MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE

REPUBLIQUE TOGOLAISE
Travail – Liberté – Patrie

Université de Lomé | CENTRE INFORMATIQUE ET DE CALCUL (CIC)

Thème :

LES ASPECTS MATHEMATIQUE ET INFORMATIQUE DES SYSTEMES RELATIONNELS DE DONNEES

Présenté par :

- **ABELE Hezouwe Marie-José**
- **ASSOU Yao Mokpokpo**
- **TOUNOU Adoté Corneille**

CHARGE DU COURS

Mr. EDARH-BOSSOU

Sommaire

Introduction

La mathématique est la science qui traite de la logique de la forme, de la quantité et de l'arrangement. Les mathématiques sont vraiment utiles et significatives dans notre vie quotidienne. En effet, elles nous aident à être logiques, à raisonner de manière soignée et à se préparer à la pensée, à la critique et à l'abstraction [1].

Les mathématiques sont utilisées dans pratiquement toutes les carrières. De toute évidence, les mathématiciens et les scientifiques s'appuient sur des principes mathématiques pour effectuer les aspects les plus fondamentaux de leur travail. Les personnes qui prennent des médicaments doivent comprendre différentes doses, en grammes ou en millilitres. Les décorateurs doivent savoir que les dimensions de leurs meubles et leurs tapis correspondront à la superficie de leurs chambres, et même les fans de football connaissent les gains de distance et les statistiques de passes. En plus les mathématiques jouent un rôle important dans la réalisation des systèmes de gestion de bases de données. C'est ainsi que notre thème se base sur les aspects mathématique et informatique des systèmes relationnels de données [2].

Dans la suite du document, nous aurons à expliquer premièrement la relation entre la théorie des ensembles et les systèmes de gestion de bases de données (SGBD) ; ensuite en deuxième position nous aurons à discuter du lien entre l'algèbre relationnelle et les SGBD ; puis finirons avec la relation entre la logique du premier ordre et les SGBD.

Liste des figures

Résumé

Avant les années 1960, la gestion des données se faisait par les systèmes de fichiers. Afin de bien gérer quantitativement, qualitativement, assurer la sécurité, rechercher d'une manière rapide et structurer les données, un nouveau système a vu le jour dans les années 1960 : le Système de Gestion de Base de Données (SGBD).

Pour une meilleure compréhension des bases de données, il est utile de faire les maths car plusieurs concepts fondamentaux utilisés pour les bases de données reposent sur les outils mathématiques.

L'objectif de notre exposé est de décrire les fondements mathématiques des Systèmes de Gestion de bases de données dit relationnels, ceux misent en œuvre dans les logiciels tels qu'Oracle, MySQL, ou ACCESS etc.

Pour atteindre cet objectif, notre travail relèvera les aspects suivants dans les systèmes relationnels de données tels que :

- La théorie des ensembles ;
- L'algèbre relationnel ;
- Logique du premier ordre.

CHAPITRE 1 : Théorie des ensembles dans les bases de données

I.1 Historique

Cantor est un mathématicien allemand. Il a créé la théorie des ensembles. Cette théorie est considérée comme une théorie fondamentale des mathématiques (au sens où elle permet de fonder les mathématiques) [1].

Plusieurs axiomatiques ont ensuite été développées : la plus connue étant celle des axiomes de ZF (Zermelo et Frankel). Notons aussi la théorie des classes de Neumann ou la théorie des types de Russell. La théorie des ensembles est à la base de ce qu'on a appelé dans les années 60 et 70 les « maths modernes » [1].

Cantor pose l'existence de 2 infinis… et même d'une infinité d'infini. Ces infinis sont appelés « Aleph ». Le plus petit des infinis est appelé Aleph-zéro : aleph0.

Cantor pose que :

card(N) =2 $^{\textbf{aleph0}}$. Il appelle cet infini « dénombrable ».

Cantor pose cette hypothèse : C'est l'hypothèse du continu. L'histoire raconte qu'il serait devenu fou à force d'essayer de la démontrer. Des mathématiciens-logiciens démontreront plus tard que l'hypothèse du continu est indémontrable [1].

I.2 Définition

La théorie des ensembles se donne comme primitives les notions d'ensemble et d'appartenance, à partir desquelles elle reconstruit les objets usuels des mathématiques : fonctions, relations, entiers naturels, relatifs, rationnels, nombres réels, complexes [2].

Pour les informaticiens, c'est un outil intuitif qui permet de comprendre plus intuitivement certaines notions telles que :

- la modélisation des données pour les bases de données et programmation objet : MEA (Modèle Entité-Association) et UML (Langage de Modélisation Unifié) ;
- le calcul relationnel pour les bases de données : produit cartésien et jointure ;
- les notions d'héritage et d'abstraction en programmation objet ;
- la déduction en intelligence artificielle [1].

Les modélisations utilisent leur propre langage graphique pour représenter les ensembles et leurs relations. Le principe consiste à montrer les ensembles, les caractéristiques des tuples qu'ils contiennent, les relations entre les ensembles et les cardinalités de ces relations [1].

I.3 Relation entre le théorème des ensembles et les systèmes de gestions des bases de données

Explication de la relation entre la théorie d'ensemble et les systèmes de gestions des bases de données.

Prenons l'exemple des lions qui mangent les gazelles.

Soit L un ensemble de lions. L = {l1, l2, l3, l4} Soit G un ensemble de gazelles. G = {g1, g2, g3}.

Nous nous intéressons à la relation : « manger » car les lions mangent les gazelles.

Nous pouvons décrire la situation par un la figure 1.

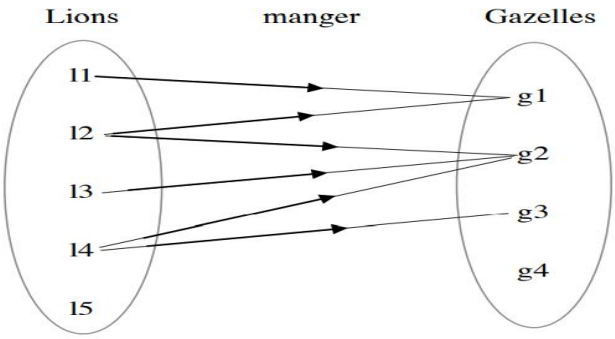

Figure 1. Schéma montrant la relation entre les lions et les gazelles

D'après la figure 1, nous pouvons dire que les lions constituent un ensemble et les gazelles forment un ensemble.

Nous savons que les lions et les gazelles sont des animaux. Donc L ⊂ A et G ⊂ A. La figure 2 illustre le lien entre animaux, lions et gazelles :

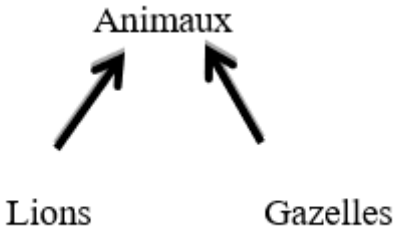

Figure 2.Schéma montrant la relation entre le genre animal

Nous pouvons représenter le tout par une inclusion d'ensembles comme le montre la figure 3 :

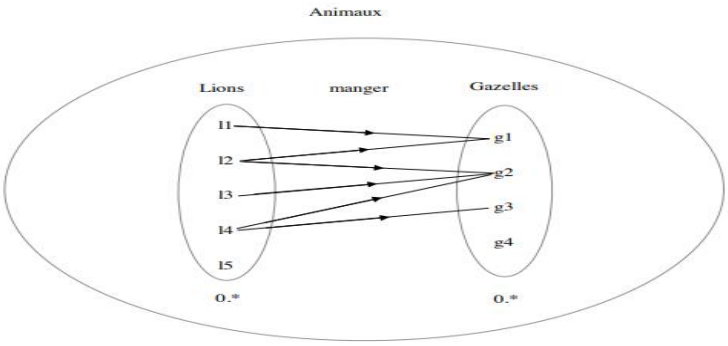

Figure 3.Schéma de l'inclusion d'ensembles

Tous les animaux ont une date de naissance, un poids et un nom. Les gazelles et les lions héritent des attributs du genre : ils ont tous une date de naissance, un poids et un nom. Les gazelles ont une longueur de cornes spécifique. Les lions ont une couleur de crinière spécifique.

- Formalisme MEA

Le MEA est une façon graphique de représenter les ensembles et leurs relations qui sont utilisées en base de données.

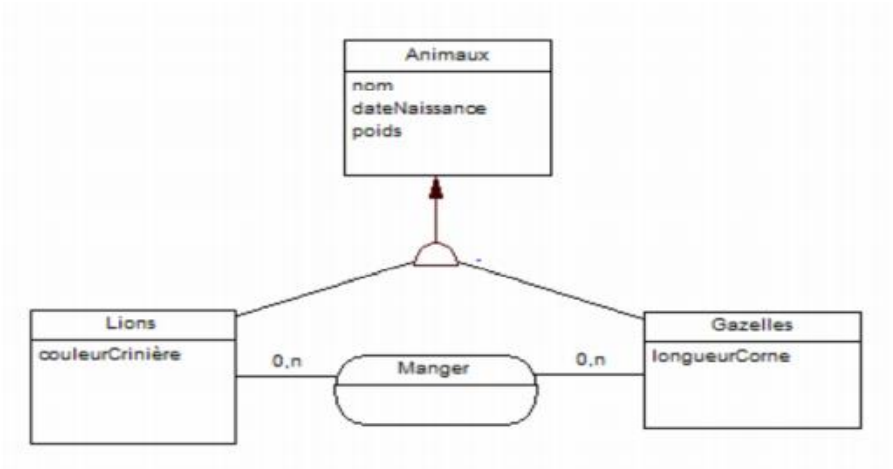

Figure 4.Schéma de formalisme MEA

- Formalisme UML L'UM

L'UML est une façon graphique de représenter les ensembles et leurs relations qui sont utilisées en base de données.

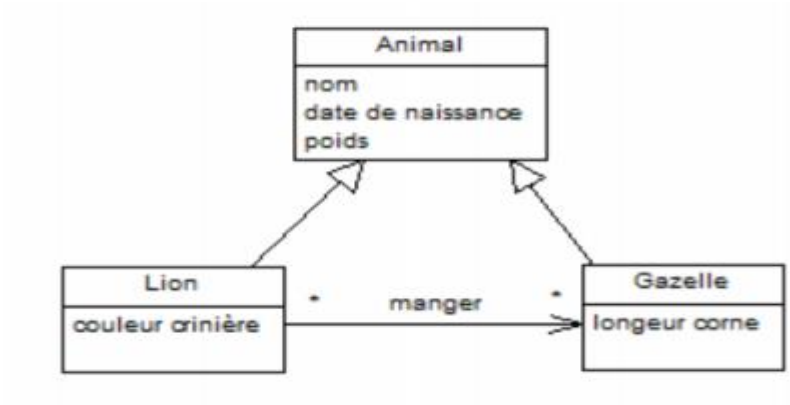

Figure 5.Schéma de formalisme UML

La théorie des ensembles dans notre cas est utilisée comme un ensemble d'appartenance dont l'ensemble des lions et gazelles appartiennent à l'ensemble des animaux. Cette appartenance nous a permis de faire le formalise UML ou MEA qui nous servira à faire du SQL grâce à l'algèbre relationnelle.

Ainsi, l'algèbre relationnelle est un langage qui s'appuie sur la théorie des ensembles [1].

Chapitre 2 : L'algèbre rationnelle dans les bases de données.

I. Rappel sur les requêtes

En informatique, une requête est une interrogation d'une base de données. Elle est destinée à obtenir des informations précises et ordonnées et peut comporter un certain nombre de critères pour préciser la demande. On peut également par le biais des requêtes :

- Effectuer des calculs ;
- Obtenir des statistiques ;
- Modifier les tables ;
- Créer de nouvelles tables ;
- Créer de nouvelles tables ;
- Etc.

Il existe plusieurs façons de créer des requêtes et l'un des plus connus est L'algèbre relationnelle qui permet de préparer une requête en utilisant un langage Mathématique.

II. Historique et Définition

L'algèbre relationnelle est un concept mathématique de relation de la théorie des ensembles .C'est un langage de requêtes dans des bases de données relationnelles, inventé en 1970 par Edgar Frank Codd, le directeur de recherche du centre IBM de San José International Business Machines Corporation. Il s'agit d'une théorie sous-jacente aux langages de requête des SGBD, comme le SQL, permettant une description symbolique préliminaire à l'utilisation d'un langage non procédural. Le but est de manipuler des relations et de produire comme résultat de nouvelles relations. Elle est souvent définie comme étant conceptuellement un langage **procédural** car en effet, les requêtes sont des suites d'opérations qui construisent la réponse. Cela s'oppose aux langages conceptuellement **déclaratifs** comme le calcul relationnel et le Datalog qui est un langage de requête et de règles pour les bases de données déductives.

III. Les types d'opérations et leurs équivalents en Base de données

Étant un langage procédural, l'algèbre relationnelle contient des opérations **ensemblistes** de la théorie des ensembles sur les relations ainsi que des opérations pour fusionner/projeter des relations dites **relationnelles.**

1. Opérateurs ensemblistes

Les différentes opérations ensemblistes sont l'union, l'intersection, la différence et le produit cartésien.

1.1. L'union

L'union de deux relations sur le même schéma est la relation sur ce schéma contenant exactement l'union des enregistrements de ces deux relations. Formellement,

$$R \cup S = \{t : t \in R \, ou \, t \in S\}$$

Équivalent SQL : UNION.

Employés de Renault

Nom	ID	Département
Harry	3415	Finance
Sally	2241	Vente
George	3401	Finance

Employés de Citroën

Nom	ID	Département
Bertrand	0808	Vente
Donald	0007	Vente

Employés de Renault ∪ *Employés de Citroën*

Nom	ID	Département
Harry	3415	Finance
Sally	2241	Vente
George	3401	Finance
Bertrand	0808	Vente
Donald	0007	Vente

1.2. L'intersection

L'intersection de deux relations sur le même schéma est la relation sur ce schéma contenant exactement les enregistrements qui apparaissent dans les deux relations. Formellement,

$$R \cap S = \{t : t \in R \, et \, t \in S\}$$

Équivalent SQL : INTERSECT

Personnes inscrits en football

Nom	ID
Harry	3415
Sally	2241
George	3401

Personnes inscrits en cours de piano

Nom	ID
Harry	3415
Bertrand	2
George	3401
Yoda	1000

Personnes inscrits en football ∩ Personnes inscrits en cours de piano

Nom	ID
Harry	3415
George	3401

1.3. La différence

La différence de deux relations sur le même schéma est la relation sur ce schéma contenant exactement les enregistrements qui apparaissent dans la première relation mais pas dans la deuxième. Formellement,

$$R - S = \{t : t \in R \, et \, t \notin S\}$$

Équivalent SQL : **EXCEPT (MINUS sous ORACLE)**

Par exemple, on peut calculer les personnes inscrites en football mais qui ne sont pas inscrits en cours de piano :

Personnes inscrits en football

Nom	ID
Harry	3415
Sally	2241
George	3401

Personnes inscrits en cours de piano

Nom	ID
Harry	3415
Bertrand	2
George	3401
Yoda	1000

Personnes inscrits en football - Personnes inscrits en cours de piano

Nom	ID
Sally	2241

1.4. Le produit cartésien

Avec le produit cartésien de deux relations, on recopie chaque tuple de la première relation pour chacun des tuples de la deuxième. Formellement,

$R \times S = \{(r, s) : r \in R \, et \, s \in S\}.$

Équivalent SQL : CROSS JOIN

Personnes

Nom	ID
Harry	3415
Sally	2241

Cadeaux

Type	Prix
livre	10
gâteau	20
ordinateur	300

Personnes - Cadeaux

Nom	ID	Type	Prix
Harry	3415	livre	10
Harry	3415	gâteau	20
Harry	3415	ordinateur	300
Sally	2241	livre	10
Sally	2241	gâteau	20
Sally	2241	ordinateur	300

2. Opérateurs relationnels

Les différentes opérations relationnelles sont la sélection, la projection et la jointure

2.1. La sélection

La sélection (ou restriction) consiste à ne retenir que les enregistrements vérifiant une condition donnée. Dans l'exemple plus bas, on ne garde que les touristes dont la ville est Paris.

- Données : Une relation R et une formule F formée d'une combinaison de comparaisons et de connecteurs logiques ;
- Résultat : $\sigma_F(R) = \{r \in R : r$ satisfait la condition donnée par $F\}$, satisfait la condition donnée par $F\}$, dans l'exemple $\sigma_{Ville='Paris'}$ (Touristes
- Équivalent SQL : WHERE

Dans l'exemple plus bas, on ne garde que les touristes dont la ville est Paris.

Touristes

idTouriste	NomT	Ville	Sport
1	Marc	Paris	Ski
2	Jean	Toulouse	Tennis
3	Franc	Marseille	Football
4	Thomas	Lyon	Voile
5	Max	Paris	Golf

Sélection des touristes où Ville vaut Paris

idTouriste	NomT	Ville	Sport
1	Marc	Paris	Ski
5	Max	Paris	Golf

2.2. La Projection

La projection permet de ne garder que certains attributs. Dans l'exemple ci-dessous, on ne garde que les attributs NomT et Ville de la relation Touristes.

- Données : Une relation R et un ensemble d'attributs A et R ;
- Résultat : $\pi_A(R)$, qui est la Relation où on ne considère que les attributs de par exemple $\pi_{NomT,Ville}$(Touristes
- Équivalent SQL : **SELECT**

Touristes

idTouriste	NomT	Ville	Sport
1	Marc	Paris	Ski
2	Jean	Toulouse	Tennis
3	Franc	Marseille	Football
4	Thomas	Lyon	Voile
5	Max	Paris	Golf

Projection de la relation Touristes sur les attributs NomT et Ville

NomT	Ville
Marc	Paris
Jean	Toulouse
Franc	Marseille
Thomas	Lyon
Max	Paris

2.2. La Jointure

Les jointures permettent d'associer plusieurs tables dans une même requête. Cela permet d'exploiter la puissance des bases de données relationnelles pour obtenir des résultats qui combinent les données de plusieurs tables de manière efficace.

- Données : deux relations R et S ;
- Résultat : $R \bowtie S = \{(a, b, c) : (a, b) \in R \text{ et } (b, c) \in S\}$
- Équivalent SQL : **JOIN**

Touristes

idTouriste	NomT	Ville	Sport
1	Marc	Paris	Ski
2	Jean	Toulouse	Tennis
3	Franc	Marseille	Football
4	Thomas	Lyon	Voile
5	Max	Paris	Golf

Destinations

idTouriste	VilleD
1	Cannes
2	Ibiza
4	Tokyo

Touristes ⋈ Destinations

idTouriste	NomT	Ville	Sport	VilleD
1	Marc	Paris	Ski	Cannes
2	Jean	Toulouse	Tennis	Ibiza
4	Thomas	Lyon	Voile	Tokyo

Chapitre 3 : Logique du premier ordre

I. Historique

Historiquement, les chercheurs ont tout d'abord essayé de modéliser les langages d'interrogation de bases de données par la logique. Ainsi sont nés les calculs relationnels, véritables langages d'interrogation fondés sur la logique du premier ordre. Ce n'est qu'à partir de la fin des années 70 que l'on a cherché à comprendre globalement les bases de données par la logique. Cela a permis de montrer qu'un SGBD était un démonstrateur de théorèmes très particulier, raisonnant sur des faits et donnant les preuves d'un théorème représentant la question. Les travaux sur l'introduction dans les SGBD de raisonnements généraux incluant non seulement des faits, mais aussi des règles déductives exprimées en logique du premier ordre, ont débuté principalement au CERT à Toulouse à la fin de la décennie 1970-1980 [Gallaire78]. Ils se sont surtout concentrés sur la compréhension des bases de données déductives par la logique. Il est a noté que La logique permet une modélisation simple des bases de données, l'introduction de langages de requêtes formels et le support de la déduction.

II. Définition du concept de la logique du premier ordre

1. Syntaxe de la logique du premier ordre

La logique du premier ordre est aussi appelé calcul de prédicat.
C'est un langage formel utilisé pour représenter des relations entre objets et pour déduire de nouvelles relations à partir de relations connues comme vraies. Elle peut être vue comme un formalisme utilisé pour traduire des phrases et déduire de nouvelles phrases à partir de phrases connues.

Elle se repose sur un alphabet qui utilise les symboles suivants :

1- Variables (x, y, z,…) ;

2- Constantes (a, b, c,…) ;

3. Des prédicats généralement notés P, Q, R ..., chaque prédicat pouvant recevoir un nombre fixe d'arguments (n pour un prédicat naire) ;

4- Les connecteurs logiques $\wedge$, $\vee$, $\Rightarrow$, $\neg$;

5- Des fonctions généralement dénotées f, g, h..., chaque fonction pouvant recevoir un nombre fixe d'arguments (n pour une fonction naire) ;

6- Les quantificateurs $\forall$ et $\exists$.

Des règles syntaxiques simples permettent de construire des formules. Un terme est défini récursivement comme une variable ou une constante ou le résultat de l'application d'une fonction à un terme. Par exemple, x, a, f(x) et f(f(x)) sont des termes. Une formule est une phrase bien construite du langage. Une formule est définie comme suit :
1. Si P est un prédicat à n arguments (n places) et t1, t2...tn des termes, alors P(t1,t2... tn) est une formule atomique ;

2. Si F1 et F2 sont des formules, alors F1 ∧ F2, F1 ∨ F2, F1 ⇒ F2 et ¬ F1 sont des formules ;
3. Si F est une formule et x une variable non quantifiée (on dit aussi libre) dans F, alors ∀x F et ∃x F sont des formules.

Nous résumons ci-dessous les concepts essentiels introduits jusque-là sous forme de notions.

Notion V.1 : Terme
Constante, variable ou fonction appliquée à un terme.

Notion V.2 : Formule atomique
Résultat de l'application d'un prédicat à n places à n termes, de la forme P(t1,t2...tn).

Notion V.3 : Formule
Suite de formules atomiques ou de négations de formules atomiques séparées par des connecteurs logiques et, ou, implique, avec d'éventuelles quantifications des variables.

Pour indiquer les priorités entre connecteurs logiques, il est possible d'utiliser les parenthèses : si F est une formule valide, (F) en est aussi une.

Afin d'éviter les parenthèses dans certains cas simples, nous supposerons une priorité des opérateurs logiques dans l'ordre descendant ¬, ∨, ∧, et ⇒

Exemples de formules formelles :
P(a,x) ∧ Q(x,y),
¬ Q(x,y),
∀x ∃y (Q(x,y) ∨ P(a,x)),
∀x (R(x) ∧ Q(x,a) ⇒ P(b,f(x))).

Afin de donner des exemples plus lisibles de formules, nous choisirons un vocabulaire moins formel, les prédicats et fonctions étant des noms ou des verbes du langage courant et les constantes des chaînes de caractères désignant par exemple des services ou des employés. Voici quelques exemples de formules proches du langage naturel :
SERVICE(informatique,pierre) ∨ EMPLOYE (informatique,marie)
∀x ((DIRIGE(pierre,x) ∨ ¬ EMPLOYE(informatique,x) ⇒ EMPLOYE(finance,x))
∀x ∃y ((DIRIGE(x,y) ∨ DIRIGE(y,x)) ⇒ MEMESERVICE(x,y))

2. Sémantique de la logique du premier ordre

Une formule peut être interprétée comme une phrase sur un ensemble d'objets : il est possible de lui donner une signification vis-à-vis de cet ensemble d'objets. Pour ce faire, il faut assigner un objet spécifique à chaque constante. Par exemple, on assigne un objet à la constante a, le service Informatique de l'entreprise considérée à la constante informatique, l'employée Marie à la constante marie, etc. L'ensemble d'objets considérés est appelé le domaine de discours ; il est noté D. Chaque fonction à n arguments est interprétée comme une fonction de D puissance n dans D. Un prédicat représente une relation particulière entre les objets de D, qui peut être vraie ou fausse. Définir cette relation revient à définir les tuples d'objets qui satisfont le prédicat. L'ensemble des tuples satisfaisant le prédicat constitue son extension.

Notion V.4 : Domaine de discours

Ensemble d'objets sur lequel une formule logique prend valeur par interprétation des constantes comme des objets particuliers, des variables comme des objets quelconques, des prédicats comme des relations entre objets et des fonctions comme des fonctions particulières entre objets.
Par exemple, la formule :$\forall x \exists y ((DIRIGE(x,y) \vee DIRIGE(y,x)) \Rightarrow MEMESERVICE(x,y))$ peut être interprétée sur l'ensemble des personnes {Pierre Paul, Marie}, qui constitue alors le domaine de discours. DIRIGE peut être interprété comme la relation binaire « est chef de » ; MEMESERVICE peut être interprété comme la relation binaire « travaille dans le même service ». La formule est vraie si Pierre, Paul et Marie travaillent dans le même service.

Un univers de discours défini peut être retenu pour interpréter la formule $\forall x (x < SUCC(x))$. En effet, cette formule peut être interprétée sur l'ensemble des entiers positifs {1,2,3...} qui est infini. < est alors la relation « est inférieur à » et SUCC la fonction qui à tout entier associe son successeur. Il est clair que cette formule est vraie sur les entiers. Ainsi, étant donnée une interprétation d'une formule sur un domaine de discours, il est possible d'associer une valeur de vérité à cette formule. Pour éviter les ambiguïtés (les formules pouvant avoir plusieurs valeurs de vérité), nous considérons seulement les formules dans lesquelles toutes les variables sont quantifées, appelées formules fermées. Toute formule fermée F a une unique valeur de vérité pour une interprétation donnée sur un domaine de discours D. Cette valeur notée V(F) se calcule ainsi :
$V(F1 \vee F2) = V(F1) \vee V(F2)$
$V(F1 \wedge F2) = V(F1) \wedge V(F2)$
$V(\neg F1) = \neg V(F1)$
$V(\forall x F) = V(F\ x=a) \wedge V(F\ x=b) \wedge$ ... où a, b, ... sont les constantes de D.
$V(\exists x F) = V(F\ x=a) \vee V(F\ x=b) \vee$ ... où a, b, ... sont les constantes de D.
V(P(a,b...)) = Vrai si [a,b...] satisfait la relation P et Faux sinon.
Un modèle d'une formule logique est une interprétation sur un domaine de discours qui rend la formule vraie. Par exemple, les entiers sont un modèle pour la formule $\forall x (x< SUCC(x))$ avec l'interprétation indiquée ci-dessus. En effet, en appliquant les règles ci-dessus, on calcule :
$V(\forall x (x < SUCC(x))) = V(1 < 2) \wedge V(2<3) \wedge V(3<4) \wedge ... = Vrai.$

B- Les bases de données logiques

La notion de base de données logique a été introduite en particulier à Toulouse à la n des années 70 [Gallaire84]. Nous définissons ici ce qu'est une base de données logique non déductive. Il s'agit d'une première approche simple des bases de données par la logique. Nous verrons plus loin que cette approche peut être étendue avec la déduction.

1. La représentation des faits

Une base de données peut être définie comme l'interprétation d'un langage logique du premier ordre L. En pratique, définir l'interprétation consiste à définir les prédicats vrais. Le langage permet d'exprimer les requêtes, comme nous le verrons ci-dessous. Une contrainte d'intégrité peut être vue comme une requête toujours vraie, donc exprimée avec le langage.
En première approche, le langage logique L ne comporte pas de fonctions. Plus particulièrement, L se compose :

1. De prédicats à n arguments, chaque argument correspondant à un type de données élémentaire ;
2. De constantes, une pour chaque valeur possible des types de données élémentaires.
Comme dans toute interprétation d'un langage logique, un prédicat représente une relation particulière entre les objets du domaine de discours, qui peut être vraie ou fausse. Ici, les objets du domaine de discours sont donc les valeurs possibles de la base. Définir cette relation revient à définir les articles ou n-uplets qui satisfont le prédicat. L'ensemble des n-uplets satisfaisant le prédicat constitue son extension. Cette extension peut être assimilée à un chier dans une base en réseau ou à une table relationnelle, comme nous le verrons plus loin. Pour rester cohérents avec les bases de données relationnelles qui peuvent être perçues comme une implémentation simple des bases de données logiques, nous appellerons l'extension d'un prédicat une table. Chaque colonne correspond à un argument et est aussi appelée attribut dans le scontexte relationnel. Comme déjà dit, une base de données logique pourra être complétée par des règles de déduction : nous parlerons alors de base de données déductive.
La figure ci-dessous illustre une base de données logique.
Elle est composée de trois prédicats définis comme suit :
PRODUIT avec les arguments Numéro de produit (NPRO), nom de produit (NPRO), quantité en stock (QTES), et couleur (COULEUR) ;
VENTE avec les arguments numéro de vente (NVEN), nom du client (NOMC), numéro du

PRODUIT	NPRO	NOMP	QTES	COULEUR
	100	Bille	100	Verte
	200	Poupée	50	Rouge
	300	Voiture	70	Jaune
	400	Carte	350	Bleu

VENTE	NVEN	NOMC	NPRV	QTEV	DATE
	1	Dupont	100	30	08-03-1999
	2	Martin	200	10	07-01-1999
	3	Charles	100	50	01-01-2000
	4	Charles	300	50	01-01-2000

ACHAT	NACH	DATE	NPRA	QTEA	NOMF
	1	01-03-1999	100	70	Fournier
	2	01-03-1999	200	100	Fournier
	3	01-09-1999	100	50	Dubois
	4	01-09-1999	300	50	Dubois

Figure 6.Base de données logiques

produit vendu (NPRV), quantité vendue (QTEV) et date (DATE) ;
ACHAT avec les arguments numéro d'achat (NACH), date de l'achat (DATE), numéro du produit acheté (NPRA), quantité achetée (QTEA) et nom du fournisseur (NOMF).
Comme on le voit, seuls les faits positifs, c'est-à-dire ceux satisfaisant l'un des trois prédicats, sont enregistrés dans la base. Ceci constitue l'hypothèse du monde fermé. Les faits non enregistrés dans une extension de prédicat sont supposés faux. Il est vrai que si vous interrogez la base de données que gère votre banque et que vous ne voyez pas de chèque de 100 000 euros crédités ce jour, c'est que le fait que cette somme ait été créditée sur votre compte est faux ! Des chercheurs ont essayé de lever cette hypothèse : on tombe alors dans l'hypothèse du monde ouvert, ou un fait non enregistré peut être inconnu [Reiter78].

PRODUIT (100, BILLE, 100, VERTE)

PRODUIT (200, POUPEE, 50, ROUGE)
PRODUIT (300, VOITURE, 70, JAUNE)
PRODUIT (400, CARTE, 350, BLEU)
VENTE (1, DUPONT, 100, 30, 08-03-1999)
VENTE (2, MARTIN, 200, 10, 07-01-1999)
VENTE (3, CHARLES, 100, 50, 01-01-2000)
VENTE (4, CHARLES, 300, 50, 01-01-2000)
ACHAT (1, 01-03-1999, 100, 70, FOURNIER)
ACHAT (2, 01-03-1999, 200, 100, FOURNIER)
ACHAT (3, 01-09-1999, 100, 50, DUBOIS)

1.1. Questions et contraintes d'intégrité

Les questions et les contraintes d'intégrité sur la base peuvent alors être exprimées comme des formules dans le langage logique. Cependant, celui-ci doit être étendu pour inclure les opérateurs de comparaison arithmétique {=, <, ≤, >, ≥, ≠} comme des prédicats particuliers, dont la valeur de vérité est calculée par les opérations usuelles des calculateurs.
La réponse à une question F(x1, ..., xn) - où x1, ..., xn sont des variables libres dans la formule F - est alors l'ensemble des n-uplets <e1, ..., ep> tels que F(e1, ..., ep) est vraie. Certaines formules doivent être toujours vraies sur l'interprétation que constitue la base : ce sont alors des contraintes d'intégrité. Cette vue logique des bases de données a donné naissance à deux calculs permettant d'exprimer des questions et des contraintes d'intégrité sur les bases : le calcul de domaine et le calcul de tuple. Dans le premier, les objets de l'interprétation logique sont les valeurs atomiques de données ; dans le second, ce sont les faits composites correspondant aux n-uplets, encore appelés tuples.Nous étudions ces deux calculs ci-dessous.

1.2.Le calcul de domaines

Les calculs relationnels de domaine et de tuples [Codd72] permettent d'exprimer des requêtes à l'aide de formules du premier ordre sur une base de données logique, ou sa représentation tabulaire qui est une base de données relationnelle. Ils ont été utilisés pour formaliser les langages d'interrogation pour les bases de données relationnelles.

1.2.1.Principes de base

Le calcul relationnel de domaines [Codd72] se déduit donc de la logique du premier ordre. Les données sont les constantes. Les prédicats utilisés sont :
1. Les prédicats extensionnels contenant les enregistrements de la base ; chaque enregistrement est un tuple typé selon un prédicat extensionnel ; les prédicats étant naires, une variable ou une constante est utilisée comme argument de prédicat ; les variables sont alors implicitement typées selon le type de l'argument pour lequel elles apparaissent ;
2. Les prédicats de comparaison entre une variable et une constante, ou entre deux variables, construits à l'aide des opérateurs {=, ≤, <, ≥, >, ≠}.
Une question en calcul relationnel de domaine s'écrit donc sous la forme suivante :
{x, y... | F(x,y, ...)}
F est une formule logique composée à partir de prédicats extensionnels et de comparaison ; les variables résultats x, y, ..., sont des variables libres dans F.
La notion suivante résume la défisnition du calcul de domaine.
Notion V.8 : Calcul relationnel de domaine

Langage d'interrogation de données formel permettant d'exprimer des questions à partir de formules bien formées dont chaque variable est interprétée comme
variant sur le domaine d'un argument d'un prédicat.
La BNF du calcul relationnel de domaine est définie.Pour simplifier, les formules sont écrites en forme prenex (quantificateurs devant). En pratique, une
simpli cation d'écriture permet de regrouper des prédicats de restriction du type (x = a) et des prédicats de définition de variable du type P (x, ...) en écrivant P (a, ...). Toute variable apparaissant dans le résultat doit être dé nie dans un prédicat extensionnel et doit rester libre dans la formule spécifiant la condition. Une variable non utilisée, ni
en résultat, ni dans un critère de comparaison, peut être remplacée par la variable muette positionnelle notée « - ».

1.2.2. Quelques exemples de calculs de domaine

Nous illustrons le calcul de domaine sur la base logique introduite ci-dessus décrivant des achats et des ventes de produits stockés dans un magasin. Le schéma de la base
est le suivant :

PRODUIT (NPRO, NOMP, QTES, COULEUR)
VENTE (NVEN, NOMC, NPRV, QTEV, DATE)
ACHAT (NACH, DATE, NPRA, QTEA, NOMF)
Le prédicat extensionnel PRODUIT se compose des attributs numéro de produit (NPRO), nom du produit (NOMP), quantité en stock (QTES) et couleur (COULEUR). Le
prédicat VENTE décrit les ventes de produits effectuées et se compose des attributs numéro de vente (NVEN), nom du client (NOMC), numéro du produit vendu (NPRV), quantité vendue (QTEV) et date de la vente (DATE). Le prédicat ACHAT dé nit les achats effectués aux fournisseurs. Il contient les attributs numéro d'achat (NACH), date
d'achat (DATE), numéro du produit acheté (NPRA), quantité achetée (QTEA) et nom du fournisseur (NOMF). Pour simpli er la lecture, nous utilisons des variables rappelant
le nom du domaine. Notez que les quanti cateurs existentiels dans les formules peuvent être omis, car ils sont implicites si la variable est libre et ne gure pas en résultat.
(Q1) Donner la liste des noms et des couleurs de tous les produits :
{(p,c) | PRODUIT (-,p,-,c)}
(Q2) Donner les noms et les quantités en stock des produits de couleur rouge :
{(p,q) | PRODUIT (-, p, q, "Rouge")}
(Q3) Donner, pour chaque produit en stock, le nom du fournisseur associé :
{(p,f) | ∃n (PRODUIT (n, p, -, -) ∧ ACHAT (-, -, n, -, f))}
(Q4) Donner, pour chaque produit en stock en quantité supérieure à 100 et de couleur rouge, les triplets nom de fournisseurs ayant vendu ce type de produit, nom de
client ayant acheté ce type de produit et nom du produit :
{(f, c, p) | ∃n ∃q (PRODUIT (n, p, q, "Rouge") ∧ ACHAT (-, -, n, -, f)
∧ VENTE (-, c, n, -, -) ∧ (q > 100))}
(Q5) Donner les noms des clients ayant acheté au moins un produit de couleur verte :
{(c) | ∃n (VENTE (-, c, n, -, -) ∧ PRODUIT (n, -, -, "Verte"))}
(Q6) Donner les noms des clients ayant acheté tous les produits stockés :
{(c) | ∀p (PRODUIT (p, -, -, -) ⇒ VENTE (-, c, p, -, -))}
(Q7) Donner les noms des produits fournis par tous les fournisseurs et achetés par au moins un client :
{(p) | ∀f (ACHAT(-,-, -, -, f) ⇒ ACHAT (-, -, p, -, f))
∧ ∃c VENTE (-, c, p, -, -)}

Conclusion

En somme, les mathématiques sont à la base de nombreuses disciplines et sciences actuelles et ont joué un grand rôle dans les concepts mathématiques d'aujourd'hui car, en effet, les premiers informaticiens étaient des mathématiciens qui ont voulu automatiser certains processus de calculs ; ce qui donna naissance aux calculateurs devenus ordinateurs de nos jours.

Les bases de données relationnelles suivent ainsi cette logique car se reposant sur des concepts mathématiques a l'instar de l'algèbre relationnelle, du calcul des prédicats (logique du premier ordre) et de la théorie des ensembles ; cette dernière ayant permis l'introduction des concepts radicalement nouveaux comme les fonctions et surtout les relations qui ont une part importante dans l'établissement des bases de données relationnelles.

Eu égard a ce qui précède, il en résulte que l'on ne saurait dissocier ces théories citées plus hauts car étant des disciplines connexes.

Webographie

[1] https://fr.wikipedia.org/wiki/Th%C3%A9orie_des_ensembles

[2]http://bliaudet.free.fr/IMG/pdf/IPI_MAT_120_Ensemble_BD_cours_complet.pdf

[3] Wikipedia https://fr.wikipedia.org/wiki/Alg%C3%A8bre_relationnelle

[4] Université de côte d'Azur https://www.i3s.unice.fr/~edemaria/cours/c3.pdf

[5] OpenClassrooms https://openclassrooms.com/fr/courses/4449026-initiez-vous-a-lalgebre-relationnelle-avec-le-langage-sql

[6] https://sgbd.developpez.com/tutoriels/cours-complet-bdd-sql/?page=algebre-relationnelle