L3 MIAGE – 2018/2019

TD/TP – Bases de données

février 2019

Une chaîne de restauration rapide, dénommée Rapi'do, informatise ses magasins, afin d'avoir quelques statistiques sur le comportement de sa clientèle. Cette chaîne possède plusieurs magasins en France, vendant tous les mêmes produits, aux mêmes prix. Chaque magasin possède une référence, un nom. On retiendra également la ville où il se situe. Comme il s'agit de restauration rapide, les produits vendus ne sont pas très variés. On les découpe en deux grands ensembles: les produits simples (boissons, sandwiches, ...) et les menus (qui sont constitués de produits simples). Chaque produit a une référence (identifiant numérique), un nom unique, un prix et une taille (petit, moyen ou grand). De plus, les produits simples sont répertoriés par catégories. On se limitera aux catégories suivantes : boisson, dessert, salade, accompagnement et sandwich. Les menus peuvent être liés à une promotion. Par exemple pour tout menu "boîte magique" on donne un jouet pour enfant. Ainsi, la promotion liée à ce menu sera "jouet enfant". On peut imaginer des promotions diverses : une place de cinéma à tarif réduit, un CD ...

Lors de la conception de cette base de données on ne s'intéresse pas aux clients mais simplement aux consommations dans les différents magasins. On mémorise le jour et l'heure de chaque consommation (par exemple, jour et heure du ticket de caisse).

A faire en TD

Question 1 : Proposer un diagramme de classe UML pour modéliser ces données.

Question 2 : En déduire les ordres de création des tables SQL qui traduisent ce diagramme de classes.

A faire en TD/TP

En TP, vous trouverez sur Moodle un script de création des tables. Chaque procédure PL/SQL devra être testée avec DBFit.

Question 3 : Créer un paquetage stocké `PAQ_PRODUITS` permettant

- d'ajouter et de supprimer des produits simples et des menus.
  - Pour les créations de nouveaux produits, les clés primaires seront toutes générées à partir d'une séquence.
  - un nom de produit qui viole la contrainte d'unicité déclenchera l'exception `DOUBLON_NOM_PRODUIT`.
  - si les contraintes de domaines définies sur les tables ne sont pas vérifiées, on déclenchera selon les cas `PB_VALEUR_TAILLE`, `PB_VALEUR_CATEGORIE` ou `PRIX_NON_POSITIF`.
  - Lorsqu'on supprime un menu, on supprime aussi de la table `FF_CONSTITUE` toutes les lignes qui le concernent.
  - Par contre, vouloir supprimer un produit simple qui entre dans la composition d'au moins un menu entraine `PRODUIT_UTILISE`
  - Supprimer un produit simple (resp. un menu) qui n'est pas en base déclenche l'exception `PRODUIT_INCONNU`.
- d'ajouter et de supprimer des produits simples dans la composition d'un menu.

- Lorsqu'on ajoute un produit simple dans la composition d'un menu, il faudra vérifier qu'un menu de taille t n'est constitué que de produits simples de taille t, sinon on déclenche l'exception `PB_COHERENCE_TAILLES`.
- Dès que l'on fait référence à un identifiant de produit simple (resp. de menu) qui n'est pas en base on déclenche l'exception `PRODUIT_INCONNU`.
- Vouloir supprimer un produit simple de la composition d'un menu alors qu'il n'en fait pas partie déclenche l'exception `PB_COMPOSITION`.
- Vouloir ajouter un produit simple à la composition d'un menu qui le contient déjà ne déclenche aucune erreur.

Toutes ces procédures déclenchent `PARAMETRE_INDEFINI` dès que l'un de leur paramètre vaut `NULL`.

Vous trouverez les tests DBFit de ce paquetage sur Moodle, archive `Rapido.zip` : vous devez dézipper ce répertoire `Rapido` et le mettre dans votre répertoire `FitNesseRoot`. Les tests seront alors accessibles à l'url `http://localhost:8085/Rapido`.

Pour les questions suivantes, vous compléterez la suite de tests `Rapido` avec vos propres tests.

Question 4 : Créer une procédure stockée `conso` permettant d'ajouter une consommation en passant en paramètre la référence du produit, la référence du magasin et éventuellement une estampille. Si ce troisième paramètre n'est pas fourni, on utilisera la valeur de l'estampille système (sous oracle, elle est donnée par la fonction `sysdate`).

On veut maintenant ajouter une gestion de stocks :

Question 5 : Définir une table `STOCK` permettant de mémoriser la quantité en stock (même 0) de chaque produit simple pour chaque magasin.

Question 6 : On veut être certain que la table `STOCK` contient 1 ligne par couple (produit, magasin). Ecrire des triggers qui assurent que ça reste vrai, même lorsqu'on crée un produit simple ou un magasin.

Question 7 : Définir un paquetage permettant d'ajouter des produits (produits simples ou menus) dans le stock d'un magasin. Ajouter un menu signifie ajouter les produits simples constituant ce menu.

Question 8 : Ecrire un ou plusieurs triggers permettant de diminuer le stock conformément aux consommations des produits. Evidemment on ne peut pas consommer de produits plus qu'il n'y en a en stock.