TD

version 4 février 2015

Cahier des charges : Achat de séjours de vacances par des clients

Une agence de voyages propose à ses clients des séjours dans des villages de vacances, qu'ils peuvent choisir, et payer grâce à un avoir.

Données La base suivante décrit les informations correspondantes.

Client(idc, nom, age, avoir) Village(idv, ville, activite, prix, capacite) Séjour(ids, idc, idv, jour)

Un client a un identifiant, un nom, un âge, et un avoir (entier).

Un village de vacances a un identifiant, une ville, l'activité qu'on peut y pratiquer, un prix par jour (entier) et un nombre de lits (capacité).

Un séjour a un identifiant, un identifiant d'acheteur, un identifiant de village, et une date. Pour simplifier, chaque séjour a une durée d'un jour, et on considère que tout à lieu dans l'année courante (non bissextile), une date sera donc simplement un jour : un entier entre 1 et 365.

Exemple *Riton* a *22* ans, et un avoir de *1700* euros. Il y a un village à *Rio*, de *250* lits, coût par jour *50* euros, où l'on pratique le *kitesurf*. Le client *2* a acheté un séjour dans le centre *11* pour le *31 décembre*.

Client(1, Riton, 23, 1700) *Village(10, Rio, kitesurf, 50, 250)* *Séjour(100, 2, 11, 365)*

Précisions et restrictions Les identifiants (entiers) sont uniques. Plusieurs clients peuvent avoir le même nom. L'agence fait un cadeau de bienvenue aux nouveaux clients, qui, en s'inscrivant, se voient créer automatiquement un avoir de 2000 euros, mais ensuite les avoirs négatifs ne sont pas autorisés. Plusieurs villages peuvent être dans la même ville. Il n'y a qu'une activité par village. Un client ne peut acheter un séjour que pour lui-même, donc un seul pour une date donnée. Il peut y avoir d'autres objets SGBD dans la base (tables, contraintes, etc.), mais on ne les connaît pas.

Accès aux données Les employés ont des comptes SGBD sur la base (les logins ne sont pas les noms), et il y a un unique compte client. Un employé peut créer des villages, les consulter et les modifier (sauf identifiant, ville et prix), consulter les séjours, et effectuer le traitement 3. Un client peut effectuer les traitements 1 et 2, consulter les villages pour lesquels il n'y a aucun séjour (mais sans la capacité), ainsi que consulter les lignes lui correspondant dans chaque table. Son nom et son identifiant suffisent comme authentification. Rien d'autre n'est possible.

Fonctionnement Les traitements ne vérifieront pas les contraintes, ni si les paramètres sont corrects.
Traitement 1 : L'inscription d'un client est faite par lui-même comme suit. Il donne son nom et son âge, une nouvelle ligne est alors créée dans *Client*, et il obtient son identifiant. Paramètre(s) : nom et âge. Retour(s) : identifiant client.
Traitement 2 : L'achat d'un séjour est fait par un client comme suit. Il arrive avec en tête une ville et un jour. Un village dans cette ville de prix journalier le plus cher est alors cherché (car plus le prix est élevé, meilleur est le village, et l'activité n'est pas prioritaire). S'il y en plusieurs l'un quelconque est choisi (par exemple le premier qui vient). Puis le séjour correspondant est créé et l'avoir du client décrémenté du prix journalier du village. S'il n'y en a aucun on ne fait rien. Paramètre(s) : identifiant client, ville et jour. Retour(s) : identifiants village et séjour, et activité ; sinon *-1, -1, néant* si pas de village dans cette ville.
Traitement 3 : La maintenance des séjours est faite par les employés comme suit. Un employé peut détruire tous les séjours de date strictement inférieure à une date donnée. Paramètre(s) : jour. Retour(s) : le nombre de séjours détruits.
Traitement 4 : L'archivage d'un séjour est effectué automatiquement (pas par les employés ni les clients) en cas de destruction, avec l'avoir du client à ce moment. Paramètre(s) : néant. Retour(s) : néant.

1 Mises à jour et interrogation

1. Donnez la partie mise à jour et interrogation complète de l'application (en mode interactif). (On ne traitera pas aujourd'hui dans cet exercice les autres problèmes BD : contraintes, etc.)

   Indication : procédez comme suit.

   (a) Donnez pour cette application la liste des actions du programmeur, et la liste des actions des autres acteurs (utilisateurs).

   (b) Donnez pour chaque action, en pseudo-SQL, le ou les ordres correspondants sur un exemple, en précisant le cas échéant les "paramètres" et leurs occurences, ainsi que les retours ("modèle d'ordre").

   (c) Donnez en SQL l'appplication complète, en indiquant pour chaque ordre quel acteur le tape et sur quel compte.

   (d) Donnez et exécutez plusieurs scénarios (exemples) de déroulement de l'application. (Mais pour simplifier aujourd'hui c'est vous qui taperez tous les ordres de tous les acteurs.)

2 Contraintes

1. On considère les contraintes naturelles sur le schéma du cahier des charges. Donnez les ordres SQL (en pseudo-SQL ou en SQL) pour les gérer quand c'est possible, et sinon énoncez-les rigoureusement en fonction des lignes et des colonnes des tables (en français ou de la manière de votre choix, n'hésitez pas à questionner un enseignant).
   Indications : il y a environ 19 contraintes SQL et 2 non SQL.
   Dans le reste de l'épreuve, on ignore cet exercice et on se replace dans la situation du cahier des charges.

   Indication : procédez comme suit.

   (a) Contraintes SQL. Pour cette question, considérez chaque colonne de chaque table, pour chaque type de contrainte SQL, et donnez à chaque fois la (ou les) contrainte(s) correspondante(s) en pseudo-SQL. Plus précisément, pour chaque colonne de chaque table, dites si c'est une clé primaire. Puis faites de même pour chaque autre contrainte parmi *fk, (not) null, unique, check*. Pour *unique*, considérez aussi tous les ensembles de colonnes à l'intérieur de chaque table (éventuellement aussi pour *pk* et *fk*).

   (b) Contraintes non SQL. Pour cette question, considérez chaque colonne de chaque table, par rapport à toutes les autres colonnes de chaque table. Enoncez rigoureusement les contraintes non SQL en français (ou formellement), en fonction des lignes et des colonnes des tables. Montrez qu'il existe une contrainte supplémentaire si on ignore le traitement 3.

   (c) Etendez votre application complète de l'exercice précédent en y intégrant les ordres SQL correspondant à la question (a) ci-dessus, et testez-la. Testez une violation de chaque contrainte. (Pour simplifier aujourd'hui c'est vous qui taperez tous les ordres de tous les acteurs.) (La gestion des contraintes non SQL sera vue ultérieurement.)