République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Akli Mohand Oulhadj de Bouira

Faculté des Sciences et des Sciences Appliquées

Département d'Informatique

Mémoire de Master

en Informatique

Spécialité : *Ingénérie des systèmes d'information et de logiciel*

Thème

Conception et réalisation d'un interpréteur d'algèbre relationnelle

---

Encadré par

— BAL Kamel

Réalisé par

— CHABANI Ayoub

— MAAREF GUEALIA Abdelhak

2018/2019

Remerciements

Nous voudrions par ce biais adresser nos sincères remerciements au dieu pour tout puissant dans sa grâce et sa miséricorde qui nous a accordé la santé, le temps et la force de réaliser ce mémoire de fin d'étude et nos remerciements à tous ceux qui ont contribué d'une manière ou d'une autre à la rédaction de ce mémoire.

Nous tenons à exprimer nos vifs remerciements à Mr BAL Kamel, de nous avoir encadré et pour ses conseils, ses motivations, sa disponibilité et sa volonté à nous aider à travers de multiples réunion.

Nous remerciements tous nos amis (es) et collègues de promotion pour leurs remarques et leurs critiques qui nous ont fait évoluer.

Dédicaces

Avec un énorme plaisir, un cœur ouvert et une immense joie, que je dédie ce modeste travail à :

Ma Mère '' Tu m'as donné la vie, la tendresse, l'amour, et le courage pour réussir. En témoignage, je t'offre ce modeste travail pour te remercier pour tes sacrifices et pour l'affection dont tu m'as toujours entourée ".

Mon Père '' L'épaule solide, l'œil attentif compréhensif et le personne le plus digne de mon estime et de mon respect. Aucun dédicace ne saurait exprimer mes sentiment, que dieu te préserve et te procure santé et lange vie".

Ma chère famille : Farida, Fadhila, Ibrahim, Bilal, Chaima, Assia, Fatima pour leur encouragement contenu et leur soutient qu'ils trouvent ici l'expression de ma haute gratitude et bien sur les petits anges Rayan, Mahdi, Iyad.

Mon très cher amis Abdelhak et sa famille.

Mes chères amis et à ceux qui aimaient :Marzouk mohamed, Med_mrabet med_amine, Boudissa_Ilyes, Brahim_Errebai, Anis, Med_benyahia, Amayasse, Fateh, Abdellah, Djamel, Tarek, Malek, Yahia.

CHABANI Ayoub

Dédicaces

Je dédie ce travail

— A ma mère **Taib Saliha**, qui m'a encouragé à aller de l'avant et qui m'a donné tout son amour pour reprendre mes études

— A l'Homme, mon précieux offre du dieu, qui doit ma vie, ma réussite et tout mon respect : mon père **Mohammed**

— A mon petit frère **Rayane**

— A mon frère **Zine ellabidine** et sa femme **Farida** et son prince **Yanis**

— A ma sœur adorable **Zahoua** et son mari : **mohamed** et sa petite ange **Nélia**

— A mon grand père **athmane** et ma grande mère Baya

— a mon oncle **Sliman** et sa femme **Hasina** et son fils **Ayoub**

— A ma tante **Malika** et son mari **Ahmed**

— A mon collègue **Ayoub** et **sa famille**

— A mes amis proches : **Mohamed marzouk , Boudissa Dahmane Lyes, Med Merabet Med Amine ,Mohamed BenYahia ,Touati Djamel, Benguerah Anis, Abbas fateh ,Errebai Ibrahim ,Larik abdelmalek, dadoune amayasse,Tarek chellali**, sans oublié ma proche **Mahmoudi Houda**

MAAREF GUEALIA Abdelhak.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

AR	Algèbre relationnelle
SQL	Structured Query language
SGBD	Système de Gestion de Base de Données
UML	Unifed Modeling Language

Introduction générale

Nous présentons dans ce mémoire notre projet de fin de cycle Master en informatique, spécialité ingénierie des systèmes d'information et du logiciel. Notre projet consistait à réaliser un interpréteur d'algèbre relationnelle.

Comme il est connu et reconnu, l'algèbre relationnelle (noté ci-après AR) est une brique centrale et très importante dans le cours de bases de données. La bonne maitrise de ce langage est fondamentale pour l'assimilation du cours de bases de données et pour pouvoir ensuite comprendre les aspects avancés dans les bases de données, comme le langage SQl et l'optimisation des requêtes pour ne citer que ces deux aspects.

Comme c'est le cas avec tous les langages, l'apprentissage et la maitrise de l'AR est nettement plus efficace via la pratique. Cependant, si pour le langage SQL, nous disposons dans chaque SGBD d'un moteur d'exécution de requête SQL qu'on peut utiliser dans les séances de travaux pratiques, les outils d'apprentissage de l'AR en travaux pratiques fond défaut. Les étudiants et les enseignants intervenant dans les TP de bases de données ont toujours exprimé ce besoin de disposer d'un outil pédagogique sous forme d'un interpréteur (calculateur) d'algèbre relationnelle qu'ils peuvent utiliser pour apprendre et tester leurs connaissances en AR.

Nous nous proposons dans ce projet de développer sous forme d'un outil pédagogique, un interpréteur d'Algèbre relationnelle. Cet outil doit fournir aux enseignants et aux étudiants toutes les fonctionnalités nécessaires pour l'apprentissage de l'AR. Nous citons

parmi ces fonctionnalités :

— Translation d'une requête AR vers une requête SQL

— Analyse syntaxique de requêtes AR

— Analyse sémantique de requêtes AR

— Exécution de requêtes AR sur un dataset de données relationnelles.

— Sauvegarde de requêtes et de translation correspondantes.

Pour la présentation de notre projet, nous avons structuré ce mémoire en deux grandes parties comme suit :

• **Partie I : Etat de l'art :** cette partie est composée des chapitre suivants :

- Chapitre1 : Introduction aux bases de données et SGBD relationnels : dans ce chapitre nous présentons le domaine des bases de données et des SGBD relationnels, nous donnerons les définitions nécessaires, nous présentons aussi les propriétés des SQGBD ainsi que le modèle relationnel.

- Chapitre 2 : Algèbre relationnelle : ce chapitre sera consacré entièrement à notre sujet d'intérêt qui est l'Algèbre relationnelle, nous donnerons les définitions et présentons en détail toutes les opérations algébriques du langage.

• **Partie II : Interpréteur d'Algèbre relationnelle :** cette partie est consacrée à la présentation des étapes du développement de notre système. Elle est composée des chapitres suivants :

- Chapitre3 : Conception de l'interpréteur d'Algèbre relationnelle : il est question dans ce chapitre de présenter les différentes stratégies qui nous ont été offertes, une comparaison et un choix de stratégie à adopter. Nous donnerons enfin en détail une description de chaque composant de notre système.

- Chapitre 4 : Implémentation : nous avons présenté ici les outils et les logiciels utilisés pour développer notre interpréteur et nous avons donné un aperçu par des captures de l'interface.

Chapitre 1

Base de Données et SGBD relationnel

1.1 Introduction

Il est primordial, avant d'entamer toute démarche de conception ou de développement d(un système, de cerner et comprendre les concepts, les modèles et les théories liés au domaine d'étude concerné. Cette première parte du mémoire sera donc consacrée à la présentation de notre domaine d'étude qui est les bases de données relationnelles. Il sera question dans un premier temps de présenter les bases de données et les systèmes de gestion de bases données (SGBD). S'en suit après une présentation du modèle relationnel et enfin une présentation détaillée du langage d'algèbre relationnelle qui est le centre d'intérêt principal de notre projet.

1.2 Qu'est ce qu'une Base de Données ?

Une base de données est un ensemble structuré et organisé de données (informations) mis à la disposition des utilisateurs. On peut aussi définir une base de données comme étant une collection de fichiers ou de tables liées entre elles et répondant à un certain nombre de normes. Une base de données est donc un ensemble d'informations sur un sujet qui est : exhaustif, non redondant et structuré. [1]
Une base de données (en anglais database), permet de stocker et de retrouver l'intégralité de données brutes ou d'informations en rapport avec un thème ou une activité. [17]
Une base de données se traduit physiquement par un ensemble de fichiers présent sur une mémoire de masse (bien souvent un disque). [1]

1.3 Intérêt des bases de données

La base de données est au centre des dispositifs informatiques de collecte, mise en forme, stockage et utilisation d'informations. Le dispositif comporte un système de gestion de base de données (abréviation : SGBD) : un logiciel moteur qui manipule la base de données et dirige l'accès à son contenu. De tels dispositifs comportent également des logiciels applicatifs, et un ensemble de règles relatives à l'accès et l'utilisation des informations. [17]

L'utilisation des bases de données présentent de nombreux intérêts comme :

- Éviter les redondances et les incohérences des données qu'entrainerait une approche où ces dernières seraient dans fichiers sans lien entre eux.
- Assurer l'indépendance d'entre données et traitement : séparation des données et traitements.
- Offrir des langages de haut niveau comme SQL pour la définition et manipulation des données. Cela permettra la manipulation des données par des non informaticiens.
- Partage des données entre utilisateurs.
- De contrôler l'intégrité, la sécurité et la confidentialité des données.
- Minimiser les coûts investis dans les supports de stockage de données (mémoires).[1]

1.4 Système de Gestion de Base de Données (SGBD)

Un système de gestion de bases de données (SGBD) est défini comme un ensemble de logiciels prenant en charge la structuration, le stockage, la mise à jour et la maintenance des données. Autrement dit, il permet de décrire, modifier, interroger et administrer les données. C'est en fait, l'interface entre la base de données et les utilisateurs .[2]

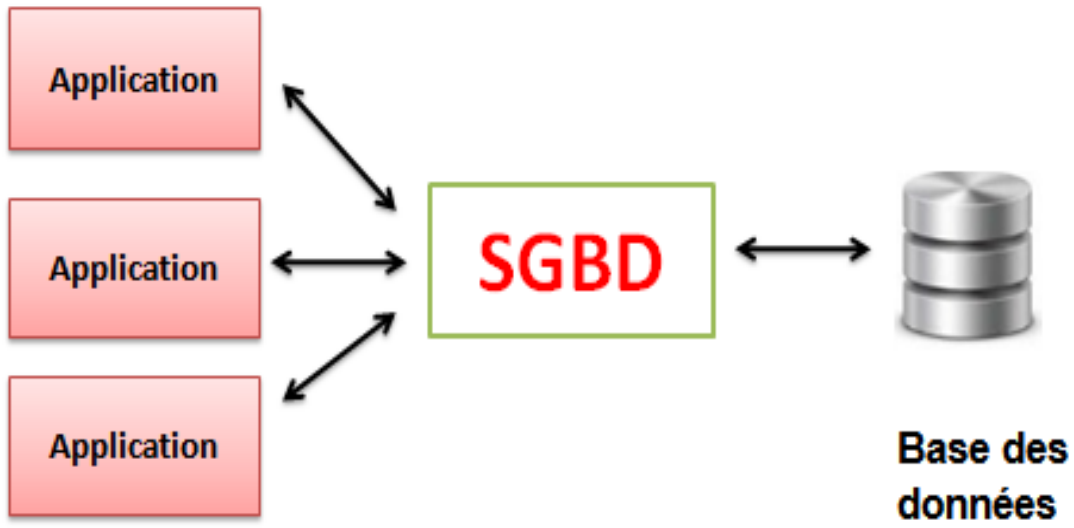

FIGURE 1.1 – Systeme de gestion de base de données

1.5 Objectifs d 'un SGBD

Les objectifs que l'on assigne généralement aux bases de données et aux systèmes qui les supportent sont les suivants :

- L'Indépendance physique : c'est l'indépendance entre la structure des données et la structure du stockage. La façon de définir les données doit être indépendante des structures utilisées pour leur stockage.
- L'Indépendance logique : un utilisateur doit pouvoir percevoir seulement la partie des données qui l'intéresse (c'est ce que l'on appelle une vue) et modifier la structure de celle-ci sans remettre en cause la majorité des applications.
- Manipulation aisée des données par des non informaticiens, ce qui suppose des langages "naturels"
- Accès efficaces aux données et obtention de résultats aux interrogations en un temps "acceptable"
- Administration centralisée : des données pour faciliter l'évolution de leur structure
- Non-redondance : chaque donnée ne doit être présente qu'une seule fois dans la base afin d'éviter les problèmes lors des mises à jour Cohérence (ou intégrité) : les

données ne doivent présenter ni ambiguïté, ni incohérence, pour pouvoir délivrer sans erreur les informations désirées. Cela suppose un mécanisme de vérification lors de l'insertion, de la modification ou de la suppression des données

— Partage des données pour un accès multi-utilisateur simultané aux mêmes données. Il faut entre autres assurer un résultat d'interrogation cohérent pour un utilisateur consultant une base pendant qu'un autre la modifie

— Sécurité des données : robustesse vis-à-vis des pannes (il faut pouvoir retrouver une base "saine" en cas de plantage au cours de modifications) et protection par des droits contre les accès non autorisés.

— Reprise après panne : ainsi que des mécanismes permettant de pouvoir revenir à l'état antérieur de la base tant qu'une modification n'est pas finie (notion de transaction).[2]

1.6 Composants des SGBD

Un SGBD va donc posséder un certain nombre de composants logiciels chargés de :

— **La description des données :** au moyen d'un Langage de Définition de Données (LDD). Le résultat de la compilation est un ensemble de tables, stockées dans un fichier spécial appelé dictionnaire (ou répertoire) des données

— **La manipulation des données :** au moyen d'un Langage de Manipulation de Données (LMD) prenant en charge leur consultation et leur modification de façon optimisée, ainsi que les aspects de sécurité. [2]

— **L'interrogation des données :** au moyen d'un Langage d'interrogation des données (LID) permet d'établir une combinaison d'opérations portant sur des tables. [4]

— **Le contrôle des données :** au moyen d'un Langage de contrôle des données (LCD) permet de contrôler l'accès aux données d'une base de données (gérer les droits et la création des utilisateurs et affectation de leurs droits). [5]

— **Les accès concurrents aux données :** en minimisant l'attente des utilisateurs et en garantissant l'obtention de données cohérentes en cas de mises à jour simultanées. [2]

1.7 Niveaux de description des données

Pour atteindre certains objectifs assignés aux SGBD, notamment l'indépendance physique et l'indépendance logique, trois niveaux de description de données ont été définis. Ces niveau de description découle de la norme Ansi/Sparc sur l'architecture des SGBD. L'architecture Ansi/Sparc est une architecture fondamentale sur laquelle reposent les SGBD modernes, elle fut proposée en 1975 par Charles Bachman qui reçut le prix Turing pour ses travaux. L'architecture Ansi-Sparc est divisée en trois niveaux, celui du schéma interne (SI), celui du schéma conceptuel (SC) et celui des schémas externes (SE). La première innovation de l'architecture Ansi-Sparc est la distinction claire entre la représentation interne des données au niveau physique (structure de données) et la représentation logique de celles-ci. Une base de données est définie et manipulée via le niveau conceptuel (SC) sans avoir à se soucier des détails de l'implémentation physique (SI). [19]

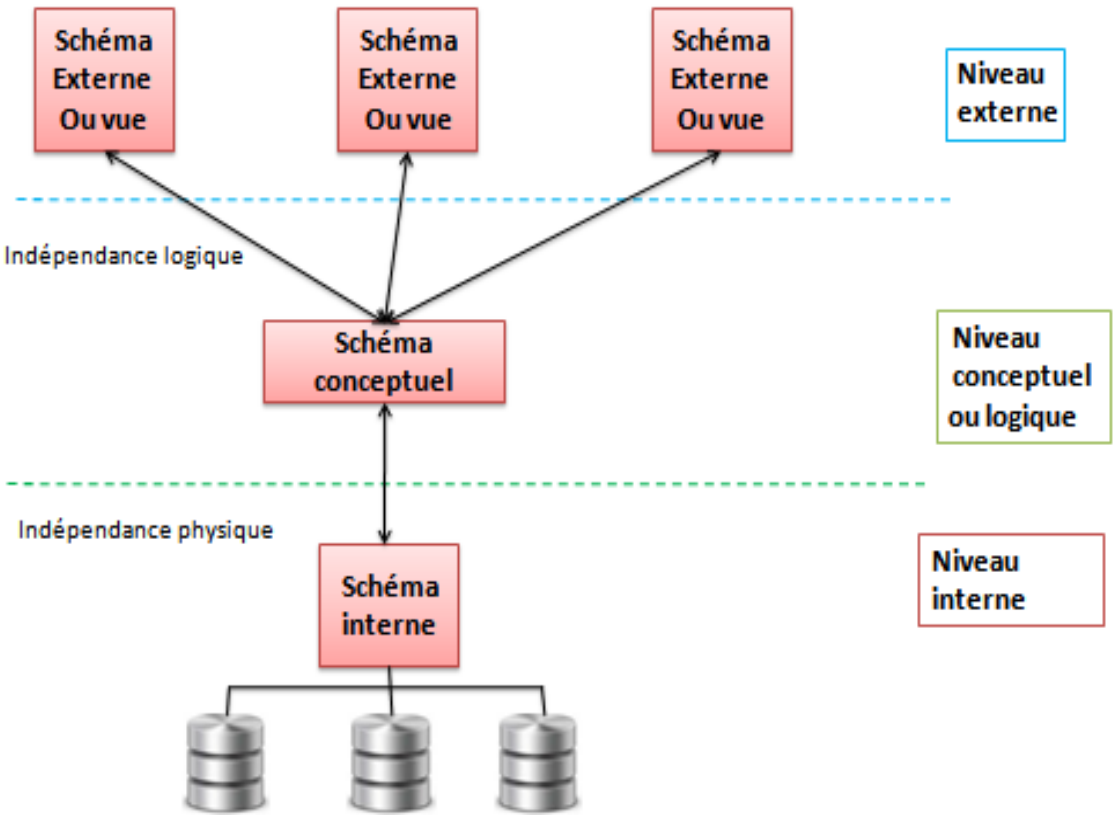

FIGURE 1.2 – archetecteur de Arsi/Spare

— **Niveau externe :** décrit comment chaque utilisateur perçoit les données, c'est ce que l'on appelle le schéma externe ou vue.

— **Niveau conceptuel** (ou logique) : décrit la structure des données dans la base,

leurs propriétés et leurs relations, indépendamment de toute préoccupation technologique d'implémentation ou d'accès par les utilisateurs, c'est ce que l'on appelle le schéma conceptuel

— **Niveau interne** (physique) : concerne le stockage des données au niveau des unités de stockage, des fichiers, c'est ce que l'on appelle le schéma interne. [2]

1.8 Le modèle relationnel

1.8.1 Définition du 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 Frank Codd est souvent retranscrite physiquement (« implémentée ») dans une base de données. [20]
Le modèle relationnel est un ensemble de concepts permettant de formaliser logiquement la description d'articles de fichiers plats, indépendamment de la façon dont ils sont physiquement stockés dans une mémoire numérique. Ce modèle a été inventé en 1970 par Edgar Frank Codd, le directeur de recherche du centre IBM de San José. Le modèle relationnel inclut des concepts pour la description de données, ainsi que des concepts pour la manipulation de données. [7]

1.8.2 Les objectifs du modèle relationnel

1. Assurer l'indépendance des applications et de la représentation interne des données
2. Gérer les problèmes de cohérence et de redondance des données
3. Utiliser des langages de données basés sur des théories solides(théorie des ensembles)
4. Être un modèle extensible permettant de modéliser et de manipuler simplement des données tabulaires, mais pouvant être étendu pour modéliser et manipuler des données complexes.
5. Devenir un standard pour la description et la manipulation des bases de données.[6]

1.8.3 Les 12 règles de Codd

Les 12 règles de Codd sont un ensemble de règles édictées par Edgar F. Codd, conçues pour définir ce qui est exigé d'un système de gestion de base de données (SGBD) afin qu'il puisse être considéré comme relationnel (SGBDR). [21]

1. L'information est représentée de façon logique sous forme de tables.
2. Les données doivent être accessibles de façon logique par les tables, les clés primaires et les colonnes.
3. Les valeurs nulles doivent être traitées uniformément comme des "informations absentes", et non pas comme des chaînes vides, ni des blancs, ni des zéros.
4. Les métadonnées doivent être stockées dans la base au même titre que les données normales.
5. Un langage unique doit permettre de définir les données, les vues sur ces données, les contraintes d'intégrité, les autorisations d'accès, les transactions, et enfin, la manipulation des données.
6. Les vues doivent refléter les mises à jour de leurs tables de base, et vice-versa.
7. Chaque action suivante doit pouvoir être réalisée par une et une seule action : Retrouver, Insérer, Modifier et Supprimer une donnée.
8. Il doit y avoir séparation logique entre les opérations et le stockage physique des données et leurs méthodes d'accès.
9. Les opérations peuvent modifier le schéma de la base de données sans qu'elle n'ait à être recréée, et sans que les applications construites au-dessus d'elle n'aient à être réécrites.
10. Les contraintes d'intégrité doivent être disponibles, et stockées dans les métadonnées et non pas dans un quelconque programme d'application.
11. Le langage de manipulation des données (LMD) ne doit pas se soucier d'où ni du comment les données sont stockées et/ou distribuées.
12. Le traitement d'une ligne doit respecter les mêmes règles et contraintes d'intégrité que les opérations portant sur des ensembles.[3]

1.8.4 Extension du modèle relationnel

Le modèle relationnel est un modèle standard, normalisé par l'ISO à travers son langage SQL. Il se veut néanmoins dès l'origine extensible, pour permettre de gérer des données plus complexes que les données tabulaires. Le modèle relationnel-objet est né de cette extension. [7]

1.8.5 Propriétés d'un SGBDr

Des objectifs cités ci-dessus découlent les propriétés fondamentales d'un SGBDr :

— Base formelle reposant sur des principes parfaitement définis.

— Organisation structurée des données dans des tables interconnectées (d'où le qualificatif relationnelles), pour pouvoir détecter les dépendances et redondances des informations.

— Implémentation d'un langage relationnel ensembliste permettant à l'utilisateur de décrire aisément les interrogations et manipulation qu'il souhaite effectuer sur les données.

— Indépendance des données vis-à-vis des programmes applicatifs (dissociation entre la partie "stockage de données" et la partie "gestion" - ou "manipulation").

— Gestion des opérations concurrentes pour permettre un accès multi-utilisateur sans conflit.

— Gestion de l'intégrité des données, de leur protection contre les pannes et les accès illicites. [2]

1.8.6 Les concepts fondamentaux du modèle relationnel : attributs, enregistrement, domaine et relation

1. **Domaine**
   — **Définition :** Ensemble caractérisé par un nom, dans lequel des données peuvent prendre leurs valeurs.
   — **Exemple de Domaines définis en intention :** Entier, Réel, Booléen, Chaîne de caractères, Monétaire (réel avec deux chiffres après la virgule), Date (chaîne de 10 caractères comprenant des chiffres et des tirets selon le patron "00-00-0000")

— **Exemple de Domaines définis en extension :** Couleur (Bleu, Vert, Rouge, Jaune, Blanc, Noir), SGBD (Hiérarchique, Réseau, Relationnel, Objet, Relationnel-Objet)

2. **Attribut et enregistrement**

— **Définition Attribut :** On appelle attribut d'une relation une colonne de cette relation. Un attribut est caractérisé par un nom et un domaine dans lequel il prend ses valeurs.

- Sont Synonymes : Champs, Propriété, Colonne

- Un attribut se distingue d'un domaine car il peut ne comporter que certaines valeurs de ce domaine.

- Les colonnes de la relation ne sont pas ordonnées et elles ne sont donc repérées que par le nom de l'attribut.

— **Définition Enregistrement :** On appelle enregistrement d'une relation une ligne de cette relation. Un enregistrement prend une valeur pour chaque attribut de la relation.

- Sont Synonymes : Tuple, N-uplet, Vecteur, Ligne

- Un enregistrement peut ne pas avoir de valeur pour certains attributs de la relation, parce que cette valeur est inconnue ou inapplicable, sa valeur est alors "nul".

3. **Relation**

Concept fondamental dans le modèle relationnel. Une relation est un sous ensemble du produit cartésien d'un ensemble fini de domaine caractérisé par un nom. Une relation R sur les domaines D1, D2 ,..., Dn , est un sous ensemble du produit cartésien D1 × D2 ×...× Dn

R ⊂ D1 × D2 × ... × Dn D1,, D2,, ... , Dn sont appelés les domaines de la relation R Une relation est donc un ensemble de vecteurs.

Exemple 1 :

Domaine : Pays = (France, Italie, Japon) et domaine Devise = (euro, yen)

Pays × Devise = ((France, euro), (France, yen), (Italie, euro), (Italie, yen), (Japon, yen), (Japon, euro))

R=Devise_Courante = ((France, euro), (Italie, euro), (Japon,yen)).

R ⊂ Pays x Devise

Exemple 2 :

Les domaines :

NOM = (Zaoui, Badaoui)

PRENOM = (Ali, Nora, Larbi)

DATE_NAISS = (Date entre 1/1/1990 et 31/12/2020)

SPORT = (judo, tennis, foot)

La relation : Eleve $\subset$ nom_elv $\times$ pren_elv $\times$ date_naiss

Eleve =((zaoui, ali,1/1/1992), (badaoui, nora , 2/2/1994))

La relation : inscription $\subset$ nom_elv $\times$ sport

inscription = ((zaoui, judo), (zaoui, foot), (badaoui, tennis))

Pour visualiser facilement le contenu d'une relation on utilise la représentation tabulaire. Chaque ligne correspond à un vecteur (ou enregistrement). Chaque colonne correspond à un domaine.

Exemple : Devise_Courante $\subset$ Pays $\times$ Devise

Pays	**Devise**
France	Euro
Italie	Euro
Japon	Yen

TABLE 1.1 – représentation tabulaire

4. **Schéma d'une relation**

Le schéma d'une relation est défini par :

- le nom de la relation

- la liste de ses attributs On note : R (A1, A2, ... , An)

Exemple :

- Eleve (nom, prenom, naiss)

- Produit (num_pdt, des_pdt,coul_pdt)

Le schéma d'une base de données est défini par l'ensemble des schémas des relations qui la composent. le schéma de la BD dit comment les données sont organisées

1.8.7 Les contraintes d'intégrités dans le modèle relationnel :

Pour assurer l'intégrité des données, le modèle relationnel a prévu un certain nombre de contraintes d'intégrité qu'on peut définir sur les données. Ces contraintes sont vérifiées par le SQBD à chaque manipulation des données. Une Contrainte d'Intégrité (CI) est une règle qui doit être vérifiée au moment de la création et de la manipulation de données afin que le résultat soit considéré correct et cohérent. Ces contraintes peuvent être :

— Contraintes de structure (contraintes de clés)

— Contraintes de référence (contrainte de clé étrangère)

— Contraintes d'unicité ou Contraintes sur les valeurs des attributs

Clés d'une relation :

Une clé est un groupe d'attributs minimum qui permet d'identifier de façon univoque un tuple dans une relation.

Toute relation doit comporter au moins une clé, ce qui implique qu'une relation ne peut pas contenir deux tuples identiques.

Afin d'être déterminants pour l'identification d'un enregistrement, tous les attributs d'une clé doivent être values, c'est-à-dire qu'aucun ne peut avoir de valeur nul. Dire qu'un groupe d'attribut est une clé équivaut à dire qu'il est unique et non nul. [9]

Si R est une relation décrite par un ensemble d'attributs A, un sous-ensemble K est une clé de R si :

- pour tout couple de tuples (t, t') de R, t(K) $\sqcup$ t'(K)
- où t(K) est la liste des valeurs données par le tuple t aux attributs composant K.

1. **Clé primaire et clés candidates**

   — Clé primaire : Si plusieurs clés existent dans une relation, on en choisit une parmi celles-ci. Cette clé est appelée clé primaire. La clé primaire est généralement choisie de façon à ce qu'elle soit la plus simple, c'est à dire portant sur le moins d'attributs et sur les attributs de domaine les plus basiques (entiers ou chaînes courtes typiquement). [9]

   — Clés candidates : On appelle clés candidates l'ensemble des clés d'une relation qui n'ont pas été choisies comme clé primaire (elles étaient candidates à cette fonction). [9]

   Par convention, la clé primaire est soulignée dans le schéma de la relation.

Exemples :

- Acteur (nom, prénom, numéro).

- Casting (film, acteur, personnage).

o Le même personnage peut être joué dans plusieurs films.

2. **Clé étrangère**

— Définition : Une clé étrangère est un attribut ou un groupe d'attributs d'une relation R1 devant apparaître comme clé primaire dans une relation R2 afin de matérialiser une référence entre les tuples de R1 et les tuples de R2. Une clé étrangère d'un tuple référence une clé primaire d'un autre tuple. [9]

— Contrainte d'intégrité référentielle : Une clé étrangère respecte la contrainte d'intégrité référentielle si sa valeur est effectivement existante dans la clé primaire d'un tuple de la relation référencée, ou si sa valeur est nul. Une clé étrangère qui ne respecte pas la contrainte d'intégrité référentielle exprime un lien vers un tuple qui n'existe pas et donc n'est pas cohérente. [9]

Par convention, les clés étrangères sont précédées d'un « # » dans le schéma d'une relation.

- Exemples

Pays (nom, # numero_devise, capitale, superficie)

Devise (numéro_devise, nbpieces, nbbillets, nom)

Le Pays référence sa monnaie via la clé primaire de la relation Devise.

Les valeurs de l'attribut numero_devise de la relation Pays sont les valeurs de l'attribut numéro_devise de la relation Devise.

Devise(4, 8, 8, 'euro'),

Devise (1,9,8,'Dollars US')

Pays ('France', 4, 'Paris', 544435)

Pays (USA,1,Washington, 78898676)

1.9 Conclusion

Après avoir vu un aperçu sur la base de données, SGBD et modèle relationnelle en détaillant de près ses titres avec des exemples simple et utiles. Maintenant on passe au chapitre suivant dans lequel on va parlé sur l'Algèbre relationnelle.

Chapitre 2

Algèbre relationnelle

2.1 Introduction :

Après avoir présenté dans les chapitres précédents la base de données, SGBD et modèle relationnelle en détail. Dans ce chapitre on va entamé la partie qui concerne l'algèbre relationnelle et ces opération

2.2 Définition d'algèbre relationnelle :

Est une méthode d'extraction permettant la manipulation des tables (ou relations) et des colonnes. Son principe repose sur la création de nouvelles tables (tables résultantes) à partir de tables existantes, ces nouvelles tables devenant des objets utilisables immédiatement.[10]

2.3 Les opérations dans l'algèbre relationnel :

— Opérations unaires : sélection, projection
— Opérations binaires : union, différence, produit cartésien
— Autres opérations : qui s'expriment en fonction des 5 opérations de base : jointure (naturelle, θ-jointure), intersection, division. [11]

2.3.1 Opérations unaires

1. **Sélection (RESTRICTION)**

   Définition : La sélection produit, à partir d'une relation, une relation résultante de même schéma mais ne comportant que les tuples qui répondent à la condition précisée en argument. On Sélection la condition C et On garde les nuplets qui satisfont C. Elle correspond à un découpage horizontal. [13]

   Notation :

   R2 = σ R1 (condition)

   Représentation graphique :

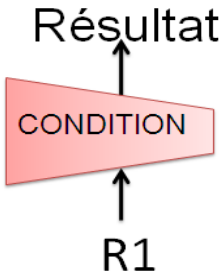

FIGURE 2.1 – Représentation graphique de Sélection

Exemple : « Quelles sont les produits de marque 'IBM' ? »

on a la table produit comme suite : PRODUIT (IdPro, Nom, Marque, Prix)

Idprod	**Nom**	**Marque**	**Prix**
P	P81	Ibm	10000
Q	MAC	Apple	2000
R	P82	Ibm	3000

TABLE 2.1 – exemple de table Produit

resultat = σ PRODUIT (Marque = 'IBM')

Idprod	**Nom**	**Marque**	**Prix**
P	P81	Ibm	10000
R	P82	Ibm	3000

TABLE 2.2 – exemplet de selection

2. **Projection :**

   Définition : La projection d'une relation R1 est la relation R2 obtenue en supprimant les attributs de R1 non mentionnés puis en éliminant éventuellement les nuplets identiques

   La projection permet d'éliminer des attributs d'une relation Elle correspond à un découpage vertical. [12]

   Notation :

   R2 = π R1 (Ai, Aj, ..., Am)

   Présentation graphique :

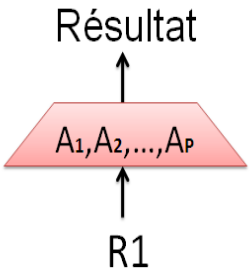

FIGURE 2.2 – Représentation graphique de projection

Exemple : PRODUIT (IdPro, Nom, Marque, Prix)

Idprod	**Nom**	**Marque**	**Prix**
P	P81	Ibm	10000
R	P82	Ibm	3000

TABLE 2.3 – table produit

Resultat = π PRODUIT (IdPro, Prix)

Idprod	**Prix**
P	10000
R	3000

TABLE 2.4 – Resultat de projection

2.3.2 Opérations binaires :

1. **Division :**

   Définition : Soit deux relations R1 (A1, A2, ... , An, B1, B2, ... , Bm) R2 (B1, B2, ... , Bm)

   Si le schéma de R2 est un sous-schéma de R1. La division de R1 par R2 est une relation R3 dont :

   le schéma est le sous-schéma complémentaire de R2 par rapport à R1

   un n-uplet (a1, a2, ... , an) appartient à R3 si (a1, a2, ... , an, b1, b2, ... , bm) appartient à R1 pour tous (b1, b2, ... , bm) $\in$ R2. [12]

   Notation :

   R3 = R1 / R2 la division de R1 par R2

   Présentation graphique :

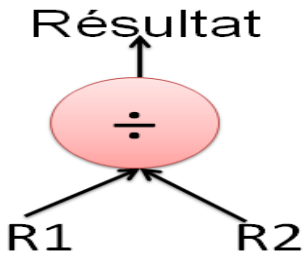

FIGURE 2.3 – Représentation graphique de la division

Exemple : Quels sont les élèves qui sont inscrits à tous les sports

inscript

Eléve	Sport
Toto	Judo
Tata	Dance
Toto	Foot
Toto	Dance

÷

Sport

Sport
Judo
Foot
Dance

$\Longrightarrow$

RES

Eléve
Toto

TABLE 2.5 – exemple de division

2. **Union :**

Définition : L'union entre deux relations de même structure (degré et domaines) donne une table résultante de même structure ayant comme éléments l'ensemble des éléments distincts des deux relations initiales. [10]

Notation : R3 = R1 ∪ R2

Présentation graphique :

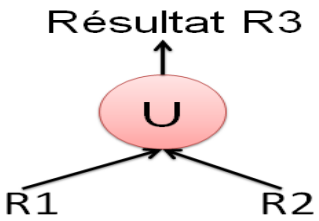

FIGURE 2.4 – Présentation graphique de l'union

Exemple :

R3 = R1 ∪ R2

R1

A	B
0	1
2	3

∪

R2

A	B
0	1
4	5

$\Longrightarrow$

R3

A	B
0	1
2	3
4	5

TABLE 2.6 – exemple de l'union

3. **Produit Cartésien**

   Définition : Le produit cartésien de deux relations R1 et R2 de schéma quelconque est une relation R3 ayant pour attributs la concaténation des attributs de R1 et de R2 et dont les tuples sont constitués de toutes les concaténations d'un tuple de R1 à un tuple de R2. [15]

   Notation :

   R3 = R1 × R2

   Présentation graphique :

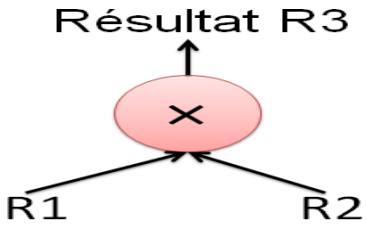

FIGURE 2.5 – Présentation graphique de Produit Cartésien

Exemple :

inscript

N	NOM	TEL
1	ALI	079999
2	AHMED	010101

×

Sport

c	DATE
2	01-9-1997

⟹

RES

N	NOM	TEL	C	DATE
1	ALI	079999	2	01-9-1997
2	AHMED	010101	2	01-9-1997

TABLE 2.7 – exemple de Produit Cartésien

2.3.3 Autres opérations :

1. **JOINTURE :**

   Définition : La jointure est une opération binaire entre deux relations R1 et R2 de schémas quelconques, qui permet d'associer, selon un critère donne portant sur au moins un attribut de chaque relation, les tuples de R1 et ceux de R2, afin de former une troisième relation R3 contenant l'ensemble de tous les tuples obtenus en concaténant chaque tuple de R1 et chaque tuple de R2 si ces deux tuples vérifient, ensemble, la condition d'association. [14]

C'est le seul operateur exploitant les attributs référentiels interrelations (clé primaire, clé étrangère).

Trois types de jointures :

— θ-jointure (theta est un critère de comparaison autre que '=')

— equi-jointure : jointure entre 2 relations avec critère d'Egalite (=) .

— Jointure naturelle : jointure entre 2 relations avec critère d'Egalite (equi-jointure) entre 2 attributs de même noms et fusion des colonnes de même nom(s).[14]

a. θ-jointure :

Définition : La θ jointure de deux relations R et S selon une qualification (condition)

Q est l'ensemble des Tuples du produit cartésien R qui satisfont à la qualification Q.

Il s'agit donc de la Restriction selon Q de R $\times$ S, c'est à dire σ Q (R $\times$ S). [14]

Notation :

JOIN (R, S, θ)

Représentation graphique :

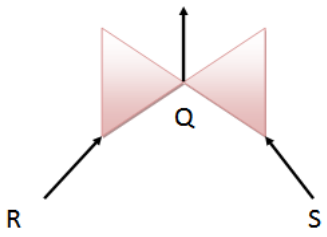

FIGURE 2.6 – Représentation graphique de jointure

Exemple :

R

A	B	C
a	d	b
b	e	g
c	f	c

S

D	E
f	d
b	e

$\Longrightarrow$

$R \bowtie (B < DetA! = C) \quad S$

A	B	C	D	E
a	d	b	f	d
b	e	g	f	d

TABLE 2.8 – exemple de θ-Jointure

b. L'equi-jointure :

Définition : L'équi-jointure de deux relations R et S est une jointure avec pour qualification Q l'égalité entre deux colonnes, c'est-à-dire : $R \bowtie (Ai = Bj) \quad S$

Avec Ai et Bj, deux attributs de R et de S respectivement.[14]

Exemple :

R

A	B	C
a	d	d
b	e	g
c	f	c

S

D	E
d	f
b	e

$\Longrightarrow$

$R \bowtie (B = D) \quad S$

A	B	C	D	E
a	d	d	d	f

TABLE 2.9 – exemple de equi jointure Jointure

c. La jointure naturelle :

Définition : La jointure naturelle de deux relations R et S est une équi-jointure sur tous les attributs de même nom dans R et dans S, suivie de la projection qui permet de ne conserver qu'un seul des cas attributs égaux de même nom.[14]

Notation :

joint (R,S)

Représentation graphique :

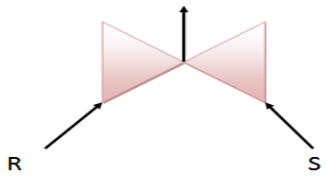

FIGURE 2.7 – Représentation graphique de jointure naturelle

Exemple :

R

A	B	C
a	d	s
b	e	g
c	f	c

S

A	B	D
a	d	d
a	d	g
c	f	c

$\Longrightarrow$

$R \bowtie S$

S	A	B	D
a	d	s	d
a	d	s	g
c	f	c	c

TABLE 2.10 – exemple de Jointure naturel

2. **Intersection**

Définition : L'intersection entre deux relations R1 et R2 de même schéma est une relation R3 de schéma identique ayant pour n-uplets les n-uplets communs à R1 et R2. [12]

Notation :

R3 = R1 ∩ R2

Présentation graphique :

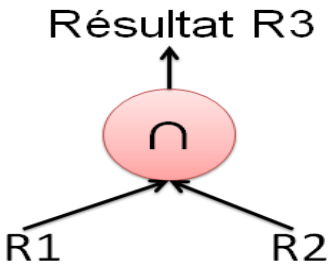

FIGURE 2.8 – Présentation graphique de l'insertion

Exemple :

inscript

N	NOM	TEL
1	ALI	079999
2	AHMED	01010

$\cap$ Sport

N	NOM	TEL
2	AHMED	01010

$\Longrightarrow$ RES

N	NOM	TEL
2	AHMED	01010

TABLE 2.11 – exemple d'intersection

3. **Différence**

Définition : La différence entre deux relations R1 et R2 de même schéma est une relation R3 de schéma identique ayant pour n-uplets les n-uplets de R1 n'appartenant pas à R2. [12]

Notation :

R3 = R1 - R2

Présentation graphique :

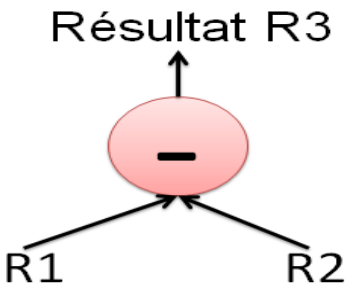

FIGURE 2.9 – Présentation graphique de différence

Exemple :

inscript

N	NOM	TEL
1	ALI	079999
2	AHMED	01010

- Sport

N	NOM	TEL
2	AHMED	01010

$\Longrightarrow$ RES

N	NOM	TEL
1	ALI	079999

TABLE 2.12 – exemple de différance

2.4 Le langage algébrique :

2.4.1 Présentation :

Le langage algébrique permet de formuler une question par une suite d'opérations de l'algèbre relationnelle. [15]

2.4.2 Comment construire une requête algébrique ?

1. Identifier les relations utiles pour exprimer la requête,
2. Recopier le schéma de ces relations, et indiquer sur ces schémas :
   — Les attributs qui font partie du résultat de la requête
   — Les conditions portant sur les attributs
   — Les liens entre les relations
3. Traduire cette figure en expression algébrique
   — faire les sélections selon les conditions portant sur les attributs
   — faire les jointures (naturelles ou thêta) selon les liens entre les relations (une jointure par lien)
   — projeter sur les attributs qui font partie du résultat. [15]

Exemple :

ACTEUR(NA,NOM,PRENOM,ADRESSE,SEXE)

VESTE(NV,MARQUE,COULEUR,TAILLE)

PORTE(NA,NV,DATE,DUREE)

Q1 :Donner les marques des vestes de taille 32 et de couleur rouge

R1 = RESTRICT(VESTE,TAILLE=32)

R2 = RESTRICT(VESTE,COULEUR='ROUGE')

R3 = INTERSECT(R1,R2)

RESULT=PROJECT(R3,MARQUE)

Q2 :Donner les noms et prénoms des acteurs qui ontmis des vestes rouges ou bleues

R1 = RESTRICT(VESTE,COULEUR='ROUGE')

R2 = RESTRICT(VESTE,COULEUR='BLEU')

R3 = UNION(R1,R2)

R4 = JOIN(R3,PORTE)

R5 = JOIN(R4,ACTEUR)

RESULT=PROJECT(R5,NOM,PRENOM)

Q3 :Donner les noms et prénoms des acteurs qui ont mis des vestes de taille 32 plus de deux heures, avec la marque de la veste.

R1 = RESTRICT(VESTE,TAILLE=32)

R2 = RESTRICT(PORTE,DUREE¿2)

R3 = JOIN(R1,R2)

R4 = PROJECT(R3,NA,MARQUE)

R5 = JOIN(R4,ACTEUR)

RESULT=PROJECT(R5,NOM,PRENOM,MARQUE)

2.4.3 Arbre algébrique :

1. **Présentation :** est une schéma des nœuds qui représente les opérations algébriques et des arcs qui représente les relations de base ou temporaires.[15]

2. **Exemple :**

R1 = JOIN(Acteur, Casting)

R2 = RESTRICT(Film, realisa = 'Besson')

R3 = RESTRICT(Film, realisa = 'Beni')

R4 = JOIN(R1, R2)

R5 = JOIN(R1, R3)

R6 = PROJECT(R4, nom, prenom)

R7 = PROJECT(R5, nom, prenom)

RESULTAT = MINUS(R7 , R6)

---

3. **L'arbre :**

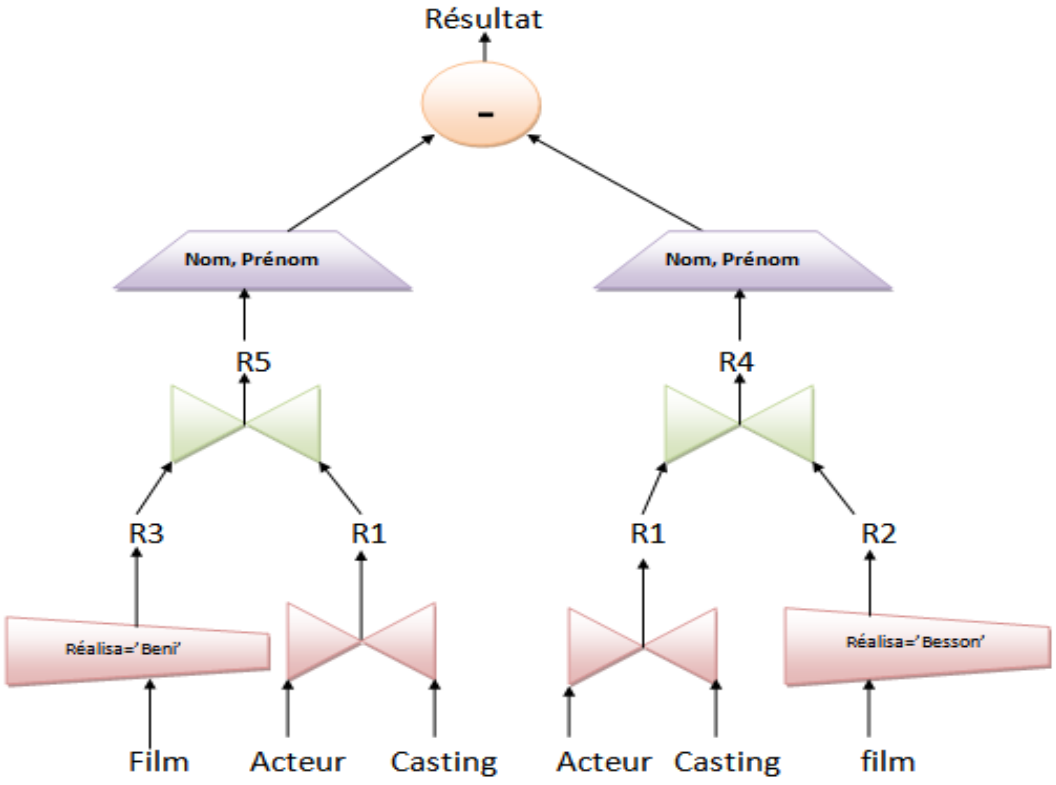

FIGURE 2.10 – exemple d'une arbre

2.4.4 Fonction et agrégats :

1. **Fonction :** Fonction de calcul en ligne appliquée sur un ou plusieurs attributs. [15]

   Exemple : prixttc = prixHT*1,17

2. **Agrégats :** Un agrégat est un partitionnement horizontal d'une relation selon des valeurs d'attributs, suivi d 'un regroupement par une fonction de calcul. Les fonctions de calcul usuelles sont :
   — **SUM (somme) :** pour calculer la somme sur un ensemble d'enregistrement
   — **COUNT (Compte) :** pour compter le nombre d'enregistrement sur une table ou une colonne distincte et pour compter le nombre total de ligne d'une table il convient d'utiliser l'étoile « * » qui signifie que l'on cherche à compter le nombre d'enregistrement sur toutes les colonnes.
   — **AVG (Moyenne) :** pour calculer la moyenne sur un ensemble d'enregistrement
   — **MAX (Maximum) :** pour récupérer la valeur maximum d'une colonne sur un ensemble de ligne. Cela s'applique à la fois pour des données numériques ou alphabétiques
   — **MIN (Minimum) :** pour récupérer la valeur minimum d'une colonne sur un ensemble de ligne. [15]

2.5 Pourquoi les SGBD utilisent l'Algèbre Relationnelle ?

L'Algèbre Relationnelle est principalement utilisé pour faire de l'optimisation de requête. En effet, pour une requête SQL, il existe plusieurs requêtes algébriques équivalentes (i.e. plusieurs requêtes en algèbre relationnelle qui retournent les mêmes tuples en résultat). Ces requêtes algébriques sont alors appelées plans d'exécution. Autant les tuples retournés sont les mêmes pour tous ces plans d'exécution, autant le temps nécessaire pour obtenir ces tuples, lui peut être très différent. Tout dépend de la façon dont les opérations qui composent le plan d'exécution sont organisés. [16]

2.6 Conclusion :

Après avoir vu un aperçu sur l'Algèbre relationnelle en détaillant de près ses titres avec des exemples simple et utiles. Maintenant on passe au chapitre suivant dans lequel on va citer les différentes stratégies d'implémentation des opérations d'algèbres relationnelle, et bien choisir la stratégie appropriée pour notre application.

Chapitre 3

Conception de l'Interpréteur

3.1 Introduction

Après avoir présenté dans les chapitres précédents, une étude bibliographique sur le domaine des bases de données et particulièrement sur l'algèbre relationnelle, nous allons à présent, nous intéresser à la conception de notre interpréteur d'Algèbre relationnelle. Nous rappelons ici que l'objectif principal de ce projet c'est de développer sous forme d'un outil pédagogique, un interpréteur d'algèbre relationnelle. Il est attendu de cet outil de :

- Implémenter les différents opérateurs d'algèbre relationnelle.
- Permettre aux étudiants d'apprendre facilement via une interface graphique, l'algèbre relationnelle.
- Aux étudiants et enseignants intervenant dans les séances de travaux pratiques de bases de données de disposer d'un logiciel permettant de mettre en œuvre toutes les connaissances théoriques sur l'algèbre relationnelle.
- Permettre aux étudiants de tester leurs connaissances dans d'algèbre relationnelle.

3.2 Stratégie d'implémentation des opérations d'algèbres relationnelle :

Pour pouvoir implémenter les opérateurs d'algèbres relationnelle, deux stratégies peuvent être adoptées : la première consiste à translater chaque opération d'algèbre relationnelle vers son esquivant en langage SQL et la deuxième stratégie consiste à implémenter via des algorithmes toutes les opérations d'AR.

3.2.1 Stratégie 1 : translation vers SQL :

Dans cette stratégie, une translation équivalente vers le langage SQL doit être définie pour chaque opération d'algèbres relationnelle. Bien évidement cette stratégie nécessite l'utilisation d'un SGBD et d'une base de données pour l'exécution de la requête SQL résultante.

Exemple de translation :

1. **La projection :**
   o AR : PROJECT (R, a1,a2,a2,. . . ., an)
   o SQL : SELECT a1,a2,a2,. . . ., an FROM R

PROJECT(CLIENT, Numclient,nomclient)	SELECT Numclient, nomclient FROM Client

TABLE 3.1 – translation de projection

2. **La restriction :**
   o AR : RESTRICT (R,condition)
   o SQL : SELECT * FROM R WHERE condition
   Avec Condition = ai valeur
   - ai un attribut de R et
   - est un opérateur de comparaison : $\times$, $\neq$,$\leq$,$\geq$, = ,. . . . . etc

RESTRICT (CLIENT, Ville = 'Bouira')	SELECT * FROM Client WHERE Ville = 'Bouira'

TABLE 3.2 – translation de restrection

3. **Le produit cartésien :**
   o AR : PRODUCT (R1,R2)
   o SQL : SELECT * FROM R1,R2

PRODUCT (CLIENT,PRODUIT)	SELECT * FROM CLIENT, PRODUIT

TABLE 3.3 – translation de produit cartésien

4. **La jointure :**

   o AR : JOIN (R1,R2, R1.a θ R2.b)

   o SQL : SELECT * FROM R1,R2 WHERE R1.a θ R2.b

   - A est un attribut de R1 et b un attribut de R2

   - θ est un opérateur de comparaison : $\times$, $\neq$,$\leq$,$\geq$, = ,. . . . . etc

JOIN (CLEINT, COMMANDE, COMMANDE.numclient = CLIENT.numclient)	SELECT * FROM CLIENT, COMANDE WHERE COMMANDE.numClient = Client.Numclient

TABLE 3.4 – translation de jointure

5. **Union :**

   o AR : union (R1, R2) OU R1 $\cup$ R2

   o SQL : SELECT * FROM R1 UNION SELECT * FROM R2

Union(magasin1client, magasin2client)	SELECT * FROM magasin1client UNION SELECT * FROM magasin2client

TABLE 3.5 – translation d'union

6. **Difference :**

   o AR : MINUS (R1,R2) ou R1-R2

   o SQL : SELECT * FROM R1 EXCEPT SELECT * FROM R2

MINUS(CLIENTinscrits, CLIENTrefus)	SELECT * FROM CLIENTinscrits EXCEPT SELECT * FROM CLIENTrefus

TABLE 3.6 – translation de différence

7. **Intersection**

   o AR : INTER (R1,R2) ou R1 ∩ R2

   o SQL :SELEC T * FROM R1 INTERSECT SELEC * FROM R2

INTER(magasin1client, magasin2client)	SELECT * FROM magasin1client INTERSECT SELECT * FROM magasin2client

TABLE 3.7 – translation d'Intersection

3.2.2 Stratégie 2 : Implémenter des algorithmes

Dans cette stratégie, consiste à implémenter via des algorithmes toutes les opérations d'AR, Bien évidement cette stratégie ne nécessite pas l'utilisation d'un SGBD et d'une base de données pour l'exécution d'algèbre relationnel par contre on nécessite un système de fichiers qu'est une façon de créer et stocker notre relations, attribut, données et les organiser dans des fichiers, après on le manipuler par des algorithmes pour obtenir des résultats.
On représente ci-dessus les différents algorithmes pour chaque opération d'AR.

Algorithmes de jointure :

La jointure est l'opération la plus coûteuse en temps de calcule. Elle peut être implémentée selon plusieurs algorithmes :

— Boucles imbriquées simples

— Tri-fusion

— Jointure par hachage

— Boucles imbriquées avec index

1. **Jointure par boucles imbriquées :**

   La jointure par boucles imbriquées est la méthode de base. Elle consiste à parcourir d'une manière imbriquée les deux tables concernées par la jointure et vérifier à chaque fois la condition de jointure entre les lignes des deux tables.

Algorithme 1 : Algorithme de jointure par boucles-imbriquées :

Entrées : tables R, S

Sortie : table de jointure J

```
Début
    J = ensemble vide
    Pour chaque r dans R répéter
        Pour chaque s dans S répéter
            Si r joignable à s alors J = J ∪ r ⋈ s
        Fin répéter
    Fin répéter
Fin
```

[22]

2. **Jointure par hachage**

   La jointure par hachage a le même principe que les boucles imbriquées avec M pages mémoire. Cet algorithme est plus sophistiqué, il se compose de deux parties. Tout d'abord, Une table de hachage H1 de l'une des relations (R ou S) est créée (la plus petite de préférence) avec un tableau associatif comportant des paires (clé, valeur). clé est la clé de hachage des valeurs des attributs en commun entre les deux relations. Valeur est une liste des numéros de lignes des tuples qui ont les mêmes valeurs donc la même clé (haché).

   Ensuite, on parcourt, par boucles imbriquées, chaque tuple de l'autre relation. On hach les valeurs des attributs en commun, puis on vérifie avec les clés de H1 l'existence de cette valeur de hachage dans la table de hachage H1. Si la valeur existe, on ajoute, à la table résultat, les tuples résultant de la fusion du tuple de la relation R avec chaque tuple dont le numéro de ligne est référencé dans la liste valeur de H1 correspondante.

Algorithme 2 : Algorithme de jointure par hachage :

```
Relation joinH(Relation R, Relation S)
var :
    resultat : Relation ; //la relation resultante de la jointure
    Hs : map de couple (int , liste des int) //la table associative de hachage de S
    h : valeur de hash
debut
     //part1
    pour chaque tuple t dans S faire
Hs[hash(t).pushback(t) ;
    fin pour
    //part2
    resultat =nouveau Relation ;
    pour chaque tuple t dans R faire
        h ← hash(t)
        si exist Hs[H]donc
            pour chaque tuple S dans Hs[h]faire
                ajouter (Resultat , fusion(t,S) ;
            fin pour
        fin si
    fin pour
    retourner resultat ;
fin
```

La Jointure par hachage est un peu plus efficace que les boucles imbriquées si les deux tables sont de grande taille

3. **Jointure par tri-fusion :**

Cet algorithme commence par trier les deux relations R et S sur les attributs à joindre dans l'ordre croissant (resp. décroissant). Il initialise un pointeur pour chacune des deux relations, puis il les pointe sur les premiers tuples de chaque relation. il avance le pointeur de R si la valeur du tuple pointé est inférieure (resp.

supérieure) à celle du pointeur de S, et il fait de même pour le pointeur de S. Dès qu'il y' a des valeurs identiques, il fait la fusion des deux tuples pointés, puis il sauvegarde le tuple résultant.

Algorithme 3 : Algorithme tri-fusion :

Entrées : tables R, S

Sortie : table de jointure J = R | >< | R.A=S.B S

Début

trier R sur l'attribut de jointure A

trier S sur l'attribut de jointure B

J = fusion (R, S)

Fin [22]

---

4. **Jointure avec une table indexée :**

   Si un index est présent sur la relation S, la deuxième boucle imbriquée est remplacée par un parcourt de l'index avec la valeur de tuple en cours et son algorithme le suivant :

Algorithme 4 : Algorithme boucles-imbriquées-index :

Entrées : tables R, S ; index sur S.B

Sortie : table de jointure J = R⋈ R.A=S.B S

Début

J = ensemble vide

Pour chaque r ∈ R répéter

Pour chaque s ∈ Index S.B (r.A) répéter

J = J ∪ r ⋈ R.A=S.B s

Fin répéter

Fin répéter

Fin [22]

Algorithme de sélection

Algorithme 5 : Algorithme de sélection :

```
Algorithme Select(R Q,) :
Debut
  Pour chaque page p de R faire
    lire(p) ;
    pour chaque tuple t et p
      Si Chek S(t,Q) then
        result = result ∪ t ;
      fin Si
    fin Pour
  fin Pour
fin [23]
```

3.3 Comparatif entre les deux stratégies :

	Les avantages	**Les inconvénients**
Translation vers SQL	- Moins compliqué pour l'implémentation - Pas besoin d'avoir une structure de stockage propre. - Toutes les opérations d'accès de manipulation et de recherche seront assuré par le SGBD. - Seul l'algorithme de translation à implémenter	- Problème essentiel : nécessité de disposer d'un SGBD.. - Variantes d'algorithmes dépends des SGBD. - Obtenu un résultat après la translation vers SQL et accéder à la base de données
Implémenter des	- Ne nécessite pas de disposer d'un SGBD. - Présente une solution avec une structure de données	- Compliqué pour le programmé - Pas d'algorithme toujours meilleur - Nécessite d'un modèle de cout pour choisir le meilleur algorithme

algorithmes	dédiée qui peut fonctionner sur n'importe environnement. - Possibilité de tester plusieurs variantes d'algorithmes.	- Nécessite de gérer la structure des données et les algorithmes d'accès et de manipulation des données.

TABLE 3.8 – comparaison entre les stratégie

Après ce comparatif, nous avons choisi d'opter dans notre systèmes pour la stratégie de translation vers SQL vu les grand avantages qu'elle nous offre. Vu la contrainte du temps qui nous est imparti, cette stratégie nous évitera d'implémeneter une structure de données et toutes les opérations de mise à jours des données. De plus étant develiopper comme outil pédagogique dans le module de bases de données, l'utilisation d'un SGBD ne posera pas de problème. La maitrise du SQl et des fonction d'un SGBD font aussi partie du programme du cours de bases de données.

3.4 Architecteur de l'interpréteur d'AR :

Le schéma suivant implique l'architecteur globale de notre système. Nous rappelons que nous avons opté pour la stratégie de translation vers SQL.

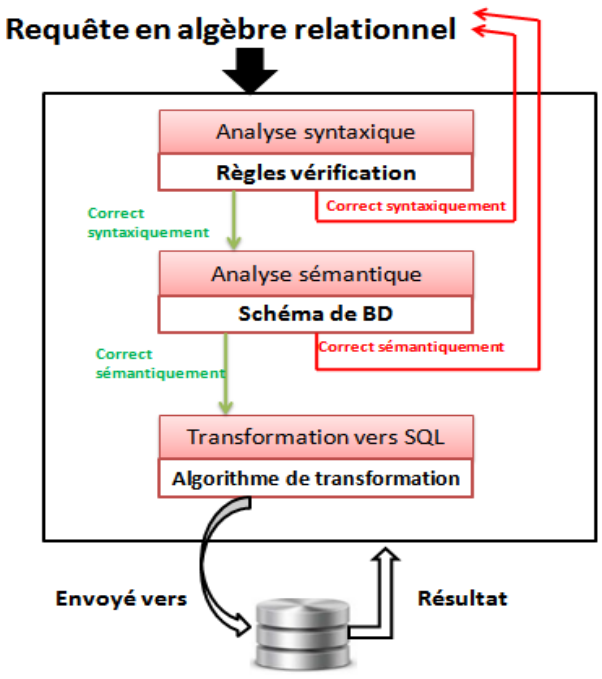

FIGURE 3.1 – Architecteur de l'interpréteur d'AR

3.4.1 Composant de notre interpréteur d'AR :

L'interpréteur d'AR que nous avons développer est constitué des composants suivants :

— L'analyse syntaxique

— Analyse sémantique

— Transformation d'algèbre relationnel ver SQL (conversion automatique)

1. **L'analyse syntaxique**

   Ce composant sert à analyser syntaxiquement la requête AR et vérifier qu'elle respecte la syntaxe prévue.

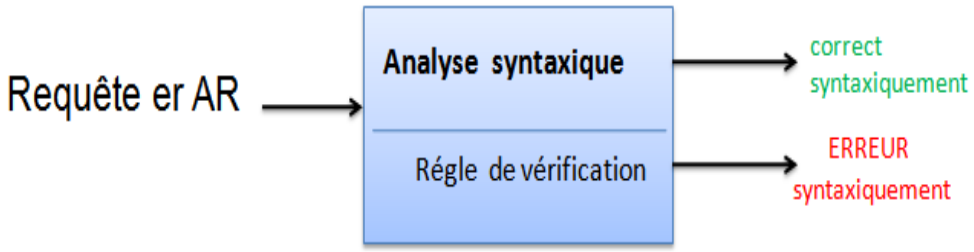

FIGURE 3.2 – vérification syntaxique

Règle de vérification :

Une requête algébrique bien écrite (correcte syntaxiquement) est généralement une expression sous la forme suivante :

QAR = ORERATION (P1,P2,[P3, P4...])

Où :

ORERATION ∈ (PROJECT,RESTRICT,JOIN,UNION,MINUS, INTER, PRODUCT, DIV)

Et P1, p2, p3, p4 des paramètres

En plus :

Si OPERATION = PROJECT Alors PROJECT (P1, P2)

Si OPERATION =RESTRICT Alors RESTRICT (p1, P2)

Si OPERATION = JOIN Alors

JOIN (P1,P2,P3) ou Sinon JOIN (P1, P2)

Si OPERATION = UNION Alors UNION (P1, P2)

Si OPERATION = INTER Alors INTER (P1, P2)

Si OPERATION = MINUS Alors MINUS (P1, P2)

Si OPERATION = PRODUCT Alors PRODUIT (P1, P2)

Si OPERATION = DIV Alors DIV (P1, P2)

SINON ERREUR

2. **Analyse sémantique :**

c'est une étape de l'interpréteur qui est responsable de détection les erreur sémantiques et des incohérences (vérification des noms des relations , vérification des noms des attributs vérification des condition dans les sélection...). Pour cela cet analyseur doit disposer des informations sur le schéma de la base de données (Noms des relations, attributs de relations et types des attributs)

Règles de vérification sémantiques :

Soit IR l'ensemble des relations

Généralement notre expression est comme la syntaxe suivante :

QAR = ORERATION (P1,P2,[P3, P4...])

Si (opération = project) alors

P1 ∈ IR et P2 ∈ attribut de P1

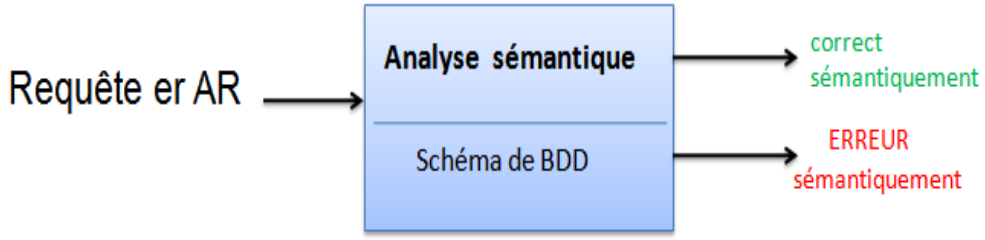

FIGURE 3.3 – vérification sémantique

Si (opération = restrict) alors

P1 ∈ IR et P2 = 'a θ valeur' où

a ∈ attribut de P1

$\theta \in (\neq, \leq, \geq, =)$

Type valeur = Type de a

Si (opération = join) alors

Si P3 = NULL alors

P1 ∈ R et P2 ∈ R

Sinon alors

P3 = P1.a θ P2.b

a ∈ attribut de P1

b ∈ attribut de P1

$\theta \in (\neq, \leq, \geq, =)$

Si (opération = union or minus or inter) alors

schéma de P1 = schéma de P2

Si (opération = div) alors

Schéma de P2 $\subset$ Schéma de P1

Si (opération = PRODUCT) alors

P1 ∈ R et P2 ∈ R

3. **Transformation d'algèbre relationnel ver SQL**

Apres l'analyse syntaxique et sémantique on a besoin d'un algorithme pour convertie les blocs algèbre relationnel en SQL

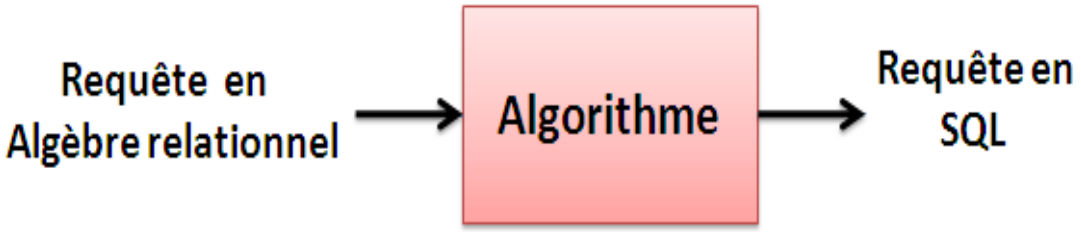

FIGURE 3.4 – schéma de transformation

Algorithme de transformation

Entrée : opération (P1, p2, [p3])

Si operation = PROJECT

alors req_SQL= select P2 from P1

Si operation = RESTRICT

Alors req_SQL= select * from p1 where p2

Si operation= JOIN

Si p3 is not null

Alors req_SQL = select * from p1,p2 where p3

Sinon

A= rechercher l'attribut commun (P1,P2)

Req_SQL= select * from p1,p2 where P1.A = P2.B

Si operation = UNION

Alors req_SQL= select * from p1 UNION select * from p2

Si operation = MINUS

Alors req_SQL= select * from p1 EXCEPT select * from p2

Si operation = INTER

Alors req_SQL= select * from p1 INTERSECT select * from p2

SI operation= PRODUIT

Alors req_SQL= select *from P1,P2

3.5 Modélisation conceptuelle de l'application

Pour modéliser un système d'information Il y a plusieurs méthodes parmi elles on peut citer Merise et méthode Itérative qui sont les plus utilisées.
Après une petite recherche sur les avantages et inconvénients des méthodes les plus utilisées on décidé de choisir le langage UML qui est le mieux adapté pour notre cas.

UML, c'est l'acronyme anglais pour (Unifed Modeling Language). UML est un langage visuel constitué d'un ensemble de schémas, appelés des diagrammes, qui donnent chacun une vision différente du projet à traiter. UML nous fournit donc des diagrammes pour représenter le logiciel à développer : son fonctionnement, sa mise en route, les actions Susceptibles d'être effectuées par le logiciel.[32]

Nous allons présenter notre application par deux vues fonctionnel et dynamique.

3.5.1 Vue fonctionnelle

La vue fonctionnelle est modélisée par le diagramme des cas d'utilisation qui permet d'identifier les possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est à dire toutes les fonctionnalités que doit fournir le système.

Les acteurs :

- **Utilisateur** : c'est toute personne qui peut utiliser le système pour apprendre l'AR et tester ses connaissances. C'est généralement l'étudiant dans notre cas.
- **Administrateur** : c'est la personne qui s'occupe du paramétrage de l'application. Dans notre cas, c'est l'enseignant chargé du TP qui peut gérer les dataset utilisés dans l'application.

Les uses cases

- Soumettre une requête
- Choisir un dataset
- Enregistrer une requête
- Afficher le schéma d'un dataset

— Afficher les données des tables

— Sauvegarder la translation d'une requête

Diagramme de cas d'utilisation d'administrateur

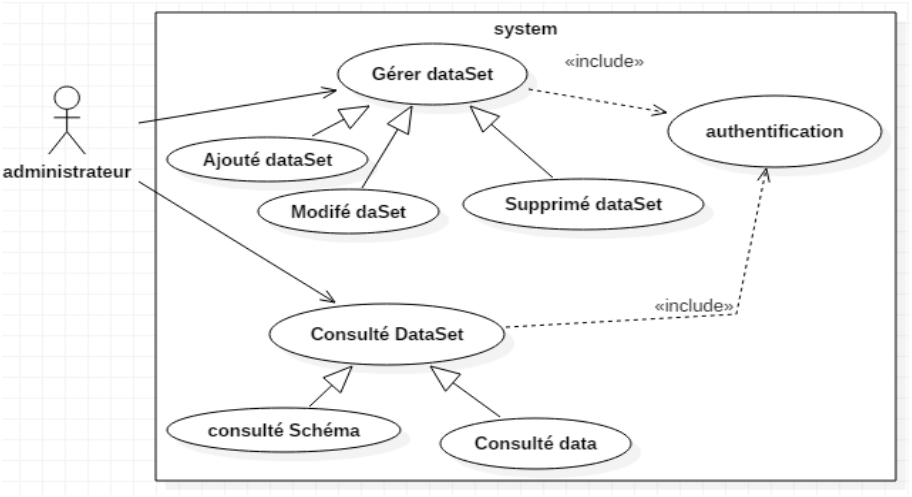

FIGURE 3.5 – Diagramme de cas d'utilisation d'administrateur

Diagramme de cas d'utilisation d'utilisateur

3.5.2 Vue dynamique

La vue dynamique est modélisée par le diagramme de séquence qui décrit les interactions entre un groupe d'objets en montrant, de façon séquentielle, les envois de message qui interviennent entre les objets. Les diagrammes de séquences permettent de d'écrire comment les éléments du système interagissent ente eux et avec les acteurs, ils mettent l'accent sur la chronologie des messages.[31]

Diagramme de séquence pour l'ajouter une base de données

Ce cas d'utilisation permet à L'administrateur d'ajouter une base de données.

Enchainement nominal :

— L'administrateur demande la fenêtre pour ajouté une BD

— Le système affiche la fenêtre

— L'administrateur saisit les données .

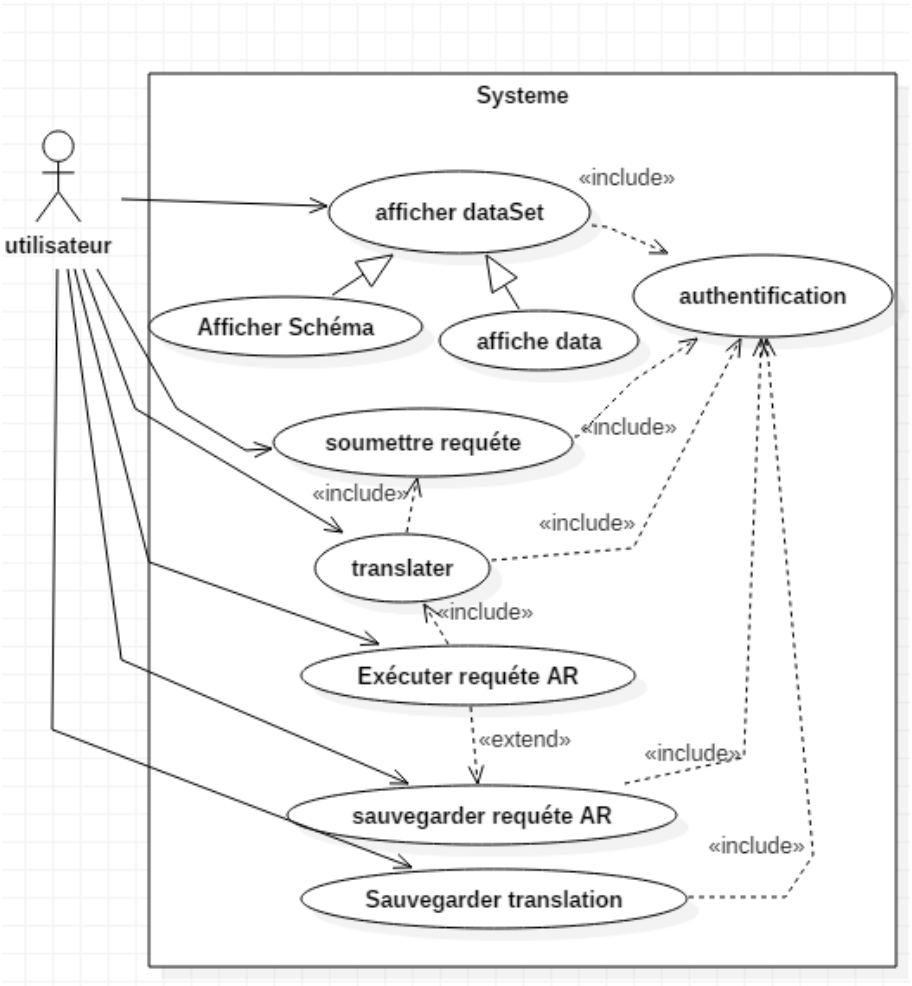

FIGURE 3.6 – Diagramme de cas d'utilisation d'utilisateur

— Le système vérifie les données au niveau de BD .

— Affichage d'un message de confirmation avec succès dans le cas ou le nom de BD n'existe pas sinon une message d'erreur.

La figure suivante montre le déroulement de l'enchainement :

Diagramme de séquence pour l'exécution d'une requête

Ce cas d'utilisation permet à L'utilisateur à exécuté requête d'algèbre jusqu'à l'obtient de résultat.

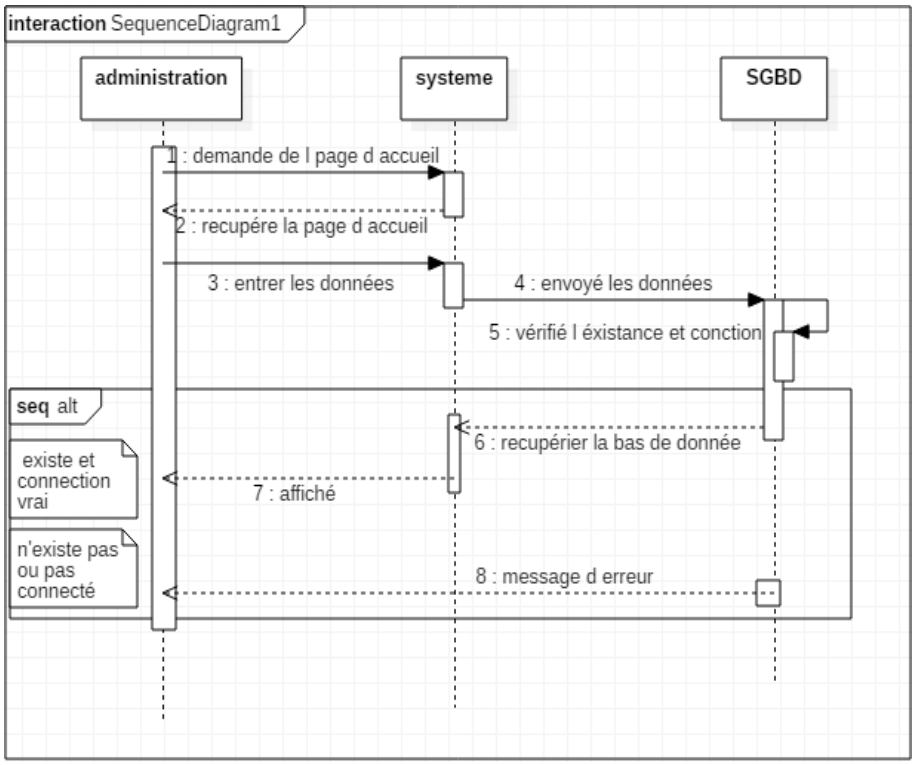

FIGURE 3.7 – Diagramme de séquence pour l'ajouter une BD

Enchainement nominal :

- L'utilisateur saisie la requête AR
- Le système vérifie la syntaxe de requête, un message d'erreur s'affiche à l'utilisateur si la syntaxe est fausse, sinon il demande le schéma de la base de données.
- Le système vérifie la requête sémantique-ment, un message d'erreur s'affiche à l'utilisateur en cas d'erreur, sinon la requête doit être translaté en SQL, et exécuter dans la base de données.
- Le système affiche le résultat à l'utilisateur.

La figure suivante montre le déroulement de l'enchainement :

3.6 Conclusion :

Jusqu'à présent on a parlé des stratégies d'implémentation des opérations d'algèbres relationnelle, selon les inconvénients et les avantages de chacune on a pu choisir la stratégie idéale pour notre interpréteur.

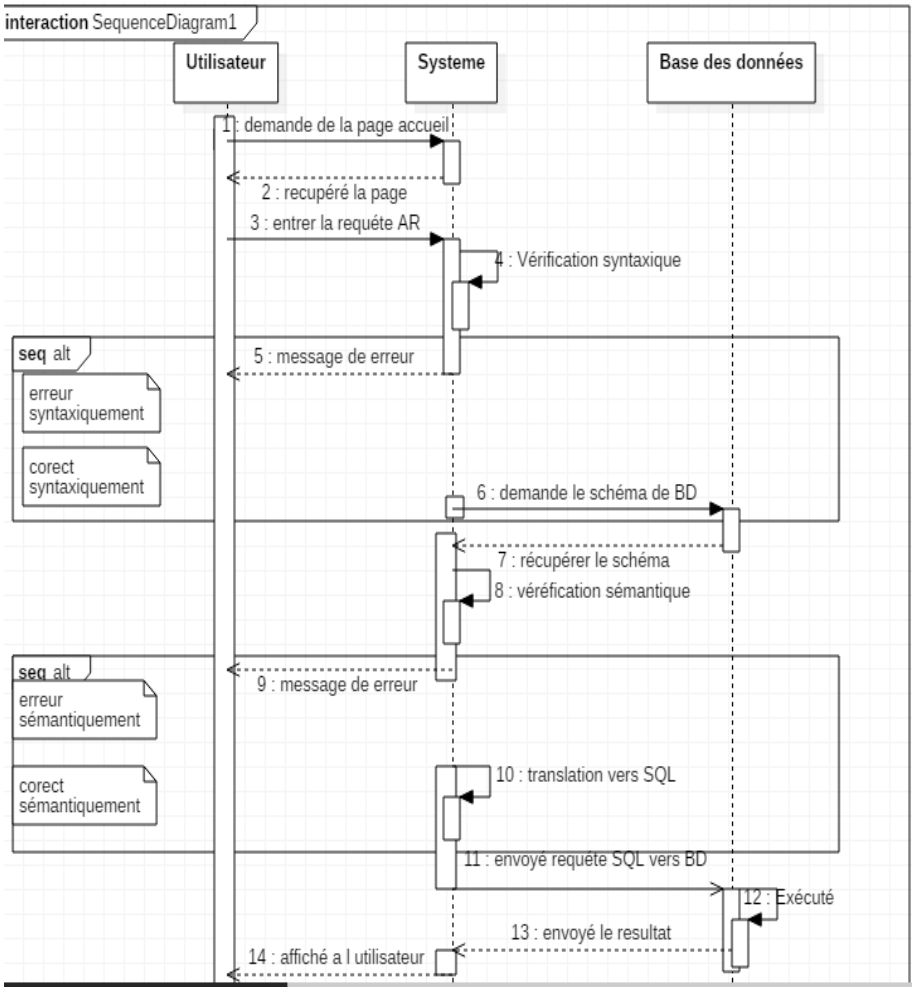

FIGURE 3.8 – Diagramme de séquence pour l'exécution d'une requête

Pour ce dernier on a mis en œuvre des algorithmes de translation utiles et on décidé de choisir le langage UML qui est le mieux adapté pour notre cas Nous allons présenter notre application par deux vues fonctionnel et dynamique.

Dans le prochain chapitre nous allons présenter l'environnement du travail et une vue réelle sur notre application.

Chapitre 4

Implémentation de l'Interpréteur

4.1 Introduction

Nous allons présenter dans ce dernier chapitre le matériel, logiciel et langage de programmation nécessaire pour l'implémentation de notre application, et nous présenterons ses différentes fonctionnalités à travers de l'application.

4.2 Environnement de travail

4.2.1 Environnement matériel :

Durant ce présent projet, tout le travail a été réalisé sur deux ordinateurs qui ont les caractéristiques techniques suivantes :

Ordinateur 01 :

Marque	HP 650
Processeur	Inter® core™ i3-2328M CPU@ 2.20GHz 2.20 GHz
RAM	2,00 Go
System d'exploitation	Windows 7 professionnel 64 bits

Table 4.1 – caractéristique de l'ordinateur N :01

Ordinateur 02 :

Marque	HP
Processeur	Inter® core™ i3-4005U CPU@ 1.70GHz 1.70 GHz
RAM	4,00 Go
System d'exploitation	Windows 7 professionnel 64 bits

TABLE 4.2 – Caractéristique de l'ordinateur N :02

4.2.2 Les logiciels utilisés :

IntelliJ IDEA est un environnement de programmation spécial ou un environnement de développement intégré (IDE) largement destiné à Java. Cet environnement est utilisé notamment pour le développement de programmes. Il est développé par une société appelée JetBrains, qui s'appelait officiellement IntelliJ. Il est disponible en deux éditions : l'édition communautaire sous licence Apache 2.0 et une édition commerciale connue sous le nom d'Ultimate Edition.[27]

XAMPP est un ensemble de logiciels permettant de mettre en place facilement un serveur Web et un serveur FTP. Il s'agit d'une distribution de logiciels libres (X Apache MySQL Perl PHP) offrant une bonne souplesse d'utilisation, réputée pour son installation simple et rapide. [25]

StarUML est un logiciel de modélisation UML open source qui peut remplacer dans bien des situations des logiciels commerciaux et coûteux . Étant simple d'utilisation, nécessitant peu de ressources système, ce logiciel constitue une excellente option pour une familiarisation à la modélisation. [30]

TeXstudio est un éditeur LaTex indépendant de la plate-forme qui vous offre des possibilités avancées pour rédiger vos textes, telles que la vérification interactive orthographe, grammaire, syntaxe, la coloration syntaxique, le pliage de code, la complétion automatique, ...etc.[24]

4.2.3 Langages de programmation utilisés

Le SQL : est un langage permettant de communiquer avec une base de données. Ce langage est notamment très utilisé par les développeurs web pour communiquer avec les données d'un site ou d'une application. [29]

Kotlin est un langage de programmation orienté objet, fonctionnel avec un typage statique. Il tourne sous la machine virtuelle Java (JVM) et peut aussi être compilé sous JavaScript, le nom Kotlin vient de l'ile Kotline au large de St Pétersbourg en Russie. [28]

Java est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton en 1991 et publié en 1995 par Sun Microsystems (acquis par Oracle en 2010), avec l'intention que les programmeurs n'écrivent le code qu'une seule fois et l'exécutent sur n'importe quel appareil.

JavaFx Ce langage peut être utilisé dans des applets intégrées à des pages Web ou dans des applications Java. Il est concis et sert à définir visuellement une interface graphique et à l'associer aux fonctions de l'application.[26]

4.3 Présentation de l'application

4.3.1 Architecture de l'application

voila l'architecture de notre application :

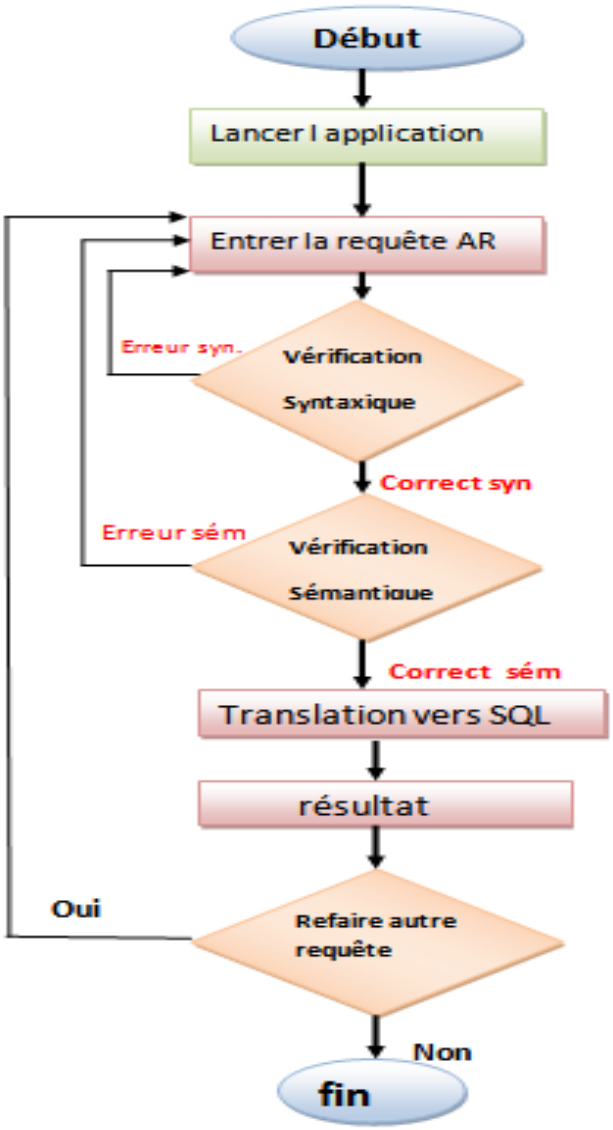

FIGURE 4.1 – Architecture de l'application

4.3.2 L'interface d'application :

il y a une seul interface principale dans notre application qui est la suivant :

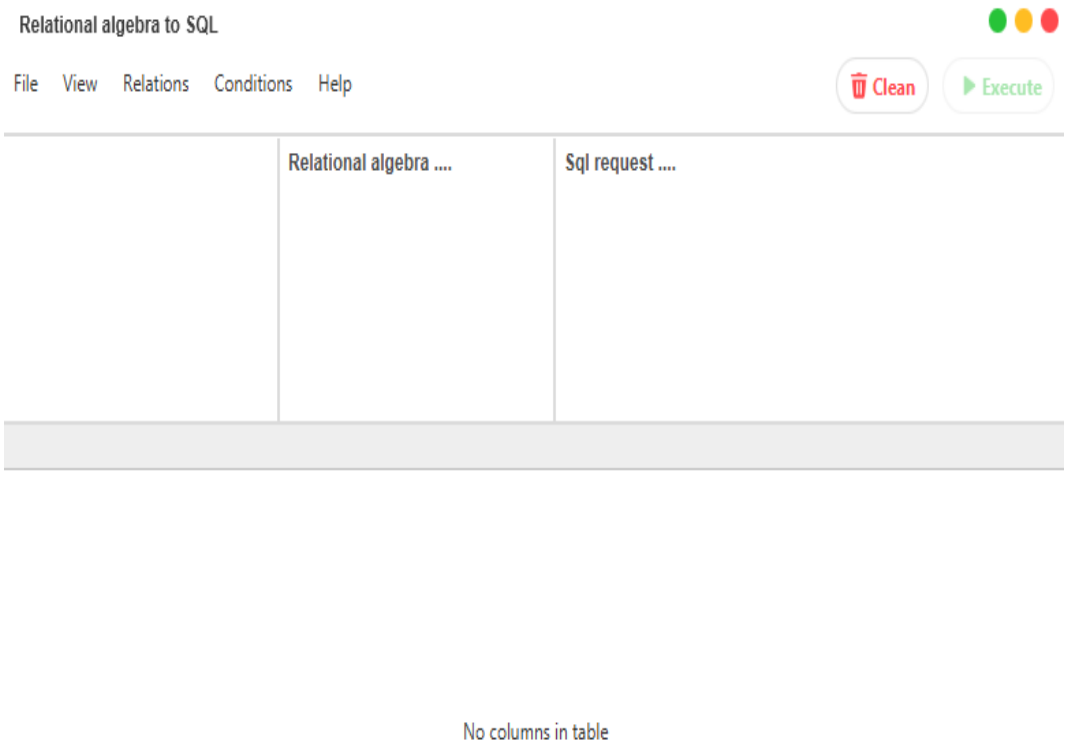

FIGURE 4.2 – l'interface principale

Le travail est fait dans cette interface, elle contient plusieurs zones et chaque zone est responsable d'une tache

- Zone 1 : barre d'outils
- Zone 2 : zone de base de données
- Zone 3 : Champ Algèbre relationnel
- Zone 4 : Zone de commande
- Zone 5 : Zone SQL
- Zone 6 : Zone résultat

la figure suivant reprisent les déférents zone de l'interface de notre application :

FIGURE 4.3 – les différents zone de l'interface

4.3.3 Les zones de l'application

- **Zone 01 : Barre d'outils**

Cette zone contient plusieurs options (file, view, Relations, condition et help)

FIGURE 4.4 – Barre d'outils

1. **File :**

   cette option et responsable pour ajouter (connecté) un base des données, déconnecté un BD et sortie de la application.

FIGURE 4.5 – Barre d'outils

— Pour **connecté** a base de donnée (ajouté un base de donnée a l'application) , clique sur Connect et remplir les informations dans le formulaire (nom, user name et password)

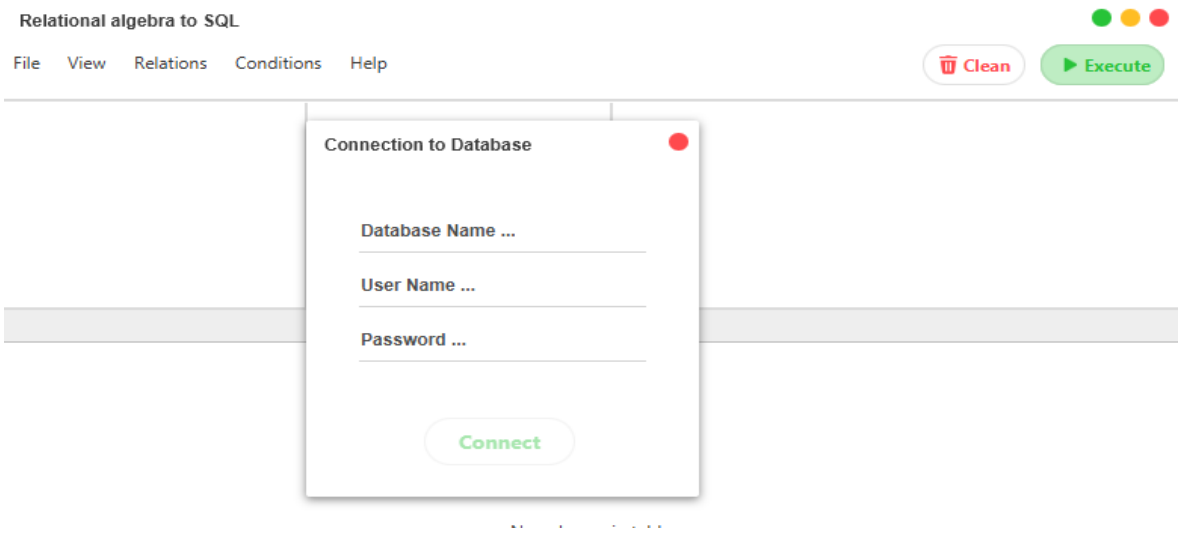

FIGURE 4.6 – formulaire de connexion

Exemple : ajouté la base des donnée ≪Test ≫

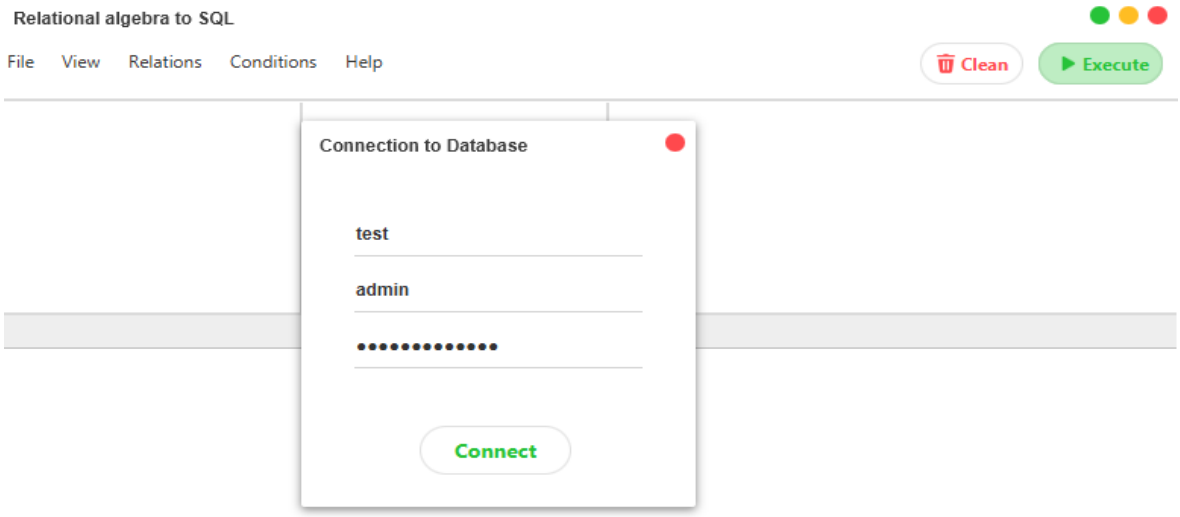

FIGURE 4.7 – exemplaire pour connecté une BD

Et voila le résultat : la base des données ≪ test ≫ a été ajouté dans la zone2 (Zone base de données existe)

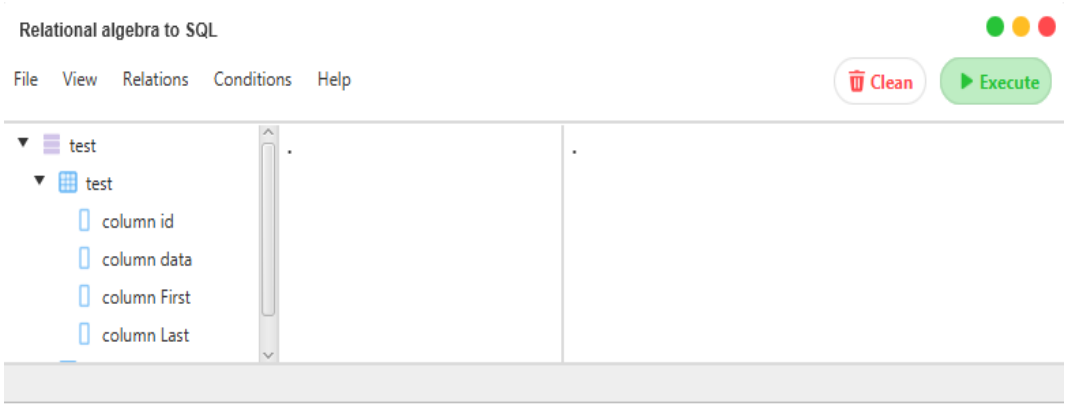

FIGURE 4.8 – affichage de BD et ces tables

— Pour **déconnecté** a n'import base des donnés cliqué sur desconnect .

— Pour **sortie** de l'application cliqué sur close .

2. View : cette option et responsable de quelque chose dans l'interface :

- hide tree : pour affiche le zone 2(barre de BD)

- show tree : pour masquer la zone 2

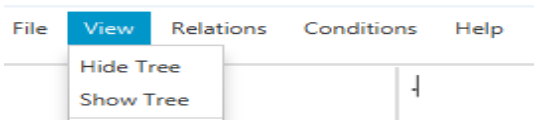

FIGURE 4.9 – View

3. Relation : cette option contient toutes les relations d'algèbre relationnel (project, restrict, union ....)

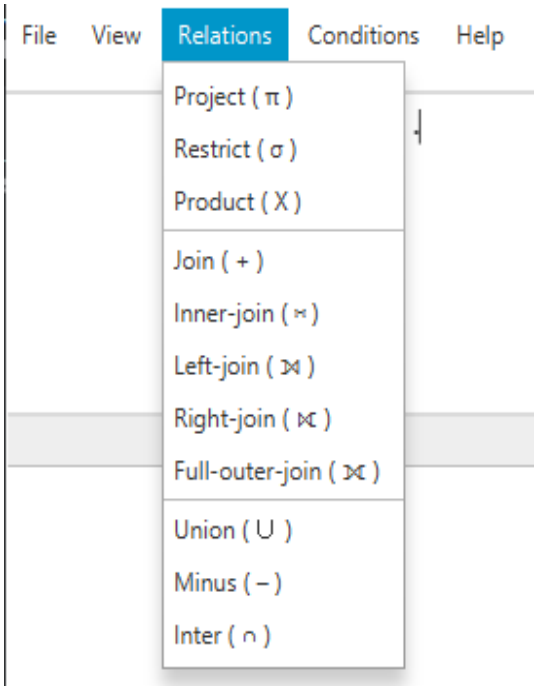

FIGURE 4.10 – Relation

Lorsque clique sur quelque relation, il affiche automatiquement sur la zone 3(zone algèbre relationnel) avec l'emplacement des paramètres, pour facilité a l'utilisateur d'écrire des requête AR correct et évité l'erreur syntaxique.

Exemple : clique sur « product »

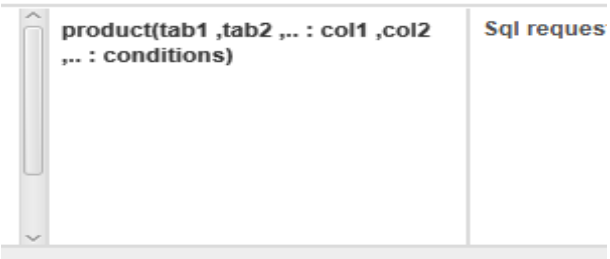

FIGURE 4.11 – Exemple d'une relation

4. Conditions : cette option contient des opérations (and, or, mod,=, ...) pour la condition, pour facilité a l'utilisateur de crée les conditions.

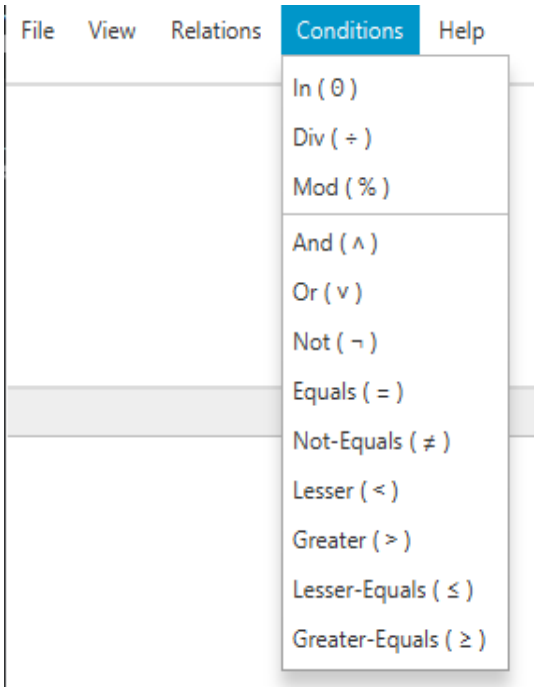

FIGURE 4.12 – Barre d'outils

- **Zone 2 : zone de base des données.**

Cette zone est responsable de l'affichage de base des données qui connecté avec sa schéma (table et les attributs de chaque une des tables)

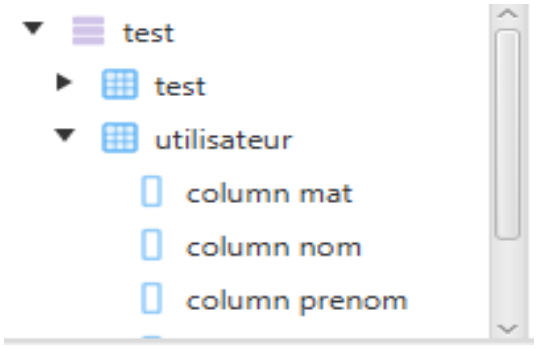

FIGURE 4.13 – champs de la saisie de l'Algèbre relationnelle

- **Zone 3 : champ d'algèbre relationnel**

Dans cette zone, on doit entrer la requête algèbre relationnel qui Que nous voulons exécuter

FIGURE 4.14 – champs de la saisie de l'Algèbre relationnelle

- **Zone 4 :Zone de commande**

Cette zone contient 2 bouton : clean et Execute

FIGURE 4.15 – zone de commande

- **Bouton clean** : :, dans le cas d'erreur ou pour renouvellement de requête en peut vider les champs (Zone algèbre relationnel, SQL, résultat) par cliqué sur cette Botton

- **Bouton execute** : lorsque en clique sur cette Botton La requête va être exécuté (donné la en requête SQL, et la résultat de requête)

- **Zone 5 : Zone SQL**

C'est une zone responsable d'affiché le résultat après la translation vers SQL

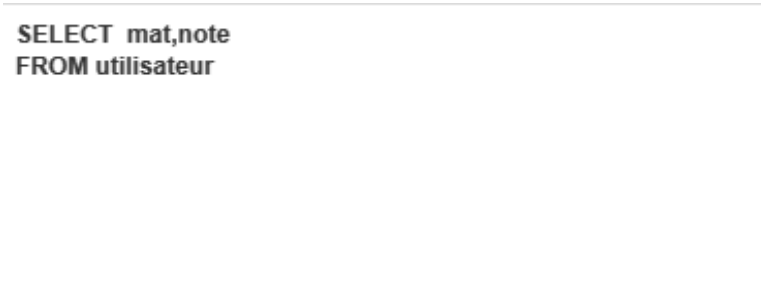

FIGURE 4.16 – champ de requête SQL translaté

• **Zone 6 : zone de résultat** C'est une zone responsable d'affiché le résultat après l'exécution de la requête SQL sur la base des données sous forme d'un tableau.

FIGURE 4.17 – Résultat retourner

4.3.4 Exemple real de l'exécution de requête AR

Exécuté la requête « **affiché les matricules et les notes** » de « **utilisateur** » dans BD « **test** »

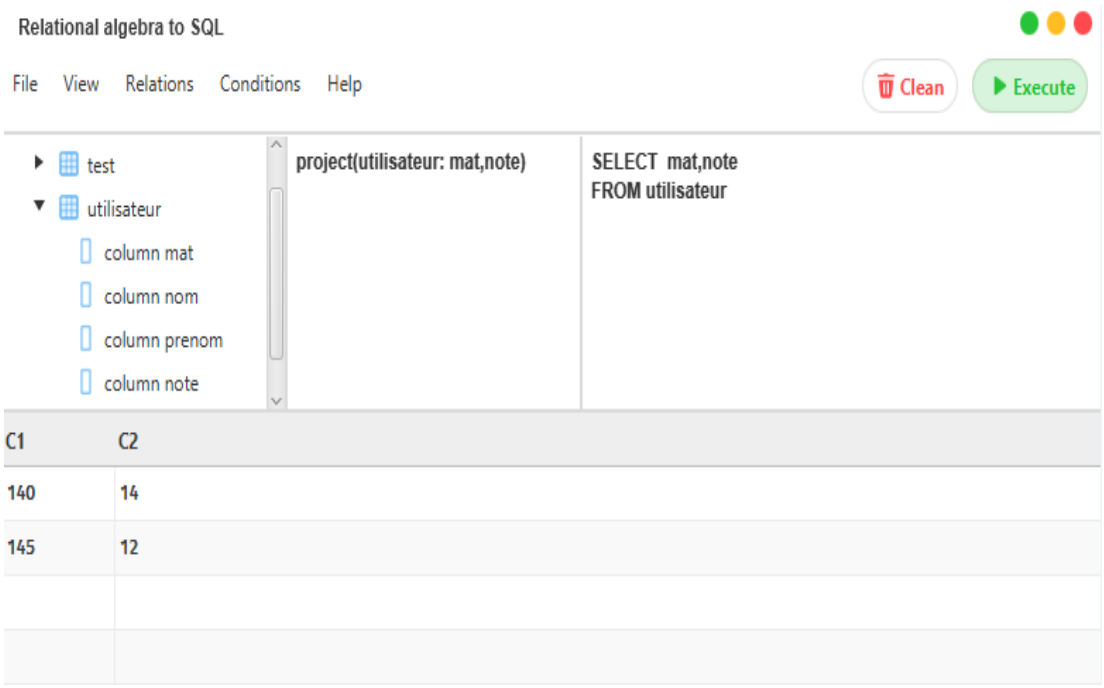

FIGURE 4.18 – Exemple d'exécution

A la fin en ajouté une chose spécial de nous, pour prendre un aperçu sur notre thème . En clique sur **help** apres **about**

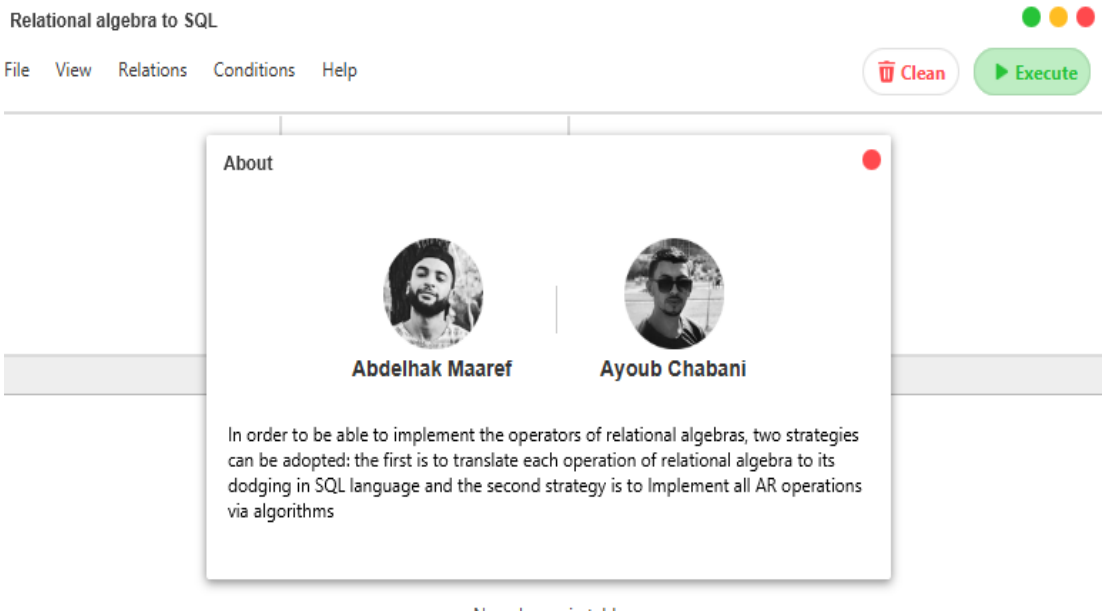

FIGURE 4.19 – about

4.4 Conclusion

Au cours de cette dernière étape de notre travail, nous avons présenté l'implémentation ainsi que les outils utilisés pour développer notre interpréteur.
A la fin, nous avons obtenu notre application qui satisfait les besoins souligniez au début et on a appliqué quelques exemples des interfaces pour apparait.

Conclusion générale

Le travail présenté dans ce mémoire entre dans le cadre du projet de fin d'études pour l'obtention du diplôme de master en informatique dans la spécialité : Ingénierie des Systèmes d'Information et du Logiciel.

Ce travail qui représente une concrétisation de cinq années d'études consistait à développer un interpréteur d'algèbre relationnelle. Développé comme un outil pédagogique, notre interpréteur répond à un besoin largement exprimé par les étudiants en informatique et même les enseignants qui interviennent dans le cours de bases de données. Ces futurs utilisateurs de notre système souhaitaient toujours avoir un outil pédagogique pour l'apprentissage et le test des connaissances dans cette brique importante dans le cours de bases de données qui est l'Algèbre relationnelle.

Ce projet inclut toutes les étapes de la conception et de réalisation de notre interpréteur par des algorithmes qui consiste à transformer des requêtes d'algèbre relationnelle en requêtes SQL et les transmettre à un SGBD pour faire l'exécution et obtenir le résultat sous forme d'un tableau. Notre interpréteur intègre un analyseur syntaxique et sémantique qui permet de vérifier le respect de la syntaxe de l'algèbre relationnelle et de vérifier aussi la correspondance des paramètres (operands) avec les noms des éléments sur schéma de la base de données utilisées.

La conception et de réalisation de notre application par le formalisme UML et la mise en œuvre des bases de données avec le gestionnaire MySQL. En fin l'exécution de l'application sous l'environnement de programmation IntelliJ IDEA nous a permis d'avoir une

bonne expérience et d'améliorer nos connaissances concernant le domaine de la programmation Orienté Objet et de nous faire une idée sur le domaine professionnel.

Nous estimons que nous avons pu atteindre les principaux objectifs tracés pour ce projet. Nous espérons que notre modeste travail sera de grand intérêt pour les étudiants de domaine informatique.

Comme perspectives futures, nous proposons d'enrichir l'outil avec des fonctionnalités avancées comme par exemple :

— La possibilité d'interpréter des opérations combinées.
— L'interprétation d'une suite d'opérations algébriques.
— L'amélioration de l'analyseur syntaxique pour souligner les parties erronées de la requête.

Bibliographie

a. Bibliographie :

- [**1**] Base de données et SGBD , Mali, Financement Kreditanstalt für Wiederaufbau (KfW), 2011-2012.

- [**2**] Olivier L. Introduction aux Systèmes de Gestion de Bases de Données Relationnelles, France, Université de Lille.

- [**3**] Bal K. Cours Bases de données L2 Informatique, Algeria, Université de Bouira, 2019.

- [**5**] CROZAT S. Le modèle relationnel, 02-10-2016.

- [**7**] Lecroq T. Bases de données, France, Université de Rouen.

- [**10**] Scholl M, Vodislav D. BASES DE DONNÉES Relationnelles, Paris, Vertigo/CNAM, 2003-2004.

- [**11**] Concepts et langages des Bases de Données Relationnelles, France, IUT de Nice.

- [**12**] Olivier C. Interpréteur d'algèbre relationnelle, Universite de Mons-Hainaut Faculté des Sciences,2003-2004.

- [**13**] Bal K. Cours de bases de données L'algèbre relationnelle, Algeria, Université de Bouira, 05-06-2017.

- [**14**] Dahak F. Chapitre 03 Le langage algébrique, Ecole nationale supérieur d'informatique (ESI).

- [**15**] Lumineau N. Bases de Données et programmation WEB Partie "Algèbre relationnelle", France, Université Claude Bernard Lyon 1, 2016-2017.

- [**22**] VODISLAV D. Bases de données avancées Évaluation et optimisation des requêtes, France, Université de Cergy-Pontoise.

- [**32**] Boussaid I. Génie Logiciel, 2013-2014.

b. Webographie :

- [**4**] https ://stph.scenari-community.org/bdd/0/co/sql0c00.html : Le langage SQL : octobre 2019

- [**6**] https ://stph.scenari-community.org/bdd/0/co/relUC012.html : Définition du modèle relationnel : octobre 2019

- [**8**] https ://stph.scenari-community.org/idl-bd/idl-bd3.pdf : Le modèle relationnel : octobre 2019

-[**9**]https ://www.editions-eni.fr/open/mediabook.aspx ?idR=00aadf0d7fe205588054f94bd885849a : L'algèbre relationnelle : octobre 2019

-[**16**]https ://www.maxicours.com/se/cours/langage-d-interrogation-des-donnees-lid/ : Langage d'interrogation des données (LID) : octobre 2019

- [**17**] https ://fr.wikipedia.org/ : Base de données : novembre 2019

- [**19**] https ://fr.wikipedia.org/wiki/Architecture_Ansi/Sparc : Architecture Ansi/Sparc : novembre 2019

- [**20**] https ://fr.wikipedia.org/ : Modèle relationnel : novembre 2019

- [**21**] https ://fr.wikipedia.org/ : 12 règles de Codd : octobre 2019

- [**23**] https ://slideplayer.fr/slide/2643849/ : Algèbre Relationnelle : Implémentation : novembre 2019

- [**24**] https ://www.pling.com/p/1131231/ : TeXstudio : décembre 2019

-[25]https ://desgeeksetdeslettres.com/web/xampp-plateforme-pour-heberger-son-propre-site-web : XAMPP : plateforme pour héberger son propre site web : décembre 2019

- [26] https ://www.scriptol.fr/programmation/javafx-script.php : JavaFX Script : décembre 2019

- [27] https ://www.techopedia.com/definition/7755/intellij-idea : IntelliJ IDEA : décembre 2019

-[28]https ://www.supinfo.com/articles/single/6088-kotlin-nouveau-langage-android : Kotlin, le nouveau langage d'Android : décembre 2019

-[29]https ://sql.sh/ : Apprendre le SQL : décembre 2019

-[30]http ://inf1410.teluq.ca/teluqDownload.php ?file=2014/01/INF1410-PresentationStarUML.pdf : Star UML : décembre 2019

- [31]https ://www.amazon.fr/UML-2 : UML 2 : décembre 2019

Résumé

L'algèbre relationnelle est une brique centrale et très importante dans le cours de bases de données. La bonne maitrise de ce langage est fondamentale pour l'assimilation du cours de bases de données. Nous proposons dans ce projet de développer sous forme d'un outil pédagogique, un interpréteur d'Algèbre relationnelle. Cet outil doit fournir aux enseignants et aux étudiants toutes les fonctionnalités nécessaires pour l'apprentissage de l'AR. L'outil prend en charge toutes les opérations nécessaires depuis la soumission de la requête à son exécution en passant par son analyse syntaxique et sémantique.

Mots clés : Base de données, algèbre relationnelle, translation, SQL, interpréteur, outils pédagogique...

Abstract

Relational algebra (RA) is a central and very important brick in the course of databases. The good comprehension of this language is fundamental for the assimilation of the course of databases. We propose in this project to develop in the form of an educational tool, an interpreter of relational algebra. This tool should provide teachers and students with all the features needed to learn the RA. The tool supports all necessary operations from the submission of the request to its execution through its syntactic and semantic analysis.

Keywords : database , translation , relational algebra, SQL, calculator, educational tools. . . .