Projet liste de tâches

Dans ce projet, il vous est demandé de réaliser la base de données pour une application de gestion de listes de tâches (« to-do » list). Cette base de données permettra aux utilisateurs de définir et donc de gérer efficacement leurs tâches. Plusieurs tâches pourront appartenir à un même projet. Certaines tâches auront des dépendances : elles ne pourront pas être réalisées avant que les autres tâches dont elles dépendent aient été réalisées. Chaque tâche aura un intitulé. Chacune des tâches pourra avoir une priorité, une catégorie, ou un lien externe (vers un document web ou un git par exemple).

Les informations suivantes sont conservées pour chaque utilisateur : son nom, son prénom, son adresse et sa date de naissance. La date d'inscription est enregistrée. Un login et un mot de passe sont demandés à l'inscription. Le login est composé de la première lettre du prénom et des `7` premières lettres du nom (en minuscules) suivies de deux chiffres. Les logins de plusieurs adhérents différents ne peuvent être identiques. Le mot de passe ne peut contenir que des lettres (minuscules/majuscules), des chiffres et le caractère '_'.

Les utilisateurs peuvent créer des tâches, leur assigner des descriptions, des dates d'échéance, des niveaux de priorité et des statuts (terminée ou non terminée). Ils peuvent également ajouter un lien externe vers une ressource pertinente telle qu'un document en ligne ou un sites web. Des dépendances entre les tâches peuvent être indiquées, indiquant ainsi que certaines tâches doivent être accomplies avant que d'autres ne puissent commencer. Certaines tâches seront périodiques (par exemple, des tâches à réaliser chaque jour, chaque semaine, etc.). Il sera possible d'indiquer pour les tâches périodiques ma date de début, la date de fin (si elle existe) et la période (toutes jours, toutes les semaines, etc.). Les utilisateurs pourront visualiser leur régularité, c'est-à-dire la proportion des taches qu'ils ont accompli sur une semaine par rapport à la proportion totale des tâches prévues sur la même période.

Dans une volonté de gamification, un score et un niveau sera attribué aux utilisateurs. Le score sera calculé d'après la proportion les tâches réalisées/non réalisées. Un programme de score pourra être défini, indiquant le nombre de point ajouté/retiré à chaque tâche terminée/non terminée pour la date d'échéance. Pour passer un niveau, il faudra atteindre un score de $(100 * 2^{(\text{niveau} - 1)})$.

Des listes de tâches permettent aux utilisateurs d'organiser leurs tâches en fonction de projets spécifiques. Chaque liste de tâches est associée à un utilisateur créateur. Plusieurs utilisateurs peuvent être assignés à une même tâche.

Les tâches devant être réalisées et les tâches passées (déjà réalisée ou dont l'échéance est expirée) seront stockées dans des tables différentes. Certains traitements afficheront ou seront exécutées sur toutes les tâches (tâches à réaliser ou passées).

Des requêtes `SQL`, des procédures `PL/SQL` et des contraintes d'intégrité seront indiquées dans la suite de projet. Elles devront être prises en compte dans la modélisation de la base de données

Partie 1, à faire individuellement (date limite : 6 novembre)

Modélisation

Dessiner le modèle entité-association de la base de données pour la gestion de liste de tâches en respectant au maximum les besoins fonctionnels indiqués dans l'énoncé.

Préciser les contraintes d'intégrité sur le modèle (contraintes que les données devront respecter à tout instant).
Les contraintes sont indiquées en version texte (paragraphe contenant la liste des contraintes qui seront à implémenter, c'est à dire à exprimer en `SQL`)

Établir le modèle logique relationnel de la base (l'ensemble des relations déduites du modèle entité-association).
Une relation sera de la forme `R(X1 : T1, X2 : T2, …, XN : TN)` avec `R` nom de la relation, `Xi` les noms des attributs et `Ti` les types abstraits (par exemple : `String` plutôt que `Varchar`). Ne pas oublier de préciser les clés primaires/clés étrangères.

Un rapport devra être remis contenant le schéma E/A, la liste des contraintes d'intégrité ainsi que le modèle relationnel.

Implémentation de la base de données

Les scripts `SQL` de création des tables, de suppression des tables et d'insertion de données de tests devront être remis. Les scripts `SQL` pourront aussi contenir des commandes de création/suppression de vues.

Les contraintes d'intégrités statiques (pouvant être implémentées sans utiliser de `PL/SQL`) seront aussi exprimées en `SQL` dans les scripts de création de tables.

Un script permettra d'exécuter tous les autres scripts afin de, si nécessaire, recréer les tables et de remplir les tables.

Rendu de la partie 1 : Un rapport, un modèle E/A, un modèle relationnel, les scripts `SQL` de création/suppression des tables/vues avec contraintes statiques et les scripts d'insertion de données.

Partie 2 (à faire en binôme ou individuellement, date limite : 23 décembre)

Requêtes SQL

Ecrire les requêtes permettant d'obtenir les résultats suivants :

1. Les listes de tâches ayant au moins 5 tâches et appartenant à des utilisateurs habitant en France.

2. Les programmes de tâche ayant le plus de points positifs (somme des points) associés aux tâches terminées.

3. Pour chaque utilisateur, son login, son nom, son prénom, son adresse, son nombre de tâches total (périodique et non-périodique) et son nombre de tâches périodiques total.

4. Pour chaque tâche, le nombre de dépendance à effectuer avant que la tâche puisse être réalisée.

5. Les 10 utilisateurs ayant gagné le plus de points sur leur score au cours de la semaine courante.

Définir les index pour optimiser vos requêtes.

Procédures et fonctions PL/SQL

1. Définir une fonction qui calcule le nombre de point gagné/perdu (pour les utilisateurs ayant un programme de score, en fonction du nombre de tâche terminée/non terminée) au cours de la semaine.

2. On supposera que la procédure est exécutée chaque semaine (le lundi, à 8h). Définir une procédure qui archive toutes les tâches passées.

3. Ecrire le code PL/SQL permettant de générer des suggestions pour un utilisateur, c'est dire N tâches qu'il pourrait ajouter à sa liste de tâche. Les tâches suggérées seront les N tâches apparaissant le plus grand nombre de fois dans les tâches des utilisateurs similaires. Les utilisateurs similaires sont les utilisateurs ayant au moins X tâches similaires avec l'utilisateur pour lequel effectuer les suggestions. Des tâches similaires sont des tâches ayant au moins Y mots communs (dans les mots communs, on ne compte pas les mots vides/stop words comme les articles ou les verbes peu significatifs type faire ou avoir).

Déclencheurs

Définir les déclencheurs associés à la base.

En particulier, définir les déclencheurs suivants :

1. La valeur de score sera stockée dans une table et mis à jour chaque fois qu'une tâche est terminée ou archivée.

2. Pour chaque tâche périodique avec une date de fin ajoutée ou modifiée, définir les tâches associée (tâche avec une date précise, par exemple une tache périodique réalisée tous les jours à 10h, dont la fin est prévue dans une semaine provoquera la définition de 7 tâches, prévue sur 7 jours, à 8h).

Rendu du projet partie 2

Vous devez déposer sur Moodle (au plus tard le 16 décembre) :

- un court rapport expliquant les choix,
- les requêtes SQL,
- les définitions de contraintes d'intégrité (version texte),
- les scripts (fonctions, procédures, déclencheurs) PL/SQL,
- les index,
- si nécessaire définir les débuts/fins de transaction pour vos opérations,
- un script permettant de tester les procédures/fonction/déclencheurs.