Michel SALA

COURS DE BASE DE DONNEES

Références utilisées

- Libourel Thérèse - Support de cours de base de données (**www.lirmm.fr/~libourel**)
- Philippe Rigaux (**http://www.lamsade.dauphine.fr/rigaux**)
- **http://solutions.journaldunet.com**
- **http://fadace.developpez.com/sgbdcmp/story/**
- Pierre Delisle **http://wwwens.uqac.ca/~pdelisle/**
- http://philippe.guezelou.free.fr/

Bibliographie

- Carrez C., Des Structures aux Bases de Données, Masson
- Marcenac, P., SGBD relationnels, Optimisation des performances, Eyrolles.
- Unified Modelling Langage, James Rumbaugh, Ivar Jacobson, Grady Booch, Campus Press UML 2.0, 2005

SOMMAIRE

1 Introduction

1.1 Les données

De tout temps, les hommes ont eu besoin de stocker des données et de les retrouver. Si nous regardons les différentes définitions du mot « données » :

Ce qui est donné, connu, déterminé dans l'énoncé d'un problème et qui sert à découvrir ce qui est inconnu

Dictionnaire Le Robert

Ce qui est admis, connu ou reconnu, et qui sert de base à un raisonnement, de point de départ pour une recherche

Dictionnaire Le Robert

Le problème que nous allons avoir si nous manipulons des données c'est que celles-ci peuvent prendre différentes formes comme :

- Paul est une personne
- Paul a trente ans
- Paul est marié avec Agnès
- Il fait 20° Celsius
- La bourse a augmenté hier de 0,2%

Pour pouvoir manipuler des données, il va falloir les structurer :

Une personne
Un nom
Un prénom
Une adresse
Une date de naissance
Un sexe
…

La structure de données évoluant habituellement plus lentement que les valeurs des données.

Pour préciser le domaine de valeurs des données il va falloir les **typer** :

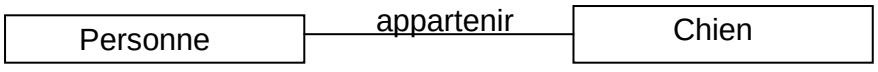
Un nom — chaîne de caractères de longueur 20
Un prénom — chaîne de caractères de longueur 20
Une adresse — elle-même pouvant être subdivisée en :

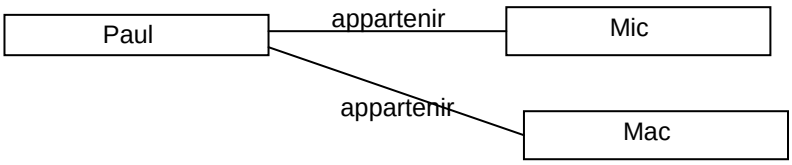
- Adresse1 chaîne de caractères de longueur 20
- Adresse2 chaîne de caractères de longueur 20
- Code postal chaîne de caractères de longueur 5
- Ville chaîne de caractères de longueur 20

Une date de naissance — date
Un sexe — valeur booléenne Vrai ou Faux.

Les principaux types de données sont : chaîne de caractères, nombre, réel, booléen, date.

Nous nous apercevons rapidement qu'il existe des relations entre structure de données. Si nous voulons représenter que **Paul a deux chiens Mic et Mac**, Paul est une instance de la structure Personne , Mic et Mac sont des instances de la structure Chien et il existe une relation entre Personne et Chien que nous pouvons appeler Appartenir qui est instancié par la phrase de départ.





Une des premières représentations des données fut les fichiers (exemple simple d'une feuille Excel) qui comprend la structure en colonne et les valeurs en ligne.

Fichier Personne

Nom	Prénom	Adresse	Date de naissance	Sexe
Martin	Paul	Montpellier	22/12/1985	M
Dupont	Amédé	Vic la Gardiole	01/08/1912	M
Hans	Helena	Paris	15/06/1989	F
Ferrari	Gaetano	Milan	08/05/1965	M

Fichier Chien :

Nom	Race	Date de naissance
Mic	Epagneul	01/12/1999
Mac	Berger allemand	05/08/2005
Sable		09/04/2004

La gestion de fichier parait simple mais le problème réside dans les accès par programmes, la sécurité des données et la concurrence des accès. Le GDF (Gestionnaire de Fichiers) ne gère pas la vision globale des données, la redondance ni la gestion des cohérences.

1.2 Les bases de données

L'idée des bases de données est de créer « un réservoir » où toutes les données seraient stockées de manière **unique** et où la **sécurité** serait centralisée. Les bases de données garantissent la persistance des données.

Une des premières définitions de base de données est :

« Une base de données peut être vue comme une **collection d'informations structurées** modélisant une « entreprise » de l'univers, et mémorisée sur un **support permanent**. »

Aux termes de l'arrêté du 22 décembre 1981 (Arr. 22 décembre 1981, JONC 17 Janv. 1982, p. 624), une base de données est « un ensemble de données relatif à un domaine défini des connaissances et organisé pour être offert aux consultations d'utilisateurs »

Dans cette base de données seront stockées toutes les données nécessaires aux traitements envisagés. Evidemment, à nous de concevoir le modèle de la base de données pour ne pas oublier des informations.

1.3 Le SGBD.

Le SGBD (Système de Gestion de Base de Données) est un logiciel qui permet de manipuler les informations dans la base de données.

Ce SGBD est l'interface entre les données stockées sur disque et les utilisateurs (administrateur, développeur, utilisateur final). Il leur fournit les outils qui vont permettre de manipuler les données (création, modification et suppression), de questionner la base (requête), de gérer les structures et de garantir la sécurité (tant au niveau accès que droits).

Les fonctions essentielles d'un SGBD sont des fonctions de **gestion de la structure** des données, d'**interrogation** et de **sécurité** des données.

Pour la gestion de la structure des données de la base, nous avons à notre disposition un **Langage de Définition des Données,** qui permet de décrire la structure **(**exemple du SGBD Oracle) :

CREATE TABLE PERSONNE
(code NUMBER (6) NOT NULL
nompers CHAR (20),
prénompers CHAR (20),
salaire NUMBER (8),
sexe BOOLEAN) ;

Pour interroger les données, nous avons à notre disposition : un **Langage de Manipulation de Données** qui comprend :

- un langage de programmation : ce langage permet de développer des programmes qui vont faire l'interface entre le SGBD et l'utilisateur (saisis des données de base, demandes d'interrogations, présentation des résultats…)
- langage d'interrogation : appeler aussi langage de requête. Dans les bases de données relationnelles, le langage standard d'interrogation est SQL (Structured Query Language).

La gestion globale de la base va être réalisée par un DBA (Database Administrator) qui peut affecter les droits d'utilisation.

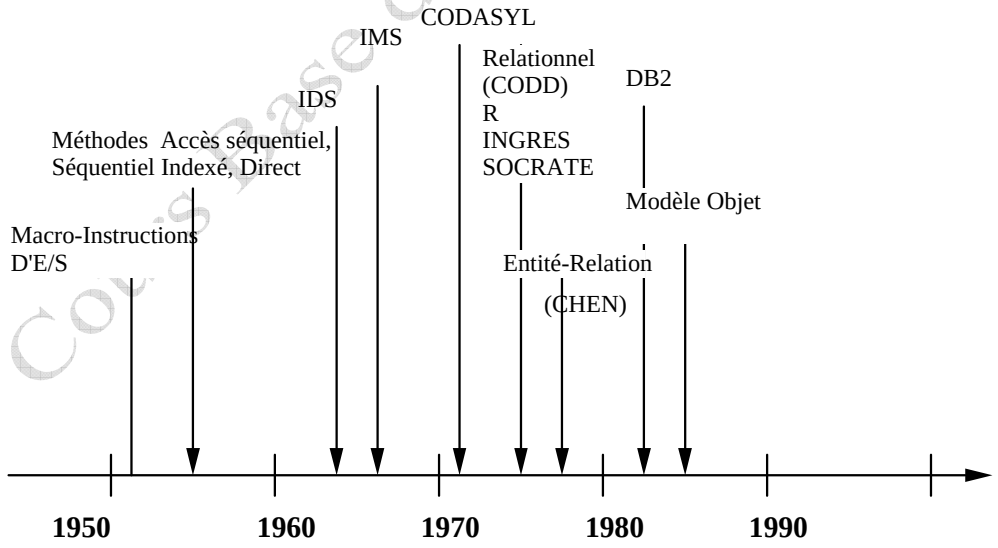
1.4 Historique

L'évolution historique des bases de données, suit les avancées réalisées dans des domaines divers comme : Systèmes d'exploitation, Langages de programmation, Logique…

Avant les années 60, les données sont stockées sous forme de fichiers. Ces SGF (Systèmes de Gestion de Fichiers) partagés comportent

- des fonctions de base (création, destruction, allocation de mémoire, localisation),
- des fonctions de contrôle (partage, résistance aux pannes, sécurité et confidentialité des données.

IDS Integrated Data Store
IMS Information Management System
CODASYL Conférence on Data System Language



Durant les années 60, naissance de la première génération de SGBD

- séparation de la description des données et des programmes d'applications,
- avènement des langages d'accès navigationnels.

IDS (1962): Dans la division manufacturing de General Electric, Charles Bachman développe sur GE 200 un système de gestion des nomenclatures qui deviendra la base de donnée hiérarchique I.D.S (IDS = Integrated Data Store). Ce système fut utilisé dans le programme Apollo.

IMS : Information Management System, modèle hiérarchique d'IBM.

Durant les années 70 naissance de la deuxième génération de SGBD

- modèle Relationnel, suite aux travaux de Codd,
- modèle Entité-Relation (ou Entité-Association), suite aux travaux de Chen.

CODASYL : Conference On DAta SYstem Language, Organisme américain de codification des systèmes de bases de **données.** Il a publié en 1959 les spécifications du **langage** COBOL ; ses travaux entre 1974 et 1981 ont ensuite produit le **modèle navigationnel** de **SGBD.** Il a défini en 1970 les normes de **standardisation** des **SGBD,** uniquement mis en oeuvre sur grands systèmes à cette époque. Le **langage** de manipulation de **données** était alors le **COBOL.**

Edgar Codd (1924-2003)

Scientifique anglais né à Portland, il étudie les mathématiques et la chimie à Oxford, avant d'obtenir son doctorat d'informatique à l'Université du Michigan. Employé chez IBM, il met au point entre les années 1960 et 1970 ses théories sur les **modèles relationnels de données**. Des concepts qui donneront naissance peu de temps après aux produits Oracle et DB2. Malgré le succès du langage SQL qui a suivi, Edgar F. Codd dénoncera cet outil qu'il considère comme une interprétation incorrecte de ses théories. Il quitte IBM pour fonder sa société de conseil et reçoit, en 1981, le prix Turing Award en récompense de ses travaux. Il est décédé à l'âge de 79 ans le 18 avril 2003.

(ref. **http://solutions.journaldunet.com**)

System/R (1974) : modèle relationnel d'IBM

INGRES : première base de données relationnelle.

Socrate : modèle réseau de CII Bull.

CHEN (1983) : dans l'article « Methodology and tools for database design », donne les bases de la modélisation Entité-Association

A partir des années 1980 et jusqu'à nos jours :

- troisième génération : Modèle Orienté Objet
- base de Données Déductives.

DB2 : base de données relationnelle IBM

1.5 Les SGBD actuels

Actuellement les 2 leaders incontestés du marché sont :

- ORACLE de l'éditeur éponyme,
- DB2 de IBM.

L'outsider est SQL Server de Microsoft, et il existe aussi :

- INFORMIX de l'éditeur éponyme,
- INGRES de COMPUTER ASSOCIATE
- POSTGRES aujourd'hui logiciel libre.

2 Les modèles

Pour concevoir une base de données, il faut en premier lieu la modéliser. Pour cela, à partir d'une problématique réelle, il va falloir extraire des « concepts » génériques et les relier entre eux.

La représentation de cette modélisation est liée au modèle utilisé. Dans notre exposé, nous allons dans un premier lieu présenté l'évolution historique de la modélisation des bases de données et nous nous focaliserons sur le modèle entité-association. Actuellement, le modèle UML (Unified Modelling Langage, James Rumbaugh, Ivar Jacobson, Grady Booch, Campus Press UML 2.0, 2005) tend à devenir le standard.

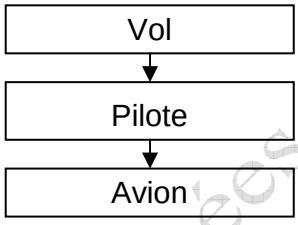
Dans la modélisation, on se focalise sur la sémantique des données que l'on manipule et des liens entre données.

2.1 Evolution historique

2.2.1 Le Modèle hiérarchique.

L'un des premiers modèles hiérarchiques fut le modèle IMS d'IBM. Dans ce modèle, les concepts sont organisés sous forme d'arbres, avec un père, des fils, des sous-fils… La base de données est constituée d'une **collection d'arbres** appelée forêt.

Exemple :



Une modélisation en arbre est simple à représenter mais peut être difficilement transposée dans des cas réels.

Dans notre schéma, pour un vol donné, nous avons plusieurs pilotes possibles, mais pas plusieurs vols pour un même pilote. Si nous voulons représenter plusieurs vols pour un même pilote, il va falloir modifier le schéma en indiquant Pilote comme père (mais nous ne pouvons pas représenter dans ca cas un vol donné, nous avons plusieurs pilotes possibles).

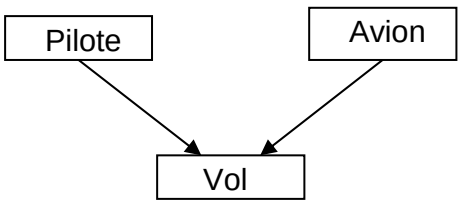
De même, si le même concept se trouve dans deux schémas, il doit y avoir duplication des données.

Les langages de définition et de manipulation des Données sont intégrés dans un langage de programmation hôte (par exemple PL/I et COBOL). Le LMD permet de parcourir les arbres d'articles, de mémoriser des positions dans ces arbres, d'insérer de supprimer ou de modifier des articles dans ces arbres.

2.2.2 Le modèle Réseau.

Ce modèle a été proposé par le groupe DBTG (Data Base Task Group) et utilisé par les produits IDS (CII Bull) et IDMS. Les liens entre articles appelés SET CODASYL et matérialisent une association entre deux types d'articles distincts. Le schéma est présenté sous la forme d'un graphe, d'un réseau connectant des objets modélisés sous forme d'**articles** comportant un ou plusieurs **champs**.

Exemple :



Dans ce modèle aussi, les langages de définition et de manipulation des Données sont intégrés dans un langage de programmation hôte. De même, à cette époque il y avait une forte interaction entre l'organisation physique des données et la base de données.

2.3 Le modèle Entité-Association

Jusque dans les années 2000, la méthode Merise (Tardieu 1983) était la méthode standard de modélisation des systèmes d'information en utilisant le modèle Entité-Association définit par CHEN (1976). Ces travaux ont été enrichis par de nombreux auteurs pour obtenir des diagrammes Entité-Association étendus.

Cette modélisation d'une représentation extrêmement simple, se base sur les deux notions entité et l'association (relation).

2.3.1 Définitions

Une **entité** est définie comme tout concept concret ou ayant les mêmes caractéristiques.

Exemple : Client, Etudiant, Voiture, Voyage, Catégorie client

Une entité comprend des **propriétés** (attributs) qui la déterminent.
Exemple : entité Client, propriétés : nom, prénom, adresse…

Toutes les entités sont distinctes et un attribut ne peut pas appartenir à deux entités.

Les entités sont reliées entre elles par des relations appelées **association**. Une association peut contenir des attributs.

Exemple :

- entre les entités Personne et Chien, l'association Appartenir
- entre les entités Client et Produit, l'association Acheter avec un attribut Quantité.

Une propriété particulière est appelée **indentifiant** (identificateur), cette propriété détermine de manière unique l'entité.

Exemple : entité Personne , identificateur numéro de sécurité sociale

Le nombre de classes d'entités associées est appelé **dimension** ou **arité** ou **degré** de l'association.

Exemple : association Appartenir,

- pour une personne donnée, elle peut posséder au minimum 0 chien et au maximum plusieurs chiens
- pour un chien donné, il peut appartenir au minimum à 0 personne (chien abandonné) et au maximum un propriétaire (dans notre exemple).

Une association peut relier une, deux ou plusieurs entités, on parle de type d'association binaire, ternaire, ... n-aire.

Exemple :

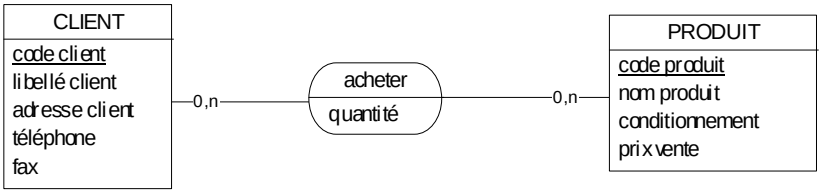
- association Appartenir : binaire,
- association Fils, unaire sur l'association Personne,
- association Vol, ternaire avec les entités Pilote, Avion et Date.

2.3.2 Représentation graphique

Diverses représentations graphiques sont utilisées, de manière générale un type d'entités est représenté par un rectangle, un type d'association par un losange ou un ovale.

Afin d'affiner la modélisation il est convenu de préciser le nombre d'associations dans lesquelles une entité peut apparaître, on attache à chaque rôle une contrainte de **cardinalité** ou **connectivité** (c'est la dimension de l'association). Celle-ci s'exprime sous la forme de deux entiers (i, j), (cardinalité minimale, cardinalité maximale) qui spécifient que toute entité pouvant jouer ce rôle devra le jouer un nombre de fois compris entre i et j. Les valeurs les plus fréquentes pour i sont 0 et 1, et pour j 1,n ou *.

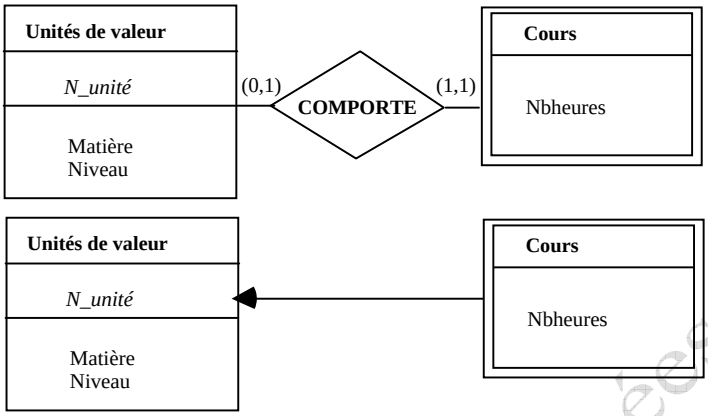
Exemple :



2.3.3 Type d'entités faibles et Types d'associations faibles.

Un type d'entités (ou un type d'associations) est dit **faible** quand l'existence d'une de ses instances est subordonnée à l'existence d'une instance d'un autre type d'entités (ou d'un autre type d'associations).

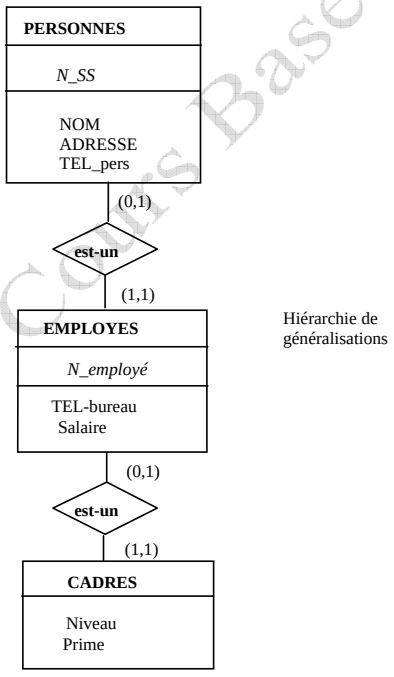
Exemple :



2.3.4 Spécialisation/Généralisation.

Un type d'entités A est une spécialisation (ou sous-type) d'un type d'entités B (appelé sur-type) si chaque entité de A est une entité de B et si chaque entité de B est associée au plus à une entité de A. On dit que A est **spécifique** ou spécialisé et que B est **générique**.

Exemple :

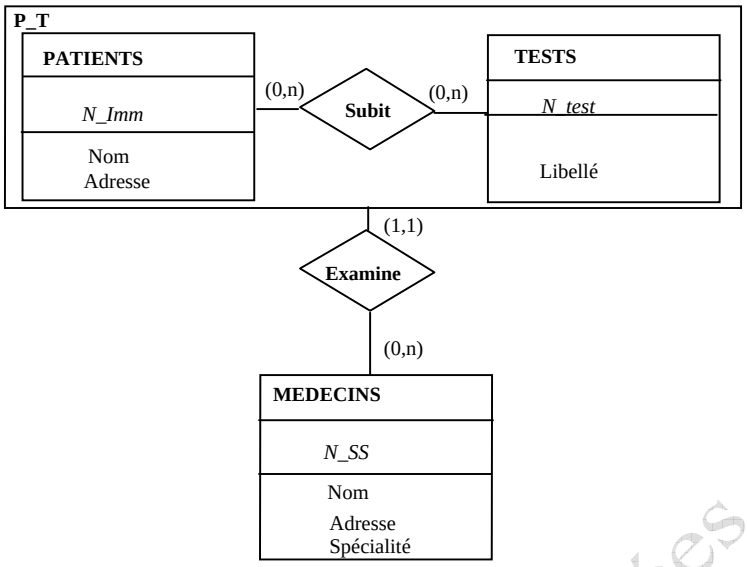


Hiérarchie de généralisations

2.3.5 Agrégation.

L'agrégation est le processus d'abstraction par lequel un type d'association entre deux ou plusieurs types d'entités est lui-même considéré comme un nouveau type d'entité sur lequel on peut définir d'autres types d'associations. L'intérêt d'un tel concept est de décrire des « entités » complexes.

Exemple :



En conclusion, le modèle E-A (plus exactement les modélisations Entités-Associations) constitue une approche sémantique « universellement » admise, qui joue le plus souvent le rôle d'intermédiaire entre les spécifications en langue naturelle et les SGBD commerciaux.

2.4 Les problèmes liés à la modélisation

2.4.1 Les problèmes

Si nous prenons le problème de la gestion d'un club associatif. Pour modéliser les Membres, nous allons créer une entité ayant comme propriétés : nom prénom, adresse, code postal, ville.
Exemple :

nom	prénom	adresse	Code postal	ville
Dupont	Pierre	12 rue du port	34210	FRONTIGNAN
Durand	Paul	Imp des roses	34200	SETE
Pons	Eric	Av Alfred	34210	FRONTIGNAN
Hartman	Hans	Place Salinger	66000	PERPIGNAN

Problème de duplication

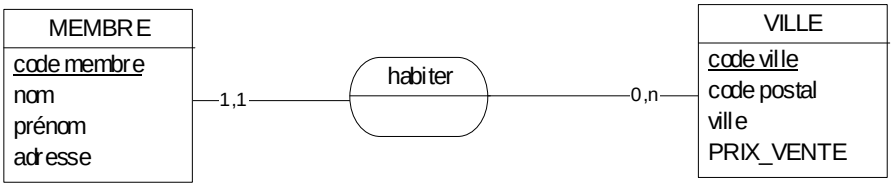
Dans ce cas, il est possible de créer plusieurs fois le même membre en mettant plusieurs lignes du même nom avec des adresses différences. Pour résoudre ce problème, il faut créer un identificateur comme par exemple code membre.

Problème de modification
Si nous nous apercevons par exemple que nous avons fait une erreur sur le code postal de Frontignan que c'est 34110 au lieu de 34210. Il va falloir faire attention de le modifier dans toutes les lignes. Si nous oublions une ligne, la base devient incohérente.

Problème de destruction
Si l'on supprime le membre Durand, nous perdons de même les informations sur SETE (dans notre exemple son code postal).

2.4.2 La solution

La solution consiste à créer 2 entités indépendantes, l'une comprenant les membres, la seconde comprenant les villes.



Nous voyons que le modèle entité-association est très simple car il suffit de mettre en place les entités avec leurs propriétés et les associations avec leurs cardinalités. Le seul inconvénient de cette méthode est qu'elle n'est pas déterministe, c'est-à-dire que nous ne pouvons pas savoir si la solution trouvée est la bonne.

2.5 Cas d'étude

2.5.1 Le cahier des charges

Pour pouvoir réaliser une modélisation, il faut partir d'un cahier des charges. Ce cahier des charges reprend la problématique du client.

Exemple :

L'association « les marcheurs du Canigou » souhaiteraient s'informatiser. Ils veulent connaitre leurs adhérents et faire un historique des marches. Pour leur adhérents ils veulent connaitre les informations nominatives et s'ils ont payé leur abonnement (20 euros annuel moitié prix pour les mineurs). Pour les marches, ils veulent connaitre le lieu, la date et la liste des participants.

2.5.2 Le dictionnaire de données

Une fois que nous avons le cahier des charges, nous allons chercher les informations nécessaires pour résoudre ce problème.

Une information doit être :

- élémentaire : on ne doit pas pouvoir la subdivisée

Exemple : adresse qui doit être décomposée en adresse1, adresse 2, code postal, ville

- non calculable, si elle est calculable, nous ne gardons que les informations qui permettent de faire le calcul

Exemple :

- informations : montant HT, montant TTC, taux TVA : nous ne conservons que 2 informations sur 3
- information : âge, il faut prendre date de naissance
- instanciable : on doit pouvoir donner une valeur

Exemple : nom client : EDF
Client : on ne peut pas donner de valeur, donc ce n'est pas une information.

Correction du cas d'étude

Nom adhérent
Prénom adhérent
Date de naissance adhérent
Cotisation annuelle
Indicateur de paiement par an
Lieu de la marche,
Date de la marche

2.5.3 La matrice de dépendances fonctionnelles

Une fois que nous avons trouvé le dictionnaire de données, nous allons essayer de trouver les concepts qui se rapportent aux informations que nous avons découvertes.

Pour pouvoir représenter la matrice de dépendances fonctionnelles, nous allons indiquer dans la première colonne les informations du dictionnaire de données et en colonne les concepts associés.

Exemple :

1/ Recherche des concepts

Nom adhérent	concept ADHERENT
Prénom adhérent	concept ADHERENT
Date de naissance adhérent	concept ADHERENT
Cotisation annuelle	concept ANNEE
Lieu de la marche	concept LIEU
Date de la marche	concept MARCHE

Remarque : Nous n'avons pas affecté à l'information « indicateur de paiement par an » de concept car cette information dépend de l'année et de l'adhérent.

2/ Matrice de dépendances fonctionnelles

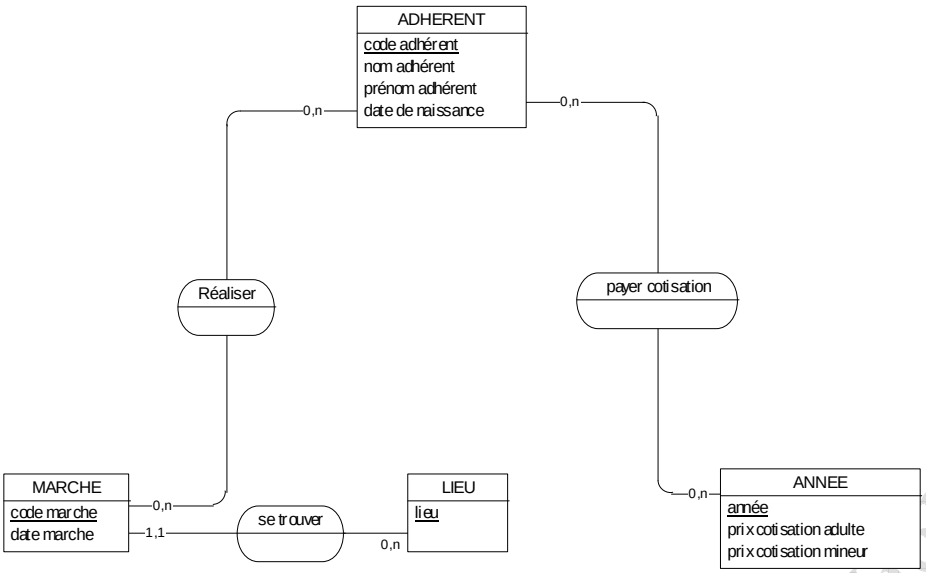
	ADHERENT	ANNEE	LIEU	MARCHE
Nom adhérent	X			
Prénom adhérent	X			
Date de naissance adhérent	X			
Cotisation annuelle		X		
Lieu de la marche			X	
Date de la marche				X

2.5.4 Le modèle entité-association

Dans la partie précédente, nous avons trouvé les entités (ADHERENT, ANNEE, MARCHE, LIEU) avec leurs attributs. Nous allons maintenant :

- rajouter des identificateurs,
- rajouter des attributs évidents si nécessaire,
- rechercher les associations,
- pour chaque association ajouter les cardinalités.

Exemple :



3 Le modèle relationnel

Suite aux travaux de Franck Codd, le modèle relationnel est le standard mondial dans le domaine de base de données. Ce succès provient des raisons suivantes :

- le concept utilisé unique est celui de relation basé sur une modélisation mathématique,
- à partir d'opérations simples de normalisation, il est possible de créer des schémas,
- il existe des SGBD qui facilitent l'utilisation des bases de données,
- il y a une indépence entre le modèle logique et le modèle physique
- un langage de requêtes standardisé (SQL) existe.

3.1 Définitions

Le modèle relationnel n'utilise que le concept mathématique de relation. Une relation dans la théorie des ensembles est un sous-ensemble du produit cartésien d'une liste de domaines. Un domaine est un ensemble de valeurs.

Exemples :

- un ensemble d'entier,
- un ensemble de chaînes de caractères,
- {rouge,vert,bleu}

Le **produit cartésien** des domaines D1, D2,..., Dn que l'on écrit

D1 x D2 x....x Dn

est l'ensemble des n-uplets ou tuples (v1, v2,..., vn) tels que vi appartient à Di pour i=1, 2,..., n
Une **relation** sur un ensemble de domaines D1, D2,..., Dn est un sous-ensemble du produit cartésien de ces domaines. On appelle n l'**arité** de la relation, et **n-uplets** (ou tuples) les éléments de la relation. Le nombre de n-uplets d'une relation est appelé son **cardinal** (ou sa cardinalité).

Il est courant de représenter une relation sous forme d'une table, où les lignes correspondent aux tuples et les colonnes aux composantes.

Exemple :

Nom personne	Date de naissance	N° immatriculation	Type voiture	Km
Durand	01/12/1956	1526 VC 34	Mégane	12 500
Hans	23/05/1985	321 WW 66	Clio	800
Herbert	06/05/1998	645 ADC 34	Escort	15 000
Piccoli	09/11/1938	28 KJ 71	2CV	35 000

On appelle **Schéma de la relation** le nom de la relation suivi de la liste de ses attributs.

Pour reprendre l'exemple ci-dessus

POSSEDE(Nom personne: caractère(20), Date de naissance : Date, N° immatriculation : caractère(10), Type de voiture : caractère(10), Km : entier)

La notion de **clé** (rencontrée dans le modèle E-A comme identificateur) s'appliquent à la structure relationnelle.

ex : FOURNISSEURS(Nom : caractère(20), Adresse : caractère(30); clé **NOM**) ;

POSSEDE(Nom personne: caractère(20), Date de naissance : Date, N° immatriculation : caractère(10), Type de voiture : caractère(10), Km : entier; clé **N° immatriculation**)

3.2 Notion de schéma de base de données relationnelle

Un **schéma de base de données relationnelle** est constitué d'un ensemble de **schémas de relations** définis sur des ensembles d'attributs et soumis à un certain nombre de **contraintes** dites contraintes d'intégrité qui expriment la sémantique de la représentation.

Un schéma de relation est donc constitué de :

- un nom
- la liste des attributs
- la signification (prédicat)
- la liste des contraintes imposées

3.3 Les contraintes.

Les contraintes d'intégrité exprimables au niveau schéma peuvent être variées :

- contraintes de domaine,
- dépendances fonctionnelles,
- contraintes référentielles ou contraintes d'inclusion,
- contraintes de dépendances fonctionnelles.

3.3.1 Contraintes de domaines.

Définition extensive ou intensive du domaine d'un attribut.

Exemples :

- l'attribut Nom Fournisseur est contraint à être une chaîne de caractères de longueur 20,
- l'attribut Couleur de la relation Voiture a ses valeurs dans l'ensemble {rouge, vert, bleu, noir, blanc}
- L'existence de la valeur d'un attribut peut être liée à la valeur d'un autre attribut.

Exemple :

- l'attribut Nommarital de la relation Personne ne prend de valeur que si la valeur de l'attribut Sexe est « féminin ».
- Règles de calcul indiquant comment la valeur d'un attribut est déduite de la valeur d'un ou plusieurs attributs.

Exemple :

- l'attribut date de naissance est contraint a avoir la même valeur pour l'année que les caractères 2 et 3 de l'attribut N_SS contraint à être une chaîne de caractères de longueur 13.

3.3.2 Dépendances fonctionnelles

Définition

Soit $\mathcal{R}$(A1, A2,..., An) un schéma de relation et G et D deux sous-ensembles de {A1, A2,..., An}, on dit que **G détermine D** ou que **D dépend fonctionnellement de G** et on note **G ->D** si pour toute relation R de $\mathcal{R}$ acceptable tous les tuples u,v de R qui ont même composante dans G ont aussi même composante dans D

Exemple :
A l'ensemble des phrases suivantes :

- une voiture est identifiée par un numéro d'immatriculation N_imm,
- une voiture a une couleur,
- à une voiture correspond un type,
- à un type de voiture correspond une puissance,

on peut associer l'ensemble de DF suivant :

{N_imm -> Type, N_imm -> Couleur, Type -> Puissance}

3.4 Normalisation

3.4.1 1ère FN

Un schéma relationnel $\mathcal{R}$ est en première forme normale si tous les attributs de la relation U sont atomiques (ou monovalués).

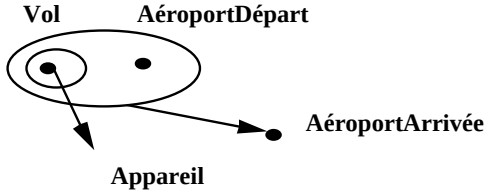
3.4.2 2ème FN

Un schéma relationnel $\mathcal{R}$ défini sur U et sur un ensemble de dépendances fonctionnelles F est en deuxième forme normale si et seulement si :

- il est en 1FN et
- tout attribut n'appartenant pas à une clé dépend directement d'une clé.

Exemple :

Soit $\mathcal{R}$(Appareil,Vol,AéroportDépart,AéroportArrivée)
avec F={ Vol ->Appareil, Vol AéroportdeDépart ->AéroportArrivée}



La clé de $\mathcal{R}$ est Vol AéroportdeDépart

Vol, AéroportdeDépart appartiennent à la clé, AéroportArrivée dépend directement de la clé, mais Appareil dépend d'une partie de la clé et non de la clé entière donc le schéma n'est pas en 2FN.

Pour passer de la première forme normale à la deuxième, il faut diviser chaque table ne satisfaisant pas les critères en deux tables distinctes.
Pour diviser une table en deux, il faut
- Créer une nouvelle table ayant pour clé la partie de la clé primaire dont dépend le ou les attributs, ainsi que ces attributs eux-mêmes.
- Éliminer ces attributs (ceux qui ne font pas partie de la clé) de la table originale.

3.4.3 3ème FN

Un schéma relationnel R défini sur U et sur un ensemble de dépendances fonctionnelles F est en 3FN si :
- il est en 2FN
- aucun attribut ne faisant pas partie de la clé ne dépend d'un autre attribut ne faisant pas partie non plus de la clé

Pour passer de 2FN à 3FN, il faut :
- diviser chaque table ne satisfaisant pas ce critère en deux tables. La nouvelle table aura comme clé l'attribut dont provient la dépendance et comme attributs, ceux qui en dépendent.
- éliminer les attributs dépendants de la table originale. La clé de la nouvelle table demeure dans l'ancienne en tant que clé étrangère.

Dans le cas des voitures usagées, toutes les voitures de la même année sont vendues au même prix (Année->Prix).
VOITURE (NoStock, Marque, Modèle, Année, Couleur, Prix, TélFabricant)

Il y a donc une DF entre "Année" et "Prix", ce qui signifie que cette table n'est pas en 3FN. Il faut donc décomposer cette table en deux.

VOITURE (NoStock, Marque, Modèle, Année, Couleur, TélFabricant) PRIXVENTE (Année, Prix)

3.4.4 Forme Normale de Boyce-Codd (FNBC)

Cette normalisation a été réalisée par R.F. Boyce et E.F. Codd, qui ont constaté que la 3FN pouvait comporter certaines anomalies. Un modèle relationnel en FNBC est considéré comme étant de qualité suffisante pour une l'implantation.

Un schéma relationnel R défini sur U et sur un ensemble de dépendances fonctionnelles F est en 3FN si :
- il est en 3FN
- aucun attribut faisant partie de la clé ne dépend d'un attribut ne faisant pas partie de la clé primaire

Pour passer de la 3FN à la FNBC, il faut :
- Diviser chaque table ne satisfaisant pas au critère ci-haut en deux tables. La nouvelle table aura comme clé l'attribut ordinaire dont provient la dépendance et comme attributs la partie de la clé qui en dépend.
- Remplacer la partie de la clé concernée par l'attribut ordinaire (qui est devenu la clé de l'autre table).

Exemple : CLAVIER (MarqueClavier, NombreTouches, TypeClavier)

Supposons qu'il y a une DF entre :
type de clavier -> le nombre de touches.
Il faut donc diviser en deux tables de la façon suivante :

CLAVIER (MarqueClavier, TypeClavier)
TOUCHES (TypeClavier, NombreTouches)

Par exemple :
Université(étudiant, matière, enseignant, note)
avec les DF
étudiant, matière →Université enseignant, note et
enseignant →Université matière

n'est pas en BCNF alors que :
Pers(nom, prénom, âge, nombreEnfants)
avec la DF
nom, prénom →Pers âge, nombreEnfants est en BCNF.

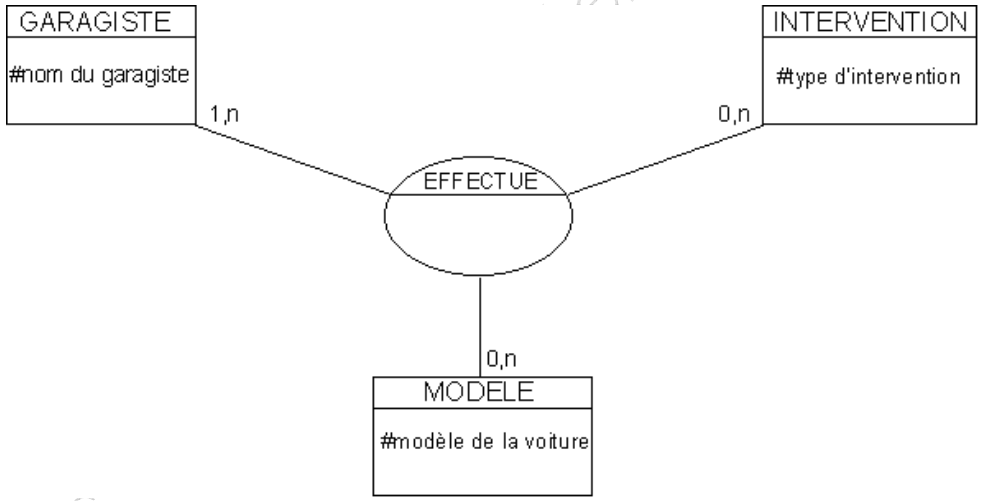
3.4.5 4ème FN

Un schéma relationnel R défini sur U et sur un ensemble de dépendances fonctionnelles F est en 4FN si :
- il est en BCFN
- il ne possède pas de Dépendance Multivaluée ou bien, X ->Y étant la Dépendance Multivaluée , il doit exister une propriété A telle que X -> A soit vérifiée.

DÉPENDANCE MULTIVALUÉE (DM)
étant donné une association A(P) et X, Y des ensembles de propriétés inclus dans P, il existe une DM entre les ensembles de propriétés X, Y lorsque :
<x,y1,z1>∈ A et <x,y2,z2>∈ A ⇒ <x,y1,z2>∈ A et <x,y2,z1>∈ A

Cette dépendance est notée X ->->Y.

Exemple :



Les données de l'association EFFECTUE peuvent être :

garagiste	intervention	modèle
Martin	électricité	Citroën
Piquard	Carrosserie	Ford
Martin	Mécanique	Renault
Piquard	Carrosserie	Fiat
Tussier	Dépannage	Peugeot
Piquard	Alarme	Fiat
Martin	électricité	Renault
Piquard	Alarme	Ford

Martin	Mécanique	Citroën

Il existe deux DM :
garagiste ->->intervention
et garagiste ->->modèle
s'expliquant par le fait qu'un garagiste qui effectue un ensemble de types d'intervention pour un ensemble de modèles de voiture, est capable d'effectuer chacun de ces types d'intervention sur chacun de ces modèles de voiture.

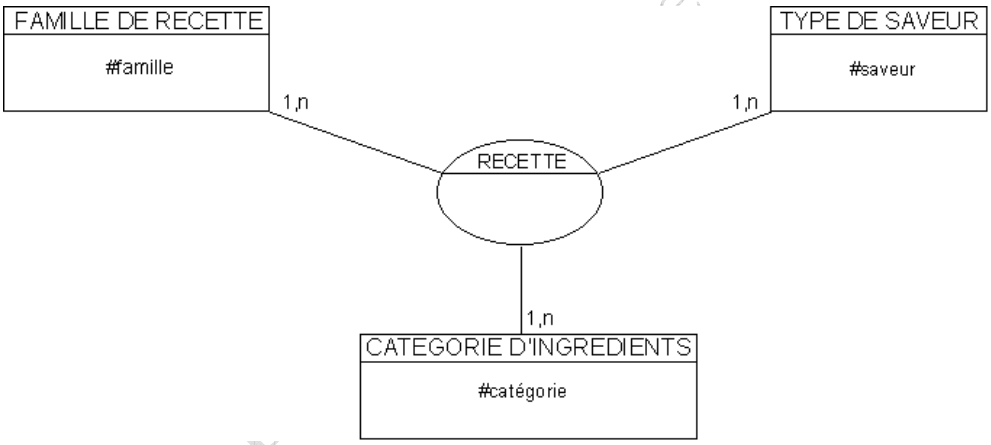
3.4.6 5ème FN

Un schéma relationnel R défini sur U et sur un ensemble de dépendances fonctionnelles F est en 5FN si :
- il est en 4FN
- elle ne possède pas de Dépendance de Jointures ou bien, *[X1]…[Xn] étant la Dépendance de Jointures, il doit exister une propriété A telle que X $\rightarrow$ A soit vérifiée

DÉPENDANCE DE JOINTURE (DJ)
étant donnée une association A(P) et X1, X2, ..., Xn des ensembles de propriétés dont l'union est P' tels que :
$P' \subset P$, $P' = U^{x}_{i-1} X_i$ et $[X_i \cap X_{i+1} \neq \varnothing]^{x-1}_{i-1}$

il existe une DJ lorsque : R = R[X1] * R[X2] * ... * R[Xn-1] * R[Xn].
Cette dépendance est notée *[X1][X2]...[Xn-1][Xn].
Si P-P' $\neq \varnothing$, la dépendance est dite partielle.
Si P = P', la dépendance est dite totale.

Exemple :



Les données de l'association RECETTE peuvent être :

famille	saveur	catégorie
Viandes	Sucré	Fruits
Viandes	Sucré	Légumes
Desserts	Sucré	Fruits
Desserts	Acide	Fruits

Ce cas correspond à X1 = (famille, saveur), X2 = (saveur, catégorie) et X3 = (catégorie, famille).

La projection RECETTE[X1] donne :

famille	saveur
Viandes	Sucré
Desserts	Sucré
Desserts	Acide

La projection RECETTE[X2] donne :

saveur	catégorie
Sucré	Fruits
Sucré	Légumes
Acide	Fruits

La jointure RECETTE[X1] * RECETTE[X2] donne :

famille	saveur	catégorie
Viandes	Sucré	Fruits
Viandes	Sucré	Légumes
Desserts	Sucré	Fruits
Desserts	Sucré	Légumes
Desserts	Acide	Fruits

La projection RECETTE[X3] donne :

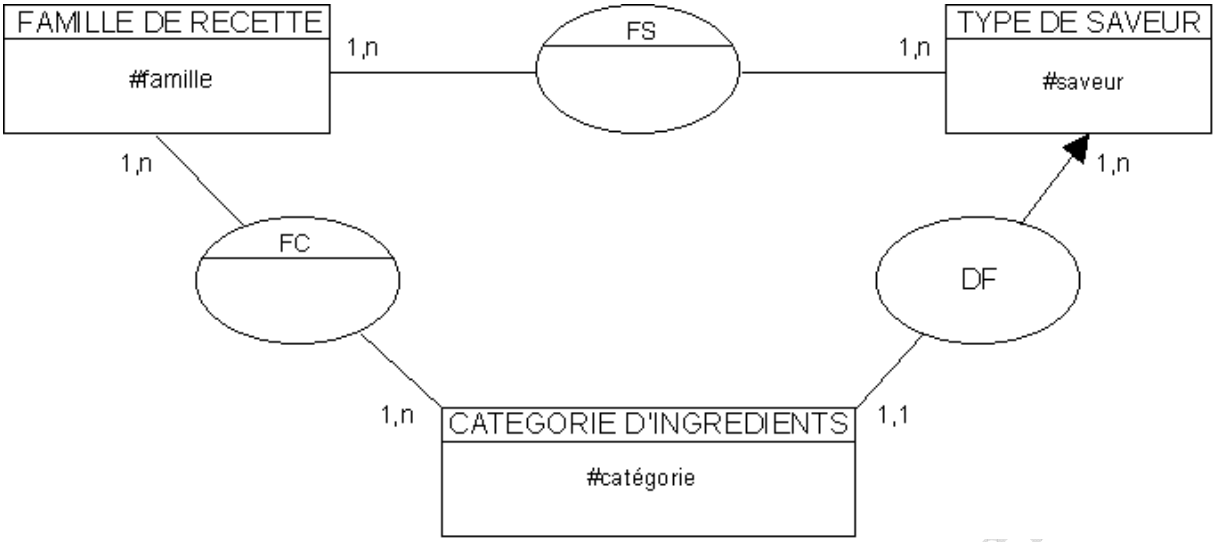
catégorie	famille
Fruits	Viandes
Légumes	Viandes
Fruits	Desserts

La jointure RECETTE[X1] * RECETTE[X2] * RECETTE[X3] redonne l'association RECETTE.

Cette dépendance indique qu'il ne suffit pas qu'une famille de recettes ait une saveur dominante et que cette saveur corresponde à une catégorie d'ingrédients, pour qu'une recette à base de cet ingrédient et de saveur spécifiée, soit fabriquée.
Un dessert possède une saveur sucrée, une saveur sucrée peut correspondre à un légume mais un dessert sucré à base de légumes n'est pas fabriqué car il n'y a pas de légumes dans un dessert.

Exemple :
L'association RECETTE est en 4FN mais pas en 5FN car il existe une Dépendance de Jointures et, puisque cette association n'est pas porteuse d'une propriété, il n'existe pas de DF.

Ce modèle présente des difficultés de mise à jour. Si l'occurrence <« Viandes », « Acide », « Vins »> est ajoutée, il faut également ajouter l'occurrence <« Viandes », « Acide », « Fruits »>. Pour que l'association RECETTE soit en 5FN, il faut la décomposer :



3.5 Du MCD au MLD

Dans la méthode Merise, du professeur Jacques Tardieu (1983), le modèle de données est décomposé en trois sous-modèles :

- MCD : Modèle Conceptuel de Données (qui correspond au modèle Entité-Association),
- MLD : Modèle Logique de Données (qui correspond au schéma de la base de données sans contraintes physiques)
- MPD : Modèle Physique de Données (qui correspond au schéma de la base de données avec les contraintes physiques)

Pour passer du MCD au MLD, nous avons 3 règles, appelées « règles de dérivation » que nous devons réaliser chronologiquement.

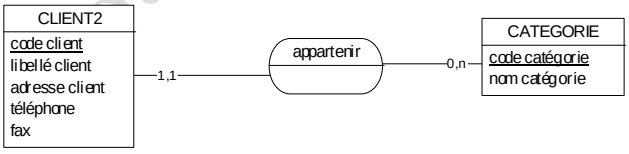
Règle 1

Toute Entité devient une table. Les propriétés de l'Entité sont les propriétés de la Table, l'identificateur de l'Entité est l'identificateur de la Table (clé).

Règle 2

Dans le cas d'une association de cardinalités maximales d'un côté égale à 1(-, 1) et de l'autre côté égale à n (-, n) on ajoute dans la table de cardinalité du côté 1, une propriété de même type que l'identificateur de la table de cardinalité maximale côté n.

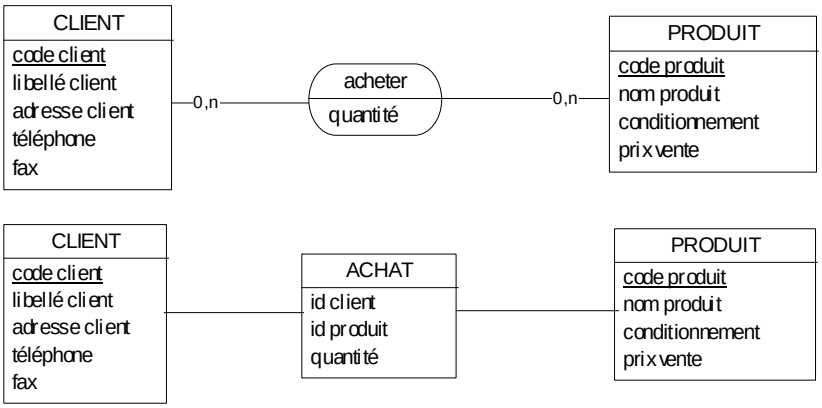
Exemple :



Règle 3

Dans le cas d'une cardinalité maximale n de chaque côté (-, n), on crée une table dans laquelle on ajoute autant de propriété de même type que l'identificateur de chaque table associé et les propriétés de l'association si nécessaire

Exemple :

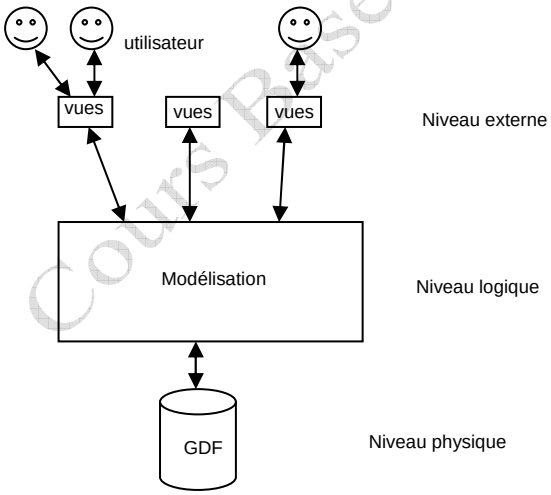


3.6 Les SGBD

3.6.1 Les niveaux d'abstraction

L'architecture de référence d'un SGBD est basée sur 3 niveaux :

- le niveau physique qui gère directement les données sur les supports permanents (gestion sur la mémoire physique (fichiers) des données, gestion de la concurrence d'accès; reprise sur pannes…),
- le niveau logique : représenté par le modèle conceptuel qui représente la sémantique de la BD, qui permet de définir la structure de données (Langage de Description de Données et Langage de Manipulation de Données)
- le niveau externe : qui représente la vision des utilisateurs, cette vision peut être multiple, elle correspond à l'interface avec l'utilisateur.



L'avantage de cette représentation est qu'il y a une indépendance entre le niveau physique et le niveau logique. Il est possible de modifier l'organisation physique des données sans modifier les applications.

Exemple : modification d'un index

De même, il est possible de modifier le schéma conceptuel sans modifier les applications.

Exemple : ajout d'attributs.

3.6.2 L'évolution des SGBD

(Ref **http://fadace.developpez.com/sgbdcmp/story/**)

1970 Edgar F. Codd rédige l'article « A Relational Model of Data for Large Shared Data Banks » dans la revue Communications of the ACM (Association for Computing Machinery)
IBM crée le SEQUEL
1973 Suite au SystemR d'**IBM**, L'université de Berckley crée le projet RDBMS, duquel découleront **Sybase**, **Informix**, NonStop SQL.
Prototype d'**Ingres**
1978 **IBM** commercialise son System Relational (System R)
Oracle V1
1982 **IBM** sort SQL/DS (DB2 sur mainframe, ancien System R)
Commercialisation de **Ingres**
1984 **Oracle** V4 (Read consistancy), portage sur IBM/VM, MVS, PC
Fondation de **Sybase**,
Lancement du projet dBase pour Macinthosh
Laurent Ribardière développe **4D**
1985 **Oracle** v.5.0 (client serveur)
Informix-SQL v.1.1

Création du projet **PostIngres**

1986 SQL86 édité par ANSI
1987 **Sybase** SQL Server sur Unix
Ratification par ISO de SQL86 sous SQL87 (SQL1)
Lancement de dBase pour Macintosh, totalement réécrit
Sortie d'**Interbase** par Ashton-Tate
4D v.3
1988 **Oracle** v.6. (verrouillage niveau ligne, Sauvegarde/restauration, PL/SQL)
Ashton-Tate sort **dBase IV**

1989 SQL89 : addentum intégrité référentielle
Postgres v.1

InformixOnline v.4

1990 Informix Online v.5
Postgres v.2

1992 **Oracle** v.7. (contraintes, procédures, déclencheurs)
Microsoft rachète **FoxPro**, un clône de dBase et lance **MS-Access**

1993 **Postgres** v.4, fin du projet de Berckley
Sybase SQL Server 4.9.2

Version client-serveur de 4D

1994 **Microsoft** SQL Server 4.21 pour Windows NT
1995 Ulf Michael Widenius (Monty), David Axmark et Allan Larsson fondent **MySQL** AB
1996 **Informix** v.8 et Universal Server v.9 (incluant DataBlades)
Postgres95 devient **PostgreSql**

Sybase Adaptive Server Enterprise 11.5

Première version publique de **MySQL**

1997 **Oracle** v.8. (objet-relationnel, partitionnement, Java)
1998 **Sybase** ASE 11.9 (verrouillage ligne, page ou page de données à choix)

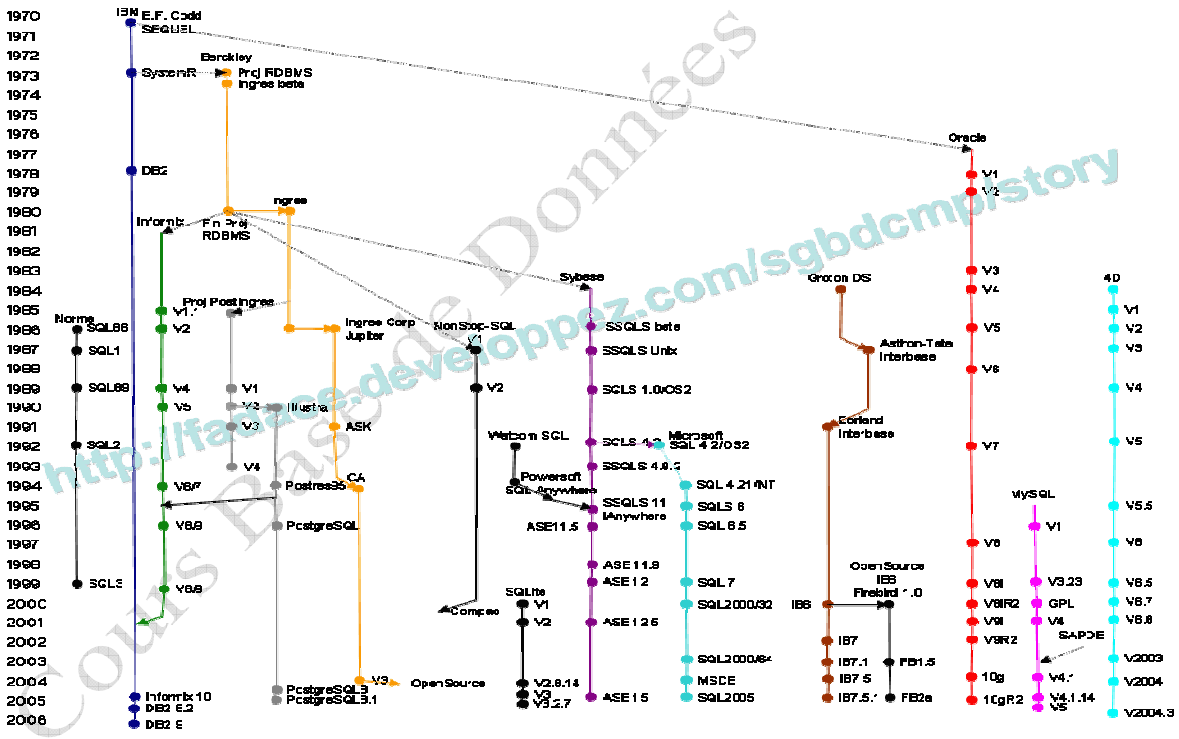
1999 **Oracle** 8i (SQLJ, Linux, XML)
MySQL 3.23 (réplication, FTS, transactionnel et intégrité référentielle avec les moteurs InnoDB et BDB)

2000 Oracle 8iR2
Microsoft SQL Server 2000 32-bit

2001 **Oracle** 9i (RAC)
IBM acquiert la division SGBDR **Informix**, **Informix** acquiert Ardent Software (DW, RedBrick)

4D v.6.8, portage de toute la gamme sous MacOS X

2002 Borland recommercialise **Interbase** 7
2003 SQL:2003 Introduction de fonctions pour la manipulation XML, »window functions »,
Microsoft SQL Server 2000 64-bit (nom de code Liberty)
4D 2003: Web services, XML, compilateur intégré
2004 CA passe **Ingres** v.3 en licence OpenSource
4D 2004
2005 IBM Informix IDS v.10
Sybase ASE 15
IBM acquiert Ascential Software
PostgreSql v.8 puis v.8.1
Microsot SQL Server 2005 (nom de code Yukon)
Oracle 10g R2, rachat d'Innobase
Interbase 7.5.1 (version commerciale); **Firebird** 2.0 Alpha
MySQL 4.1.14 & v.5.0 (procédures stockées, triggers, vues, data dictionary, moteur Archive)
SQLite 3.2.7



4 L'algèbre relationnelle

L'algèbre relationnelle est une algèbre mathématique à partir de laquelle est basé le modèle relationnel. Dans cette algèbre il existe des opérateurs ensemblistes et des opérateurs beaucoup plus spécifiques appelés opérateurs relationnels.

Cette algèbre s'applique sur une ou plusieurs relations et le résultat est lui-même une relation sur laquelle on peut appliquer des opérateurs.

4.1 Les opérateurs ensemblistes

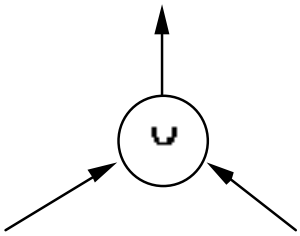
4.1.1 L'union

Cette opération porte sur deux relations de même schéma qui donne comme résultat une relation de même schéma ayant pour tuples ceux appartenant à l'une ou l'autre des relations ou aux deux relations.

Représentation :

UNION (Relation 1, Relation 2)
ou Relation 1 ∪ Relation 2

Représentation graphique



Relation 1 Relation 2

UNION

Exemple :

ENSEIGNANT(code , nom , prénom , ville de naissance)
ETUDIANT(code , nom , prénom , ville de naissance

ENSEIGNANT

Code	Nom	Prénom	Ville de naissance
001	DUPONT	Jean	Mireval
002	MULLER	Herbert	Berlin
003	PAOLI	Paul	Bastia

ETUDIANT

Code	Nom	Prénom	Ville de naissance
110	N GUYEN	Xan	Péquin
2052	ERZORG	Enzo	Turin
303	BERTHE	Pascal	Montpellier

ENSEIGNANT ∪ ETUDIANT

Code	Nom	Prénom	Ville de naissance
001	DUPONT	Jean	Mireval
002	MULLER	Herbert	Berlin
003	PAOLI	Paul	Bastia
110	N GUYEN	Xan	Péquin
2052	ERZORG	Enzo	Turin
303	BERTHE	Pascal	Montpellier

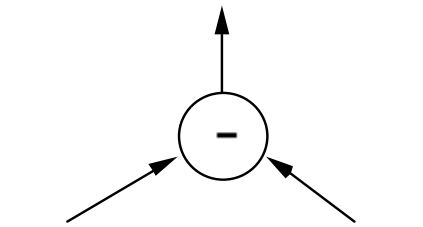
4.1.2 La différence

Cette opération porte sur deux relations de même schéma qui donne comme résultat une relation de même schéma ayant pour tuples ceux appartenant à la première relation et pas à la seconde.

Représentation :

DIFFERENCE (Relation 1, Relation 2)
ou Relation 1 - Relation 2

Représentation graphique



Relation 1 Relation 2

DIFFERENCE

Exemple :

R1

X	**Y**
u	v
w	x
y	z

R2

X	**Y**
t	u
w	x

R1-R2

X	**Y**
u	v
y	z

4.1.3 L'intersection

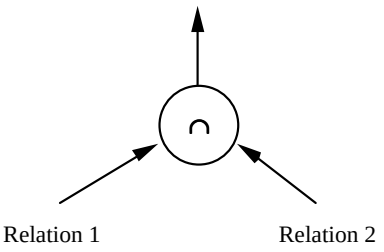
Cette opération porte sur deux relations de même schéma qui donne comme résultat une relation de même schéma ayant pour tuples ceux appartenant à la première relation et à la seconde.

Cette opération peut être obtenue à partir de l'opération Différence
R1 ∩ R2 = R1- (R1-R2) ou R2 -(R2-R1)

Représentation :

INTERSECTION(Relation 1,Relation 2)
ou Relation 1 ∩ Relation 2

Représentation graphique :



INTERSECTION

Exemple :

R1

X	**Y**
u	v
w	x
y	z

R2

X	**Y**
t	u
w	x

R1∩R2

X	**Y**
w	x

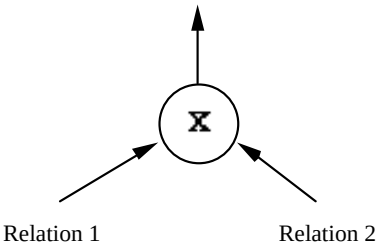
4.1.4 Le produit cartésien

Cette opération porte sur deux relations qui donne comme résultat une relation ayant comme schéma la juxtaposition de ceux des relations opérandes et pour tuples toutes les combinaisons des tuples de la première relation et de la seconde relation.

Représentation :

Relation 1 **X** Relation 2

Représentation graphique :



PRODUIT CARTESIEN

Exemple :

ENSEIGNANT

Code enseignant	Nom	Prénom	Ville de naissance
001	DUPONT	Jean	Mireval
002	MULLER	Herbert	Berlin
003	PAOLI	Paul	Bastia

VOITURE

N° immatriculation	Type	Couleur
74 GH 34	Visa	Blanche
1489 WWC 66	C5	Rouge

ENSEIGNANT X VOITURE

Code enseignar	Nom	Prénom	Ville de naissar	N° immatricula	Type	Couleur
001	DUPONT	Jean	Mireval	74 GH 34	Visa	Blanche
001	DUPONT	Jean	Mireval	1489 WWC 66	C5	Rouge
002	MULLER	Herbert	Berlin	74 GH 34	Visa	Blanche
002	MULLER	Herbert	Berlin	1489 WWC 66	C5	Rouge
003	PAOLI	Paul	Bastia	74 GH 34	Visa	Blanche
003	PAOLI	Paul	Bastia	1489 WWC 66	C5	Rouge

4.2 Les opérateurs relationnels

4.2.1 La sélection

Cette opération porte sur une relation qui donne comme résultat une relation de même schéma ayant pour tuples ceux vérifiant la condition précisée en opérande.

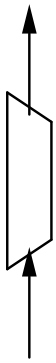
Représentation :

SELECTION (Relation 1, condition) avec condition du type : Attribut opérateur Valeur

ou Relation 1 **:** condition

ou $\sigma_{condition}$ (Relation 1)

Représentation graphique



Relation 1

SELECTION

Exemple :

ENSEIGNANT

Code enseignant	Nom	Prénom	Ville de naissance
001	DUPONT	Jean	Mireval
002	MULLER	Herbert	Berlin
003	PAOLI	Paul	Bastia

σville de naissance = « Berlin » ENSEIGNANT

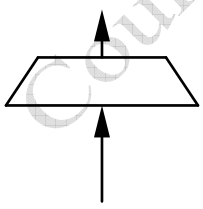
Code enseignant	Nom	Prénom	Ville de naissance
002	MULLER	Herbert	Berlin

4.2.2 La projection

Cette opération porte sur une relation qui donne comme résultat une relation dont le schéma est celui de la relation opérande à laquelle on a supprimé tous les attributs non mentionnés en opérande. On supprime ensuite tous les tuples doublons.

Représentation

PROJECTION (Relation1, attri i,....)
ou Relation1 **[**attri i,....**]**
ou $\Pi_{\text{attri i,..}}$ (Relation1)



Relation 1

PROJECTION

Exemple :

ENSEIGNANT

Code enseignant	Nom	Prénom	Ville de naissance
001	DUPONT	Jean	Mireval
002	MULLER	Herbert	Berlin
003	PAOLI	Paul	Bastia

Π code enseignant, nom(ENSEIGNANT)

Code enseignant	Nom
001	DUPONT
002	MULLER
003	PAOLI

4.2.3 La jointure

Cette opération consiste à rapprocher selon une condition les tuples de deux relations afin de former une nouvelle relation qui contient l'ensemble de tous les tuples obtenus en concaténant un tuple de la première relation et un tuple de la seconde relation vérifiant la condition de rapprochement. (celle-ci est du type Attribut1 opérateur Attribut 2).

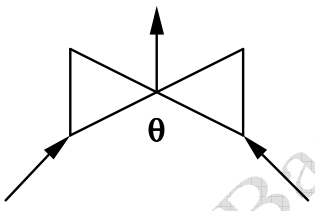
Représentation :

JOIN(Relation 1, Relation 2,Condition)
ou
R $\underset{\theta}{\bowtie}$ **S**

Avec R et S les 2 relations et θ la condition de jointure.

Remarque : si l'opérateur θ est = l'opération est appelée **équijointure**.

Représentation graphique :



JOINTURE

Remarques :

On utilisera le plus souvent la **jointure naturelle** (notée $\bowtie$) c'est à dire celle où la condition est l'égalité des attributs de **même nom** en ne conservant qu'une seule fois ces attributs dans la relation 3.

Exemple

ENSEIGNANT

Code enseignant	Nom	Prénom	Ville de naissance
001	DUPONT	Jean	Mireval
002	MULLER	Herbert	Berlin
003	PAOLI	Paul	Bastia

VOITURE

N° immatriculation	Type	Couleur	Code propriétaire
74 GH 34	Visa	Blanche	001
1489 WWC 66	C5	Rouge	003

JOIN(ENSEIGNANT, VOITURE, code enseignant = code propriétaire)

Code enseignant	Nom	Prénom	Ville de naissance	N° immatriculat	Type	Couleur	Code propriétaire
001	DUPONT	Jean	Mireval	74 GH 34	Visa	Blanche	001
003	PAOLI	Paul	Bastia	1489 WWC	C5	Rouge	003

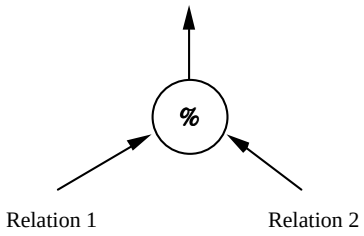
4.2.4 Le quotient

Cette opération est complémentaire, l'ensemble des opérateurs relationnels
{union, différence, produit cartésien, sélection, projection}
peut engendrer les autres opérateurs relationnels. Un langage possédant ces opérateurs ou pouvant les engendrer est dit **complet.**

La relation quotient d'une relation 1 par une Relation2 est la relation R1 **%** R2 qui regroupe exclusivement toutes les parties d'occurrence de la relation R1 qui sont associées à toutes les occurrences de la relation R2.

Les contraintes de cette opérations sont que R2 ne doit pas être vide, tous les attributs de R2 doivent être présents dans R1 et R1 doit posséder au moins un attribut de plus que R2.

Représentation graphique :



QUOTIENT

Exemple

ENSEIGNANT

Nom enseignant
DUPONT
MULLER
PAOLI

ETUDIANT

Nom étudiant
N GUYEN
ERZORG
BERTHE

ENSEIGNEMENT

Nom enseignant	Nom étudiant
DUPONT	ERZORG
MULLER	N GUYEN
DUPONT	N GUYEN
MULLER	ERZORG
MULLER	BERTHE
PAOLI	N GUYEN

ENSEIGNEMENT % ENSEIGNANT

Nom enseignant
MULLER

5 Le langage SQL

5.1 Présentation

Le langage SQL (Structured Query Language) est un dérivé de SEQUEL, lui-même dérivé de SQUARE, langage d'interrogation associé au prototype System R de chez IBM.

SQL a été normalisé par l'ANSI (American National Standards Institute) et par l'ISO (International Organization for Standardization). Voici les principales étapes de ce processus :

- première norme ANSI en 1986 ,
- première norme ISO (SQL1) en 1987, révisée en 1989 ,
- deuxième norme ISO (SQL2) en 1992 ,
- troisième norme (SQL3) en cours de rédaction depuis 1999 par l'ANSI et l'ISO, après une très longue gestation, et avec beaucoup de retard sur l'événement.

Malgré la normalisation, chaque éditeur présente une version SQL légèrement modifié tant au niveau de la syntaxe.

En SQL

Table (table)désigne une collection de lignes ou n-uplets (généralisation de la notion de relation car plusieurs lignes peuvent être identiques)

Colonne (columns) désigne un attribut
Ligne (Row) désigne un n-uplet

5.2 La commande SELECT

En SQL La commande SELECT appelée question ou interrogation ou requête (query) est fondamentale. La commande visualise une table par sélection, projection et jointure d'autres tables.

La forme la plus simple de la commande SELECT est :

SELECT projections
FROM liste de tables
Where condition

5.2.1 La projection

Visualisation de toutes les colonnes

SELECT *

FROM Tab ;

Exemple :

ENSEIGNANT

Code	Nom	Prénom	Ville de naissance
001	DUPONT	Jean	Mireval
002	MULLER	Herbert	Berlin
003	PAOLI	Paul	Bastia
004	PICARD	Georges	Mireval

SELECT * FROM ENSEIGNANT ;

001	DUPONT	Jean	Mireval
002	MULLER	Herbert	Berlin
003	PAOLI	Paul	Bastia
004	PICARD	Georges	Mireval

Visualisation d'une partie des colonnes

```
SELECT col1, col2
FROM Tab;
```

Exemple :

SELECT ville de naissance FROM ENSEIGNANT ;

Ville de naissance
Mireval
Berlin
Bastia
Mireval

Ceci n'est pas une projection dans la mesure où on peut obtenir des n-uplets identiques. Si on souhaite éliminer les n-uplets identiques, le mot SELECT doit être suivi de DISTINCT.

```
SELECT  DISTINCT col1,col2
FROM Tab ;
```

Exemple :

SELECT DISTINCT ville de naissance FROM ENSEIGNANT ;

Ville de naissance
Mireval
Berlin
Bastia

5.2.2 La sélection

Visualisation d'une partie de l'ensemble des données d'une table

```
SELECT *
FROM Tab
WHERE condition ;
```

Exemple:

SELECT * FROM ENSEIGNANT WHERE ville de naissance = 'Mireval';

001	DUPONT	Jean	Mireval
004	PICARD	Georges	Mireval

Pour exprimer les conditions on peut utiliser

des opérateurs

=, <, >, <=,>=, <>

des prédicats

BETWEEN teste l'appartenance d'une valeur à un intervalle donné (intervalle fermé)

Exemple : salaire BETWEEN 9000 AND 15000

Exemple
ENSEIGNANT

Code	Nom	Nombre d'enfants	Ville de naissance
001	DUPONT	2	Mireval
002	MULLER	4	Berlin
003	PAOLI	1	Bastia
004	PICARD	0	Mireval

SELECT Nom FROM ENSEIGNANT WHERE Nombre d'enfants BETWEEN 0 AND 2 ;

001	DUPONT	2	Mireval
003	PAOLI	1	Bastia
004	PICARD	0	Mireval

IN teste l'appartenance d'une valeur à une liste donnée
A [NOT] IN (S)
Exemple : lieu IN ('Versailles','Paris','Montpellier','Vierzon')

Exemple :

ENSEIGNANT

Code	Nom	Nombre d'enfants	Ville de naissance
001	DUPONT	2	Mireval
002	MULLER	4	Berlin
003	PAOLI	1	Bastia
004	PICARD	0	Mireval

SELECT * FROM ENSEIGNANT WHERE Ville de naissance IN ('Mireval','Bastia') ;

001	DUPONT	2	Mireval
003	PAOLI	1	Bastia
004	PICARD	0	Mireval

LIKE recherche toute valeur d'une colonne de type chaîne de caractères contenant une chaîne de caractères donnée

A LIKE modèle
Exemple : lieu LIKE '__ri_' (le _ représente un et un seul caractère)
lieu LIKE '%er%' (le % représente toute chaîne de caractères, la chaîne vide comprise)

Exemple :

ENSEIGNANT

Code	Nom	Nombre d'enfants	Ville de naissance
001	DUPONT	2	Mireval
002	MULLER	4	Berlin
003	PAOLI	1	Bastia
004	PICARD	0	Mireval

SELECT Nom FROM ENSEIGNANT WHERE Ville de naissance LIKE ‘B%’ ;

MULLER
PAOLI

EXISTS teste le « non vide » d’un ensemble

des opérateurs logiques

NOT, AND, OR

AND

	VRAI	**FAUX**
VRAI	VRAI	FAUX
FAUX	FAUX	FAUX

OR

	VRAI	**FAUX**
VRAI	VRAI	VRAI
FAUX	VRAI	FAUX

5.2.3 La jointure

Visualisation de l’ensemble (ou d’une partie) des données de plusieurs tables ou vues

On a défini tab1(col1, col2, col3), tab2(col1, col4)

```
SELECT tab1.col1, col2, col4
FROM tab1, tab2
WHERE tab1.col1=tab2.col1 and col2=col4;
```

Effectue une jointure entre tab1 et tab2, la colonne col1 est commune aux deux tables, le nom de la colonne pour être identifié est préfixé par le nom de sa table suivi d’un point.

Exemple
ENSEIGNANT

Code enseignant	Nom	Prénom	Ville de naissance
001	DUPONT	Jean	Mireval
002	MULLER	Herbert	Berlin
003	PAOLI	Paul	Bastia

VOITURE

N° immatriculation	Type	Couleur	Code enseignant
74 GH 34	Visa	Blanche	001
1489 WWC 66	C5	Rouge	003

SELECT Nom, N° immatriculation
FROM ENSEIGNANT, VOITURE
WHERE ENSEIGNANT.code enseignant=VOITURE.code enseignant ;

DUPONT	74 GH 34
PAOLI	1489 WWC 66

Remarque : Il est possible d’effectuer des « auto-jointures »

```
SELECT A.col1,B.col2
FROM tab1 A,tab1 B
WHERE NOT (A.col1=B.col1) AND A.col2=B.col3 ;
```

Remarque : A et B sont des variables lignes ou **alias**

5.2.4 Les sous-requêtes imbriquées

La condition exprimée dans la clause WHERE peut porter sur une sous-requête ou requête imbriquée Celle-ci :

- doit être enfermée dans des parenthèses,
- ne doit avoir qu'une seule colonne ou expression dans sa clause SELECT,
- retourner une et une seule valeur exceptée quand elle est précédée de IN, ALL, ANY,
- le prédicat EXISTS est le seul qui permet l'emploi de SELECT * dans une sous-requête.

Exemple
ENSEIGNANT(Nom, Matiere)
ENSEIGNEMENT(code enseignement, nom enseignant, salle)

Salles utilisée pour la matière informatique

```
SELECT DISTINCT salle
FROM ENSEIGNANT, ENSEIGNEMENT
WHERE Nom enseignant=Nom AND matiere= 'informatique' ;
ou

SELECT DISTINCT salle
FROM ENSEIGNEMENT
WHERE nom enseignant  IN
(SELECT Nom
FROM ENSEIGNANT
WHERE matiere = 'informatique');
```

5.2.5 L'union

```
SELECT col1
FROM tab1
UNION
SELECT col2
FROM tab2 ;
```

Les informations résultantes de chaque requête doivent être du même type.

Exemple
ENSEIGNANT

Code enseignant	Nom enseignant	Prénom enseignant	Ville de naissance
001	DUPONT	Jean	Mireval
002	MULLER	Herbert	Berlin
003	PAOLI	Paul	Bastia

ETUDIANT

Code étudiant	Nom étudiant	Prénom étudiant	Date de naissance
110	N GUYEN	Xan	01/12/87
2052	ERZORG	Enzo	08/06/89
303	BERTHE	Pascal	03/04/85

```
SELECT Nom étudiant FROM ETUDIANT
UNION
SELECT Nom enseignant FROM ENSEIGNANT ;
```

N GUYEN
ERZORG
BERTHE
DUPONT
MULLER
PAOLI

Remarque : dans certain SQL, il existe aussi les fonctions INTERSECT (intersection) et EXCEPT (à l'exception de).

5.2.6 Les fonctions d'évaluation de table (fonctions agrégatives)

Comptage du nombre de lignes

SELECT COUNT(*)

FROM tab
WHERE condition ;

SELECT COUNT(DISTINCT <colonne>)
FROM tab
WHERE condition ;

Exemple
VOITURE

N° immatriculation	Type	Couleur
74 GH 34	Visa	Blanche
1489 WWC 66	C5	Rouge
15 FG 34	Mégane	Blanche
454 GHX 34	2CV	Blanche
14 RT 66	105	Verte
1456 HY 34	Clio	Noire
58 YG 66	Twingo	Mauve
45 YHG 34	607	Blanche

SELECT COUNT(couleur) FROM VOITURE ;
8
SELECT COUNT(DISTINCT couleur) FROM VOITURE ;
5
SELECT COUNT(*) FROM VOITURE ;
8

Moyenne

AVG ([DISTINCT] <colonne>)

Exemple

ETUDIANT

Code étudiant	Nom étudiant	Prénom étudiant	Note
110	N GUYEN	Xan	12
2052	ERZORG	Enzo	3
303	BERTHE	Pascal	15

SELECT AVG(note) FROM ETUDIANT ;
10

Somme

SUM ([DISTINCT] <colonne>)

Exemple
SELECT SUM(note) FROM ETUDIANT ;
30

Minimum

MIN (<colonne>)

Exemple
SELECT MIN(note) FROM ETUDIANT ;
3
SELECT MIN(nom étudiant) FROM ETUDIANT ;
BERTHE

Maximum

MAX (<colonne>)

5.2.7 Le Groupage - Partionnement

par la clause GROUP BY liste <colonne> HAVING <condition>

La clause HAVING effectue une sélection à l'image du WHERE mais de groupes de la partition et non plus des lignes de la table.

Toutes les colonnes représentées hors des calculs d'agrégation doivent figurer dans la clause GROUP BY.

Exemple
ENSEIGNEMENT(code enseignement, nom enseignant, salle)

Nombre de salles utilisées par enseignant
SELECT nom enseignant, COUNT(DISTINCT salle)
FROM ENSEIGNEMENT
GROUP BY nom enseignant ;

Liste des enseignants qui utilisent plus de 6 salles
SELECT nom enseignant
FROM ENSEIGNEMENT
GROUP BY nom enseignant
HAVING COUNT(DISTINCT salle) >6 ;

5.2.8 Les Tris

par la clause

ORDER BY liste { <colonne> / <entier> } [ASC / DESC]

Les colonnes utilisées dans la clause ORDER BY doivent faire partie de la clause SELECT

Exemple
ETUDIANT(Code étudiant, Nom étudiant, Prénom étudiant) ;

Trie des noms des étudiants par ordre alphabétique puis sur les notes décroissantes

SELECT Nom étudiant, Note
FROM ETUDIANT
ORDER BY Nom étudiant, Note DESC ;

5.2.9 La forme complète de la commande SELECT

SELECT [ALL / DISTINCT] { liste <colonne/expression> / * }

FROM liste [<propriétaire> .] { <table> / <vue> } [<variable ligne>]

[WHERE <condition>]

[GROUP BY liste <colonne>] [HAVING <condition>]

[UNION <commande SELECT>]

[ORDER BY liste { <colonne> / <entier> } [ASC / DESC]] ;

5.3 Définition de données

Le langage de définition de données permet de déclarer tous les objets décrivant la structure (c'est-à-dire le schéma) de la base : tables, attributs, index...

Il existe diverses variantes syntaxes au niveau de la définition des données, nous utiliserons celle du SQL ORACLE.

5.3.1 Définition de table

Définir une table c'est déclarer son nom et sa structure. La définition de chaque attribut consiste à donner son nom, son type, sa taille et éventuellement et dire s'il peut prendre ou non les valeurs NULL.

CREATE TABLE <table> (liste <colonne> <type> <contraintes> ;

<table> :	nom de la table
<colonne> :	nom de l'attribut
<type> :	type de l'attribut

Les types de données admis par ORACLE sont :

char(n)	Un type chaîne de caractères à longueur fixe (n caractères).
varchar2(n)	Un type chaîne de caractères à longueur variable mais ne dépassant pas n caractères.
number(n,d)	Un type numérique à n chiffres et à d décimales
date	Un type permettant de représenter des dates. Attention la syntaxe diffère selon le pays ('jj/mm/aa' ou 'jj-mmm-aa').
long	Un type texte de taille maximum 2 Go.
raw	Un type binaire (256 octets maximum).
long raw	Idem mais jusqu'à 2 Go.

Quelques contraintes :

null	Des cellules de la colonne peuvent ne pas contenir de valeur.
not null	Toutes les cellules de la colonne doivent contenir une valeur du type mentionné et respectant les contraintes apposées.
primary key	Les valeurs d'une telle colonne servent à accéder efficacement une entrée de la table. Il est clair que chaque valeur doit être unique.
forein key	Clé étrangère sur une autre table.

references Les valeurs de cette colonne sont en fait des références sur des entrées d'une autres table. C'est par ce mécanisme que l'on relie des informations dispersées sur plusieurs tables.
check(exp) Toute valeur de la colonne doit vérifier cette condition

Exemple :

CREATE TABLE ENSEIGNANT

(code NUMBER (6) NOT NULL,
nomens CHAR (20), indice NUMBER (8) CONSTRAINT BETWEEN 0 AND 100,
adresse CHAR (80)
categorie NUMBER (3),
PRIMARY KEY code,
FOREIN KEY categorie REFERENCES CAT);

Il est possible de créer une table non vide à partir d'autres tables existant dans la base.

CREATE TABLE <table> …
AS <commande SELECT>;

5.3.2 Définition d'index.

Les index ou accélérateurs d'accès aux données sont définis par :

CREATE [UNIQUE] INDEX <index> ON <table> (liste <colonne> [ASC/DESC]);

Remarques :

- L'option UNIQUE spécifie que l'ensemble des attributs de l'index est un identifiant de la table.
- L'option ASC/DESC permet de trier les données permettant un accès plus rapide aux emplacements des données dans la table concernée.

Exemple :

CREATE INDEX code ON ENSEIGNANT ;

5.3.3 Définition de vue

CREATE VIEW [<propriétaire>]<vue> [liste (<colonne>)] AS <commande SELECT> ;

Une vue est une table virtuelle. Une vue n'est pas stockée mais elle a les mêmes propriétés qu'une table. Quand la table est mise à jour, la vue l'est aussi.

L'avantage d'une vue est que l'on ne montre à l'utilisateur que les attributs souhaités.

Création d'une vue sous Oracle

CREATE [OR REPLACE] [FORCE | NOFORCE] VIEW [propriétaire .] nom_vue

(nom_colonne,...,nom_colonneN)
AS instruction_sélection...;
[WITH CHECK OPTION

OR REPLACE remplace la vue si elle existe déjà.
FORCE et NOFORCE indiquent respectivement que la vue est créée sans se soucier de l'existence des tables qu'elle référence ou des privilèges adéquats sur les tables, la vue est créée seulement si les tables existent et si les permissions requises sont données.
WITH CHECK OPTION limite les insertions et les mises à jour exécutées par l'intermédiaire de la vue.

5.3.4 Définition de synonymes

Nommer autrement une table ou une vue est possible par :

CREATE [PUBLIC] SYNONYM <synonyme> FOR [<propriétaire>]<table>/<vue> ;

5.4 Modification de données.

5.4.1 Modification de la structure d'une table

Modifier une table revient à :

- ajouter un ou plusieurs attributs,
- modifier le type d'un attribut,
- supprimer un attribut.

La suppression d'un attribut n'est pas exprimable directement en SQL. Elle peut-être réalisée d'une façon indirecte par la création d'une nouvelle table en omettant de recopier l'attribut.

Ajout de nouveaux attributs dans une table

Pour ajouter un attribut, il suffit de le nommer et de donner son type et éventuellement sa taille et de spécifier NOT NULL.

ALTER TABLE <table> ADD (liste <colonne> <type> [NULL/NOT NULL]...) ;

Exemple :

Ajout de l'attribut 'sexe' dans la table ENSEIGNANT
ALTER TABLE ENSEIGNANT ADD (Sexe CHAR) ;

Modification du type d'un attribut

Cette modification concerne :

a) la réduction de la taille : seulement si l'attribut ne contient que des valeurs 'nulles',
b) la modification du type = même condition que a),
c) l'augmentation de la taille est toujours possible,
d) la suppression de l'interdiction de présence de valeurs nulles (passage de NOT NULL à NULL) : toujours possible.

La modification est réalisable par la Commande :

ALTER TABLE <table> MODIFY (liste <colonne> [<type>] [NULL/NOT NULL]);

Exemple :

Augmentation de l'attribut nomens CHAR de 20 à 25 caractères
ALTER TABLE ENSEIGNANT MODIFY nomens CHAR (25) ;

5.4.2 Renommer une table

On peut changer le nom d'une table en la renommant. Mais il faut prendre soin de répercuter cette modification au niveau des applications et des vues définies sur cette table

RENAME <ancienne_table> TO <nouvelle_table> ;

La possibilité de définir des synonymes permet de pallier aux problèmes posés par le renommage d'une table. En effet, une table peut-être référencée soit par son nom initial soit par le synonyme.

5.4.3 Mise à jour des données d'une table

Ajout d'un n-uplet ou d'une ligne dans une table :

INSERT INTO <table> [liste <colonne>] VALUES (liste <valeur>) ;

Exemple :

Table ENSEIGNANT(numéro, nom, adresse, date naissance, sexe)

INSERT INTO ENSEIGNANT values (21,'LEONARD','12 rue des tulips 34000 MONTPELLIER', 15/01/1985, 'M') ;

Ajout d'une ou plusieurs lignes sélectionnées à partie d'autres tables :

INSERT INTO <table> <commande SELECT>;

Mise à jour de données à l'intérieur d'une table

```
UPDATE <table>
SET liste <colonne>=<expression>
[WHERE condition ];
```

Remarque : si la clause WHERE n'existe pas la mise à jour concerne toutes les lignes de la table.

Exemple
Augmentation de l'indice de +1 pour la categorie 3
UPDATE ENSEIGNANT SET indice=indice+1 WHERE categorie=3 ;

5.4.4 Suppression

Suppression d'une table

C'est une opération rare et qui est à manipuler avec prudence surtout lorsqu'il existe des vues définies sur cette table. Cette commande est utile pour supprimer des tables de travail.

DROP TABLE <table> ;

Elle a pour effet de supprimer la table spécifiée, et tous les index, synonymes correspondants.

Suppression d'un synonyme

DROP [PUBLIC] SYNONYM <synonyme>;

Suppression d'un index.

DROP INDEX <index> [ON <table>] ;

Suppression d'une vue.

DROP VIEW <vue> ;

Suppression de données à l'intérieur d'une table

DELETE FROM <table> [WHERE <condition>] ;

Remarque : si la clause WHERE n'existe pas la suppression concerne toutes les lignes de la table.

Exemple
Suppression de tous les enseignants de catégorie 3
DELETE FROM ENSEIGNANT WHERE catégorie=3 ;

5.5 Les droits.

L'accès d'un utilisateur est contrôlé au moment de la connexion. (login et password). La commande GRANT permet d'attribuer un ou plusieurs privilèges à un utilisateur ou un rôle sur des instructions (create database, create table, create view, etc.) ou divers objets tels que des tables, des vues, des colonnes ou encore des procédures stockées.

Les droits d'accès à un objet (table, vue, ..) dépendent :

- de son autorité administrative
- de sa propriété d'être ou non propriétaire des objets
- d'autorisations explicites accordées.

GRANT liste <droit ou privilège>/ALL ON <ressource> TO liste <usager>/PUBLIC [WIDTH GRANT OPTION] ;

GRANT

{{{ système_privilège | rôle | ALL PRIVILEGES }…
TO {{ utilisateur | rôle | PUBLIC } …}
[IDENTIFIED BY mot_passe] [WITH ADMIN OPTION]}
{{ objet_privilège | ALL [PRIVILEGES] } [(nom_colonne [, nom_colonne]...)]
[, { objet_privilègeN | ALL [PRIVILEGES] } [(nom_colonne [, nom_colonneN]...)]]
ON {{ schema . objet } |
{{ DIRECTORY nom_répertoire } |
{ JAVA { SOURCE | RESOURCE } [schéma .] objet } }}
TO { { utilisateur | rôle | PUBLIC } [, { utilisateurN | rôle | PUBLIC }] }
[WITH GRANT OPTION] [WITH HIERARCHY OPTION] }};

(ref **http://www.laltruiste.com/**)

IDENTIFIED BY	permet d'identifier un utilisateur existant par un mot de passe ou pour créer un utilisateur inexistant.
WITH ADMIN OPTION	fournit à un utilisateur la possibilité d'administrer un rôle.
ALL PRIVILEGES	indique l'attribution de toutes les autorisations appliquées au compte de sécurité indiqué.
PUBLIC	indique l'attribution de privilèges ou de rôles à tous les utilisateurs.
ON	indique l'application des autorisations sur un objet indiqué.
DIRECTORY	indique l'objet DIRECTORY sur lequel les privilèges doivent être accordés.

JAVA { SOURCE | RESOURCE }

permet de spécifier une source Java ou l'objet de schéma de ressource sur lequel les privilèges doivent être accordés.

FROM	indique la liste des comptes de sécurité.
TO	indique la liste des comptes de sécurité.
WITH GRANT OPTION	indique la possibilité pour le compte de sécurité de pouvoir lui-même accorder des privilèges à d'autres utilisateurs.

WITH HIERARCHY OPTION

indique la permission d'étendre les privilèges d'objets spécifiés à tous les sous-objets.