Base de Données

[Systèmes Informatiques & Logiciels]

Licence 1 & 2

Mr Eric Koffi **AGONDJIHOSSOU**

Tél : **+229 97571544** - E-mail : **aeric@headlines-group.net**

COURS

PROGRAMME

Chapitre I - **GENERALITES SUR LES SGBD**

Chapitre II – **LES BASES DE DONNEES RELATIONNELLES**

Chapitre III – **L'ALGEBRE RELATIONNELLE**

Chapitre IV – **LE LANGAGE SQL**

GENERALITES SUR LES S.G.B.D

A – DEFINITION

1 – Système d'information

Un système d'information (SI) est un ensemble organisé de ressources (matériels, logiciels, personnel, données et procédures) qui permet de collecter, regrouper, classifier, traiter et diffuser de l'information sur un environnement donné.

L'apport des nouvelles technologies de l'Information (NTIC) est à l'origine du regain de la notion de système d´information. L'utilisation combinée de moyens informatiques, électroniques et de procédés de télécommunication permet aujourd'hui -selon les besoins et les intentions exprimés- d'accompagner, d'automatiser et de dématérialiser quasiment toutes les opérations incluses dans les activités ou procédures d'entreprise.

2 – Système informatique

Un système informatique est le système de traitement automatique de l'information. Il se compose de deux parties : la partie matérielle (Hardware) qui est l'ensemble des éléments physiques constituant la machine et la partie logicielle (Software) qui est l'ensemble des logiciels (programmes) de tra

itement de l'information.

3 – Base de Données

Une Base de Données (son abréviation BD, en anglais DB, database) est un ensemble (avec le moins de redondances possibles) structuré d'informations élémentaires enregistrées dans un ordinateur et accessible de façon sélective à une communauté d'utilisateurs.

Cette communauté est constituée des programmeurs et des utilisateurs différents. Cette base de données est gérée par un Système de Gestion de Base de Données. Ainsi, la notion BD est généralement couplée à celle de réseau afin de pouvoir mettre en commun ces informations : d'où le nom de base.

C’est aussi est une collection de données inter-reliées. C'est une entité cohérente logiquement et véhiculant une certaine sémantique. Une BD est faite pour enregistrer des faits, des opérations au sein d'un organisme (administration, banque, université, hôpital, ...). Les BD ont une place essentielle dans l'informatique.

4 – SGBD

Le logiciel qui gère une base de données s'appelle un système de gestion de base de données. On le désigne généralement par son sigle SGBD (DBMS en anglais, pour Data Base Management System). En fait, il devrait s'appeler "logiciel de gestion de base de données" car, en informatique, le mot "système" désigne généralement l'ensemble matériel + logiciel. Mais l'expression SGBD est consacrée par l'usage, et nous n'avons pas d'autre choix que l'adopter.

Tous les SGBD présentent à peu près les mêmes fonctionnalités. Ils se distinguent par leur coût, par le volume de données qu'ils sont capables de gérer, par le nombre d'utilisateurs qui peuvent interroger la base simultanément, par la facilité avec laquelle ils s'interfacent avec les autres logiciels d'application utilisés par l'entreprise, etc.

Il existe des bases de données de toutes tailles, depuis les plus modestes (une liste des numéros de téléphone utilisée par une seule personne), jusqu'aux plus grandes (la base des données commerciales d'un magasin à succursales multiples, contenant des téraoctets de données ou plus, et utilisée par le service marketing).

Un Système de Gestion de Bases de Données (SGBD) ou Data Base Management System (DBMS) en anglais, est un ensemble de programmes qui permettent à des utilisateurs de créer et maintenir une base de données. Les activités supportées sont la définition d'une base de données (spécification des types de données à stocker), la construction d'une base de données (stockage des données proprement dites) et la manipulation des données (principalement ajouter, supprimer, partager, retrouver des données).
Un SGBD sépare la partie description des données, des données elles mêmes. Cette description est stockée dans un dictionnaire de données (également géré dans le SGBD) et peut être consultée par les utilisateurs.

Le nombre d'utilisateurs utilisant une base de données est également extrêmement variable. Une BDD peut servir à une seule personne, laquelle l'utilise sur son poste de travail, ou être à la

disposition de dizaines de milliers d'agents (comme dans les systèmes de réservation des billets d'avion par exemple).

5 – Modèle de données

Un modèle de données est un ensemble de concepts permettant de décrire la structure d'une base de données. La plupart des modèles de données incluent des opérations permettant de mettre à jour et de questionner la base. Le modèle de données le plus utilisé est le modèle relationnel.

6 – Schéma de base de données

Un schéma de base de données (ou compréhension ou intension) est la description des données à gérer. Il est conçu dans la phase de spécification et est peu évolutif (pratiquement statique).

7 – Extension d'une base de données

Une extension d'une base de données correspond aux données de la base à un instant donné. Par définition cet état est dynamique.

B – HISTORIQUE

1 – Avant les SGBD (Années 50-60)

Ecritures des programmes par des programmeurs d'application utilisant le système de gestion de fichiers pour gérer et exploiter les données. Les risques étaient liés au manque de sécurité et la multiplication des efforts (programmes similaires écrits dans différents services pour des besoins proches).

Conséquences :

- **redondances** : fichiers contenant les mêmes données, mais utilisées par des personnes différentes,
- risque d'**incohérences** : du fait des redondances et des mises à jour non centralisées (ex : adresse d'un fournisseur),
- **intégrité** des données : respect de contraintes qui peuvent être programmées (ex : contrôles sur date de naissance, sur code postal, numéro de tél.),

- problèmes liés à la **sécurité** : utilisateurs de différents niveaux d'expérience et avec différents droits d'accès => mots de passe,
- problèmes liés au **partage** des données : accès en lecture / écriture.

2 – Avec les SGBD

* **Années 62-63** : Apparition du concept de base de données.
* **Années 65-70** : SGBD première génération (C'est-à-dire le modèle hiérarchique [arbres] et réseau [graphes]).
- IMS d'IBM (hiérarchique).
- IDS de General Electric (réseau), a servi de modèle pour CODASYL.
* **A partir des années 70** : SGBD seconde génération fondée sur le modèle relationnel. Ces SGBD avaient beaucoup d'avantages du coté de la spécification des moyens d'accès aux données.
* **Années 80:**

MRDS (CII HB), QBE (Query by Example), SQL / IDS (IBM), INGRES, ORACLE

* **De nos jours** :
- Les données sont plus variées et plus riches (textes, sons, images...).
- Bases de connaissances, systèmes experts.
- Bases de données déductives.
- Génie logiciel et SGBD.
- Accès intelligent et naturels (langage naturel par exemple)
- Communication multimédia.

- Principaux systèmes : Oracle, DB2 (IBM), Ingres, Informix, Sybase, SQL Server (Microsoft), O2, Gemstone

- Sur micro : Access, Paradox, FoxPro, 4D, WinDev
- Sharewares : MySQL, MSQL, PostgreSQL, Instant DB

C –FONCTIONNALITES D'UN SGBD IDEAL

Nous verrons ici les caractéristiques souhaitables des SGBD qui ne sont pas forcément prises en compte par les SGBD commerciaux.

1- Contrôler la redondance d'informations

La redondance d'informations pose différents problèmes (coût en temps, coût en volume et risque d'incohérence entre les différentes copies). Un des objectifs des bases de données est de contrôler cette redondance, voire de la supprimer, en offrant une gestion unifiée des informations complétées par différentes vues pour des classes d'utilisateurs différents.

2- Partage des données

Une base de données doit permettre d'accéder à la même information par plusieurs utilisateurs en même temps. Le SGBD doit inclure un mécanisme de contrôle de la concurrence basé sur des techniques de verrouillage des données (pour éviter par exemple qu'on puisse lire une information qu'on est en train de mettre à jour).

Le partage des données se fait également par la notion de vue utilisateur, qui permet de définir pour chaque classe d'utilisateurs la portion de la base de données qui l'intéresse (et dans la forme qui l'intéresse).

3- Gérer les autorisations d'accès

Une base de données étant multi-utilisateurs, se pose le problème de la confidentialité des données. Des droits doivent être gérés sur les données, droits de lecture, mise à jour, création, ... qui permettent d'affiner la notion de vue utilisateur.

4- Offrir des interfaces d'accès multiples

Un SGBD doit offrir plusieurs interfaces d'accès, correspondant aux différents types d'utilisateurs pouvant s'adresser à lui. On trouve des interfaces orientées utilisateur final (langages de requêtes déclaratifs comme SQL avec mise en œuvre graphique, interface de type formulaire, ...) ou bien orientées programmeurs d'applications (interface avec des langages de programmation classiques comme par exemple l'approche SQL immergé ou "Embedded SQL").

5- Représenter des relations complexes entre les données

Un SGBD doit permettre de représenter des données inter-reliées de manière complexe. Cette facilité s'exprime à travers le modèle de données sous-jacent au SGBD. Chaque modèle de données offre ses propres concepts pour représenter les relations. On peut citer les modèles hiérarchique, réseau (première génération de modèles), relationnel (génération actuelle), sémantiques (ou orientés vers la conception tel qu'Entité-Association, Z, ...) ou orienté-objet (la génération future ?).

6- Vérifier les contraintes d'intégrité

Un schéma de base de données se compose d'une description des données et de leurs relations ainsi que d'un ensemble de contraintes d'intégrité. Une contrainte d'intégrité est une propriété de l'application à modéliser qui renforce la connaissance que l'on en a. On peut classifier les contraintes d'intégrité, en contraintes structurelles (un employé a un chef et un seul par exemple) et contraintes dynamiques (un salaire ne peut diminuer). Les SGBD commerciaux supportent automatiquement un certain nombre de contraintes structurelles, mais ne prennent pas en compte les contraintes dynamiques (elles doivent être codées dans les programmes d'application).

7- Assurer la sécurité et la reprise après panne

Une base de données est souvent vitale dans le fonctionnement d'une organisation, et il n'est pas tolérable qu'une panne puisse remettre en cause son fonctionnement de manière durable. Les SGBD fournissent des mécanismes pour assurer cette sécurité. Le premier mécanisme est celui de la transaction qui permet d'assurer un comportement atomique à une séquence d'actions (elle s'effectue complètement avec succès ou elle est annulée). Une transaction est une séquence d'opérations qui fait passer la base de données d'un état cohérent à un nouvel état cohérent. L'exemple typique est celui du débit-crédit pour la gestion d'une carte bancaire. Ce mécanisme permet de s'affranchir des petites pannes (style coupure de courant).

En ce qui concerne les risques liés aux pannes disques, les SGBD s'appuient sur un mécanisme de journalisation qui permet de régénérer une base de données automatiquement à partir d'une version de sauvegarde et du journal des mouvements.

D – LES DIFFERENTS MODELES D'UNE BASE DE DONNEES

Le principal objectif des bases de données est de rendre indépendant les données vis-à-vis des applications. L'accès à ces données par des applications pose des problèmes. Ainsi, pour simplifier des problèmes d'accès, plusieurs modèles logiques de base de données et de systèmes de gestion de base de données ont vu le jour.

L'organisation des données au sein d'une BD a une importance essentielle pour faciliter l'accès et la mise à jour des données :

- ***Les modèles hiérarchiques et réseau sont issus du modèle graphe :***

  . Données organisées sous forme de graphe ;

  . Langages d'accès navigationnels (adressages des liens de chaînages) ;

  . On les appelle « modèle d'accès »

- ***Le modèle relationnel est fondé sur la notion mathématique de relation :***

  . Introduit par Codd (Recherche IBM) ;

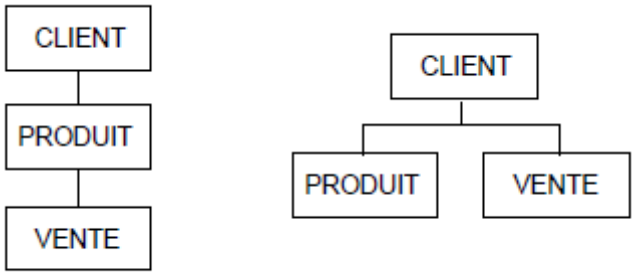
  . Données organisées en tables (adressages relatif) ;

  . Stratégie d'accès déterminée par le SGBD

1- Le modèle hiérarchique

- Schéma logique représenté par un arbre
  - . Nœuds : segment (regroupement de données)
  - . Arc : lien hiérarchique 1 : N

- Exemple de schéma hiérarchique

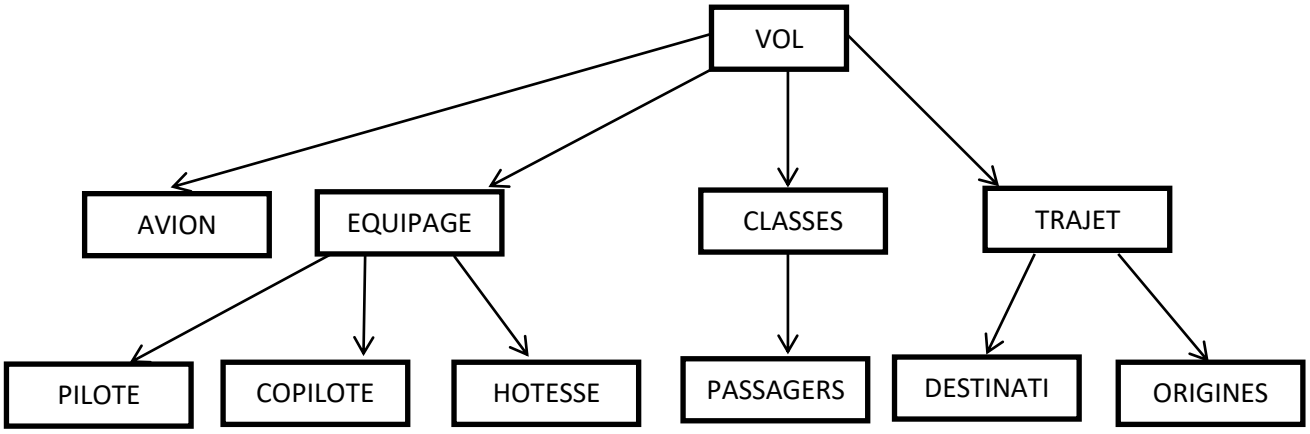


- Choix possible entre plusieurs arborescences (le segment racine est choisi en fonction de l'accès souhaité)

- Dissymétrie de traitement pour des requêtes symétriques. En prenant l'exemple précédent, considérer les 2 requêtes : Trouver les numéros de produits achetés par le client x et trouver les no de clients qui ont acheté le produit p.
Elles sont traitées différemment suivant le choix du segment racine (Client ou Produit)

- Adéquation du modèle pour décrire des organisations à structure arborescente (ce qui est fréquent en gestion)

Historiquement le premier, il consiste à organiser les données de façon arborescente ; ce qui constitue une structure simple à gérer. Cette structure est hiérarchique où chaque élément n'a qu'un supérieur. Le nombre de connexion est limité : il n'y a pas de relations entre les branches de même niveau.



Ce modèle ne permet que des interrogations simples. Par exemple, quel est le trajet du vol 512 ou quel est le pilote du vol 304. Mais il n'est pas aisé de savoir sur quel vol est inscrit le passager Grimaud.

2- Le modèle réseau

- Schéma logique représenté par un graphe
    . Nœuds : article (représente une entité)
    . Arc : lien hiérarchique 1 : N

- Exemple de schéma réseau

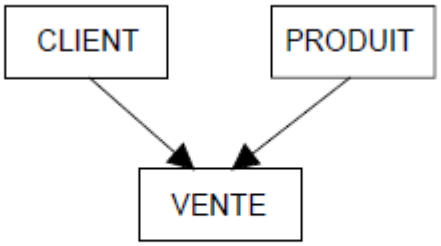
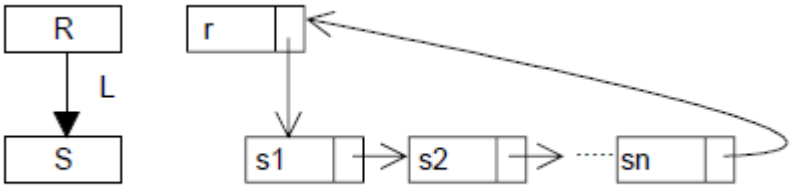


Diagramme de Bachman

- Langage navigationnel pour manipuler les données

- Implémentation d'un lien par une liste circulaire



Le modèle réseau est une extention du modèle précédent : il permet d'établir des connexions entre les différents éléments. De cette manière, on dispose d'un plus grand nombre d'interrogations possibles mais, elles doivent être toujours prévues lors de la constructions de la base de données.

3- Le modèle relationnel

- En 1970, CODD présente le modèle relationnel
- Schéma logique représenté par des relations
- Le schéma relationnel est l'ensemble des relations qui modélisent le monde réel
- Les relations représentent les entités du monde réel (comme des personnes, des objets, etc.) ou les associations entre ces entités
- Exemple de modèle relationnel. :

CLIENT (IdCli, nom, ville)

PRODUIT (IdPro, nompro, prix, qstock)

VENTE (#IdCli, #IdPro, date, qte)

- Représentation des données sous forme de tables

CLIENT

IdCli	Nom	Ville
X	Smith	Paris
Y	Jones	Paris
Z	Blake	Nice

PRODUIT

IdPro	Nom	Prix	Qstock
P	Auto	100	10
Q	Moto	100	10
R	Velo	100	10
S	Pedalo	100	10

VENTE

IdCli	IdPro	Date	Qte
X	P		1
X	Q		2
X	R		3
Y	P		4
Y	Q		5
Z	Q		6

- Avantages du modèle relationnel
  - simplicité de présentation : représentation sous forme de tables
  - opérations relationnelles : algèbre relationnelle et langages assertionnels
  - indépendance physique : optimisation des accès et stratégie d'accès déterminée par le système
  - indépendance logique : concept de vues
  - maintien de l'intégrité : contraintes d'intégrité définies au niveau du schéma

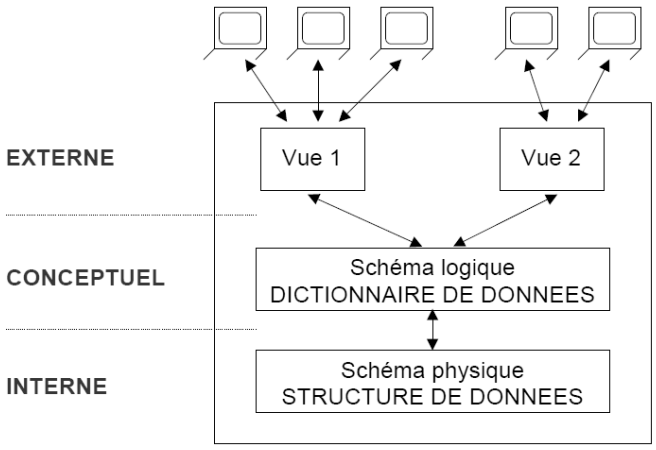
Le modèle relationnel permet de se libérer de la contrainte suivante : Connaître à l'avance les interrogations que l'on effectuera. Ainsi, les données sont stockées sous la forme de la relation dans

des tables. Ce type de structure permet d'établir des connexions au moment de l'exécution. On pourra donc effectuer toutes sortes d'interrogations plus ou moins complexes.
L'accès aux bases de données relationnelles s'effectue en appliquant les trois opérations de base suivante : la projection, la sélection et la jointure.
Avec le modèle relationnel, il est nécessaire de convertir les données d'applications sous forme de tables. L'interface entre une application et une base de données est effectuée à l'aide de langages spécifiques, dont le plus connu est le SQL.

E – ARCHITECTURE LOGIQUE D'UN SGBD

La plupart des SGBD suivent l'architecture standard ANSI/SPARC qui permet d'isoler les différents niveaux d'abstraction nécessaires pour un SGBD.

1 – Architecture ANSI/SPARC *(Standard Planning And Requirement Comitte)*



SGBD
Niveaux de représentation des données

Elle est définie sur trois niveaux :

- **Niveau interne ou physique** : décrit le modèle de stockage des données et les fonctions d'accès. Il correspond aux structures de stockage et aux moyens d'accès (index).

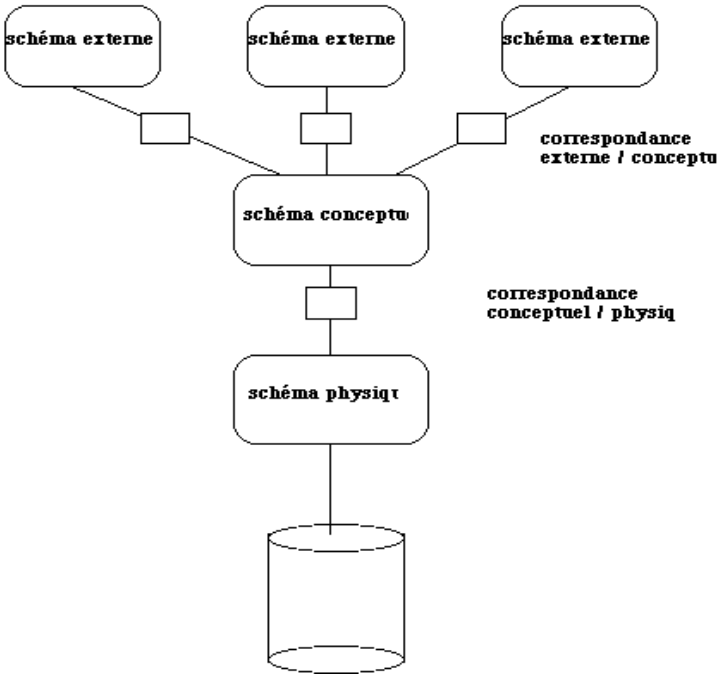
- **Niveau conceptuel ou logique** : décrit la structure de la base de données globalement à tous les utilisateurs (limite la redondance).

Le schéma conceptuel est produit par une analyse de l'application à modéliser et par intégration des différentes vues utilisateurs. Ce schéma décrit la structure de la base indépendamment de son implantation. Il contient donc la description des données et des contraintes d'intégrité (Dictionnaire de Données). Le schéma logique découle d'une activité de modélisation.

- N**iveau externe** : correspond aux différentes vues des utilisateurs.

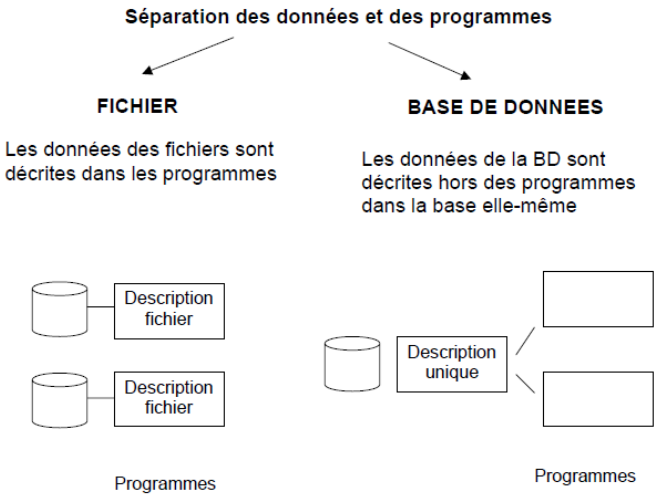
Chaque schéma externe donne une vue sur le schéma conceptuel à une classe d'utilisateurs. Le concept de vue permet d'obtenir l'indépendance logique. La modification du schéma logique n'entraîne pas la modification des applications (une modification des vues est cependant nécessaire) Chaque vue correspond à la perception d'une partie des données, mais aussi des données qui peuvent être synthétisées à partir des informations représentées dans la BD (par ex. statistiques).

Le SGBD doit être capable de faire des transformations entre chaque niveau, de manière à transformer une requête exprimée en termes de niveau externe en requête du niveau conceptuel puis du niveau physique.



La plupart des SGBD ne séparent pas complètement ces trois niveaux, mais respectent néanmoins ces principes de séparation.

2 – Indépendance données – programmes



L'architecture à trois niveaux permet de supporter le concept d'indépendance données - programmes, c'est à dire la capacité de modifier le schéma de la base de données à un niveau donné, sans remettre en cause le schéma aux niveaux supérieurs :

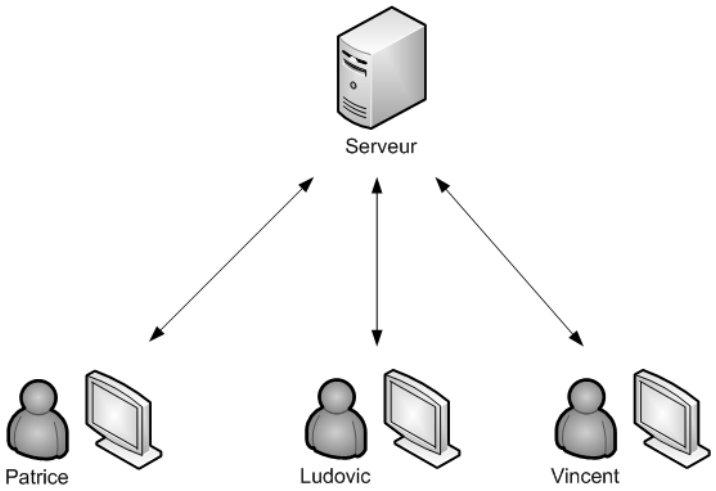
- **indépendance logique** : on peut changer le niveau conceptuel sans remettre en cause les schémas externes ou les programmes d'application. L'ajout ou le retrait de nouveaux concepts ne doit pas modifier des éléments qui n'y font pas explicitement référence,

- **indépendance physique** : on peut changer le schéma physique sans remettre en cause le schéma conceptuel (et les schémas externes). On peut modifier l'organisation physique des fichiers, rajouter ou supprimer des méthodes d'accès.

Le modèle relationnel, contrairement à ces prédécesseurs, permet un certain niveau d'indépendance sans aller jusqu'à une indépendance complète.

F – ARCHITECTURE OPERATIONNELLE D'UN SGBD

1 – L'architecture Client Serveur



L'architecture client/serveur possède deux types d'ordinateurs sur un réseau : les clients et les serveurs. Elle possède donc deux niveaux et s'appelle *two-tiers* en anglais tandis que les architectures multi-tiers (ou distribuées) scindent le serveur en plusieurs entités (par exemple, un serveur d'application qui lui-même est le client d'un serveur de base de données).

C'est une architecture hiérarchisée mettant en jeu d'une part un serveur de données gérant les données partagées en exécutant le code SGBD avec d'éventuelles procédures applicatives, d'autres part des clients pouvant être organisés en différents niveaux supportant les applications et la présentation, et dans laquelle les clients dialoguent avec les serveurs via un réseau en utilisant des requêtes de type question-réponse.

a – Avantages

- Toutes les données sont centralisées sur un seul serveur, ce qui simplifie les contrôles de sécurité et la mise à jour des données et des logiciels.
- Les technologies supportant l'architecture client/serveur sont plus matures que les autres.

- Une administration au niveau serveur, les clients ayant peu d'importance dans ce modèle, ils ont moins besoin d'être administrés
- Toute la complexité/puissance peut être déportée sur le serveur(s), les utilisateurs utilisant simplement un client léger sur un ordinateur terminal qui peut être simplifié au maximum.
- Recherche d'information : Les serveurs étant centralisés, cette architecture est particulièrement adaptée et véloce pour retrouver et comparer de vaste quantité d'information (Moteur de Recherche sur le Web), ce qui semble être rédhibitoire pour le P2P beaucoup plus lent.

b – Inconvénients

- Si trop de clients veulent communiquer avec le serveur au même moment, ce dernier risque de ne pas supporter la charge (alors que le réseau pair à pair fonctionne mieux en ajoutant de nouveaux participants).
- Si le serveur n'est plus disponible, plus aucun des clients ne fonctionne (le réseau pair à pair continue à fonctionner, même si plusieurs participants quittent le réseau).
- Les coûts de mise en place et de maintenance sont élevés.
- En aucun cas les clients ne peuvent communiquer entre eux, entrainant une asymétrie de l'information au profit des serveurs.

c – Exemples

- La consultation de pages sur un site web fonctionne sur une architecture client/serveur. Un internaute connecté au réseau via son ordinateur et un navigateur web est le client, le serveur est constitué par le ou les ordinateurs contenant les applications qui délivrent les pages demandées. Dans ce cas, c'est le protocole de communication HTTP qui est utilisé.

- Les courriels sont envoyés et reçus par des clients et gérés par un serveur de messagerie. Les protocoles utilisés sont le SMTP, et le POP ou l'IMAP.

- La gestion d'une base de données centralisée sur un serveur peut se faire à partir de plusieurs postes clients qui permettent de visualiser et saisir des données.

- Le système X Window fonctionne sur une architecture client/serveur. En général le processus client (une application graphique, xeyes par exemple) tourne sur la même

machine que le serveur mais peut être aussi bien lancé sur un autre ordinateur faisant partie du réseau.

- Un client léger est un ordinateur léger ou ancien s'appuyant sur un serveur central qui héberge et exécute toutes les applications.

2 – L'architecture BD répartie

C'est une architecture composée de plusieurs serveurs coopérant à la gestion de bases de données composées de plusieurs sous-bases gérées par un seul serveur, mais apparaissant comme des bases uniques centralisées pour l'utilisateur. Une base de données répartie (distribuée) est une base de données logique dont les données sont distribuées sur plusieurs SGBD et visibles comme un tout. Les données sont échangées par Messages. Si les données sont dupliquées, on parle plutôt de BD répliquée.

a - Principe fondamental

- Autonomie locale : La BD locale est complète et autonome (intégrité, sécurité, gestion), elle peut évoluer indépendamment des autres (upgrades…) ;
- Egalité entre sites : Un site en panne ne doit pas empêcher le fonctionnement des autres sites (mais perturbations possibles) ;
- Fonctionnement continu : La distribution permet résistance aux fautes et aux pannes (en théorie) ;
- Localisation transparente : Accès uniforme aux données quel que soit leur site de stockage
- Fragmentation transparente : Des données (d'une même table) éparpillées doivent être vues comme un tout ;
- Indépendance à la réplication : Les données répliquées doivent être maintenues en cohérence (délai possible) ;
- Requêtes distribuées : L'exécution d'une requête peut être répartie (automatiquement) entre plusieurs sites (si les données sont réparties) ;
- Transactions réparties : Le mécanisme de transactions peut être réparti entre plusieurs sites ;

- Indépendance vis-à-vis du matériel : Le SGBD fonctionne sur les différentes plateformes utilisées ;
- Indépendance vis-à-vis du SE : Le SGBD fonctionne sur les différents SE… ;
- Indépendance vis-à-vis du réseau : Le SGBD est accessible à travers les différents types de réseau utilisés ;
- Indépendance vis-à-vis du SGBD : La base peut être distribuée sur des SGBD hétérogènes ;

b - Conception de BD répartie

- On met en place une BD répartie qu'en cas de réel besoin

* Démarche de conception délicate
* Gestion complexe
* L'évolution du SI peut invalider la solution retenue…

- Des raisons valables :

* Volumes de données, sites distants, etc.
* Fusions de SI

G – LES UTILISATEURS DE LA BASE DE DONNEES

1 – L'administrateur

C'est le responsable de la Base de Données et plus précisément de sa création et de sa maintenance. Il dispose d'un ensemble d'outils logiciels qui permettent de l'assister dans sa tâche. On distingue :

* l'administrateur principal qui définit le schéma conceptuel, conditionne l'évolution de la base et définit les modalités d'accès et de protection des données,

* l'administrateur d'application qui définit le sous modèle adapté à l'application, élabore les schémas externes (tables qui seront accessibles aux utilisateurs), définit des règles de correspondance entre schéma externe et schéma interne.

Remarque : C'est à partir des travaux des administrateurs que va s'élaborer le **dictionnaire des données.** Il correspond à la mémoire conservée de tous les objets ayant été créés. Ce dictionnaire permet de faire des statistiques d'après coups et ne sera jamais effacé.

2– Le programmeur d'applications

Il réalise des bibliothèques de programmes pour la manipulation et le traitement des données (Interrogation, mise à jour, ...). On utilise pour cela un Langage de Manipulation de Données (LMD).

Exemple : langage SQL.

3– L'utilisateur final

C'est une personne accédant à la Base de Données, à l'aide d'un terminal ou un ensemble de programmes d'applications préconçues. L'accès à la Base de Données est toujours réalisé à travers un sous schéma qui décrit la manière dont les données sont vues par l'utilisateur.

LES BASES DE DONNEES RELATIONNELLES

A – VUE SUR LE MODELE ENTITE-ASSOCIATION

1- La phrase élémentaire

a- Définition

Une phrase élémentaire est une phrase sémantiquement irréductible et non ambiguë. C'est aussi une relation binaire entre un sujet et un complément.

b- Principes

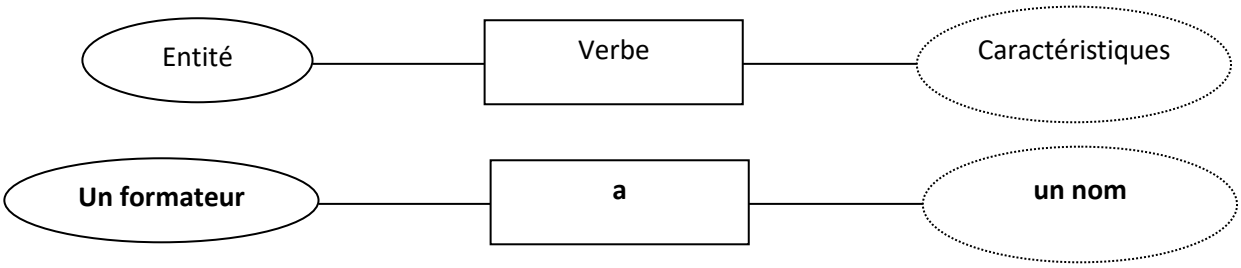
b1) On ne conserve qu'un verbe par phrase.

b2) Le sujet identifie toujours une ENTITE (un ACTEUR ou un OBJET) du système

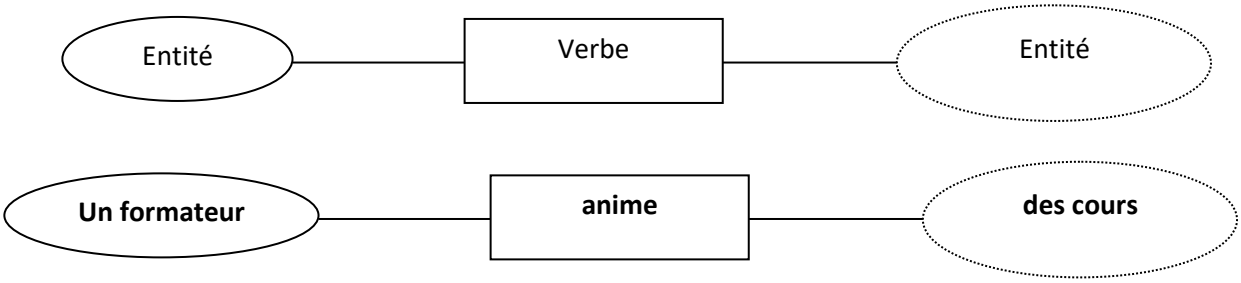
b3) Le complément est soit une caractéristique de l'entité, soit une entité.

c- Description d'une entité et relation entre deux entités

- Un formateur a un nom



- Un formateur anime des cours



2 – Entité

Une entité est la représentation d'un objet concret ou abstrait. Une entité exprime un type ou une classe d'objets dont l'existence est reconnue par le système d'information en cours d'étude. Les objets d'un type exprimé par une entité sont tous décrits par les mêmes caractéristiques appelées propriétés.

3 - Association

Une association traduit un rapport entre entités. Une association est la prise en charge par le système d'information de l'existence d'un lien entre des objets de certaines entités. Une association peut aussi être caractérisée par des propriétés (comme une entité).

4 – Propriété

Une propriété est la qualité propre, le caractère fonctionnel qui appartient à tous les objets d'une espèce, d'un type d'objet. Une propriété prend toujours une valeur. L'ensemble des valeurs prises par toutes les propriétés qui qualifient une entité représente une occurrence de cette entité.

5 – Identifiant

L'identifiant d'une entité est une de ses propriétés à laquelle il est alloué une valeur propre par objet particulier de l'entité. L'identifiant permet de déterminer de manière unique chaque objet occurrence de l'entité. L'identifiant a une existence obligatoire par entité exprimée dans le système d'information.

6 – Détermination d'une entité ou d'une association

Une entité ou une association est totalement déterminée par :

- Un nom ;
- Un identifiant ;
- Une liste de propriétés ;

L'association a donc un identifiant qui, par définition est constitué de l'ensemble des identifiants des entités associées.

7 – Type et nature d'une propriété

a-Type

Une propriété peut être de type numérique (propriété quantitative), alphabétique (propriété signalétique), alphanumérique (propriété codée) ou de type date.

b-Nature

Une propriété peut être de nature élémentaire, calculée ou composée.

8 – Dictionnaire des données

Afin d'éviter un certain nombre d'anomalies sur les données retenues, des considérations relatives à la structure et à la nature des propriétés sont à prendre en compte et constituent le vocabulaire de l'entreprise appelé Dictionnaire des données.

Exemple : Un formateur anime des cours

Illustration

Variable	**Signification**	**Type**	**Longueur**	**Nature**	**Remarque**
NumForm	Numéro formateur	N	2	E	Identifiant
NomForm	Nom du formateur	A	20	E	
RefCours	Référence du cours	AN	2	E	Identifiant

9 – Dépendances fonctionnelles

a - Notion

Un attribut (ou un groupe d'attributs) B est dit "fonctionnellement dépendant" d'un attribut (ou d'un groupe d'attributs) A si : a1 = a2 => b1 = b2, a1, a2, b1, b2 étant des réalisations (valeurs) des attributs A et B dans des nuplets de la base de données.

On dit alors que A "détermine" B, et on note A -> B.

Exemple :

Soit le schéma de relation PERSONNE (No_SS, Nom, Adresse, Age, Profession). Les dépendances fonctionnelles qui s'appliquent sur ce schéma de relation sont les suivantes : No_SS -> Nom, No_SS -> Adresse, No_SS -> Age, No_SS -> Profession.

On pourra aussi écrire : No_SS -> Nom Adresse Age Profession.

L'attribut No_SS détermine tous les attributs du schéma de relation. Il s'agit d'une propriété de la clé d'un schéma de relation.

b - Axiomes d'Armstrong et couverture minimale

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 :

b1) **transitivité** : si X -> Y, et Y -> Z, alors X -> Z,

b2) **augmentation** : si X -> Y, alors XZ -> Y pour tout groupe Z d'attributs appartenant au schéma de relation,

b3) **réflexivité** : si X contient Y, alors X -> Y.

A partir de ces trois axiomes de base, on peut déduire d'autres règles :

b4) **union** : si X -> Y et Y -> Z, alors X -> YZ,

b5) **pseudo-transitivité** : si X -> Y et WY -> Z, alors WX -> Z,

b6) **décomposition** : si X -> Y et Z contenu dans Y, alors X -> Z.

c – Graphes des dépendances fonctionnelles

On appelle graphe des dépendances fonctionnelles, un graphe dans lequel les sommets sont des propriétés et les arcs des dépendances fonctionnelles. On le dessine par étapes.

c1) Les identifiants élémentaires sont reliés aux propriétés qu'elles déterminent,

c2) Les identifiants composés sont reliés aux propriétés qu'elles déterminent,

Exemple : Réaliser le dictionnaire des données puis la couverture minimale et dessiner le graphe de dépendance fonctionnelle de la liste des propriétés ci-dessous

- Code produit alimentaire
- Numéro restaurateur
- Numéro commande
- Libellé produit alimentaire
- Date commande
- Nom restaurateur
- Prénom restaurateur
- Quantité commandée
- Date de livraison
- Rue de livraison
- Ville de livraison

10 – Règles de gestion

a-Définition

Une règle de gestion décrit la liaison d'une entité à une autre. Elle exprime au sein d'une association la relation d'une entité à une autre.

b-Exemple

Au sujet du système d'information des produits alimentaires, les règles de gestion pourront être exprimées comme suit :

R1 : Toute commande est passée par un restaurateur et un seul

R2 : Une commande peut concerner plusieurs produits alimentaires

R3 : Un même produit peut faire l'objet de plusieurs commandes

R4 : Une facture concerne au plus une commande

11 – Cardinalités

Les règles de gestions sont traduites par les cardinalités.

Une cardinalité est un entier qui quantifie pour une occurrence donnée d'une entité dans une association, le nombre d'occurrences de l'autre entité (ou des autres entités) qui lui sont reliées.

Dans la pratique, on exprime que la cardinalité maximale et la cardinalité minimale de l'ensemble des cardinalités des occurrences d'une entité. On ne retient que la valeur 0 ou 1 pour la cardinalité minimale et 1 ou n pour la cardinalité maximale. Les 4 combinaisons possibles de couples utilisées sont :

- (0,1) pour au plus un ;
- (0,n) pour plusieurs ;
- (1,1) pour au moins un, au plus un ;
- (1,n) pour au moins un et plusieurs.

Les cardinalités sont portées sur le MCD

12 – Le Modèle Conceptuel des Données (MCD)

C'est un schéma de synthèse des résultats d'analyse de données d'un système d'information.

Exemple : Présentons maintenant le MCD du système d'information des produits alimentaires.

B – LES STRUCTURES DE DONNEES DE BASE DU MODELE RELATIONNEL

1 – Domaine

Un domaine est un ensemble de valeurs caractérisées par un nom. Les domaines sont les ensembles dans lesquels les propriétés prennent valeur.

Autrement dit, un domaine D est un ensemble de valeurs atomiques. Le terme atomique signifie que ces valeurs sont considérées comme insécables au niveau du modèle.

Comme ensemble, un domaine peut être défini en extension et en intention. Les domaines de base : ENTIER, REEL, CARACTERE sont définis en intention.

Exemples

Domaine des noms : N = {Jean, Paul, Pierre, ...} = D1

Domaine des villes : V = {Paris, Nice, Avignon, ...} = D2

2 - Relation

Une relation est un sous ensemble du produit cartésien d'une liste de domaines caractérisé par un nom. Représentée la relation sous forme de table, chaque ligne ou vecteur correspond à un tuple et chaque colonne d'en-tête nommé attribut correspond à un domaine.

$R \subset D1 \times D2 \times D3 \times ... \times Dn$

3 - Attribut

Un attribut est une colonne d'une relation caractérisée par un nom.

La table ci-dessus définit la relation en ***extension*** (ensemble de tous les cas possibles à un instant donné).

Exemples :

Attribut ↓

HABITE	D1	D2
	Paul	Nice
	Martine	Nice
	Pierre	Paris
Tuple →	Jacques	Avignon

RELIE	D2	D2
	Avignon	Nice

La relation est définie en ***intention*** au moyen d'un schéma qui contient le nom de la relation et la liste des attributs.

Exemple : HABITE (NOM, VILLE)

4 – Table

L'ensemble des occurrences d'une entité (ou d'une association) constitue sa table.

C'est une notion dynamique (c'est – à dire liée à la mémoire vive). Son correspondant physique s'appelle fichier. La table est visualisée sur l'écran dans un objet graphique.

5 – Base de Données relationnelle

Une base de données relationnelle est un ensemble de schémas de relations.

C – CONCEPTION DE SCHEMA RELATIONNEL

Introduit par E.F. Codd en 1970, il a été le premier système commercialisé à la fin des années 70, système efficace au début des années 80. C'est un modèle fondé sur la **théorie des ensembles** et la notion de **relation** (bases mathématiques).

Une relation se définit ici comme : R (A1 : D1, A2 : D2, … An : Dn), avec :

R = nom de la relation, Ai = noms des attributs, Di = domaines de définition des attributs (valeurs possibles) , n = cardinalité de la relation

Exemple : PERSONNE (Nom : char (20), Prénom : char (20), Age : integer, Adresse :
Varchar (50), CP : integer, Ville : char (20))

Ce qui est écrit ci-dessus constitue en fait le **schéma de la relation** PERSONNE. La **relation** PERSONNE est représentée sous la forme d'une table :

Nom	Prénom	Age	Adresse	CP	Ville
Dupont	Pierre	50	7, rue du Port	17000	La Rochelle
Martin	Alain	33	4, place de la Gare	87000	Limoges

Une ligne de la table constitue un élément de la relation, ou **n-uplet**. Elle représente aussi une personne, instance de la relation PERSONNE. D'un point de vue logique (mathématique), il s'agit d'un prédicat qui met en liaison les attributs de la relation. Il n'y a pas d'ordre sur les lignes (ni sur les colonnes) dans une table / relation. Il n'y a pas non plus d'information sur l'organisation physique (stockage des données) qui est de ce fait cachée à l'utilisateur.

D - DU MODELE ENTITE-ASSOCIATION AU MODELE RELATIONNEL

· Une entité se traduit par une relation, qui contient tous les attributs de l'entité
· Une association se traduit par une relation, qui contient tous les attributs de l'entité ainsi que les attributs de l'association

Exemple : schéma entité-association « Cours / Etudiants / Profs » (Réaliser le schéma)

Il se traduit par le **schéma relationnel de BD** (ensemble de schémas de relation) suivant :
ETUDIANT (Num_Etudiant **integer**, Nom **char (20)**, Adresse **varchar (50)**)
COURS (Num_Cours **integer**, Nom **char (20)**)
PROFS (Num_Prof **integer**, Nom **char (20)**, Adresse **varchar (50)**)
COURS_SUIVIS (Num_Etudiant **integer**, Num_Cours **integer**)
COURS_ENSEIGNES (Num_Prof **integer**, Num_Cours **integer**)

E – LA NORMALISATION

L'objectif de la normalisation est de construire un schéma de base de données cohérent et possédant certaines propriétés vérifiées par la satisfaction de formes normales. Pour une application spécifique, il est en effet possible de proposer plusieurs schémas. Les questions qui se posent alors sont les suivantes :

a) qu'est-ce qu'un bon schéma ?

b) quel schéma choisir ?

Un mauvais schéma défini lors de la phase de conception peut conduire à un certain nombre d'anomalies pendant la phase d'exploitation de la base :

- des redondances d'information,
- des anomalies lors des opérations de mise à jour (insertions, suppressions, modifications).

Exemple :

Soit le schéma de relation FOURNISSEUR (Nom_Fournisseur, Adresse, Produit, Prix).

Une relation (table) correspondant à ce schéma pourra éventuellement contenir plusieurs produits pour un même fournisseur. Dans ce cas, l'adresse du fournisseur sera dupliquée dans chaque n-uplet (redondance). Si on souhaite modifier l'adresse d'un fournisseur, il faudra rechercher et mettre à jour tous les n-uplets correspondant à ce fournisseur. Si on insère un nouveau produit pour un fournisseur déjà référencé, il faudra vérifier que l'adresse est identique. Si on veut supprimer un fournisseur, il faudra retrouver et supprimer tous les n-uplets correspondant à ce fournisseur (pour différents produits) dans la table. Ces anomalies n'apparaitront pas si on décompose le schéma initial de base de données. Par contre, la décomposition d'un schéma relationnel au cours de la normalisation risque d'entrainer une dégradation des performances, du fait des opérations de jointure nécessaires.

Les 3 premières formes normales ont été proposées par E.F. Codd ("inventeur" du modèle relationnel) en 1972. La forme normale dite de Boyce-Codd a été proposée en 1974. Les 4ème (1977) et 5ème (1979) formes normales ont été proposées ensuite par Fagin, mais elles ne concernent que des cas rares et très spécifiques. Les formes normales s'appuient sur les dépendances fonctionnelles entre attributs d'un schéma de base de données.

1 - 1ere Forme Normale (FN1)

Une relation est en 1ère forme normale si elle ne contient que des "valeurs atomiques", c'est-à-dire pas de "groupes répétitifs".

Exemple

La valeur "Jacques" affectée à l'attribut Prénom

Contre-exemple : la valeur "Jacques, Paul" affectée à l'attribut Prénom, à moins qu'il ne s'agisse d'un prénom composé. De la même façon, on ne pourra pas mettre dans un même attribut Parent tous les enfants d'une personne. Il faudra, soit prévoir autant de colonnes que de nombre d'enfants possibles, soit insérer dans la base autant de n-uplets que d'enfants en répétant à chaque fois le nom du parent.

Remarque

Cette restriction est très contraignante pour certaines classes d'applications telles que la conception assistée par ordinateur (CAO) ou les systèmes d'information géographique (SIG).

2 - 2eme Forme Normale (FN2)

La 2ème forme normale s'appuie sur la notion de DF "**complète**" (ou "pleine"). Une DF X -> Y est complète si, X et Y étant attributs d'une même relation, Y n'est pas fonctionnellement dépendant d'un sous-ensemble de X.

Exemple

Si AB -> C sur R (A, B, C), on ne peut avoir ni A -> C ni B -> C.

Dans le cas contraire, c'est-à-dire si on peut enlever un attribut de X, et que la DF est toujours applicable, on dit qu'elle est "**partielle**".

Définition

Une relation est en 2ème forme normale si elle est déjà en 1ère forme normale, et si tout attribut n’appartenant pas à la clé (primaire) dépend complètement de cette clé, c’est-à-dire si toutes les DF s’appliquant sur R sont complètes.

Remarque

Si toutes les DF sont des DF "simples", avec un seul attribut en partie gauche, ou si les clés sont atomiques, alors la relation est FN2.

3 - 3eme Forme Normale (FN3)

En décomposant R en R1 et R2, on élimine des risques d'erreurs, mais on peut avoir d'autres types d'erreurs, lors d'opérations de mise à jour, du fait des DF "**transitives**".

Définition

Une DF X -> Z est transitive lorsqu'on a X -> Y et Y-> Z (application de la transitivité avec les axiomes d'Armstrong).

Une relation est en troisième forme normale si elle satisfait FN1 et FN2, et si aucun attribut n'appartenant pas à la clé (primaire) ne dépend de la clé par des DF transitives, c'est-à-dire si aucune DF transitive ne s'applique sur cette relation.

Pour rendre la relation FN3, il faut donc éliminer les DF transitives en plaçant certains attributs dans une autre relation.

Autre définition

Une relation est FN3 si, pour toute DF X -> A s'appliquant sur R avec A non inclus dans X, soit X est clé de R, soit A fait partie d'une clé de R.

Exemple

Soit le schéma de relation **R** (Nom_F, Adresse_F, Produit, Prix), avec comme ensemble de DF **F** = {Nom_F -> Adresse_F ; Nom_F Produit -> Prix}.

La clé de **R** est [Nom_F Produit]. Pour Nom_F -> Adresse_F, Nom_F n'est pas clé, et Adresse_F ne fait pas partie de la clé. La relation n'est donc pas FN3.

4 - Forme Normale de BOYCE-CODD (FNBC)

Avec FN3, les DF partielles et transitives ont été éliminées pour les clés primaires, mais il faut également considérer les autres clés possibles (clés "candidates") si elles existent.

Remarque

Si la relation ne contient qu'une clé et qu'elle est FN3, alors elle est aussi FNBC.

Définition

Une relation est FNBC si elle est FN1, FN2 et FN3, et si toutes les parties gauches des DF sont clés candidates pour la relation.

Autre définition

Une relation est FNBC si, pour toute DF X -> A s'appliquant sur R avec A non inclus dans X, X est clé (primaire ou candidate) de R.

Exemple

Soit le schéma de relation **R** (Nom_F, Adresse_F, Produit, Prix), avec comme ensemble de DF

F = {Nom_F -> Adresse_F ; Nom_F Produit -> Prix}.

La clé de **R** est [Nom_F Produit]. Pour Nom_F -> Adresse_F, Nom_F n'est pas clé, et la relation n'est donc pas FNBC.

5 - Algorithme de décomposition FNBC (SPI)

Soit un schéma de relation **R**, et une couverture minimale **F'** de DF s'appliquant sur **R**. Une décomposition SPI de **R**, qu'on appellera **D**, est construite de manière itérative :

a) **D** est initialisée à **R**,

b) Soit **T** une relation de **D** qui ne soit pas FNBC. Cela signifie qu'il y a une DF

X ->A s'appliquant sur **T** telle que X n'est pas clé de **T**, et que A n'est pas un sous ensemble de X.

On décompose alors **T** en **T1** contenant A et les attributs de X, et **T2** contenant tous les attributs de **T** sauf A.

c) **D** est réinitialisée avec **T1** et **T2**, et on boucle sur b) jusqu'à ce que toutes les relations soient FNBC.

6 - Algorithme de décomposition fn3 (SPI et préservant les DF)

Soit un schéma de relation **R**, et une couverture minimale **F'** de DF s'appliquant sur **R**. Une décomposition SPI et SPD de **R**, qu'on appellera **D**, est construite de la façon suivante :

a) pour chaque DF X -> A, on crée une relation Ri (X, A),

b) si on a plusieurs DF telles que X -> A1, X -> A2, …, X -> An, alors on regroupe tous ces attributs dans une même relation Rj (X, A1, A2, …, An),

c) les attributs n'apparaissant dans aucune relation (pas présents dans les DF) sont regroupés dans une même relation,

d) pour avoir une décomposition SPI, il faut s'assurer qu'il y ait au-moins une clé de R dans une des relations de décomposition. Si ce n'est pas le cas, il faut soit ajouter une relation contenant une clé de R, soit ajouter des attributs dans une des relations de décomposition afin de satisfaire cette contrainte.

ALGEBRE RELATIONNELLE

(Le langage algébrique)

Préambule

Un Langage de Manipulation des Données (LMD) est constitué de commandes qui permettent :

- L'interrogation de la base de données.
- La modification de la base (insertion, mise à jour, ...).
- La programmation à partir d'un langage Hôte.

Il existe trois types de langages :

- Langages fondés sur l'algèbre relationnelle (SQL).
- Langages fondés sur le calcul des prédicats et le calcul de tuples (QUEL).
- Langages fondés sur le calcul des prédicats et le calcul relationnel de domaines (DRC).

Introduction

Le premier langage étudié dans ce cours est *l'algèbre relationnelle*. Il consiste en un ensemble d'opérations qui permettent de manipuler des *relations*, considérées comme des ensembles de tuples : on peut ainsi faire *l'union* ou la *différence* de deux relations, *sélectionner* une partie de la relation, effectuer des *produits cartésiens* ou des *projections*, etc. Une propriété fondamentale de chaque opération est qu'elle prend une ou deux relations en entrée, et produit une relation en sortie. Cette propriété permet de *composer* des opérations : on peut appliquer une sélection au résultat d'un produit cartésien, puis une projection au résultat de la sélection et ainsi de suite.

En fait on peut construire des *expressions algébriques* arbitrairement complexes qui permettent d'exprimer des manipulations sur un grand nombre de relations.
Une *requête* est une expression algébrique qui s'applique à un ensemble de relations (la base de données) et produit une relation finale (le résultat de la requête). On peut voir l'algèbre relationnelle comme un langage de programmation très simple qui permet d'exprimer des requêtes sur une base de données relationnelle.

A – LES OPERATIONS DE L'ALGEBRE RELATIONNELLE

L'algèbre se compose d'un ensemble d'opérateurs, parmi lesquels 5 sont nécessaires et suffisants et permettent de définir les autres par composition.
Ce sont : la sélection, la projection, le produit cartésien, l'union et la différence.
Les deux premiers sont des opérateurs *unaires* (ils prennent en entrée une seule relation) et les autres sont des opérateurs *binaires*. A partir de ces opérateurs il est possible d'en définir d'autres, et notamment la *jointure*, qui est la composition d'un produit cartésien et d'une sélection.
Ces opérateurs sont maintenant présentés tour à tour.

1 – La sélection (Restriction)

La sélection (restriction) R' s'applique à une relation R, et extrait de cette relation les tuples qui satisfont un critère de sélection. Ce critère peut être :
– La comparaison entre un attribut A de la relation R et une constante a. Cette comparaison s'écrit A (-) a, où (-) appartient à {=, ≤, ≥, ≠, <, >}
– La comparaison entre deux attributs A1 et A2, qui s'écrit A1(-) A2 avec les mêmes opérateurs de comparaison que précédemment.
– La notation est : R'=Selection (R, A (-) a) avec R le nom de la table ou de la relation.

Exemple :

……
……

On obtient donc le résultat :

……
……

La sélection a pour effet de supprimer des lignes, mais chaque ligne garde l'ensemble de ses attributs.

2 – La projection

La projection d'une relation R (A1, A2, ..., An) sur les attributs Ai1, …, Aip ($p \leq n$) est une relation R' (Ai1, Ai2, ..., Aip) dont les tuples sont obtenus par élimination des valeurs de R n'appartenant pas à R' et par suppression des tuples en double. Donc, contrairement à la sélection, on ne supprime pas des lignes mais des colonnes. La notation est : R'=Projection (R, Ai1, Ai2, ..., Aip) avec R le nom de la table ou de la relation.

Exemple :

……
……

On obtient donc le résultat :

……
……

En principe le nombre de lignes dans le résultat est le même que dans la relation initiale. Il y a cependant une petite subtilité : comme le résultat est une relation, il ne peut pas y avoir deux lignes identiques (il n'y a pas deux fois le même élément dans un ensemble). Il peut arriver qu'après une projection, deux lignes qui étaient distinctes initialement se retrouvent identiques, justement parce que l'attribut qui permettait de les distinguer a été supprimé. Dans ce cas on ne conserve qu'une seule des deux (ou plus) lignes identiques.

Exemple :

……

……

On obtient donc le résultat :

……

……

3 – Le produit cartésien

Le premier opérateur binaire, et le plus important, est le produit cartésien. Le produit cartésien entre deux relations R et S se note RxS, et permet de créer une nouvelle relation où chaque tuple de R est associé à chaque tuple de S.

Voici deux relations R et S :

A	B
a	b
x	y

C	D
c	d
u	v
x	y

Et voici le résultat de $R \times S$:

A	B	C	D
a	b	c	d
a	b	u	v
a	b	x	y
x	y	c	d
x	y	u	v
x	y	x	y

Le nombre de lignes dans le résultat est exactement |R|x|S| (|R| et |S| dénotent respectivement le nombre de lignes dans les relations R et S). En lui-même, le produit cartésien ne présente pas un grand intérêt puisqu'il associe aveuglément chaque ligne de R à chaque ligne de S. Il ne prend vraiment son sens qu'associé à l'opération de sélection, ce qui permet d'exprimer des *jointures*, opération fondamentale qui sera détaillée plus loin.

Conflits de noms d'attributs

Quand les schémas des relations R et S sont complètement distincts, il n'y a pas d'ambigüité sur la provenance des colonnes dans le résultat. Par exemple on sait que les valeurs de la colonne A dans RxS viennent de la relation R. Il peut arriver (il arrive de fait très souvent) que les deux relations aient des attributs qui ont le même nom. On doit alors se donner les moyens de distinguer l'origine des colonnes dans la table résultat en donnant un nom distinct à chaque attribut.

Voici par exemple une table T qui a les mêmes noms d'attributs que R. Le schéma du résultat du produit cartésien RxT a pour schéma (A, B, A, B) et présente donc des ambigüités, avec les colonnes A et B en double.

A	B
m	n
o	p

La table T

La première solution pour lever l'ambiguité est de préfixer un attribut par le nom de la table d'où il provient. Le résultat de $R \times T$ devient alors :

R.A	R.B	T.A	T.B
a	b	m	n
a	b	n	p
x	y	m	n
x	y	n	p

Au lieu d'utiliser le nom de la relation en entier, on peut s'autoriser tout synonyme qui permet de lever l'ambigüité.

4 – L'union

Il existe deux autres opérateurs binaires, qui sont à la fois plus simples et moins fréquemment utilisés. Le premier est l'union. L'expression R u S crée une relation comprenant tous les tuples existant dans l'une ou l'autre des relations R et S. Il existe une condition impérative : *les deux relations doivent avoir le même schéma*, c'est-à-dire même nombre d'attributs, mêmes noms et mêmes types. L'union des relations R (A, B) et S (C, D) données en exemple ci-dessus est donc interdite (on ne saurait pas comment nommer les attributs dans le résultat). En revanche, en posant $S' = \rho_{C \to A, D \to B}(S)$, ,il devient possible de calculer R u S'avec le résultat suivant :

A	B
a	b
x	y
c	d
u	v

Comme pour la projection, il faut penser à éviter les doublons. Donc le tuple (x, y) qui existe à la fois dans R et dans S'ne figure qu'une seule fois dans le résultat.

5 – La différence

Comme l'union, la différence s'applique à deux relations qui ont le même schéma.
L'expression R-S' a alors pour résultat tous les tuples de R qui ne sont pas dans S'.
Voici la différence de R et S', les deux relations étant définies comme précédemment.

A	B
a	b

La différence est le seul opérateur qui permet d'exprimer des requêtes comportant une négation (on veut 'rejeter' quelque chose, on 'ne veut pas' des lignes ayant telle propriété). Il s'agit d'une fonctionnalité importante et difficile à manier : elle sera détaillée plus loin.

6 – La jointure

Toutes les requêtes exprimables avec l'algèbre relationnelle peuvent se construire avec les 5 opérateurs présentés ci-dessus. En principe, on pourrait donc s'en contenter. En pratique, il existe d'autres opérations, très couramment utilisées, qui peuvent se construire par composition des opérations de base. La plus importante est la jointure.
C'est une *composition* de deux opérations (un produit cartésien, une sélection) afin de rapprocher des informations réparties dans plusieurs tables, mais ayant des liens entre elles. Cette opération est une *jointure*, que l'on peut directement, et simplement, noter :

Exemple :

……

……

On obtient donc le résultat :

……
……

La jointure consiste donc à rapprocher les lignes de deux relations pour lesquelles les valeurs d'un (ou plusieurs) attributs sont identiques. De fait, dans 90% des cas, ces attributs communs sont la clé primaire d'une des relations et la clé étrangère dans l'autre relation.

Remarque : Si on n'exprime pas de critère de rapprochement, la jointure est équivalente à un produit cartésien.

Il faut être attentif aux ambigüités dans le nommage des attributs qui peut survenir dans la jointure au même titre que dans le produit cartésien. Les solutions à employer sont les mêmes : on préfixe par le nom de la relation ou par un synonyme clair, ou bien on renomme des attributs avant d'effectuer la jointure.

B- EXPRESSION DE REQUETES AVEC L'ALGEBRE

Cette section est consacrée à l'expression de requêtes algébriques complexes impliquant plusieurs operateurs. On utilise la composition des opérations, rendue possible par le fait que tout operateur produit en sortie une relation sur laquelle on peut appliquer à nouveau des operateurs.

1 – Sélection généralisé

Regardons d'abord comment on peut généraliser les critères de sélection. Jusqu'à présent on a vu comment sélectionner des lignes satisfaisant un critère de sélection. Maintenant supposons que l'on veuille jumeler plusieurs critères simultanément. Ce qui revient à pouvoir exprimer une sélection avec une *conjonction* de critères. La requête devient donc :

……
……
……
……

Donc la composition de plusieurs sélections permet d'exprimer une conjonction de critères de recherche. De même la composition de la sélection et de l'union permet d'exprimer la *disjonction*.

Ce qui permet de s'autoriser la syntaxe suivante, où le '± ' dénote le 'ou' logique.

Enfin la *différence* permet d'exprimer la *négation* et "d'éliminer" des lignes. Cette requête est équivalente à une certaine sélection :

En résumé, les opérateurs d'union et de différence permettent de définir une sélection où le critère q est une expression booléenne quelconque. Attention cependant : si toute sélection avec un 'ou' peut s'exprimer par une union, l'inverse n'est pas vrai.

2 – Requête conjonctives

Les requêtes dites *conjonctives* constituent l'essentiel des requêtes courantes. Intuitivement, il s'agit de toutes les recherches qui s'expriment avec des 'et', par opposition à celles qui impliquent des 'ou' ou des 'not'. Dans l'algèbre, ces requêtes sont toutes celles qui peuvent s'écrire avec seulement les trois opérateurs liés à la sélection, la projection et le produit cartésien.

Les plus simples sont celles où on n'utilise que la sélection et la projection.

En voici quelques exemples.

1. ……
   ……
   ……
2. ……
   ……
   ……
3. ……
   ……
   …….....

Des requêtes légèrement plus complexes - et extrêmement utiles - sont celles qui impliquent la jointure (le produit cartésien ne présente d'intérêt que quand il est associé à la sélection). On doit utiliser la jointure dès que les attributs nécessaires pour évaluer une requête sont répartis dans au moins deux tables. Ces "attributs nécessaires" peuvent être :
– Soit des attributs qui figurent dans le résultat ;
– Soit des attributs sur lesquels on exprime un critère de sélection.

En pratique, la grande majorité des opérations de jointure s'effectue sur des attributs qui sont clé primaire dans une relation, et clé secondaire dans l'autre. Il ne s'agit pas d'une règle absolue, mais elle résulte du fait que la jointure permet le plus souvent de reconstituer le lien entre des informations qui sont naturellement associées, mais qui ont été réparties dans plusieurs relations au moment de la modélisation logique de la base. Cette reconstitution s'appuie sur le mécanisme de clé étrangère qui a été étudié dans le chapitre consacré à la conception.

Voici quelques autres exemples qui illustrent cet état de fait :

1. ……
   ……
   ……
2. ……
   ……
   ……
3. ……
   ……
   …….....

3 – Les requêtes avec union (u) et différence (-)

Pour finir, voici quelques exemples de requêtes impliquant les deux opérateurs union et différence. Leur utilisation est moins fréquente, mais elle peut s'avérer absolument nécessaire puisque ni l'un ni l'autre ne peuvent s'exprimer à l'aide des trois opérateurs "conjonctifs" étudiés précédemment. En particulier, la différence permet d'exprimer toutes les requêtes où figure une négation : on veut

sélectionner des données qui *ne* satisfont *pas* telle propriété, ou tous les “untels” *sauf les* ’x’ et les ’y’, etc.

Illustration concrète sur la base de données avec la requête suivante :

Voici quelques exemples complémentaires qui illustrent ce principe.

1. ……
   ……
   ……
2. ……
   ……
   ……
3. ……
   ……
   …….....

Exercice d’application

Soient les relations :

VIN (num_vin, cru, annee, degre),

BUVEURS (num_buv, nom, adresse, nationalite),

ABUS (num_buv, num_vin, quantite)

1 - Quels sont les noms et adresses des buveurs ayant bu plus de dix bouteilles de Chablis 1979 et quel est le degré de ce vin ?

2 – Quels sont les vins qui ont été produit avant 1960 et qui ont un degré supérieur ou égal à 2 ?

3 – Donnez le numéro et le nom des buveurs français.

LE LANGAGE SQL

I- La définition des données

A- Création de Table

L'instruction **CREATE TABLE**, permet de créer une nouvelle table.

Syntaxe

CREATE TABLE<Nom table>(<*champ1*><*type*>*[(*<*taille*>)] [NOT NULL] [, <*champ2*><*type*>*[(*<*taille*>)] [NOT NULL] [, <*champn*>*……*]

Les types de données habituels sont :

Type de données	Signification
INTEGER	Pour les entiers
DECIMAL (x, y)	X pour la longueur total dont y décimales
CHAR(n)	Pour les chaines de caractère de longueur n
DATE	Pour les dates (forme jj/mm/aaaa)

Exemple

Soit à créer la base de données de l'application 2 dont nous rappelons ici le SRD ou SLR :

FILIERE (CodFil, LibFil)

MATIERE (CodMat, LibMat)

SPORT (CodSp, LibSp)
ELEVE (Nmat, NomEle, Prenom, Sexe, Age, Datins, Tel, #CodFil, #CodSp)

DISPENSER (#CodFil, #CodMat, Coef)

CREATE TABLE FILIERE (CodFil INTEGER (4), LibFil CHAR (20));

B- Suppression de table

L'instruction **DROP TABLE** permet de supprimer une table précédemment créée ou un index d'une table

Syntaxe

DROP TABLE< Nom table> ;

Exemple

DROP TABLEFILIERE ;

II- L'interrogation des données

Une requête SQL s'écrit sous la forme générale suivante :

SELECT< Liste des attributs>

FROM< Liste des tables>]

[WHERE< Conditions sur un ou plusieurs attributs>]

[GROUP BY< attribut>]

[HAVING< Conditions sur un ou plusieurs calculs d'agrégation>]

[ORDER BY<Liste des attributs>]

A- Projection

Opérateur unaire générant une table de degré inférieur au degré de la table source ;

La requête SQL s'écrit sous la forme suivante

SELECT [DISTINCT] [TOP <nombre>] < Liste des attributs>

FROM< Liste des tables> ;

- ***DISTINCT*** : permet d'éliminer les éventuels duplicatas
- ***TOP*** : permet de limiter le résultat au nombre des première lignes spécifié par <nombre>, cette clause exige le tri au préalable.

  Exemple

  R : Liste des filières disponibles dans cet établissement

  SELECT *

  FROM FILIERE ;

  NB : cette requête affichera la table FILIERE telle qu'elle se présente.

  Afficher les codes et noms de toutes les filières

  SELECT CodFil, LibFil

  FROM FILIERE;

B- Selection

La requête SQL s'écrit sous la forme suivante :

Syntaxe

SELECT< Liste des attributs>

FROM< Liste des tables>]

WHERE< Conditions sur un ou plusieurs attributs> ;

Opérateur unaire permettant de réduire la cardinalité d'une table, c'est-à-dire ne donnant que des occurrences (enregistrement) vérifiant la condition spécifiée.

Exemple

R : Liste des élèves garçons

SELECT *

FROM ELEVE

WHERE sexe=''M'';

Les conditions s'expriment de plusieurs manières à s'avoir :

- <Attribut ><opérateur de comparaison><valeur>

Les opérateurs de comparaison sont :<, >, =, <=, >=, <>

Les opérateurs

- **Opérateur Between …And**

Détermine si la valeur d'une expression est comprise dans un intervalle de valeurs données.

Syntaxe

<Attribut> [**Not**] **BETWEEN** < bonne inférieure>**AND**<bonne supérieure>

Exemple

Liste des filles inscrites au cours du mois d'octobre 2009

Select *

From ELEVE

Where Sexe=''F'' **AND** Datins **between** # 01/10/2009 # **AND** # 31/10/2009#;

- **Opérateur In**

Détermine si la valeur d'une expression est égale à l'une des valeurs comprises dans une liste donnée.

Syntaxe

<Attribut> [**Not**]**In** (< valeur1>, <valeur2>,….< valeur n>.)

Exemple

R : Trouver le nom et prénoms des élèves qui ont (12, 15, 17, et 18ans).

Select nomEle, PreEle

From ELEVE

Where Age **IN** (12, 15, 17, 18);

- **Opérateur Like**

Compare une expression chaine avec un modèle dans une expression SQL

Syntaxe

<Attribut> [**Not**]**Like**< chaine de caractère> ;

Exemple

R : Liste des noms des matières commençant par A

Select LibMat

From MATIERE

Where LibMat **LIKE** ‘’A*’’;

La chaîne de caractère peut contenir des caractères génériques

Symbole	Utilisation	Exemple
*	Pour n’importe quelle chaine de caractère	Wh* trouve what, why
?	Pour un seul caractère	B ?ll trouvre ball ;bell ; bill
#	Pour un seul chiffre	1#3 trouve 103 ;113 ;123
[ ]	Représente un seul caractère parmi ceux indiqué entre crochets	B[ae]ll trouve ball et bell
!	Représente tout caractère ne figurant pas entre crochets	B[!bae]ll trouve bill et bull

		mais pas bell ni ball
-	Représente l'un des caractères de la plage indiquée	B [a-c]d trouve bad, bbd, et bcd

C- Jointure

Opération permettant d'extraire des données issues de plusieurs tables tout en liant ces dernières deux à deux par la valeur de la clé primaire d'une table et celle de la clé étrangère de la seconde table.

Syntaxe

SELECT< Liste des attributs>

FROM< table1>, < table2>

WHERE<Table1.Champ commun= Table2.Champ commun>

AND< Conditions sur un ou plusieurs attributs> ;

Exemple

- Liste des élèves inscrits dans le mois de septembre 2009 (matricule, nom, prénom des élèves et la désignation de leur filière)

SELECT Nmat, NomEle, PreEle, LibFil

FROM ELEVE, FILIERE

WHERE ELEVE.CodFil=FILIERE.CodFil

AND Datins B**etween** # 01/09/2009# **and** #30/09 /2009#;

D- Les regroupements (GROUP BY)

Cette clause **GROUP BY** permet de constituer des sous-ensembles d'occurrences dans une table.

La clause **HAVING** sert de critère de sélection pour les groupes c'est dire permet d'extraire les sous-ensembles d'occurrences qui satisfont la condition spécifiée.

Syntaxe

SELECT< Liste des attributs>

FROM< Liste des tables>

WHERE< Conditions sur un ou plusieurs attributs>

GROUP BY< attribut>

HAVING< Conditions sur un attribut de regroupement ou sur le résultat d'un calcul

D'agrégation>

Exemple

R : Liste des filières ayant le total des coefficients inférieur à cinq

Select FILIERE.codFil, LibFil, **SUM** (coef) as Total

From DISPENSER, FILIERE

Where DISPENSER.CodFil=FILIERE.CodFil

Group By FILIERE.CodFil, LibFil

Having SUM (coef) <5;

E- Les fonctions d'agrégations

Syntaxe	*Explication*
SUM (<attribut>)	Retourne le cumule des valeurs de l'attribut sur tout l'ensemble des occurrences ou par groupes en cas de présence de GROUP BY
COUNT (<attribut>)	Retourne le nombre de valeurs de l'attribut sur tout l'ensemble des occurrences ou par groupes en cas de présence de GROUP BY
AVG (<attribut>)	Retourne la moyenne des valeurs de l'attribut sur tout l'ensemble des occurrences ou par groupes en cas de

	présence de GROUP BY
MIN (<attribut>)	Retourne la plus petite valeur de l'attribut sur tout l'ensemble des occurrences ou par groupes en cas de présence de GROUP BY
MAX (<attribut>)	Retourne la plus grande valeur de l'attribut sur tout l'ensemble des occurrences ou par groupes en cas de présence de GROUP BY
FIRST (<attribut>)	Affiche la première valeur dans la table résultat
LAST (<attribut>)	Affiche la dernière valeur dans la table résultat

Exemple

R : Afficher l'effectif du complexe

Select **Count** (*) as Effectif

From ELEVE;

F- Le Tri des résultats (ORDER BY)

La clause ORDER BY, placé à la fin d'un SELECT, permet de trier le résultat obtenu selon la valeur d'un ou de plusieurs attributs. Le Tri peut être croissant (ASC) ou décroissant (DESC)

Syntaxe

SELECT< Liste des attributs>

FROM< Liste des tables>

WHERE< Conditions sur un ou plusieurs attributs>

GROUP BY< attribut>

HAVING< Conditions sur un ou plusieurs calculs d'agrégation>

ORDER BY<attributs><ordre > [, <attributs><ordre >, ….<attributs><ordre >]

Exemple

R : Liste des filières du plus grand au plus petit effectif

Select FILIERE.CodFil, LibFilcount(*) as Effectif

From ELEVE, FILIERE

Where ELEVE.CodFil=FILIERE.CodFil

Group ByFILIERE.CodFil, LibFil

Order By Count (*) **DESC**;

III- les mises à jour

A- Insertion d'enregistrements dans une table

Syntaxe

INSERT INTO<Nom table> [< Champ1, champ2,….champ n>] **VALEUR**< Valeur1, valeur2, ….valeur n>

Cette instruction ajoute un ou plusieurs enregistrements à une table. C'est ce qu'on appelle une ***requête Ajout***

Exemple

INSERT INTO MATIERE (CodMat, LibMat) **VALUE** (''ANG'', ''Anglais'') ;

B- Modification d'enregistrement dans une table

Syntaxe

UPDATE<Nom table>

SET<Champ>=<nouvelle valeur>

[WHERE< condition>];

Elle crée une ***requête de mise à jour*** qui modifie les valeurs des champs d'une table spécifié, selon des conditions déterminées.

Exemple

UPDATE MATIERE

SETCodMat=''AL'',LibMat=''Algorithme''

C- Suppression d'enregistrements dans une table

Syntaxe

DELETE

FROM<*nom table*>

WHERE<*condition*> *;*

Elle permet de supprimer un ou plusieurs enregistrements d'une table vérifiant la condition devant WHERE

Exemple

Supprimer tous les élèves dont l'âge est inférieur à 12

Delete
From ELEVE
Where Age < 12;

IV- **Notion de Sous –Requêtes SQL**

Une sous-requête est une instruction SELECT imbriquée dans une autre instruction SELECT. Dans ce cas seuls les attributs du premier SELECT sont projetés. La sous –requête doit être placée entre parenthèses. Elle ne doit pas comporter de clause ORDER BY, mais peut inclure les clauses GROUP BY et HAVING.

Syntaxe

SELECT< liste des attributs>

FROM< nomtable1>

WHERE<attribut><opérateur>

(**SELECT**<attribut>

FROM<nomtable2>

WHERE<condition>) ;

Le résultat d'une sous requête est utilisé par la requête du niveau supérieur. Une sous-requête est exécutée avant la requête de niveau supérieur.

Une sous-requête peut ramener une ou plusieurs lignes. Les opérateurs de comparaison (**=, <, >, <=, >=**) permettent d'en extraire une et les opérateurs **IN, ANY**, **ALL** permettent d'en ramener plusieurs.

Remarques

- **IN** compare un élément à une donnée quelconque d'une liste ramenée par la sous-requête. Cet opérateur est utilisé pour les équijointures et les insertions. L'opérateur **NOT IN** est employé pour les différences

- ☞ **ANY** compare l'élément à chaque donnée ramenée par la sous requête. L'opérateur = **AN**Y équivaut à IN. L'opérateur < **ANY** signifie « inférieur à au moins une des valeurs », donc « inférieur au maximum ». L'opérateur > **ANY** signifie « supérieur à au moins une des valeurs » donc « supérieur au minimum »
- ☞ **ALL** compare l'élément à ceux ramenés par la sous requête. L'opérateur < **ALL** signifie « inférieur à toutes les valeurs », donc « inférieur au minimum ». L'opérateur > **ALL** signifie « supérieur à toutes les valeurs » donc « supérieur au maximum »

Opérateur EXISTS

Il permet de créer une sous requête qui renvoie vrai si elle retourne de valeur. En d'autres termes il permet d'interrompre la sous requête dès le premier enregistrement trouvé. Si aucun enregistrement n'est extrait par la sous-requête, la valeur FAUX est retournée. L'opérateur NOT (c'est-à-dire NOT EXISTS) retourne la valeur VRAI si aucun enregistrement n'est extrait par la sous-requête. Il peut être employé pour les différences.

APPLICATIONS

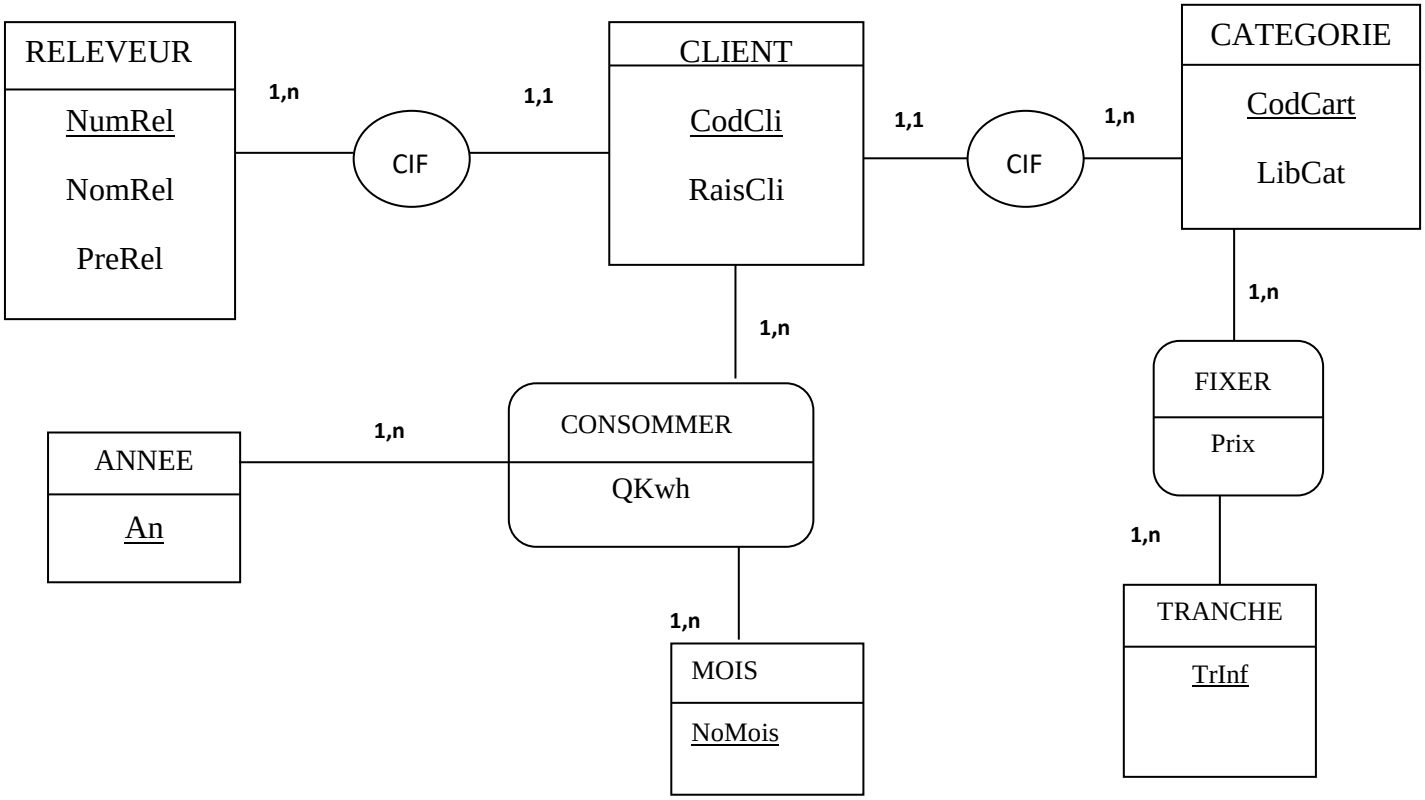
APPLICATION *1*

Exercice 1

La **S2E** (**Société EGOTA-EGOTCHI**) est une société de distribution d'énergie électrique à la population du ZOGBINLO, pays situé à l'Est du continent africain et qui est dans le besoin permanent de cette énergie pour son développement.

Par catégorie de client les tarifs d'électricité sont définis selon les tranches. Les consommations des clients (Kwh) sont relevées à la fin de chaque mois de l'année.

La société utilise actuellement une base de données dont un extrait du schéma conceptuel est présenté ci-dessus.



1) Définir : Propriété ; Entité, Clé primaire, cardinalité
2) Proposez un dictionnaire des données correspondant à ce MCD
3) Déterminer le nombre d'entités, d'associations, d'associations binaires et de CIF
4) Justifier les cardinalités entre : CATEGORIE et CIF puis CLIENT et CIF
5) Dresser la liste des dépendances fonctionnelles élémentaires et directes et en déduire la matrice

6) Présenter le Schéma Logique Relationnel correspondant au SCD ou MCD

Exercice 2 :

Considère la **Base de Données** d'une compagnie **ALINAGNON** spécialisé dans le transport inter-urbain dont le Schéma Relationnelle (**SRD**) se présente comme suit :

CATEGORIE (CodCat, LibCat)

CONDUCTEUR (CodCond, NomCond, PreCond, AgeCond)

AUTOCAR (NumImmat, NombrePlace, #CodCat)

LIGNE (CodLigne, LibLigne Distance)

VOYAGE (NoVoy, DateVoy, Heure, Prix, #NumImmat, #CodCond, #CodLigne)

Rédiger en SQL les requêtes ci-après

1. Afficher la liste des noms, prénoms et âge des conducteurs.
2. Afficher la liste des conducteurs (nom et prénom) ayant fait la ligne **Cotonou-Parakou** le 08 juin 2010.
3. Afficher la liste des conducteurs des moins âgés aux plus âgés.
4. Liste des conducteurs dont le prénom contient la syllabe **yoyo**
5. Liste des véhicules utilisés durant le mois de Novembre 2010
6. Nom, prénom et âges des conducteurs ayant conduit un véhicule de 5 places sur la route **Cotonou-Ouidah** durant le mois de mai 2010.
7. Nombre de véhicules de catégorie LUXE.
8. Liste des véhicules non utilisés durant la période de mars 2010.
9. Liste des nouvelles lignes créées (affichez le code et le libellé) qui n'ont pas encore connu de voyage.

Exercice 3 : Conception de la base de données BURO

BURO société de vente de matériel de bureau : papeterie, fourniture, armoires, sièges… Voici les informations présentes sur une commande de BURO :

- Date
- N°commande
- N° de client
- Société
- Contact
- Adresse
- Ville
- Tel et Fax
- Email client
- Adresse livraison
- Date livraison
- N° du vendeur
- Nom du vendeur
- Ref produit
- Description
- Prix unitaire
- Quantité
- Sous total ht (pour chaque ligne de commande)
- Total HT
- TVA
- Total TTC
- Escompte
- Net à payer

1) Regrouper ces informations dans des tables, en respectant les contraintes suivantes :

- Regrouper les champs dans des tables qui peuvent être reliées
- Pas de dédoublement des champs, sauf pour les champs communs
- Pas de champs calculables

2) Définissez les clés primaires et les clés étrangères pour les relations entre les tables
3) Déduisez le schéma conceptuel des données correspondant.

Exercice 4 : Clé primaire et clé étrangère (Conception sous MS ACCESS)

On considère la structure d'une base de données « bibliothèque » composée des 5 tables suivantes :

OUVRAGE (num_ouvrage, éditeur, titre, premier_auteur)

EXEMPLAIRE (num_ouvrage, num_exemplaire, date_acquisition, prix, état_exemplaire) avec état_exemplaire = {sorti, disponible, réservé}

ABONNE (num_abonné, nom_abonné, date_abonnement, nbre_emprunts_en_cours, état_abonné) avec état_abonné = {valide, relancé, exclu}

DEMANDE_EMPRUNT (num_demande, date_demande, num_abonné, num_ouvrage, état_demande) avec état_demande = {en_attente, acceptée, refusée}

EMPRUNT (num_emprunt, date_emprunt, num_demande, num_ouvrage, num_exemplaire)

Donner les clés primaires et étrangères pour chacune des tables ci-dessus.

APPLICATION *2*

Soit le schéma logique des données ci-après :

FILIERE (CodFil, LibFil)

MATIERE (CodMat, LibMat)

SPORT (CodSp, LibSp)

ELEVE (Nmat, NomEle, PreEle, Sexe, Age, Datins, #CodFil, #CodSp)

DISPENSER (#CodFil, #CodMat, Coef)

Propositions des extensions des différentes tables

Table **FILIERE**

CodFil	**LibFil**
BF	*Banque Finance*
CGAF	*Contrôle de Gestion Audit et Finance*
GRH	*Gestion des Ressources Humaines*
IG	*Informatique de Gestion*
TL	*Transport Logistique*
MAC	*Marketing Action Commerciale*

Table **SPORT**

CodSp	**LibSp**
F1	*Footbal*
F2	*Handball*
F3	*Basketball*
F4	*Tennis*
F5	*Gymnastique*
F6	*Volleyball*

Table **MATIERE**

Table **DISPENSER**

CodMat	**CodFil**	**Coef**
TEEO	*GRH*	*2*
TEEO	*IG*	*2*
SIG	*CGAF*	*3*
AL	*IG*	*5*
SIG	*BF*	*2*
ME	*IG*	*5*

EC	*GRH*	*2*

CodMat	LibMat
TEEO	*Technique d'Expression Ecrite et Orale*
SIG	*Système d'Information de Gestion*
ANG	*Anglais*
AL	*Algorithme*
ME	*Merise*
CA	*Comptabilité Analytique*
EC	*Economie*

ELEVE

NMat	NomEle	PreEle	Sexe	Age	Datins	CodFil	CodSp
E001	*AKISSI*	*Delta*	*F*	*23*	*20/09/2009*	*IG*	*F3*
E002	*KOUAME*	*Henri*	*M*	*26*	*16/09/2009*	*CGAF*	*F3*
E003	*ADEBAYO*	*Salami*	*M*	*22*	*22/09/2009*	*GRH*	*F2*
E004	*BELLO*	*Missilath*	*F*	*25*	*01/10/2009*	*IG*	
E005	*DINDANE*	*Arounan*	*M*	*30*	*04/10/2009*	*CGAF*	*F1*
E006	*MOULERO*	*Kafayath*	*F*	*20*	*09/10/2009*	*IG*	*F2*
E007	*KOFFI*	*Vigor*	*M*	*21*	*12/10/2009*	*GRH*	
E008	*BIO*	*Gilbert*	*M*	*23*	*20/09/2009*	*IG*	*F3*
E009	*TESSI*	*Izack*	*M*	*26*	*25/10/2009*	*CGAF*	*F2*
E010	*BELLO*	*Azizath*	*F*	*28*	*30/10/2009*	*GRH*	*F2*

E011	*KOUAME*	*Henri*	*M*	*30*	*30/08/2009*	*GRH*	
E012	*AKIBOU*	*Bastou*	*M*	*25*	*30/10/2009*	*IG*	*F1*
E013	*KOFFI*	*Toulassi*	*M*	*30*	*20/09/2009*	*CGAF*	*F2*

1- Créer la base de données selon le SLD puis enregistrer les données de chaque table
2- En tenant compte des extensions des différentes tables et le schéma logique relationnel, rédiger en algèbre relationnelle (lorsque c'est possible) puis en SQL les requêtes ci-après et en déduire les tables réponses

R1 : Listes des noms, prénoms et sexe des élèves.

R2 : Listes des élèves avec indications de numéro matricule, nom, prénom et age .

R3 : Listes des matières (Nom matière).

R4 : Listes des filières disponibles dans cet établissement.

R5 : Listes des élèves garçons.

R6 : Nom et prénom des élèves de moins de 25 ans d'âge.

R7 : Nom, prénom et sexe des élèves inscrits le 20/09/2009.

R8 : Liste des noms des matières commençant par A.

R9 : Liste des élèves dont le prénom contient la syllabe tou

R10 : Liste des élèves inscrits dans le mois d'août 2009 et ceux inscrits dans le mois d'octobre 2009 (matricule, nom et prénom des élèves).

R11: Liste des filles inscrites au cours du mois d'octobre 2009.

R 12 : Liste des élèves inscrites en IG (matricule, nom, prénom des élèves)

R13 : Liste des élèves inscrits dans le mois de septembre 2009 (matricule, nom, prénom des élèves et la désignation de leur filière).

R14 : Liste des élèves pratiquant le handball ou le basketball avec indication de Nom, Prénom, code et nom de sport pratiqué.

R15 : Liste des matières enseignées dans la filière IG (Code, libellé et coefficient des matières).

R16 : Liste des élèves dispenses du sport

R17 : Effectif du complexe

R18 : Effectif de la filière ‘’Gestion des Ressources Humaines’’

R19 : Effectif de chaque filière

R20: Liste des filières du plus grand au plus petit effectif

R21: Pour chaque libellé de filière , donnez le total des coefficients

R22: Afficher le code de la filière ayant le total des coefficients le plus élevé.

R23: Liste des filières ayant le total des coefficients inférieur à cinq

R24 : Afficher le libellé de la matière ayant le plus petit coefficient en ‘’IG’’ et de même le coefficient.

APPLICATION *3*

Objectifs :

- Comprendre la notion de base de données
- Comprendre les concepts permettant de structurer une base de données
- Construire une base de données sur Access en utilisant les fonctionnalités de création de table, de champ et de lien entre tables

Créer une base de données nommée **UNIVERSITE**, qui contient 2 tables : **DIPLOME** et **ETUDIANT**

1. Créer d’abord la table **DIPLOME** avec les champs suivants :
   CodeD (NumeroAuto) , **LibD** (texte, longueur 10) , **Duree** (numérique, octet)
2. Appliquer les propriétés de champs suivantes :

- Clé primaire sur le champ **CodeD**, insérer la légende « Code du diplôme »
- Saisie obligatoire du champ **LibD** et affiché en majuscule
- Pour le champ **Duree**, insérer la légende « Durée du cursus en années »

3. Créer ensuite la table **ETUDIANT** avec les champs suivants :

NEtudiant (Numérique, entier long), **Nom** (texte, longueur 60), **Prenom** (texte, longueur 60), **Sexe** (texte, longueur 1), **Datenaissance** (Date/Heure, date complète), **Doublant** (Oui/Non), **Droits** (Monétaire) et **Statut** (Liste de choix).

- Clé primaire sur le champ **NEtudiant**
- Saisie obligatoire du nom d’étudiant et affiché en majuscule

- Pour le champ Sexe, autoriser uniquement la saisie de la lettre M ou F
- Pour le champ Doublant, la valeur par défaut est Non
- Pour le champ Droits, la saisie est refusée si la valeur dépasse 1000, et le message d'erreur « Le montant doit être inférieur à 1000 » est affiché
- Pour le champ statut, les seules valeurs autorisées sont : Formation Continue, Formation Initiale, Formation Alternance

4. Pour établir une relation entre les 2 tables, que faut-il ajouter à la table ETUDIANT ?
5. Etablir la relation avec intégrité référentielle
6. Saisir dans chaque table quelques enregistrements de votre choix et vérifier les contraintes (par exemple, sur les doublons, sur les droits …)
7. Vérifier l'intégrité référentielle (par exemple en saisissant un code diplôme inexistant, en supprimant un diplôme…)

APPLICATION *4*

Objectifs :

- Définir la structure d'une base de données
- Réaliser une base de données (déclaration des tables, des champs et des liens)
- Ecrire des requêtes en utilisant l'interface graphique Access

Exercice 1 : Clinique de médecine

Soit la base de données suivante :

PATIENT (NoPatient, NoAssSociale, Nompat, Prenompat)

MEDECIN (NoMedecin, Nommed, Prenommed)

DIAGNOSTIC (NoDiagnostic, descdiagnostic)

TRAITEMENT (NoTraitement, desctraitement)

ENTREE_DOSSIER (NoDossier, DateVisite, #NoPatient, #NoMedecin, #NoTraitement, #NoDiagnostic)

1) Donnez le code SQL pour créer la base de données

2) Vous réalisez que la taille de l'attribut "description" de la table DIAGNOSTIC n'est pas adéquate. Donnez le code SQL pour la modifier pour une chaîne de longueur variable de 255 caractères maximum.

3) Donnez le code SQL pour ajouter les attributs "NoTelephone" et "DateNaissance" dans la table PATIENT.

4) Donnez le code SQL pour entrer les données suivantes dans la base de données

5) Vous avez entré le mauvais traitement dans l'entrée de dossier no. 3. Modifiez l'enregistrement pour donner le traitement no. 2 au lieu du no. 1.

6) Effectuez les requêtes SQL simples suivantes :

- Afficher toutes les informations de tous les patients ;
- Afficher le nom et le prénom de tous les patients ;
- Afficher le nom et le prénom des patients dont le nom de famille est 'Delisle' ;
- Afficher le nom et le prénom des patients nés après 1976 ;
- Afficher les noms de famille différents des patients ;
- Afficher les patients en ordre croissant de date de naissance ;
- Afficher les entrées de dossier où le patient traité est de no. 111111 et le médecin traitant est de no. 67899

7) Effectuez les jointures suivantes :

- Afficher toutes les entrées de dossier et les informations de leurs patients respectifs ;
- Afficher les entrées de dossier de Pierre Delisle ;
- Afficher la description des traitements dont a bénéficié Pierre Delisle ;
- Afficher, du plus jeune au plus vieux, le nom et le prénom des patients traités par René Lajoie le 26 avril 2008.

Exercice 2 : Bibliothèque

Soit la base de données suivante :

SPECIALITE (NoSpecialite, Description)

SECTION (NoSection, Emplacement, Description)

LIVRE (CodeISBN, Titre, #NoSpecialité, #NoSection)

FOURNISSEUR (NoFournisseur, Nom)

EXEMPLAIRE (NoExemplaire, #CodeISBN, #NoFournisseur)

ABONNE (NoAbonne, Nom, Prenom)

CARTE (NoCarte, DateDebut, DateFin, #NoAbonne)

EMPRUNT (#NoExemplaire, #NoCarte, DateLocation, DateRetour)

Effectuez les requêtes SQL suivantes :

1- Afficher la liste des livres classés dans les sections 1 et 4 ;

2 - Ajouter un attribut adresse à la table ABONNE ;

3 - Ajouter le fournisseur 'Livres du Québec inc.' à la base de données ;

4 - Afficher le nom et le prénom des abonnés qui se sont abonnés ou ont renouvelé leur carte en 2012 ;

5 - Afficher le code et le titre des livres qui ont été empruntés le 28 avril 2012, triés par ordre alphabétique de titre ;

6 - Afficher le nom et le prénom des abonnés qui ont déjà emprunté le livre intitulé 'Nos amis les français';

7 - Prolonger tous les abonnements échus le 25 avril 2012 au 25 mai 2012 ;

8 - Afficher le titre des livres de science-fiction empruntés durant le mois d'avril 2012.

Exercice 3 : Gestion des incidents

Afin d'assurer la qualité des produits attendues par les Clients, l'entreprise cherche à optimiser la gestion des pannes pouvant survenir dans les infrastructures de production nécessaires à la fabrication du Ciment. Voici un extrait de la base de données :

TECHNICIEN (idTech, nom, prenom, specialite)
STATION (idstat, nom, Position, coordLat, coordLong,phase)
MACHINE (idmach, état, dateMiseEnService, dateDernièreRévision, #idStat)
TYPEINCIDENT (idType, description, tempsRéparationPrévu)
INCIDENT (idInd, remarques, dateHeure, dateHeureCloture,#idmach,#idType)
INTERVENTION (idInterv, dateHeureDébut, dateHeureFin, #idInd, #idTech)

1. Rédiger la requête SQL permettant d'obtenir la liste par ordre alphabétique des noms et prénoms des techniciens ayant réalisé une intervention sur la Machine identifiée par Ber001.
2. Rédiger la requête SQL permettant d'obtenir la liste des phases ayant connue un incident de "sur-chauffage" pour le mois Mai 2019.
3. Rédiger la requête SQL permettant d'obtenir la liste des noms des stations ayant eu plus de dix incidents.

Exercice 4 : Du producteur au consommateur

Soit le modèle relationnel suivant :

PRODUCTEUR (#raison_sociale : chaîne(25), ville:chaîne(255))

CONSOMMATEUR (#login: chaîne(10), #email:chaîne(50), nom:chaîne(50),prenom:chaîne (50), ville:chaîne(255))

PRODUIT (#id:entier, description:chaîne(100), produit-par=>Producteur, consomme par login=>Consommateur, consomme-par-email=>Consommateur)

On ajoute que :

-(nom, prenom, ville) est une clé candidate de Consommateur
-Tous les produits sont produits
-Tous les produits ne sont pas consommés

Question

1 - Rétro-concevez le modèle conceptuel sous-jacent à ce modèle relationnel.

2 - Insérez les données dans votre base de données correspondant aux assertions suivantes :

--L'entreprise de Compiègne "Pommes Picardes SARL" a produit 4 lots de pommes, et 2 lots de cidre.

--Il existe trois utilisateurs consommateurs dans la base, donc les adresses mails sont : Al.Un@compiegne.fr - Bob.Deux@compiegne.fr - Charlie.Trois@compiegne.fr. Ce sont des employés de la ville de Compiègne qui habitent cette ville. Leur mail est construit sur le modèle Prenom.Nom@compiegne.fr. Leur login est leur prénom.

3 - Modifiez les données de votre base de données pour intégrer les assertions suivantes :

--1 lots de pommes a été consommés par Al Un.

--2 lots de pomme ont été consommé par Bob Deux.

--Tous les lots de cidre ont été consommés par Al Un.

4 - Charlie Trois n'ayant rien consommé, modifiez votre base de données afin de le supprimer de la base.

Exercice 5 : Hôtel

Soit la Base de données Hôtel qui contient 3 Tables « Chambre », « Client » et « Réservation » qui sont définies comme suit :

CHAMBRE (Num_Chambre, Prix, Nbr_Lit, Nbr_Pers, Confort, Equ)

CLIENT (Num_Client, Nom, Prenom, Adresse)

RESERVATION (#Num_Client, #Num_Chambre, Date_Arr, Date_Dep)

Exprimer les requêtes suivantes en SQL :

1 - Les numéros de chambres avec TV.

2 - Les numéros de chambres et leurs capacités.

3 - La capacité théorique d'accueil de l'hôtel.

4 - Le prix par personne des chambres avec TV.

5 - Les numéros des chambres et le numéro des clients ayant réservé des chambres pour le 09/02/2004.

6 - Les numéros des chambres coûtant au maximum 80 Euro ou ayant un bain et volant au maximum 120 Euro.

7 - Les Nom, Prénoms et adresses des clients dont le nom commence par « D ».

8 - Le nombre de chambres dont le prix est entre 85 et 120 Euro.

9 - Les noms des clients n'ayant pas fixé la date de départ.

Exercice 6 : Bibliothèque Bis

On suppose qu'une bibliothèque gère une base de données dont le schéma est le suivant (les clés primaires des relations sont soulignées) :

EMPRUNT (Personne, Livre, DateEmprunt, DateRetourPrevue, DateRetourEffective)

RETARD (Personne, Livre, DateEmprunt, PenaliteRetard)

Exprimer, lorsque cela est possible, les requêtes suivantes en algèbre relationnelle et en SQL.

1 - Quelles sont les personnes ayant emprunté le livre « Recueil Examens BD » ?

2 - Quelles sont les personnes n'ayant jamais rendu de livre en retard ?

3 - Quelles sont les personnes ayant emprunté tous les livres (empruntés au moins une fois) ?

4 - Quels sont les livres ayant été empruntés par tout le monde (i.e. tous les emprunteurs) ?

5 - Quelles sont les personnes ayant toujours rendu en retard les livres qu'elles ont empruntés ?

Exercice 7 : Cafétéria

L'objectif d'être capable de réaliser n'importe quelles requêtes. Faire les requêtes SQL permettant de répondre aux demandes sur la base de données suivantes :

SERVIR (Cafe, Boisson)

FREQUENTER (#Client, #Cafe)

APPRECIER (Client, Boisson)

Exprimer les requêtes suivantes en SQL.

1 - Les Cafés qui servent une Boisson Apprécier par 'Ahmed'.

2 - Les Clients qui vont dans les mêmes Cafés que Ahmed.

3 - Les Clients qui fréquentent au moins un Café où l'on sert une Boisson qu'ils aiment.

4 - Les Clients qui ne fréquentent aucun Café où l'on sert une Boisson qu'ils aiment.

5 - Les Clients qui fréquentent tous les Cafés.

6 - Les Clients qui fréquentent tous les Cafés qui servent au moins une Boisson qu'ils aiment.

7 - Les Clients qui ne fréquentent que les Cafés qui servent une Boisson qu'ils aiment.

8 - Donner pour chaque Client, le nombre de Cafés servant une Boisson qu'ils aiment.

9 - Les Clients qui fréquentent au moins 2 Cafés où l'on sert une Boisson qu'ils aiment

Exercice 9: Spectacle

Un organisme de gestion de spectacles, de salles de concert et de vente de billets de spectacles gère une base de données dont le schéma relationnel est le suivant :

SPECTACLE (Spectacle_ID, #Salle_ID, Titre, DateDéb, Durée, Chanteur)

CONCERT (Concert_ID, #Spectacle_ID, Date, Heure)

SALLE (Salle_ID, Nom, Adresse, Capacité)

BILLET (Billet_ID, #Concert_ID, Num_Place, Catégorie, Prix)

VENTE (Vente_ID, #Billet_ID, Date_Vente, MoyenPaiement)

Les attributs soulignés sont les attributs appartenant à la clé primaire. Ils sont de type entier.

L'attribut Salle_ID de la relation Spectacle est une clé étrangère qui fait référence à l'attribut de même nom de la relation Salle. L'attribut Spectacle_ID de la relation Concert est une clé étrangère qui fait référence à l'attribut de même nom de la relation Spectacle. L'attribut Concert_ID de la relation Billet est une clé étrangère qui fait référence à l'attribut de même nom de la relation Concert.

L'attribut Billet_ID de la relation Vente est une clé étrangère qui fait référence à l'attribut de même nom de la relation Billet.

Exprimez, lorsque cela est possible, les requêtes suivantes en algèbre relationnelle, en calcul relationnel à variable nuplet et en SQL.

1 - Quelles sont les dates du concert de Corneille au Zenith ?

2 - Quels sont les noms des salles ayant la plus grande capacité ?

3 - Quels sont les chanteurs n'ayant jamais réalisé de concert à la Cygale ?

4 - Quels sont les chanteurs ayant réalisé au moins un concert dans toutes les salles ?

5 - Quels sont les dates et les identificateurs des concerts pour lesquels il ne reste aucun billet invendu ?

Exercice 10 : Base TOURISME D'HIVER

Cette application se focalise tout particulièrement sur les requêtes imbriquées et tout particulièrement celles introduites par les opérateurs (ANY, ALL, IN, EXISTS. . .)

Soit le schéma de la base de données TOURISME D'HIVER suivante :

STATION (numSta, nomSta, altitude, dept)

HOTEL (numHot, nomHot, #numSta, categorie)

CHAMBRE (#numHot, numCh, nbLits)

CLIENT (numCli, nomCli, adrCli, telCli)

RESERVATION (#numcli, #numHot, #numCh, dateDeb, dateFin, nbPers)

numSta : numéro identifiant de station ; *nomSta* : nom de station

altitude : altitude où se trouve la station ; *dept* : département ;

numHot : numéro identifiant d'hôtel ; *nomHot* : nom de l'hôtel ;

numSta : numéro référençant la station où se trouve l'hôtel ;

categorie : catégorie de l'hôtel ; *numHot* : numéro référençant l'hôtel ;

numch : numéro de la chambre ; *nbLits* : nombre de lits dans la chambre

numCli : numéro identifiant du client ; *nomcli* : nom du client ;

adrCli : adresse du client ; *telCli* : numéro de téléphone du client ;

numCli : numéro référençant le client qui a effectué la réservation

numHot : numéro référençant l'hôtel concerné par la réservation

numCh : numéro référençant la chambre réservée

dateDeb : date de début de séjour ;

dateFinN : date de fin de séjour ;

nbPers : nombre de personnes concernées par la réservation.

Donner les requêtes SQL permettant de répondre aux besoins suivants :

1. Donner le nom des hôtels qui se trouvent dans un département contenant le mot ’OU’.

2. Donner le nom des clients ayant effectué une réservation dans un hôtel d’une station de ski se trouvant à une altitude supérieure à 1500m. Vous proposerez une requête SQL ne contenant aucun signe ’=’.

3. Donner le nom des clients n’ayant jamais réservé dans un hôtel se trouvant dans le département ’ALIBORI’.

4. Donner le nom de la station d’altitude la plus élevée du département ’BORGOU’.

5. Donner le nom des clients ayant fini au moins un de leurs séjours avant le début de n’importe quel séjour effectué par le client nommé ’FASSINOU Narsiste’.

6. Donner le nom des hôtels qui ont au moins une chambre de capacité strictement plus grande qu’une des chambres de l’hôtel ’JECO’ de la station ’Chamrousse’.

7. Donner le nom des clients qui ont occupé au moins une fois une des chambres réservées par le client nommé ’GRIMAUD Julien’. (Une version avec un IN et une version avec un ANY)

8. Donner le nom des stations ayant au moins un hôtel nommé ’Beau Rivage’.

9. Donner le nom des clients n’ayant jamais réservé d’hôtels dans une station du département ’LES COLLINES’ (sans utiliser de IN).

10. Donner le nom des clients ayant fait une réservation en 2021 (vous utiliserez une requête imbriquée dans le FROM pour identifier les réservations de 2021)

11. Donner le nom des hôtels ayant hébergé au moins une fois tous les clients de l’hôtel ’Les 3 marmottes’