Projet Bases de Données (L3-M1 Bio-Info)

Le projet s'effectue par groupe de 2 personnes.

La récupération de données existantes (e.g., noms de groupes, genre musicaux, lieux réels) est non seulement autorisée, mais encouragée. En revanche, les règles anti-plagiat de l'Université s'appliquent à votre travail et toute utilisation d'éléments extérieurs devra être explicitement évoquée et sourcée. Vous pouvez discuter des différents aspects du projet avec les autres groupes, mais toute copie partielle ou totale de code entre plusieurs groupes est interdite.

Rendu – Date : 8 Janvier 2024

Une Une *unique* archive déposée sur Moodle contenant :

Code

Format	:	L'ensemble de vos fichiers de code et un fichier `README.txt` expliquant brièvement à quoi correspond chaque fichier, déposée sur Moodle.
Consignes	:	Implémentez dans `PostgreSQL` votre base de données et les requêtes permettant de l'utiliser, en utilisant votre modélisation E/R et les consignes détaillées à partir de la page 2.

Rapport

Format	:	Rapport entre 4 et 8 pages au format PDF. Vous pouvez utiliser `https://www.freepdfconvert.com/fr` pour convertir en PDF.
Consignes	:	Vous inclurez votre modèle entités-associations et un schéma relationnel de vos tables. Vous expliquerez les choix qui ont motivé votre modélisation, et les limitations de votre modèle. Vous décrirez également les requêtes que vous avez choisies d'implémenter.

Soutenance – Date : 10 janvier 2024

Format	:	Présentation orale par *tous* les membres du groupe.
Consignes	:	Vous débuterez votre soutenance par une présentation de votre travail. Nous vous poserons ensuite des questions sur vos choix de modélisation, les contraintes d'intégrité utilisées, les points forts et les points faibles de votre approche, le fonctionnement de vos requêtes, etc.

1 Un réseau social centré sur la musique

Un nouveau réseau social open source, participatif et sans publicité va voir le jour. Il permettra à ses utilisateurs de découvrir une multitude de nouveaux sons parmi les genres existants et d'aller à de concerts.

Dans ce projet, nous vous proposons de participer à la création de la base de données qui sera utilisée par le réseau social.

Les utilisateurs du réseau social pourront non seulement être des personnes, mais également des groupes, des associations organisant des concerts, des salles de concert, etc. Il sera possible à un utilisateur de suivre ou bien d'être ami avec une autre entité du réseau. Notez que la relation d'amitié est symétrique, mais qu'il est possible que A suive B sans que B suive A (sur le modèle du *follows* de Instagram). Il sera également possible pour les utilisateurs d'annoncer des concerts. Les caractéristiques de ces concerts pourront être listées (lieu si connu et légal, prix, organisateurs, line-up, nombre de places disponibles, besoin en volontaires pour aider l'organisation, cause en soutien duquel le concert aura lieu, espace extérieur, possibilité d'emmener ses enfants, etc.). Les utilisateurs qui sont des personnes pourront indiquer être intéressés ou participer à un évènement (mais pas les deux à la fois). Pour les concerts ayant déjà eu lieu, il sera possible d'archiver un ensemble de données telles que nombre de participants, avis des participants concernant telle performance de tel groupe lors du concert, photos, vidéos, etc. De manière plus générale, les utilisateurs du réseau social pourront enregistrer un certain nombre d'avis concernant groupes, morceaux, concerts et lieux. Ces avis pourront être exprimés au moyen de notes, mais aussi de commentaires rédigés qui pourront être tagués par des mots clefs. Ces mots clefs pourront également

servir à taguer groupes, lieux et événements. Les genres musicaux et leurs sous genres pourront être utilisés comme mots clefs, mais n'importe quel autre mot pourra également être choisi (sur le modèle des hashtags de Instagram ou Twitter).

Les utilisateurs auront par ailleurs la possibilité d'afficher dix playlists maximum sur leur page, composées chacune d'au plus vingt morceaux. Chaque morceau pourra être assigné une certaine note par tout utilisateur, mais ces notes pourront par la suite être modifiées (un utilisateur pourra ainsi décider que "Je danse le Mia" ne survit pas aussi bien à l'épreuve du temps que "Demain c'est loin"). Les playlists pourront elles aussi être taguées. Notez que les playlists des groupes n'affichent que des morceaux de ces groupes.

Plusieurs services seront proposés aux utilisateurs pour consulter leur historique, ainsi que certaines suggestions (concerts auxquels ils ont assistés, genres voisins qu'ils pourraient explorer, personnes avec lesquelles ils pourraient avoir des affinités, villes dans lesquelles ils pourraient trouver à s'amuser,...).

Concevez une base de données qui permette de gérer ce réseau social. Si vous le souhaitez, vous pourrez vous inspirer des réseaux existants (comme Facebook ou feu Myspace) pour ajouter quelques fonctionnalités (publication de "posts" par exemple), mais aussi ajouter vos propres fonctionnalités. Réfléchissez au préalable aux requêtes que vous serez amenés à implémenter (Section 3 et 4) pour savoir quelles informations seront indispensables dans votre base. Vous commencerez par réaliser le modèle conceptuel de données (schéma entités-associations) pour le Rendu 1, puis vous pourrez passer à l'implémentation dans PostgreSQL pour le Rendu 2. Utilisez les différents outils étudiés en cours et en TP (clés primaires/étrangères, contraintes d'intégrité, types adaptés ...). Expliquez dans le rapport (Rendu 2bis) et lors de la soutenance (Rendu 3) comment vous avez conçu cette base de données, les choix que vous avez effectués, les avantages, limites et améliorations possibles de votre modèle, etc.

Prenez bien garde à dégager les contraintes externes, i.e., veillez à bien séparer ce qui relèvera du SGBD et ce qui relèvera de l'application et précisez-le dans le rapport. Il est normal que vous ne puissiez pas implémenter toutes les contraintes externes au niveau du SGBD avec ce que nous avons appris pour le moment. Vous êtes libres d'implémenter des triggers si vous le souhaitez, mais ce n'est pas demandé et pas au programme de cette année.

Aide à l'utilisation de PostgreSQL

En plus de vos cours et TPs, vous êtes encouragés à consulter la documentation en français de PostgreSQL disponible ici :

`https://docs.postgresql.fr/current/`

Les sections `I. Tutoriel` (`https://docs.postgresql.fr/current/tutorial.html`) et `II. Langage SQL` (`https://docs.postgresql.fr/current/`) sont les plus pertinentes. Il y est expliqué comment compter, calculer un maximum, ajouter des contraintes lors de la création d'une table (empêcher un attribut d'être négatif par exemple), faire des opérations mathématiques, etc.

Vous pouvez également consulter le fichier `Installation et commandes basiques pour PostgreSQL` disponibles sur Moodle :

`https://moodle.u-paris.fr/course/view.php?id=1655`

2 Peuplez vos tables

Maintenant que vos tables sont créées, vous pouvez les remplir avec les informations adaptées (groupes, albums, utilisateurs, concerts...). Insérez au moins une centaine de tuples au total dans votre base de données. Les tables seront alimentées à partir de fichiers csv. Vous mettrez tous les fichier csv utilisés dans un répertoire nommé CSV afin de ne pas les mélanger avec les scripts sql. Vous pourrez alimenter vos tables avec la technique qui consiste à :

— importer les fichier csv avec COPY dans des tables temporaires,

— « oublier » les attributs inutiles en supprimant les colonnes via `ALTER TABLE ... DROP COLUMN`,

— alimenter vos tables à partir de ces tables temporaires, par exemple former des couples nom, prénom à partir de deux tables, une avec des prénoms et une autres avec des noms.

Bien sûr le travail le plus ingrat consiste à produire les fichiers csv avec les données. Vous pouvez par exemple écrire dans votre langage de programmation préféré (Java, Python, ...) une boucle for qui affiche des tuples générés automatiquement (avec la syntaxe attendue par PostgreSQL), et les recopier ensuite dans votre code PostgreSQL.

Vous pouvez aussi utiliser directement les structures de contrôle intégrées à PostgreSQL :

`https://docs.postgresql.fr/current/plpgsql-control-structures.html`

Vous pouvez également utiliser des outils de génération de données en ligne tels que :

— `https://www.mockaroo.com/`,

— `https://generatedata.com/`,

— etc.

Les plus courageux peuvent récupérer des données réelles. Vous pouvez par exemple consulter :

— `https://www.kaggle.com/datasets?search=music`

— `https://www.kaggle.com/datasets?search=concert`

— `https://www.kaggle.com/search?q=spotify`

— `https://data.world/datasets/music`

— etc.

Attention, il est irréaliste de trouver sur internet des fichiers parfaitement prêts à être utilisés tels quels. La récupération et le nettoyage de vraies données peut être un exercice fastidieux et se fait d'habitude au moyen de librairies spécialisées (R et Python sont les languages typiquement utilisés pour ces tâches, voir par exemple `https://www.simplilearn.com/tutorials/data-analytics-tutorial/spotify-data-analysis-project`). Attention notamment au format des données. Si vous souhaitez vraiment récupérer de vraies données, le plus simple sera probablement de récupérer des données au format csv.

3 Effectuez des requêtes

Il est maintenant temps d'utiliser votre base de données pour faire vivre votre réseau social. Imaginez 20 questions sur la base de données que vous avez modélisée, et écrivez des requêtes SQL permettant d'y répondre. L'originalité des questions et la difficulté des requêtes (si tant est que celle-ci soit nécessaire) seront prises en compte dans la notation. Parmi vos requêtes, il faut au minimum :

— une requête qui porte sur au moins trois tables ;

— une 'auto jointure' ou 'jointure réflexive' (jointure de deux copies d'une même table)

— une sous-requête corrélée ;

— une sous-requête dans le FROM ;

— une sous-requête dans le WHERE ;

— deux agrégats nécessitant GROUP BY et HAVING ;

— une requête impliquant le calcul de deux agrégats (par exemple, les moyennes d'un ensemble de maximums) ;

— une jointure externe (LEFT JOIN, RIGHT JOIN ou FULL JOIN) ;

— deux requêtes équivalentes exprimant une condition de totalité, l'une avec des sous requêtes corrélées et l'autre avec de l'agrégation ;

— deux requêtes qui renverraient le même résultat si vos tables ne contenaient pas de nulls, mais qui renvoient des résultats différents ici (vos données devront donc contenir quelques nulls), vous proposerez également de petites modifications de vos requêtes (dans l'esprit de ce qui sera présenté dans le cours sur l'information incomplète) afin qu'elles retournent le même résultat ;

— une requête récursive (par exemple, une requête permettant de calculer quel est le prochain jour off d'un groupe actuellement en tournée) ;
Exemple : Napalm Death est actuellement en tournée (Campagne for Musical Destruction 2023), ils jouent sans interruption du 28/02 au 05/03, mais ils ont un jour off le 06/03 entre Utrecht (05/03) et Bristol (07/03). En supposant qu'on est aujourd'hui le 28/02, je souhaite connaître leur prochain jour off, qui est donc le 06/03.

— une requête utilisant du fenêtrage (par exemple, pour chaque mois de 2022, les dix groupes dont les concerts ont eu le plus de succès ce mois-ci, en termes de nombre d'utilisateurs ayant indiqué souhaiter y participer).

Prenez l'habitude de bien programmer vos requêtes SQL. Structurez vos requêtes et surtout indentez-les correctement. N'utilisez pas les sous-requêtes là où une jointure suffirait. N'utilisez pas les vues si c'est uniquement dans le but de simplifier l'écriture d'une requête complexe. Enfin, vous pouvez essayer de trouver une requête qui vous semble difficile (ou impossible) à effectuer dans votre modèle, et suggérer une modification de votre modélisation initiale qui faciliterait l'implémentation de cette requête.

4 Conseils pour la présentation

Pour rendre votre projet plus interactif, vous pouvez préparer des requêtes dont certains paramètres seront fournis par l'utilisateur. Par exemple, afin de retourner les lieux potentiels dans lesquels on peut organiser des concerts dans une ville donnée, on pourrait considérer la requête suivante :

```
SELECT lieu
FROM concert
WHERE ville='Paris';
```

Mais comment faire si la ville n'est pas connue d'avance ?

Préparation de requêtes paramétrées Vous pouvez créer une requête paramétrée en utilisant la syntaxe suivante.

```
PREPARE recherche_par_ville(VARCHAR) as
SELECT lieu
FROM concert
WHERE ville=$1;
```

Vous pouvez ajouter autant de paramètres que vous souhaitez. Dans la requête, vous utiliserez $1, $2... pour en obtenir la valeur. La requête est préparée mais pas exécutée. Pour l'exécuter, on tape sous `psql` :

```
EXECUTE recherche_par_ville('Marseille');
```

La requête est alors exécutée avec la valeur 'Marseille' à la place du paramètre $1.

Variables d'environnement et prompt La commande `prompt` de `psql` permet de faire la saisie au clavier. La valeur est sauvegardée dans une variable d'environnement (`v_ville` dans l'exemple qui suit).

```
\prompt 'Tapez le nom de ville -> ' v_ville
SELECT lieu
  FROM concert
 WHERE ville = :v_ville;
```

Lorsque la requête est exécutée, `:v_ville` prend la valeur de la variable d'environnement `v_ville`.

Tous ces éléments permettent de faire un script interactif que vous exécuterez au cours de votre présentation.

Manipulation des dates Le type Postgres `DATE` permet de stocker une date et le type `TIME` permet de stocker l'heure. Il existe également un type `TIMESTAMP` qui contient la date et l'heure.

Une chaîne peut être convertie en type `DATE` avec la fonction `to_date(chaine_a_convertir, format)`, où `format` est une chaîne de la forme `'YYYYMMDD'` qui indique le format dans lequel est donnée la date. On peut aussi écrire `date '2015-03-05'` pour obtenir une date. *Attention, selon la configuration de postgres, la date '12-11-2015' peut être interprétée comme le 12 novembre, ou le 11 décembre.*

On peut ajouter un entier n à une date pour obtenir la date n jours plus tard, ou soustraire une date à une autre pour connaître le nombre de jours entre les deux dates.

Plusieurs fonctions prédéfinies en Postgres permettent de manipuler les données de type `DATE, TIME, TIMESTAMP`.

Fonction	Usage	Exemple
`EXTRACT`	extraire un jour/mois/... d'une date ou d'un timestamp	`EXTRACT (DAY FROM madate)`
`INTERVAL`	préciser une unité de temps (jour/heure/semaine...)	`heureRDV + INTERVAL '2 hours'`
`CURRENT_DATE`	Obtenir la date du jour	
`CURRENT_TIME`	Obtenir l'heure actuelle	

On peut énumérer des valeurs dans un intervalle avec la fonction `generate_series`. Par exemple, `generate_series(1,5)` énumère les valeurs 1, 2, 3, 4, 5. `generate_series(1,5,2)` énumère les valeurs de 1 à 5 avec un pas de 2 : 1, 3, 5. `generate_series(current_date+time '10:00', current_date+time '16:00' , interval '30 minutes')` énumère les créneaux de 30 minutes aujourd'hui entre 10h et 16h.