagir entre elles, s'envoyer des messages, se suivre, etc. Lorsque ces données deviennent très grandes, de l'ordre de plusieurs millions, il est inimaginable de les stocker dans un simple fichier, ou un tableur, qui ferait office de base de données.

Les systèmes de gestion de base de données, ou SGBD, sont des outils qui aident à manipuler des bases de données. Leur fonction est double : permettre de décrire la structure des données a priori, et gérer la cohérence des données et des liens entre elles.

Définition **Système de Gestion de Bases de Données**

Un *SGBD* est un logiciel qui prend en charge la structuration, le stockage, la mise à jour et la maintenance d'une base de données. Il est l'unique interface entre les informaticiens et les données (définition des schémas, programmation des applications), ainsi qu'entre les utilisateurs et les données (consultation et mise à jour).

Exemple **Exemples de SGBD**

- Oracle est un SGBD commercial **relationnel** et relationnel-objet situé parmi les leaders du marché dans le domaine des applications métiers.
- PostgreSQL est un SGBD libre **relationnel** puissant qui offre une alternative libre (licence BSD) aux solutions commerciales d'Oracle ou IBM. PostgreSQL dispose d'une documentation d'excellente qualité et d'un support très proche du standard SQL, ce qui en fait également un bon outil d'apprentissage.
- Access est un SGBD **relationnel** commercial, qui offre une interface graphique permettant de concevoir rapidement des applications de petite envergure ou de réaliser des prototypes.
- MongoDB est un SGBD **non-relationnel** libre (licence Apache) orienté document. Il permet de gérer facilement de très grandes quantités de données réparties sur de nombreux ordinateurs. Il se base sur le format JSON.
- MySQL est un SGBD **relationnel** open source (licence GPL et commerciale) très utilisé pour la réalisation de sites web dynamiques. Depuis la version 4 MySQL implémente la plupart des fonctions attendues d'un SGBD relationnel. MySQL a son pendant libre, MariaDB.
- SQLite est un SGBD **non-relationnel** libre. C'est un système très léger, utilisé pour des applications embarquées. Il est entièrement écrit en C et comporte la majorité des fonctionnalités attendues de SQL.

Fondamental **Objectifs des SGBD**

- **Indépendance physique des données**

Le changement des modalités de stockage de l'information (optimisation, réorganisation, segmentation, etc.) n'implique pas de changements des programmes.

- **Indépendance logique des données**

L'évolution de la structure d'une partie des données n'influe pas sur l'ensemble des données.

- **Manipulation des données par des non-informaticiens**

L'utilisateur n'a pas à savoir comment l'information est stockée et calculée par la machine, mais juste à pouvoir la rechercher et la mettre à jour à travers des IHM ou des langages *assertionnels* simples.

- **Administration facilitée des données**

Le SGBD fournit un ensemble d'outils (dictionnaire de données, audit, tuning, statistiques, etc.) pour améliorer les performance et optimiser les stockages.

- **Optimisation de l'accès aux données**

Les temps de réponse et de débits globaux sont optimisés en fonctions des questions posées à la BD.

- **Contrôle de cohérence (intégrité sémantique) des données**

Le SGBD doit assurer à tout instant que les données respectent les règles d'intégrité qui leurs sont imposées.

- **Partageabilité des données**

Les données sont simultanément consultables et modifiables.

- **Sécurité des données**

La confidentialité des données est assurée par des systèmes d'authentification, de droits d'accès, de chiffrement des mots de passe, etc.

- **Sûreté des données**

La persistance des données, même en cas de panne, est assurée, grâce typiquement à des sauvegardes et des journaux qui gardent une trace persistante des opérations effectuées.

Complément

Pourquoi des SGBD ? (cf. p.28)

Complément **SGBD relationnel et non-relationnel**

Jusqu'au début des années 2000, la plupart des bases de données étaient relationnelles ; mais avec l'arrivée des géants du web (Google, Amazon ou Facebook), entreprises qui gèrent des quantités énormes de données, s'est développé un mouvement important de développement de bases de données non-relationnelles, également appelées NoSQL pour *Not-Only SQL*.

Exercice : Appliquer la notion : Enquête Wikipédia

À partir de la page Wikipédia sur les SGBD, classer les SGBDR en fonction de leur licence : fr.wikipedia.org/wiki/Système_de_gestion_de_base_de_données¹.

SQLite

MariaDB

Oracle Database

Microsoft SQL Server

PostgreSQL

DB2

Microsoft Access

MySQL

Logiciel libre

Logiciel Freemium

Logiciel propriétaire

IV. Introduction à DB Fiddle

Objectifs

- Savoir utiliser l'interpréteur SQL en ligne DB Fiddle.

¹ https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_gestion_de_base_de_donn%C3%A9es

Mise en situation

Il existe une grande diversité de systèmes de gestion de base de données : chacun est bien adapté à une situation donnée : de gros volumes, comme pour un moteur de recherche ? Besoin d'une cohérence parfaite, comme pour des transactions bancaires ?

Avant de faire un choix, il est utile de tester les différents SGBD, prendre en main leur syntaxe et choisir celui qui répond à vos besoins.

Mais installer tous les SGBD sur votre machine peut-être long et pénible. Dans ce module, vous apprendrez à utiliser DBFiddle, une application web qui permet de tester plusieurs SGBD sans aucune installation préalable.

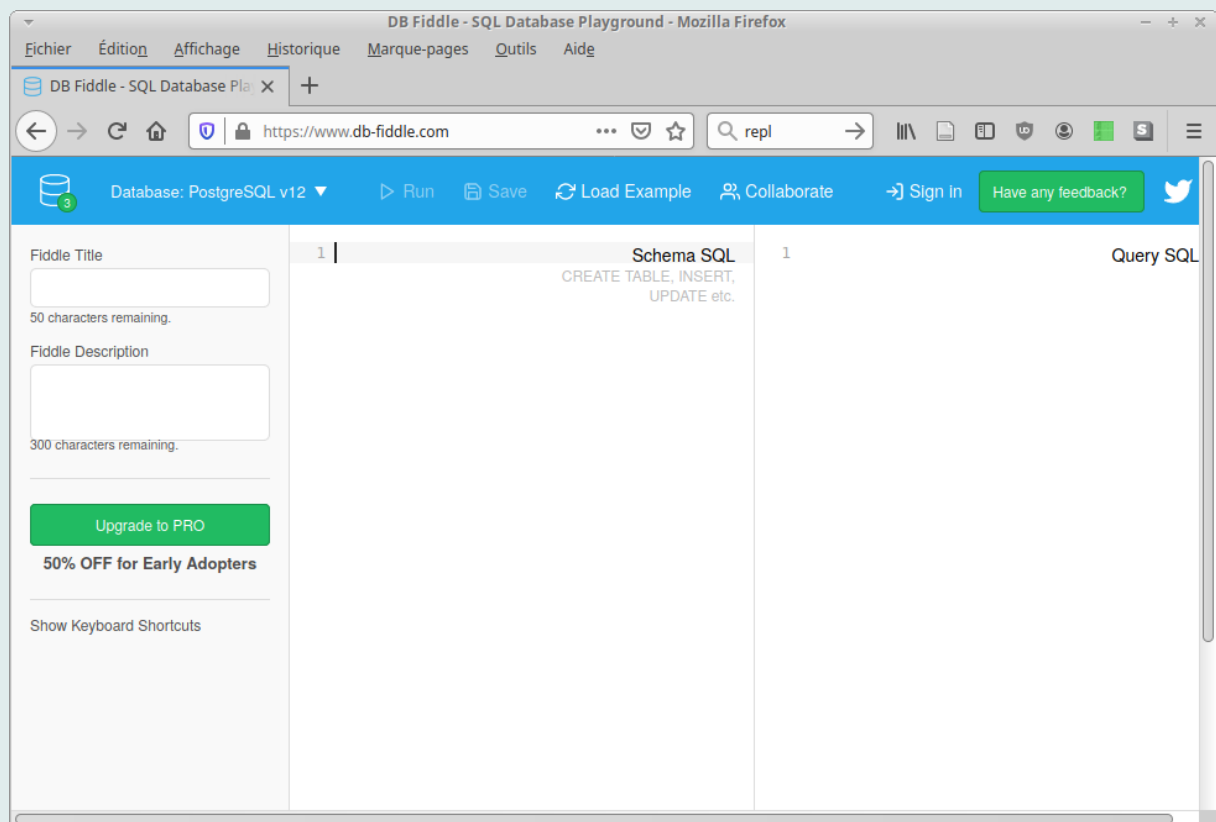
Définition Interpréteur SQL

Un interpréteur SQL est un programme qui traite des commandes fournies par l'utilisateur pour interagir avec une base de données. On parle parfois de *REPL* pour *Return Eval Print Loop*.

Définition DB Fiddle

DB Fiddle est un interpréteur SQL en ligne, il est disponible sur le Web et il permet de créer et exécuter du code SQL sans installer de SGBD.


db-fiddle.com¹

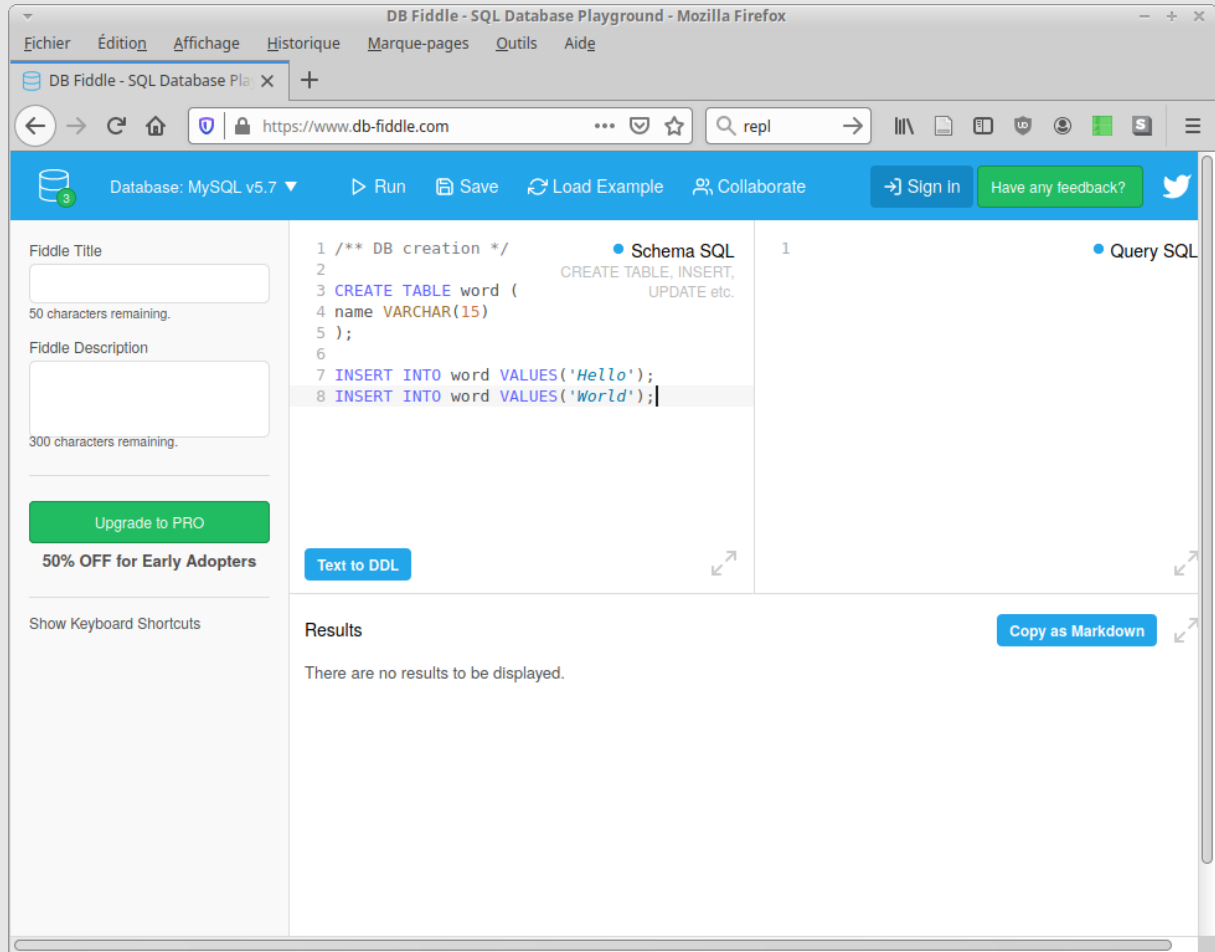


Interface d'accueil DB Fiddle

¹ <https://www.db-fiddle.com/>

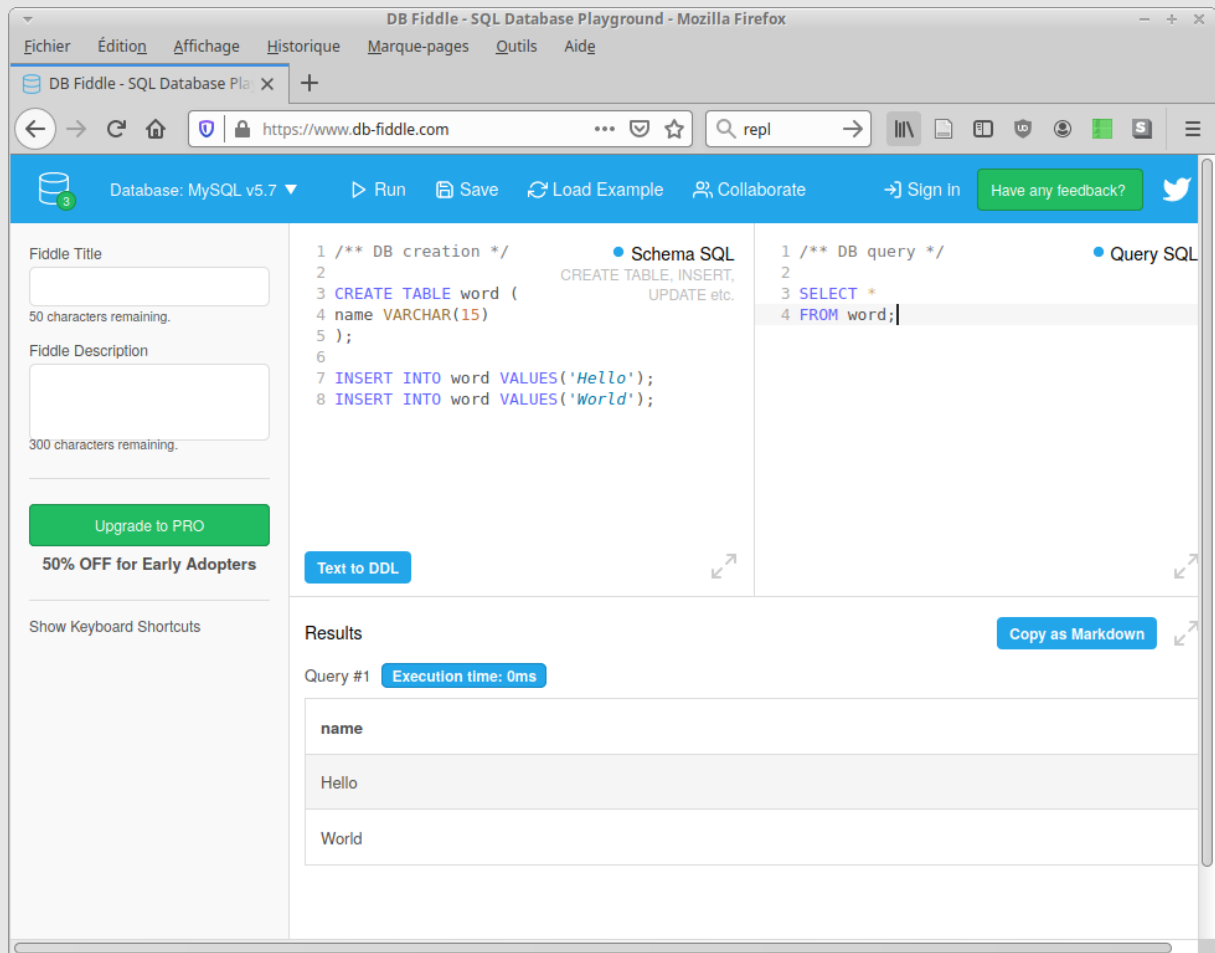
Méthode **Créer une BD**

Pour créer une base de données (commande SQL CREATE TABLE) ou gérer les données (commande SQL INSERT, UPDATE, DELETE) il faut insérer le code SQL correspondant dans la partie gauche puis cliquer sur  **Run** (ou CTRL+RETURN au clavier).



Méthode Interroger une BD

Pour poser des question à la base de données (commande SQL SELECT) il faut ajouter le code SQL correspondant dans la partie droite puis cliquer sur **Run** (ou CTRL+RETURN au clavier).



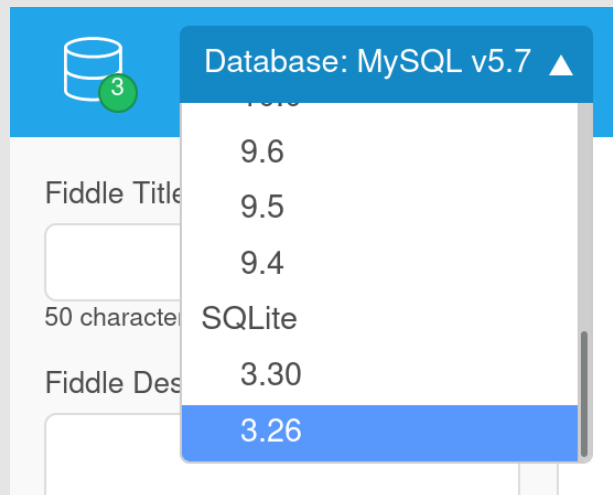
Exemple

Il est possible de tester DB Fiddle avec ce code de type *Hello World*.

```
1 /** DB creation */
2
3 CREATE TABLE word (
4   name VARCHAR(15)
5 );
6
7 INSERT INTO word VALUES('Hello');
8 INSERT INTO word VALUES('World');
9
10 /** DB query */
11
12 SELECT *
13 FROM word;
```

Méthode Choisir un SGBD

En haut à gauche un menu déroulant permet de choisir le type de SGBD à utiliser. On travaillera par défaut avec la dernière version de PostgreSQL ou de MySQL.



Menu déroulant du clic droit sur interpréteur

Complément DB Disco

DB Disco est une alternative à DB Fiddle limitée à PostgreSQL.
pic.crzt.fr/dbdisco¹

Exercice : Appliquer la notion

Copier ces commandes dans un interpréteur SQL.

```

1 CREATE TABLE liste(
2 nombre INT
3 );
4
5 INSERT INTO liste VALUES (2);
6 INSERT INTO liste VALUES (7);
7 INSERT INTO liste VALUES (3);
1 SELECT MAX(nombre) FROM liste;
```

Quel est le résultat de l'exécution de ce code ?

V. Application de base de données

Objectifs

- Connaître la différence entre une application et une base de données ;
- Savoir citer quelques exemples d'application de bases de données.

¹ <http://pic.crzt.fr/dbdisco>

Mise en situation

Avez-vous déjà utilisé une base de données ? Lorsque vous effectuez des achats en ligne, avez-vous l'impression de manipuler directement une base de données ?

Probablement pas, car une base de données seule n'est pas directement utilisable par un utilisateur humain : elle n'est utilisable que par les informaticiens qui connaissent son langage de programmation et par les applications qui ont été programmées pour s'en servir.

Une base de donnée va donc souvent de pair avec une application, qui est l'interface entre les utilisateurs et les données elles-mêmes.

Définition Application de base de données

On appelle application de base de données un logiciel informatique permettant à un utilisateur final de manipuler (lire ou écrire) les données d'une base de données.

Exemple Application web

Une application web est composée d'interfaces en HTML qui permettent d'écrire et de lire des données dans une base de données, via un langage applicatif, comme par exemple PHP.

Exemple Application web Mastodon

L'application *Mastodon* est composée d'interfaces web permettant d'entrer des données (saisir son profil, *tooter* un post, *booster* un post, etc. et de sortir des données (consulter son fil *Mastodon*, faire une recherche sur un *hashtag*, sur un utilisateur) d'une base de données.

Ces informations (profil des utilisateurs, *toots*, *hashtags*, etc.) sont stockées dans des bases de données des serveurs fédérés de *Mastodon*.

Exemple Application de bureau Access

Avec un logiciel comme Access on peut réaliser à la fois une base de données et une application permettant de manipuler cette base de données pour des besoins bureautiques simples.

Exemple Une application en programmation

Dans le cas de programmes informatiques, on peut stocker des données dans une base pour effectuer des traitements plus efficacement. Par exemple, si on dispose d'un tableau de nombres, on peut faire la somme des éléments de colonnes bien plus facilement, rapidement et sûrement qu'en écrivant soit même un programme.

Exemple Compagnie aérienne

Une base de données de gestion de l'activité d'une compagnie aérienne concerne les voyageurs, les vols, les avions, le personnel, les réservations, etc.

Une application à partir d'une telle base de données pourra permettre la gestion des réservations, des disponibilités des avions en fonction des vols à effectuer, des affectations des personnels volants, etc.

Exercice : Appliquer la notion

Parmi les options suivantes, lesquelles correspondent à des applications de bases de données ?

- ☐ Twitter
- ☐ SQLite
- ☐ Le logiciel d'une caisse enregistreuse de supermarché
- ☐ Une application de SMS sur téléphone

VI. Donnée (en relationnel) : table, objet, propriété, domaine, atomicité

Objectifs

- Connaître le concept de relation (ou table) ;
- Savoir représenter des données simples en relationnel ;
- Connaître le concept de donnée atomique.

Mise en situation

Prenez une application de vente de livres en ligne : quelles sont les données qu'il faut gérer ? En vrac, on peut penser aux livres, aux auteurs, aux clients, aux éditeurs, etc.

Si vous deviez réaliser une base de données simplifiée pour stocker ces différentes entités, il est probable que vous dessiniez des tableaux : un tableau pour les auteurs, un tableau pour les clients, etc.

Et ça tombe bien, car ce sont exactement ces tableaux que l'on appelle relations, et qui sont à la base du modèle relationnel.

Rappel Base de données relationnelle

Une base de données relationnelle permet d'organiser les données en tables (appelées relations).
Chaque case de la table contient une information atomique.

Définition Ligne (objet)

Chaque ligne de la table correspond à un objet que l'on veut gérer dans la base de données : une voiture, une personne, une espèce, etc.

On parle aussi d'**enregistrement**.

Fondamental

Toutes les lignes d'une même table correspondent à des objets du même type, donc dans une table, on met soit des voitures, soit des personnes, mais on ne mélange pas les deux.

Définition Colonne (propriété)

Chaque colonne de la table correspond à une propriété des objets qui se trouvent dans la table ; tous les objets de la table partagent donc les mêmes propriétés.

On parle aussi d'**attribut**.

Définition **Domaine**

Chaque colonne de la table est associée à un domaine de valeur fixé **a priori**, par exemple : entier, texte, booléen, etc.

On parle aussi de **type** de données.

Définition **Cellule (donnée en relationnel)**

Une donnée en relationnel, c'est le contenu d'une cellule d'une table, qui correspond à la propriété d'un objet.

Une table ou relation (en relationnel)

propriété 1 domaine : d1	propriété 2 domaine : d2	...
objet1, donnée 1	objet1, donnée 2	...
objet2, donnée 1	objet2, donnée 2	...
...

Exemple

Exemple de relation instanciée

espèce domaine : texte	eucaryote domaine : booléen	...
bactéries	false	...
archées	false	...
...

Attention **Atomicité (contre-exemple)**

Pour que la base de données fonctionne correctement on veille à ne mettre qu'une seule donnée par cellule, c'est le **principe d'atomicité** en relationnel.

Un mauvais exemple de relation : les données ne sont pas atomiques (il y a plusieurs données par case de la table)

espèce, domaine : texte
bactéries : procaryotes unicellulaires
archées : procaryotes unicellulaires
protistes : eucaryotes unicellulaires
champignons : eucaryotes multicellulaires qui décomposent
végétaux : eucaryotes multicellulaires qui photosynthétisent
animaux : eucaryotes multicellulaires qui ingèrent

Exercice : Appliquer la notion

Ordonner les mots pour compléter la table suivante :

*Billets
de
train*

Horodatage	Numéro Train	Prix payé (en €)	Classe
31-12-2019 14:35	67394	142.90	1
12-12-2019 14:35	68173	(3)	1
01-04-2019 9:30	(2)	53.90	2
(1)	67399	34.90	2
04-12-2019 13:50	68137	3.30	(4)

128.90

62940

03-08-2019 10:20

2

VII. Langage de données : l'exemple du langage SQL

Objectifs

- Connaître la notion de langage orienté donnée ;
- Savoir tester une instruction en SQL.

Mise en situation

Imaginez : vous développez une bibliothèque musicale, et vous avez identifié la structure des données que vous souhaitez gérer : les artistes d'un côté, les albums de l'autre, les playlists des utilisateurs, etc.

Au moment de passer à la pratique, vous vous demandez alors : comment expliquer cette structure au système de gestion de base de données ? Comment insérer de nouvelles données ? Comment récupérer les données existantes ?

La réponse tient en trois mots : grâce à SQL. Ce langage est le couteau suisse des bases de données, et permet aux développeurs de communiquer avec les SGBD.

Définition Langage de données

Un langage de données est un langage informatique permettant de décrire et de manipuler les schémas et les données d'une BD.

Remarque Synonyme

On parle aussi de langage orienté données.

Fondamental SQL

SQL est le langage consacré aux SGBD relationnels et relationnels-objet.

Il permet de :

- créer des tables, en définissant le domaine de chaque colonne ;
- insérer des lignes dans les tables ;
- lire les données entrées dans la base de données.

Exemple Création de table en SQL (définition du schéma de données)

```
1 CREATE TABLE student (
2 number INTEGER PRIMARY KEY,
3 name TEXT,
4 city TEXT
5 );
```

Cette instruction permet de créer une relation `student` comportant les propriétés `number`, `name` et `city` de domaines, respectivement, entier, texte et texte (`number` est la clé primaire de la table, il servira à identifier les enregistrements).

Exemple Insertion de ligne en SQL (création de données)

```
1 INSERT INTO student (number, name, city)
2 VALUES (1, 'Holmes', 'Londres');
```

Cette instruction permet de créer l'étudiant numéro 1, de nom Holmes qui habite la ville de Londres.

Exemple Manipulation de données en SQL (exploitation des données)

```
1 SELECT name
2 FROM student
3 WHERE city = 'Londres';
```

Cette instruction permet de rechercher les noms de tous les étudiants habitant la ville de Londres.

Complément Autres langages de données

- XQuery est un langage de données mobilisé dans les bases de données arborescentes XML.
- Les bases NoSQL proposent des langages de données spécifiques, souvent inspirés du SQL. Par exemple le langage de MongoDB permet de manipuler une base de contenus JSON.

Exercice : Appliquer la notion

Exécuter l'instruction textuelle SQL ci-dessous qui, permet de créer une table dans une base de données.

```
1 CREATE TABLE composition (
2 aliment TEXT,
3 calories FLOAT,
4 joules FLOAT,
5 glucides FLOAT,
6 protides FLOAT,
7 lipides FLOAT
8 );
9
```



```
10 INSERT INTO composition(aliment, calories, joules, glucides, protides, lipides)
11 VALUES ('Abricot', 277, 1158, 63.4, 4.6, 0.4);
12
1 SELECT aliment
2 FROM composition;
```

Quelle valeur de retour est obtenue ?

VIII. Essentiel

Toute application a besoin de stocker des données. Mais une donnée, c'est assez abstrait, et si chaque application invente son propre système de stockage, la perte de temps et le risque d'erreurs sont immenses.

On utilise donc des bases de données. Lorsque les données sont structurées *a priori*, comme c'est le cas pour des personnes ou des musiques, on parle de base de données relationnelles.

Les systèmes de gestion de bases de données sont des outils qui permettent aux informaticiens de manipuler des bases de données. Comment ? Grâce au langage SQL, qui permet de communiquer avec quasiment tous les SGBD, en ayant besoin d'apprendre une seule syntaxe.

Pour tester les différents SGBD sans avoir à les installer sur votre ordinateur, vous pouvez utiliser l'application web DBFiddle.

IX. Auto-évaluation

A. Exercice final

Exercice : Synonymes

Exercice

Quels sont les synonymes du mot **relation** au sens d'une base de données relationnelle ?

- ☐ table
- ☐ tableau
- ☐ lien
- ☐ association
- ☐ ami

Exercice

Quels sont les synonymes du mot **ligne** au sens d'une base de données relationnelle ?

- ☐ propriété
- ☐ objet
- ☐ enregistrement
- ☐ table

Exercice

Quels sont les synonymes du mot **colonne** au sens d'une base de données relationnelle ?

- ☐ valeur
- ☐ type
- ☐ propriété
- ☐ attribut

Exercice

Quels sont les synonymes du mot **donnée** au sens d'une base de données relationnelle ?

- ☐ colonne
- ☐ cellule
- ☐ case
- ☐ ligne

Exercice

Quels sont les synonymes du mot **domaine** au sens d'une base de données relationnelle ?

- ☐ propriété
- ☐ type
- ☐ données
- ☐ ensemble de valeurs

Exercice

Classer les SGBD suivants selon qu'ils sont essentiellement relationnel ou non-relationnel.

S'aider de Wikipédia le cas échéant.

Microsoft Access

MySQL

Microsoft SQL Server

Cassandra

Neo4j

MariaDB

Redis

DB2

OrientDB

Elasticsearch

4D

MongoDb

PostgreSQL

SGBD relationnel

SGBD non-relationnel

Exercice

Utiliser un interpréteur SQL en ligne comme DB Fiddle pour donner le résultat de l'exécution des codes SQL ci-après.

Exercice

```

1 CREATE TABLE adresse (
2   pk_id INTEGER PRIMARY KEY,
3   nom VARCHAR(30),
4   prenom VARCHAR(30),
5   code_postal INTEGER,
6   ville VARCHAR(30)
7 );
8
9
10 INSERT INTO adresse (pk_id, nom, prenom, code_postal, ville) VALUES (1, 'Boulgakov',
    'Mikhaïl', 60200, 'Compiègne');
11 INSERT INTO adresse (pk_id, nom, prenom, code_postal, ville) VALUES (2, 'Tolstoï', 'Alexis',
    60420, 'Dompierre');
12 INSERT INTO adresse (pk_id, nom, prenom, code_postal, ville) VALUES (4, 'Gogol', 'Nikolaï',
    60420, 'Mery-la-Bataille');
13 INSERT INTO adresse (pk_id, nom, prenom, code_postal, ville) VALUES (5, 'Pouchkine',
    'Alexandre', 60680, 'Canly');
1 SELECT ville
2 FROM adresse
3 WHERE nom='Tolstoï';

```

Exercice

```

1 CREATE TABLE adresse (
2   pk_id INTEGER PRIMARY KEY,
3   nom VARCHAR(30),
4   prenom VARCHAR(30),
5   code_postal INTEGER,
6   ville VARCHAR(30)
7 );
8
9
10 INSERT INTO adresse (pk_id, nom, prenom, code_postal, ville) VALUES (1, 'Boulgakov',
    'Mikhaïl', 60200, 'Compiègne');
11 INSERT INTO adresse (pk_id, nom, prenom, code_postal, ville) VALUES (2, 'Tolstoï', 'Alexis',
    60420, 'Dompierre');
12 INSERT INTO adresse (pk_id, nom, prenom, code_postal, ville) VALUES (4, 'Gogol', 'Nikolaï',
    60420, 'Mery-la-Bataille');
13 INSERT INTO adresse (pk_id, nom, prenom, code_postal, ville) VALUES (5, 'Pouchkine',
    'Alexandre', 60680, 'Canly');
1 SELECT MAX(code_postal)
2 FROM adresse;

```

Exercice

```

1 CREATE TABLE voiture(
2   pk_immatriculation CHAR(7) PRIMARY KEY,
3   modele VARCHAR(30),
4   marque VARCHAR(30),
5   couleur VARCHAR(30)
6 );
7
8
9 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('AA123AA', 'Clio',
10  'Renault', 'Noir');
11 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('AB123NB', '807',
12  'Peugeot', 'Bleu');
13 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('DE001TR', 'Clio',
14  'Renault', 'Rouge');
15 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('AM007JB', '205',
16  'Peugeot', 'Rose');
17 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('BK2000B',
18  'Cayenne', 'Porsche', 'Noir');
19 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('ZX987FR', 'Twingo',
20  'Renault', 'Jaune');
21
22 SELECT couleur
23 FROM voiture
24 WHERE modele = '205';

```

Exercice

```

1 CREATE TABLE voiture(
2   pk_immatriculation CHAR(7) PRIMARY KEY,
3   modele VARCHAR(30),
4   marque VARCHAR(30),
5   couleur VARCHAR(30)
6 );
7
8
9 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('AA123AA', 'Clio',
10  'Renault', 'Noir');
11 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('AB123NB', '807',
12  'Peugeot', 'Bleu');
13 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('DE001TR', 'Clio',
14  'Renault', 'Rouge');
15 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('AM007JB', '205',
16  'Peugeot', 'Rose');
17 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('BK2000B',
18  'Cayenne', 'Porsche', 'Noir');
19 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('ZX987FR', 'Twingo',
20  'Renault', 'Jaune');
21
22 SELECT LEFT(pk_immatriculation, 2)
23 FROM voiture
24 WHERE couleur='Noir' AND marque='Renault'

```

Exercice

Indiquer quels attributs sont atomiques dans la table question ci-dessous.

```

1 CREATE TABLE question (
2   label VARCHAR(255) PRIMARY KEY,
3   answer VARCHAR(255) NOT NULL,
4   questionnaire VARCHAR(255) NOT NULL

```

```

5);
6
7 INSERT INTO question VALUES (
8   '1. Qui était roi de France de 742 à 814 ?',
9   'Charlemagne (Carolingiens)',
10  'Histoire'
11);
12
13 INSERT INTO question VALUES (
14  '2. Qui était roi de France de 940 à 996 ?',
15  'Hugues Capet (Capétiens)',
16  'Histoire'
17);
18
19 INSERT INTO question VALUES (
20  '3. Qui était roi de France de 1462 à 1515 ?',
21  'Louis XII (Valois)',
22  'Histoire'
23);

```

label

answer

questionnaire

Attribut atomique

Attribut non atomique

B. Défi

Ce défi peut être réalisé avec un SGBD PostgreSQL ou MySQL ou un interpréteur en ligne tel que DB Fiddle ou DB Disco

1. Exercice : Créer une table

Créer la base de données correspondant aux instructions suivantes :

```

1 CREATE TABLE student (
2   ss_number VARCHAR(13) PRIMARY KEY,
3   univ_number VARCHAR(20) UNIQUE NOT NULL,
4   name VARCHAR(50) NOT NULL,
5   firstname VARCHAR(50)
6 );

```

Exercice

Que fait cette instruction CREATE ?

- ☐ Elle crée une relation nommée `student`.
- ☐ Elle crée une table nommée `student`.
- ☐ Elle crée une base de données `student`.

Exercice

Combien d'attributs possède la relation `student` ?

Exercice

Quel est le domaine de l'attribut `ss_number` ?

- ☐ Chaîne de caractères
- ☐ Entier
- ☐ Booléen
- ☐ Date

2. Exercice : Alimenter et interroger une la table

Créer et initialiser la base de données correspondant aux instructions suivantes :

```
1 CREATE TABLE student (
2   ss_number VARCHAR(13) PRIMARY KEY,
3   univ_number VARCHAR(20) UNIQUE NOT NULL,
4   name VARCHAR(50) NOT NULL,
5   firstname VARCHAR(50)
6 );

1 INSERT INTO student (ss_number, univ_number, name, firstname)
2 VALUES ('1800675001066', 'AB3937098X', 'Dupont', 'Pierre');
3
4 INSERT INTO student (ss_number, univ_number, name, firstname)
5 VALUES ('2820475001124', 'XGB67668', 'Durand', 'Anne');
```

Exercice

Que font ces instructions INSERT ?

- ☐ Elles ajoutent deux objets dans la relation `student`.
- ☐ Elles ajoutent deux enregistrements dans la relation `student`.
- ☐ Elles ajoutent deux tables dans la base de données.

Exercice

Quel est le nom de famille de l'étudiant qui a pour numéro de sécurité sociale de 1800675001066 ?

Exercice

Que fait cette instruction ?

```
1 SELECT univ_number, name, firstname
2 FROM student;
```

- ☐ Elle retourne les valeurs de **tous** les attributs de **tous** les enregistrements présents dans la table `student`.
- ☐ Elle retourne les valeurs d'une **partie** des attributs de **tous** les enregistrements présents dans la table `student`.
- ☐ Elle retourne les valeurs d'une **partie** des attributs d'une **partie** des enregistrements présents dans la table `student`.

Exercice

Que fait cette instruction ?

```
1 SELECT *
2 FROM student
```

```
3 WHERE ss_number = '2820475001124';
```

- ☐ Elle retourne l'intégralité du contenu de la table `student`.
- ☐ Elle retourne les valeurs des attributs de l'enregistrement dont l'attribut `ss_number` a pour valeur 2820475001124.

Exercice

Que fait cette instruction ?

```
1 SELECT name, firstname
2 FROM student
3 WHERE ss_number = '2820475001124';
```

- ☐ Une projection
- ☐ Une restriction

3. Exercice : Notion de contraintes

Créer et initialiser la base de données correspondant aux instructions suivantes :

```
1 CREATE TABLE student (
2 ss_number VARCHAR(13) PRIMARY KEY,
3 univ_number VARCHAR(20) UNIQUE NOT NULL,
4 name VARCHAR(50) NOT NULL,
5 firstname VARCHAR(50)
6 );

1 INSERT INTO student (ss_number, univ_number, name, firstname)
2 VALUES ('1800675001066', 'AB3937098X', 'Dupont', 'Pierre');
3
4 INSERT INTO student (ss_number, univ_number, name, firstname)
5 VALUES ('2820475001124', 'XGB67668', 'Durand', 'Anne');
```

Contrainte de domaine

Ajouter cette instruction :

```
1 INSERT INTO student (ss_number, univ_number, name, firstname)
2 VALUES ('XXXXXXXXXXXXXX', 'XXXXXX', 'Dupont', 'Pierre');
```

Pourquoi le système a-t-il renvoyé une erreur ?

- ☐ `ss_number` doit comporter des chiffres.
- ☐ `univ_number` doit avoir pour valeur un nombre.
- ☐ L'étudiant nommé Pierre Dupont est déjà présent dans la base de données.
- ☐ La valeur fournie pour `ss_number` contient un nombre trop important de caractères.

Contraintes d'unicité

Ajouter ces instructions :

```
1 INSERT INTO student (ss_number, univ_number, name, firstname)
2 VALUES ('1800675001066', 'HGYT67655Y', 'Duchemin', 'Pierre');
3
4 INSERT INTO student (ss_number, univ_number, name, firstname)
5 VALUES ('2810592012232', 'XGB67668', 'Durand', 'Anne');
6
7 INSERT INTO student (ss_number, univ_number, firstname)
8 VALUES ('2950192013333', 'HGYT67655Y', 'Aline');
```

Pourquoi ces insertions sont-elle rejetées ?

- ☐ La première instruction est rejetée car le `ss_number` existe déjà avec la valeur 1800675001066.
- ☐ La seconde instruction est rejetée car le `univ_number` existe déjà avec la valeur XGB67668.
- ☐ La troisième instruction est rejetée car `name` n'est pas renseigné.

Exercice

Pourrait-on insérer dans la table une seconde personne qui aurait le prénom « *Aline* » et le nom « *Duchemin* » ?

- ☐ Oui
- ☐ Non

4. Exercice : Notion de références

Créer et initialiser la base de données correspondant aux instructions suivantes :

```

1 CREATE TABLE student (
2   ss_number VARCHAR(13) PRIMARY KEY,
3   univ_number VARCHAR(20) UNIQUE NOT NULL,
4   name VARCHAR(50) NOT NULL,
5   firstname VARCHAR(50)
6 );

1 INSERT INTO student (ss_number, univ_number, name, firstname)
2 VALUES ('1800675001066', 'AB3937098X', 'Dupont', 'Pierre');
3
4 INSERT INTO student (ss_number, univ_number, name, firstname)
5 VALUES ('2820475001124', 'XGB67668', 'Durand', 'Anne');

1 CREATE TABLE course (
2   code VARCHAR(4) NOT NULL,
3   student VARCHAR(13) NOT NULL,
4   PRIMARY KEY (code, student),
5   FOREIGN KEY (student) REFERENCES student(ss_number)
6 );

1 INSERT INTO course (code, student)
2 VALUES ('NF17', '1800675001066');
3
4 INSERT INTO course (code, student)
5 VALUES ('NF26', '1800675001066');
6
7 INSERT INTO course (code, student)
8 VALUES ('NF29', '1800675001066');
9
10 INSERT INTO course (code, student)
11 VALUES ('NF17', '2820475001124');
```

Exercice

Quels sont les cours suivis par Pierre Dupont ?

- ☐ NF17
- ☐ NF26
- ☐ NF29

Contraintes d'intégrité référentielle

Ajouter ces deux instructions :

```
1 INSERT INTO course (code, student)
2 VALUES ('NF26', '2820475001124');
3
4 INSERT INTO course (code, student)
5 VALUES ('NF17', '1700792001278');
```

Laquelle de ces instructions provoque une erreur ?

- ☐ La première renvoie une erreur.
- ☐ La seconde renvoie une erreur.

5. Exercice : Projection, restriction et jointure

Créer et initialiser la base de données correspondant aux instructions suivantes :

```
1 CREATE TABLE student (
2 ss_number VARCHAR(13) PRIMARY KEY,
3 univ_number VARCHAR(20) UNIQUE NOT NULL,
4 name VARCHAR(50) NOT NULL,
5 firstname VARCHAR(50)
6 );

1 INSERT INTO student (ss_number, univ_number, name, firstname)
2 VALUES ('1800675001066', 'AB3937098X', 'Dupont', 'Pierre');
3
4 INSERT INTO student (ss_number, univ_number, name, firstname)
5 VALUES ('2820475001124', 'XGB67668', 'Durand', 'Anne');

1 CREATE TABLE course (
2 code VARCHAR(4) NOT NULL,
3 student VARCHAR(13) NOT NULL,
4 PRIMARY KEY (code, student),
5 FOREIGN KEY (student) REFERENCES student(ss_number)
6 );

1 INSERT INTO course (code, student)
2 VALUES ('NF17', '1800675001066');
3
4 INSERT INTO course (code, student)
5 VALUES ('NF26', '1800675001066');
6
7 INSERT INTO course (code, student)
8 VALUES ('NF29', '1800675001066');
9
10 INSERT INTO course (code, student)
11 VALUES ('NF17', '2820475001124');
```

Exercice

Que réalise cette instruction ?

```
1 SELECT *
2 FROM student
3 WHERE name = 'Dupont';
```

- ☐ Projection
- ☐ Restriction
- ☐ Produit
- ☐ Jointure

Exercice

Que réalise cette instruction ?

```
1 SELECT name, firstname
2 FROM student;
```

- ☐ Projection
- ☐ Restriction
- ☐ Produit
- ☐ Jointure

Exercice

Que réalise cette instruction ?

```
1 SELECT *
2 FROM student s JOIN course c
3 ON c.student = s.ss_number;
```

- ☐ Projection
- ☐ Restriction
- ☐ Produit
- ☐ Jointure

Exercice

Que réalise cette instruction ?

```
1 SELECT *
2 FROM student, course;
```

- ☐ Projection
- ☐ Restriction
- ☐ Produit
- ☐ Jointure

6. Exercice : Fonctions et agrégats

Créer et initialiser la base de données correspondant aux instructions suivantes :

```
1 CREATE TABLE student (
2   ss_number VARCHAR(13) PRIMARY KEY,
3   univ_number VARCHAR(20) UNIQUE NOT NULL,
4   name VARCHAR(50) NOT NULL,
5   firstname VARCHAR(50)
6 );

1 INSERT INTO student (ss_number, univ_number, name, firstname)
2 VALUES ('1800675001066', 'AB3937098X', 'Dupont', 'Pierre');
3
4 INSERT INTO student (ss_number, univ_number, name, firstname)
5 VALUES ('2820475001124', 'XGB67668', 'Durand', 'Anne');

1 CREATE TABLE course (
2   code VARCHAR(4) NOT NULL,
3   student VARCHAR(13) NOT NULL,
```

```
4 PRIMARY KEY (code, student),
5 FOREIGN KEY (student) REFERENCES student(ss_number)
6);

1 INSERT INTO course (code, student)
2 VALUES ('NF17', '1800675001066');
3
4 INSERT INTO course (code, student)
5 VALUES ('NF26', '1800675001066');
6
7 INSERT INTO course (code, student)
8 VALUES ('NF29', '1800675001066');
9
10 INSERT INTO course (code, student)
11 VALUES ('NF17', '2820475001124');
```

Exercice

Quelle information renvoie l'instruction suivante :

```
1 SELECT COUNT(code)
2 FROM course
3 WHERE student = '1800675001066';
```

- ☐ Le nombre d'étudiants
- ☐ Le nombre de cours
- ☐ Le nombre d'étudiants suivant le cours 1800675001066.
- ☐ Le nombre de cours suivis par l'étudiant 1800675001066.

Exercice

Quelle information renvoie l'instruction suivante :

```
1 SELECT student, COUNT(code)
2 FROM course
3 GROUP BY student;
```

- ☐ Le nombre d'étudiants.
- ☐ Le nombre de cours.
- ☐ Pour chaque étudiant, le nombre de cours qu'il suit.
- ☐ Pour chaque cours, le nombre d'étudiants le suivant.

Exercice

Quelle est la requête qui retourne, pour chaque cours, le nombre d'étudiants inscrits ?

- ☐

```
SELECT code, COUNT(student)
FROM course
GROUP BY code;
```
- ☐

```
SELECT code, COUNT(code)
FROM course
GROUP BY student;
```

Contenus annexes

1. Pourquoi des SGBD ?

Jadis...

Avant l'avènement des SGBD, chaque application informatique dans l'entreprise impliquait sa propre équipe de développement, ses propres supports physiques, ses propres fichiers, ses propres normes, ses propres langages, etc.

Conséquences...

L'existence conjointe et croissante de ces applications indépendantes a des effets négatifs, tels que :

- La multiplication des tâches de saisie, de développement et de support informatique ;
- La redondance anarchique des informations dans les fichiers ;
- L'incohérence des versions simultanées de fichiers ;
- La non-portabilité des traitements en raison des différences dans les formats et langages ;
- La multiplication des coûts de développement et de maintenance des applications.

Problèmes...

Les conséquences précédemment citées se répercutent sur l'entreprise en générant des problèmes humains et matériels ;

- Coûts en personnels qualifiés et en formations ;
- Remise des pouvoirs de décision entre les mains de spécialistes informatiques ;
- Tout changement matériel ou logiciel a un impact sur les applications ;
- Tout changement de la structure des données nécessite de modifier les programmes.

Or...

En réalité les applications ne sont jamais totalement disjointes, des données similaires (le cœur de l'information d'entreprise) sont toujours à la base des traitements.

On peut citer typiquement :

- Les données comptables
- Les données clients et fournisseurs
- Les données relatives à la gestion des stocks
- Les données relatives aux livraisons
- Les données marketing et commerciales
- Les données relatives au personnel
- ...

Solutions des exercices

Exercice p. 5 Solution n°1

Parmi les raisons suivantes, lesquelles justifient l'utilisation d'une base de données ?

- ☒ S'abstraire des modalités de stockage physique de l'information.
- ☒ Minimiser la redondance d'information.
- ☒ Faciliter la mise à jour de données.
- ☒ Contrôler la cohérence des données.
- ☒ Sécuriser l'accès aux données.
- ☒ Mieux gérer l'accès concurrent aux données.

Exercice p. 7 Solution n°2

À partir de la page Wikipédia sur les SGBD, classer les SGBDR en fonction de leur licence : fr.wikipedia.org/wiki/Système_de_gestion_de_base_de_données¹.

Logiciel libre	Logiciel Freemium	Logiciel propriétaire
PostgreSQL	MySQL	Oracle Database
MariaDB		Microsoft Access
SQLite		Microsoft SQL Server
		DB2



- PostgreSQL, SQLite et MariaDB sont des logiciels libres : leur code est disponible ouvertement à l'inspection à l'exécution, et est modifiable.
- Oracle est une solution propriétaire gérée par l'entreprise éponyme. DB2 est la solution d'IBM. Microsoft propose un SGBDR classique (SQL Server) et une solution plus modeste pour des usages bureautique (Access).
- MySQL est une solution *freenium* d'Oracle : c'est un logiciel ouvert et libre mais il existe des licences propriétaires pour des usages commerciaux.

Exercice p. 11 Solution n°3

Copier ces commandes dans un interpréteur SQL.

```

1 CREATE TABLE liste(
2 nombre INT
3 );
4
5 INSERT INTO liste VALUES (2);
6 INSERT INTO liste VALUES (7);
7 INSERT INTO liste VALUES (3);
1 SELECT MAX(nombre) FROM liste;
```

¹ https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_gestion_de_base_de_donn%C3%A9es

Quel est le résultat de l'exécution de ce code ?

7



DB Fiddle - SQL Database Playground - Mozilla Firefox

Fichier Édition Affichage Historique Marque-pages Outils Aide

DB Fiddle - SQL Database Playground

Database: MySQL v5.7 Run Save Load Example Collaborate Sign In Have any feedback?

Fiddle Title
50 characters remaining.

Fiddle Description
300 characters remaining.

Upgrade to PRO
50% OFF for Early Adopters

Show Keyboard Shortcuts

Schema SQL
CREATE TABLE, INSERT, UPDATE etc.

```
1 CREATE TABLE liste(
2 nombre INT
3 );
4
5 INSERT INTO liste VALUES (2);
6 INSERT INTO liste VALUES (7);
7 INSERT INTO liste VALUES (3);
```

Query SQL

```
1 SELECT MAX(nombre) FROM liste;
```

Results

Query #1 Execution time: 1ms

MAX(nombre)
7

Exercice p. 12 Solution n°4

Parmi les options suivantes, lesquelles correspondent à des applications de bases de données ?

- ☒ Twitter
- ☐ SQLite
- ☒ Le logiciel d'une caisse enregistreuse de supermarché
- ☒ Une application de SMS sur téléphone

Exercice p. 14 Solution n°5

Ordonner les mots pour compléter la table suivante :

Billets
de
train

Horodatage	Numéro Train	Prix payé (en €)	Classe
31-12-2019 14:35	67394	142.90	1
12-12-2019 14:35	68173	(3)	1
01-04-2019 9:30	(2)	53.90	2
(1)	67399	34.90	2
04-12-2019 13:50	68137	3.30	(4)

03-08-2019 10:20	62940	128.90	2
------------------	-------	--------	---

Exercice p. 16 Solution n°6

Exécuter l'instruction textuelle SQL ci-dessous qui, permet de créer une table dans une base de données.

```

1 CREATE TABLE composition (
2 aliment TEXT,
3 calories FLOAT,
4 joules FLOAT,
5 glucides FLOAT,
6 protides FLOAT,
7 lipides FLOAT
8 );
9
10 INSERT INTO composition(aliment, calories, joules, glucides, protides, lipides)
11 VALUES ('Abricot', 277, 1158, 63.4, 4.6, 0.4);
12
13
14 SELECT aliment
15 FROM composition;
```

Quelle valeur de retour est obtenue ?

Abricot

Exercice p. 17 Solution n°7

Exercice

Quels sont les synonymes du mot **relation** au sens d'une base de données relationnelle ?

- ☒ table
- ☒ tableau
- ☐ lien
- ☐ association
- ☐ ami

Exercice

Quels sont les synonymes du mot **ligne** au sens d'une base de données relationnelle ?

- ☐ propriété
- ☒ objet

☒ enregistrement

☐ table

Exercice

Quels sont les synonymes du mot **colonne** au sens d'une base de données relationnelle ?

☐ valeur

☐ type

☒ propriété

☒ attribut

Exercice

Quels sont les synonymes du mot **donnée** au sens d'une base de données relationnelle ?

☐ colonne

☒ cellule

☒ case

☐ ligne

Exercice

Quels sont les synonymes du mot **domaine** au sens d'une base de données relationnelle ?

☐ propriété

☒ type

☐ données

☒ ensemble de valeurs

Exercice p. 18 Solution n°8

Classer les SGBD suivants selon qu'ils sont essentiellement relationnel ou non-relationnel.

S'aider de Wikipédia le cas échéant.

SGBD relationnel
PostgreSQL
Microsoft Access
MySQL
4D
MariaDB
Microsoft SQL Server

DB2	SGBD non-relationnel
	Neo4j
	MongoDb
	Cassandra
	OrientDB
	Redis
	Elasticsearch

Exercice p. 19 Solution n°9

Exercice

```

1 CREATE TABLE adresse (
2   pk_id INTEGER PRIMARY KEY,
3   nom VARCHAR(30),
4   prenom VARCHAR(30),
5   code_postal INTEGER,
6   ville VARCHAR(30)
7 );
8
9
10 INSERT INTO adresse (pk_id, nom, prenom, code_postal, ville) VALUES (1, 'Boulgakov',
11   'Mikhaïl', 60200, 'Compiègne');
12 INSERT INTO adresse (pk_id, nom, prenom, code_postal, ville) VALUES (2, 'Tolstoï', 'Alexis',
13   60420, 'Dompierre');
14 INSERT INTO adresse (pk_id, nom, prenom, code_postal, ville) VALUES (4, 'Gogol', 'Nikolaï',
15   60420, 'Mery-la-Bataille');
16 INSERT INTO adresse (pk_id, nom, prenom, code_postal, ville) VALUES (5, 'Pouchkine',
17   'Alexandre', 60680, 'Canly');
18
19 SELECT ville
20 FROM adresse
21 WHERE nom='Tolstoï';

```

Dompierre

Exercice

```

1 CREATE TABLE adresse (
2   pk_id INTEGER PRIMARY KEY,
3   nom VARCHAR(30),
4   prenom VARCHAR(30),
5   code_postal INTEGER,
6   ville VARCHAR(30)
7 );
8
9
10 INSERT INTO adresse (pk_id, nom, prenom, code_postal, ville) VALUES (1, 'Boulgakov',
11   'Mikhaïl', 60200, 'Compiègne');
12 INSERT INTO adresse (pk_id, nom, prenom, code_postal, ville) VALUES (2, 'Tolstoï', 'Alexis',
13   60420, 'Dompierre');
14 INSERT INTO adresse (pk_id, nom, prenom, code_postal, ville) VALUES (4, 'Gogol', 'Nikolaï',
15   60420, 'Mery-la-Bataille');

```

```

13 INSERT INTO adresse (pk_id, nom, prenom, code_postal, ville) VALUES (5, 'Pouchkine',
    'Alexandre', 60680, 'Canly');
1 SELECT MAX(code_postal)
2 FROM adresse;

```

60680

Exercice

```

1 CREATE TABLE voiture(
2 pk_immatriculation CHAR(7) PRIMARY KEY,
3 modele VARCHAR(30),
4 marque VARCHAR(30),
5 couleur VARCHAR(30)
6 );
7
8
9 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('AA123AA', 'Clio',
    'Renault', 'Noir');
10 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('AB123NB', '807',
    'Peugeot', 'Bleu');
11 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('DE001TR', 'Clio',
    'Renault', 'Rouge');
12 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('AM007JB', '205',
    'Peugeot', 'Rose');
13 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('BK2000B',
    'Cayenne', 'Porsche', 'Noir');
14 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('ZX987FR', 'Twingo',
    'Renault', 'Jaune');
15
1 SELECT couleur
2 FROM voiture
3 WHERE modele = '205';

```

Rose

Exercice

```

1 CREATE TABLE voiture(
2 pk_immatriculation CHAR(7) PRIMARY KEY,
3 modele VARCHAR(30),
4 marque VARCHAR(30),
5 couleur VARCHAR(30)
6 );
7
8
9 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('AA123AA', 'Clio',
    'Renault', 'Noir');
10 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('AB123NB', '807',
    'Peugeot', 'Bleu');
11 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('DE001TR', 'Clio',
    'Renault', 'Rouge');
12 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('AM007JB', '205',
    'Peugeot', 'Rose');
13 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('BK2000B',
    'Cayenne', 'Porsche', 'Noir');
14 INSERT INTO voiture (pk_immatriculation, modele, marque, couleur) VALUES ('ZX987FR', 'Twingo',
    'Renault', 'Jaune');
15
1 SELECT LEFT(pk_immatriculation, 2)
2 FROM voiture
3 WHERE couleur='Noir' AND marque='Renault'

```

AA

Exercice p. 20 Solution n°10

Indiquer quels attributs sont atomiques dans la table `question` ci-dessous.

```

1 CREATE TABLE question (
2   label VARCHAR(255) PRIMARY KEY,
3   answer VARCHAR(255) NOT NULL,
4   questionnaire VARCHAR(255) NOT NULL
5 );
6
7 INSERT INTO question VALUES (
8   '1. Qui était roi de France de 742 à 814 ?',
9   'Charlemagne (Carolingiens)',
10  'Histoire'
11 );
12
13 INSERT INTO question VALUES (
14   '2. Qui était roi de France de 940 à 996 ?',
15   'Hugues Capet (Capétiens)',
16   'Histoire'
17 );
18
19 INSERT INTO question VALUES (
20   '3. Qui était roi de France de 1462 à 1515 ?',
21   'Louis XII (Valois)',
22   'Histoire'
23 );

```

Attribut atomique	Attribut non atomique
questionnaire	label
	answer



- L'attribut `label` possède deux informations : le numéro de question et la question en elle même.
- L'attribut `answer` possède deux informations : le nom du roi et sa dynastie.

Exercice p. 21 Solution n°11

Exercice

Que fait cette instruction `CREATE` ?

- ☒ Elle crée une relation nommée `student`.
- ☒ Elle crée une table nommée `student`.
- ☐ Elle crée une base de données `student`.



Cette instruction crée une relation (aussi appelée table) dans une base de données, qui elle est déjà existante.

Une base de données relationnelle est principalement constituée de tables (ou « relations » d'où le nom de relationnel). Une table est basiquement un élément d'organisation de l'information constitué de colonnes (ou attributs) et de lignes (ou enregistrements).

Nous allons dans un premier temps créer le schéma d'une table, c'est à dire définir ses colonnes. Pour cela nous utiliserons l'instruction SQL LDD `CREATE`.

Exercice

Combien d'attributs possède la relation `student` ?

4

 La table `student` comporte quatre attributs.

Exercice


Quel est le domaine de l'attribut `ss_number` ?

☒ Chaîne de caractères

☐ Entier

☐ Booléen

☐ Date

 La table `student` comporte quatre attributs qui sont des chaînes de caractères.
Le premier `ss_number` a des valeurs qui ont une longueur maximale de 13 caractères.

Exercice p. 22 Solution n°12

Exercice

Que font ces instructions INSERT ?

☒ Elles ajoutent deux objets dans la relation `student`.

☒ Elles ajoutent deux enregistrements dans la relation `student`.

☐ Elles ajoutent deux tables dans la base de données.


 Cette instruction ajoute deux enregistrements ou objets dans la relation `tEtu`.

Une fois les colonnes de la table définies, nous pouvons en déclarer les lignes. Nous utilisons pour cela l'instruction SQL LMD INSERT.

Exercice

Quel est le nom de famille de l'étudiant qui a pour numéro de sécurité sociale de 1800675001066 ?

Dupont

 Le premier objet correspond à un étudiant nommé Pierre Dupont, de numéro de sécurité sociale 1800675001066 et de numéro étudiant AB3937098X. Le respect de la casse est important : la chaîne 'Dupont' n'est pas égale à la chaîne 'dupont'.

Exercice


Que fait cette instruction ?

```
1 SELECT univ_number, name, firstname
2 FROM student;
```

☐ Elle retourne les valeurs de **tous** les attributs de **tous** les enregistrements présents dans la table `student`.

☒ Elle retourne les valeurs d'une **partie** des attributs de **tous** les enregistrements présents dans la table `student`.

☐ Elle retourne les valeurs d'une **partie** des attributs d'une **partie** des enregistrements présents dans la table `student`.


 Cette instruction retourne les valeurs de trois des attributs des enregistrements présents dans la table à l'exception de l'attribut `ss_number`.

On appelle cette opération une projection.

Exercice

Que fait cette instruction ?


```
1 SELECT *
2 FROM student
3 WHERE ss_number = '2820475001124';
```

- ☐ Elle retourne l'intégralité du contenu de la table `student`.
- ☒ Elle retourne les valeurs des attributs de l'enregistrement dont l'attribut `ss_number` a pour valeur 2820475001124.
-  Elle ne retourne que les enregistrements dont l'attribut `ss_number` a pour valeur 2820475001124. On appelle cette opération une restriction.

Exercice

Que fait cette instruction ?

```
1 SELECT name, firstname
2 FROM student
3 WHERE ss_number = '2820475001124';
```

- ☒ Une projection
- ☒ Une restriction
-  Elle retourne les valeurs des attributs `name` et `firstname` de l'enregistrement dont l'attribut `ss_number` a pour valeur 2820475001124. Cette instruction réalise à la fois une instruction et une projection.


Exercice p. 23 Solution n°13

Contrainte de domaine

Ajouter cette instruction :

```
1 INSERT INTO student (ss_number, univ_number, name, firstname)
2 VALUES ('XXXXXXXXXXXXXXXX', 'XXXXXX', 'Dupont', 'Pierre');
```

Pourquoi le système a-t-il renvoyé une erreur ?

- ☐ `ss_number` doit comporter des chiffres.
- ☐ `univ_number` doit avoir pour valeur un nombre.
- ☐ L'étudiant nommé Pierre Dupont est déjà présent dans la base de données.
- ☒ La valeur fournie pour `ss_number` contient un nombre trop important de caractères.
-  La chaîne comporte ici 15 caractères alors que la longueur de celle-ci doit être de 13 caractères au plus. Lorsque l'on définit une table, on définit également des contraintes sur cette table, qui serviront à contrôler son **intégrité**, par rapport à des règles que l'on aura fixées.

C'est notamment le cas des contraintes de domaine, qui permettent de vérifier qu'une colonne prend ses valeurs parmi un ensemble déterminé (les chaînes de 10 caractères au plus, les entier de 1 à 1000, etc.).


Contraintes d'unicité

Ajouter ces instructions :

```
1 INSERT INTO student (ss_number, univ_number, name, firstname)
2 VALUES ('1800675001066', 'HGYT67655Y', 'Duchemin', 'Pierre');
3
4 INSERT INTO student (ss_number, univ_number, name, firstname)
5 VALUES ('2810592012232', 'XGB67668', 'Durand', 'Anne');
```


```
6
7 INSERT INTO student (ss_number, univ_number, firstname)
8 VALUES ('2950192013333', 'HGYT67655Y', 'Aline');
```

Pourquoi ces insertions sont-elle rejetées ?

- ☒ La première instruction est rejetée car le `ss_number` existe déjà avec la valeur 1800675001066.
 - ☒ La seconde instruction est rejetée car le `univ_number` existe déjà avec la valeur XGB67668.
 - ☒ La troisième instruction est rejetée car `name` n'est pas renseigné.
-  Les contraintes d'unicité permettent d'assurer que toutes les valeurs d'une colonne seront différentes pour chaque ligne. Elles se définissent en SQL par les mots clés PRIMARY KEY ou UNIQUE.

Exercice


Pourrait-on insérer dans la table une seconde personne qui aurait le prénom « Aline » et le nom « Duchemin » ?

- ☒ Oui
 - ☐ Non
-  Cela est possible car il n'y a pas de contrainte, seuls les numéros (sécurité sociale et numéro d'étudiant) doivent être uniques.

Exercice p. 24 Solution n°14

Exercice

Quels sont les cours suivis par Pierre Dupont ?

- ☒ NF17
 - ☒ NF26
 - ☒ NF29
-  Une base de données est en général constituée de plusieurs tables. Ces tables se référencent entre elles en utilisant une clé étrangère : c'est à dire qu'une des colonnes de la table est utilisée pour faire référence à la colonne d'une autre table.


Ici, on a créé une seconde table, qui permet d'associer des cours aux étudiants

Contraintes d'intégrité référentielle


Ajouter ces deux instructions :

```
1 INSERT INTO course (code, student)
2 VALUES ('NF26', '2820475001124');
3
4 INSERT INTO course (code, student)
5 VALUES ('NF17', '1700792001278');
```

Laquelle de ces instructions provoque une erreur ?

- ☐ La première renvoie une erreur.
 - ☒ La seconde renvoie une erreur.
-  La seconde renvoie une erreur : en effet, le numéro de sécurité sociale que l'on essaie d'insérer dans la table `course` n'est pas présent dans la table `student`.

Lorsque nous avons défini la table `course`, nous avons défini une contrainte supplémentaire, dite d'intégrité référentielle : contrainte de type FOREIGN KEY. Puisque l'attribut `student` de la table `course` est une clé étrangère référençant `student` alors une erreur est retournée.

 La contrainte d'intégrité référentielle permet d'être certain que les références sont cohérentes.


Exercice p. 25 Solution n°15

Exercice

Que réalise cette instruction ?

```
1 SELECT *  
2 FROM student  
3 WHERE name = 'Dupont';
```

- ☐ Projection
- ☒ Restriction
- ☐ Produit
- ☐ Jointure


 Cette instruction est une restriction, on pose une condition sur les données à récupérer. Ici, il s'agit de récupérer uniquement les lignes pour lesquelles le nom est Dupont.

Exercice

Que réalise cette instruction ?

```
1 SELECT name, firstname  
2 FROM student;
```

- ☒ Projection
- ☐ Restriction
- ☐ Produit
- ☐ Jointure


 Cette instruction est une projection, on ne sélectionne que certains attributs d'une table.

Exercice

Que réalise cette instruction ?

```
1 SELECT *  
2 FROM student s JOIN course c  
3 ON c.student = s.ss_number;
```

- ☐ Projection
- ☐ Restriction
- ☐ Produit
- ☒ Jointure

 Une instruction SELECT peut porter sur plusieurs tables à la fois. Ici on utilise une jointure.
Il s'agit d'une jointure car elle ne renvoie que les étudiants qui sont associés à un cours.


Exercice

Que réalise cette instruction ?

```
1 SELECT *
```

2 FROM student, course;

- ☐ Projection
- ☐ Restriction
- ☒ Produit
- ☐ Jointure

 Il s'agit d'un produit car il associe à chaque étudiant tous les cours de la table course.


Exercice p. 26 Solution n°16

Exercice

Quelle information renvoie l'instruction suivante :

```
1 SELECT COUNT (code)
2 FROM course
3 WHERE student = '1800675001066';
```

- ☐ Le nombre d'étudiants
- ☐ Le nombre de cours
- ☐ Le nombre d'étudiants suivant le cours 1800675001066.
- ☒ Le nombre de cours suivis par l'étudiant 1800675001066.

 L'instruction SELECT permet également d'effectuer des calculs qui portent sur plusieurs lignes, ce que l'on appelle des agrégats.

Ici on réalise une restriction sur un étudiant qui permet de sélectionner uniquement les enregistrements associés aux cours qu'il suit.


On compte ensuite le nombre d'enregistrements : on a donc bien le nombre de cours suivis par l'étudiant identifié.

Exercice

Quelle information renvoie l'instruction suivante :

```
1 SELECT student, COUNT (code)
2 FROM course
3 GROUP BY student;
```

- ☐ Le nombre d'étudiants.
- ☐ Le nombre de cours.
- ☒ Pour chaque étudiant, le nombre de cours qu'il suit.
- ☐ Pour chaque cours, le nombre d'étudiants le suivant.

 On groupe ici les enregistrements par la clef étrangère des étudiants.


On compte ensuite pour chaque clef étrangère le nombre d'enregistrements.

Exercice

Quelle est la requête qui retourne, pour chaque cours, le nombre d'étudiants inscrits ?

☒ `SELECT code, COUNT(student)
FROM course
GROUP BY code;`

☐ `SELECT code, COUNT(code)
FROM course
GROUP BY student;`

 On groupe ici les enregistrements par la clef étrangère des cours. On compte ensuite pour chaque clef étrangère le nombre d'enregistrements.