Bases de Données

Jean Fruitet
jf@univ-mlv.fr

Université de Marne-La-Vallée

Septembre 1997

1. Introduction aux systèmes de gestion de bases de données

1.1. Un peu d'histoire

Les Systèmes de Gestion de Bases de Données (SGBD) ont vu le jour dans les années 60 pour gérer d'importants volumes de données de gestion. Il s'agissait de systèmes propriétaires (appartenant à une marque d'ordinateur, par exemple IBM [International Buisinesss Machine - Big Blue]), sur grands systèmes [main frame] conçus selon un schéma d'organisation «hiérarchique» ou «réseau».

Le modèle relationnel de Codd

En 1970, Codd, chercheur chez IBM , proposa le **modèle relationnel**.
Ce modèle conceptuel constitue un progrès important car il repose sur une représentation unifiée de l'information sous forme de tables. Il dispose d'un fondement mathématique solide avec l'algèbre relationnelle (opérations ensemblistes). Il permet une plus grande indépendance entre les applications, les données et le support physique [*hardware* et *software*]. Il propose une démarche cohérente et unifiée pour la description (Langage de Description des Données - LDD) et pour l'interrogation (Langage de Manipulation des Données - LMD). Enfin le modèle relationnel supporte le langage SQL [*Sequel : Standard English Query Language*] aussi bien comme LDD que LMD, basé lui-aussi sur l'algèbre relationnelle.

Les SGBD dans les année 90

De nombreux SGBD sont aujourd'hui disponibles sur micro ordinateurs. La plupart sont dotés de capacités relationnelles, bien que l'ancêtre des SGBD sur micro, DBase (Borland) ne soit qu'un gestionnaire de fichiers structurés avec un langage de programmation. On peut citer FoxPro (clone de DBase) et Access (Microsoft) et Paradox (Borland).
Sur stations de travail et mini ordinateurs sous Unix, trois ou quatre SGBD relationnels dominent : Oracle, Ingres, Informix, Sybase.
DB2 (IBM) est un SGBD relationnel sur main frame.

Les SGBD, qui sont la raison d'être de l'informatique de gestion, ont vu leur domaine d'utilisation s'élargir considérablement. Bases de Connaissances, Systèmes Experts, Systèmes d'Information Géographique, Edition de Documents Informatisés (EDI), Systèmes d'Information Documentaire (SID), Conception Assistée par Ordinateur (CAO), Gestion de Production Assistée par Ordinateur (GPAO) sont des domaines où une information structurée est enregistrée et gérée par un SGBD générique (relationnel) et traitée selon des besoins spécifiques, l'interface et les outils de traitement dépendant plus particulièrement de l'application.

1.2. Base de données et Système de Gestion de Base de Donnée [*Data Base Management System*]

Une base de données est un ensemble structuré de données enregistrées avec le minimum de redondance pour satisfaire simultanément plusieurs utilisateurs de façon sélective en un temps opportun.

L'approche base de données correspond à une triple évolution :

- évolution des entreprises (volumes importants de données, centralisées ou réparties, qui doivent être accessibles en temps utile,...)

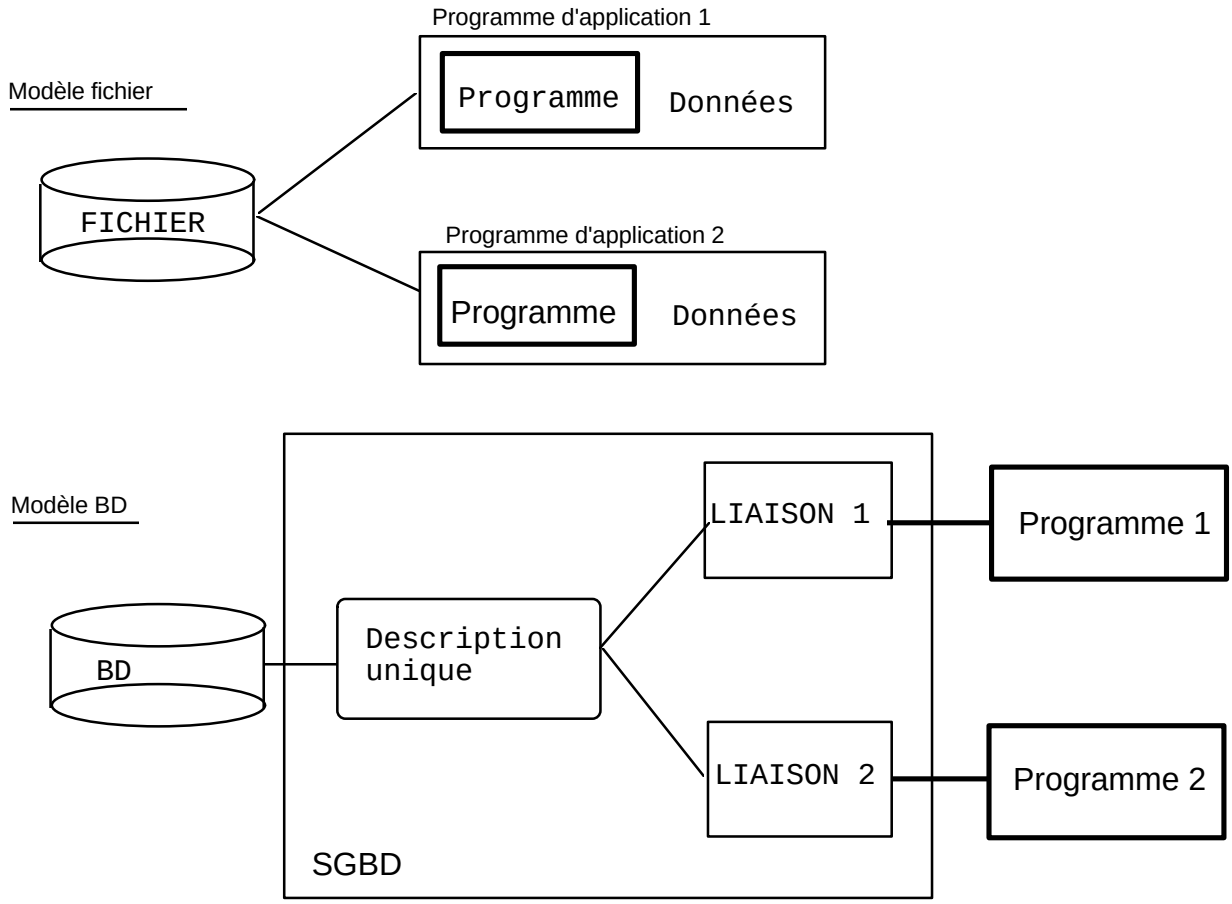
- évolution technologique (accroissement des performances, intégration des composants, diminution des coûts, ...)
- évolution des systèmes d'exploitation (SE) et des architectures : extension logicielle du matériel initial, les SE réalisent une machine virtuelle très puissante qui définit un environnement pour des langages de haut niveau ; architectures client serveur et réseaux, combinant de façon transparente des machines et des applications hétérogènes.

Un système organisé autour d'une base de données est **centré sur les données**, contrairement aux systèmex de gestion plus anciens (et dépassés) basés sur les fonctions et les traitements (par exemple : chaîne de traitement de la paye, chaîne de la facturation, gestion des stocks, etc.)

Cependant les modes informatiques changent et même l'approche base de données est remise en question. On voit émerger un nouveau modèle, dit modèle objet, qui est lui centré sur les structures. Nous n'aborderons pas cette approche dans ce cours.

Dans l'**approche gestion de fichiers,** les fichiers sont définis pour un ou plusieurs programmes de traitement. Les données d'un fichier sont directement associées à un programme par une description contenue dans le programme de traitement lui-même. Il n'existe aucune indépendance entre le programme et les données. Toute modification de la structure des données nécessite la réécriture du programme.

Dans l'**approche base de données**, la partie de structuration et de description des données est unifiée et séparée des programmes d'application. Bien sûr la gestion de ces données (stockage, modification, recherche) qui est étroitement dépendante de leur structuration, est fournie par le Système de gestion des données, les applications ne communiquant avec les données qu'au travers de l'interface de gestion. D'où l'indépendance entre les données et les applications, qui peuvent être modifiées indépendamment. Le programmeur des applications (et a fortiori l'utilisateur) n'a pas à connaître l'organisation physique des données...



Exemple : Base de données d'une compagnie aérienne

Les données sont relatives aux passagers, aux vols, aux appareils, aux équipages... Les requêtes sont très variées :
- une réservation : «liste des passagers qui ont réservé sur un vol déterminé» ;
- un équipage : «quel est le pilote du vol Air France Paris-Londres du 15 octobre, départ 17h20 ?»
- un appareil : «quelle est la date de la dernière révision de l'appareil Airbus A300 numéro X ?»

1.1 Systèmes de Gestion de Base de Données

Un Système de Gestion de Base de Données (SGBD - *DBMS*) permet à un utilisateur de communiquer avec une base de données pour :
- décrire et organiser les données sur les mémoires secondaires (disques)
- rechercher, sélectionner et modifier les données

Un SGBD offre la possibilité à l'utilisateur de manipuler les représentations abstraites des données, indépendamment de leur organisation et de leur implantation sur les supports physiques (mémoires).

On peut considérer un SGBD comme un interpréteur d'un langage de programmation de haut niveau qui, dans le cas idéal, permet à l'utilisateur de décrire précisément ce qu'il veut obtenir et non comment l'obtenir : 'quoi' et non 'comment', c'est-à-dire formuler une assertion et non décrire une procédure (langage assertionnel langage procédural).

Exemple : (SQL)

1. Donner la liste des numéros de vols au départ de Toulouse qui sont des vols bleus»

```
SQL> SELECT N-VOL
        FROM VOL
        WHERE TYPE-VOL = «BLEU» AND ORIGINE = «TOULOUSE» ;
```

2. Table EMPLOYES

MATRICULE	NOM	SERVICE	TELEPHONE
0079	HENRION	Comptabilité	2111
0101	PIERRE	Comptabilité	2211
0126	MARTINON	Ventes	6312
0846	JEANCARD	Comptabilité	2112
2312	BARON	Distribution	3756
4684	DENIS	Réception	3855
6587	MARTIN	Réception	4422

Requête : A partir de la table EMPLOYES, fournir tous les noms par ordre alphabétique des personnes appartenent au service **Comptabilité** avec leur numéro de **Téléphone**

```
SQL> SELECT NOM, TELEPHONE
     FROM EMPLOYES
     WHERE SERVICE = «Comptabilité»
     ORDER BY NOM ;
SQL> NOM   TELEPHONE
     HENRION    2111
     JEANCARD   2112
     PIERRE     2211
```

Un SGBD assure
- la **description** des données,
- leur **recherche** et **mise à jour**,
- la **sûreté** : vérifier les droits d'accès des utilisateurs ; limiter les accès non autorisés ; crypter les informations sensibles
- la **sécurité** : sauvegarde et restauration des données ; limiter les erreurs de saisie, de manipulation
- l'**intégrité** : définir des règles qui maintiennent l'intégrité de la base de données (contraintes d'intégrité)
- la **concurrence** d'accès : détecter et traiter les cas où il y a conflit d'accès entre plusieurs utilisateurs et les traiter correctement.

1.3. Mise en oeuvre d'un SGBD

On distingue trois niveaux d'appréhension d'une base de données. A chaque niveau correspond un schéma de représentation :

- le niveau interne avec le schéma physique
- le niveau conceptuel avec le schéma conceptuel
- le niveau externe avec les vues

Le niveau interne

Le schéma physique spécifie comment les données sont enregistrées sur les mémoires secondaires (disques, bandes, tambours, ...).

La base physique de données, seule, a une existence matérielle.
Cette base est elle-même perçue à différents niveaux d'abstractions :

- enregistrement, article (*struct* ou *record* d'un langage de programmation)
- enregistrement logique [*logical record*]
- fichier [*file*]
- Octet / mot machine
- bit / adresse physique en mémoire

Le niveau conceptuel

Le schéma conceptuel décrit en termes abstraits mais fidèles la réalité du domaine d'application (par exemple l'entreprise).

Pour une base de données d'une compagnie aérienne le niveau conceptuel exprimera la réalité en termes de vol, équipage, passager, horaire...

Le SGBD fournit un langage de définition de données (LDD - *Data Description Language*), qui spécifie le schéma conceptuel. C'est un langage de haut niveau, qui décrit et exprime la «base de données» conceptuelle par référence à un **modèle de données** (un outil formel utilisé pour comprendre et interpréter le monde réel). Un graphe Entité-Association constitue un exemple de modèle de données.

Le niveau externe

Il s'agit de décrire à l'aide d'un schéma externe parfois appelé «vue» [*view*] la façon dont seront perçues les données par un programme d'application. Une **vue** est une représentation abstraite d'une partie de la base de données conceptuelle (ou un sous-schéma du schéma conceptuel).

Dans l'exemple de la compagnie aérienne, le service de réservation ne s'occupe que des vols et des passagers, et n'a donc pas accès aux données relatives au personnel et à l'affectation des pilotes sur les différents vols.

En général, une vue est un sous-ensemble de la base conceptuelle de données. Cependant, dans certains cas, une vue peut être «plus abstraite» que la base conceptuelle de données, par exemple les données qu'elle utilise se déduisent de la base conceptuelle de données mais ne sont pas présentent dans cette base.

Exemple 1 : Une vue fournissant l'âge des employés, qui sera calculé à partir de la date de naissance, seule enregistrée dans le système...

Exemple 2 : Tableau *n* lignes de *m* colonnes d'entiers

- au niveau conceptuel

int A[*n*][*m*]

- au niveau physique

A enregistré séquentiellement à partir de l'adresse a_0 ; A[i][j] est implanté à l'adresse a_0+sizeof(**int**)*(*m**(*i*-1)+*j*-1)

- une vue du tableau A peut être définie en déclarant une fonction **f**(*i*) définie par

$$\mathbf{f}(i) = \sum_{j=0,m-1} A(i,j)$$

On «voit» les sommes des éléments des différentes lignes.

Lors de la conception d'une base de données on raisonne sur le schéma conceptuel. Lors de l'exploitation de la base des données, on s'intéresse aux données effectivement présentes. L'ensemble des données présentes à un instant déterminé est dénommé l'**extension** de la base (instances).

1.4. Indépendance physique - Indépendance logique

La classification précédente suggère deux types d'indépendances :

- l'**indépendance physique** : on change le schéma physique sans modifier le schéma conceptuel et sans redéfinir les vues. Les programmes d'application n'ont pas à être réécrits quand on change de configuration matérielle ou de version du système d'exploitation.
- l'**indépendance logique** : on peut modifier le schéma conceptuel en ajoutant des informations à certaines classes d'objets ou introduire de nouveaux objets sans modifier les programmes d'application.

1.5. SGBD et Langages

Dans les langages de programmation classiques, les déclarations et les instructions exécutables appartiennent au même langage. Dans le SGBD on exprime les déclarations et les instructions exécutables dans deux langages différents. En effet dans un SGBD les données existent en permanence et doivent être déclarées une fois pour toutes, contrairement aux variables des programmes classiques qui disparaissent de la mémoire quand le programme s'arrête.

Le langage de description des données (LDD) spécifie le schéma conceptuel

C'est un langage descriptif des types d'entités, de leur attributs et domaines et des associations (ou relations) entre ces entités.
On l'utilise lors de la définition de la base de données, lors des modifications de schéma et pour préciser la façon dont les données sont enregistrées et comment y accéder (correspondance entre le schéma conceptuel et le schéma physique).

Langage de manipulation des données, ou langage d'interrogation [*query language*] (LMD)

Pour interroger la base, mettre à jour les données et effectuer les manipulations sur celles-ci.

Exemples :

- enregistrer un nouvel employé
- rechercher un vol avec une place disponible de Paris à Londres le 15 septembre...

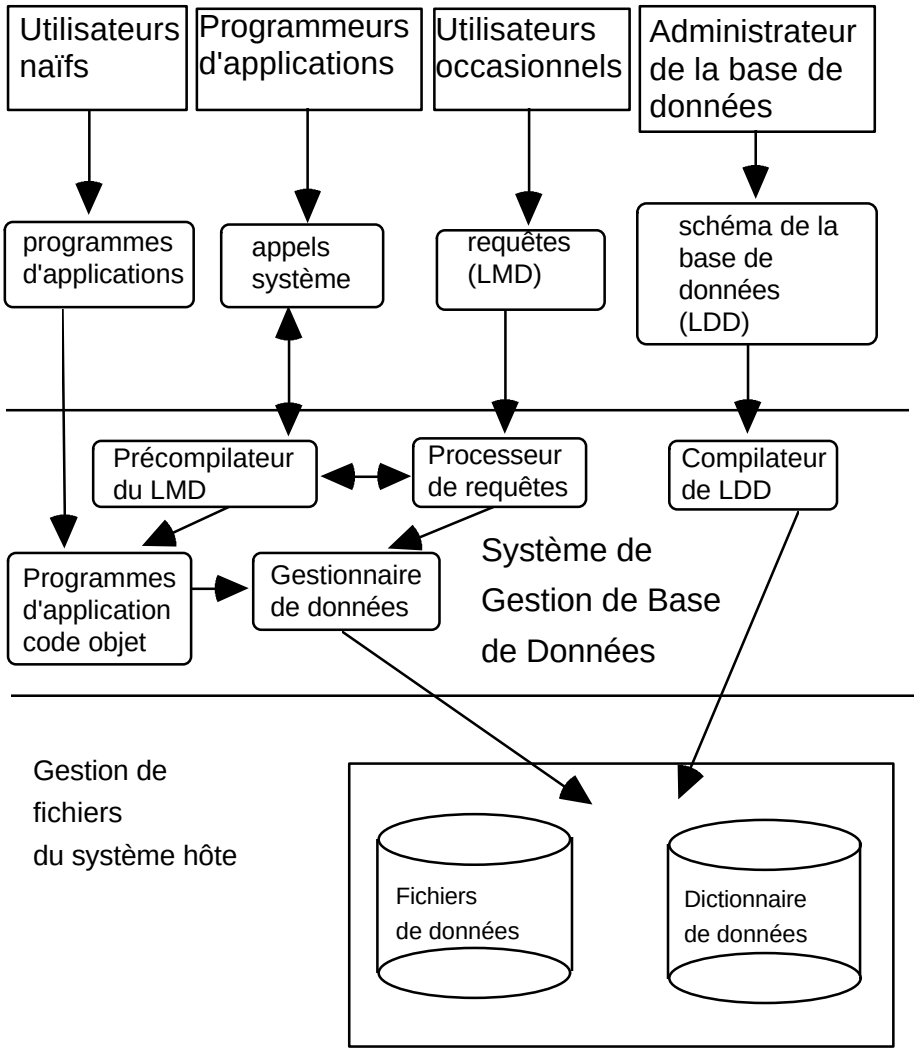
Généralement un programme d'application (ex. : gérer les réservations) est écrit dans un langage de programmation traditionnel, dénommé «langage hôte» (C, COBOL, ...), mais la communication avec la base de donnée s'effectue par des instructions du LMD, activées à partir du langage hôte.
On conserve ainsi la puissance créatrice des langages de programmation (interfaces, contrôles, aides à l'utilisateur, copyright) pour l'application tout en profitant de la généralité et de la portabilité du LMD.
Le portage d'un système à l'autre en est facilité.

1.6. Les intervenants

L'**administrateur** de la base de données a la responsabilité de la gestion du système dans son ensemble :

- la définition du schéma original de la base
- le choix des structures de données et des méthodes d'accès au niveau physique
- la modification du schéma et de l'organisation physique en fonction de l'évolution de la base
- la gestion des droits d'accès et des privilèges des utilisateurs
- la spécification des contraintes d'intégrité
- les sauvegardes et restaurations
- la programmation (ou l'acquisition) d'applications

Les **utilisateurs occasionnels** interagissent avec le système sans écrire de programme mais en formulant leurs requêtes avec le LMD.
Les **utilisateurs habituels** (ex. : Service de la paye) utilisent des programmes d'application prédéfinis et permanents.



Rôle du gestionnaire de données :
- interaction avec le gestionnaire de fichier du système hôte
- intégrité de la base
- sécurité
- sauvegardes et restaurations
- contrôle des accès concurrents

2. Le modèle entité-association

L'informatisation de nombre d'activités nécessite leur modélisation, c'est-à-dire leur expression sous une forme symbolique (le plus souvent mathématique) susceptible d'être représentée en machine.
Le modèle EA (Entité-Association ou Entité-Relation) fournit un outil informel pour analyser les situations du monde réel (entreprises, institutions...)

2.1. Entité

Une entité est un être ou un objet (concret ou abstrait) qui existe et peut être distingué d'un autre objet
Exemple : une personne (Dupond), un véhicule (vélo), un concept (emploi), un sentiment (douleur, joie...)
On regroupe les entités de même nature en un **ensemble d'entités**, par exemple toutes les personnes, les véhicules, les sentiments...

Classes d'entités

La **classe d'entités** représente de manière abstraite l'ensemble d'entités ; on définiera par exemple la classe d'entités PERSONNE, VEHICULE, SENTIMENT...
Un ensemble d'entités se définit :
- en extension : {masculin, féminin}
- en intention : {x est une personne telle que sexe(x) = masculin}
- par un produit cartésien

X, Y, Z désignent des noms d'ensembles
X x Y x Z désigne l'ensemble des triplets de la forme (x, y, z)
avec x appartenant à X, y appartenant à Y et z appartenant à Z

Exemple : au rayon produits frais d'un super marché, les denrées peuvent être considérées comme des ensembles d'entités définis par 3 ensembles : NOM, PRIX, POIDS

DENREE

NOM	**PRIX (au kilo)**	**POIDS**
Oranges	6	2
Carottes	5.40	1
Haricots verts	15.50	0.5

(Carottes, 5.40, 1) est un **tuple** (ici un triplet) de l'ensemble d'entités DENREE.

Attribut, valeur, domaine, clé

Un **attribut** est une propriété caractéristique des entités de même classe. Un attribut associe à chaque entité une **valeur** appartenant à un domaine. Un **domaine** est un ensemble de valeurs acceptables pour l'attribut considéré ; le domaine de l'attribut PRIX est l'ensemble des réels positifs.
Exemple
Les entités de l'ensemble d'entités ETUDIANT ont les attributs NOM, DDN (date de naissance), ADRESSE; les domaines de valeurs sont ici les ensembles de chaînes de caractères qui permettent

de préciser le nom (de type *texte*), la date de naissance (qui peut être précisément de *type date*) et l'adresse d'un étudiant (*texte*).

Une **clé** est un attribut ou un ensemble d'attributs dont les valeurs identifient de manière unique une entité au sein de l'ensemble d'entités.

Exemple pour ETUDIANT (entité), NOM, DDN et ADRESSE ne forment pas une clé car deux étudiants jumeaux ne sont pas distingués par ces tuples. Par contre les attributs NUMERO D'ETUDIANT, et N° SS sont chacuns des clés.

Notations

$E = (A_1, A_2, ..., A_n)$, est le *schéma* de l'ensemble d'entités E ; A_i sont des attributs.

$D(A_1)$ est le *domaine* de l'attribut A_1

$e = (a_1, ..., a_n)$ est une entité, c'est à dire une instanciation ou une occurrence de E avec chaque a_i élément de $D(A_i)$

$K(E) = (A_i, A_j, A_k)$ est la *clé* de $(A_1, A_2, ..., A_n)$

(A_i, A_j, A_k) --> $(A_1, A_2, ..., A_n)$ signifie que (A_i, A_j, A_k) détermine de façon unique $(A_1, A_2, ..., A_n)$

Le choix des attributs, des domaines et des clés constitue une étape essentielle lors de la définition d'un modèle du monde réel. Parmi toutes les clés qui identifient une entité dans un ensemble, on appelle *clé primaire* celle qui est retenue par le concepteur de la base de données pour identifier l'ensemble considéré.

Entité dominante et entité subordonnée

Si l'existence d'une entité y dépend de l'existence d'une entité y, x est dominante et y subordonnée. Si x est éliminé de la base de données, y doit l'être aussi.

Exemple : une base de données BANQUE ; les entités CLIENT, EMPLOYE, COMPTE, TRANSACTION, avec

- COMPTE défini par les attributs N_COMPTE et SOLDE
- TRANSACTION défini par les attributs N_TRANSACTION, DATE, MONTANT

Un compte peut être concerné par plusieurs transactions. Une transaction doit être associée à un compte. Si un compte est supprimé, on supprime également toutes les transactions qui le concernent (l'inverse n'est pas vrai). L'entité COMPTE est dominante, TRANSACTION est l'entité subordonnée.

Généralisation et hiérarchie

Un ensemble d'entités E_1 est un sous-ensemble de E_2 si toute occurrence de E_1 est aussi une occurrence de E_2. L'ensemble d'entités E1 hérite des attributs de E_2. Par exemple l'ensemble des PILOTES est un sous-ensemble de l'ensemble des EMPLOYES d'une compagnie aérienne.

Un ensemble d'entités E est une généralisation de E_1, E_2, E_n si chaque occurrence de E est aussi une occurrence d'une et une seule entité E_1, E_2;... E_n. Les ensembles E_1, E_2, ..., E_n sont des spécialisations de l'ensemble d'entités E. Par exemple l'ensemble des VEHICULES est une généralisation de l'ensemble des AUTOMOBILES et des CYCLES. Les ensembles d'entité E_1,

E_2, E_n héritent des *attributs de E* et possèdent en outre des attributs spécifiques qui expriment leur **spécialisation**.
Exemples :
VEHICULES(*Marque, Modèle*) ; AUTOMOBILE(*Marque, Modèle***, Immatriculation, Puissance**) ; CYCLE(*Marque, Modèle*, **Type**).
EMPLOYÉ et INGÉNIEUR, SECRÉTAIRE et TECHNICIEN :chaque occurrence d'EMPLOYÉ est une occurrence d'INGÉNIEUR, de SECRÉTAIRE ou de TECHNICIEN.

Notation «EST-UN» [*IS A*] : A «EST-UN» B si l'ensemble B est une extension de A ou A un cas particulier de B.
Exemple : Base de données d'une compagnie aérienne
Ensemble d'entités EMPLOYE
Ensemble d'entités PILOTE
PILOTE «EST-UN» EMPLOYE
L'ensemble PILOTE peut ne pas avoir d'attribut mais simplement être défini par la relation avec un autre ensemble AVION, par exemple en indiquant que le pilote est qualifié sur AirBus.
La relation «EST-UN» de PILOTE vers EMPLOYE définit uniquement chaque pilote.

2.2. Association

Une association d'entités est un regroupement d'entités traduisant une certaine réalité. Comme pour les entités, on regroupe les associations de même nature en classe d'association.

Exemple : entre les entités ETUDIANT et ENSEIGNEMENT on peut considérer la classe d'association INSCRIT pour traduire le fait qu'un étudiant est inscrit à un enseignement.

INSCRIT
ETUDIANT ----------ENSEIGNEMENT

Cette association est une table ordonnée de ces ensembles d'entités

ETUDIANT				INSCRIT	ENSEIGNE MENT	
#ETUDIANT	NOM	ADRESSE	DDN.	DINSC	COURS	CODE
245	Durand Pierre	2 Rue Desroses	22/11/73	10/09/93	Algèbre1	102
1854	Perrot Marthe	5 Rue Du Bois	15/1/75	15/10/94	Algèbre1	102

Exemple : pour traduire le fait que Irène Curie est la fille de Marie Curie on pourra utiliser une association de classe A-POUR-MERE entre les deux entités représentant ces personnes. La classe d'association A-POUR-MERE peut alors être considérée comme un sous-ensemble du produit cartésien de l'ensemble d'entités PERSONNE avec lui-même.

A-POUR-MERE
PERSONNE---------- PERSONNE

A-POUR-MERE

PERSONNE		PERSONNE	
NOM	PRENOM	NOM	PRENOM
Curie	Irène	Curie	Marie

Attribut d'une association

Un attribut d'une classe d'association est une propriété qui dépend de toutes les entités intervenant dans l'association

Exemple : La classe d'association INSCRIT définie entre les classes d'entités ETUDIANT et ENSEIGNEMENT a pour attribut l'année de première inscription de l'étudiant à l'enseignement. Cette année d'inscription et attribut de l'association et non de l'une des entités, car il faut connaître l'étudiant et l'enseignement pour pouvoir la déterminer.

Type d'association

Le **type d'association** caractérise le nombre de liens autorisés entre entités. Il sert à distinguer les associations en fonction du nombre d'entités et de classes d'entités qu'elles mettent en jeu.

On distingue :

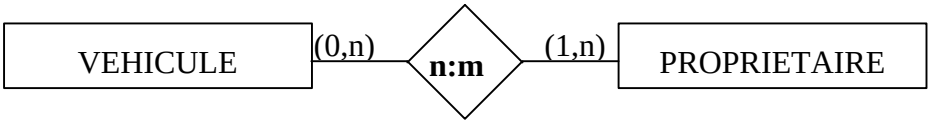
- les associations **n-aires** qui relient plus de deux entités.
- les association **binaires** qui ne relient que deux entités
  - . de **type 1:1** (ou un-à-un) si à une entité de E peut correspondre par l'association A au plus une entité de F et que, réciproquement à une entité de F ne peut correspondre au plus qu'une entité de E.
  - . de **type 1:n** (ou un-à-plusieurs) : si à une entité de E peut correspondre par l'association A plusieurs entités de F mais à une entité de F au plus une entité de E.
  - . de **type n:n** (ou plusieurs-à-plusieurs) : si à une entité de E peuvent correspondre plusieurs entités de F et réciproquement.

Cardinalité

La **cardinalité** d'un couple entité-association est définie de la manière suivante :
Etant données une classe d'entité E et une classe d'association A reliant E à une (ou plusieurs) autre(s) classes(s) d'entités, on définit *m* (respectivement *M*) le nombre minimum (respectivement maximum) d'associations de classe A pouvant exister pour une entité donnée de classe E. Alors *(m,M)* est la **cardinalité** du couple **(E,A)**.

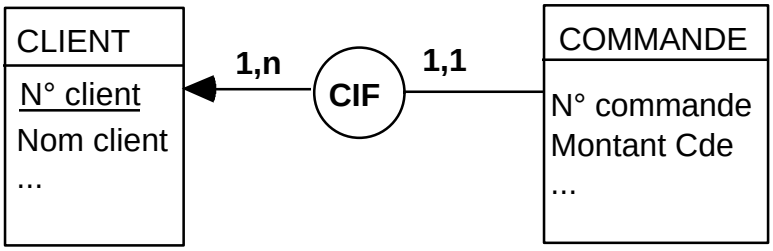
Remarques : Le minimum m peut valoir 0, 1 ou un entier strictement plus grand que 1. Le maximum M peut valoir 1 ou une valeur n>1, n n'étant souvent pas précisé de manière numérique, faute de connaissance suffisante.

Exemple :
La relation entre PROPRIETAIRE et VEHICULE est de type **n :m** car il est légalement possible d'acheter un véhicule à plusieurs, bien que ce soit peu fréquent, et une même personne puisse posséder plusieurs véhicules... Quant aux cardinalités, elles sont (0,n) et (1,n) car certains véhicules sont abandonnés (0,n) mais qu'il faut posséder au moins un véhicule pour être propriétaire (1,n) !

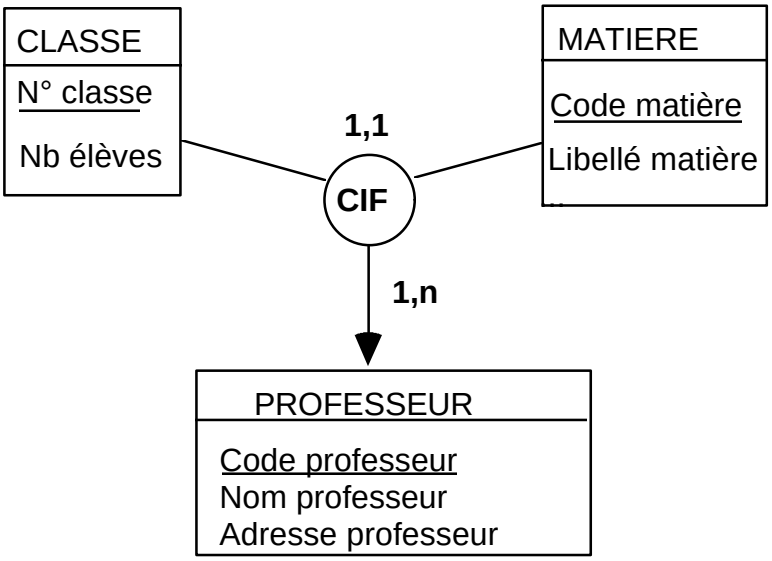


Contrainte d'identité fonctionnelle (CIF) : Quand on détermine, entre une association et une entité, *une cardinalité présentant les valeurs 0,1 ou 1,1*, l'association est particulière. On

l'appellera alors *contrainte d'identité fonctionnelle* (CIF). Cette association particulière n'est en général pas nommée. Elle indique que l'une des entités est totalement déterminée par la connaissance de l'autre ; par exemple si on connaît une commande bien précise, on connaît un client bien précis...



Une CIF qui met en relation plus de deux entités est une CIF multiple.

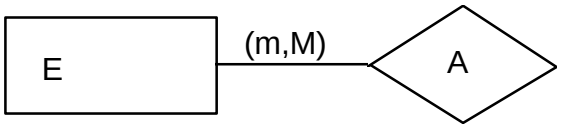


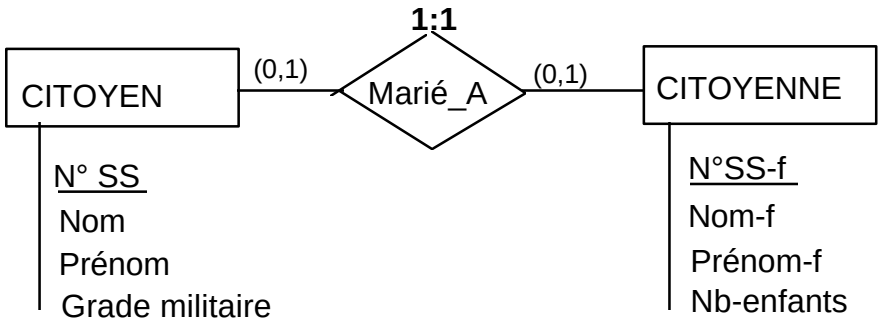
2.3. Diagramme Entité-Association

Un graphe Entité-Association (E-A) décrit la structure d'ensemble d'une base de données en combinant les objets graphiques suivants :

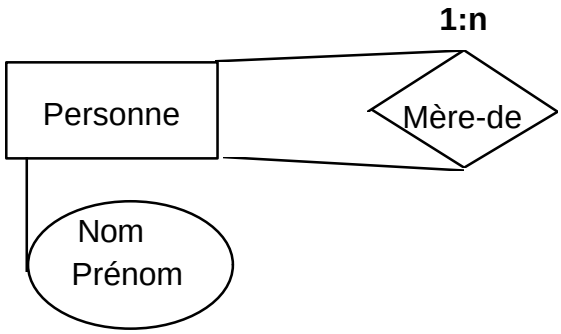
- des **rectangles** qui représentent des ensembles d'objets, c'est-à-dire des **entités** concrètes ou abstraites (par exemple : lecteur, ouvrage, compte bancaire, client...)
- des **ellipses** (soit écrits en colonnes) qui représentent des **attributs** attachés aux entités (le nom, l'adresse, le titre, la cote, numéro,...)
- des **losanges**, qui représentent des **relations** ("a emprunté", "possède le compte", "suit le cours de",...)
- des **arêtes** qui relient les attributs (ellipses) aux entités (rectangles) et les entités aux associations (losanges).

Cardinalité (m,M) d'un coupe entité (E) - association (A)

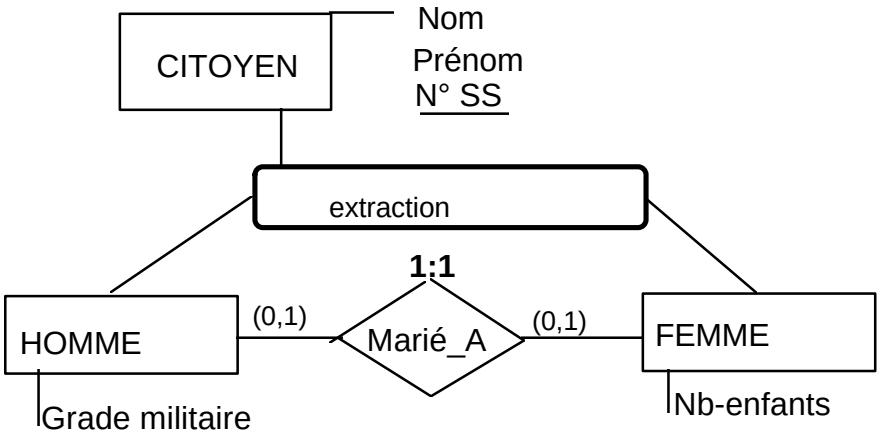




Une vision sexiste de la société...



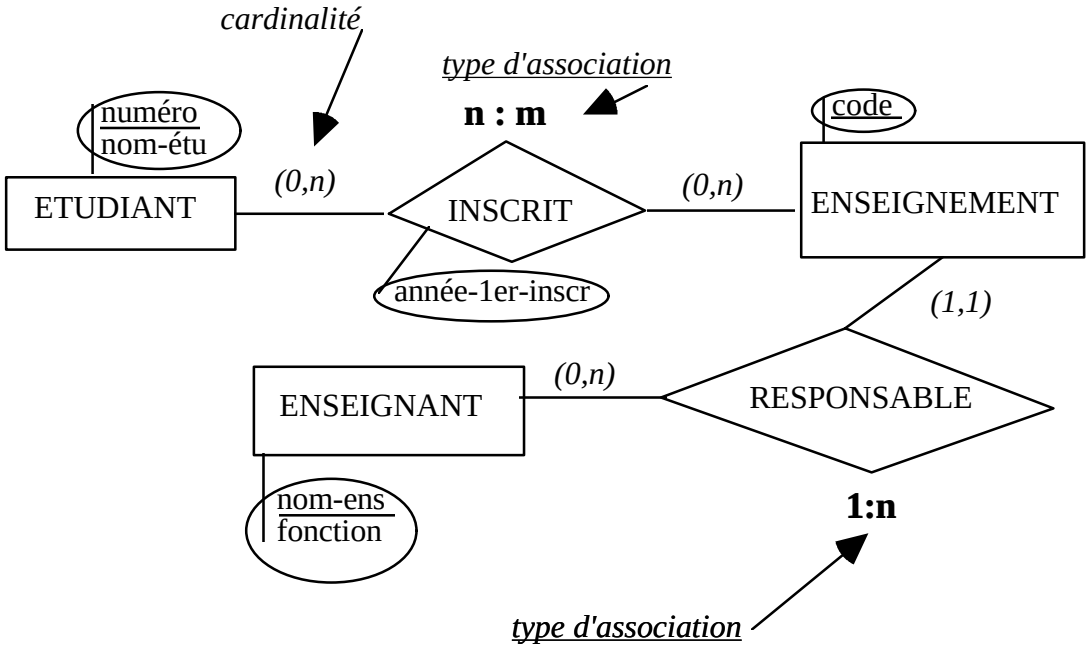
Le **domaine** d'un attribut caractérise à la fois le type de donnée (énuméré, caractère, entier, réel, chaîne de carcatères...) et l'ensemble des valeurs admissibles de celui-ci. Le domaine peut être une liste énumérée, un intervalle numérique ou une chaîne alphanumérique.
La **clé** primaire d'une classe d'entités est un attribut ou un ensemble d'attributs qui permet d'identifier explicitement chaque entité de la classe (par exemple le NUMERO_DE_LECTEUR). Dans le diagramme les clés sont soulignées.
La **restriction** consiste à prélever un sous-ensemble d'une classe d'entités d'un certain niveau pour créer une classe de niveau inférieur (par exemple classe des véhicules, restriction à la sous-classe des deux-roues).



On a extrait de l'ensemble des CITOYENS les deux sous-ensembles HOMME et FEMME...

L'**extension** (ou **agrégation**) est l'opération inverse de la restriction, qui consiste à réunir plusieurs classes de façon à créer une classe de niveau supérieur (exemple : classe des périodiques et des monographies pour créer la classe des publications).

Exemple de diagramme entité-association



Le diagramme ENTITE-ASSOCIATION ci-dessus modélise entre deux ensembles d'entités ETUDIANT et ENSEIGNEMENT une classe d'association INSCRIT qui traduit qu'un étudiant est inscrit à un enseignement.
Pour un ensemble d'entités ENSEIGNANT, il exprime qu'un enseignant est RESPONSABLE d'un enseignement (on suppose qu'il n'y a pas d'enseignants homonymes).

2.4. Dictionnaire des données

Le **dictionnaire des données** liste les entités et leurs attributs, en spécifiant le domaine de chacun ainsi que leur catégorie :

- données élémentaires (information stockée)
- données d'information déduite ou calculée d'utilisation fréquente (ce qui évite de refaire le calcul plusieurs fois) ainsi que les règles de calcul
- données calculées de type situation ou historique (total HT des commandes par mois...)
- paramètres utilisés dans des cas particuliers (TVA) ...

Il se présente sous forme d'une grille d'analyse :

Nom de la donnée	Format	Type					Règle de calcul	Contrainte d'intégrité	Document
		Elémentaire	Calculée	Paramètre	Signalétique	Situation			
Nom client	Alpha	X			X				Facture
Code postal	Num	X			X				Facture
Ville client	Alpha	X			X				Facture
...	...								...
Total HT	Num		X			X	Somme		Facture
Taux TVA	Num			X				18,60%	Facture

2.5. Règles de validation[1]

Ces règles doivent être respectées pour la cohérence du modèle Entité-Association.

- Chaque entité possède un identifiant
- Chaque propriété (attribut) d'une occurrence d'entité ne possède, au plus, qu'une valeur.
- Toutes les propriétés doivent être élémentaires.
- Toutes les propriétés autres que l'identifiant doivent dépendre pleinement et directement - de l'identifiant.
- A chaque occurrence d'une association correspond une et une seule occurrence de chaque entité participant à l'association.
- Pour une occurrence d'une association, il ne doit exister au plus qu'une valeur pour chaque propriété (attribut) de cette association.
- Chaque propriété d'une association doit dépendre pleinement et directement de tout identifiant (clé) et non pas d'une partie seulement de l'identifiant.
- Une cardinalité (0,1) ou (1,1) indique une contrainte d'intégrité fonctionnelle (CIF) et réciproquement.

[1] D'après J. GABAY «Apprendre et pratique MERISE»

3. Le modèle relationnel

Le modèle relationnel a été proposé par Codd à IBM-St-José en 1970.
Une **BD relationnelle** est une BD dont le schéma est un ensemble de schémas de **relations** et dont les occurrences sont des **tuples** ou **n-uplets** de ces relations.
Autrement dit, les entités et les associations du modèle E-A sont représentées exclusivement par des relations (des tables). Une entité est représentée par sa liste d'attributs. Une association est représentée par la liste des clés des entités qu'elle associe et ses propres attributs.

Les objets de la base sont les tuples (lignes) des tables.
Les objets sont manipulés (LMD - langage de manipulation des données) par des **opérateurs algébriques relationnels** (UNION, INTERSECTION, PRODUIT CARTESIEN, SELECTION, PROJECTION, JOINTURE, ...)

Un Système de Gestion de Base de Données (SGBD) est dit **minimalement relationnel** si :
- les informations de la base sont représentées par des **tables**
- il n'y a pas de pointeurs visibles (pour l'utilisateur) sur les tables
- le système supporte les opérateurs relationnels
  . restriction (sélection)
  . projection
  . jointure

Un SGBD est **complètement relationnel** si de plus :
- il réalise toutes les opérateurs de l'algèbre relationnel
- il y a unicité des clés (pas de doublons)
- il assure la contrainte référentielle (exemple : pouvoir s'assurer que le produit dont on a passé commande se trouve bien dans la relation PRODUIT)

3.1. Relation

Un domaine est un ensemble de valeurs.
Exemple :
Domaines D1 = D2 = {chaînes de caractères}
D3 = {entiers}.

Un attribut est une variable prenant ses valeurs dans un domaine.
Exemple :
attribut A1=NOM à valeurs dans D1;
attribut A2=ADR à valeurs dans D2;
attribut A3=NUM à valeurs dans D3;

Une relation sur les attributs A1, A2,..., An, de domaines respectifs D1, D2,..., Dn, est un sous-ensemble du produit cartésien des domaines D1, D2, ..., Dn, soit un ensemble de n-uplets de D1.D2..Dn.

Exemple :
r = {(DUPONT, PARIS, 2140)}, (DURAND, ORLY,1123), (DUBOIS, NOISY,3425)}

Représentation d'une relation

Chaque tuple (n-uplet) de la relation est écrit dans une ligne d'un tableau dont les noms des colonnes sont les attributs de la relation.
Chaque tuple est unique. Les duplications ne sont pas autorisées. L'ordre des tuples est indifférent.

Exemple :

NOM	ADR	NUM
DUPONT	PARIS	2140
DURAND	ORLY	1123
DUBOIS	NOISY	3425

Le schéma de la relation **r** est la liste des attributs de **r** avec, pour chacun, son domaine, parfois sous-entendu.

Exemple :
Le schéma de r est R = (NOM : D1, ADR : D2, NUM : D3),
écrit en abrégé R = (NOM, ADR, NUM).
On dit que r est une relation de schéma R (attention : *ne pas confondre r et R*).
Lorsqu'on repère chaque attribut d'une relation par un nom, l'ordre des colonnes n'est pas important.

Clé d'une relation

Une clé est une liste ordonnée d'attributs qui caractérise un tuple (n-uplet) de la relation.
Une clé primaire caractérise un tuple de manière unique.

Exemple : R = (NOM, ADR, NUM)

NOM	ADR	NUM
DUPONT	PARIS	2140
DURAND	ORLY	1123
DURAND	PARIS	453
DUBOIS	NOISY	3425

(NOM, ADR, NUM), (NOM, ADR) et (NUM) sont des clés.
(NOM, ADR) et (NUM) sont des clés primaires.
Par contre (NOM) ou (ADR) ne sont pas des clés à eux tout seuls.

Remarque : Avec le schéma (NOM, ADR), la modélisation ne permet pas des homonymes habitant la même ville.

3.2. Schéma de base de données relationnelle

Un schéma de base de données relationnelle B est un ensemble de schémas de relations R1, R2,..., Rp.

Une base de données b de schéma B est un ensemble de relations r1, r2, .., rp de schémas respectifs R1, R2, ..., Rp.

3.3. Passage modèle Entité-Association / modèle Relationnel.

La modélisation EA des données étant effectuée, il faut implanter la structure obtenue en machine, par exemple sous forme d'un SGBD relationnel. Nous allons donc transformer notre structure sous une forme relationnelle. On dit aussi que l'on transforme le diagramme EA en schéma relationnel.

Principes.

Principes généraux.

Traduction d'une classe d'entité
Toute classe d'entité se traduit par une table relationnelle, dont les attributs sont :
— l'identifiant de l'objet, qui forme la clé primaire du tuple correspondant.
— tous les attributs de la classe.

Traduction des liens n:1
Tout lien *n:1* se traduit par un attribut dans la table représentant la classe de départ. Cet attribut représente l'identifiant de l'objet de la classe d'entité d'arrivée. Si le lien comporte des attributs, ceux-ci sont placés dans la table de départ.

Traduction des liens 1:n
Tout lien *1:n* se traduit par un attribut dans la table représentant la classe d'arrivée. Cet attribut représente l'identifiant de l'objet de la classe de départ. Si le lien comporte des attributs, ceux-ci sont placés dans la table d'arrivée.

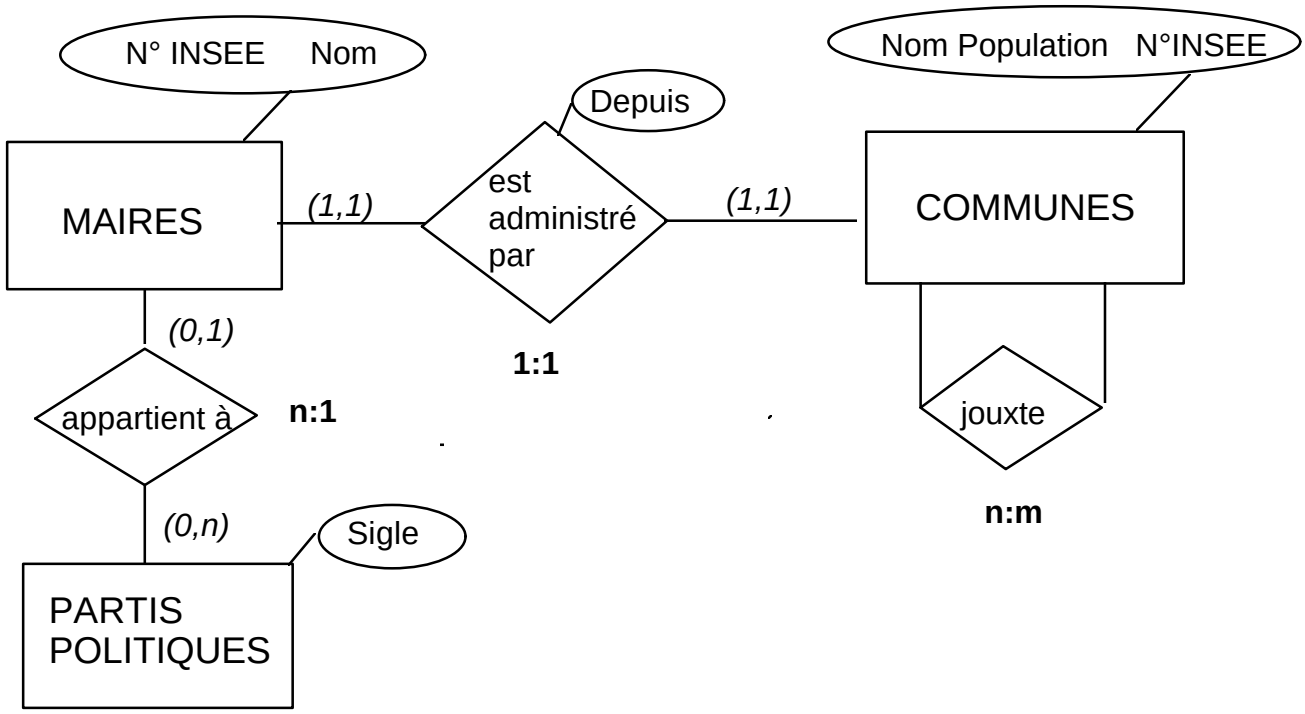
Traduction des liens n:m
Tout lien *n:m* se traduit par une table relationnelle, dont les attributs sont:
— l'identifiant de l'objet de départ.
— l'identifiant de l'objet d'arrivée.
— les attributs (éventuels) du lien.
La clé primaire est constituée des trois attributs.

Principes d'optimisation:

Il est souvent utile de minimiser le nombre de tables d'un schéma. C'est pourquoi, après l'application des principes généraux, on cherche à regrouper des tables. Ce sont généralement des tables issues de sous-classes d'une même classe. Encore faut-il qu'elles présentent un maximum d'attributs et de liens en commun.

Exemple

On considère le diagramme EA suivante:



Les liens sont de type :
— *1:1* pour "est administrée par"
— *n:1* pour "appartient à"
— *n:m* pour "jouxte".

La traduction des entités donne:

MAIRES (#INSEE-Maire, Nom)
COMMUNES (#INSEE-Commune, nom, population)
PARTIS POLITIQUES (#Sigle)

La traduction des liens

La traduction du lien *1:1* entre COMMUNES et MAIRES donne:

COMMUNES(#INSEE-Commune, nom, population, *est administrée par, depuis*)

Alternativement, "est administré par" étant *1:1*, on peut aussi le traduire par le lien inverse "administre" et l'intégrer à la relation MAIRES :

MAIRES (#INSEE-Maire, nom, *administre, depuis*)

La traduction du lien *n:1* donne :

MAIRES(#INSEE-Maire, Nom, *appartient*)

La traduction du lien *n:m* donne:

JOUXTE (#INSEE-Commune1 , #INSEE-Commune2)

Optimisation :

La table PARTIS POLITIQUES n'a qu'un seul attribut et elle n'est reliée qu'à une seule table. On peut donc envisager de la supprimer et de modifier la table MAIRES :

MAIRES (#INSEE-Maire, Nom, *administre, depuis, Parti-politique*)

Schéma final :

Il faut ensuite choisir des noms d'attributs non ambigüs et bien faire apparaître dans les intitulés les clés étrangères :

MAIRES (#INSEE-Maire, Nom-maire, #INSEE-Commune, Date-élection, Sigle-Parti)
COMMUNES (#INSEE-Commune, Nom-commune, Population)
JOUXTE (#INSEE-Commune1 , #INSEE-Commune2)

3.4. Les Langages de Manipulation de Données (LMD)

La notion de relation correspond à un certain état de la base de données. Pour passer d'un état à l'autre, on utilise des opérations agissant sur les relations. Un langage de manipulation des données (LMD) se compose d'un ensemble de commandes permettant l'interrogation de la base et d'un ensemble de commandes permettant de modifier celle-ci (insertion, mise à jour, suppression). Le LMD est souvent intégré à un langage de programmation classique appelé langage hôte afin de réaliser des transactions programmées.

Le langage relationnel a engendré le développement de langages d'interrogation assertionnels qui permettent de définir les données que l'on souhaite visualiser sans dire comment y accéder.

Il y a trois grandes classes de LMD relationnels

- Les langages algébriques, où l'expression d'un besoin d'utilisateur se fait à l'aide d'opérations dont les opérandes sont des relations.

  Issus de l'algèbre relationnelle de Codd, ils consistent en une séquence d'opérateurs sur les relations ; on peut aussi en donner une représentation graphique sous forme d'arbre.

  Exemple : le langage **SQL** (Sequel : Structured English QUEry Language)

- Les langages prédicatifs où l'utilisation de prédicats permet de sélectionner l'ensemble des tuples souhaités.

  On distingue :

  - le calcul relationnel de tuples, issus de la logique des prédicats

  Exemple : le langage **QUEL** (QUEry Language) sur système Ingres.

  - le calcul relationnel de domaines des relations

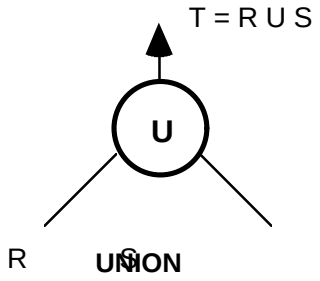
  Exemple : **QBE** (Query By Example)

3.5. Langages basés sur l'algèbre relationnelle

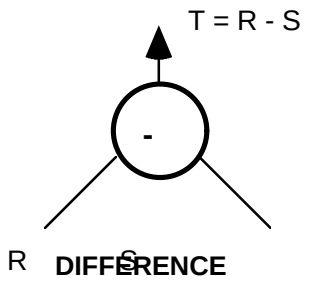
Les opérateurs de base

- opérateurs ensemblistes (binaires) **:**

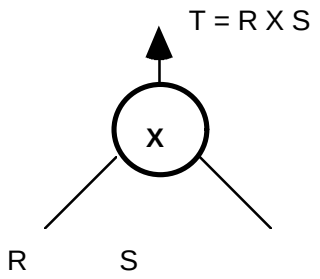
R3 = OPERATEUR(R1, R2)



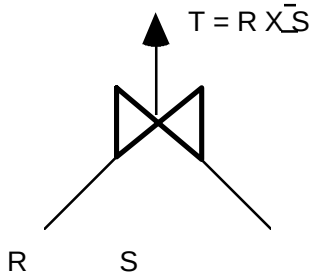
UNION



DIFFERENCE



PRODUIT CARTESIEN



JOINTURE

UNION

L'union de deux relations r et s de *même schéma* R est une relation de même schéma contenant l'ensemble des tuples appartenant à r ou à s.

Exemple :

Soient deux relations P1 et P2 de schéma PROD:

PROD(NP, NOP, QTES, COUL)

P1

NO	NOP	QTES	COUL
100	X	10	R
110	Y	15	B
120	Z	20	R

P2

100	X	10	R
130	W	10	V

P3 = **P1 ∪ P2**

100	X	10	R
110	Y	15	B
120	Z	20	R
130	W	10	V

DIFFERENCE

La différence de 2 relations r et s *de même schéma* (dans l'ordre R, S) est une relation T de même schéma contenant les tuples appartenant à r et n'appartenant pas à s.

Exemple :

P4 = P1 - P2

P4 = **P1 - P2**

110	Y	15	B
120	Z	20	R

PRODUIT CARTESIEN

Le produit cartésien de 2 relations r et s de schéma quelconque R et S, est une relation ayant pour attributs la concaténation de ceux de R et S et dont les tuples sont toutes les concaténations d'un tuple de r à un tuple de s.

Exemple :

PROD(NP, NOP, COUL) et DEPOT(ND, ADRED)

P5

100	X	B
120	Y	V

D

5	A1
7	A2

P5xD

NP	NOP	COUL	ND	ADRED
100	X	B	5	A1
100	X	B	7	A2
120	Y	V	5	A1
120	Y	V	7	A2

- opérateurs unaires

PROJECTION

La projection d'une relation r de schéma R(A1, A2,...,An) sur les attributs Ai_1, Ai_2, ..., Ai_p avec $i_j \neq i_k$ et $p<n$) est une relation r' de schéma R'(Ai_1, Ai_2, ..., Ai_p) dont les tuples sont obtenus par

élimination des valeurs des attributs de r n'appartenant pas à r' et par suppression des tuples en double.

La projection supprime des colonnes de la table initiale (et les lignes en double).

Exemple :

P6

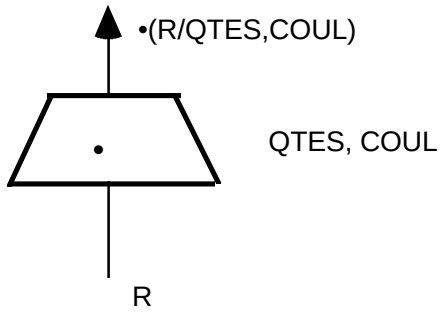
NO	NOP	QTES	COUL
100	X	10	R
110	Y	15	B
140	Z	10	R
160	W	5	V

La projection sur les attributs QTES et COUL donne

P7

QTES	COUL
10	R
15	B
5	V

L'opérateur Π est représenté graphiquement par un trapèze à base plus large



RESTRICTION ou SELECTION

La restriction d'une relation r par une qualification Q est une relation r' de même schéma R dont les tuples sont ceux de r satisfaisant la qualification Q.

La sélection (restriction) supprime des lignes dans la table initiale.

Exemple :

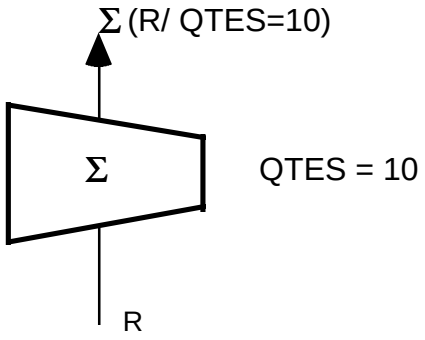
Soit P6 et Q=(QTES = 10) alors P8 = P6[Q]

La restriction donne :

P8

NO	NOP	QTES	COUL
100	X	10	R
140	Z	10	R

L'opérateur restriction est représenté par un trapèze à base gauche plus haut que la base droite.



Les opérateurs supplémentaires

Les opérateurs se déduisant des précédents et donc redondants. Ils sont utilisés en pratique parce que moins coûteux en temps de calcul que la combinaison d'opérateurs qui les définit.

JOINTURE

La jointure de deux relations r et s selon une qualification Q est l'ensemble des tuples du produit cartésien RxS satisfaisant Q. (On peut donc la programmer comme un produit cartésien suivi d'une sélection).

Exemple :

PROD(NP, NOP, COUL, ADR-FAB) et DEPOT(ND, ADR-D)

P9

NO	NOP	COUL	ADR-FAB
100	X	B	A1
110	Y	V	A2
150	Z	B	A2

D

ND	ADR-D
5	A3
7	A1
12	A2

Soit Q = (ADR-FAB = ADR-D)

$|X|_Q$ (P9,D)

NO	NOP	COUL	ADR-FAB	ND	ADR-D
100	X	B	A1	7	A1
110	Y	V	A2	12	A2
150	Z	B	A2	12	A2

Cas particuliers

EQUI-JOINTURE

de r et s de schéma R et S sur les attributs Ai et Bj : c'est la jointure selon Q= (Ai=Bj)

THETA-JOINTURE

de r et s de schéma R et S sur les attributs Ai et Bj : c'est la jointure selon Q= (Ai Θ Bj) avec Θ ∈ {<, >, , , }

AUTO-JOINTURE

de r et s de schéma R selon l'attribut Ai : c'est la jointure de r avec elle-même selon Q= (Ai = Ai)

JOINTURE NATURELLE

de r et s de schéma R et S notée r |X| s est l'équi-jointure de r et s sur tous les attributs de même nom dans R et S, suivie de la projection qui élimine les doublures de tuples.

Exemple :

PROD(NP, NOP, COUL, FAB) et PRIX(NP, PU, FAB)

NO	NOP	COUL	FAB
100	X	B	F1
110	Y	V	F2
120	Z	B	F2

PRIX

NP	PU	FAB
100	25	F1
110	30	F2
120	10	F2

Jointure naturelle de P et PRIX

NO	NOP	COUL	FAB	PU
100	X	B	F1	25
110	Y	V	F2	30
120	Z	B	F2	10

INTERSECTION

L'intersection de deux relations r et s *de même schéma* R est une relation T de même schéma contenant les tuples appartenant à la fois à r et s. On vérifiera que l'intersection s'exprime aussi par la différence.

$R \cap S = R - (R - S)$

DIVISION

Le quotient de la relation r de schéma R(A1, A2, ..., An) par la sous-relation s de schéma S(Ap+1, ..., An) est la relation q de schéma Q(A1, ..., Ap) formée de tous les tuples qui, concaténés à chacun des tuples de s donne *toujours* un tuple de r.

Soit a_i une valeur de A_i

$q = \{(a_1 , a_2 , .., a_p) / \forall (a_{p+1} .. a_n) \in s, (a_1 , .., a_p, a_{p+1}.,.,.a_n) \in r\}$

La division permet de rechercher l'ensemble de tous les sous-tuples satisfaisant une sous-relation de la relation.

On a : R/S = T-U avec T = P(R)/ A1, A2, ..., Ap) et U= Π((TxS) - R) / A1, A2, ..., Ap)

Exemple

PROD(NOP, COUL) et COUL(COUL)

X	B
Y	B
Z	V
X	R
T	J

B
R

Le quotient de PROD par COUL est NOP(NOP)

X

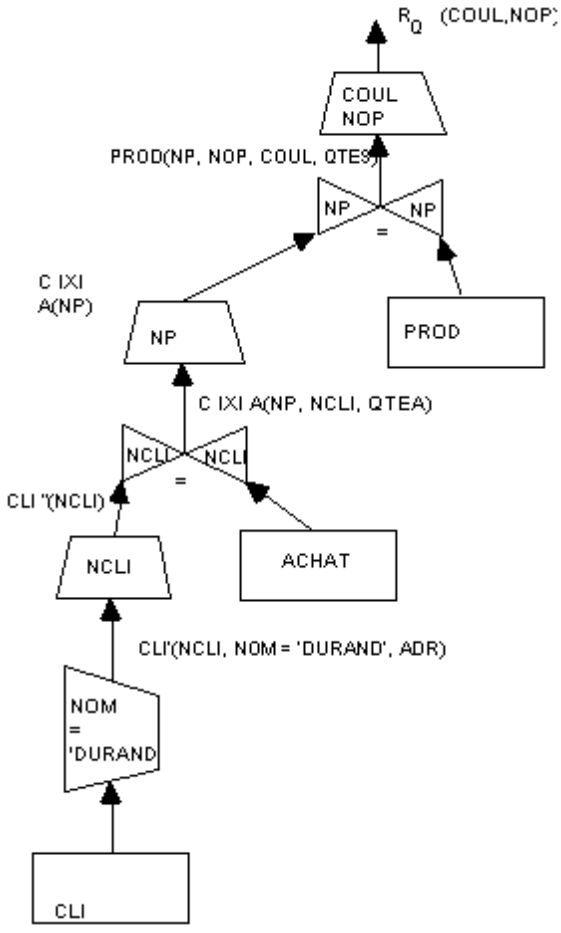
Composition d'opérateurs

Avec les opérateurs, il est possible de composer la plupart des requêtes que l'on peut faire sur une base de données relationnelle.
Les requêtes s'expriment à l'aide d'une succession d'opérations : Union, Différence, Jointure, Restriction et Projection.

Exemple :
Schéma de la Base de Données Relationnelle:

PROD(NP, NOP, COUL, QTES) ACHAT(NP, NCLI, QTEA) CLI(NCLI, NOM, ADR)

Question : «Couleur et nom des produits achetés par le client 'DURAND' ?»



4. SQL

La plupart des SGBD qui présentent les données sous forme de tables supportent un langage de requête dénommé SQL (Sequel : Structured Query Language) qui a été proposé en 1973 par une équipe de chercheurs d'IBM. La syntaxe de base du langage SQL se retrouve sur la plupart des implémentations.

4.1. Généralités

Organisation des données

Sous forme de tables [tables], de colonnes [columns], de clés primaires [primary keys] et de clés étrangères [foreign keys].

Gestion des données

Elle consiste à ajouter et retirer des lignes d'une table. Il est possible de modifier les valeurs d'une colonne dans certaines lignes d'une table. SQL permet aussi d'ajouter et de supprimer une table dans une base de données, et d'ajouter une colonne dans une table.

Accès aux données

Grâce au langage SQL, qui permet de définir les propriétés d'ensemble de données sans faire référence aux techniques d'accès. Une requête SQL produit une table, à partir d'extraits d'une ou plusieurs tables de la base de données. Les données peuvent être déduites ou calculées, et le résultat d'une requête stocké ensuite dans la base de données.

Présentation des données extraites

Elles sont affichées à l'écran ou imprimées sous forme de tables, de rapports, ordonnées ou non, selon les spécifications de l'utilisateur.

4.2. SQL langage de manipulation de données (LMD)

Tant la définition des structures de données (le schéma relationnel, les attributs des tables, les contraintes d'intégrité) que la manipulation du contenu des tables (les tuples) s'effectue au moyen de requêtes SQL. Chaque requête est la traduction d'une expression de l'algèbre relationnel. Le langage SQL peut être hébergé par un programme hôte du type Pascal, C ou Cobol.

Exemple : schéma d'une base de données Produit-Achat-Client

PRODUIT (NP, LIB, COUL, QTES, PRIX)
ACHAT(NP, NCLI, QTEA)
CLIENT(NCLI, NOM, ADR)

Projection

SELECT liste_d'attributs
FROM nom_relation **;**

Exemple : «Liste des Libellés et des Quantités par Produit»
SELECT LIB**,** QTES

FROM PRODUIT **;**

Pour éliminer les doubles il faut rajouter le mot clé **DISTINCT**

SELECT DISTINCT LIB**,** QTES
FROM PRODUIT **;**

Sélection

Une sélection est une restriction suivie d'une projection.

Restriction

SELECT *
FROM nom_relation
WHERE qualification **;**

Exemple : «Liste des Produits de Couleur 'Rouge' dont la Quantité est inférieure à 10».

SELECT *
FROM PRODUIT
WHERE COUL = 'Rouge' AND QTES < 10 **;**

Sélection

Exemple : «Liste des Numéros de produits et Libellés des Produits de Couleur 'Rouge' **ou** dont la quantité est supérieure à 10».

SELECT NP**,** LIB
FROM PRODUIT
WHERE COUL='Rouge' **OR** QTES>10 **;**

Qualification

La condition de sélection introduite par la clause WHERE peut être constituée d'une expression booléenne de conditions élémentaires (opérateurs AND, OR, NOT et parenthèses).

Une condition élémentaire peut porter sur
- un opérateur de comparaison : =, < , >, <>, <=, >=
- l'existence d'un tuple au moins : EXIST
- un quantificateur existentiel comparant une valeur à un ensemble
  - ANY : il existe un au moins
  - ALL : quel que soit , pour tout
- la présence de la valeur NULL
- l'appartenance
  - à une liste : IN
  - à un intervalle : BETWEEN
- la présence de certains caractères dans une valeur : LIKE

Tri

Il est possible de trier les résultats suivant l'ordre ascendant ou descendant d'un ou plusieurs attributs.

SELECT NP, LIB
FROM PRODUIT

WHERE COUL = ‘Rouge’ **AND** QTES < 10
ORDER BY NP **DESC**, LIB **ASC ;**

Formats et calculs

La clause SELECT peut spécifier des calculs et des formats d’affichage des données extraites par la requête.

Exemple :

```
SELECT ‘Valeur du stock de ‘,LIB,’ (en mF) :’, QTES*PRIX/1000
FROM PRODUIT;
```

Valeur du stock de Vis (en mF) : 15
Valeur du stock de Pince (en mF) : 0,4
Valeur du stock de Ciment (en mF) : 200

Jointure

Produit cartésien (jointure sans qualification)

SELECT *
FROM relation1, relation2 **;**

Exemple : «Produit cartésien des tables Produit et Achat»

SELECT *
FROM PRODUIT, ACHAT**;**

Jointure avec qualification :

- par restriction du produit cartésien

Exemple : «Liste des produits qui ont fait l'objet d'un achat»

SELECT *
FROM PRODUIT, ACHAT
WHERE PRODUIT.NP = ACHAT.NP **;**

- par requêtes imbriquées avec l’opérateur **IN** :

Exemple : «Libellé des produits qui ont été achetés»

SELECT LIB
FROM PRODUIT
WHERE NP **IN** (**SELECT** NP
FROM ACHAT) **;**

Combinaison d’opérateurs

Exemple : «Nom des clients ayant passé une commande supérieure ou égale à 10000 vis, par ordre aphabétique».

SELECT DISTINCT NOM
FROM PRODUIT, ACHAT, CLIENT
WHERE ACHAT.NCLI = CLIENT.NCLI
AND ACHAT.NP = PRODUIT.NP
AND PRODUIT.LIB = ‘Vis’
AND ACHAT.QTEA >= 10000
ORDER BY NOM **ASC ;**

Le calcul de cette requête risque de prendre énormément de temps et nécessite un important espace mémoire puisque toutes les tables PRODUIT, ACHAT, CLIENT doivent etre chragées en mémoire en même temps pour réaliser le produit cartésien. On l'accélère en utilisant des requêtes imbriquées qui limitent la taille des tables intermédiaires.

```
SELECT DISTINCT NOM
FROM CLIENT
WHERE NCLI IN
        (SELECT NCLI
        FROM ACHAT
        WHERE QTEA >= 10000
        AND NP IN
                (SELECT NP
                FROM PRODUIT
                WHERE LIB = 'Vis') )
ORDER BY NOM ASC ;
```

Union

Exemple : «Libellés des Produits de Couleur rouge OU dont la quantité est inférieure à 10».

```
SELECT LIB
FROM PRODUIT
WHERE COUL = 'Rouge'
UNION
SELECT LIB
FROM PRODUIT
WHERE QTES < 10 ;
```

Fonctions de calcul

COUNT : nombre de valeurs
SUM : somme
AVG : moyenne [average]
MAX : maximun
MIN : minimum

Exemple : «Quantité moyenne des stocks de vis»

```
SELECT AVG(QTES)
FROM PRODUIT
WHERE LIB='Vis' ;
```

Qualification plus complexe

GROUP BY liste_attributs HAVING : qualification avec fonctions de calcul
Exemple : «Numéro des produits achetés par plus de 100 clients»

```
SELECT NP
FROM ACHAT
GROUP BY NP HAVING COUNT(*) > 100 ;
```

4.3. SQL : Langage de description de données (LDD)

La description du schéma des tables et la modification des attributs s'effectue au moyen de requêtes SQL avec les mots réservés CREATE, DROP et ALTER.

Création d'une table

Cette opération crée une table (vide). Entre parenthèses le nom de chaque colonne et son domaine

Exemple : CLIENT(NCLI, NOM, ADR, TEL)

```
CREATE TABLE CLIENT (NCLI CHAR (4),
        NOM CHAR (12),
        ADR CHAR (30),
        TEL NUMERIC) ;
```

Types de données

SQL admet pour domaine des attributs différents types dont :
- NUMERIC : réel ou entier selon le format spécifié
- SMALLINT : entier court (16 bits)
- INTEGER : entier signé long (32 bits)
- FLOAT : décimal
- CHAR (n) : chaîne fixe de n caractères
- DATE : toute date
- RAW : données binaires sans format

Clés

Une clé primaire est spécifiée par les mots réservés PRIMARY KEY

```
CREATE TABLE CLIENT (NCLI CHAR (4),
        NOM CHAR (12),
        ADR CHAR (30),
        PRIMARY KEY (NCLI))
```

Une clé secondaire est spécifiée par le mot réservé UNIQUE

```
CREATE TABLE CLIENT (NCLI CHAR (4),
        NOM CHAR (12),
        ADR CHAR (30),
        PRIMARY KEY (NCLI),
        UNIQUE (NOM, ADR)) ;
```

Contrainte de référence

Une référence à une clé étrangère est spécifiée par le mot réservé FOREIGN KEY (clé) REFERENCES (table.attribut)

Exemple : ACHAT(NP, NCLI, QTEA)

```
CREATE TABLE ACHAT (NP INTEGER,
        NCLI CHAR (4),
        QTEA FLOAT,
        FOREIGN KEY (NP) REFERENCES (PRODUIT.NP),
        FOREIGN KEY (NCLI) REFERENCES (CLIENT.NCLI)) ;
```

Caractère facultatif / obligatoire d'une colonne

Par défaut toute colonne est facultative. Le caractère obligatoire d'une colonne se déclarera par la clause NOT NULL :

Exemple : PRODUIT (NP, LIB, COUL, QTES) :

```
CREATE TABLE PRODUIT (NP INTEGER NOT NULL,
        LIB CHAR(10) NOT NULL,
        COUL CHAR(6),
        QTES INTEGER NOT NULL,
        PRIMARY KEY (NP) )
```

Suppression d'une table

Toute table peut être supprimée.
Exemple : **DROP TABLE** CLIENT

Ajout et retrait d'une colonne

La commande suivante ajoute la colonne PRIX à la table PRODUIT :
ALTER TABLE PRODUIT **ADD** PRIX INTEGER
Une colonne peut être supprimée par
ALTER TABLE PRODUIT **DROP** PRIX

4.4. SQL pour les mises à jour (LMD)

Les clause INSERT, DELETE, UPDATE permettent l'ajout et la suppression de tuples dans une table existante.

Insérer un tuple

Exemple :
INSERT INTO PRODUIT <200, 'Pince', 'Rouge', 200, 25.50>

Supprimer un tuple

Exemple : supprimer tous les produits achetés par Martin :

```
DELETE PRODUIT
WHERE 'Martin' IN
        (SELECT NOM
        FROM ACHAT, CLIENT, PRODUIT
        WHERE PRODUIT.NP = ACHAT.NP
        AND ACHAT.NCLI = CLIENT.NCLI) ;
```

Modifier un tuple

Exemple : Mettre la quantité à 0 pour tous les clients d'adresse 'Noisy-Le-Grand 93160'

```
UPDATE ACHAT
SET QTEA = 0
WHERE 'Noisy-Le-Grand 93160' IN
        (SELECT ADR
        FROM CLIENT, ACHAT
        WHERE ACHAT.NCLI = CLIENT.NCLI) ;
```

5. Dépendances fonctionnelles et décomposition en formes normales

Le choix des relations est primordial quand on définit un schéma relationnel. Il est guidé par les dépendances entre les données, c'est-à-dire les contraintes que les données de la base doivent vérifier.

5.1. Intégrité et dépendance fonctionnelle

Considérons une base de données clients - fournisseurs décrite par les relations suivantes :

PRIX_FOURN (FNOM, FADRESSE, PNOM, COUTS)
COMMANDES (NUM_COMDE, NOM, PNOM, QUANTITE)
CLIENTS (NOM, ADRESSE_C, BALANCE)

Les problèmes que soulève ce schéma :
- **redondance** : l'adresse d'un fournisseur est répétée dans plusieurs tuples. Il n'est pas justifié de spécifier l'adresse du fournisseur avec le prix du produit, car ces information n'ont rien à voir. Si un fournisseur a 1000 produits à son catalogue, son adresse est répétée 1000 fois dans la base...
- **anomalies de mise à jour** : c'est une conséquence de la redondance ; il est possible de modifier l'adresse d'un fournisseur dans un tuple sans le faire dans tous. Si l'adresse d'un fournisseur change il faut remettre à jour tout son catalogue de prix...
- **anomalies d'insertion et de suppression** : on ne peut entrer l'adresse d'un fournisseur si on n'a pas la connaissance d'au moins un produit qu'il peut livrer, ainsi que son prix. Avec la suppression du catalogue de prix d'un fournisseur, toute référence à celui-ci disparaît...

Une modification du schéma relationnel clients-fournisseurs permet d'éviter ces difficultés :

FOURNISSEURS (FNOM, FADRESSE)
PRIX(FNOM, PNOM, COUT)
COMMANDES (NUM_COMDE, NOM, PNOM, QUANTITE)
CLIENTS (NOM, ADRESSE_C, BALANCE)

Il est évident que la connaissance de FNOM d'un fournisseur entraîne la connaissance de son adresse FADRESSE. C'est ce type de dépendance qu'on doit exploiter...
Les contraintes suivantes seront exprimées par des dépendances fonctionnelles

- "Un fournisseur n'a qu'une adresse"
- "Un produit délivré par un fournisseur unique a un coût unique"
- "Un numéro de commande détermine le nom du client, le nom du produit et la quantité commandée"

Relation FOURNISSEURS :
FNOM --> FADRESSE
Relation PRIX :
FNOM PNOM --> COUTS
Relation COMMANDES :
NUM_COMDE --> NOM PNOM QUANTITE
NOM PNOM --> NUM_COMDE QUANTITE
Relation CLIENTS
NOM --> ADRESSE_C BALANCE

Définition d'une dépendance fonctionnelle

Définition

Soit R(*A1, A2, A3,..., An*) un schéma de relation et *X* et *Y* deux sous-ensembles de {*A1,A2,...,An*}. On dit qu'il y a une dépendance fonctionnelle de *Y* sur *X* et on écrit *X* --> *Y* si, quel que soit un exemplaire acceptable, r, de R, alors tous les tuples *u* et *v* de R qui ont les mêmes composantes dans *X*, ont aussi les mêmes composantes dans *Y*.

Exemple

On considère le schéma relationnel R(A,B,C,D) et la table

	A	B	C	D
1	a	b	c	d
2	a	b	c	d'
3	a'	b	c	d'

- L'ensemble des parties de R a 16 éléments (2^n)

P(R) = { {}, {A}, {B}, {C}, {D}, {A,B}, {A,C}, {A,D}, {B,C}, {B,D}, {C,D}, {A,B,C}, {A,B,D}, {A,C,D}, {B,C,D}, {A,B,C,D} }.

- L'ensemble des dépendances fonctionnelles sur R :

F(R) = {{A}->{B}, {A}-->{C}, {A,B}-->{C}, {A, C}-->{B}, {D}-->{B}, {D}-->{C}, {D,B}-->{C}, {D,C}-->{B} }

On calcule ces dépendances en vérifiant par exemple que {A,B}-->{C} car les projections selon (A,B,C) des tuples 1,2 sont identiques et la projection de 3 est unique.

Par contre **on n'a pas** {A,B}-->{D} car les projections selon (A,B,D) des tuples 1,2 sont différentes sur l'attribut D alors qu'elles sont identiques sur les attributs A et B.

Exemple :

Soit le schéma relationnel R(PROF, CODMAT, J, H, SALLE)

Un tuple (p,m,s,j,h) d'une relation r de schéma R signifie "L'enseignant p enseigne la matière m dans la salle s le jour j à l'heure h".

Dans la mesure ou un enseignant ne peut se trouver dans deux salles à la fois un jour donné à une heure donnée pour un cours donné, et que deux cours ne peuvent être donnés simultanément par la même personne dans la même salle un jour et une heure donnée, on a les dépendances :

df1 : PROF,H,J --> SALLE, CODMAT

df2 : H,J,SALLE --> PROF, CODMAT

Si de plus la connaissance d'un enseignant implique la connaissance de la matière enseignée on a

df3 : PROF --> CODMAT

Mais si un professeur peut enseigner plusieurs matières

PROF -/-> CODMAT

REMARQUE

Une dépendance fonctionnelle est une propriété qui s'applique à un ensemble de relations ; elle ne peut être déduite d'une seule table car c'est une propriété du schéma, une assertion sur la réalité qui ne peut donc être prouvée !

Clé et dépendance fonctionnelle

Définition

Soit R(*A1, A2, A3,..., An*) un schéma de relation.
Soient *X* un sous-ensembles de {*A1,A2,...,An*}
On dit que X est une clé pour R si :

1) *X* --> *A1,A2,...,An*
2) Quel que soit *Y* inclus dans *X*,
   si *Y* --> *A1,A2,...,An* alors *X = Y.*

En clair cela signifie que tous les attributs de la relation sont en dépendance de la clé.

Le point 1 exprime l'unicité des tuples
Le point 2 traduit qu'une clé est tout ensemble d'attributs minimal impliquant l'unicité des tuples.

Si un attribut est clé d'une relation, alors tous les attributs de cette relation sont en dépendance fonctionnelle de la clé

Clé primaire et clé candidate

Si une relation a plusieurs clés, on peut en choisir une, appelée clé primaire; les autres clés sont dites clés candidates.

Exemple :
Relation COMMANDES (NUM-COMDE, NOM, PNOM, QUANTITE)
avec les dépendances :

NUM-COMDE --> NOM PNOM QUANTITE
NOM PNOM --> NUM-COMDE QUANTITE

La clé primaire NUM-COMDE
Une clé candidate NOM PNOM

5.2. Conséquence logique et clôture

Intuitivement, l'existence d'une dépendance fonctionnelle peut avoir pour conséquence d'autres dépendances fonctionnelles. Par exemple, si on a une dépendance fonctionnelle X --> Y et que X**C**Z (X inclus dans Z) alors la connaissance des attributs de Z implique la connaissance de Y, c'est-à-dire Z --> Y.

Conséquence logique (|=)

Définition

Soit *F* un ensemble de dépendances fonctionnelles pour un schéma de relation R, et *X --> Y* une dépendance fonctionnelle.
On dit que *X --> Y* est la conséquence logique de *F*, et on l'écrit

F |= *X --> Y*,

si tout exemplaire de relation r de R qui satisfait les dépendances de *F* satisfait aussi

X --> Y.

Clôture (ou fermeture) d'un ensemble de dépendances

La notion de clôture permet de prendre en compte toutes les dépendances fonctionnelles conséquences logiques d'un ensemble de dépendances données.

Définition

On appelle clôture de *F* (ou fermeture de *F*), l'ensemble

$F^+ = \{X \text{ --> } Y \ / \ F \models X \text{ --> } Y\}$

Dépendances fonctionnelles élémentaires, transitives, directes

Certaines dépendances sont plus fondamentales que d'autres puisque les autres peuvent s'en déduire...

Définition

On appelle dépendance fonctionnelle **élémentaire** une dépendance fonctionnelle de la forme

X --> *A*

où *A* est un attribut unique non inclus dans *X*,
telle que, quel que soit *Y* inclus dans *X*, la dépendance *Y* --> *A* **IMPLIQUE** *X = Y*.

Une dépendance fonctionnelle élémentaire **n'est donc pas obtenue par augmentation** de son membre gauche.

Définition

On appelle dépendance fonctionnelle **transitive** une dépendance fonctionnelle de la forme

X --> *A*

où A est est un attribut unique non inclus dans X,
telle que'il existe Y non inclus dans X ,

X --> *Y* **ET** *Y* --> *A* **ET** *Y* -/-> *X*

Définition

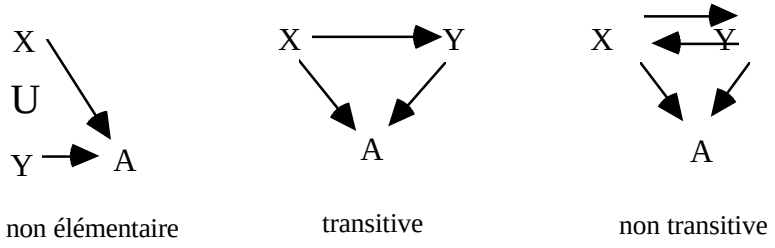
On appelle dépendance fonctionnelle **directe** une dépendance fonctionnelle de la forme

X --> *A*,

où A est un attribut unique non inclus dans X,
telle que quel que soit Y non inclus dans X,

X --> *Y* **ET** Y -> A **IMPLIQUE** *Y* --> *X*.

En d'autres termes une dépendance est directe si elle n'est pas obtenue par transitivité à partir d'autres dépendances.



non élémentaire transitive non transitive

Schémas de dépendances de A sur X

Calcul de la clôture d'un ensemble de dépendances

La notion de clôture a été introduite pour pour prendre en compte l'ensemble de toutes les dépendances fonctionnelles conséquences logique d'un ensemble de dépendances données. On obtient la clôture en itérant les règles d'Amstrong.

Le calcul de F+ peut être très long mais il peut être remplacé par le calcul de

$X+ = \{A \ / \ A \in U \wedge X \rightarrow A\}$

au moyen de l' itération suivante :

$X = X$

$X^{(i+1)} = X^{(i)} \bigcup \{A \ / \ \exists\ Y \rightarrow Z \in F$ tel que $Y \subset X^{(i)}$ et $A \in Z\}$

Règles d'Amstrong

Soient R(A,B,...) et F= {df1,...} l'ensemble des dépendances fonctionnelles sur R, et X, Y,.. des éléments de l'ensemble des parties de {A,B,...}.

a) Réflexivité

Si Y inclus dans X alors X-->Y.

b) Augmentation

Pour tout Z inclus dans {A,B,...}
si X-->Y alors XZ-->YZ.
et
si X-->Y alors XZ-->Y.

c) Transitivité

Si X-->Y et Y-->Z alors X-->Z.

Théorème:

Si X-->Y se déduit de F en appliquant les règles a, b, c
alors X-->Y appartient à F^+, la fermeture (clôture) de F.

Réciproquement, toute dépendance fonctionnelle X-->Y de F^+ se déduit de F par application des règles d'Amstrong.

Propriétés complémentaires des règles d'Amstrong

Ces règles se déduisent des règles d'Amstrong.

d) Additivité / union

X-->Y, X-->Z alors X-->YZ.

e) Pseudo-transitivité

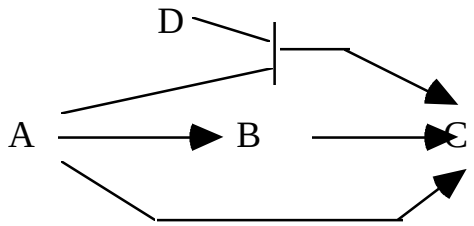
X-->Y, WY-->Z alors XW-->Z.

f) Décomposition

X-->Y alors X-->Z si Z est inclus dans Y.

Représentation graphique de dépendances fonctionnelles (df)

On considère l'ensemble F = {A-->B, B-->C, AD-->C, A-->C}
Le graphe des dépendances fonctionnelles de F est donné par :



Clé minimale d'un schéma de relation et fermeture de df

On considère la relation ADR (RUE, VILLE, CODEPOSTAL)
muni des dépendances fonctionnelles

F : df1 : RUE, VILLE --> CODEPOSTAL
df2 : CODEPOSTAL --> VILLE

Définition :
On dit que X est une clé de R muni de F si une des conditions équivalentes suivantes est satisfaite

1) X-->R appartient à F+
2) F |= X-->R
3) Toute relation r sur R qui satisfait F vérifie X-->A,B,C...

Montrons que (RUE, CODE POSTAL) et (RUE, VILLE) sont des clés pour ADR.

(RUE, CODEPOSTAL) est une clé de ADR car :
CODEPOSTAL --> VILLE est dans F
Par augmentation (CODEPOSTAL, RUE) --> (VILLE, RUE)
Par augmentation (CODEPOSTAL, RUE) --> (VILLE, RUE, CODEPOSTAL)
Donc (RUE, CODEPOSTAL)--> ADR

De même (RUE,VILLE) --> CODEPOSTAL est dans F
Par augmentation (RUE, VILLE) --> (RUE, VILLE, CODEPOSTAL)
Donc (RUE, VILLE)-->ADR.

Clé minimale

Un ensemble d'attributs X de R est une **clé minimale** de R muni de l'ensemble de dépendances fonctionnelles F si et seulement si X est une clé et si tout sous-ensemble strict de X n'en est pas une.
(RUE, CODE POSTAL) et (RUE, VILLE) sont des clés minimales pour ADR.

5.3. Couverture minimale d'un ensemble de dépendances

Pour tout ensemble de dépendances, F il existe un ensemble équivalent F' (au sens $F^+ = F'^+$) qui soit **minimal**.

Couverture minimale de F

On appelle couverture minimale de F un ensemble G de dépendances fonctionnelles tel que :

1) $G^+ = F^+$
2) Pour aucune dépendance fonctionnelle X-->A de G on n'a
   G-{X--A} |= G
3) Pour aucune dépendance fonctionnelle X-->A de G on n'a
   G |= (G-{X-->A}) **U** Y-->A

Il n'y a pas unicité de la couverture minimale en général.

Conditions de minimalité

1- $\forall X \rightarrow Y \in F$, Y ne comporte qu'un seul attribut,

2 - $\forall X \rightarrow A \in F$, l'ensemble $F - \{X \rightarrow A\}$ n'est pas équivalent à F,

3 - $\forall X \rightarrow A \in F$ et $Z \subset X$, l'ensemble

$(F - \{X \rightarrow A\}) \cup \{Z \rightarrow A\}$ n'est pas équivalent à F.

5.4. Formes normales

Les relations sont classifiées en fonction de leurs propriétés vis-à-vis des dépendances fonctionnelles à l'aide de la notion de forme normale.
Plus le degré de normalité est élevé, plus les anomalies de mise-à-jour sont réduites.

Première forme normale (1FN)

Les relations sont des ensembles d'attributs "atomiques", c'est-à-dire indécomposables.

Deuxième forme normale (2FN)

Un schéma de relation est dit en deuxième forme normale (2FN) si elle est en 1FN et si tout attribut qui n'appartient à aucune des clés minimales du schéma ne dépend d'aucun sous-ensemble de clés du schéma.

Autrement dit il faut éviter la configuration suivante :

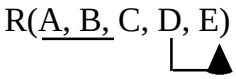
R(A, B, C, D, E)

Si D n'appartient pas à une clé minimale de R
alors il ne faut pas que D dépende d'une partie
de la clé.

Troisième forme normale (3FN)

Un schéma de relation est dit en troisième forme normale (3FN) si elle est en 2FN et si tout attribut qui n'appartient à aucune des clés minimales du schéma ne dépend que des clés du schéma et des ensembles d'attributs qui le contiennent.

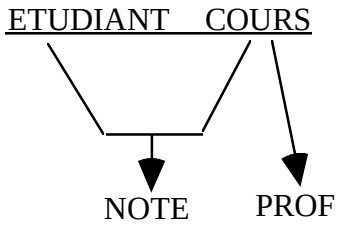
Autrement dit il faut éviter la configuration suivante :



Cette configuration n'est pas 3FN !

Soit la relation de schéma R(COURS, ETUDIANT, NOTE, PROF)
muni de F = {{COURS}-->{PROF}, {COURS,ETUDIANT}-->{NOTE}}

Montrons que R n'est pas en 3NF.



R n'est pas en 3NF, en effet :

- R admet comme seule clé minimale {COURS,ETUDIANT}
- PROF n'appartient pas à cette clé et dans {COURS}-->{PROF}, {COURS} n'est pas une clé.

Pour savoir si un schéma est en 3FN , on doit :

(a) chercher toutes les clés minimales

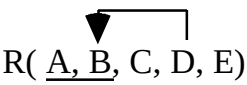
(b) en déduire les attributs A qui n'appartiennent à aucune clé minimale

(c) regarder toutes les df X-->A de F+, avec A déterminé au (b), A non inclus dans X et tester pour chacune d'elle si X est une clé.

Remarque : Si tous les attributs de R appartiennent à une clé minimale de R, alors R est en 3FN.

Forme normale de Boyce-Codd (FNBC)

Une relation est FNBC si elle est en 3FN et si tout attribut ne dépend que des clés minimales du schéma et des ensembles le contenant.
Autrement dit il faut éviter la configuration :



Ce schéma n'est pas en FNBC !

Exemple :
Montrer que le schéma relationnel ADR(VILLE, RUE, CODEPOSTAL)
muni de F ={(RUE, VILLE) --> CODEPOSTAL, (CODEPOSTAL)--> VILLE}

n'est pas en FNBC.

(a) Recherchons les clés minimales :
(RUE, CODEPOSTAL) est une clé de ADR car :
CODE POSTAL --> VILLE est dans F
Par augmentation (CODEPOSTAL, RUE) --> (VILLE, RUE)
Par augmentation (CODEPOSTAL, RUE) --> (VILLE, RUE, CODEPOSTAL)
Donc (RUE, CODEPOSTAL)--> ADR

De même RUE,VILLE --> CODEPOSTAL est dans F
Par augmentation RUE, VILLE -->RUE,VILLE,CODEPOSTAL
Donc RUE,VILLE-->ADR.

Les clés minimales sont donc (RUE,VILLE) et (RUE,CODEPOSTAL)
et la remarque "si tous les attributs de R appartiennent à une clé minimale, alors R est en 3FN" s'applique...

(b) Mais ADR n'est pas en FNBC car on trouve la dépendance
{CODEPOSTAL}--> {VILLE} alors que CODEPOSTAL n'est pas une clé.

Propriétés des formes normales

- Si tous les attributs de R appartiennent à une clé minimale de R, alors R est en 3FN.
- Un schéma en 3FN qui n'admet qu'une seule clé est en FNBC
- Un schéma muni d'une seule dépendance fonctionnelle
  X-->Y avec X **U** Y = R est en FNBC.

5.5. Dépendance multivaluée

X et Y sont en dépendance multivaluée X-->>Y dans le schéma relationnel R si pour toute relation r sur R, à chaque valeur de X est associé un ensemble de valeurs $\{y_i\}$ de Y, indépendamment des autres attributs de R.
Autrement dit :
X-->>Y/Z est équivalent à

$$\forall r \in R \ ((x,y,z) \in r \text{ et } (x',y',z') \in r$$
$$\Rightarrow \ (x,y',z) \in r \text{ et } (x,y,z') \in r)$$

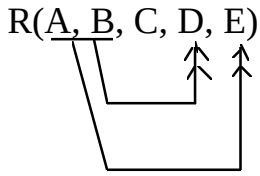
Ce qui en clair signifie que la dépendance multivaluée exprime une **indépendance** entre deux objets Y et Z reliés à un troisième objet X.

Exemple : Soit la relation R(ETUDIANT, LIVRE, NOTE)
La dépendance multivaluée
ETUDIANT-->>LIVRE/NOTE
signifie qu'un ETUDIANT peut avoir plusieurs livres indépendamment du fait d'avoir des notes et inversement.

Quatrième forme normale (4FN)

R est en 4FN si R est FNBC et s'il n'existe pas de dépendance multivaluée non triviale autre que les dépendances fonctionnelles.

Autrement dit il faut éviter la configuration :



Ceci n'est pas en 4FN !

5.6. Décomposition en formes normales

Dans l'exemple choisi au début de ce chapitre :
PRIX_FOURN (FNOM, PNOM, FADRESSE, COUTS)
les dépendances fonctionnelles sont :
FNOM --> FADRESSE
(FNOM PNOM) --> COUTS.
La clé de cette relation est (FNOM, PNOM).

La difficulté provient de ce que la partie gauche de la première dépendance ne contient pas la clé de la relation ; comme il peut y avoir plusieurs tuples qui ont le même fournisseur, il y a redondance de son adresse.
Par contre si on décompose en relations FOURNISSEURS (FNOM, FADRESSE) et PRIX(FNOM, PNOM, COUTS), la partie gauche de la première dépendance fonctionnelle est une clé de la relation FOURNISSEURS et ce n'est pas une dépendance fonctionnelle de la relation PRIX.

Décomposition sans perte d'un schéma de relation

Pour résoudre les difficultés inhérentes à certaines relations, il est intéressant de pouvoir les transformer pour obtenir un autre ensemble de relations qui soit équivalent. Deux propriétés doivent être conservées lors de cette décomposition : la jointure et les dépendances fonctionnelles, c'est-à-dire les contraintes d'intégrité.

On appelle décomposition d'un schéma de relation R(A,B,...) le remplacement de R par un ensemble de schémas de relations $R_1;R_2,...R_p$ obtenus à partir de R par projection et de telle sorte que la réunion des attributs des R_i soit égale à R.
Une décomposition est sans perte d'information (SPI) si toutes les relations r sur R considérées sont égales à la jointure des relations r_i obtenues par projection de r sur les schémas de R_i.

Théorème:

Soit R(X,Y,Z) et X-->Y dans F. Alors la décomposition de R en S(X,Y) et T(X,Z) est sans perte d'information.
Réciproquement, si la décomposition de R en S et T est sans perte d'information, alors X-->Y ou X-->Z appartient à F+

La décomposition doit conserver les dépendances fonctionnelles.

On dit que la décomposiotion de R en $R_1;R_2,...R_p$ préserve les df (est sans perte de df - SPD) si la fermeture de la réunion des F_i est égale à F+.
En général, la réunion des F_i est différente de F.

Décomposition avec perte de dépendance

Si le schéma **R** = {X, Y, Z} n'est pas en 3FN et si la df: X-->Y appartient à l'ensemble **F** des df imposées sur **R,** il faut remplacer **R** par **R1**=(X,Y), muni de la df: X-->Y, et par **R2**=(X,Z) muni des df de **F+** qui s'écrivent avec des attributs de **R2**.

5.7. Conclusion

- Les dépendances fonctionnelles sont des propriétés provenant du monde réel qui indiquent les exemplaires des relations qui sont acceptables.
- La donnée d'un ensemble de dépendances fonctionnelles induit d'autres dépendances, conséquences logiques des premières.
- Les formes normales sont des propriétés des relations vis-à-vis des dépendances fonctionnelles qui mesurent leur redondance potentielle :
  - la FNBC caractérise des relations sans redondance
  - le 3FN limite ces redondances
- La décomposition d'un schéma relationnel
  - en FNBC : préserve la jointure mais pas toujours les dépendances
  - en 3FN : préserve dépendance et jointure.

6. Bibliographie

ACSIOME, Modélisation dans la conception des systèmes d'information, Masson, 1989
C. Carrez, Des structures aux bases de données, Dunod, 1990
G. Gardarin, Les bases de données, les systèmes et leurs langages, Eyrolles.
GALACSI, Les systèmes d'information, Analyse et conception, Dunod, 1984
GALACSI, Conception des bases de données : du schéma conceptuel au schéma physique, Dunod, 1989
J. Gabay, Apprendre et pratiquer MERISE, Masson, 1991
P.-A. Goupille, J.-M. Rousse, Analyse informatique pour les IUT et BTS, Masson 1993
J.-L. Hainaut, Bases de données et modèles de calcul, Outils et méthodes pour l'utilisateur, InterEditions, 1994
H.F. Korth, A. Silberschatz, Database System Concepts, Mac Graw-Hill, (1986)
J.P. Matheron, Comprendre MERISE, Eyrolles, 1989
J.D. Ullman, Principles of Database and Knowledge Base System, Vol. 1, Computer Science Press (1988)

7. Table des matières