**I - INTRODUCTION AUX BASES DE DONNEES**

1 - Qu’est-ce qu’une base de données

2 – Système de Gestion de Base de Données

2.1 Définition et principes de fonctionnement

2.2 Objectifs

2.3 Niveaux de description des données ANSI/SPARC

3 Quelques SGBD connus et utilisés

4 PostgreSQL

**II - SGBD RELATIONNEL**

1 - Introduction au modèle relationnel

2 - Éléments du modèle relationnel

3 - Algèbre relationnelle

3.1 Introduction

3.2 Sélection

3.3 Projection

3.4 Union

3.5 Intersection

3.6 Différence

3.7 Produit cartésien

3.8 Jointure, thêta-jointure, équijointure, jointure naturelle

a) Jointure

b) Thêta-jointure

c) Equijointure

d) Jointure naturelle

2.3.9 Division

**1.1 - Qu’est-ce qu’une base de données**

**Une base de données (BD)** est un ensemble structuré et organisé permettant le stockage de grandes quantités d’informations afin d'en cxfaciliter l'exploitation (ajout, mise à jour, recherche de données).

**1.2 – Système de Gestion de Base de Données**

**1.2.1 Définition et principes de fonctionnement**

**Un système de gestion de base de données (SGBD ou DBMS en anglais)** est un ensemble de programmes qui permet la gestion et l'accès à une base de données. Il héberge généralement plusieurs bases de données, qui sont destinées à des logiciels ou des thématiques différentes.

On distingue couramment les SGBD classiques, dits **SGBD relationnels** (SGBD-R), des **SGBD orientés objet** (SGBD-O). En fait, un SGBD est caractérisé par **le modèle de description des données** qu’il supporte (relationnel, objet etc.). Les données sont décrites sous la forme de ce modèle, grâce à un Langage de Description des Données (LDD). Cette description est appelée schéma.

Une fois la base de données spécifiée, on peut y insérer des données, les récupérer, les modifier et les détruire. Les données peuvent être manipulées non seulement par un Langage spécifique de Manipulation des Données (LMD) mais aussi par des langages de programmation classiques.

Actuellement, la plupart des SGBD fonctionnent selon un mode client/serveur. Le serveur (sous-entendu la machine qui stocke les données) reçoit des requêtes de plusieurs clients et ceci de manière concurrente. Le serveur analyse la requête, la traite et retourne le résultat au client.

Quel que soit le modèle, un des problèmes fondamentaux à prendre en compte est la cohérence des données. Par exemple, dans un environnement où plusieurs utilisateurs peuvent accéder concurremment à une colonne d’une table par exemple pour la lire ou pour l’écrire, il faut s’accorder sur la politique d’écriture. Cette politique peut être : les lectures concurrentes sont autorisées mais dès qu’il y a une écriture dans une colonne, l’ensemble de la colonne est envoyée aux autres utilisateurs l’ayant lue pour qu’elle soit rafraîchie.

**1.2.2 Objectifs**

**Les principaux objectifs fixés aux SGBD** afin de résoudre les problèmes causés par une gestion sous forme de fichiers à plat sont les suivants :

· **Indépendance physique** : La façon dont les données sont définies doit être indépendante des structures de stockage utilisées.

· **Indépendance logique** : Un même ensemble de données peut être vu différemment par des utilisateurs différents. Toutes ces visions personnelles des données doivent être intégrées dans une vision globale.

· **Accès aux données** : L’accès aux données se fait par l’intermédiaire d’un Langage de Manipulation de Données (LMD = insert, update, delete). Il est crucial que ce langage permette d’obtenir des réponses aux requêtes en un temps « raisonnable ». Le LMD doit donc être optimisé, minimiser le nombre d’accès disques, et tout cela de façon totalement transparente pour l’utilisateur.

· **Administration centralisée des données (intégration)** : Toutes les données doivent être centralisées dans un réservoir unique commun à toutes les applications. En effet, des visions différentes des données (entre autres) se résolvent plus facilement si les données sont administrées de façon centralisée.

· **Non redondance des données** : Afin d’éviter les problèmes lors des mises à jour, chaque donnée ne doit être présente qu’une seule fois dans la base.

· **Cohérence des données** : Les données sont soumises à un certain nombre de contraintes d’intégrité qui définissent un état cohérent de la base. Elles doivent pouvoir être exprimées simplement et vérifiées automatiquement à chaque insertion, modification ou suppression des données. Les contraintes d’intégrité sont décrites dans le Langage de Description de Données (LDD).

· **Partage des données** : Il s’agit de permettre à plusieurs utilisateurs d’accéder aux mêmes données au même moment de manière transparente. Si ce problème est simple à résoudre quand il s’agit uniquement d’interrogations, cela ne l’est plus quand il s’agit de modifications dans un contexte multi-utilisateurs car il faut : permettre à deux (ou plus) utilisateurs de modifier la même donnée « en même temps » et assurer un résultat d’interrogation cohérent pour un utilisateur consultant une table pendant qu’un autre la modifie.

· **Sécurité des données** : Les données doivent pouvoir être protégées contre les accès non autorisés. Pour cela, il faut pouvoir associer à chaque utilisateur des droits d’accès aux données.

· **Résistance aux pannes** : Que se passe-t-il si une panne survient au milieu d’une modification, si certains fichiers contenant les données deviennent illisibles ? Il faut pouvoir récupérer une base dans un état « sain ». Ainsi, après une panne intervenant au milieu d’une modification deux solutions sont possibles : soit récupérer les données dans l’état dans lequel elles étaient avant la modification, soit terminer l’opération interrompue.

**1.2.3 Niveaux de description des données ANSI/SPARC**

Pour atteindre certains de ces objectifs (surtout les deux premiers), trois niveaux de description des données ont été définis par la norme ANSI/SPARC :

· **Le niveau externe** correspond à la perception de tout ou partie de la base par un groupe donné d’utilisateurs, indépendamment des autres. On appelle cette description le schéma externe ou vue. Il peut exister plusieurs schémas externes représentant différentes vues sur la base de données avec des possibilités de recouvrement. Le niveau externe assure l’analyse et l’interprétation des requêtes en primitives de plus bas niveau et se charge également de convertir éventuellement les données brutes, issues de la réponse à la requête, dans un format souhaité par l’utilisateur.

· **Le niveau conceptuel** décrit la structure de toutes les données de la base, leurs propriétés (i.e. les relations qui existent entre elles : leur sémantique inhérente), sans se soucier de l’implémentation physique ni de la façon dont chaque groupe de travail voudra s’en servir.

Dans le cas des SGBD relationnels, il s’agit d’une vision tabulaire où la sémantique de l’information est exprimée en utilisant les concepts de relation, attributs et de contraintes d’intégrité. On appelle cette description le schéma conceptuel.

· **Le niveau interne ou physique** s’appuie sur un système de gestion de fichiers (SGF) pour définir la politique de stockage ainsi que le placement des données (SGF OU SCHEMA INTERNE). Le niveau physique est donc responsable du choix de l’organisation physique des fichiers ainsi que de l’utilisation de telle ou telle méthode.

**SGBD INTERNE SGBD EXTERNE**

**Ordinateur**

**1.3 Quelques SGBD connus et utilisés**

Il existe de nombreux systèmes de gestion de bases de données, en voici une liste non exhaustive :

**ACCESS** : plate-forme Windows, monoposte, licence commerciale

**SQL SERVER**: plate-forme Windows, mode client/serveur, licence commerciale

**ORACLE**: plateformes Windows et Linux, mode client/serveur, licence commerciale

**SYBASE** : plateformes Windows et Linux, mode client/serveur, licence commerciale

**POSTGRESQL** : plateformes Windows et Linux, mode client/serveur, licence libre

**MYSQL** : plateformes Windows et Linux, mode client/serveur, licence libre

**1.4 PostgreSQL**

Les systèmes traditionnels de gestion de bases de données relationnelles (SGBDR) offrent un modèle de données composé d’une collection de relations contenant des attributs relevant chacun d’un type spécifique. PostgreSQL apporte une puissance additionnelle substantielle en incorporant les quatre concepts de base suivants afin que les utilisateurs puissent facilement étendre le système : classes, héritage, types, fonctions. D’autres fonctionnalités accroissent la puissance et la souplesse : contraintes, déclencheurs, règles, intégrité des transactions.

Ces fonctionnalités placent PostgreSQL dans la catégorie des bases de données relationnelles objet mais bien que PostgreSQL possède certaines fonctionnalités orientées objet, il appartient avant tout au monde des SGBDR. C’est essentiellement l’aspect SGBDR de PostgreSQL qui sera abordé dans ce cours.

L’une des principales qualités de PostgreSQL est d’être un logiciel libre, c’est-à-dire gratuit et dont les sources sont disponibles. Il est possible de l’installer sur les systèmes Unix/Linux et Win32.

PostgreSQL fonctionne selon une architecture client/serveur, il est ainsi constitué :

– d’une partie serveur, c’est-à-dire une application fonctionnant sur la machine hébergeant la base de données (le serveur de bases de données) capable de traiter les requêtes des clients;

– d’une partie client devant être installée sur toutes les machines nécessitant d’accéder au serveur de base de données (un client peut éventuellement fonctionner sur le serveur lui-même).

**II - SGBD relationnel**

**2.1 - Introduction au modèle relationnel**

**Modèle hiérarchique**

**RACINE**

**N1 N2 N3 N4**

**N5 N6**

**Modèle réseau**

**I1 I2**

**I3**

**I4**

**I5**

**Modèle relationnel (SGBDR)**

**Modèle orienté objet (SGBDOO)**

**Modèle relationnel-objet (SGBDRO)**

Edgar Frank Codd, chercheur chez IBM, étudiait à la fin des années 1960, a mis en place de nouvelles méthodes pour gérer de grandes quantités de données car les modèles et les logiciels de l’époque ne le satisfaisaient pas. Mathématicien de formation, il était persuadé qu’il pourrait utiliser des branches spécifiques des mathématiques (la théorie des ensembles et la logique des prédicats du premier ordre) pour résoudre des difficultés telles que la redondance des données, l’intégrité des données ou l’indépendance de la structure de la base de données avec sa mise en œuvre physique.

En 1970, Codd publia un article où il proposait de stocker des données hétérogènes dans des tables, permettant d’établir des relations entre elles.

Un premier prototype de Système de gestion de bases de données relationnelles a été construit dans les laboratoires d’IBM. Depuis les années 80, cette technologie a mûri et a été adoptée par l’industrie. En 1987, le langage SQL, qui étend l’algèbre relationnelle, a été standardisé.

Dans ce modèle, les données sont représentées par des tables, sans préjuger de la façon dont les informations sont stockées dans la machine. Les tables constituent donc la structure logique du modèle relationnel. Au niveau physique, le système est libre d’utiliser n’importe quelle technique de stockage dès lors qu’il est possible de relier ces structures à des tables au niveau logique. Les tables ne représentent donc qu’une abstraction de l’enregistrement physique des données en mémoire.

Le succès du modèle relationnel auprès des chercheurs, concepteurs et utilisateurs est dû à la puissance et à la simplicité de ses concepts. En outre, contrairement à certains autres modèles, il repose sur des bases théoriques solides, notamment la théorie des ensembles et la logique des prédicats du premier ordre.

Les objectifs du modèle relationnel sont :

– proposer des schémas de données faciles à utiliser ;

– améliorer l’indépendance logique et physique;

– mettre à la disposition des utilisateurs des langages de haut niveau ;

– optimiser les accès à la base de données;

– améliorer l’intégrité et la confidentialité ;

– fournir une approche méthodologique dans la construction des schémas.

De façon informelle, on peut définir le modèle relationnel de la manière suivante:

– les données sont organisées sous forme de tables à deux dimensions, encore appelées relations, dont les lignes sont appelées n-uplet ou tuple en anglais;

– les données sont manipulées par des opérateurs de l’algèbre relationnelle;

– l’état cohérent de la base est défini par un ensemble de contraintes d’intégrité.

**2.2 Éléments constitutifs du modèle relationnel**

**Un attribut** est un identificateur (un nom) décrivant une information stockée dans une base.

Ex : l’âge d’une personne, le nom d’une personne, le numéro de sécurité sociale.

**Le domaine d’un attribut** est l’ensemble, fini ou infini, de ses valeurs possibles.

Par exemple, l’attribut numéro de sécurité sociale a pour domaine l’ensemble des combinaisons de quinze chiffres et nom a pour domaine l’ensemble des combinaisons de lettres (chaîne de caractère).

**Une relation** est un sous-ensemble du produit cartésien de n domaines d’attributs (n > 0). Elle est représentée sous la forme d’un tableau à deux dimensions dans lequel les n attributs correspondent aux titres des n colonnes.

**Un schéma de relation** précise le nom de la relation ainsi que la liste des attributs avec leurs domaines.

**PERSONNE**

|  |  |  |
| --- | --- | --- |
| **N°SECU** | **NOM** | **PRENOM** |
| 100 | NDIAYE | MALLE |
| 200 | FALL | MOUSSA |
| 300 | FAYE | NGOR |
| 400 | SY | DJIBY |

Exemple du **schéma de relation**

Personne (N° sécu : Entier, Nom : Chaîne, Prénom : Chaîne)

La relation Personne peut être simplifiée comme suit :

Personne (N° sécu, Nom, Prénom)

**Le degré d’une relation** est son nombre d’attributs.

**Une occurrence,** ou **n-uplets,** ou **tuples**, est un élément de l’ensemble figuré par une relation.

Autrement dit, une occurrence est une ligne du tableau qui représente la relation.

La **cardinalité d’une relation** est son nombre d’occurrences.

**Une clé candidate** d’une relation est un ensemble minimal des attributs de la relation dont les valeurs identifient à coup sûr une occurrence.

La valeur d’une clé candidate est donc distincte pour tous les tuples de la relation. La notion de clé candidate est essentielle dans le modèle relationnel.

Toute relation a au moins une clé candidate et peut en avoir plusieurs. Ainsi, il ne peut jamais y avoir deux tuples identiques au sein d’une relation. Les clés candidates d’une relation n’ont pas forcément le même nombre d’attributs. Une clé candidate peut être formée d’un attribut arbitraire, utilisé à cette seule fin.

**La clé primaire** d’une relation est une de ses clés candidates. Pour signaler la clé primaire, ses attributs sont généralement soulignés.

**Une clé étrangère** dans une relation est formée d’un ou plusieurs attributs qui constituent une clé primaire dans une autre relation.

Un schéma relationnel est constitué par l’ensemble des schémas de relation.

Une base de données relationnelle est constituée par l’ensemble des n-uplets des différentes relations du schéma relationnel

**2.3 Algèbre relationnelle**

**2.3.1 Introduction**

L’algèbre relationnelle est un support mathématique cohérent sur lequel repose le modèle relationnel. L’objet de cette section est d’aborder l’algèbre relationnelle dans le but de décrire les opérations qu’il est possible d’appliquer sur des relations pour produire de nouvelles relations.

L’approche suivie est donc plus opérationnelle que mathématique.

On peut distinguer trois familles d’opérateurs relationnels :

- **Les opérateurs unaires (Sélection, Projection)** : ce sont les opérateurs les plus simples, ils permettent de produire une nouvelle table à partir d’une autre table.

- **Les opérateurs binaires ensemblistes (Union, Intersection Différence)** : ces opérateurs permettent de produire une nouvelle relation à partir de deux relations de même degré et de même domaine.

- **Les opérateurs binaires ou n-aires (Produit cartésien, Jointure, Division)** : ils permettent de produire une nouvelle table à partir de deux ou plusieurs autres tables.

Les notations ne sont pas standardisées en algèbre relationnelle. Ce cours utilise des notations courantes mais donc pas forcément universelles.

**2.3.2 Sélection (opérateur unaire)**

La sélection génère une relation regroupant exclusivement toutes les occurrences de la relation R qui satisfont le prédicat en ramenant tous les attributs, on la note :

**σ prédicat (R).**

**EXEMPLE**

En d’autres termes, la sélection permet de choisir (i.e. sélectionner) des lignes dans le tableau. Le résultat de la sélection est donc une nouvelle relation qui a les mêmes attributs que R. Si R est vide (i.e. ne contient aucune occurrence), la relation qui résulte de la sélection est vide.

Le tableau 2 montre un exemple de sélection.

**ETUDIANT**

|  |  |  |
| --- | --- | --- |
| **NUMERO** | **NOM** | **PRENOM** |
| **10** | **DIAITE** | **BOUBACAR** |
| **20** | **NDIAYE** | **MAIMOUNA** |
| **30** | **NDIAYE** | **AMADOU LAMINE** |

Exemple de sélection sur la relation ETUDIANT

σNom=’NDIAYE’(Personne)

|  |  |  |
| --- | --- | --- |
| **NUMERO** | **NOM** | **PRENOM** |
| 20 | NDIAYE | MAIMOUNA |
| 30 | NDIAYE | AMADOU LAMINE |

**2.3.3 Projection (opérateur unaire)**

La projection consiste à supprimer les attributs autres que A1, … An d’une relation et à éliminer les n-uplets qui apparaissent en double dans la nouvelle relation ; on la note **Π (A1, … An) R**

**Exemple**

**Afficher le nom et le prénom de toutes les personnes**

**Π (Nom, Prenom) PERSONNE**

|  |  |
| --- | --- |
| **NOM** | **PRENOM** |
| DIAITE | BOUBACAR |
| NDIAYE | MAIMOUNA |
| NDIAYE | AMADOU LAMINE |

En d’autres termes, la projection permet de choisir des colonnes dans le tableau. Si R est vide, la relation qui résulte de la projection est vide, mais pas forcément équivalente (elle contient généralement moins d’attributs).

**ETUDIANT**

|  |  |  |
| --- | --- | --- |
| **NUMERO** | **NOM** | **PRENOM** |
| 10 | DIAITE | BOUBACAR |
| 20 | NDIAYE | MAIMOUNA |
| 30 | NDIAYE | AMADOU LAMINE |

Exemple de projection sur la relation ETUDIANT (projection sur le nom et le prénom.

Π NOM, PRENOM (ETUDIANT)

|  |  |
| --- | --- |
| **NOM** | **PRENOM** |
| DIAITE | BOUBACAR |
| NDIAYE | MAIMOUNA |
| NDIAYE | AMADOU LAMINE |

REMRQUES

Le prédicat utilise généralement les opérateurs de comparaison suivants : <, >, =, >=, <=, <> ( !=).

Soient les relations suivantes :

PRODUIT (Code :Chaine, Designation : Chaine, Stock : entier, Seuil :entier)

Client (Numero : entier, Nom : Chaine, Adresse : Chaine)

1°) Afficher les produits qui ont un stock supérieur à 100.

**R1=σ Stock>100(PRODUIT)**

**Prédicat**

2°) Afficher les produits dont le stock est inférieur au seuil.

**R1=σ Stock<seuil(PRODUIT)**

3°) Afficher les clients qui habitent Dakar

**σ Adresse=’DAKAR’(CLIENT)**

4°) Afficher le code et la désignation des produits dont le seuil est inférieur à 150.

5°) Afficher le nom et l’adresse des clients qui habitent à THIES.

**2.3.4 Union (opérateur binaire)**

L’union (U) est une opération portant sur deux relations **R1 et R2 ayant le** **même schéma** et construisant une troisième relation constituée des n-uplets appartenant à chacune des deux relations R1 et R2 sans doublon, on la note R1 ∪ R2.

R1 U R2 = R2 U R1(Commutativité)

Comme nous l’avons déjà dit, R1 et R2 doivent avoir les mêmes attributs et si une même occurrence existe dans R1 et R2, elle n’apparaît qu’une seule fois dans le résultat de l’union. Le résultat de l’union est une nouvelle relation qui a les mêmes attributs que R1 et R2. Si R1 et R2 sont

vides, la relation qui résulte de l’union est vide. Si R1 (respectivement R2) est vide, la relation qui résulte de l’union est identique à R2 (respectivement R1).

**COMMUTATIVITE**

R1 ∪ R2= R2 ∪ R1

**EXEMPLE**

**Relation** R1

|  |  |  |
| --- | --- | --- |
| NUMERO | NOM | PRENOM |
| 10 | DIOP | MOUSSA |
| 20 | NDIAYE | AMADOU |
| 30 | DIAITE | BOUBACAR |
| 40 | NDIAYE | MAIMOUNA |

**Relation** R2

|  |  |  |
| --- | --- | --- |
| NUMERO | NOM | PRENOM |
| 10 | DIOP | MOUSSA |
| 40 | NDIAYE | BINETA |
| 50 | DIOP | GOLO |
| 60 | SY | MOUSSA |

**La relation R1 ∪ R2 donne**

**Relation** R3 = R1 U R2 (renvoie tous les tuples de R1 et les tuples de R2 qui ne figurent dans R1)

|  |  |  |
| --- | --- | --- |
| NUMERO | NOM | PRENOM |
| 10 | DIOP | MOUSSA |
| 20 | NDIAYE | AMADOU |
| 30 | DIAITE | BOUBACAR |
| 40 | NDIAYE | MAIMOUNA |
| 40 | NDIAYE | BINETA |
| 50 | DIOP | GOLO |
| 60 | SY | MOUSSA |

**2.3.5 Intersection**

L’intersection est une opération portant sur deux relations R1 et R2 ayant le même schéma et construisant une troisième relation dont les n-uplets sont constitués de ceux appartenant aux deux relations, on la note

R1 ∩ R2.

R1 ∩ R2 = R2 ∩ R1 (Commutativité)

Comme nous l’avons déjà dit, R1 et R2 doivent avoir les mêmes attributs. Le résultat de l’intersection est une nouvelle relation qui a les mêmes attributs que R1 et R2. Si R1 ou R2 ou les deux sont vides, la relation qui résulte de l’intersection est vide.

**EXEMPLE**

**Relation** R1

|  |  |  |
| --- | --- | --- |
| NUMERO | NOM | PRENOM |
| 10 | DIOP | MOUSSA |
| 20 | NDIAYE | AMADOU |
| 30 | DIAITE | BOUBACAR |
| 40 | NDIAYE | MAIMOUNA |

**Relation** R2

|  |  |  |
| --- | --- | --- |
| NUMERO | NOM | PRENOM |
| 10 | DIOP | MOUSSA |
| 40 | NDIAYE | BINETA |
| 30 | DIAITE | BOUBACAR |
| 60 | SY | MOUSSA |

**La relation R1∩ R2 donne**

**Relation** R3 = R1 ∩ R2 (renvoie tous les tuples qui figurent à la fois dans R1 et dans R2)

|  |  |  |
| --- | --- | --- |
| NUMERO | NOM | PRENOM |
| 10 | DIOP | MOUSSA |
| 30 | DIAITE | BOUBACAR |

**2.3.6 Différence**

La différence est une opération portant sur deux relations R1 et R2 ayant le même schéma et construisant une troisième relation dont les n-uplets sont constitués de ceux ne se trouvant que dans la relation R1 (les tuples se trouvant dans R1 et ne se trouvant pas dans R2); on la note R1 − R2.

Il s’agit une opération binaire ensembliste non commutative essentielle dont la signature est : relation \* relation -> relation

R1-R2 est différent de R2-R1 (pas commutatif)

Comme nous l’avons déjà dit, R1 et R2 doivent avoir les mêmes attributs. Le résultat de la différence est une nouvelle relation qui a les mêmes attributs que R1 et R2. Si R1 est vide, la relation qui résulte de la différence est vide. Si R2 est vide, la relation qui résulte de la différence est identique à R1.

**EXEMPLE**

**Relation** R1

|  |  |  |
| --- | --- | --- |
| NUMERO | NOM | PRENOM |
| 10 | DIOP | MOUSSA |
| 20 | NDIAYE | AMADOU |
| 30 | DIAITE | BOUBACAR |
| 40 | NDIAYE | MAIMOUNA |

**Relation** R2

|  |  |  |
| --- | --- | --- |
| NUMERO | NOM | PRENOM |
| 10 | DIOP | MOUSSA |
| 40 | NDIAYE | BINETA |
| 30 | DIAITE | BOUBACAR |
| 60 | SY | MOUSSA |

**R1-R2 donne R3**

**Relation R3 est :**

|  |  |  |
| --- | --- | --- |
| NUMERO | NOM | PRENOM |
| 20 | NDIAYE | AMADOU |
| 40 | NDIAYE | MAIMOUNA |

**R2-R1**

|  |  |  |
| --- | --- | --- |
| NUMERO | NOM | PRENOM |
| 40 | NDIAYE | BINETA |
| 60 | SY | MOUSSA |

**Donc R1-R2 est différente de R2-R1, par conséquent l’opérateur de différence n’est pas commutatif.**

**2.3.7 Produit cartésien**

Le produit cartésien est une opération portant sur deux relations R1 et R2 et qui construit une troisième relation regroupant exclusivement toutes les possibilités de combinaison des occurrences des relations R1 et R2, on la note R1 × R2.

R3 = R1 X R2

R3 est une relation est une relation qui a comme attributs l’ensemble des attributs de R1 et de R2.

R3 est une relation est une relation qui a comme nombre de lignes le nombre de lignes de R1 multiplié par le nombre de lignes de R2.

Le résultat du produit cartésien est une nouvelle relation qui a tous les attributs de R1 et tous ceux de R2. Si R1 ou R2 ou les deux sont vides, la relation qui résulte du produit cartésien est vide. Le nombre d’occurrences de la relation qui résulte du produit cartésien est le nombre d’occurrences de R1 multiplié par le nombre d’occurrences de R2.

FAMILLE

|  |  |
| --- | --- |
| **CODEF** | **NOMF** |
| F01 | ALIMENTAIRE |
| F02 | COSMETIQUE |
| F03 | ELECTRONIQUE |

PRODUIT

|  |  |  |
| --- | --- | --- |
| **CODEP** | **NOMP** | **PU** |
| P01 | LAIT | 1500 |
| P02 | IPHONE X | 800000 |
| P03 | PARFUM XXL | 15000 |
| P04 | FARINE | 1000 |

FAMILLE X PRODUIT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CODEF** | **NOMF** | **CODEP** | **NOMP** | **PU** |
| F01 | ALIMENTAIRE | P01 | LAIT | 1500 |
| F01 | ALIMENTAIRE | P02 | IPHONE X | 800000 |
| F01 | ALIMENTAIRE | P03 | PARFUM XXL | 15000 |
| F01 | ALIMENTAIRE | P04 | FARINE | 1000 |
| F02 | COSMETIQUE | P01 | LAIT | 1500 |
| F02 | COSMETIQUE | P02 | IPHONE X | 800000 |
| F02 | COSMETIQUE | P03 | PARFUM XXL | 15000 |
| F02 | COSMETIQUE | P04 | FARINE | 1000 |
| F03 | ELECTRONIQUE | P01 | LAIT | 1500 |
| F03 | ELECTRONIQUE | P02 | IPHONE X | 800000 |
| F03 | ELECTRONIQUE | P03 | PARFUM XXL | 15000 |
| F03 | ELECTRONIQUE | P04 | FARINE | 1000 |

**TD**

Soient les schémas de relation suivants :

CATEGORIE (CodeCat, NomCat)

PRODUIT (CodePro, NomPro, PrixPro, StockPro, SeuilPro, **#**CodeCat)

CLIENT (NumCli, NomCli, AdresseCli, TelCli)

COMMANDE (NumCom, DateCom, **#**NumCli)

DETCOM (**#**CodePro,#NumCom, Qte, Prix)

**QUESTIONS**

1°) Afficher le nom et l’adresse de tous les clients

2°) Afficher la liste des produits qui ont atteint leur seuil.

3°) Afficher le nom et l’adresse de tous les clients qui habitent Dakar.

Affichons d’abord tous les clients qui habitent Dakar

4°) Afficher la liste des produits dont le stock est compris entre 100 et 200.

5°) Afficher le code, le nom et le seuil de tous les produits dont le stock est égal à zéro.

**2.3.8 Jointure, thêta-jointure, équijointure, jointure naturelle**

**a) Jointure**

La jointure est une opération portant sur deux relations R1 et R2 qui construit une troisième relation regroupant exclusivement toutes les possibilités de combinaison des occurrences des relations R1 et R2 qui satisfont l’expression logique E. La jointure est notée

R1 R2

Expression logique

Il s’agit d’une opération binaire commutative dont la signature est:

(Voir schéma ci-avant)

Si R1 ou R2 ou les deux sont vides, la relation qui résulte de la jointure est vide.

En fait, la jointure n’est rien d’autre qu’un produit cartésien suivi d’une sélection :

R1 ▷◁E R2 = σE (R1 × R2)

**Relation** Famille

|  |  |
| --- | --- |
| **CODEF** | NOMF |
| F01 | Alimentaire |
| F02 | Electronique |
| F03 | Cosmétique |

**Relation Produit**

|  |  |  |
| --- | --- | --- |
| CODEP | NOMP | **CODEF** |
| P01 | Sucre | F01 |
| P02 | Parfum | F03 |
| P03 | Farine | F01 |
| P04 | Iphone 11 | F02 |

PRODUIT CARTESIEN

Le produit cartésien de Famille et de produit est une relation qui a pour attributs CODEF, NOMF, CODEP, NOMP, CODEF et dont le nombre de tuples est égal au nombre de tuples de Famille multiplié par le nombre de tuples de Produit.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CODEF | NOMF | CODEP | NOMP | CODEF |
| F01 | Alimentaire | P01 | Sucre | F01 |
| F01 | Alimentaire | P02 | Parfum | F03 |
| F01 | Alimentaire | P03 | Farine | F01 |
| F01 | Alimentaire | P04 | Iphone 11 | F02 |
| F02 | Electronique | P01 | Sucre | F01 |
| F02 | Electronique | P02 | Parfum | F03 |
| F02 | Electronique | P03 | Farine | F01 |
| F02 | Electronique | P04 | Iphone 11 | F02 |
| F03 | Cosmétique | P01 | Sucre | F01 |
| F03 | Cosmétique | P02 | Parfum | F03 |
| F03 | Cosmétique | P03 | Farine | F01 |
| F03 | Cosmétique | P04 | Iphone 11 | F02 |

**b) Thêta-jointure**

Une thêta-jointure est une jointure dans laquelle l’expression logique E est une simple comparaison entre un attribut A1 de la relation R1 et un attribut A2 de la relation R2. La thêta-jointure est notée

Famille Produit

Famille.CODEF=Produit.CODEF

On restreint le résultat du produit cartésien en utilisant la condition de jointure Famille.CODEF =Produit.CODEF

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CODEF | NOMF | CODEP | NOMP | CODEF |
| F01 | Alimentaire | P01 | Sucre | F01 |
| F01 | Alimentaire | P03 | Farine | F01 |
| F02 | Electronique | P04 | Iphone 11 | F02 |
| F03 | Cosmétique | P02 | Parfum | F03 |

**c) Equijointure**

Une équijointure est une thêta-jointure dans laquelle l’expression logique E est un test d’égalité entre un attribut A1 de la relation R1 et un attribut A2 de la relation R2.

L’exemple ci-avant est une équijointure entre la relation Famille et la relation Produit sur la base de l’attribut CODEF.

Remarque : Il vaut mieux écrire R1 ▷◁A1=A2 R2 que R1 ▷◁A1,A2 R2 car cette dernière notation peut prêter à confusion avec une jointure naturelle explicite

**d) Jointure naturelle**

Une jointure naturelle est une jointure dans laquelle l’expression logique E est un test d’égalité entre les attributs qui portent le même nom dans les relations R1 et R2. Dans la relation construite, ces attributs ne sont pas dupliqués mais fusionnés en une seul colonne par couple d’attributs. La jointure naturelle est notée R1 ▷◁ R2. On peut préciser explicitement les attributs communs à R1 et R2 sur lesquels porte la jointure : R1 ▷◁A1, …, An R2.

Généralement, R1 et R2 n’ont qu’un attribut en commun. Dans ce cas, une jointure naturelle est équivalente à une équijointure dans laquelle l’attribut de R1 et celui de R2 sont justement les deux attributs qui portent le même nom.

**EXERCICE**

Soient les schémas de relation suivants :

CATEGORIE (CodeCat, NomCat)

PRODUIT (CodePro, NomPro, PrixPro, StockPro, SeuilPro, **#**CodeCat)

CLIENT (NumCli, NomCli, AdresseCli, TelCli)

COMMANDE (NumCom, DateCom, **#**NumCli)

DETCOM (**#**CodePro,#NumCom, Qte, Prix)

**1°)** Afficher pour chaque commande, la date, le numéro et le nom du client.

La relation R1 se retrouve avec toutes les propriétés de CLIENT et de COMMANDE. Pour restreindre R1 aux propriétés demandées c’est à dire la date et le numéro de la commande et le nom du client, on fait une projection sur R1.

**2°)** Afficher pour chaque commande, la date, le numéro et le nom du client. On ne retiendra que les commandes effectuées entre le 01-01-2020 et le 31-03-2020.

La relation R2 de la question 1°) affiche la date, le numéro de commande et le nom du client de toutes les factures. Par conséquent, pour répondre à cette question, il suffira juste de restreindre la relation R2 sur la période indiquée comme suit :

**3°)** Afficher les détails des commandes (NomPro, Quantité, Date de la commande et le numéro de la commande).

Les attributs de R5 sont constitués des attributs de COMMANDE, DETCOM et PRODUIT.

**4°)** Afficher pour chaque produit, le code, le nom et le nom de la catégorie du produit.

**AUTRES OPERATEURS SPECIFIQUES**

* **UNION**

L’opérateur UNION notée U entre deux relations R1 et R2 est une relation R3 contenant les tuples appartenant à R1 et ceux appartenant à R2.

NB

Les relations R1 et R2 doivent la même structure c’est-à-dire même structure (même nombre d’attributs et les attributs de même rang doivent avoir le même type de données)

**Division**

La division est une opération portant sur deux relations R1 et R2, telles que le schéma de R2 est strictement inclus dans celui de R1, qui génère une troisième relation regroupant toutes les parties d’occurrences de la relation R1 qui sont associées à toutes les occurrences de la relation R2 ; on la note R1 ÷ R2.

Il s’agit d’une opération binaire non commutative dont la signature est :

relation × relation —→ relation

Autrement dit, la division de R1 par R2 (R1 ÷ R2) génère une relation qui regroupe tous les n-uplets qui, concaténés à chacun des n-uplets de R2, donne toujours un n-uplet de R1.

La relation R2 ne peut pas être vide. Tous les attributs de R2 doivent être présents dans R1 et R1 doit posséder au moins un attribut de plus que R2 (inclusion stricte). Le résultat de la division est une nouvelle relation qui a tous les attributs de R1 sans aucun de ceux de R2. Si R1 est vide, la relation qui résulte de la division est vide.

**Relation** Enseignement

**Relation** Etudiant

**Relation** R

Germain Durand

Robert Durand

Tableau 10 : Exemple de division : R = Enseignement ÷ Etudiant.

La relation R contient donc tous les enseignants de la relation Enseignement qui enseignent à tous les étudiants de la relation Etudiant.

**EXERCICE**

**TypeCompte** (IDTypeCompte, NomTypeCompte)

**Client** (NumeroCni, NomClient, AdresseClient, TelephoneClient)

**Compte** (NumeroCp, DateCreation, DateFermeture, Solde, #NumeroCni, #IDTypeCompte)

**Operation** (IdOpe, DateOpe, MontantOpe, SensOpe, #NumeroCp)

**QUESTIONS**

1°) Afficher le numéro et la date de création de tous les comptes

2°) Afficher le numéro de compte, le numéro de la CNI et le solde de tous les comptes

3°) Afficher le numéro de compte, le numéro de la CNI et le solde de tous les comptes dont le solde est supérieur 1 500 000 FCFA.

4°) Afficher les opérations du compte C00025 effectuées au cours du premier trimestre de l’année 2020.

5°) Afficher le numéro de compte et le nom du client qui n’ont effectué aucune opération.

6°) Afficher les types de compte qui ne contiennent aucun compte.

**Division**

La division est une opération portant sur deux relations R1 et R2, telles que le schéma de R2 est strictement inclu dans celui de R1, qui génère une troisième relation regroupant toutes les parties d’occurrences de la relation R1 qui sont associées à toutes les occurrences de la relation R2 ; on la note R1 ÷ R2.

Il s’agit d’une opération binaire non commutative dont la signature est :

relation ÷ relation —→ relation

Autrement dit, la division de R1 par R2 (R1 ÷ R2) génère une relation qui regroupe tous les n-uplets de R1 qui, concaténés à chacun des n-uplets de R2, donne toujours un n-uplet de R1.

EXEMPLE

**R1**

|  |  |  |  |
| --- | --- | --- | --- |
| **A1** | **A2** | **A3** | **A4** |
| X | Y | Z | T |
| X | Z | Y | S |
| Y | X | T | Z |
| Z | Y | T | X |

R2

|  |  |
| --- | --- |
| **A3** | **A4** |
| Y | S |
| S | Y |
| T | X |

Si R3=R1÷R2

Quels sont les attributs de R3 :ce sont les attributs de R1 qui ne figurent dans R2. Ce sont les attributs de R1 tel que si on les combine avec les attributs de R2, on obtient un tuple (n-uplet ou ligne) complet de R1.

Donner R3

|  |  |
| --- | --- |
| **A1** | **A2** |
|  |  |
| X | Z |
| Z | Y |

**TD**

Grade (NumGrade, SalBase)

Departement (NumDep, NomDep, Localite)

Employe (Matricule, Nom, DateEmbauche, Poste, Salaire, Commission, #NumDep, #NumGrade)

**1°)** Afficher le pour chaque employé, son matricule, son grade et son salaire.

**2°)** Afficher la liste des employés qui ont un salaire supérieur à 200 000.

3°) Afficher le matricule, le grade et le salaire de tous les employés qui ont un salaire supérieur à 200 000.

**4°)** Afficher le nom et le salaire de tous les employés qui sont des informaticiens.

TD

Soit une base de données « Ventes » composées des relations suivantes :

Client (NumeroCli, NomCli, VilleCli, DateArrivee)

Facture (NumeroFac, DateFac**, #**NumeroCli)

Produit (NumeroPro, NomPro, PuPro, StockPro, SeuilPro)

DetailsFac (IdDeatils, PuFac, QteFac, **#**NumeroFac, #NumeroPro)

QUESTIONS

1°) Afficher le numéro, le nom et le stock des produits dont le seuil est inférieur au stock.

2°) Afficher, pour chaque détail facture, la date de la facture, le nom du produit, la quantité et le prix.

3°) Afficher le numéro et le nom de tous les clients arrives avant le 01-012020.