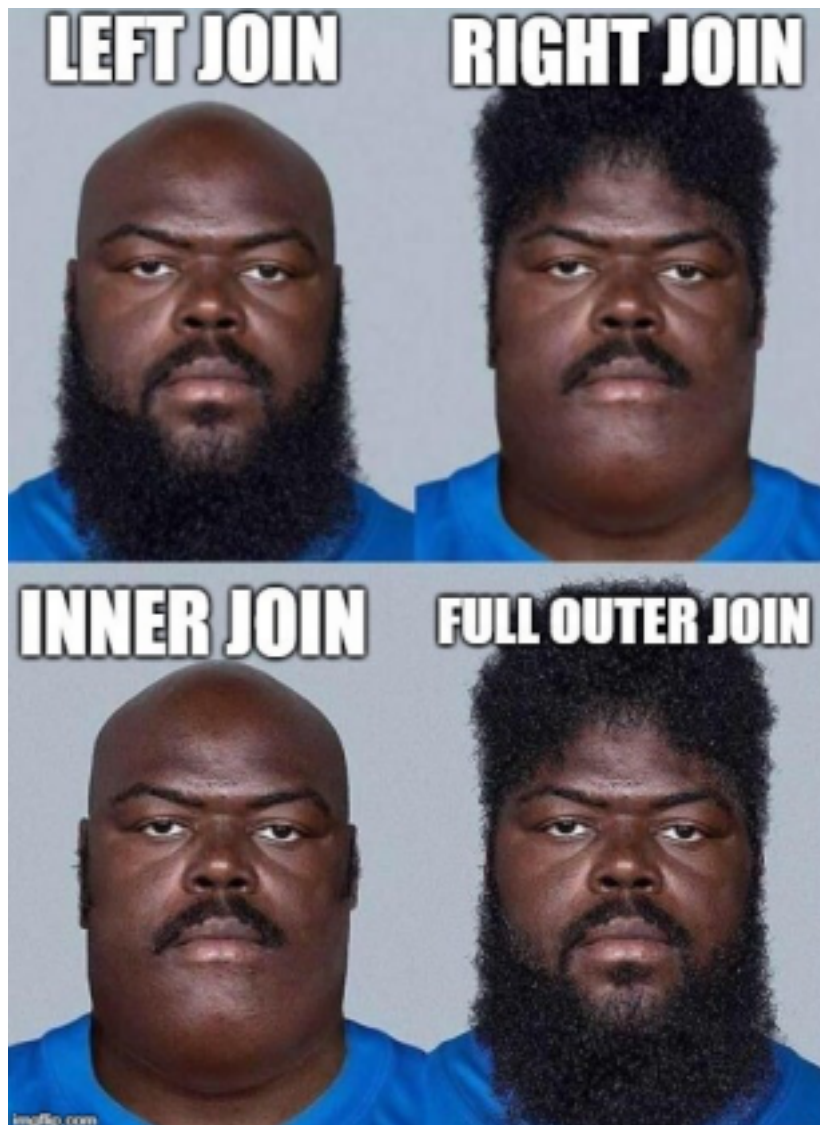




Base de Données (BDD)



TYPE OF SQL JOINTS (SOURCE : [HTTPS://WWW.REDDIT.COM/R/DATABASE/](https://www.reddit.com/r/DATABASE/))

Objectifs du module

Maîtriser les concepts et l'utilisation d'un SGBD

Le stockage de l'information fait partie des challenges actuels et historiques de l'informatique. Plusieurs générations de chercheurs en informatique ont travaillé - et travaillent encore - à modéliser et optimiser les systèmes de stockage, selon différents types de contraintes.

Dans l'histoire de l'informatique, le besoin d'un système de stockage performant et présentant les caractéristiques suivantes s'est fait ressentir très tôt :

- **Structuré** : Un schéma clair et précis des chemins d'accès aux données est réalisable;
- **Massif** : Un grand nombre de données peut être géré. Aujourd'hui, on parle de téraoctets/jour;
- **Persistant** : Il reste stable et pérenne, *a contrario* des programmes informatiques dont l'état est perdu après la fermeture de l'application;
- **Sécurisé** : Il présente des sécurités contre les failles tant au niveau logiciel (software) que physique (hardware) --> sauvegardes fréquentes, ...;
- **Concurrent** : La même base de données reste accessible à plusieurs utilisateurs en même temps sans risque pour son intégrité;
- **Efficient** : d'abord la performance, puis la performance et finalement la performance - bons résultats avec un minimum d'efforts;
- **Fiable** : Il est opérationnel 99.999999999.....% du temps. Avec autant de 9, après la virgule, que nécessaire.

Les systèmes RDBMS (*Relational Database Management Systems*) ou SGBD en français ont permis de répondre à ces besoins. Ils sont issus du modèle de données relationnel décrit par Edgar Codd en 1969.

Pourquoi apprendre les SGBDR ?

Méthode de stockage privilégiée dans le monde industriel

Il est à 90% probable que les systèmes informatiques avec lesquels vous interagissez ou interagirez dans votre vie quotidienne utilisent au moins un SGBD pour sauvegarder l'information.

Transformation de l'informatique : d'une science de calcul vers une science de la donnée

Autrefois, la donnée n'avait qu'un but opérationnel précis : enregistrer une opération, afin de faire constater un événement/état. De nos jours, la donnée est plus qu'un répertoire d'octets, mais une opportunité d'extraction de connaissances.

Compétences

À la fin de ce module vous serez en mesure de :

- Comprendre une partie des généralités et la terminologie associées aux domaines de l'architecture et l'ingénierie Data
- Exploiter un SGBD à l'aide du langage de programmation SQL pour :
 - Définir un schéma relationnel (*Data Definition Language*)
 - Manipuler la donnée (*Data Manipulation Language*)
- Vous connecter à un SGBD :
 - À l'aide d'un client GUI
 - À l'aide du langage de programmation python
- Maîtriser les diagrammes de classes du langage de modélisation unifié (UML) pour:
 - Comprendre l'architecture d'une BDD relationnelle
 - Concevoir une BDD relationnelle
- (*Optionnel*) Avoir des notions de normalisation :
 - o 1NF, 2NF, 3NF
 - o 4NF

Organisation



Itération #1 (½ jour PM)

- Travail en équipe, par îlot, réflexion sur les mots :
 - o Données
 - o Modèle de données
 - o Structuration de données
 - o Base de données
 - o SQL / noSQL

Itération #2 (1 jour)

- Synthèse des notions évoquées lors de l'itération 1
- En autonomie : Manipulation des données en SQL (site web sqlzoo)

Itération #3 (1 jour)

- AM: S'interfacer avec une BDD
- PM: Travail en équipe, par îlot
 -   Pour le jour 3, pensez à prendre des écouteurs et des crayons!
 - Lecture d'un diagramme UML
 - Complétion d'un diagramme UML

Itération #4 (1 jour)

- Créer un script pour alimenter une BDD

Itération 1 ($\frac{1}{2}$ jour)

Introduction : BDD, SQL et noSQL

Objectif(s):

- Apprendre à utiliser le sous-ensemble du langage SQL pour la manipulation de données
- (Optionnel) : Avoir des notions d'algèbre relationnelle

Quelques éléments de langage pour débiter..à vous de compléter!

Le **modèle des données relationnel** peut se décrire comme suit :

- C'est un ensemble de **relations** (*i.e. des tables*)
- Chaque relation contient un ensemble d'**attributs** (*i.e. colonnes*)
- Une relation contient des **tuples** (*i.e. lignes*) avec des valeurs pour chaque attribut
- Chaque attribut est **typé**, et le type est souvent *atomique* (e.g. *entier, réel, etc.*)

Un **schéma** est la description structurelle d'une base relationnelle.

Une **clé** est un attribut (ou ensemble d'attributs) qui permet d'accéder à un ensemble de tuples.

Une **clé primaire** est un attribut (ou ensemble d'attributs) qui permet d'identifier un seul et unique tuple.

SGBD - Le *système de gestion de base de données relationnelle* est un système qui permet de stocker l'information sous un modèle de données relationnelles.

Parmi les SGBD très connus et utilisés se trouvent : Oracle, SQL Server, PostgreSQL, MySQL et SQLite. **SQL** - *Structured Query Language* est le langage qui permet d'exploiter un SGBD.

SQL est un langage:

- **Déclaratif** : Nous énonçons ce que l'on cherche et non pas le comment ; c'est le travail du SGBD de trouver l'algorithme pour donner le bon résultat de manière optimale. De ce fait, on dit que SQL est un langage de haut niveau.
- **Normalisé** : En théorie une requête doit pouvoir s'exécuter sur les différents SGBD. Dans la pratique, selon la complexité de la requête, elle sera modifiée pour s'ajuster à l'opinion du SGBD. Les changements sont, en règle générale, simples.
 - La norme ISO 9075 définit des fonctionnalités obligatoires et optionnelles du langage. Les SGBD implémentent ces fonctionnalités, *a minima* celles obligatoires, et ajoutent généralement leurs propres spécificités.

SQL est un langage issu de l'algèbre relationnelle.

Consignes

- Répondre aux questions suivantes:
 - Qu'est-ce qu'une donnée?
 - Comment définir une base de données?
 - Existe-t-il différents types de données? Si oui, lesquels connaissez-vous?
 - Pourquoi utiliser un SGBD?
 - Quelles sont les principales différences entre SQL et noSQL? Pourquoi choisir l'un plutôt que l'autre? Des exemples de cas réels ?
- Réaliser un tableau de synthèse des avantages et inconvénients de différents SGBD.
- Petit test de fin de journée: <https://librecours.net/module/bdd0/intro-sgbd/quiz.xhtml>

Quelques éléments de lecture

Si vous avez envie d'un peu de théorie, vous pouvez commencer à lire ou écouter les ressources suivantes : R1.1 et R1.2.

Références / Ressources

R1.1 Databases

<https://datascientest.com/base-de-donnees-definition>

[Database systems - Cornell University Course](#)

R1.2 Relational Algebra

<https://slideshowes.com/doc/159317/relational-algebra---university-of-toronto>

<https://www.youtube.com/watch?v=tii7xcFilOA>

<https://www.youtube.com/watch?v=GkBf2dZAES0>

Itération 2 (1 jour)

N'hésitez pas à revoir et implémenter les éléments de théorie investigués lors de la première itération avant de débiter.

Découverte de SQL - (1 jour)

Définition de SQL: ? *à remplir par l'apprenant!*

Consignes

- Rafraîchissement des notions découvertes lors de l'itération 1: Questionnaires via Plickers
- Direction <https://sqlzoo.net/> afin de réaliser les exercices 0 jusqu'à 9.
 - « 9 - Window function », et « 9+ Covid » ne sont pas obligatoires, mais si vous êtes curieux, foncez !

 *Pour éviter de perdre votre travail. Vous pouvez créer un compte avec votre e-mail Campus.*

Quelques éléments de lecture

SQL est un langage assez intuitif. Il est cependant possible que vous bloquiez au niveau des jointures. Dans ce cas, vous pouvez consulter la R2.1.

Références / Ressources

R2.1 SQL & Jointures – Université de Lyon
<https://sql.sh/ressources/cours-sql-sh-.pdf>
<https://sql.sh/2401-sql-join-infographie>

Ensuite à vous de trouver vos cheat-sheet.... ;-)

Itération 3 (2 jour)

A - S'interfacer à une BDD (AM)

Objectifs proposés :

Se servir d'une BDD à l'aide des différents outils :

- à l'aide d'une interface visuelle (GUI),
- à l'aide du langage Python et avec la librairie Pandas.
- (optionnel) quels outils pour quel besoin ?

Consignes

Pour cette itération, nous allons utiliser le logiciel SQLite. SQLite est un SGBD, sa qualité principale est la légèreté du système. Il est très probable que certaines des applications mobiles installées dans votre téléphone utilisent SQLite pour stocker l'information.

C1 - Installation de SQLite

C1.1 : Sur machine Linux

```
$ sudo apt-get install sqlite3  
$ sudo apt-get install sqlitebrowser
```

Ouvrir l'app Ubuntu Software Application > Chercher DB Browser for SQLite > Installer



sqlitebrowser
★★★★★

SQLite Database Browser is a visual tool used to create, design and edit database files compatible with SQLite. Its interface is based on QT, and is meant to be used for users an...

C1.2 : Sur machine Windows

Pour ceux avec une machine Windows, vous allez installer SQLite en même temps que «DB Browser for SQLite».

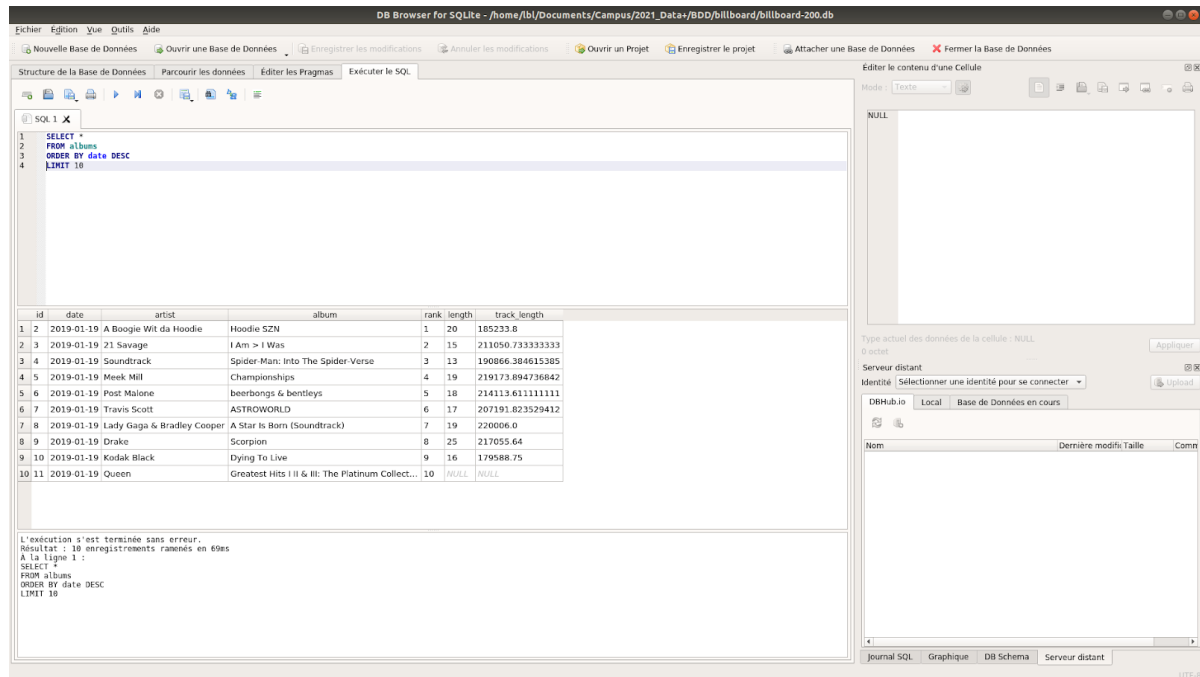
«DB Browser for SQLite» est un client visuel (i.e. GUI) pour les BDD de type SQLite. Un client visuel est un programme capable de se connecter à un SGBD, et de simplifier notamment l'étape de compréhension d'une BDD.

Pour l'installer dans une machine Windows :

1. Téléchargez [DB.Browser.for.SQLite-3.12.1-win64.zip](#)
2. Décompresser le fichier Zip
3. Cliquer sur l'exécutable DB Browser for SQLite

C2 - S'interfacer avec un GUI

1. Téléchargez les données qui se trouvent dans la ressource R3.1
2. Connectez-vous à la BDD à l'aide de votre client (i.e. DB Browser for SQLite)
3. Pour comprendre les données, référez-vous aux ressources R3.2 et R3.3
4. Trouvez le top 10 albums des 20 dernières années.



5. *Fact checking* . Vérifiez à l'aide de requêtes SQL les informations de la ressource R3.2.

Attention:

- Aux valeurs dans chaque colonne
- Aux duplicatas
- Aux erreurs de syntaxe dans les cases
 - a. Most weeks at Number One.
 - b. Top 10 albums of All-Time. Idem que le point 4, mais sans conditions sur les dates.
 - c. (Bonus) Tout autre qui vous semble intéressant !

⚠ Attention : Peut être que certains attributs doivent être modifiés afin que vos requêtes s'exécutent correctement.

Quelques pistes de réflexion question code (sans nettoyage initial):

DB Browser for SQLite - /home/lbl/Documents/Campus/2021_Data+/BDD/billboard/billboard-200.db

Structure de la Base de Données | Parcourir les données | Éditer les Pragma | Exécuter le SQL

```

1 SELECT *, count(date)
2 FROM albums
3 WHERE
4   (rank == 1)
5   AND
6   (date BETWEEN '1964*' AND '2015*')
7   GROUP BY album
8   ORDER BY count(date) DESC

```

id	date	Michael Jackson	Thriller	rank	length	track_length	count(date)
1	362802	1984-04-14	Thriller	1	30	282689.866666667	37
2	428002	1978-01-14	Fleetwood Mac	1	58	NULL	31
3	68602	2012-06-23	Adele	1	11	261895.454545455	25
4	423202	1978-07-01	Soundtrack	1	21	224416.142857143	24
5	355002	1985-01-12	Prince And The Revolution	1	9	NULL	24
6	294402	1990-11-03	M.C. Hammer	1	NULL	NULL	21
7	267602	1993-05-29	Whitney Houston	1	12	280579.166666667	20
8	279802	1992-03-28	Garth Brooks	1	NULL	NULL	18
9	538593	1967-06-10	The Monkees	1	30	150713.4	18
10	320402	1988-05-07	Soundtrack	1	13	203054.0	18
11	367002	1983-11-19	The Police	1	11	253214.818181818	17
12	274402	1992-10-03	Billy Ray Cyrus	1	10	215845.0	17
13	291202	1991-02-23	Vanilla Ice	1	15	231595.133333333	16
14	216002	1998-05-09	Soundtrack	1	57	NULL	16

L'exécution s'est terminée sans erreur.
Résultat : 989 enregistrements ramenés en 182ms
À la ligne 1 :
SELECT *, count(date)
FROM albums
WHERE
(rank == 1)
AND
(date BETWEEN '1964*' AND '2015*')
GROUP BY album
ORDER BY count(date) DESC

Éditer le contenu d'une Cellule
Mode : Texte
1 362802

Type actuel des données de la cellule : Texte / Numérique
6 caractères

Serveur distant
Identité Sélectionner une identité pour se connecter

DBHub.io Local Base de Données en cours

Nom Dernière modification Taille Commite

Journal SQL Graphique DB Schema Serveur distant

DB Browser for SQLite - /home/lbl/Documents/Campus/2021_Data+/BDD/billboard/billboard-200.db

Structure de la Base de Données | Parcourir les données | Éditer les Pragma | Exécuter le SQL

```

1 SELECT *, count(date)
2 FROM albums
3 WHERE rank == 1
4 GROUP BY album
5 ORDER BY count(date) DESC

```

id	date	Michael Jackson	Thriller	rank	length	track_length	count(date)
1	362802	1984-04-14	Thriller	1	30	282689.866666667	37
2	568998	1963-08-17	Andy Williams	1	12	162825.333333333	33
3	428002	1978-01-14	Fleetwood Mac	1	58	NULL	31
4	68602	2012-06-23	Adele	1	11	261895.454545455	25
5	423202	1978-07-01	Soundtrack	1	21	224416.142857143	24
6	355002	1985-01-12	Prince And The Revolution	1	9	NULL	24
7	294402	1990-11-03	M.C. Hammer	1	NULL	NULL	21
8	267602	1993-05-29	Whitney Houston	1	12	280579.166666667	20
9	279802	1992-03-28	Garth Brooks	1	NULL	NULL	18
10	538593	1967-06-10	The Monkees	1	30	150713.4	18
11	320402	1988-05-07	Soundtrack	1	13	203054.0	18
12	367002	1983-11-19	The Police	1	11	253214.818181818	17
13	274402	1992-10-03	Billy Ray Cyrus	1	10	215845.0	17
14	291202	1991-02-23	Vanilla Ice	1	15	231595.133333333	16
15	216002	1998-05-09	Soundtrack	1	57	NULL	16
16	404202	1980-04-26	Pink Floyd	1	26	186898.538461538	15
17	493802	1971-09-25	Carole King	1	12	222795.166666667	15
18	535202	1967-10-07	The Beatles	1	13	186493.923076923	15
19	192202	1981-06-20	REO Speedwagon	1	19	211323.105263158	15

L'exécution s'est terminée sans erreur.
Résultat : 1857 enregistrements ramenés en 155ms
À la ligne 1 :
SELECT *, count(date)
FROM albums
WHERE rank == 1
GROUP BY album
ORDER BY count(date) DESC

Éditer le contenu d'une Cellule
Mode : Texte
NULL

Type actuel des données de la cellule : NULL
0 octet

Serveur distant
Identité Sélectionner une identité pour se connecter

DBHub.io Local Base de Données en cours

Nom Dernière modification Taille Commite

Journal SQL Graphique DB Schema Serveur distant

C3 - S'interfacer avec Python

1. Créer un jupyter notebook
→ [iteration3.ipynb](#)
2. Connectez-vous à la base de données à l'aide du module sqlite3.
 - a. Utilisez la ressource R3.4 pour comprendre le module sqlite sur Python
 - b. Créez une fonction ou une classe qui vous permet de facilement de :
 - i. Se connecter à votre BD
 - ii. Effectuer une requête
 - iii. Récupérer le résultat
3. Effectuez des requêtes SQL depuis Python.
 - a. Pour tester le bon fonctionnement
 - b. Puis pour répondre aux questions proposées en 2

C4 - S'interfacer avec Pandas

Il est possible de créer un *dataframe* à partir d'une requête SQL (cf R3.5).

Créez une seule requête SQL, et à l'aide de pandas répondez aux questions suivantes :

1. Effectuez la moyenne par année de toutes les caractéristiques. Quelle est la tendance que vous constatez ?
2. Quelle est l'année dont le niveau sonore «loudness » a été le plus haut ?
3. Quelle est la clé musicale la plus populaire - en prenant en compte le mode (e.g. majeur, mineur) ?



Bonus → tracez des courbes.

Quel graphique vous permet de mieux comprendre la popularité des tonalités, en faisant la différence entre majeur et mineur par note ?

Références / Ressources

R 3.1 – Données Billboard 200

<https://www.dropbox.com/s/z6bccb74k0d97f/billboard.zip?dl=0>

R 3.2 – C'est quoi Billboard 200 ?

https://en.wikipedia.org/wiki/Billboard_200

R 3.3 – Documentation sur les features de chanson par Spotify

<https://developer.spotify.com/documentation/web-api/reference/get-audio-features>

R 3.4 – Module Sqlite sur Python 3.6

<https://docs.python.org/3.6/library/sqlite3.html>

R 3.5 – Dataframe from SQL

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_sql.html

B - Modélisation d'une BDD - (PM)

Objectifs proposés :

- Lire un diagramme de BDD
- Traduire un diagramme en schémas (instructions SQL)
- Savoir insérer des données:
 - à l'aide de Python (Pandas)
 - d'un fichier SQL (bulk insert)

Consignes

Pour cette itération, nous allons utiliser le logiciel SQLite. SQLite est un SGBD, sa qualité principale est la légèreté du système. Il est très probable que certaines des applications mobiles installées dans votre téléphone utilisent SQLite pour stocker l'information.


C5.1 - Lire et compléter un diagramme UML

C5.1.1 : Un peu de lecture

Prenez du temps pour étudier les ressources R3.6 et R3.7

C5.1.2 : Discussion

La suite se déroule en équipe, par îlot et se focalise sur le document ci-dessous. Ce diagramme de classes correspond à la structuration de la donnée pour le système de transport d'une ville.

 Remarque : Système de transport != Système GPS.
Il ne s'agit donc pas d'un diagramme concernant les résultats d'un calculateur d'itinéraire tels que Google Maps.

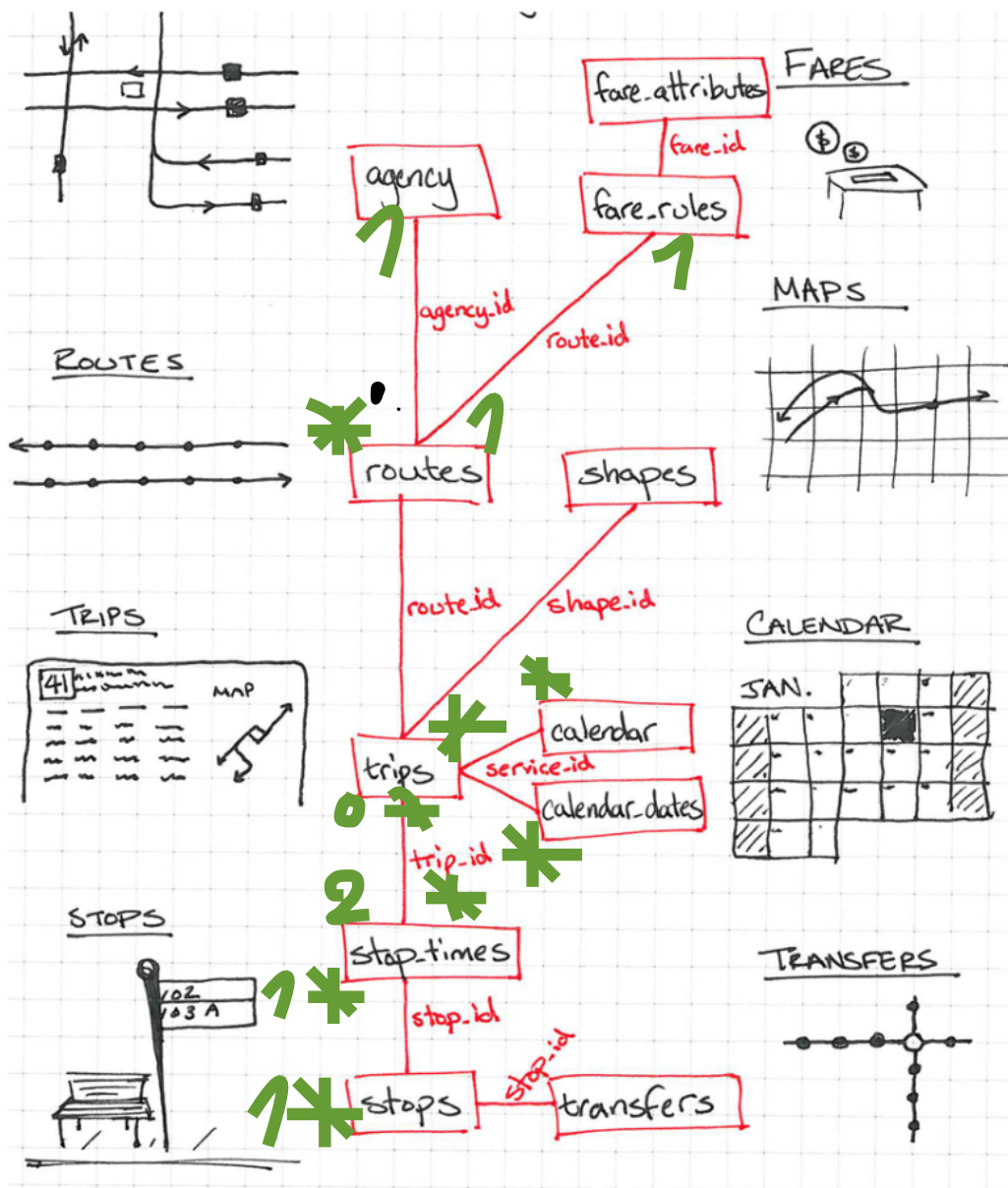


Figure 3.1 - Structuring transport systems

C5.1.3 : Diagramme

Le diagramme manque d'une information clé: la multiplicité des associations. En équipe, retrouvez les multiplicités des associations suivantes:

- Agency - Routes
- Trips - StopTimes
- StopTimes - Stops
- Trips - Calendar
- Trips - Frequencies

Pour mieux comprendre la logique métier, vous pouvez vous appuyer sur la ressource R3.8 et R3.9.

C5.2 - Création d'un schéma

Toujours en équipe, travaillez ensemble pour traduire votre diagramme de classes en un fichier SQL, lequel sera nommé "gtf_schema.sql". Les classes à traduire sont les suivantes:

1. Agency
2. Routes
3. Trips
4. StopTimes
5. Stops

→ ex : <https://gist.github.com/denysvitali/cf33fb42c3cfd26c0aabc8e849f8252d>

Une fois votre schéma finalisé, réalisez les opérations suivantes:

1. Créer une BDD nommée "gtfs_tag.db"
2. Exécuter votre schéma sur votre BDD
3. Vérifier, à l'aide du GUI, que vos instructions ont bien été prises en compte

Pour réussir cette étape, privilégiez la compréhension de la ressource R3.8. Utilisez R3.10 pour Commencer votre schéma, et R3.11 pour l'exécuter.

C5.3 - Insertion des données via Pandas

En autonomie, vous insérerez l'information correspondant aux classes:

- agency
- stops

Pour ce faire, vous utiliserez la méthode pandas permettant d'insérer des données dans une BDD.

C5.4 - Insertion des données (*bulk insert*)

En autonomie, vous travaillerez avec les données GTFS du réseau TAG (ressource R3.12). Votre mission sera d'insérer les données pour les classes suivantes:

- trips
- stoptimes

Voici les étapes à suivre pour ce faire (e.g. stops.txt):

1. Créer, en python, un fichier SQL (e.g. insert_stops.sql)
2. Exécuter le fichier dans la BDD

Des précisions concernant ces étapes sont présentées page suivante.

Dans un jupyter notebook

→ [Iteration3b.ipynb](#)

Étape 1 - implémenter une fonction qui génère une commande insertion SQL

Sa signature est la suivante :

```
def gen_insert_query(table_name:str, a_dict:dict) -> str
```

Les paramètres de cette fonction sont :

- tablename : le nom de la table (e.g. gtfs_stops)
- a_dict : dictionnaire Python

Elle retourne une chaîne de caractère qui représente l'insertion SQL.

Étape 2 - implémenter une fonction qui génère des commandes d'insertion SQL

Sa signature est la suivante :

```
def get_insert_queries(tablename:str, df: pd.DataFrame) -> list
```

Les paramètres de cette fonction sont :

- tablename : le nom de la table (e.g. gtfs_stops)
- df : le dataframe pandas

Elle retourne une liste de chaînes de caractère qui représente les insertions SQL correspondant au dataframe.

Étape 3 - implémenter une procédure qui crée un fichier SQL

Sa signature est la suivante :

```
def gen_insert_file(filename, tablename, df)
```

Les paramètres de cette procédure sont :

- filename : est le nom de fichier (e.g. insert_stops.sql)
- tablename : le nom de la relation où nous allons insérer nos tuples (e.g. gtfs_stops)
- df : un dataframe pandas

Étape 4 – Exécutez votre fichier SQL sur la BDD

Afin de rendre la transaction de votre fichier SQL efficace, regardez les mots clés BEGIN et COMMIT du langage SQL.

C5.6 - Réflexions

Vous avez inséré des données avec deux méthodes différentes (pandas, bulk Insert):

1. Quelle méthode est plus rapide et facile à implémenter ?
2. Quelle méthode est plus rapide pour insérer l'information ?
3. Dans quel scénario préconisez-vous l'utilisation d'une méthode ou l'autre ?

C5.7 - (👤 Bonus) Créer un calculateur d'itinéraire

À partir de :

- A - Un point GPS de départ, en format tuples de float;
- B – Un point GPS d'arrivée, en format tuples de float;
- H – une heure de part en format datetime;

Calculez l'itinéraire des lignes TAG à prendre.

Références / Ressources

R 3.6 – Diagrammes des classes (UML) - Stanford

<https://www.youtube.com/watch?v=LmS4Y99fNaQ>

<https://www.youtube.com/watch?v=X89KLfrNOPo>

R 3.7 – Référence diagrammes de classes – Microsoft

<https://docs.microsoft.com/en-us/visualstudio/modeling/uml-class-diagrams-reference?view=vs-2015>

R 3.8 – Référentiel GTFS

<https://developers.google.com/transit/gtfs/reference>

R 3.9 – GTFS en dessin

<https://xang1234.github.io/isochrone/>

R 3.10 – Exemple de schéma sql

<https://www.dropbox.com/s/t4s7fuo0fxynjqk/schema.sql?dl=0>

R 3.11 – Introduction à SQLite

<https://www.youtube.com/watch?v=giAMt8Tj-84>

<https://www.youtube.com/watch?v=xyCxLKEQPAs>

R 3.12 – GTFS Réseau TAG

<https://www.data.gouv.fr/fr/datasets/horaires-theoriques-du-reseau-tag/> → *data files for the job*

Quelques lectures complémentaires:

- <https://drive.google.com/drive/folders/1uAVFUAS-tnDwRrbMlxjBmUP8ONlYzudu?usp=sharing>
- <https://www.youtube.com/watch?v=rnz-vK8lesE>
- <https://towardsdatascience.com/arm-yourself-to-select-your-first-database-8bc9008bf8ec>
- <https://www.ionos.fr/digitalguide/sites-internet/developpement-web/diagramme-de-classes-uml/>

Les attaques par injection de SQL : qu'est-ce que c'est, comment s'en protéger en Python

<https://realpython.com/prevent-python-sql-injection/>

⇒ Prenez l'habitude d'écrire vos requêtes SQL en respectant les bonnes pratiques

Les contraintes: à quoi ça sert, quels sont les types utilisables

<https://vertabelo.com/blog/database-constraints-types/>

Un élément fondamental des bases de données : les index

<http://cerig.pagora.grenoble-inp.fr/tutoriel/bases-de-donnees/chap03.htm>

⇒ Ce lien n'est qu'une introduction, le sujet est vaste, mais il est utile d'en comprendre au moins les bases!

Bonus

Pour ceux ayant déjà fini à ce stade, 2 options:

1 - Créer une BDD avec les données RTE

2 - Commencer une sensibilisation au noSQL: <https://datascientest.com/nosql-tout-savoir>

Puis:

A vos remarques: <https://notes.parinix.org/BDD>