UNIVERSITE DE BEJAIA
FACULTE DES SCIENCES EXACTES
DEPARTEMENT D'INFORMATIQUE
2ème année Licence

Année universitaire 2011/2012

MODULE DE BASES DE DONNÉES

Plan

Mme Z.TAHAKOURT
Maitre assistante
Université de Béjaia

Chapitre I : INTRODUCTION AUX BASES DE DONNEES

Plan du Chapitre

A/ Bases de données

1- Définition

Une base de données (BD) est un ensemble structuré de données enregistrées avec le minimum de redondance dont la gestion est assurée par un logiciel appelé SGBD(système de gestion de bases de données), par « données » on sous-entend tout fait significatif pouvant être enregistré, Exemples de BD :

1. BD des compagnies aériennes pour la gestion de la réservation de billets d'avions,
2. BD d'un cabinet médical pour la gestion des patients.
3. BD d'un service de scolarité d'une école pour la gestion de la scolarité de ses éléves.

2- Cycle de vie d'une BD

Lorsqu'une entreprise décide d'informatiser ses services, le premier problème à résoudre est de déterminer les informations qu'il conviendra de mettre dans la BD. Cette phase d'analyse et de réflexion qui aboutit à déterminer le futur contenu de la BD s'appelle *phase de conception* de la BD. Une fois que la phase de conception terminée vient la *phase d'implémentation*, qui consiste à mettre la BD dans l'ordinateur via un SGBD. Une fois que le SGBD aura pris connaissance de la description de la BD, il sera possible aux utilisateurs d'entrer les données et exprimer les requêtes de mise à jour (ajout de nouvelles informations, supprimer les information périmées, modification des informations) cette phase est appelée *phase d'utilisation*. On appelle cycle de vie d'une BD la suite des phases conception , implémentation et utilisation.

3- Modèle de données et Schéma de données

on appelle *modèle de données* l'ensemble des concepts qui permettent la description des données d'une BD et les règles d'utilisation de ces règles.
On appelle *schéma de données* l'expression de la description de la BD en employant un modèle de données.

4- Trois niveaux de description

Au cours des différentes phases de la vie d'une BD, plusieurs descriptions ou schémas sont sont successivement élaborés, chacun répondant à un objectif bien déterminé et complémentaire.

4-1 Schéma conceptuel

lors de la phase conception, il est nécessaire que les utilisateurs puissent discuter de leurs besoins et exprimer leurs visions sous forme d'une description de la future BD. Cette description s'appuiera sur un langage formel basé sur un certain nombre de concepts bien établis, le modèle utilisé est dit conceptuel et la description ainsi obtenue est dite *schéma conceptuel.*

4-2 Schéma interne

lors de la phase implémentation, on décrit la structure physique de stockage de la BD. On spécifie comment les données sont enregistrées sur les mémoires secondaires (disques, bandes, tambours, ...). Lors de cette phase on s'occupe de tous les détails de stockage des données. L'ensemble de ces détails est appelé *schéma interne*.

5- Approche fichier versus Approche base de données

Dans l'approche fichier, les fichiers sont définis pour un ou plusieurs programmes de traitement. Les données de ces fichiers sont directement associées à un programme par une description contenue dans le programme lui même.

INSERTION DIMAGE.

Dans l'approche base de données la partie de structuration et de description des données est unifiée et séparée des programmes d'application. Les applications ne communiquent avec les données qu'au travers de l'interface du SGBD. D'où l'indépendance entre les données et les applications, qui peuvent être modifiées indépendamment.

Exemple

On désire conserver les notice (informations) d'articles de journaux dans un fichier. Chaque notice occupera un enregistrement du fichier. Une notice contient les infos suivante :
le nom et l'adresse de l'éditeur du journal, le nom du journal et le nom de son rédacteur en chef, le numéro du journal dans lequel l'article a paru, le nom et l'adresse de l'auteur de l'article et le titre de l'article.
Exemple d'enregistrements :

INSERTION DIMAGE.

Solution1 : avec un fichier

Définition du fichier:

1. fichier séquentiel indexé contenant les enregistrements d'articles de journaux
2. création d'un index pour chaque champ d'enregistrement (excepté pour le titre de l'article)

Problèmes rencontrés avec cette solution :

1. Taille du fichier énorme en raison des données redondantes. Par ex: les quatre premiers champs des articles parus dans un même journal sont identiques.
2. Opérations de consultation (lecture dans le fichier) .
3. Accès par une seule clé faciles à réaliser: Ex: recherche de l'éditeur d'un journal donné, recherche de tous les articles écrits par un auteur donné etc.
4. Accès par plusieurs clés simultanément (plus difficiles à réaliser): quelle clé d'accès choisir?) Ex: recherche de tous les articles parus dans le journal de Genève n°20.
5. Opérations de modification (écriture dans le fichier).

6. Problèmes plus graves, connus sous le nom d'anomalies de mise à jour.

- *Lors de l'insertion d'un article :*

Akx de	Genève	Campus	T.Boyson	40	K.Bosko	...

----> Deux éditeurs différents pour le journal *Campus*, lequel est correct?

- *Lors de suppression d'un article :*

Uni GE	Genève	Campus	T.Boyson	40	K.Bosko	...

----> on perd l'information que « Uni GE » est l'éditeur de *Campus, et que son rédacteur en chef est T.Boyson.*

- *Lors de la modification d'un article :*

si on modifie le nom du rédacteur en chef d'un journal dans un enregistrement, il faut répercuter cette modification dans tous les enregistrements concernant le même journal.

Solution2 : avec une base de données

1.Modélisation conceptuelle de la réalité perçue à l'aide d'un modèle de données (concepts)
2.Règles de validation du schéma conceptuel (méthode de normalisation)
3.Réalisation informatique: avec un Système de Gestion de Base de Données (SGBD)
4.Programmation des accès aux données avec un langage spécifique

B/ Les SGBD

1- Définition

Le SGBD est un ensemble de programmes, non modifiables par l'utilisateur, permettant de créer et de manipuler une BD.

2- Architecture fonctionnelle d'un SGBD

INSERTION DIMAGE.
Au niveau d'abstraction (sans entrer dans les détails)le plus élevé, un SGBD peut être vu comme une boite noire, assurant la gestion de la BD conformément aux requêtes des utilisateurs.

2-1 Interface d'accès utilisateur

permet aux utilisateurs d'exprimer des requêtes, soit pour définir le contenu de la BD (avec le LDD : langage de définition de données) , soit pour interroger la BD, soit enfin pour apporter des modification à ce qui a été enregistré (avec le LMD : langage de manipulation de données).

2-2 Interface d'accès physique

Elle permet au SGBD d'accéder aux données sur les supports physiques (disque, tambour …).

Grâce à ces deux interface, un SGBD assure un objectif fondamental qui est celui de l'indépendance programme/données, à savoir, d'une part la possibilité pour un utilisateur de modifier sa vue de la base et ses traitement sans avoir à se soucier des choix qui ont été opéré au niveau interne en matière de fichiers, d'autre part inversement, la possibilité pour l'administrateur système de modifier ses choix, pour améliorer les performances, par exemple, sans que cela ait un impact sur les utilisateurs c-à-d leurs requêtes d'interrogation ou de mise à jour ou leurs programmes d'application qui utilisent la BD).

3- Apports (avantages) d'un SGBD

Du point de vue fonctionnel, les apports d'un SGBD sont les suivants :

3.1 Partage de données

A un instant donné, on pourrait avoir plusieurs utilisateurs qui voudraient accéder à la même donnée. Le SGBD est en mesure de contrôler cette utilisation simultanée d'une même donnée, pour cela il dispose d'un module appelé « contrôleur de concurrence », assurant l'exclusion mutuelle.

3.2 Restreindre les accès non autorisés

Le SGBD possède un module capable de contrôler les accès à la BD, un utilisateur ne peut accéder à une donnée que s'il en a le droit, qui est préalablement attribué par l'administrateur de la BD.

3.3 Assurer le respect des règles de cohérence définie sur les données.

A priori, après chaque modification sur la BD toutes les règles de cohérence doivent être vérifiées sur toutes les données. Exple : le poste d'un employé est en rapport avec son salaire, donc si une mise à jour est portée sur son poste, ceci doit entrainer automatiquement une mise à jour sur son salaire. Ces traitements doivent pouvoir être effectués sans arrêter le système

3.4 Récupération de données en cas de panne

Si une opération de mise à jour est en cours d'exécution, et une panne de courant, par exemple se produit, alors il se pourrait que la BD rentre dans un état incohérent. Le SGBD dispose d'un module prévu spécialement pour ce genre de problème, qui permet de ramener la BD à son dernier état cohérent.

Exemple

Soit deux compte bancaires, dont les soldes sont 2500DA et 1500DA

N°	Nom	Montant
1	Ali	25000
2	Bilal	1500

Requête de màj : transférer 200DA du compte de Ali vers le compte de Bilal.

```
Début d'exécution
        compte1-200
                        <--------  panne de courant : État incohérent de la BD
        compte2+200
```

3.5 Assurer la confidentialité et la sécurité des données

Il est nécessaire de pouvoir spécifier qui a le droit d'accéder ou de modifier tout ou une partie d'une BD. Le SGBD permet de se prémunir contre les manipulations illicites qu'elles soient intentionnelles ou accidentelles. Cela nécessite d'une part une identification des utilisateurs, et d'autre part une spécification des droits (ajout, suppression et modification).

5- Trois types d'utilisateurs

Un certain nombre d'utilisateurs peuvent accéder aux données ou en obtenir à la demande grâce aux applications et aux interfaces fournies par le SGBD. Chaque type d'utilisateur nécessite des possibilités logicielles différentes :

5-1- l'administrateur de la BD :

C'est la personne en charge de l'implémentation de la BD dans l'organisation. Il possède tous les privilèges système autorisés par le SGBD et peut attribuer des niveaux d'accès (privilèges) aux autres utilisateurs.

5-2- Les utilisateurs finaux

Ce sont les personnes assises à leur station de travail et qui interagissent directement avec la BD. Ils peuvent avoir à répondre aux requêtes de personnes extérieures à l'organisation, de répondre rapidement aux questions de leurs supérieurs hiérarchiques ou de générer des états périodiques. Certains utilisateurs finaux ont l'autorisation de modifier les données, alors que d'autres n'ont que la possibilité d'afficher les données mais pas de les modifier.

5-3- Les programmeurs d'application

Ils interagissent avec la BD d'une manière différentes. Ils accèdent aux données depuis leurs programmes écrits dans des langages de haut niveau tels que le C++, Visual Basic, Java, PHP,…etc. les programmeurs conçoivent des programmes tels que la gestion des salaires, qui nécessitent d'accéder aux données et de les modifier.

6- Architecture opérationnelle des SGBD

6-1 L'architecture client/serveur

Le principe est le suivant : le client demande une ressource et le serveur la lui fournit directement. Deux cas peuvent se présenter :

1) Le serveur fournit la réponse en utilisant ses propres ressources (architecture 2-tier ou à 2 niveaux), c'est-à-dire qu'il ne fait appel à aucun autre serveur pour fournir une partie de la réponse.
2) Le serveur fait appel à un autre serveur, pour obtenir une partie de la réponse (architecture 3-tier ou à 3 niveaux).

INSERTION DIMAGE.

L'architecture client/serveur des SGBD est entrée sur la scène informatique au début des années 90. Ils ont eu un impact considérable sur la technologie des SGBD. L'idée générale est simple et élégante : distinguer les fonctionnalités qui sont sensée être fournies, et les diviser en deux classes: celles du serveur et celles du client. Ceci fournit une architecture, à deux niveaux qui rend la gestion des SGBD plus facile.

Le serveur fait la majeure partie du travail de gestion des données (traitement et optimisation de requête, que la gestion des transactions ainsi que la gestion du stockage des données.

Le client en plus de son programme d'application, et son interface utilisateur il dispose d'un module SGBD client qui est responsable de la gestion des données qui sont cachées.

INSERTION DIMAGE.

6-2 L'architecture distribuée (ou répartie)

Les systèmes de BD sont passés d'une étape où chaque application définit et maintient ses propres données à l'étape où les données sont définies et administrée de façon centralisée, puis à une étape où les données sont définies et administrée de façon distribuée.

INSERTION DIMAGE.

Une base de données distribuée (BDD) est une collection de plusieurs BDs mises en relation et distribuées à travers un réseau d'ord. Un SGBD distribué est alors défini comme étant le logiciel qui permet la gestion de la BDD, et qui rend la distribution transparente à l'utilisateur.

Une architecture distribuée d'un SGBD peut définie comme étant composée e plusieurs serveur coopérant à la gestion de bases de données composées de plusieurs sous bases gérées par un seul serveur mais apparaissant comme une BD unique à l'utilisateur.

Chapitre II : MODELE RELATIONNEL-NORMALISATION DE RELATION

Plan du Chapitre

Mme Z.TAHAKOURT

1- Introduction

Le modèle relationnel a été inventé par Mr CODD en 1970. Il est basé sur des concepts très simples. C'est le modèle de données le plus utilisé par les SGBDs actuels. C'est un modèle de données plus simple que celui de l'E/A.

2- Définitions

Les objets et associations du monde réel sont représentés par un concept unique qui est la « relation ». les relations sont des tableaux à deux dimensions appelés « tables ».

Une relation est définie par :

1. Son nom,
2. liste des couples (attribut, domaine),
3. Son (ses) identifiant(s).

Ces trois informations constituent le schéma ou « intention » de la relation.

La population ou « extension » d'une relation est constituée de l'ensemble des tuples de la relation.

On appelle schéma d'une BDR (BD relationnelle) l'ensemble des schémas de ses relations.

On appelle BDR l'extension de toutes ses relations.

On appelle cardinalité (arité) d'une relation le nombre de tuples que son extension contient.

On appelle degré d'une relation le nombre d'attributs que son intension contient.

3- Concept de normalisation

Afin de dégager quelques propriétés d'un schéma relationnel mal conçu, considérons le schéma de l'exemple suivant sur la livraison des fournitures. Dans cet exemple on voudrait garder les informations concernant les produits et leurs fournisseurs :

FOURNITURE(NF, NP, NomF, AdrF,designationP , QTE)

dont l'extension est la suivante :

NF	NP	NomF	Date	AdrF	designationP	QTE
01	A	Ali	01/05/1985	Béjaia	Pain	205
…						

Cette exemple soulève plusieurs types de problèmes :

1. Redondance : l'adresse et le nom d'un fournisseur est répétée pour chaque pièce qu'il fournit

2. Anomalie de mise à jour : lors d'une mise à jour, on risque de modifier l'adresse d'un fournisseur dans un tuple mais pas dans les autres. Un même fournisseur possèdera plusieurs adresses !

3. Anomalies d'insertion : on n'enregistre pas l'adresse d'un fournisseur, si on n'achète pas au moins une pièce à ce fournisseur.

4. Anomalies de suppression : si on supprime toutes les pièces proposées par un fournisseur ce dernier n'apparaît plus.

Le schéma précédent peut se décomposer selon deux schémas :

Fournisseur (NF,NomF, AdrF) , Produit (NP, DesignationP) et Livraison (NF,NP,date, Qte)

Le processus de transformation d'une relation posant des problèmes lors des m-à-j en relations n'ayant pas ces problèmes, est appelé processus de normalisation ou décomposition.

Cette décomposition élimine les problèmes précédents. En revanche, pour trouver l'adresse d'un fournisseur qui fournit une pièce donnée, il faut recourir à une jointure entre Fournisseur, et Livraison. L'opération peut se révéler coûteuse en temps, mais si on doit choisir entre la cohérence de la base de données et le gain de temps c'est la cohérence qui prime.

On mesure la qualité d'une relation (ou sa capacité à représenter le monde réel) par son degré de normalisation. Une relation peut être de la moins bonne à la meilleure : 1FN, 2FN, 3FN, BCNF, 4FN, 5FN,…).

Les procédures qui permettent de décomposer une relation en " bonnes relations", reposent sur le concept de *dépendance fonctionnelle*(DF).

4- Concept de dépendance fonctionnelle(DF)

Une *Dépendance fonctionnelle* (DF) sur un schéma relationnel R, est une proposition de la forme Si , pour tous tuples t1 et t2 de R qui ont la même valeur pour l'attribut A, on a t1 et t2 qui ont les même valeurs pour l'attribut B, alors on dit que A détermine fonctionnellement B.. On note : A -> B. A est appelé : *Prémisse*, et B : *conclusion*.

Remarque
Si l'attribut A détermine plusieurs attributs B ,C et D…
On écrit : A -> BCD

Exemple :
R(A,B,C)
On a :
A -> B ; A -> C ; c'est tout. Toutes les autres DF ne sont pas satisfaites.

A	B	C
1	5	4
2	5	6
3	6	6

4-1 Dépendance fonctionnelle élémentaire

Une dépendance fonctionnelle X->A est dite élémentaire, si :

1. A est un attribut unique n'appartenant pas à X ;
2. $\nexists X' \in X /$ tel que X' -> A.

Exemple :

Soit une relation R(A,B,C). A->B : DF élémentaire. A->BC : DF non élémentaire B->C :DF élémentaire.

4-2 Dépendance fonctionnelle triviale

Une dépendance fonctionnelle X->A est dite triviale si et Seulement si : $A \in X$

Exemple : Soit une relation R(A,B,C). Ab->B: DF triviale. A->BC : DF non triviale

4-3 Dépendance fonctionnelle déduite

Une dépendance fonctionnelle X->A est dite déduite si partant de X on peut arriver à A sans utiliser la DF X->A. en d'autres mots, X -> A est redondante dans F si elle est conséquence de F - {X -> A}.

Exemple :

R(A,B,C,D) ;
F={A->B, ----→ non déduite
B->C, ----→ non déduite
A->C} ----→ déduite

5- Propriété des dépendances fonctionnelles : Axiomes d'Armstrong

A partir d'un ensemble F de dépendances fonctionnelles entre les attributs d'un schéma de relation R, on peut en déduire d'autres à partir des trois propriétés suivantes :

1. réflexivité : si X contient Y, alors X -> Y.
   Cette règle stipule que tout ensemble d'attribut détermine lui-même ou une partie de lui même.
2. Augmentation : si X -> Y, alors XZ -> Y pour tout groupe Z d'attributs appartenant au schéma de relation
   Cette règle signifie que si X-> Y, alors les deux ensemble peuvent être enrichis par un troisième.
3. transitivité : si X -> Y, et Y -> Z, alors X -> Z,

ces trois axiomes de base sont connus sousle nom d'*Axiomes d'Armstrong, plusieurs* autres règles peuvent se déduire de ces axiomes:

1. union : si X -> Y et Y -> Z, alors X -> YZ,
2. pseudo-transitivité : si X -> Y et WY -> Z, alors WX -> Z,
3. décomposition : si X -> Y et Z contenu dans Y, alors X -> Z.

6- La fermeture d'un ensemble d'attributs

Soit X un ensemble d'attribut et F un ensemble de dépendances fonctionnelles. On appelle fermeture de X sous F, notée $\mathbf{X^+}$, l'ensemble des attributs de R qui peuvent être déduits de X à partir de F en appliquant les axiomes d'Armstrong. Ainsi, Y sera inclus dans X^+ ssi X -> Y.

Algorithme de Calcul de la fermeture d'un ensemble d'attributs :

1. initialiser (X)+ à X,
2. trouver une dépendance fonctionnelle de F possédant en partie gauche des attributs inclus dans (X)+,
3. ajouter dans (X)+ les attributs placés en partie droite de la dépendance fonctionnelle,
4. répéter les étapes 2) et 3) jusqu'à ce que X+ n'évolue plus.

Exemple

Soit F = { A -> D ; AB -> E ; BI -> E ; CD -> I ; E -> C }.
Question : calculer la fermeture, sous F, de AE.
Solution : au départ, $(AE)^+ = \{A,E\}$,
A -> D permet d'ajouter D : $(AE)^+ = \{A,E,D\}$
E -> C permet d'ajouter C : (AE)+ = AEDC,
CD -> I permet d'ajouter I : (AE)+ = AEDCI.
Question : calculer la fermeture, sous F, de BE.
Solution : au départ, (BE)+ = BE,
E -> C permet d'ajouter C : (BE)+ = BEC.
Exercice :

Soit F = {AB -> C ; B -> D ; CD -> E ; CE -> GH ; G -> A }.
Question : en utilisant la notion de fermeture d'un ensemble d'attributs, montrer que AB -> E,
Solution : B -> D |= AB -> D par augmentation,
AB -> C et AB -> D |= AB -> CD par union,
AB -> CD et CD -> E |= AB -> E par transitivité.
Question : en utilisant la notion de fermeture d'un ensemble d'attributs, montrer que BG -> C,
Solution : G -> A |= BG -> A par augmentation,
BG -> BG |= BG -> B par projection,
BG -> A et BG -> B |= BG -> AB par union,
BG -> AB et AB -> C |= BG -> C par transitivité.
Question : en utilisant la notion de fermeture d'un ensemble d'attributs, montrer que AB -> G.
Solution : AB -> E et AB -> C |= AB -> CE par additivité,
AB -> CE et CE -> GH |= AB -> GH par transitivité,
AB -> GH |= AB -> G par projection.

7- La couverture minimale d'un ensemble de DFs

Un ensemble de DF F est dit couverture minimale, si aucune DF f de F ne peut être déduite à partir de F en appliquant les propriétés des DFs.

Voici quelques précisions sur la manière d'obtenir une couverture minimal à partir d'un ensemble F de DF. Pour simplifier un ensemble de DF, on doit enlever les DF non-élémentaires ainsi que les DF redondantes.

Exemple1 :

Soit R(A,B,C). F={AB->C ,B->C }. Montrer que F n'est pas une couverture minimale. Trouver sa couverture minimale

La DF AB->C est déduite car : B->C =>(augmentation par A) AB->AC => (décomposition) AB->A et AB->C (cqfd).

Donc F n'est pas une couverture minimale. sa couverture minimale F'={B->C}.

Exemple2:

Soit R(A,B,C,D,E). F={AB->C ,B->E, C->D, E->D, C->E, BC->D }. Montrer que F n'est pas une couverture minimale. Trouver sa couverture minimale

F'={AB->C ,B->E, E->D, C->E}.

Exemple3

Considérons l'ensemble suivant :
F={A -> B,B C -> D,A C -> B D E,D -> E}Cet ensemble s'écrit : {A -> B,B C -> D,A C -> B ,A C -> D,A C ->E D -> E}
Calculer la couverture minimale de F.
La DF (A C -> B) n'est pas élémentaire car (A -> B) donc on peut l'enlever. La DF (A C -> D) est conséquence de (A -> B) (car alors (A C -> B C)) et de (B C -> D), donc elle est redondante. De même (A C ->E) est redondante car elle se déduit de (A -> B), (B C -> D) et (D -> E). On obtient l'ensemble minimal suivant : F'={A -> B ,B C -> D,D -> E}

8- Graphe de DFs

Un Graphe des dépendances est un ensemble de nœuds et d'arcs tels que :

1. Chaque nœud du graphe est un attribut atomique, ou non.
2. Chaque arc du graphe est une dépendance fonctionnelle.
3. Les arcs sont orientés.

Dans le cas où toutes les DFs sont non déduite, on obtient un *graphe minimum de DFs.*

Exemple :

Soit R(A,B,C,D,E) une relation. F={AB->C, B->E, C->B, C->E, D->C}

Donner le graphe minimum de DF.

F'={ AB->C, B->E, C->B, D->C }la DF C->E est déduite.

9- Clé minimale, clé candidate, clé primaire, clé alternative et super clé

Soit R (A1, ..., An) un schéma relationnel. X un sous ensemble de {A1, ..., An}. On dit que X est une « clé minimale » de la relation R, si X respecte les deux contraintes suivantes :

1. Unicité. Deux n-uplets distincts de R ne peuvent avoir même valeur de K.
2. Irréductibilité (ou minimalité). Il n'existe pas de sous-ensemble strict de K garantissant la règle d'unicité.

En d'autres mots :

1. X -> A1 ... An (X^+=Attr(R)), et
2. $\nexists X' \in X$/ tel que X' -> A1 ... An (X'^+=Attr(R)).

Si la relation comporte plusieurs clés minimales, celles-ci sont appelées clés *candidates*, le modèle relationnel exige que seul une de ces clés candidates soit choisie comme clé *primaire* pour cette relation ; les autres clés, s'il y en a, sont appelées clés *alternatives*.

On appelle « super clé» d'une relation, un sous-ensemble d'attributs K' de la relation, telle que : K∈K', où K est une clé minimale de la relation. C-à-d que K' est une clé ne garantissant pas la contrainte de minimalité.

Exemple1

Chambre(Nom_Cu, N°Ch, N°Bloc, NbPlaces)

Exemple2

On considère la relation Fournisseur(Nom, Article, Prix, Adresse)
F={Nom,Article->Prix ; Nom->Adresse}
Montrer que (nom,article) est une clé minimale de fournisseur.
Solution

1- $(nom,article)^+$={nom, article, prix, adresse}=Attr(Fournisseur)
2- Nom^+={nom, adresse}≠ Attr(Fournisseur)
3- $Article^+$={article}}≠ Attr(Fournisseur)

Donc, (nom,article) est une clé minimale de fournisseur.

10- Propriété des clés minimales

Soit R une relation et F un ensemble de DF. Alors :

1. Propriét1 : tout attribut de R qui ne figure pas dans les conclusions des DFs doit appartenir à toutes les clés minimales.
2. Propriét2 : si l'ensemble des attributs de R qui ne figurent pas dans les conclusions des DFs forment une clé minimale, alors celle-ci est unique.

Exemple

On considère la relation R(A,B,C,D) ; F={A->BC ; C->B}
Trouver toutes les clés minimales de R.

Solution

Les attributs qui ne figure pas dans les conclusions des DFs, sont : AD, on va calculer $(AD)^+$.
$(AD)^+$ = {A, D, B, C} =Attr(R). D'après la Propriét1 {AD} appartiendrait à toutes les clés min de R
A^+= {A, B, C} ≠ Attr(R). D^+={D}≠ Attr(R). D'après la Propriét2 AD est l'unique clé minimale de R.

11- Décomposition des relations

Revenons à notre objectif initial, qui est la décomposition d'une relation afin d'éviter les redondances et les anomalies qui en découlent. On peut décomposer R en R1,...,Rn en s'assurant de ne pas perdre de l'information ni de DF. On définit pour cela les deux propriétés suivantes :

11-1 Décomposition sans perte d'information

Une décomposition est dite sans perte d'information si la jointure naturelle des sous-relations calcule exactement tous les tuples de la relation initiale. Autrement dit, si on a : R = R1 ⋈R2 ⋈... ⋈Rn . Considérons la relation R suivante :

R	A	B	C	D
	a	5	x	2
	b	5	y	1
	c	5	x	2

R1	B	C	D
	5	x	2
	5	y	1

R2	A	B
	a	5
	b	5
	c	5

Cette relation a pour schéma R(A,B,C,D) que l'on décompose en R1(A,B) et R2(B,C,D). Cette décomposition n'est pas sans perte d'information puisque R ne vaut pas R1 ⋈R2.
Par contre, si l'on décompose R en R1(A,B,C) et R2(A,D), on obtient une décomposition sans perte d'information car R1 et R2 vallent : et dans ce cas, R = R1 ⋈R2.

R1	A	B	C
	a	5	x
	b	5	y
	c	5	x

R2	A	D
	A	2
	b	1
	c	2

11-2 Décomposition sans perte de dépendances fonctionnelles

On aimerait aussi que la décomposition préserve les dépendances fonctionnelles. Si R est une relation qui satisfait un ensemble de dépendance F, et si R se décompose en R1,...,Rn, on note F_i l'ensemble des DF de R qui portent sur Attr(R_i). La décomposition préserve les dépendances fonctionnelles si (F_1 ∪... ∪F_n) est équivalent à F.

Exemple :

Reprenons l'exemple. La relation R(A,B,C,D) avec l'ensemble de DF F={A -> BC,C -> D,D -> B} La décomposition de R en R1(A,B,C) et R2(A,D) ne préserve pas les DF. En effet F_1, ensemble des DF qui portent sur les attributs de R1 est {A -> BC} et l'ensemble F_2 des DF qui portent sur les attributs de R2 est vide.

Chapitre III : L'ALGEBRE RELATIONNELLE

Introduction

Les opérateurs

- **La projection**
- **La sélection**
- **Union, différence, intersection de deux relations ayant le même schéma**
- **Le Renommer**
- **Le produit cartésien**
- **La thêta jointure**
- **La jointure naturelle de deux relations ayant au moins un attribut en commun**
- **La division**

Exercices

1- Introduction

L'algèbre relationnelle (AR) a été inventé par T.Codd comme étant un ensemble d'opérateurs, qui à partir d'une ou plusieurs relations existante, créent en résultat une nouvelle relation temporaire (c.-à-d. a une durée de vie limité généralement, la relation est détruite à la fin du programme utilisateur). La relation résultat a exactement les mêmes caractéristiques qu'une relation de la BD, et peut donc être manipulée de nouveau par les opérateurs de l'AR.
L'AR comprend :

- Cinq (05) opérateurs de base : Projection, sélection, union, différence et produit cartésien.
- Un opérateur syntaxique qui ne fait que modifier le schéma de la relation et pas les tuples.

A partir de ces opérateurs d'autres opérateurs ont été déduits comme, l'intersection, la jointure et la division.

2- Les opérateurs

2-1 La projection π

Cet opérateur construit une relation résultat où n'apparaissent que les attributs X de la relation, elle est notée : $\pi_X(R)$. Les doublons éventuels seront supprimés.

Exemple

Soit R (A, B, C) une relation.

R'= $\pi_{A,B}(R)$.

R	A	B	C
	1	2	3
	1	2	4
	5	6	7

R'	A	B
	1	2
	5	6

2-2 La sélection σ

Cet opérateur construit une relation résultat où n'apparaissent que certains tuples de la relation initiale. Les tuples retenus sont ceux qui satisfont une condition explicite appelée *prédicat de sélection*.

Les différentes conditions que l'on va rencontrer sont : $<$, $>$, $\leq$, $\geq$, $=$, $\neq$, ..., reliées les unes aux autres par les opérateurs logiques : et, ou, ...

R	A	B	C
	1	2	3
	1	2	4
	5	6	7

Exemple1

Soit R (A, B, C) une relation.

R'= $\delta_{A=C}(R)$.

R'	A	B	C
	8	1	8

8	1	8

Exemple2

Soit personne (NP, Nom, Age) une relation.

Donner la liste des personnes âgées de plus de 40 ans.

Resultat= $\delta_{Age>40}(Personne)$.

2.3 L'Union, Différence, Intersection de deux relations ayant le même schéma

Soit R et S deux relations ayant le même schéma (A1, A2,…An).

- Union (**R∪S**) : crée une relation temporaire de même schéma et de population égale à l'ensemble des tuples de R et ceux de S.
- Différence(**R−S**) : crée une relation temporaire de même schéma et de population égale à l'ensemble des tuples qui se trouvent dans R mais pas dans S.
- Intersection(**R∩S**) : crée une relation temporaire de même schéma et de population égale à l'ensemble des tuples qui sont communs àR et à S.

Exemple1

Soit R (A, B, C) et S(A, B, C) deux relations.

R	A	B	C
	1	2	g
	1	2	a
	5	6	b
	8	1	c

S	A	B	C
	11	22	b
	1	2	a
	5	6	b
	8	1	d

R ∩S	A	B	C
	1	2	a
	5	6	b

R∪S	A	B	C
	1	2	g
	1	2	a
	5	6	b
	8	1	c
	11	22	b
	8	1	d

R-S	A	B	C
	1	2	g
	8	1	c

Exemple2

Soit la BD relationnelle, dont le schéma est :

Fournisseur (NF, NomF, Adr)

Produit (NP, NomP, Couleur)

ALivré (NF ,NP, Date, Qtité)

Question : Ecrire les requêtes suivantes en AR.

1) Les noms des fournisseurs qui habitent « *Aokas* »
2) Les produits rouges.
3) Les noms des produits rouges.

4) Les noms des produits rouges et des produits verts.
5) Les noms des produits NON rouges.

Réponse :

1) Les noms des fournisseurs qui habitent « *Aokas* » :

$$\pi_{NomF}[\ \sigma_{AdrF='Aokas'}\ (Fournisseur)]$$

2) Les produits rouges

$$\sigma_{Couleur='Rouge'}(Produit)$$

3) Les noms des produits rouges

$$\pi_{NomP}[\ \sigma_{Couleur='Rouge'}(Produit)]$$

4) Les noms des produits rouges et des produits verts.

$$\pi_{NomP}[\ \sigma_{Couleur='Rouge'}(Produit)] \cup \pi_{NomP}[\ \sigma_{Couleur='Vert'}(Produit)]$$

5) Les noms des produits NON rouges.

$$\pi_{NomP}\ (Produit) - \pi_{NomP}[\ \sigma_{Couleur='Rouge'}(Produit)]$$

2-3 Le Renommer α

Cet opérateur permet de changer le nom de un ou plusieurs attribut(s) d'une relation.

$$\alpha_{\ [NomAttr1\ :NouvNomAttr1\ ;\ NomAttr2\ :NouvNomAttr2\ ;\ldots]}\ (R)$$

Exemple

Soit R (A, B, C) une relation.

$R' = \alpha_{[A\ :D\ ;\ C\ :E]}(R)$

R	A	B	C
	1	2	g
	1	2	a
	5	6	b
	8	1	c

R'	D	B	E
	1	2	g
	1	2	a
	5	6	b
	8	1	c

2-4 Le Produit cartésien X

Le produit cartésien de deux relations R et S de schéma quelconque, noté **RXS**, est une relation ayant pour schéma l'union des deux schéma de R et de S, et dont les tuples sont toutes les concaténations possibles d'un tuple de R à un tuple de S.

Exemple
Soit R (A, B, C) et S (C, D, E) deux relations.

R	A	B	C
	1	2	g
	5	2	a

S	C	D	E
	3	8	a
	5	5	a
	7	1	b

R'= R X S

R'	A	B	R.C	S.C	D	E
	1	2	g	3	8	a
	1	2	g	5	5	a
	1	2	g	7	1	b
	5	2	a	3	8	a
	5	2	a	5	5	a
	5	2	a	7	1	b

2-5 La thêta jointure

La thêta jointure entre deux relations R et S selon un prédicat θ, notée **R** $\bowtie_\theta$ **S** est une relation ayant pour schéma l'union des deux schéma de R et de S, et dont les tuples sont ceux de RXS qui satisfont θ.

Exemple
Soit R (A, B, C) et S (C, D, E) deux relations.
R'= R X S

R	A	B	C
	1	2	g
	5	2	a

S	C	D	E
	c	8	a
	a	5	g
	d	1	b

RXS

A	B	R.C	S.C	D	E
1	2	g	c	8	a
1	2	g	a	5	g
1	2	g	d	1	b
5	2	a	c	8	a
5	2	a	a	5	g
5	2	a	d	1	b

$R' = R \bowtie_\theta S \; ; \theta = (A <= B) et (R.C = E)$

R'	A	B	R.C	R.C	D	E
	1	2	g	a	5	g

2-6 La jointure naturelle $\bowtie$ de deux relations ayant au moins un attribut en commun

Etant donnée deux relations R(X, Y) et S(Y,Z). X, Y et Z sont soit des attributs soit des ensembles d'attributs. La jointure entre R et S, notée R $\bowtie$ S, est la relation T(X,Y,Z) dont les tuples sont la concaténation « composition » d'un tuple de R et d'un tuple de S ayant la même valeur pour Y.

Exemple1

Soit R (A, B, C) et S (C, D, E) deux relations.

R	A	B	C
	1	2	g
	5	2	a

S	C	D	E
	c	8	a
	a	5	g
	d	1	b

R'= R ⋈ S

RXS	A	B	R.C	S.C	D	E
	1	2	g	c	8	a
	1	2	g	a	5	g
	1	2	g	d	1	b
	5	2	a	c	8	a
	5	2	a	a	5	g
	5	2	a	d	1	b

R'	A	B	C	D	E
	5	2	A	5	g

Remarque: si X est un ensemble d'attributs communs entre R et S, alors on a :

R ⋈ S= R ⋈ θ S tel que: θ=R.X=S.X

Exemple2

Soit la BD relationnelle suivante :

Client (NClient, NomC, PrenomC, Adr)

Produit (NProduit, LibelleProduit, Poids, Couleur)

Commande (NClient, NProduit, Date, Quantite)

Ecrire en AR les requetes suivantes :

1) Quels sont les noms des clients ayant commandé le produit N°102 ?
2) Quels sont les N° des clients ayant commandé le produit de libellé « Lessive Machine» ?
3) Quand est ce que le client DALI Amokrane a commandé de la lessive machine ?

Réponse :

1) Quels sont les noms des clients ayant commandé le produit N°102 ?

$\pi_{NomClient}[Client \bowtie_{NClient} (\sigma_{NProduit=102}(Commande))]$

2) Quels sont les N° des clients ayant commandé le produit de libellé « Lessive Machine» ?

$\pi_{NClient}[Commande \bowtie_{NProduit} (\sigma_{NomProduit='Lessive\ Machine'}(Produit))]$

3) Quand est ce que le client DALI Amokrane a commandé de la lessive machine ?

$\pi_{Date}[\sigma_{(NomClient='DALI')\ ET\ (PrenomClient='Amokrane')}(Client) \bowtie_{NClient} (Commande \bowtie_{NProduit} (\sigma_{NomProduit='Lessive\ Machine'}(Produit)))]$

2-7 La Division

Soit deux relations R(A_1,A_2,…A_p, A_{p+1}, …A_n) et S(A1,A2,…Ap). La division de R sur S, notée **R/S** est la relation T(A_{p+1}, …A_n) dont les tuples sont ceux concaténés à chacun des tuples de S, donne toujours un tuple de R.

Exemple 1
Soit R (A, B, C) et S (B, C) deux relations.

R

A	B	C
1	1	1
1	2	0
1	2	1
1	3	0
2	1	1
2	3	3
3	1	1
3	2	0
3	2	1

S

B	C
1	1
2	0

R/s	A
	1
	3

S

B	C
3	3
2	0

R/s	A

Exemple2
Reconsidérons l'exemple précedent,
Client (NClient, NomC, PrenomC, Adr)
Produit (NProduit, LibelleProduit, Poids, Couleur)
Commande (NClient, NProduit, Date, Quantite)

Ecrire en AR les requetes suivantes :

1) Quels sont les noms des produits ayant été commandé par tous les clients ?
2) Quels sont les noms des clients ayant commandé tous les produits ?

Réponse

1) Quels sont les noms des produits ayant été commandé par tous les clients ?

$$\pi_{\text{LibelleProduit}} [\pi_{\text{NProduit}} (\text{Commande} / \pi_{\text{NClient}}(\text{Client})) \bowtie_{\text{NProduit}} \text{Produit}]$$

2) Quels sont les noms des clients ayant commandé tous les produits ?

$$\pi_{\text{NomClient}} [\pi_{\text{NClient}} (\text{Commande} / \pi_{\text{NProduit}}(\text{Produit})) \bowtie_{\text{NClient}} \text{Client}]$$

EXERCICE
Question : A partir du modèle relationnel suivant, écrire en AR les requêtes ci-dessous.

CANDIDAT (NCandidat, NomCandidat, PrenomCandidat, date-naissance)

EPREUVE (NEpreuve, libelleEpreuve , dateRedaction, dateEpreuve, coefficient, CodeExamen#)
EXAMEN (CodeExamen, libelleExamen)
ENSEIGNANT (NEnseignant, nomEnseignant, prenomEnseignant)
PASSER (NCandidat#, NEpreuve#, note)
REDIGER (NEnseignant#, NEpreuve#)
INSCRIRE (CodeExamen#, NCandidat#, appréciation)

1) Liste des noms et prénoms de tous les candidats.

2) Quels sont les prénoms des candidats de noms MARTIN ?

3) Quelles sont les notes du candidat N° 102 à l'examen CAP_TF18 ?

4) Quels sont les candidats (numéro, nom, prénom) dont la note est 10 en français ?

5) Quels sont les examens (libellé examen, libellé épreuve) dont au moins une épreuve a été rédigé par le professeur Jeanne HYMME ?

6) Quels sont les candidats (numéro, nom, prénom) ayant passé le CAP Tourneur sur bois, ou le BEP Tourneur sur bois ? (utiliser l'UNION)

7) Quels sont les candidats (numéro, nom, prénom) ayant passé le CAP Tourneur sur bois, et le BEP Tourneur sur bois ? (utiliser l'INTERSECTION)

8) Quels sont les candidats (numéro, nom, prénom) ayant passé le CAP Tourneur sur bois, et pas le BEP Tourneur sur bois ?

9) Quel est le nombre d'épreuves passées par le candidat N° 102 au CAP Tourneur sur bois

Chapitre IIII : LE LANGAGE SQL

Mme Z. TAHAKOURT

1- Introduction

SQL est un langage de définition, de manipulation et de contrôle de données relationnelles.

- Langage de définition des données (LDD) :
  Permet de créer les BD et les tables dans une BD relationnelle, ainsi qu'en modifier et en supprimer.
- Langage de manipulation de données(LMD) :
  Permet de sélectionner, insérer, modifier ou supprimer des données dans une table d'une BD relationnelle.
- Langage de contrôle de données (LCD) :
  Permet de protéger les données des accès non autorisés, il est possible avec SQL de définir des permissions au niveau des utilisateurs d'une BD.

2- Domaines de base

Les domaines utilisés dans les bases de données sont conformes aux types classiquement utilisés dans les langages de programmation, à savoir :

- Entier : INTEGER ;
- Décimal : DECIMAL (m,n) ou NUMBER(mn) où m est le nombre de chiffres et n le nombre de chiffres après la virgule. Ainsi DECIMAL(4,2) peut représenter des chiffres comme 99,99.
- Réel flottant : FLOAT ;
- Chaîne de caractères : CHAR (n) et VARCHAR(n). Les"CHAR" font systématiquement n caractères (les chaînes plus courtes sont complétées par des espaces, les VARCHAR sont taillés au plus juste dans la limite des n caractères ;

3- Base de données exemple

La base de données qui sera utilisée comme de support pour illustrer les requêtes exprimées tout au long de ce cours, est relative à une banque qui désire conserver des infos sur tous ses clients, et aussi sur toutes les agences de la banque mère.

```
Banque(Agence, Avoir, Ville)
Clientèle(Client, Adresse)
Dépôt(N°Cpt, Client, Agence, Solde)
Crédit(N°Prêt, Client, Agence, Montant)
```

4- Les commandes du LDD

4-1 Création d'un schéma de base de données

Requête1 : création de la base de données *BanqueMere* présentée ci-dessus.

```
CREATE DATABASE BanqueMere
```

4-2 Création d'un schéma de relation

Requête2 : création de la table *Agence* de la BD *BanqueMere*.

```
CREATE TABLE Banque
       (      Agence        CHAR(50),
              Avoir  FLOAT,
              Ville  CHAR(20)
       ) ;
```

4-3 Ajout d'un attribut

Requête3 : ajouter l'attribut *NomDirecteur* à la table *Agence*.

```
ALTER TABLE Banque
       ADD COLUMN NomDirecteur CHAR(50);
```

4-4 Suppression d'un schéma de relation

Requête4 : supprimer la table *Clientèle* de *BanqueMere*.

```
DROP TABLE Clientele ;
```

5- Les commandes du LMD

5-1 Langage d'interrogation

Syntaxe générale :

```
SELECT <liste d'attributs projetés>

FROM <liste de relations>

WHERE <liste de critères de restriction et de jointure>
```

Les clauses se renseignent dans l'ordre suivant :

- La partie FROM décrit les relations qui sont utilisables dans la requête (c'est à dire l'ensemble des attributs que l'on peut utiliser). C'est par là que l'on doit commencer par écrire une requête ;
- La partie WHERE exprime la (ou les conditions) que doivent respecter les attributs d'un tuple pour pouvoir être dans la réponse. Cette partie est optionnelle ;
- La partie SELECT indique le sous-ensemble des attributs qui doivent apparaitre dans la réponse (c'est le schéma de la relation résultat). Bien entendu ces attributs doivent appartenir aux relations indiquées dans la partie FROM.

A- Requêtes simples portant sur une seule relation

Les requêtes mono-relation ne concernent par définition qu'une seule relation. Les opérateurs de l'algèbre relationnelle exprimés sont donc la projection et la restriction.

Requête5 : trouver tous les clients de la banque.

En algèbre relationnelle : Clientele
En SQL :

```
SELECT *
FROM Clientele
```

Requête6 : trouver toutes les agences de la banque mère.

En algèbre relationnelle : $\pi_{\text{Agence}}(\text{Banque})$

En SQL :

```
SELECT NomAgence
FROM Agence
```

Requête7 : trouver tous les clients de la banque qui habitent la ville de « Brookleen » par ordre croissant de leurs noms.

En algèbre relationnelle : $\pi_{\text{Client}}[\ б_{\text{Adresse= 'Brookleen'}}(\text{Clientele})]$
Ici on ne peut pas exprimer l'ordre.
En SQL :

```
SELECT Client
FROM Clientele
WHERE Adresse= 'Brookleen'
ORDER BY Client
```

Requête8 : trouver les clients qui ont un compte à l'agence « Perryridge ».

En algèbre relationnelle : $\pi_{\text{Client}}[\ б_{\text{Agence= 'Perryridge'}}(\text{Depot})]$
En SQL :

```
SELECT Client
FROM Depot
WHERE Agence= 'Perryridge'
```

Requête9 : trouver les clients qui ont un compte à l'agence « Perryridge » dont le solde dépasse 3000 .

En algèbre relationnelle : $\pi_{Client}[\ б_{(Agence= 'Perryridge') et(Solde>3000)}(Depot)]$
En SQL :

```
SELECT Client
FROM Depot
WHERE (Agence= 'Perryridge')AND(Solde>3000)
```

Requête10 : trouver les clients qui ont un compte à l'agence « Perryridge » dont le solde dépasse 3000 et dont les nom commencent par 'B' ou 'b' .

En algèbre relationnelle : $\pi_{Client}[\ б_{(Agence= 'Perryridge') et(Solde>3000)}(Depot)]$
Ici on ne peut pas exprimer le fait que le nom commence par 'B' ou 'b'.

En SQL :

```
SELECT Client
FROM Depot
WHERE (Agence= 'Perryridge') AND(Solde>3000)AND((Client LIKE
'B%')OR(Client LIKE 'b%')
```

B- Requêtes portant sur plusieurs relations

Requête11 : trouver les clients qui ont bénéficié d'un crédit à l'agence « Perryridge » et qui ont un compte dans cette même agence.

En algèbre relationnelle :

$\pi_{Client}[\ б_{(Agence= 'Perryridge')}(Depot)] \cup \pi_{Client}[\ б_{(Agence= 'Perryridge')}(Credit)]$

En SQL :

```
SELECT Client
FROM Depot
WHERE (Agence= 'Perryridge')
UNION
SELECT Client
FROM Credit
WHERE (Agence= 'Perryridge')
```

Requête12 : trouver les clients qui ont un crédit et un compte à l'agence « Perryridge ».

En algèbre relationnelle :

$\pi_{Client}[\ б_{(Agence= 'Perryridge')}(Depot)] \cap \pi_{Client}[\ б_{(Agence= 'Perryridge')}(Credit)]$

En SQL :

```
SELECT Client
FROM Depot
WHERE (Agence= 'Perryridge')
INTERSECT
SELECT Client
FROM Credit
WHERE (Agence= 'Perryridge')
```

Requête13 : trouver les clients qui ont un compte à l'agence « Perryridge » mais pas de crédit dans cette même agence.

En algèbre relationnelle :

$\pi_{\text{Client}}[\ \sigma_{(\text{Agence= 'Perryridge'})}(\text{Depot})] - \pi_{\text{Client}}[\ \sigma_{(\text{Agence= 'Perryridge'})}(\text{Credit})]$

En SQL :

```
SELECT Client
FROM Depot
WHERE (Agence= 'Perryridge')
MINUS
SELECT Client
FROM Credit
WHERE (Agence= 'Perryridge')
```

Remarque :
Il faut être très vigilant lors de la manipulation des trois opérateurs ensemblistes l'union, l'intersection et la différence: les deux relations en entrée doivent impérativement être de même schéma.

Formulation de la jointure

Requête14 : trouver les clients avec leurs adresses, qui ont un compte à l'agence « Perryridge ».

En algèbre relationnelle :

$\pi_{\text{Client, Adresse}}[\ \sigma_{(\text{Agence= 'Perryridge'})}(\ \text{Depot} \bowtie \text{Clientele})]$

En SQL :

```
SELECT Client, Adresse
FROM Depot, Clientele
WHERE (Agence= 'Perryridge') AND (Depot.Client= Clientele.Client)
```

Ou,

```
SELECT Client, Adresse
FROM Depot d, Clientele c
WHERE (Agence= 'Perryridge') AND (d.Client=c.Client)
```

On peut également exprimerla jointure en utilisant l'opérateur IN,

```
SELECT Client, Adresse
FROM Clientele
WHERE (Client IN (SELECT Client
       FROM Depot
       WHERE Agence= 'Perryridge')
```

Requête15 : trouver les noms, l'adresse et le solde des clients ayant des comptes dans les agences de la ville de « brookleen ».

En algèbre relationnelle :

$\pi_{\text{Client,Adresse, Solde}} [\sigma_{\text{(Ville= 'Brookleen')}} \text{Banque} \bowtie (\text{Clientele} \bowtie \text{Depot})]$

En SQL :

```
SELECT Client,Adresse,Solde
FROM Banque b, Clientele c, Depot d
WHERE (Ville= 'Brookleen') AND (c.Client= d.Client) AND (d.Agence=
b.Agence)
```

Ou bien,

```
SELECT Client,Adresse,Solde
FROM Clientele c, Depot d
WHERE (c.Client= d.Client) AND
     (Agence IN
          (SELECT Agence
           FROM Banque
           WHERE Ville='Brookleen')
     )
```

5-2 Langage de mise à jour

A- Insertion de tuples

Syntaxe:

```
INSERT INTO <NomTable> [Col1, col2,…,Coln]
VALUES (val1, val2,… ,valn)
```

Requête16 : Insérer dans la base un nouveau client : <Ahmed, Tichy>.

```
INSERT INTO Clientele Client, Adresse
VALUES ('Ahmed', 'Tichy')
```

B- Suppression de tuples

Syntaxe:

```
DELETE FROM <NomTable>
[WHERE {CONDITION}]
```

Requête17 : supprimer de la base les agences de la ville de « Brookleen ».

```
DELETE FROM Banque
WHERE Ville=' Brookleen '
```

C- Modification de tuples

Syntaxe:

```
UPDATE <NomTable>
SET col1=nouv_val1, col2=nouv_val2,…, coln=nouv_valn
[WHERE {CONDITION}]
```

Requête18 : Modifier l'adresse du client 'Ahmed' suite à son déménagement, sa nouvelle adresse est 'Toudja'.

```
UPDATE CLIENTELE
SET Adresse='Toudja'
WHERE Client='Ahmed'
```

5-3 Autres formes de requêtes SQL

5-3-1 Les opérateurs IN et NOTIN

Requête19 : trouver les clients qui ont un compte et un crédit à l'agence « Perryridge ».

```
SELECT Client
FROM Credit
WHERE (Agence='Perryridge') and
(Client IN (SELECT Client
            FROM Depot
           WHERE Agence='Perryridge')
)
```

Requête20 : trouver les clients qui ont un compte à l'agence « Perryridge » mais pas de crédit dans cette agence.

```
SELECT Client
FROM Credit
WHERE (Agence='Perryridge') and
(Client NOTIN (SELECT Client
            FROM Depot
           WHERE Agence='Perryridge')
)
```

5-3-2 L'opérateurs >ANY

Le langage SQL offre la possibilité d'inclure l'opérateur >ANY, ce dernier est utile lorsque se présente dans la requête un segment de phrase tel que : Plus grand que quelconque.

Requête21 : trouver les agences dont les avoirs sont supérieurs à celui d'une agence de la ville de « Brookleen ».

```
SELECT Agence
FROM Banque
WHERE (Avoir >ANY
           (SELECT Avoir
           FROM Banque
           WHERE Ville='Brookleen')
      )
```

5-3-3 L'opérateurs >ALL

Cet opérateur est utile lorsque se présente dans la requête un segment de phrase tel que : Plus grand que tous.

Requête22 : trouver les agences dont les avoirs sont supérieurs à ceux de toutes les agences de la ville de « Brookleen ».

```
SELECT Agence
FROM Banque
WHERE (Avoir >ALL
           (SELECT Avoir
           FROM Banque
           WHERE Ville='Brookleen')
      )
```

5-3-4 L'opérateurs CONTAINS

Requête23 : trouver les clients qui ont un compte dans toutes les agences situées dans la ville de « Brookleen ».

En algèbre relationnelle : $\pi_{Client,Agence}(Depot) / \pi_{Agence}[\sigma_{(Ville='Brookleen')}(Banque)]$

```
SELECT s.Client                          (SELECT Agence
FROM depot d                              FROM Banque
WHERE (SELECT t.Agence                    WHERE Ville='Brookleen')
       FROM Depot
       WHERE s.Client=t.Client)
      CONTAINS
```

5-4 Les fonctions d'agrégation

Les agrégats sont des fonctions de calcul. Elles s'appliquent sur l'ensemble des valeurs prises par un attribut et renvoie une valeur unique. Cinq agrégats sont définis : Count, Sum, Avg, Min, Max. La fonction Count(*), un peu particulière, compte les tuples d'une relation.

5-4-1 COUNT

Requête24 : Quel est le nombre de clients de l'agence « Perryridge ».

```
SELECT COUNT(*)
FROM Depot
WHERE Agence=' Perryridge '
```

5-4-2 AVG

Requête25 : Quel est l'avoir moyen des agences de la banque.

```
SELECT AVG(Avoir)
FROM Banque
```

5-4-3 SUM

Requête26 : Quel le total des soldes des comptes du client « Ali ».

```
SELECT SUM(Solde)
FROM Depot
WHERE Client='Ali'
```

5-4-4 Min et Max

Requête27 : Quel est le plus petit (resp. grand) crédit octroyé par l'agence « Perryridge ».

```
SELECT MIN(Montant)
FROM Credit
WHERE Agence=« Perryridge ».
```

```
SELECT MAX(Montant)
FROM Credit
WHERE Agence=« Perryridge ».
```