Contents

JOUR 3M

Via ce module tu pourras :

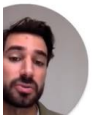
- Comprendre pourquoi SQL est utilisé
- Visualiser ce qu'est une base de données relationnelle
- Comprendre les contraintes liées au modèle relationnel
- Savoir ce que sont les clefs primaires et étrangères
- Rédiger tes premières requêtes en SQL
- Construire et modifier un schéma de base de données

Qu'est-ce que le Big Data ?

- Le *Big Data* est un ensemble de moyens d'extraction, d'analyse et de traitement d'immenses volumes de données (ou *data sets*) qui sont trop larges pour être traités avec les outils traditionnels d'analyse de données
- L'architecture Big Data repose sur plusieurs étapes : l'extraction de la donnée, le stockage de la donnée possible par plusieurs outils (datalake, data warehouse...), et enfin l'analyse, notamment via le SQL

Pourquoi avons-nous besoin de nouvelles infrastructures pour nos données ?

4 V



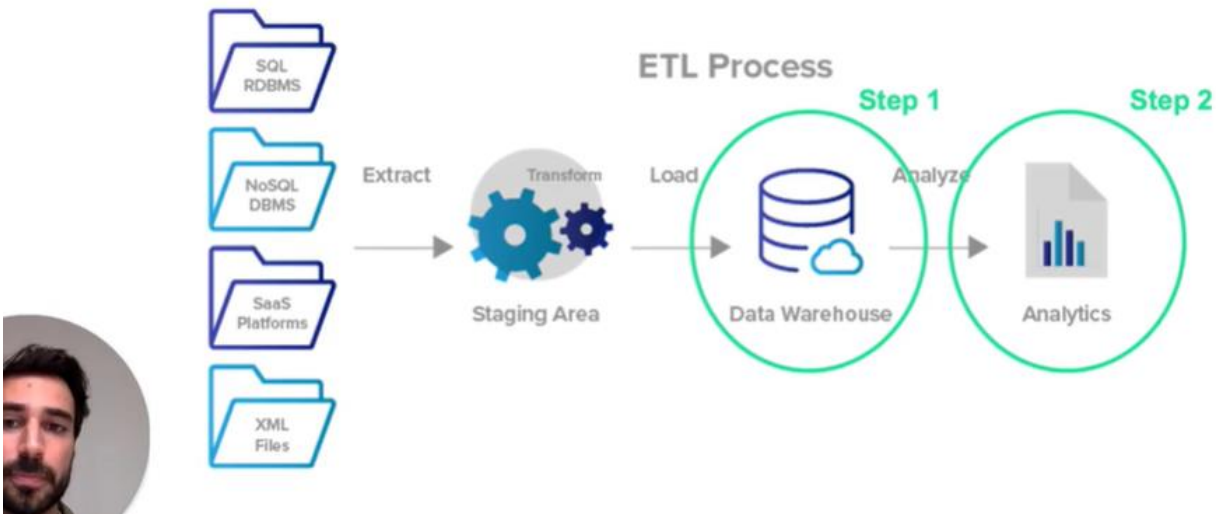
Volume

Velocity

Variety

Veracity

Coup d'oeil à l'architecture du Big Data



Comment stocker des informations sur ses clients ?

Comment concevoir la donnée ?

- Le stockage de données se fait sous forme de tables, composées de plusieurs colonnes (= attributs) et lignes (= données stockées)
- Chaque table doit disposer de sa clé primaire : il 'agit d'une ou plusieurs colonnes permettant d'identifier chaque ligne
- Chaque valeur de la clé primaire doit être unique

Que faire si deux clients se nomment Jean Dupond ? Comment les différencier ?

Nom	Prénom	Email	Adresse	Téléphone
Delarue	Daniel	Daniel.Delarue@gmail.com	Londres	709817786
Dupond	Jean	jean.dupond@gmail.com	Paris	687657763
Dupond	Jean	jean.dupond@hotmail.com	Lyon	687657645

Clé primaire - Contrainte d'unicité

ID_client	Nom	Prénom	Email	Adresse	Téléphone
1	Delarue	Daniel	Daniel.Delarue@gmail.com	Londres	709817786
2	Dupond	Jean	jean.dupont@gmail.com	Paris	687657763
3	Dupond	Jean	jean.dupont@hotmail.com	Lyon	687657645

Clé primaire - Contrainte d'unicité

Une clé primaire est un ensemble de colonnes au sein d'une table T telle que deux lignes distinctes de la table ne peuvent avoir les mêmes valeurs dans ces colonnes.
La valeur de la clé primaire est utilisée pour identifier une ligne dans T.

Ainsi une table relationnelle est définie par ses colonnes(ou attribut) et ses lignes uniques (une clé primaire et autant de valeurs que de colonnes).

Les types de variables en SQL

Il existe différents types de données que vous pouvez stocker dans votre table :

- Les entiers : INT
- Les nombres décimaux : FLOAT
- Les dates : DATE, DATETIME, TIMESTAMP
- Les données texte : CHAR, VARCHAR, TEXT
- Les données nulles : NULL

Les Types

clients	type
id_client	integer
nom	text/varchar(20)
prénom	varchar(20)
email	varchar(100)
adresse	varchar(100)
téléphone	integer

Quelles type de variables allons nous stocker ?

Renseigner une valeur par défaut et auto-incrémentation

commandes	Type et option
id_commande	Integer NOT NULL AUTO_INCREMENT
id_client	Integer NOT NULL
date	Date DEFAULT "1900-01-01"

Nous n'avons pas la date, mais on veut quand même une valeur par défaut.

Pourquoi et comment lier des tableaux de données ?

Une base de données est dite relationnelle, car les différentes tables qui la composent peuvent être reliées les unes avec les autres. Cela est possible grâce aux clés :

- Les clés primaires, qui permettent d'identifier chacune des lignes d'une table

- Les clés étrangères, qui référencent les clés primaires d'autres tables

Par rapport aux tableurs classiques, une base de données relationnelle que l'on peut visualiser avec SQL a deux avantages :

- Éviter les redondances : les informations sont regroupées sous formes de tables thématiques
- Recouper les informations : SQL permet de visualiser des informations qui peuvent être stockées dans plusieurs tables différentes

id_client	id_commande	nom_produit	quantité
1	12	Brosse à dent	1
1	34	Dentifrice	25

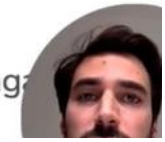
Id_command est un clé primer et id_clien est une clé étranger.

Le modèle de base de données relationnelle

Si vous restez sur tableur, il vous sera impossible de lier facilement vos tables pour croiser l'information (VLOOKUP ou INDEX/MATCH ne sont pas construites pour ça)

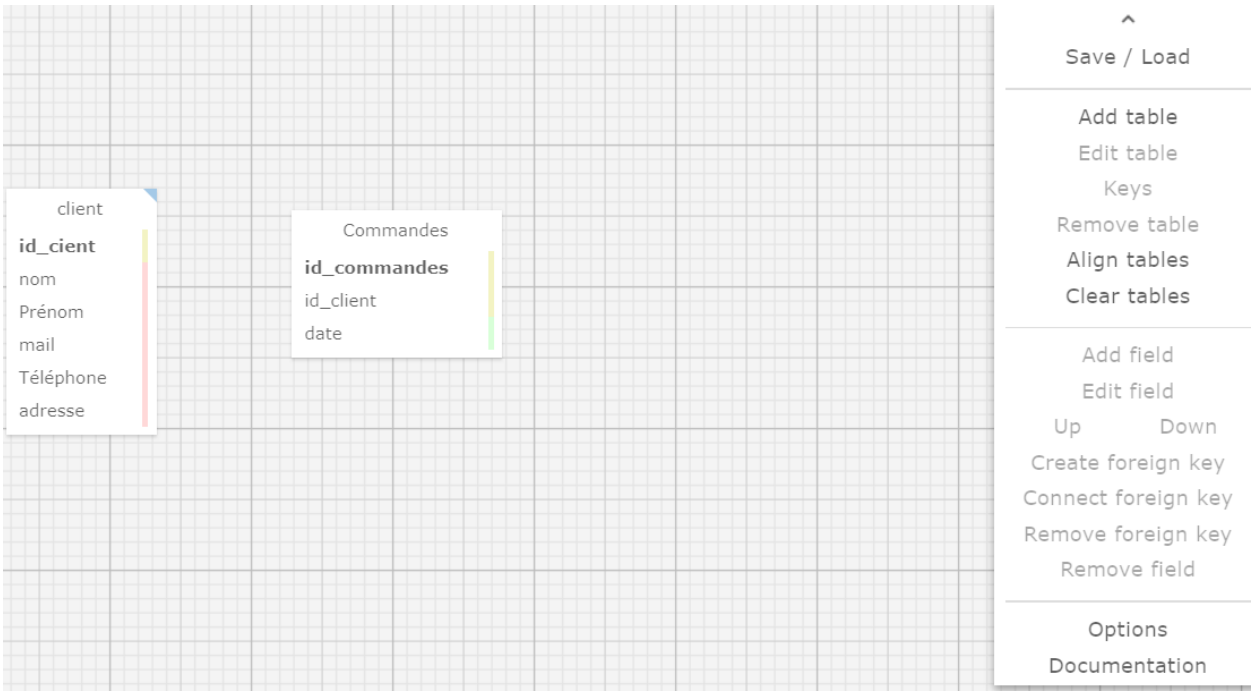
→ Avec une base de données relationnelle, le bénéfice principal est de produire des informations pertinentes en joignant les tables **avec une simple et unique requête.**
→ Un autre point positif est l'absence de **redondance des données.** (L'élimination des données redondantes fait partie des pratiques de normalisation de données).

Pour dialoguer avec une base de données relationnelle, on utilise le lang



Construire une table selon un modèle de base de données relationnelles

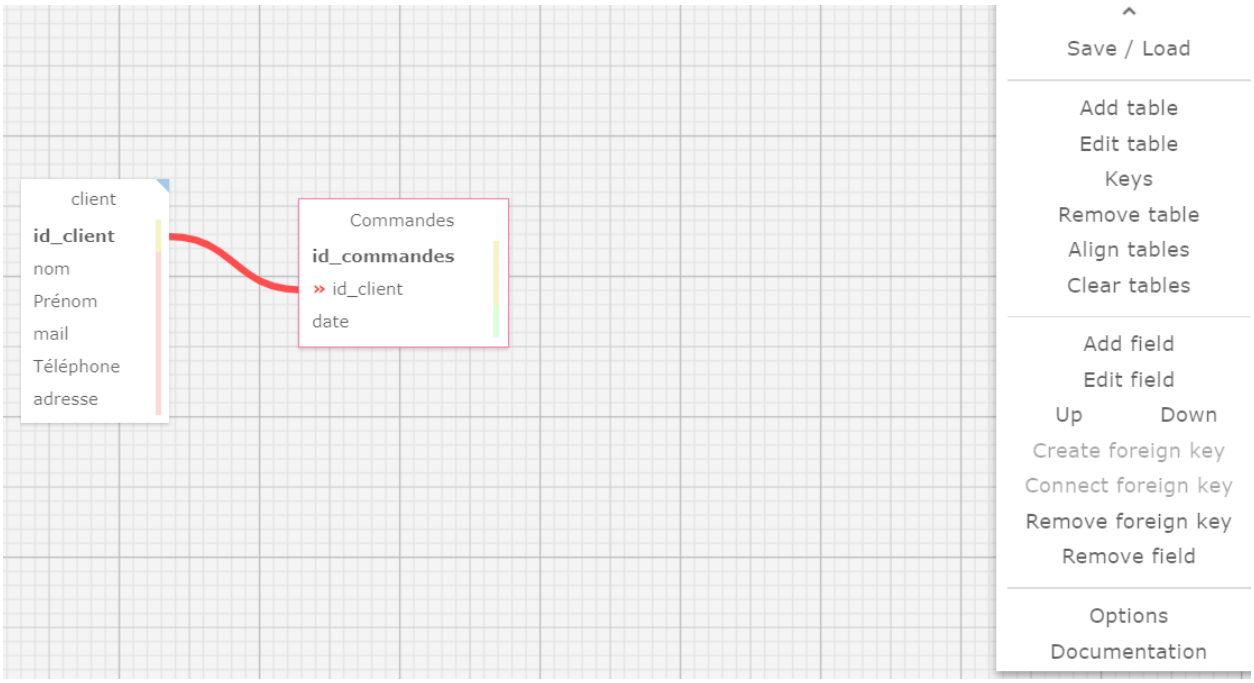
https://sql.toad.cz/?

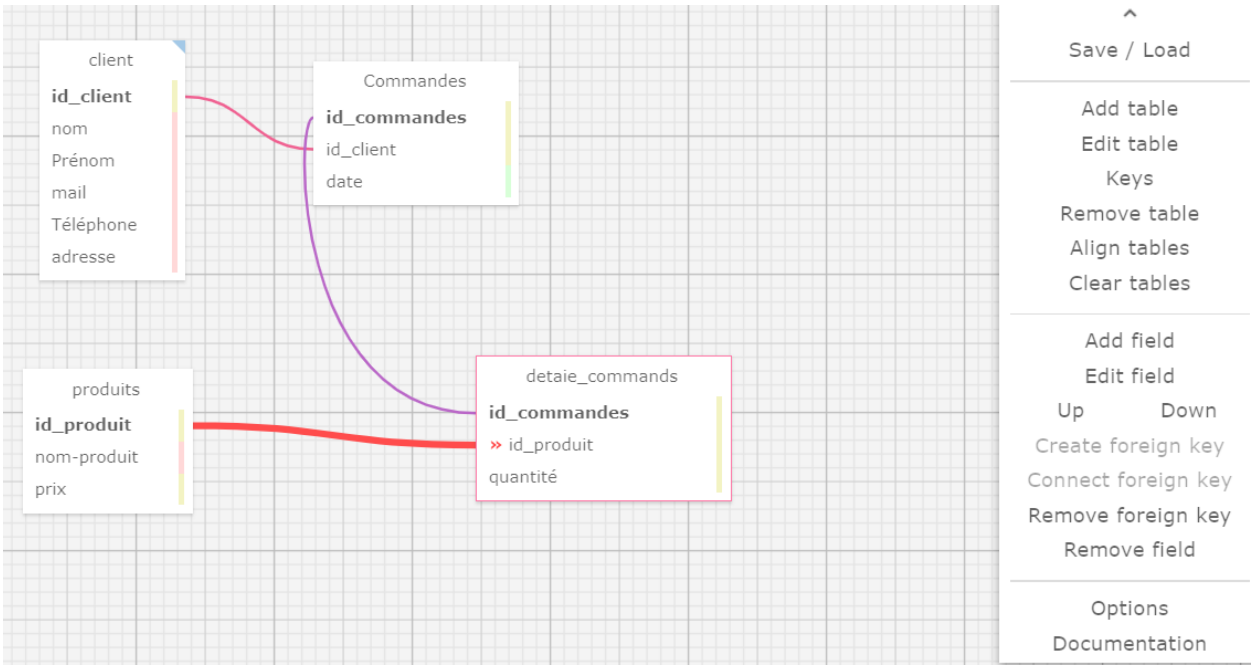


Les différentes relations entre les tables

Il existe plusieurs relations entre les tables :

- Relations *one-to-many*. Exemple : un client peut commander plusieurs fois. La table client et la table commandes ont donc une relation one-to-many.
- Relation *many-to-many*. Exemple : un produit peut se retrouver dans plusieurs commandes, et une commande peut contenir plusieurs produits. La table commandes et a table produits ont donc une relation many-to-many
- Les tables ayant une relation *many-to-many* doivent être reliées par une table intermédiaire, autrement l'unicité de la clé primaire n'est plus respectée

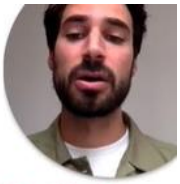

client
id_client
nom
Prénom
mail
Téléphone
adresse
Commandes
id_commandes
» id_client
date
Save / Load
Add table
Edit table
Keys
Remove table
Align tables
Clear tables
Add field
Edit field
Up
Down
Create foreign key
Connect foreign key
Remove foreign key
Remove field
Options
Documentation


client
id_client
nom
Prénom
mail
Téléphone
adresse
Commandes
id_commandes
id_client
date
produits
id_produit
nom-produit
prix
detaie_commands
id_commandes
» id_produit
quantité
Save / Load
Add table
Edit table
Keys
Remove table
Align tables
Clear tables
Add field
Edit field
Up
Down
Create foreign key
Connect foreign key
Remove foreign key
Remove field
Options
Documentation

Garantir l'intégrité des données

- Le fait d'avoir des clés primaires et étrangères dans plusieurs tables imposent une intégrité référentielle. Autrement dit, si on modifie des données d'une table, il faut que cette modification se répercute dans toutes les autres tables qui font référence aux données modifiées.
- Plusieurs cas possibles : Restrict, Set Default, Set Null, Cascade
- Lors de la création du schéma de la base de données, on peut imposer certaines contraintes : implicites, explicites et sémanti

Intégrité référentielle (Referential integrity actions)



Comment se comporte l'attribut id_client dans la table 'commandes' si l'on modifie l''id_client' dans la table clients ?

- Restrict : Produit une erreur et empêche la modification si on essaie d'altérer une clé primaire qui est utilisée ailleurs. C'est l'action par défaut en intégrité référentielle.
- Set Null : On applique la valeur NULL à toutes les clés étrangères qui font référence à la clé primaire modifiée.
- Set Default : On applique la valeur par défaut à toutes les clés étrangères qui font référence à la clé primaire modifiée. (la valeur par défaut est spécifiée lors de la définition de la table)
- Cascade : utilisée avec l'option ON UPDATE, toutes les clés étrangères sont mise à jour conformément à la nouvelle valeur de la clé primaire. Utilisée avec l'option ON DELETE, toutes les lignes qui contiennent une clé étrangère liée à la clé primaire modifiée sont supprimées.

Intégrité référentielle (Referential integrity actions)

clients
id_client
nom
prénom
email
adresse
téléphone

1 - N

commandes
id_commande
id_client
nom_produit
quantité

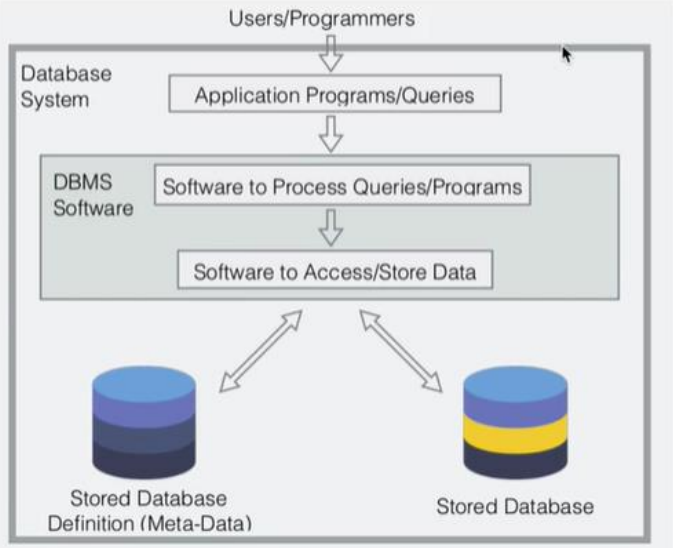
Comment se comporte l'attribut id_client dans la table 'commandes' si l'on modifie l''id_client' dans la table clients ?

Récapitulatif des contraintes que l'on peut imposer modèle de données

- **Contraintes implicites** : Inhérent au modèle de données
  - Exemples : pas de lignes redondantes
- **Contraintes explicites** : elles sont exprimées lors de la construction du modèle de données (contraintes d'intégrité)
  - Exemple: contrainte de domaines (les prix enregistrés doivent être supérieur à 0), clés primaires, valeurs NULL, clés étrangères
- **Contraintes sémantiques** (les règles "Business"): Ne peuvent être explicitement définies dans le schéma.
  - Exemple: Le montant de la livraison doit être inférieur au prix de vente

Interagir avec les bases de données

Comment est construit un modèle de données ?



Comment interagir avec une base de données ?

- Il faut pour cela utiliser un DBMS (*Data Base Management System*) ou SGBD en français (Système de Gestion de Bases de Données) : c'est le logiciel qui va récupérer les informations contenues dans la base de données
- Pour communiquer avec la BDD par l'intermédiaire du DBMS, il faut rédiger une requête écrite en SQL
- Le SGBD va ensuite afficher le résultat de la requête sur son interface

DBMS – DataBase Management Systems

Comme les projets gratuits Apache et Linux, PostgreSQL n'est pas controlée par une organisation unique, mais repose sur une communauté globale de développeurs et d'entreprises.

MariaDB — Dérivé de MySQL

SQLite — Base de données open-source

ORACLE MySQL — Propriété d'Oracle

SQL Server 2012 — Propriété Microsoft

Les services clouds ont aussi leur propre DBMS

Google BigQuery

amazon REDSHIFT

Azure

On utilise le langage SQL pour dialoguer avec le DBMS

DDL

DDL ou "Data Definition Language". C'est la langage pour gérer le schéma et les descriptions de la base de données

DML

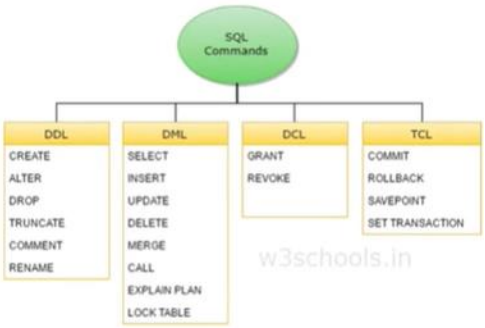
DML ou "Data Manipulation Language". C'est le langage utiliser pour stocker, modifier, récupérer, supprimer et mettre à jour des données dans la base de données.

DCL

DCL ou "Data Control Language". Il comprend des commandes comme GRANT, qui concerne surtout les droits et permissions du système de base de données.

TCL

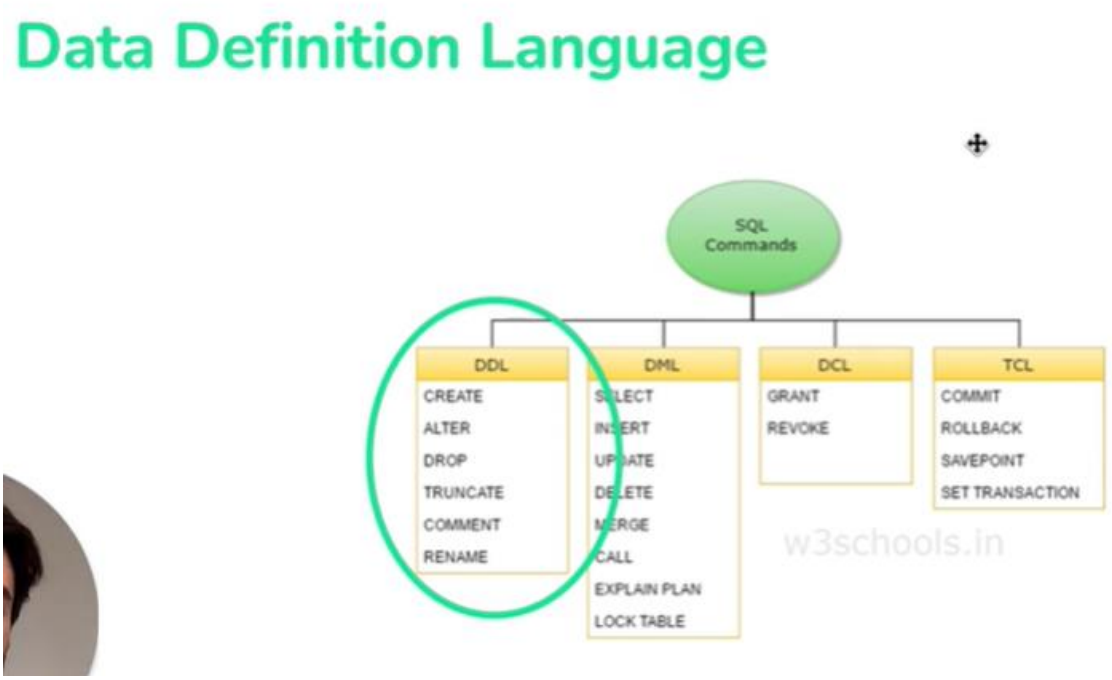
TCL ou Transaction Control Language. Il est utilisé pour gérer les modification des données dans une table suite à des requêtes DML



Sql est un langage qui nous permet d'interagir avec le DBMS SQL Lite

JOURS 3A DDL

Les outils pour coder en SQL



Trouve ici le lien pour télécharger et utiliser les deux outils mentionnés dans la vidéo !

Lien vers l'outil pour écrire des requêtes SQL directement dans ton navigateur :

SQL Online

Lien pour télécharger Sublime Text :

Sublime Text

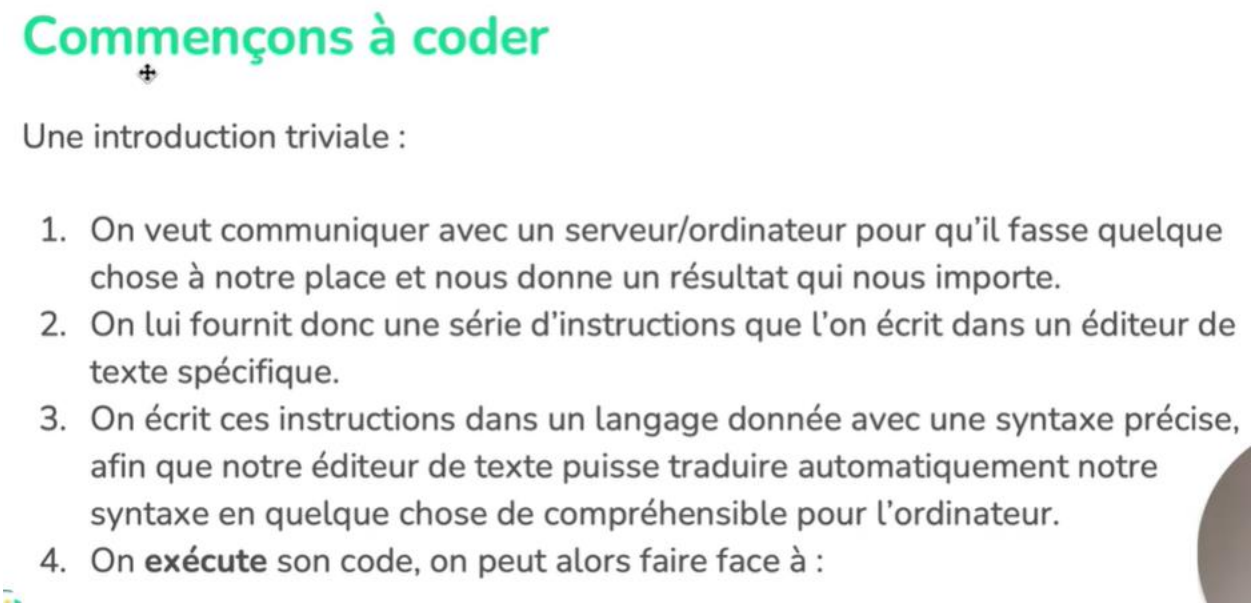
2 points importants sur SQL Online :

1. Attention à bien coder sur MariaDB dans SQL Online (équivalent MySQL open source)
2. Vous pouvez aussi, pour chaque nouvelle requête appuyer sur le "+" pour ouvrir un nouveau script (et ainsi éviter de perdre voter code précédemment écrit)

Si vous voulez un benchmark des outils utilisés pour coder en SQL, rendez vous sur : https://www.castordoc.com/blog/sql-editor-benchmark-and-market-analysis

Premières requêtes SQL - Créer une table - Partie 1

- SQL est un langage pensé et optimisé pour gérer les bases de données
- Sa syntaxe est très proche de l'anglais, ce qui en fait un langage facile à comprendre et à apprendre pour les débutants
- Pour écrire votre code, vous utiliserez SQL Online, qui agira comme un SGBD : vous pourrez rédiger vos requêtes, et afficher le résultat



Votre code dans un éditeur de texte : (notez que l'éditeur colore les mots clés de votre langage)

```
Entrée [3]: import numpy
            age_amis = [21, 22, 23]
            age_moyen = numpy.mean(age_amis)
            print("Age moyen : " + str(age_moyen)) #problème sur le type
```

Le(s) résultat(s) :

Ce qu'on voulait obtenir : **Age moyen : 22.0**

Quelque chose de surprenant : **Age moyen : zftb0X32**

Très souvent : un rapport de bug

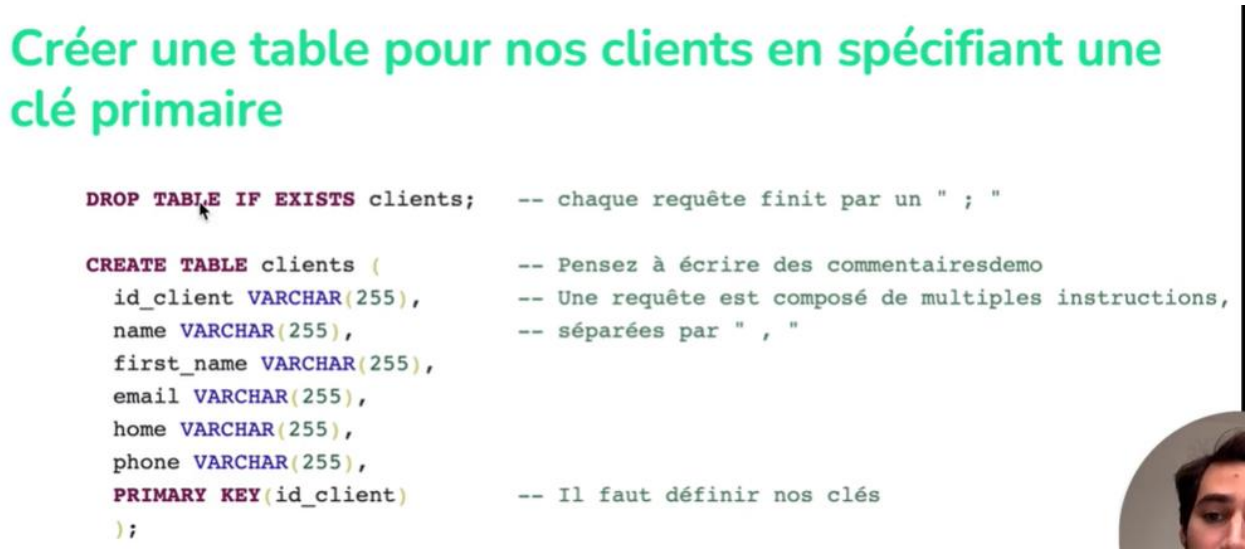
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
<ipython-input-2-abaldb76f2d9> in <module>
      2 age_amis = [21, 22, 23]
      3 age_moyen = "numpy.mean(age_amis)"
----> 4 print("Age moyen : " + str(ae_moyen)) #problème sur le type

NameError: name 'ae_moyen' is not defined
```

Live code

Dans ce cours nous allons donc écrire en langage SQL : un langage pensé, créé et optimisé pour gérer les bases de données.

Notre outil éditeur de texte (et bien plus) est SQL online avec lequel on peut :
-Ecrire notre code en éditeur de texte
-l'exécuter
-Visualisez le résultat de nos requêtes et lire les rapports de bug
-Beaucoup d'autres outils

Écrivons maintenant nos premières lignes de code !



Premières requêtes SQL - Créer une table - Partie 2

Attention à bien selectionner **MariaDB** dans SQLiteonline (dans le menu à gauche du site). Si tu codes sur un autre DBMS SQL, ça ne fonctionnera pas.

Pour créer une table en SQL à partir de 0 :

- Par précaution, rédiger DROP TABLE IF EXISTS si jamais une table du même nom existe déjà
- Vous devez utiliser la commande CREATE TABLE
- Au sein de la requête, vous devez spécifier le nom des colonnes, le type de variables associées, identifier les clés primaire et étrangère, et spécifier le type d'intégrité référentielle

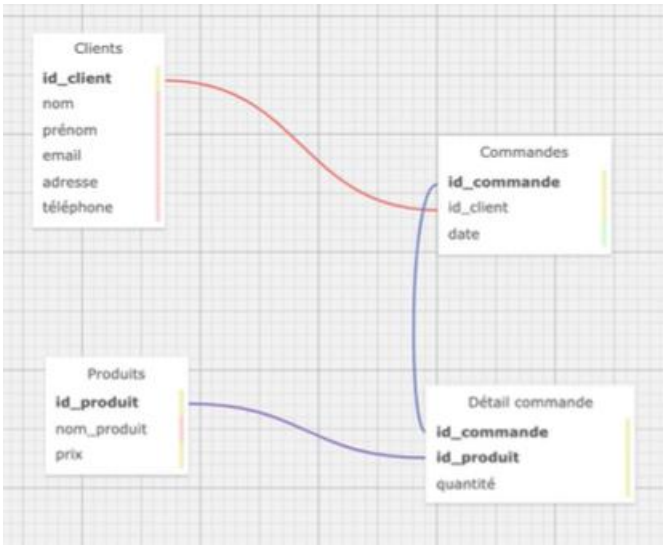
Trouve le fichier avec le code ci-dessous si tu ne l'as pas encore téléchargé !

cours_DDL.sql

Conclusion - construction de schémas

- ONDRAS permet de générer automatiquement le code associé à la création de votre schéma de données.
- Il faut pour cela aller dans SAVE / LOAD > GENERATE SQL (MYSQL)

- Attention, le code est en MySQL, il faut donc choisir la rubrique MariaDB (un dérivé de MySQL)



Modifier les données de sa table manuellement

Modifier le schéma de la table

ALTER TABLE ADD / DROP / ALTER COLUMN

Modifier le schéma de la table. Attention aux contraintes spécifiées

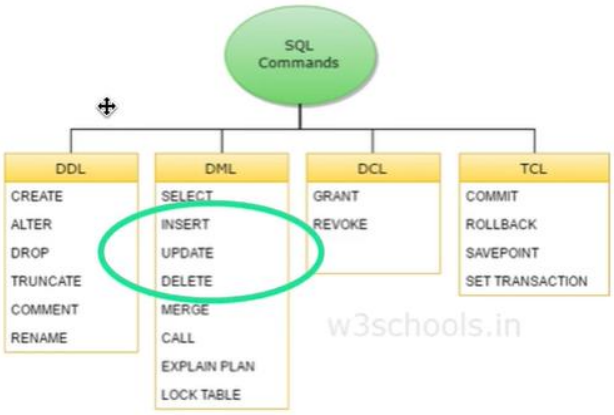
Ajouter et/ou modifier un attribut de la table client

```
ALTER TABLE orders ADD FOREIGN KEY(id_client) REFERENCES clients(id_client);

ALTER TABLE clients MODIFY COLUMN first_name CHAR(100);

ALTER TABLE clients ADD tel CHAR(100);

ALTER TABLE clients DROP tel;
```

Altérer ses données

Manipulation de data :

Data Manipulation Language : manipuler des données brutes



Insert / Modify / Delete data

INSERT INTO / UPDATE / DELETE

Pour interagir avec les données des tables
Attention aux contraintes

Entrer des données dans la table client

```
INSERT INTO clients VALUES ('B062',
                            'GOFFIN',
                            'Raphael',
                            'raphael.goffin@gmail.com',
                            '72 r. de la Gare',
                            '0632432325');
```

Lorsque nos données sont sous format CSV ?

→ On n'ajoute pas les lignes manuellement

- Selon les outils utilisés, il faut d'abord créer la structure de notre base de données, avec toutes les contraintes, puis importer les données.

- Il est possible de se passer de la création du schéma, et d'importer le fichier CSV tel quel, mais on se passe alors de la définition des contraintes, il faut donc faire attention aux options appliquées par notre outil.

Mettre à jour une donnée – Le client B062 a changé de nom

```
UPDATE clients SET name = 'TARTAMPION'
WHERE id_client = 'B062'
```

```
DELETE FROM clients WHERE name = 'TARTAMPION';
```

Les bonnes pratiques créer nos tables et colonnes

En général :

- S'assurer que le nom est unique et ne correspond pas à un mot clé réservé par le langage (order)
- Les noms doivent commencer par une lettre et ne pas terminer avec un underscore.
- N'utiliser que des lettres, chiffres et underscore dans les noms.
- Remplacer les espaces par des underscore (first name devient first_name).
- Éviter les abréviations et s'assurer de la bonne compréhension des noms.
- Utiliser des formes plurales pour les noms. Par exemple préférer personnel à employés et employés à employé.

Pour les colonnes:

- Quand c'est possible ne pas utiliser simple id comme nom de clé primaire.
- Ne pas ajouter de colonne homonyme à une table et vice-versa.
- Toujours écrire en lettre minuscule sauf quand cela se justifie.

Récap - Relations entre les tables

Relation entre les tables : one to many

clients
id_client
nom
prénom
email
adresse
téléphone

1 - N

commandes
id_commande
id_client
date

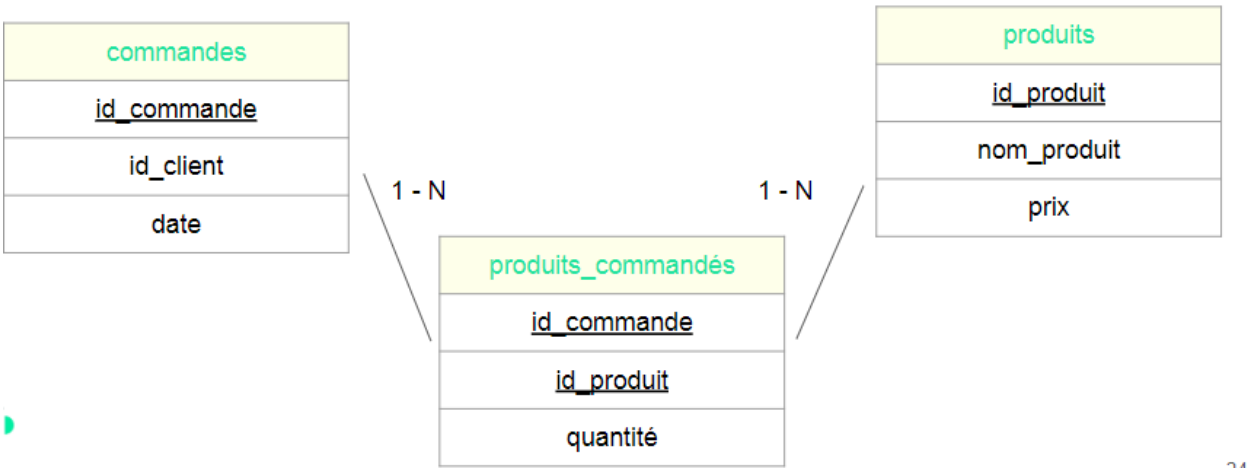
Relation entre les tables : Many – to – Many

commandes
id_commande
produits ?
id_client
date

N - N

produits
id_produit
nom_produit
prix

Relation entre les tables :
Passer de many-to-many à one-to-many



JOUR4M SFW

☞ Objectifs

Via ce module tu pourras :

- Appeler une table d'une base de données, et sélectionner les colonnes que tu veux
- Filtrer une table avec les conditions que tu souhaites appliquer
- Ordonner tes résultats par ordre croissant ou décroissant

Présentation de la base de données du chapitre

Les données que vous allez utiliser sont issues d'un site e-commerce spécialisé dans les brosses à dents nouvelle génération. Le schéma contient 4 tables :

- CLIENT : toutes les informations relatives aux client. Solde = compte client (ex : si le solde est à 100€, la personne dispose d'un avoir de 100€)

- COMMANDE : qui a commandé et quand

- DETAIL : qu'est-ce qui a été commandé et en quelle quantité

- PRODUIT : liste des produits proposés sur le site e-commerce, et le prix associé

Télécharge les fichiers dont tu as besoin

Tu trouveras ici deux fichiers .SQL à télécharger pour le chapitre :

Le premier BBD_brush.sql contient le code nécessaire à la création de la base de données qu'on vient de présenter, afin que tu puisses l'ajouter toi même à SQL online pour faire tes requêtes.

Le deuxième livecode_DML.SQL contient toutes les requêtes que l'instructeur va écrire dans ce chapitre, afin que tu puisses y avoir accès facilement pour les tester en même temps que l'instructeur, les modifier etc..

- **livecode_DML.sql**2,1 KO

TÉLÉCHARGEMENTS'OUVRE DANS UNE NOUVELLE FENÊTRE

- **bdd_brush.sql**4,1 KO

TÉLÉCHARGEMENTS'OUVRE DANS UNE NOUVELLE FENÊTRE

La base des requêtes SQL - SELECT

Revoilà le fichier bdd_bbrush.SQL avec le code nécessaire à la création de la base de données si tu ne l'as pas téléchargé précédemment !

Pour uploader les données sur SQLite :

- Cliquer sur MariaDB, puis click to connect
- DROP la table demo
- Copier / coller l'intégralité du code contenu dans le fichier .sql et exécuter la requête

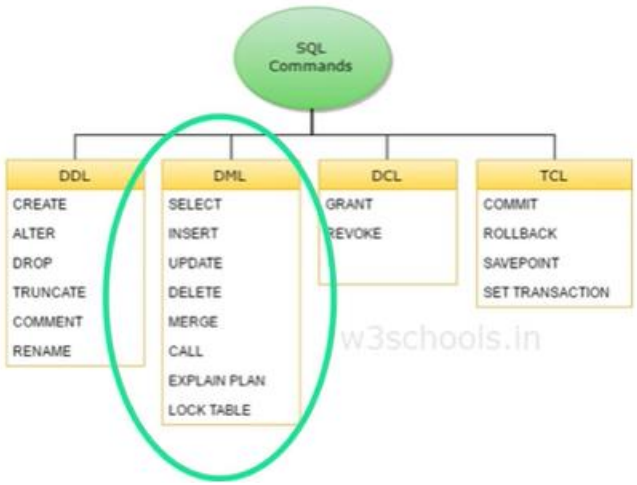
SELECT et FROM sont les deux mots-clés permettant d'appeler une table dans un schéma de BDD :

- SELECT permet de choisir les colonnes que l'on veut afficher
- FROM permet de sélectionner la table correspondante
- SELECT DISTINCT permet de sélectionner une colonne sans les doublons

- **bdd_brush.sql**4,1 KO

TÉLÉCHARGEMENTS'OUVRE DANS UNE NOUVELLE FENÊTRE

Le langage de manipulation de données



Requêter la base de données

SELECT / FROM / WHERE

Accès aux données dans une certaine table et respectant certaines contraintes souhaitées.

SQL Online

bdd_brush-01

```
SELECT *
FROM COMMANDE
LIMIT 1;
```

to see just sirst ine

Filtrer son select avec WHERE - partie I

WHERE permet de rajouter des conditions à la requête, et donc de filtrer les résultats affichés. Exemple :

- SELECT nom FROM CLIENT WHERE localite = "Toulouse";

```
SELECT nom
from CLIENT
WHERE localite ='Toulouse'
```

- SELECT id_commande FROM DETAIL WHERE qcom > 20 AND qcom < 100;

```
SELECT id_commande
from DETAIL
WHERE qcom>20 and qcom<100;
```

WHERE permet aussi d'identifier les valeurs nulles (une valeur qui n'a pas été renseignée).
Exemple :

SELECT id_produit FROM PRODUIT WHERE qstock IS NULL;

```
SELECT id_produit
from PRODUIT
WHERE qstock is NULL;
```

IN permet de se substituer au OR dans certains cas. Exemple :

- WHERE localite IN ('Toulouse', 'Lyon')
- au lieu de WHERE localite = 'Toulouse' OR localite = 'Lyon'

```
SELECT nom
from CLIENT
WHERE localite IN('toulouse','lyon');
```

EXC.

```
SELECT nom
from CLIENT
WHERE localite IN('Toulouse','Lile','Bruxelles');
```

Filtrer sa sélection avec WHERE - partie II

BETWEEN permet de se substituer à AND dans certains cas. Exemple :

- WHERE qcom BEWTEEN 20 AND 100
- Au lieu de : WHERE qcom > 20 AND qcom < 100

```
SELECT id_commande
from DETAIL
WHERE qcom BETWEEN 20 AND 100;
```

Le mot-clé LIKE, avec un REGEX permet de rechercher une chaîne de caractère spécifique.
Exemple :

- WHERE description_produit LIKE '%brosse%'
- Cela signifie : toutes les produits dont la description contient le mot "brosse"

```sql
SELECT id_produit, description_produit
FROM PRODUIT
WHERE description_produit LIKE '%brosse%';
```

id_produit	description_produit
CS262	Brosse_dent_2018
CS264	Brosse_dent_2019
CS464	Brosse_dent_2020

EXC :

```sql
SELECT id_produit, description_produit
FROM PRODUIT
WHERE description_produit LIKE '%ubrush%';
```

id_produit	description_produit
PH222	UBrush_enfant
PS222	UBrush_adulte

Ordonner sa sélection avec ORDER BY

Pour sélectionner plusieurs colonnes, tu peux donner un nom à chaque colonne que tu souhaites nommer, en séparant par des virgules

ORDER BY permet de trier les résultats par ordre croissant ou décroissant :

- Par défaut, ORDER BY trie par ordre croissant

```
1 SELECT *
2 FROM DETAIL
3 ORDER BY qcom;
4
```

id_commande	id_produit	QCOM
30186	PA45	3
30189	CS464	10
30185	PA60	15
30179	PA60	20
30184	PA45	20

- ORDER BY qcom : trier les quantités commandées par ordre croissant

```
1 SELECT *
2 FROM DETAIL
3 ORDER BY qcom DESC;
4
```

id_commande	id_produit	QCOM
30185	CS464	260
30188	CS464	180
30184	CS464	120
30188	PH222	92
30188	PA60	70

- ORDER BY qcom DESC : trier les quantités commandées par ordre décroissant
- On peut ORDER BY plusieurs colonnes. Comme dans le SELECT, les noms de colonnes doivent être séparés par des virgules :
- ORDER BY nom, solde DESC

```sql
SELECT id_client, nom , solde
FROM CLIENT
ORDER BY nom, solde DESC;
```

id_client	nom	solde
C003	AVRON	-1700.00
C400	FERARD	350.00
L422	FRANCK	0.00
B512	GILLET	-8700.00
B062	GOFFIN	-3200.00
S712	GUILLAUME	0.00

```sql
SELECT id_client, nom , solde
FROM CLIENT
ORDER BY 2, 3 DESC;
```

id_client	nom	solde
C003	AVRON	-1700.00
C400	FERARD	350.00
L422	FRANCK	0.00
B512	GILLET	-8700.00
B062	GOFFIN	-3200.00
S712	GUILLAUME	0.00

```sql
SELECT id_client, nom , solde
FROM CLIENT
ORDER BY solde, nom DESC;
```

id_client	nom	solde
B512	GILLET	-8700.00
S127	VANDERKA	-4580.00
B062	GOFFIN	-3200.00
C123	MERCIER	-2300.00
D063	MERCIER	-2250.00
C003	AVRON	-1700.00

PDF : select_from_where-

JOUR 4A Fonctions d'agrégation

Les fonctions d'agrégation et le GROUP BY

Les fonctions d'agrégation permettent d'effectuer des calculs sur votre table, et de grouper les résultats.

5 fonctions d'agrégation :

- COUNT()
- SUM()
- AVG()
- MAX()
- MIN()

MariaDB

```
1 SELECT COUNT(*)
2 FROM CLIENT;
3
4
```

COUNT(*)
16

Une fonction d'agrégation s'accompagne dans la plupart des cas d'un GROUP BY. Ex : calculer le nombre de clients par ville :

SELECT localite, COUNT(*) AS nb_clients
FROM CLIENT
GROUP BY localite

```
1 SELECT localite , COUNT(*)
2 FROM CLIENT
3 GROUP BY localite
4 ORDER BY COUNT(*) DESC;
```

localite	COUNT(*)
Toulouse	5
Lyon	4
Poitiers	3
Bruxelles	1
Lille	1
Geneve	1

```sql
SELECT localite , COUNT(*)  AS 'TOTAL_client'
FROM CLIENT
GROUP BY localite
ORDER BY COUNT(*) DESC;
```

localite	TOTAL_client
Toulouse	5
Lyon	4
Poitiers	3
Bruxelles	1
Lille	1
Geneve	1

```sql
SELECT localite , SUM(solde) AS 'Total_solde'
FROM CLIENT
GROUP BY localite
ORDER BY Total_solde;
```

localite	Total_solde
Toulouse	-12650.00
Lyon	-10080.00
Geneve	0.00
Paris	0.00
Bruxelles	0.00
Lille	720.00
Poitiers	1600.00

EXC :

Filtrer ses Group By avec HAVING

- Vous pouvez filtrer les résultats de votre requête GROUP BY en utilisant la clause HAVING
- Contrairement au WHERE, elle doit se place en dessous du GROUP BY dans votre requête

```
SELECT localite, COUNT(*) AS total
FROM CLIENT
GROUP BY localite
HAVING total BETWEEN 3 AND 5;
```

Bonus Slides: Récapitulatif sur Order By - Group By et fonctions d'agrégation

PDF : RecapGroupby

Bonus slides: Recap clause HAVING

PDF : RecapHaving