Base de données L3 : projet

Système de réservation

À rendre par binôme le 4 mai 2015.

Modalités pratiques :

- le projet est à faire en binôme ;
- **une pré-soutenance aura lieu le mercredi 1er avril de 15h30 à 18h30 pour valider votre modélisation** ;
- ne commencez pas le travail SQL avant d'obtenir la validation de votre modélisation ;
- une grande liberté vous est laissée dans le sujet : chaque binôme devra donc avoir un projet personnalisé ;
- un rapport au format pdf ainsi que votre code SQL doivent être remis sur Didel le **4 mai 18h00** au plus tard. Aucun retard ne sera accepté. En cas de force majeure contacter votre chargé de TD le plus tôt possible ;
- les modalités d'inscription pour les pré-soutenances et les soutenances seront détaillées sur Didel en temps utile.

Politique concernant le plagiat. Vous pouvez bien sûr discuter des différents aspects du travail avec d'autres étudiants mais il est strictement interdit de copier ou vous approprier en partie ou intégralement du code, des données, du texte, des figures des graphes, ou toute autre composante du projet, quelle qu'en soit la source. Si vous êtes amenés à incorporer des éléments provenant de sources extérieures, vous devez obligatoirement en citer la source. Votre travail sera soumis à un détecteur de plagiat et tout écart sera sanctionné.

Description

L'objectif de ce projet est de modéliser puis de créer une base de données permettant la gestion d'un système de réservations. Les réservations peuvent concerner, à votre guise :

- des restaurants ;
- des salles de concert ou de théâtre ;
- des installations sportives (gymnases, stades, terrains de tennis, etc.) ;
- des professions libérales (médecins, coiffeurs, garagistes, etc.) ;
- tout autre lieu de votre choix à condition que les réservations aient des prix variables et des durées variables en fonction du type de prestation.

Vous pouvez demander à votre chargé de TD si vous voulez vérifier que votre choix convient.

Fonctionnalités

Grâce à votre base de données, un utilisateur devra pouvoir, au minimum :
- effectuer une recherche de disponibilité selon certains critères (date, heure, tarif, etc.) ;
- effectuer la réservation d'un créneau libre pour un certain nombre de personnes ;
- obtenir des réductions à partir d'un certain nombre de réservations effectuées en moins d'un an ;
- annuler et éventuellement se faire rembourser si la réservation est suffisamment éloignée dans le temps.

L'administrateur devra pouvoir, au minimum :
- effectuer des statistiques sur l'occupation du lieu ;
- visualiser les recettes ;
- supprimer un utilisateur ou une réservation ;
- annuler une représentation ou une journée.

Vous donnerez au moins une requête SQL pour illustrer chacun des points ci-dessus.

Modélisation et tables

Votre modélisation doit être suffisamment riche pour être réaliste et donner lieu à des requêtes suffisamment complexes, mais pour la partie SQL, il faut éviter de trop compliquer, quitte à faire des hypothèses simplificatrices et à n'implémenter qu'un sous-ensemble de votre modélisation. Pour donner un ordre de grandeur, le nombre de tables en résultant doit être compris entre 5 et 12.

Identifiez un maximum de contraintes d'intégrité et de règles de gestion au moment de la modélisation. On pourra vous questionner à ce sujet lors de la soutenance. Intégrez un maximum de contraintes lors de la création des tables.

Les tables devront être peuplées par une quantité significative de données pertinentes permettant d'apprécier les résultats des requêtes — volume indicatif : le nombre total de lignes (somme sur toutes les tables) pourra être de l'ordre de 100. Ne perdez pas trop de temps à peupler vos tables avec des données réelles, ce n'est pas l'objectif de ce projet.

Requêtes

Requêtes imposées

En plus des fonctionnalités demandées, vous devez donner les requêtes SQL permettant de répondre aux questions suivantes.

1. Quel a été le taux d'occupation d'un lieu en 2014 ?
2. Donner les recettes du mois en cours, par type de prestation, en ordre croissant de recettes.
3. Donner la liste des prestations n'ayant fait l'objet d'aucune réservation le mois dernier.
4. Quelle a été la journée la plus occupée de chaque semaine, pour les 8 dernières semaines ? Comptez la durée des réservations et non pas le nombre de réservations par jour.
5. En moyenne, combien de jours à l'avance les clients font-ils une réservation ?
6. Pour chaque mois de 2014, identifiez le client ayant le plus dépensé sur le site.
7. Pour chaque trimestre de 2014, calculer le pourcentage — sur le chiffre d'affaires total — représenté par les recettes générées par un certain type de prestation. Attention à tenir compte des trimestres lors desquels cette prestation n'aurait généré aucune recette.
8. Quels sont les clients qui sont venus au moins une fois par semaine ces six derniers mois ?
9. Lorsqu'un client quelconque prend des prestations le même jour, certaines se retrouvent toujours dans le même ordre. Ainsi dans le cas d'un hôpital, si un patient doit subir une anesthésie et une opération, elles sont toujours programmées dans cet ordre-là. Retrouver les paires de prestations qui ont ce genre de dépendances.
10. On remarque que des clients viennent toujours ensemble. Écrivez une requête qui associe à chaque client les autres clients qui viennent systématiquement les mêmes jours qu'eux.

Requêtes personnelles

Imaginez 10 questions sur le système de réservation que vous avez modélisé, et écrivez les requêtes SQL pour 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 les 10 requêtes, il faut au minimum :

— que toutes les requêtes portent sur au moins deux tables ;
— une requête qui porte sur au moins trois tables ;
— une sous-requête corrélée ;
— une sous-requête dans le SELECT ;
— une sous-requête dans le FROM ;
— une sous-requête dans le WHERE ;
— deux agrégats nécessitant GROUP BY et HAVING ;
— une jointure externe (LEFT JOIN ou RIGHT JOIN) ;

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 pour éviter d'écrire des requêtes complexes. Utilisez les USING uniquement là ou c'est justifié.

Rapport et soutenance

La **pré-soutenance** durera entre 5 et 10 minutes. Son but est de valider votre modélisation. Le seul document demandé est donc le modèle conceptuel de données (schéma entités-relations). Déposez celui-ci sur Didel avant votre soutenance et amenez-en une copie sur papier. L'inscription sur Didel aux pré-soutenances est obligatoire. Les modalités seront données sur Didel dans la semaine qui précède la date de pré-soutenance.

Ne commencez pas votre code SQL avant d'avoir obtenu la validation de votre modèle.

Votre code doit s'exécuter *sans aucune modification* sur la version postgres installée sur la machine nivose. Si vous travaillez sur une machine personnelle il vous incombe de faire les modifications nécessaires avant de rendre le projet.

Le **rapport** final devra contenir :

— votre modèle entités-relations ;
— une explication des choix que vous avez été amenés à faire lors de la modélisation ;
— un schéma relationnel de vos tables ;
— une description des requêtes proposées ;
— toute explication nécessaire à la compréhension de votre projet. Si vous faites un système de réservation de schmilblick, il est impératif d'expliquer brièvement les particularités des schmilblick qui sont essentielles à la compréhension de votre modélisation et de vos requêtes.

Vous devez rendre sous Didel tout ce qui est nécessaire pour exécuter votre projet, à savoir :

— le script de création de vos tables ;
— les scripts contenant vos requêtes ;
— un bref descriptif (fichier README) expliquant le contenu de chaque fichier.

La **soutenance** finale aura lieu la semaine du 4 mai (date exacte communiquée ultérieurement). Vous débuterez celle-ci par une courte présentation de votre projet (5 minutes, cf. ci-dessous pour des conseils). Vous apporterez :

— une copie du rapport ;
— le code de vos requêtes, avec pour chacune un commentaire expliquant ce qu'elle fait.

Pour la soutenance, soyez prêts à répondre à des questions concernant, entre autres, les contraintes d'intégrité, les choix de modélisation, les points forts et les points faibles du travail réalisé, les parties incomplètes, les améliorations qui restent à faire, etc.

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, si vous faites une recherche de restaurants par ville, vous pourriez avoir une requête comme suit :

```
SELECT nom
  FROM restaurant
 WHERE ville='Cancale';
```

Mais comment faire si vous souhaitez faire une requête dont 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 nom
  FROM restaurant
 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('Cancale');
```

La requête est alors exécutée avec la valeur 'Cancale' 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 nom
  FROM restaurant
 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.