TD 1 : Fichiers de données

Exercice 1 - Dépendances fonctionnelles

1. Soient les attributs {num_train, gare, ville, horaire} pour la modélisation d'un réseau ferroviaire. Exprimer les dépendances fonctionnelles suivantes :

- "*Certaines villes possèdent plusieurs gares (mais la réciproque n'est pas vraie).*"
- "*Un train ne peut passer dans une même gare à deux horaires différents.*"

2. Soient les attributs {id_enseignant, num_salle, date, heure}. Exprimer les dépendances fonctionnelles suivantes :

- "*Un enseignant ne peut enseigner dans deux salles différentes pour le même créneau horaire (date et heure)*"
- "*Les séances sont assurées par un enseignant unique.*"

3. Soient les attributs {code_vol, aéroport_départ, aéroport_arrivée, date_heure_départ, durée, code_appareil} servant à décrire des vols affrétés par une compagnie aérienne . Exprimer les dépendances fonctionnelles :

- "*Le code du vol détermine le trajet.*"
- "*Le trajet détermine la durée*".
- "*Pour une date donnée, un seul appareil assure le vol*"
- "*Un même appareil ne peut être affrété pour deux vols partant au même moment.*"

Exercice 2

Soit l'ensemble d'attributs décrivant la commande d'un produit en une certaine quantité par un certain client:

E = {produit, quantité, prix_unitaire, montant, nom_client, prénom_client, téléphone, voie, ville, code_postal, pays, date, trimestre, mois, année}

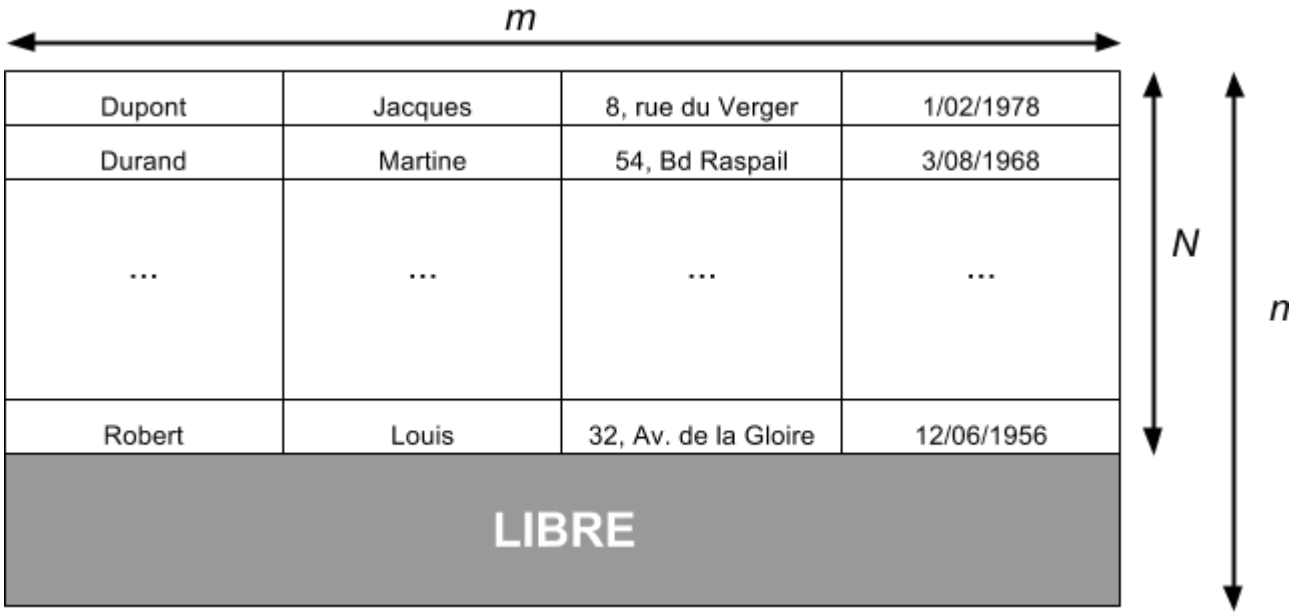
1. Essayez de trouver des dépendances fonctionnelles au sein de cet ensemble d'attributs.
2. A partir des dépendances définies précédemment, trouvez une clé du schéma de table "Commande" composé de l'ensemble de ces attributs :

Commande(produit, quantité, prix_unitaire, montant, nom_client, prénom_client, téléphone, voie, ville, code_postal, pays, date, trimestre, mois, année)

1. Cette table est-elle 2FN? 3FN? Indiquez les modifications à apporter pour obtenir un schéma normalisé.

Exercice 3 : page de données

On considère une zone mémoire de taille n x m octets, organisée sous la forme d'un tableau de données contenant des informations sur une liste de N clients (avec N < n). On parle de page de données. Chaque ligne correspond à un client différent (taille m).



1. Donnez la complexité pour les opérations suivantes:
    1. insertion d'un nouveau client
    2. recherche d'un client
    3. suppression d'un client
2. Donnez un algorithme permettant d'éviter l'insertion de doublons. Quelle est sa complexité?
3. Que faire pour accélérer les temps de recherche?
4. Que faire lorsque la page est pleine?

Exercice 4 : recherche par clé

On suppose maintenant que chaque tuple est désigné par sa « clé » k (entier). Un tuple est alors décrit par la série de valeurs (clé, information$_1$, ..., information$_m$) où la clé donne « accès » à l'information.

En pratique, 2 grandes familles de méthodes d'accès sélectives (recherche par clé):

- méthodes d'accès par index : utilisation d'une table pour mémoriser l'association clé / adresse tuple.
- méthodes d'accès par hachage : déterminent l'emplacement du tuple à partir d'une fonction de la clé,

1. **Recherche par index**
    1. Ecrire une fonction `cree_index` qui crée une liste associant à chaque clé le numéro de la case dans laquelle le tuple est enregistré.
    2. Quel est le temps nécessaire pour trier l'index sur les clés?
    3. Une fois l'index trié, quel est le temps nécessaire pour trouver un tuple dans le tableau à

partir de sa clé?
4. Quel problème se pose lors de l'ajout de nouveaux tuples dans le tableau?

2. **Table de hachage**. Une table de hachage est une méthode d'indexation "automatique" dans laquelle l'index est remplacé par une fonction qui "décide" dans quelle case chaque tuple doit être rangé : si k est la clé du tuple, la fonction H(k) qui donne le numéro de la case où est stocké le tuple. Remarque : la taille du tableau est fixée à l'avance (n cases)
   1. Quelle fonction H choisir pour répartir uniformément les tuples dans les cases?
   2. Donner l'algorithme qui remplit la table de hachage à partir d'un ensemble de N tuples et de n cases. Une fois la table créée, quel est le temps nécessaire pour savoir si le tuple de clé k est dans le tableau ?
   3. Quel serait le nombre "idéal" de cases n pour stocker N tuples? En pratique, si on connaît N à l'avance, comment choisir au mieux n pour éviter que les cases "débordent".
   4. Quels sont les avantages et les inconvénients de la table de hachage?
   5. Quelle structure de données, vue l'an dernier, correspond à ce mode de stockage à base de clé ?