

3 Programme du troisième semestre

3.1 Bases de données

On se limite volontairement à une description applicative des bases de données en langage SQL. Il s'agit de permettre d'interroger une base présentant des données à travers plusieurs relations. On ne présente pas l'algèbre relationnelle ni le calcul relationnel.

Notions	Commentaires
Vocabulaire des bases de données : tables ou relations, attributs ou colonnes, domaine, schéma de tables, enregistrements ou lignes, types de données.	On présente ces concepts à travers de nombreux exemples. On s'en tient à une notion sommaire de domaine : entier, flottant, chaîne; aucune considération quant aux types des moteurs SQL n'est au programme. Aucune notion relative à la représentation des dates n'est au programme; en tant que de besoin on s'appuie sur des types numériques ou chaîne pour lesquels la relation d'ordre coïncide avec l'écoulement du temps. Toute notion relative aux collations est hors programme; on se place dans l'hypothèse que la relation d'ordre correspond à l'ordre lexicographique usuel. NULL est hors programme.
Clé primaire.	Une clé primaire n'est pas forcément associée à un unique attribut même si c'est le cas le plus fréquent. La notion d'index est hors programme.
Entités et associations, clé étrangère.	On s'intéresse au modèle entité-association au travers de cas concrets d'associations 1 – 1, 1 – *, * – *. Séparation d'une association * – * en deux associations 1 – *. L'utilisation de clés primaires et de clés étrangères permet de traduire en SQL les associations 1 – 1 et 1 – *.
Requêtes SELECT avec simple clause WHERE (sélection), projection, renommage AS. Utilisation des mots-clés DISTINCT, LIMIT, OFFSET, ORDER BY.	Les opérateurs au programme sont +, –, *, / (on passe outre les subtilités liées à la division entière ou flottante), =, <>, <, <=, >, >=, AND, OR, NOT.
Opérateurs ensemblistes UNION, INTERSECT et EXCEPT, produit cartésien.	
Jointures internes T_1 JOIN T_2 ... JOIN T_n ON ϕ . Autojointure.	On présente les jointures en lien avec la notion de relations entre tables. On se limite aux équi-jointures : ϕ est une conjonction d'égalités.
Agrégation avec les fonctions MIN, MAX, SUM, AVG et COUNT, y compris avec GROUP BY.	Pour la mise en œuvre des agrégats, on s'en tient à la norme SQL99. On présente quelques exemples de requêtes imbriquées.
Filtrage des agrégats avec HAVING.	On marque la différence entre WHERE et HAVING sur des exemples.
Mise en œuvre	
La création de tables et la suppression de tables au travers du langage SQL sont hors programme. La mise en œuvre effective se fait au travers d'un logiciel permettant d'interroger une base de données à l'aide de requêtes SQL. Récupérer le résultat d'une requête à partir d'un programme n'est pas un objectif. Même si aucun formalisme graphique précis n'est au programme, on peut décrire les entités et les associations qui les lient au travers de diagrammes sagittaux informels. Sont hors programme : la notion de modèle logique <i>vs</i> physique, les bases de données non relationnelles, les méthodes de modélisation de base, les fragments DDL, TCL et ACL du langage SQL, les transactions, l'optimisation de requêtes par l'algèbre relationnelle.	

3.2 Dictionnaires et programmation dynamique

Les dictionnaires sont utilisés en boîte noire dès la première année ; les principes de leur fonctionnement sont présentés en deuxième année. Ils peuvent être utilisés afin de mettre en mémoire des résultats intermédiaires quand on implémente une stratégie d'optimisation par programmation dynamique.

Notions	Commentaires
Dictionnaires, clés et valeurs.	On présente les principes du hachage, et les limitations qui en découlent sur le domaine des clés utilisables.
Usage des dictionnaires en programmation Python.	Syntaxe pour l'écriture des dictionnaires. Parcours d'un dictionnaire.
Programmation dynamique. Propriété de sous-structure optimale. Chevauchement de sous-problèmes. Calcul de bas en haut ou par mémorisation. Reconstruction d'une solution optimale à partir de l'information calculée.	La mémorisation peut être implémentée à l'aide d'un dictionnaire. On souligne les enjeux de complexité en mémoire. Exemples : partition équilibrée d'un tableau d'entiers positifs, ordonnancement de tâches pondérées, plus longue sous-suite commune, distance d'édition (Levenshtein), distances dans un graphe (Floyd-Warshall).
Mise en œuvre	
Les exemples proposés ne forment une liste ni limitative ni impérative. Les cas les plus complexes de situations où la programmation dynamique peut être utilisée sont guidés. On met en rapport le statut de la propriété de sous-structure optimale en programmation dynamique avec sa situation en stratégie gloutonne vue en première année.	

3.3 Algorithmique pour l'intelligence artificielle et l'étude des jeux

Cette partie permet notamment de revisiter les notions de programmation et de représentation de données par un graphe, qui sont vues en première année, en les appliquant à des enjeux contemporains.

Notions	Commentaires
Algorithme des k plus proches voisins avec distance euclidienne.	Matrice de confusion. Lien avec l'apprentissage supervisé.
Algorithme des k -moyennes.	Lien avec l'apprentissage non-supervisé. La démonstration de la convergence n'est pas au programme. On observe des convergences vers des minima locaux.
Jeux d'accessibilité à deux joueurs sur un graphe. Stratégie. Stratégie gagnante. Position gagnante. Détermination des positions gagnantes par le calcul des attracteurs. Construction de stratégies gagnantes.	On considère des jeux à deux joueurs (J_1 et J_2) modélisés par des graphes bipartis (l'ensemble des états contrôlés par J_1 et l'ensemble des états contrôlés par J_2). Il y a trois types d'états finals : les états gagnants pour J_1 , les états gagnants pour J_2 et les états de match nul. On ne considère que les stratégies sans mémoire.
Notion d'heuristique. Algorithme min-max avec une heuristique.	L'élagage alpha-beta n'est pas au programme.
Mise en œuvre	
La connaissance dans le détail des algorithmes de cette section n'est pas un attendu du programme. Les étudiants acquièrent une familiarité avec les idées sous-jacentes qu'ils peuvent réinvestir dans des situations où les modélisations et les recommandations d'implémentation sont guidées, notamment dans leurs aspects arborescents.	