Examen SI70 – Conception des systèmes d'information

Les différentes parties concernent le même thème mais peuvent être réalisées indépendamment – Le barème fourni est indicatif – Une feuille de note manuscrite et nominative recto-verso format A4 autorisée

De nos jours, les gens préfèrent consommer des produits alimentaires produits localement. La société LOCAVORE s'est constituée en vue de répondre à cette demande et souhaite mettre en relation les clients (acheteurs) et les producteurs locaux.

La société LOCAVORE désire permettre aux acheteurs de :

- Composer un panier de produits locaux sur un site internet et passer commande en ligne
- Livrer les produits à domicile

Quant aux producteurs, ceux-ci doivent pouvoir indiquer sur le site quels sont les produits qu'ils proposent.

La société LOCAVORE assure l'intégralité du transport en récupérant les produits auprès des différents producteurs et en les acheminant auprès des clients. Elle gère aussi le paiement en prélevant les clients du montant de la commande et en reversant l'argent aux producteurs (naturellement une commission est prélevée par la société sur le montant de chaque commande).

Modélisation d'un système d'information (7 points)

En vue d'assurer la livraison des colis aux clients, la société dispose d'une flotte de véhicules effectuant des livraisons. Lors d'une tournée de livraison, chaque véhicule passe d'abord collecter les produits auprès des producteurs avant de livrer les colis aux clients.

Un véhicule est identifié par son numéro d'immatriculation, il est caractérisé par la marque du constructeur, son modèle, sa date d'acquisition et sa date de première immatriculation.

- Tournée de livraison

Lors d'une tournée, un véhicule est associé à une liste de commandes. Un véhicule peut effectuer plusieurs tournées par jour et chaque tournée est caractérisée par sa date et son heure de début, le véhicule affecté et les différentes commandes associées. Chaque commande correspond à un client et à une liste de produits ; on connaît également la date et l'heure de livraison prévues. Chaque produit est caractérisé par son nom et est proposé par un fournisseur unique. Les clients, tout comme les fournisseurs, sont caractérisés par leur nom et leur adresse.

- Maintenance des véhicules

Les véhicules subissent à échéance régulière un contrôle technique et un entretien périodique (aussi appelé révision). Les contrôles techniques sont effectués indépendamment pour chaque véhicule. Un contrôle technique est caractérisé par sa date et son résultat (soit "OK", soit "Contre-visite"). Les révisions quant à elle sont réalisées pour plusieurs véhicules simultanément. Chaque révision est caractérisée par sa date de réalisation, l'établissement où elle a eu lieu, et la liste des véhicules concernés. Lors d'une révision, chaque véhicule est associé à une liste d'opérations (par exemple,

“vidange”, “changement filtre à air”, etc.) et chaque opération est associée à un prix global (comprenant pièce et main d’œuvre).

- **Proposez un diagramme entité-association des données du domaine “tournée de livraison”**
- **Proposez un diagramme entité-association des données du domaine "maintenance des véhicules”**
- **Transcrivez le diagramme entité-association du domaine “maintenance des véhicules” en un schéma relationnel de base présentant les différentes tables nécessaires (vous préciserez les clés primaires et clés étrangères)**

Exploitation d’une base de données relationnelles (9 points)

Le modèle relationnel suivant présente la structure de la base de données liées aux différents produits proposés par les producteurs et pour lesquels les clients peuvent passer commande. Il présente également les informations concernant les ventes. Ce modèle est utilisé pour la gestion de la société afin d’obtenir des informations concernant les produits actuellement proposés et obtenir des statistiques sur les ventes.

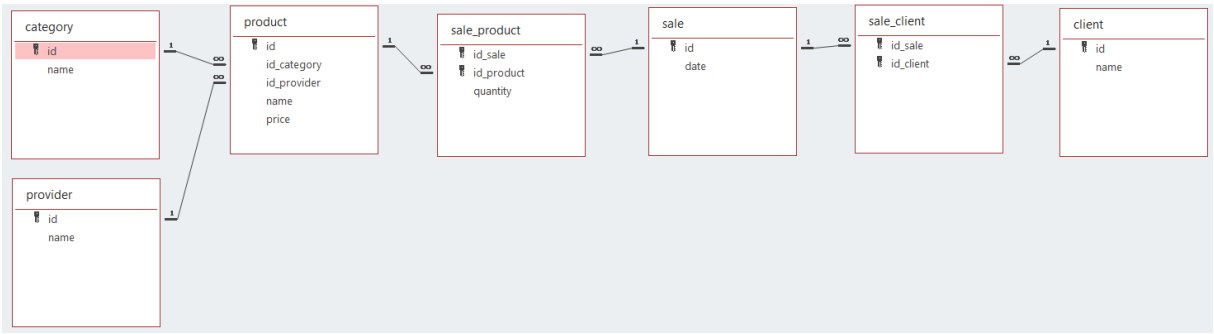

Figure 1 - Schéma relationnel de la partie de la base liées aux ventes

Analyse

A partir du schéma relationnel fourni (Figure 1**), produisez le diagramme entité-association correspondant.**

Manipulation de la base de données

- Ecrire la requête SQL permettant de créer la table *product* (les clés primaires et étrangères sont de type entier, *name* est de type chaîne de caractères, *price* est de type numérique)
- Ecrire la requête SQL permettant d’ajouter le champ *first_name* (type chaîne de caractères) à la table *client*

Manipulation des données

Écrivez les requêtes SQL permettant de réaliser les opérations suivantes :

- Récupérer la liste de tous les produits dont le prix est inférieur à 10€. La liste sera triée par prix décroissant.
- Récupérer la liste de tous les produits dont le nom de catégorie est « Fromage ». Le résultat ne contiendra que les noms et prix des produits.
- Récupérer le nom de tous les clients ayant passé commande le 01/02/2019.
- Récupérer le nom du fournisseur (*provider*) du produit s’étant le plus vendu (en nombre de ventes).

- Récupérer le chiffre d'affaire total du 30/03/2019.
- Afficher la liste des noms des fournisseurs et pour chacun d'entre eux, donner le chiffre d'affaire correspondant pour l'année 2018.
- Afficher le nom des clients ayant réalisé une commande contenant un produit du fournisseur dont le nom est « Laiterie Delavache » et un produit du fournisseur « Charcuterie Ducochon ».

Architecture applicative (4 points)

Le site internet fournissant les différentes fonctionnalités aux clients et producteurs est implémenté en Java en utilisant Spring Boot.

JPA (Java Persistence API) est utilisé en tant qu'ORM (Object Relational Mapping) pour accéder aux données de la base de données.

Une classe *MainController* expose différents points d'accès (URL) permettant d'accéder aux différentes pages et aux différentes fonctionnalités de l'application.

L'une de ces fonctionnalités est l'affichage de la liste des produits d'un producteur. Le code correspondant est le suivant :

```
@Controller
public class MainController {

	@Autowired
	private ProductRepository productRepository;

	...
	@GetMapping("/productsByProducer")
	public ModelAndView getProductsByProducer(@RequestParam Integer producerId) {
		final ModelAndView mv = new ModelAndView("products");
		final List<Product> products =
			this.productRepository.findByProducerId(producerId);

		mv.addObject("products", products);

		return mv;

	}
	...
}
```

L'interface *ProductRepository* est définie ainsi :

```
public interface ProductRepository extends JpaRepository<ProductEntity, Integer> {

	...
	List<ProductEntity> findByProducerId(Integer producerId);
	...
}
```

La page HTML suivante « products.html » utilise le moteur de templating Thymeleaf pour mettre en forme les données d'une liste de produits :

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Liste produits</title>
```

```
</head>
<body>

<div>
    Liste des produits :
</div>

<table>
    <tr th:each="p : ${products}">
        <td th:text="${p.name}"></td>
        <td th:text="${p.category.name}"></td>
        <td th:text="${p.price}"></td>
    </tr>
</table>
</body>
</html>
```

Écriture des entités

Pour que l'affichage de la page fonctionne, il est nécessaire de disposer de trois classes *ProductEntity, ProducerEntity* et *CategoryEntity* respectivement associées aux tables *product, producer* et *category*. Un produit n'appartient qu'à une catégorie mais une catégorie est associée à plusieurs produits.

En utilisant les annotations JPA, notamment *@Entity*, *@Id*, *@Column*, *@OneToOne*, *@ManyToOne*, *@OneToMany* ou *@ManyToMany*, écrivez le code de la classe *ProductEntity*. L'écriture des *imports* est ignorée. Il ne faut pas écrire plus d'attributs que nécessaires.

Sauvegarde d'une entité

Sans écrire la page HTML correspondante, écrivez la fonction du *MainController* permettant d'enregistrer une nouvelle catégorie de produits en base de données. La catégorie sera saisie par un formulaire en utilisant la méthode POST et l'URL « /newCategory ». Le seul paramètre demandé sera *name*, le nom de la catégorie.

On supposera que l'interface *CategoryRepository* ***extends*** *JpaRepository<Category, Integer>* a été définie, et que, pour rappel, celle-ci dispose d'une méthode *save(Category category)* permettant de sauvegarder une catégorie dans la base de données.