# Évaluation 2/2 - B3 - Développement SQL : Projet SQL

Date: 21/02/23

Auteur: Paul Schuhmacher

Version: 2

---



## Comment rendre votre travail

> *Merci de **lire attentivement** les consignes !*

**Publier** votre travail **sur un dépôt git public** (GitHub, Gitlab, BitKeeper, etc.) et fournir **le lien du dépôt**. Le dépôt contiendra les scripts MySQL et un fichier `README` donnant les instructions pour installer et utiliser votre projet.

> *Penser à utiliser un `.gitignore` pour éviter de pousser sur votre dépôt le contenu de `node_modules` et du dossier `db` contenant le volume docker de la base de données!*

Déposez ensuite votre travail sur **le devoir Teams** crée à cet effet. Voici les consignes sur le document à rendre :

- Document PDF, avec le nom du fichier formaté comme suit `evaluation2_x_abc.pdf`, où `x` est la première lettre de votre nom et `abc` votre prénom (je soumettrai donc le sujet `evaluation2_s_paul.pdf` ). Merci d'indiquer aussi votre nom et prénom **dans le document** !

Dans le document PDF :

- Nom et prénom
- **Le lien de votre dépôt Github/Gitlab qui héberge votre projet**.

> *Vous devez rendre votre travail **avant la date butoir fixée ensemble sous peine de pénalité** : 1 point le premier jour de retard, 2 points le deuxième jour de retard, et ainsi de suite jusqu'à un minimum de 0.*

# Notation

Le projet est noté sur 30 (ramené sur 20), **coefficient 2**.

- **Documentation du projet** : **6 points**. Le dépôt doit contenir un `README.md` (Markdown) *bien formé* (voir les instructions ci-dessous). Les commandes à effectuer doivent être indiquées. Les consignes sont respectées, les scripts SQL sont commentés. Le texte **a été corrigé par un outil de correction automatique**.
- **Conception (MCD)** : **14 points**. Modèle Conceptuel des Données bien établi et normalisé, types adéquats, clefs bien formées, données pertinentes en lien avec le recueil des besoins fourni en annexe. MCD **lisible**.
- **Implémentation/Requêtes (Scripts SQL)** : **12 points**. Votre projet doit implémenter les spécifications demandées (répondre aux besoins). **Chaque script SQL (fichier) doit pouvoir être exécuté en une seule commande.** Chaque requête doit être commentée.

> *Ne pas répondre à une question bonus n'est pas pénalisant. Cela n'apporte que des points en plus.*

# À rendre

L'URL de votre dépôt. Votre dépôt doit contenir *a minima* :

- Un fichier `README.md` avec les sections suivantes :
  - **Table des matières** (vous pouvez la générer automatiquement), pour naviguer facilement dans le document
  - **Utiliser les scripts SQL** : instructions pour lancer chaque script SQL. Idéalement, je dois pouvoir copier/coller les instructions dans l'ordre indiqué **sans me poser de questions**.
  - **Conception** :
    - **Dictionnaire des données (optionnel)** : le dictionnaire des données avec les colonnes suivantes : Libellé, Code (nom dans le système), Type (A, N, AN, D, B), Longueur, Obligatoire (Oui ou Non), Règle de calcul, contrainte d'intégrité, commentaire. Voir le template avec quelques exemples. Le dictionnaire des données peut être réalisé en Markdown ou mieux, dans *une feuille de classeur* (Excel, Libre Office Calc)

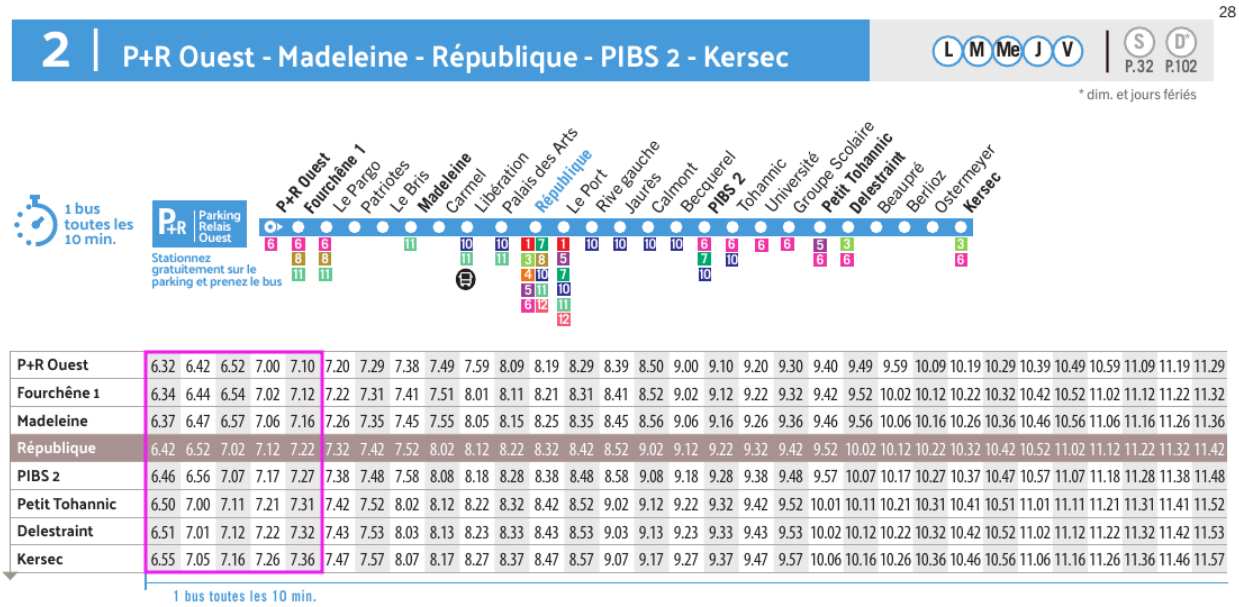
        - **Le modèle conceptuel des données (MCD)**, au format png, jpeg ou svg.
    - **Remarques (optionnel)** : si vous voulez faire des remarques sur votre travail, difficultés rencontrées, modifications apportées, etc. C'est l'endroit pour justifier vos choix.
    - **Références (optionnel)** : la liste des références (sites web, cours, livre, article, billet de blog, etc.) qui vous ont aidé à concevoir et développer votre système
- Les scripts SQL **commentés** (par convention on leur donnera l'extension `.sql`) : `schema.sql`, `data.sql` et `queries.sql`

> ***Le dictionnaire des données n'est pas obligatoire ici**, un MCD clair et éventuellement commenté suffira. Vous êtes encouragés à l'écrire tout de même en guise d'exercice.*
>
> *Un dépôt sur le web **est un site web** et le fichier `README.md` en est sa page d'accueil. Soignez-le pour aider vos utilisateur·ices (et vous-même !) à (ré)utiliser votre projet.*

# Sujet

Vous devez modéliser une partie de la base de données de la société Kicéo, en charge des transports en commun de Vannes. La base de données doit permettre de mémoriser et retrouver les horaires de toutes les lignes de bus de la ville.

Votre système d'information doit être capable de produire *à la demande* les horaires d'une ligne. Par exemple, voici les horaires de la ligne 2 aux arrêts principaux :




*Source de l'image (accéder au PDF complet), téléchargeable sur le site web de Kiceo*

## Remarques, simplifications par rapport au réel

- On ne s'intéresse pas à la mise en forme ou mise en page, **uniquement aux données**;
- On ne s'intéressera qu'à la ligne 2;
- On ne s'intéressera qu'*aux arrêts principaux* de la ligne 2 (ceux figurant sur la fiche dans le tableau ou en gras sur le schéma de la ligne);
- On ne considérera *que les jours de la semaine*, pas les samedis, dimanche et jours fériés. Plus précisément, on ne s'intéressera qu'au lundi;
- On se limitera aux horaires de **6.32 (P+R Ouest)** à **7.10 (P+R Ouest)**;
- On se limitera à la ligne 2 **P+R Ouest direction Kersec**.

## Conception et DDL (14pt)

1. **Réaliser le MCD** du système d'information permettant de répondre au besoin. Aidez-vous des entités déjà identifiées. Penser à faire figurer les noms des associations, des cardinalités (multiplicité) à chaque extrémité de chaque association. Vous pouvez également ajouter des commentaires sur le schéma si nécessaire. Vous pouvez utiliser la notation d'UML ou Merise au choix. Vous pouvez (et êtes encouragé·e) **à utiliser un logiciel de conception** pour réaliser le diagramme comme MySQL Workbench ou le plus modeste AnalyseSI. **Penser à déposer une image de votre MCD** (format png, jpeg ou svg) sur le dépôt comme indiqué dans les consignes.
2. **Implémenter le schéma avec MySQL**. **Cette étape est automatique si vous utilisez un logiciel de conception**. Apportez des modifications si nécessaire. Créer un script `schema.sql` contenant toutes les instructions pour générer votre base de données uniquement. Le script doit permettre de regénérer le schéma à volonté *en une seule commande*. Fournir la commande (avec le programme `mysql`) à executer pour régénérer le schéma, comme indiqué dans les consignes. *Tip: Penser à `DROP` les tables avant de les créer et à utiliser `IF NOT EXISTS`*.
3. **Créer un fichier `data.sql`** qui permet d'insérer **le jeu de données de la ligne 2** (horaires aux arrêts principaux de la ligne 2 pour les jours de semaine, etc.). Le script doit permettre de régénérer le jeu de données à volonté *en une seule commande*. Fournir la commande (avec le programme `mysql`) à executer pour régénérer le jeu de données, comme indiqué dans les consignes. *Tip: Penser à `DELETE` les anciennes données lors de la régénération*.

## Requêtes (12pt)

Une fois votre base conçue et contenant les données indiquées, créer un fichier `queries.sql` contenant les requêtes SQL répondant aux besoins suivants :

1. Afficher la table des horaires dans l'ordre chronologique à l'arrêt **Madelaine**. Faire de même pour l'arrêt **République**.

Sorties attendues :

```
+--------------------------------------+
| Horaires à l'arrêt Madelaine (Lundi) |
+--------------------------------------+
| 06:37:00                             |
| 06:47:00                             |
| 06:57:00                             |
| 07:06:00                             |
| 07:16:00                             |
+--------------------------------------+
```

```
+----------------------------------------+
| Horaires à l'arrêt République (Lundi)  |
+----------------------------------------+
| 06:42:00                               |
| 06:52:00                               |
| 07:02:00                               |
| 07:12:00                               |
| 07:22:00                               |
+----------------------------------------+
```

2. Afficher le parcours complet de la ligne 2 **Direction Kersec** (la liste des arrêts dans l'ordre de passage). *Tip: Au besoin, penser à désactiver le mode `ONLY_FULL_GROUP_BY` par défaut avec l'instruction :*

```sql
SET sql_mode=(SELECT REPLACE(@@sql_mode,'ONLY_FULL_GROUP_BY',''));
```

*Désactiver ce mode permet notamment d'utiliser `ORDER BY` sur une colonne même si cette colonne n'est pas utilisée dans le `GROUP BY`. Il vous revient la charge de vous assurer que la valeur utilisée pour ordonner soit bien unique pour chaque groupe, au risque d'obtenir des résultats non déterministes.*

Sortie attendue :

```
+---------------------------------------+
| Parcours de la ligne 2 Direction Kersec |
+---------------------------------------+
| P+R Ouest                             |
| Fourchêne1                            |
| Madelaine                             |
| République                            |
| PIBS 2                                |
| Petit Tohannic                        |
| Delestraint                           |
| Kersec                                |
+---------------------------------------+
```

3. Des travaux sont en cours dans la ville. L'arrêt **Petit Tohannic** ne sera plus desservi temporairement. Un arrêt non desservi temporairement doit indiquer l'arrêt le plus proche auquel se rendre. On redirigera les voyageur·ses vers l'arrêt **Delestraint**. **Modifier le schéma de la base de données** pour modéliser ce besoin. Placer les instructions pour modifier la base dans le fichier `schema.sql`.
4. A l'arrêt **Petit Tohannic**, au lien d'afficher la table des horaires, afficher le message `"Arrêt temporairement non desservi."`. *Tip: utiliser l'instruction `CASE`. Voici la syntaxe générale :*

```sql
SELECT
    colonne1,
    CASE
        WHEN condition THEN valeur_si_vrai
```

```
    ELSE valeur_si_faux    END AS resultatFROM        votre_table;
```

5. Écrire la requête pour mettre à jour l'arrêt **Petit Tohannic** et la requête pour afficher le message. Sortie attendue :

```
+-------------------------------------------------------------------------------+
| Horaires à l'arrêt Petit Tohannic (Lundi)                                     |
+-------------------------------------------------------------------------------+
| L'arrêt n'est pas desservi. Veuillez vous reporter à l'arrêt Delestraint.     |
+-------------------------------------------------------------------------------+
```

6. Ajouter la ligne 2 dans la direction opposée **Direction P+R Ouest** ainsi que les stations qu'elle dessert. **Réutiliser les mêmes horaires que pour l'autre direction. On se limitera ici a un seul trajet de bus, le premier de la journée.**.
7. Afficher pour chaque ligne, le parcours complet (liste des arrêts dans l'ordre de passage). On y indiquera également les arrêts temporairement non desservis. *Tip: Utiliser la fonction* `GROUP_CONCAT`.

Sortie attendue :

```
+---------------------+-------------------------------------------------------------------------------------------+
| Ligne               | Arrêts desservis                                                                          |
+---------------------+-------------------------------------------------------------------------------------------+
| 2 Direction Kersec  | P+R Ouest,Fourchêne1,Madelaine,République,PIBS 2,Petit Tohannic,Delestraint,Kersec        |
| 2 Direction P+R Ouest | Kersec,Delestraint,Petit Tohannic,PIBS 2,République,Madelaine,Fourchêne1,P+R Ouest      |
+---------------------+-------------------------------------------------------------------------------------------+
```

8. Durant le cours nous avons évoqué les indexs mais n'avons pas abordé comment ils sont fabriqués et comment ils améliorent la lecture dans une base de données. Réaliser un mini travail de veille sur l'utilisation des indexs dans une base de données MySQL. **Présenter en quelques lignes** : le principe de fonctionnement et la différence entre un index de type *B-Tree* et *Hash*. Vous pouvez vous aider d'un schéma. (**3pt**)

9. Étant donnés les besoins actuels sur votre base, et d'après sa structure, sur quelles colonnes serait-il judicieux de créer des indexs ? Pourquoi ?

10. **Écrire une procédure stockée** `insert_schedule(time_start ,time_end, step)` permettant d'insérer des horaires, minute par minute dans une table `schedule`. Sortie attendue :

```
CALL insert_schedule('06:32', '06:41', '0:01');
SELECT horaires FROM schedule:
+----------+
| horaires |
+----------+
| 06:32:00 |
| 06:33:00 |
| 06:34:00 |
| 06:35:00 |
| 06:36:00 |
| 06:37:00 |
| 06:38:00 |
| 06:39:00 |
| 06:40:00 |
```

```
| 06:41:00 |+----------+
```

# Annexes

## Template de dictionnaire des données

Voici un template de dictionnaire des données avec quelques exemples.

| Libellé | Code | Type | Longueur | Obligatoire ? | Regle de calcul/Contrainte/Commentaire |
|---|---|---|---|---|---|
| Date de commande | date | D | 6 | Oui | Format jjmmaa, jj de 01 à 31, mm de 01 à 12 |
| Quantité commandée | qtite | N | 3 | Oui | > 0 |
| Rue du client | rue_cli | AN | 30 | Oui | |
| Ville du client | ville_cli | A | 30 | Oui | |
| Code postal du client | cp_cli | AN | 5 | Oui | |
| Adresse du client | adresse_cli | 60 | 30 | Oui | adresse_cli = rue_cli + ville_cli + cp_cli |
| Numéro de téléphone fixe du client | tel_dom_cli | AN | 14 | Non | |

Remarques :

- La colonne **Longueur** indique la taille de la donnée en octet/byte. Un caractère (ASCII) est généralement encodé sur un octet. Pour un entier, cela indiquera le nombre d'octets sur lequel il est encodé. Un entier de longueur 1 permet de créer $2^{8^1} = 256$ codes, donc de 0 à 255. En MySQL, c'est un `TINYINT`. Un entier de longueur 4 permet de créer $2^{8^4} = 4294967296$ codes, donc de 0 à 4294967295. En MySQL, c'est un `INT`.
- Type : **A**lphabétique, **N**umérique, **A**lpha**N**umérique, **B**ooléen, **D**ate
- `adresse_cli = rue_cli + ville_cli + cp_cli` indique que `adresse_cli` est calculée à partir de `rue_cli`, `ville_cli` et `cp_cli`.