LP DIM - Bases de données – TD2
LMD & LDD (Partie 1)

Un Accompagnateur Moyenne Montagne (AMM) est un professionnel breveté qui emmène des personnes en montagne. Il souhaite gérer dans une base de données les sorties qu'il encadre. Cette base de données sera constituée des informations suivantes.

Tout d'abord il voudrait gérer un catalogue des différents sommets sur lesquels il propose d'emmener ses clients. Un sommet est caractérisé par son nom, son altitude, la référence de la carte IGN où il est cartographié (nous considérerons qu'une seule carte est associée au sommet). Une même carte peut référencer plusieurs sommets.

N° sommet	Sommet	Altitude	Carte
1	Tournette	2351m	3431 Ouest (Lac d'Annecy)
2	Mont Charvin	2409m	3531 Ouest (Megève - Col des Aravis)
3	Trou de la Mouche	2453m	3430 Est (La Clusaz - Le Grand Bornand)
4	Tête Pelouse	2537m	3430 Est (La Clusaz - Le Grand Bornand)

Un sommet peut être gravi par plusieurs itinéraires différents (« par le versant Nord », « voie normale », …). Pour chaque itinéraire d'un sommet, on connait le titre de l'itinéraire, son code difficulté (F, PD-, PD, PD+, AD-, AD, AD+, D, TD, TD+, ED), son orientation (Nord (N), Nord-Est (NE), Nord-Ouest (NO), Est (E), Sud-Est (SE),…), sa dénivelée, son temps de parcours théorique. Le numéro de l'itinéraire dépend du numéro de sommet.

N° sommet	Sommet	N° itinéraire	Itinéraire	Difficulté	Dénivelée	Orientation	Temps de parcours (mn)
1	Tournette	1	versant Nord-Est	PD	1350m	NE	270
3	Trou de la Mouche	1	par la combe du Grand Crêt	PD	1010m	NO	210
3	Trou de la Mouche	2	par la combe de Paccaly	PD+	1010m	NO	240

Pour chaque sortie (une seule par jour au maximum), l'AMM souhaite référencer les personnes qu'il a emmenées et les conditions de la sortie : la date de la sortie, les conditions météo, un commentaire sur la sortie. Une sortie concerne un itinéraire ou plusieurs itinéraires : en général, un (boucle) ou deux (un pour l'aller et un second pour le retour). Il est également important de connaître la durée de chaque itinéraire emprunté lors de la sortie.

Date	Participants	Météo	Commentaire	Itinéraires	Durée de l'itinéraire (mn)

Ces clients étant plutôt fidèles, il gère un petit annuaire où il enregistre les nom, prénom, adresse (rue, CP et ville), téléphone fixe, email et âge (date de naissance).

Q1 :

a. A partir de l'exemple fourni dans le document « Elaboration MCD », réaliser le MCD permettant de gérer les informations précédentes.

b. Réaliser le MLD relationnel.

c. Réaliser le MPD.
Quelques règles simples d'optimisation à appliquer pour créer le MPD :
- Conservation ou non des tables de référence (i.e. tables en général ayant peu de données) ? Si le contenu de celles-ci n'évolue pas et n'évoluera pas dans le futur, il n'est pas nécessaire de les conserver => ne pas générer la table et la remplacer par une contrainte de validation (check) au

niveau de la clé étrangère. Cela évitera en outre d'utiliser une jointure inutile lors des requêtes.
- Limitation des clés primaires composées à 2 champs au maximum (sinon fichier d'index imposant en mémoire). Comment faire alors pour assurer l'unicité des champs composant la clé primaire si ceux-ci ne le sont plus ?

Pour répondre aux questions 2 à 11, utilisez les annexes et/ou la doc PostgreSQL (http://docs.postgresqlfr.org/)**.**

Q2.
A partir du fichier « Doc complémentaire - Enoncé TD2 » et du fichier « Script partiel TD2.sql », compléter le script de création des tables correspondant au MPD (pour les types de données se référer à la documentation, chapitre 8 ou au cours). Ce script doit également créer les contraintes d'intégrité adéquates (clés primaires, contraintes de validation, contraintes d'intégrité référentielle, contraintes d'unicité).
Un script doit contenir trois parties :

1. Suppression des tables. Instruction `DROP TABLE` (Cf. documentation) en utilisant l'option `IF EXISTS`.
2. Création des tables (`CREATE TABLE`). Vous créerez également dans cette partie les contraintes de clé primaire, uniques et de validation (`CHECK`). Vous nommerez l'ensemble des contraintes de façon appropriée (**Cf. annexe 1**). Vous pourrez exécuter au fur et à mesure votre script.
3. Création des contraintes d'intégrité référentielle (clés primaires). Instruction `ALTER TABLE… ADD CONSTRAINT…`

Exécuter ce script plusieurs fois dans le schéma `Public` pour voir s'il fonctionne.

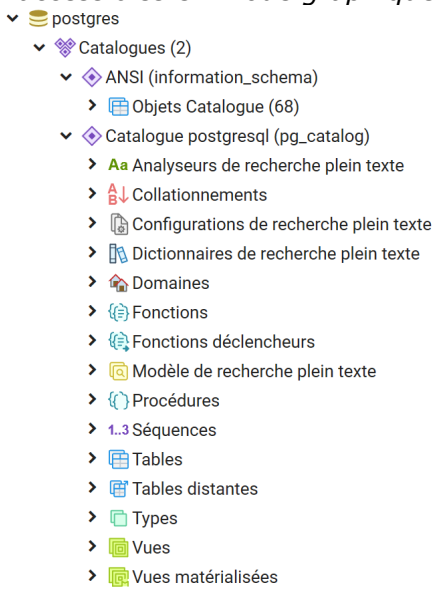
Par la suite, répondez aux questions suivantes dans un fichier texte

Q3 : Schéma d'informations
Le schéma d'informations (`information_schema`) (Cf. chapitre 34 de la documentation) consiste en un ensemble de vues système contenant des informations sur les objets définis dans une base de données. Les vues du schéma d'information ne contiennent pas d'information sur les fonctionnalités spécifiques à PostgreSQL; pour cela, vous devez utiliser les catalogues système (`pg_catalog`) ou d'autres vues spécifiques à PostgreSQL.

Visualisez les données stockées dans les vues du schéma d'informations (`information_schema`) de la base postgres (vues système à utiliser : `tables`, `columns`, `table_constraints`, `constraint_column_usage`, `constraint_table_usage`, `check_constraints`, `key_column_usage`, `referential_constraints`) ;
Ex. :

```
select * from information_schema.tables where table_schema = 'public';  -- Il est nécessaire de préfixer par le nom du schéma, sinon la recherche se fait dans le schéma actuel (i.e. public)
select * from information_schema.constraint_column_usage;
```

Remarque : les catalogues sont aussi accessibles en mode graphique dans PgAdmin 4



A quoi sert la vue système `table_constraints` ? Expliquer son contenu.

Expliquez la différence entre les vues système `constraint_column_usage` et `constraint_table_usage`.

A quoi sert la vue système `key_column_usage` ? Expliquer son contenu.

Remarque : il est toujours nécessaire de bien connaître les vues et/ou tables système quand on utilise un SGBD. Celles-ci sont spécifiques au SGBD utilisé (Oracle, MySQL,…).

Q4 : Catalogue système
En plus du schéma `public` et de ceux créés par les utilisateurs, chaque base de données contient un schéma `pg_catalog`. Celui-ci contient les tables systèmes et tous les types de données, fonctions et opérateurs intégrés.

Visualisez les données du catalogue système (`pg_catalog`) : tables `pg_tables`, `pg_constraint`
Ex. : `select * from pg_tables;`
Expliquer le contenu de la table `pg_constraint`.

Q5 : Séquence
Définissez une séquence (<=> numéro automatique) (**Cf. Annexe 2**), nommée `CLIENT_SEQ`, afin de faciliter la mise en place d'un numéro pour chaque membre. Cette séquence doit commencer avec la valeur 1 et elle possède un pas d'incrément de 1. Elle sera associée à la colonne `ID_CLIENT` de la table `CLIENT`. Après création, visualisez les données de la vue système `sequences` du schéma d'informations.
NB : l'essai de la séquence sera réalisé par la suite, lors d'insertion de données.

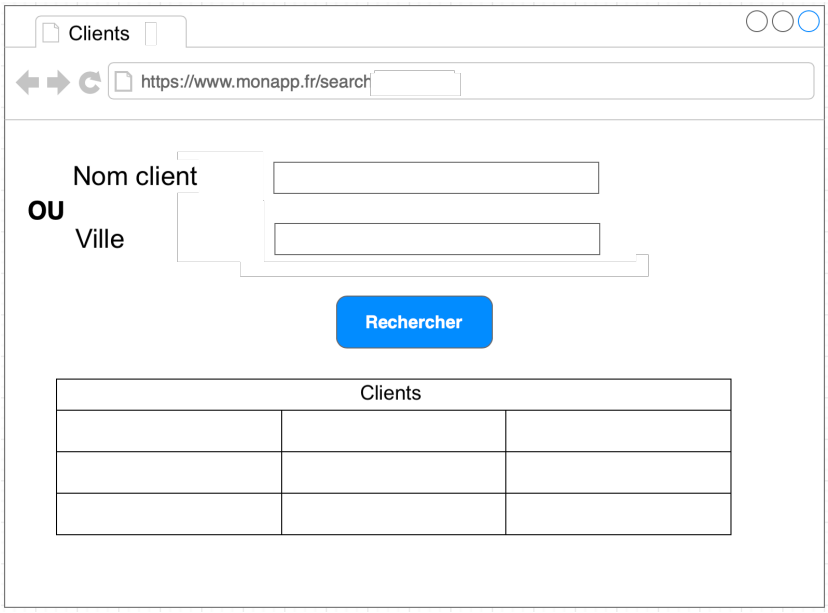
Q6 : Contrainte d'unicité
Les clients sont très nombreux et si certains réalisent souvent des sorties en montagne, d'autres au contraire le font peu fréquemment. Pour ces derniers, il n'est pas souhaitable d'avoir des informations en double dans la base. Aussi il ne doit pas être possible d'avoir deux clients qui possèdent mêmes nom, prénom et numéro de téléphone fixe. Définissez une contrainte d'intégrité afin de satisfaire cette nouvelle exigence. La contrainte sera ajoutée sur la table des clients (instruction `alter table`) (visualisez les données des vues système `constraint_column_usage`, `constraint_table_usage` et `key_column_usage` avant et après l'ajout des contraintes).

Q7 : Contraintes de validation
De plus en plus de clients possèdent deux numéros de téléphone : un pour le poste fixe de leur domicile et un pour leur téléphone portable. Or la base ne nous permet de stocker qu'un seul numéro de téléphone. Apportez les modifications de structure nécessaires pour prendre en compte cette modification.
Comme cette nouvelle colonne, nommée `MOBILE`, va contenir des informations relatives à un numéro de téléphone portable, mettez en place une contrainte de validation afin de vous assurer que le numéro de téléphone saisi commence bien par `06` ou par `07` (utilisez des fonctions de manipulation de chaines et notamment la commande `like`). Visualisez les données des vues système `constraint_column_usage`, `constraint_table_usage` et `key_column_usage`.

Q8 : Index
Afin d'améliorer les performances d'accès aux données, définissez un index sur toutes les colonnes de type clé étrangère (ou groupes de colonnes si FK composée de plusieurs champs). Les opérations de jointure seront plus rapides ; nous y reviendrons lors d'un prochain TP sur l'optimisation).
Indications : il y a 7 index à créer, i.e. autant que de flèches dans le MPD (flèche = FK).

Q9 : Index

Vous venez de réaliser la maquette suivante permettant de rechercher un client. Quels index proposez-vous ? Ecrire le script permettant de les créer.



Q10 : Modifiez la table des sorties afin que la colonne `DATE_SORTIE` prenne par défaut la valeur de la date du jour (clause `DEFAULT` ; cf. commande `ALTER TABLE`).

Q11 : À l'usage, on se rend compte que lorsque l'on souhaite supprimer une sortie, il faut nécessairement supprimer toutes les lignes présentes dans la table `PARTICIPE` qui font référence à la ligne de la table `SORTIE` que l'on souhaite supprimer. Comment est-il possible de rendre automatique une telle suppression ? (*Modifiez le comportement de la contrainte de clé étrangère en cas de suppression de la ligne maître).* Visualisez les données de la vue système `referential_constraints`. Quelles modifications ont été réalisées ?

Q12 : La table `PARTICIPE` n'est pas bien nommée. Le nom `PARTICIPATION` semble préférable. Renommez la table afin de prendre en compte cette nouvelle exigence (cf. commande `ALTER TABLE`).

ANNEXES

1. Création d'une table

La création consiste à définir (en fonction de l'analyse) le nom des colonnes, leur type, une valeur par défaut à la création de la ligne (default), les règles de gestion s'appliquant à la colonne (`CONSTRAINT`). Si une règle de gestion concerne plusieurs colonnes de la ligne, on définit une contrainte de table.

Syntaxe

```
CREATE TABLE nom (nomcolonne type [DEFAULT expr]
[[CONSTRAINT nom contrainte-de-colonne...],...
[,CONSTRAINT nom contrainte-de-table...]);
```

Dénomination des contraintes

Les contraintes peuvent être nommées afin d'être plus facilement manipulées ultérieurement (activation, suppression). Dans le cas où aucun nom n'est affecté explicitement à une contrainte, PostgreSQL génère automatiquement un nom (`xxx_fk` ou `xxx_pk`,…).
Lors de l'affectation explicite d'un nom à une contrainte, on utilise en général la dénomination suivante :

`Table_Colonne_TypeDeContrainte` OU `TypeDeContrainte_Table_Colonne`

`Table` Nom de la table sur laquelle est définie la contrainte.
`Colonne` Nom de la (ou des) colonne(s) sur laquelle est définie la contrainte.
`TypeDeContrainte` :

`PK`	Clé primaire
`UQ` ou `UK`	Clé unique
`CK`	Check
`FK`	Clé étrangère

2. Les séquences

La création d'un objet `SEQUENCE` met à disposition de l'utilisateur un générateur de nombres. Les séquences sont utilisées pour générer des numérotations automatiques, en particulier pour la création de valeurs de clé primaire. Son utilisation est plus souple et donne de meilleures performances que la gestion manuelle des compteurs par l'intermédiaire d'une table. Elle ne garantit pas cependant l'absence de "trous" dans la numérotation. La séquence est un simple générateur de numéros et tous les numéros sont différents mais si des numéros sont demandés à une séquence et ne sont pas utilisés par la suite, alors ces numéros sont perdus. La séquence est en effet un objet à part entière et peut être utilisée par plusieurs tables.
Chaque valeur séquence s'exprime au maximum sur 28 chiffres significatifs.
Il est possible, sous PostgreSQL, de remplacer l'utilisation d'une séquence par un type de données `SERIAL` associé au champ. A la différence d'un type `SERIAL`, une séquence n'est pas forcément associée à un champ d'une table et doit être appelé lors de chaque insertion d'enregistrement.

Syntaxe

```
CREATE SEQUENCE nom [paramètres];
ALTER SEQUENCE nom paramètres ;
DROP SEQUENCE nom ;
```

Paramètres

`START WITH n`
Valeur initiale.

`INCREMENT BY n`
Pas d'incrémentation. Il peut être positif ou négatif.

`MINVALUE n/NO MINVALUE`
Valeur limite minimum ou non.

`MAXVALUE n/NO MAXVALUE`
Valeur limite maximum ou non.

`CYCLE/NO CYCLE`
`CYCLE` force la séquence à repasser à `MINVALUE` lorsque `MAXVALUE` a été atteinte (séquence croissante) ou à `MAXVALUE` lorsque `MINVALUE` a été atteinte (séquence décroissante).

`CACHE`
Force l'anticipation de la génération des valeurs suivantes de la séquence en mémoire. A pour effet d'améliorer les temps de réponse de la séquence.

`OWNED BY *table.colonne*, OWNED BY NONE`
Permet d'associer la séquence à une colonne de table spécifique. De cette façon, la séquence sera automatiquement supprimée si la colonne (ou la table entière) est supprimée. La table indiquée doit être dans le même schéma que la séquence. `OWNED BY NONE`, valeur par défaut, indique qu'il n'y a pas d'association.

L'utilisation se fait de la façon suivante :

- Créer une séquence ascendante appelée `serie`, démarrant à 101 :
  `CREATE SEQUENCE serie START 101;`
- Sélectionner le prochain numéro de cette séquence :
  `SELECT nextval('serie');`
- Utiliser cette séquence dans une commande `INSERT` :
  `INSERT INTO distributors VALUES (nextval('serie'), 'nothing');`