1 Modèle relationnel

1.1 Modèle relationnel

Le modèle relationnel est une manière de modéliser les relations existantes entre plusieurs informations, et de les ordonner entre elles. Cette modélisation qui repose sur des principes mathématiques mis en avant par Edgar Franck Codd est souvent retranscrite physiquement dans une base de données.

1.1.1 Objet

Un objet est représenté par un ensemble de caractéristiques qui le rendent unique. On nomme ces tuples, des entités.

1.1.2 Relation

Une relation est un ensemble d'attributs et d'entités. (Dans le vocabulaire des bases de données, on utilise indifféremment les termes relation et table.)

1.1.3 Contrainte d'intégrité

Une contrainte d'intégrité est une propriété vérifiée à tout instant et qui garantit la cohérence des données. (On appelle clé primaire l'attribut qui garantit l'unicité de l'entité.)

1.1.4 Contrainte d'entité

La contrainte d'entité garantit que chaque entité d'une relation est unique et l'identifie de manière non ambiguë.

1.1.5 Contrainte de référence

Dans le MCD, une association entre deux relations est (souvent) traduite par une clé étrangère. La contrainte de référence garantit qu'une entité d'une relation B mentionne une entité existante dans une relation A.

1.1.6 Schéma relationnel

L'ensemble des relations constitue un schéma relationnel.

Représenter une entité

```
Colis(numéro entier, masse entier, longueur entier, largeur entier, hauteur entier, statut c
```

1.1.7 MCD

Le MCD permet de représenter le système d'information indépendamment de son aspect informatique.

1.1.8 MLD

`https://louisvandevelde.be/index.php?dos=my&fic=meris`

2 POO

2.1 Instance

Une instance est une construction concrète, produite à partir de l'objet modèle. L'instance est manipulable dans le programme.

2.2 Classe

Une classe est une modélisation d'un élément du monde réel. Une classe possède :

- des attributs : il s'agit des caractéristiques spécifiques de l'objet (données),
- des méthodes : il s'agit des fonctionnalités que peut réaliser l'objet (traitements et services).

2.3 Récursivité

2.3.1 Fonction récursive

Une fonction récursive est une fonction :

- qui s'appelle elle-même tant qu'une condition est vérifiée,
- possède une condition d'arrêt, pour ne pas s'appeler indéfiniment.

3 Linux

3.1 Shell

Le shell est un programme qui sert d'interface entre le noyau et l'utilisateur.

3.2 Liste chaînée

Dans une liste chaînée, chaque élément :

- prend une place libre quelconque en mémoire,
- connaît l'emplacement de l'élément suivant.

3.3 Pile

Une pile est une liste chaînée dans laquelle on n'accède qu'au dernier élément. Les piles (stack) sont fondées sur le principe du dernier arrivé premier sorti : **Last In First Out**.

3.4 File

Une file est une liste chaînée dans laquelle on accède au premier et au dernier élément. Une file est fondée sur le principe : **First In First Out**.

4 Arbres

4.1 Arbre

Un arbre est défini par :

- un nœud particulier qui constitue la racine,
- plusieurs sous-ensembles d'autres arborescences reliées à la racine.

On nomme :

- nœud-fils l'ensemble des nœuds reliés à un même nœud-père,
- feuilles les nœuds qui n'ont pas de fils.

4.2 Taille

La taille d'un arbre est le nombre de nœuds de la structure.

4.3 Hauteur / profondeur

La hauteur (ou profondeur) d'un arbre est la longueur du plus grand chemin entre la racine et une feuille.

5 Algorithmie

5.1 Coût / Complexité temporelle

Le coût (ou complexité temporelle) représente le nombre d'étapes que l'algorithme doit réaliser pour exécuter sa tâche.

5.2 Terminaison

La terminaison d'un programme est la vérification qu'il finit par s'arrêter, c'est-à-dire que le nombre d'instructions exécutées est fini. Pour prouver la terminaison dans une boucle, on cherche un variant de boucle : une expression qui change à chaque itération, jusqu'à un cas limite.

5.3 Variant de boucle

Un variant de boucle est une expression numérique qui doit être modifiée à chaque itération de la boucle et qui doit garantir que la boucle finira par se terminer.

5.4 Correction d'un programme

La correction d'un programme est le fait de vérifier qu'il réalise effectivement ce qui était prévu. Dans une boucle, on cherche un invariant de boucle : une expression qui est vraie avant chaque itération.

5.5 Invariant de boucle

En programmation, une boucle est une instruction qui permet de répéter l'exécution d'une partie d'un programme. Un invariant de boucle est une propriété qui est vraie avant et après chaque répétition.

6 Graphes non orientés

6.1 Définition

Un graphe est une collection d'éléments mis en relation entre eux.

6.2 Ordre

L'ordre du graphe est le nombre de ses sommets.

6.3 Non orienté

Un graphe est non orienté quand ses arêtes peuvent être parcourues dans les deux sens.

6.4 Adjacence

Deux sommets reliés par une arête sont adjacents.

6.5 Degré d'un sommet

Le degré d'un sommet est le nombre d'arêtes de ce sommet.

6.6 Complétude

Un graphe est complet si tous les sommets sont adjacents à tous les autres.

6.7 Propriétés

La somme des degrés d'un graphe est paire.

$$\sum_{s \in S} \deg(s) = 2A$$

6.8 Représentation en mémoire - Sommets adjacents

Dans un graphe orienté, les sommets adjacents n'ont pas la même position : ils peuvent être des prédécesseurs ou des successeurs.

6.9 Matrice d'adjacence

Dans une matrice d'adjacence, le sommet représenté sur la ligne est le départ de l'arête (le prédécesseur). Il est aussi possible d'établir une matrice d'adjacence pour un graphe orienté. Le principe reste le même : si le sommet i (ligne) est lié au sommet j (colonne), nous avons un 1 à l'intersection (0 dans le cas contraire). En ligne nous avons les prédécesseurs et en colonne les successeurs.

6.10 Dictionnaire d'adjacence

Un dictionnaire d'adjacence liste les sommets adjacents à chaque sommet. On peut faire un dictionnaire des prédécesseurs et un des successeurs.