Les données en tables

Table de données

On appelle **donnée** toute information sur une personne, un objet, un concept. Par exemple, un nom, une date de naissance ou de création, ou encore une description d'une image sont des données. Lorsqu'on doit traiter un certain nombre de données, il faut les organiser dans une structure qui facilite ce traitement. Les **tables de données** servent justement à organiser les données. Le tableau ci-dessous contient quelques informations tirées d'un fichier clients.

Descripteur → Champ → Entrée →

Nom	Prénom	Code postal	Ville
Tabor	Louis	38100	Grenoble
Chastain	Lucie	71200	Le Creusot
Turcotte	Karim	92110	Clichy

Les noms sur la première ligne s'appellent les **descripteurs**. Ils permettent de comprendre à quoi correspondent chacune des informations des clients. Chacune des autres lignes correspond à une **entrée**, ou **élément**. Dans l'exemple, ce sont les clients. Chaque case du tableau s'appelle **champ**, ou **attribut**. Le nom d'un client, ou son code postal est un champ.

Exploitation d'une table

Une fois que l'on a une table de données, on va généralement vouloir en faire quelque chose. On peut, par exemple, chercher toutes les entrées correspondant à un critère donné. Dans nos clients, nous pourrions chercher ceux qui habitent Grenoble. On peut également souhaiter compter combien il y a d'entrées qui correspondent à ce critère.
Lorsqu'il y a beaucoup de données, l'ordre dans lequel elles sont affichées peut être important. C'est pourquoi une des opérations élémentaires consiste à trier les valeurs. Le plus "naturel" consiste à trier les entrées dans l'ordre lexicographique des champs. Dans l'exemple, cela revient à trier par nom, puis prénom, puis code postal et enfin par Ville, si nécessaire. Mais on peut également décider de trier par code postal en priorité. Ou par prénom. Avec des tables plus complexes, il est même capable de les trier par somme total des achats, ou par temps moyen de connexion.
L'exploitation informatique des données devient réellement intéressante lorsque la taille de la table est massive. Par exemple, imaginons que nous lançions Nesflix, une plateforme de streaming de films et de séries. Il semble logique de faire une table de données contenant tous les films et séries vus par chaque utilisateur. On peut alors chercher combien de personnes ont regardé un film donné. Un utilisateur A vient de regarder "*Le retour de la vengeance 2*". On cherche tous les utilisateurs ayant également vu ce film. On regarde les autres films qu'ils ont vu. On cherche tous les films vu par ces utilisateurs que A n'a pas vus. Il y a "*Pas content du tout 3*" qui a été vu par 60 % d'entre eux. Il semble donc normal de recommander ce film à A.
Ceci est juste un petit exemple de ce qu'on appelle le **Big Data**. L'idée est de traiter des données massives de données pour obtenir des informations inaccessibles pour un humain, à cause du temps de travail que cela demanderait. Cela sert, par exemple, à entraîner des intelligences artificielles à reconnaître des éléments dans des images ou à faire le profil des utilisateurs et déterminer le contenu qui pourrait les intéresser.

Opérations sur les tables

Lorsqu'on veut manipuler une, ou plusieurs tables, il y a plusieurs opérations que l'on peut effectuer. Tout d'abord, lors de l'import d'une table, il y a une étape de **validation**. Il est nécessaire de s'assurer que les valeurs sont valides : par exemple, le code postal doit être un nombre de 5 chiffres. Il faut aussi s'assurer que les valeurs obligatoires, comme le nom, sont bien remplies. Les entrées non valides peuvent être corrigées ou abandonnées.
L'**extraction** consiste, elle, à produire une sous-table ne contenant qu'une partie des descripteurs et/ou une partie des objets correspondant à certains critères particuliers. Par exemple, on peut extraire une table contenant uniquement les codes postaux des clients. Il faudra alors faire attention à ne pas répéter plusieurs fois le même code postal. On pourra également rajouter une colonne pour compter le nombre de fois où chaque code postal apparaît.
La **réunion** de deux tables consiste juste à mettre en commun les données de deux tables ayant les mêmes descripteurs. Il faut néanmoins faire attention aux doublons ou au contraire aux données contradictoires. Par exemple, un client ayant été enregistré à 2 adresses différentes. On pourra, par exemple, mettre une priorité à l'une des deux tables.
La **jointure** de deux tables permet au contraire de fusionner deux tables ayant uniquement certains descripteurs en commun. On pourra, par exemple, mettre en commun les adresses et les numéros de téléphone des clients contenus dans deux tables différentes. Là, il faut voir comment faire pour les entrées n'apparaissant que dans une seule des tables. On peut soit les mettre de côté, ou alors, laisser vide le champ manquant.

Le format CSV

Il existe de nombreux formats utilisés pour stocker les données, comme XML ou JSON. L'un des plus répendus est le format CSV : *Comma Separated Values*. Son grand avantage réside dans sa simplicité.

```
Nom,Prénom,"Code postal",Ville
Tabor,Louis,38100,Grenoble
Chastain,Lucie,71200,"Le Creusot"
Turcotte,Karim,92110,Clichy
```

Il s'agit d'un simple fichier texte, où chaque ligne correspond à une ligne du tableau et les champs sont séparés par des virgules. Dans l'exemple ci-dessus, on peut remarquer que lorsqu'il y a un espace dans le champ, il est alors entouré par des guillemets pour éviter toute ambiguïté. En pratique, les séparateurs de champs peuvent être remplacés par des points virgules ou des tabulations.

Pour aller plus loin

Sur le web, les données sont souvent stockées dans des **bases de données** qui peuvent être manipulées par un langage de programmation spécifique, tel que SQL. Cela sera étudié l'année prochaine dans le programme de terminale NSI.
Les données ne sont généralement pas stockées dans une seule table, mais dans plusieurs qui sont liées les unes aux autres. On parle de **modèle relationnel**. On pourra par exemple associer à chaque client un identifiant unique et faire une table contenant les informations personnelles, une autre avec les commandes de chaque client. Si le client change d'adresse, on la modifie dans la table des informations personnelles et cela n'affecte pas la table avec les commandes, puisque l'identifiant n'est pas modifié. En limitant la redondance dans les tables, on évite les conflits et les incohérences.