Livret - 12

Principes des bases de données

SGBD relationnel, algèbre relationnelle

SQL

RM di scala

Cours informatique programmation

Rm di Scala - http://www.discala.net

12 : Principes des bases de données

Plan du chapitre:

1. Introduction et Généralités

1.1 Notion de système d'information

L'informatique est une science du traitement de l'information, laquelle est représentée par des données Aussi, très tôt, on s'est intéressé aux diverses manières de pouvoir stocker des données dans des mémoires auxiliaires autres que la mémoire centrale. Les données sont stockées dans des périphériques dont les supports physiques ont évolué dans le temps : entre autres, d'abord des cartes perforées, des bandes magnétiques, des cartes magnétiques, des mémoires à bulles magnétiques, puis aujourd'hui des disques magnétiques, ou des CD-ROM ou des DVD.

La notion de fichier est apparue en premier : le fichier regroupe tout d'abord des objets de même nature, des enregistrements. Pour rendre facilement exploitables les données d'un fichier, on a pensé à différentes méthodes d'accès (accès séquentiel, direct, indexé).

Toute application qui gère des systèmes physiques doit disposer de paramètres sémantiques décrivant ces systèmes afin de pouvoir en faire des traitements. Dans des systèmes de gestion de clients les paramètres sont très nombreux (noms, prénoms, adresse, n°Sécu, sport favori, est satisfait ou pas,..) et divers (alphabétiques, numériques, booléens, ...).

Dès que la quantité de données est très importante, les fichiers montrent leurs limites et il a fallu trouver un moyen de stocker ces données et de les organiser d'une manière qui soit facilement accessible.

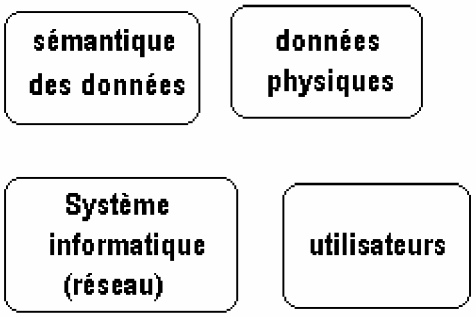
Base de données (BD)

> Une BD est composée de données stockées dans des mémoires de masse sous une forme structurée, et accessibles par des applications différentes et des utilisateurs différents. Une BD doit pouvoir être utilisée par plusieurs utilisateurs en "même temps".

Une base de données est structurée par définition, mais sa structuration doit avoir un caractère universel : il ne faut pas que cette structure soit adaptée à une application particulière, mais qu'elle puisse être utilisable par plusieurs applications distinctes. En effet, un même ensemble de données peut être commun à plusieurs systèmes de traitement dans un problème physique (par exemple la liste des passagers d'un avion, stockée dans une base de données, peut aussi servir au service de police à vérifier l'identité des personnes interdites de séjour, et au service des douanes pour associer des bagages aux personnes….).

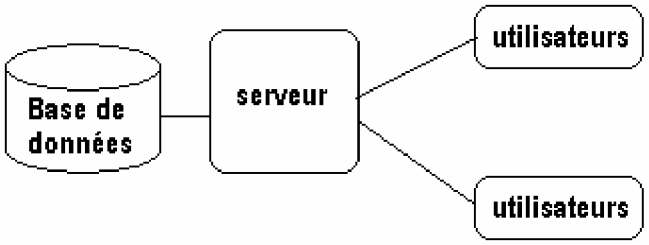
Système d'information

> Dans une entreprise ou une administration, la structure sémantique des données, leur organisation logique et physique, le partage et l'accès à de grandes quantités de données grâce à un système informatique, se nomme un système d'information.



Les entités qui composent un système d'information

L'organisation d'un SI relève plus de la gestion que de l'informatique et n'a pas exactement sa place dans un document sur la programmation. En revanche la cheville ouvrière d'un système d'information est un outil informatique appelé un **SGBD** (système de gestion de base de données) qui repose essentiellement sur un système informatique composé traditionnellement d'une **BD** et d'un réseau de postes de travail consultant ou mettant à jour les informations contenues dans la base de données, elle-même généralement située sur un ordinateur-serveur.



Système de Gestion de Base de Données (SGBD)

Un SGBD est un ensemble de logiciels chargés d'assurer les fonctions minimales suivantes :

- Le maintien de la cohérence des données entre elles,
- le contrôle d'intégrité des données accédées,
- les autorisations d'accès aux données,
- les opérations classiques sur les données (consultation, insertion , modification, suppression)

La cohérence des données est subordonnée à la définition de contraintes d'intégrité qui sont des règles que doivent satisfaire les données pour être acceptées dans la base. Les contraintes d'intégrité sont contrôlées par le moteur du SGBD :

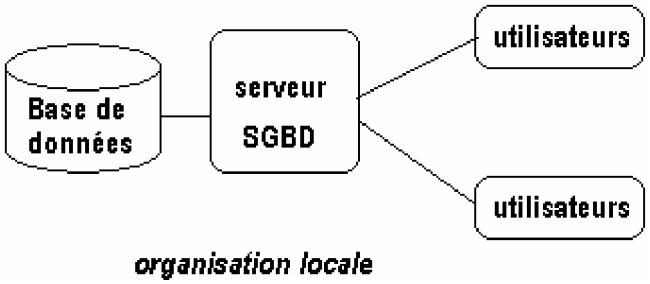
- au niveau de chaque champ, par exemple le : prix est un nombre positif, la date de naissance est obligatoire.
- Au niveau de chaque table - voir plus loin la notion de clef primaire : deux personnes ne doivent pas avoir à la fois le même nom et le même prénom.
- Au niveau des relations entre les tables : contraintes d'intégrité référentielles.

Par contre la redondance des données (formes normales) **n'est absolument pas vérifiée automatiquement** par les SGBD, il faut faire des requêtes spécifiques de recherche

d'anomalies (dites post-mortem) **à postériori**, ce qui semble être une grosse lacune de ces systèmes puisque les erreurs sont déjà présentes dans la base !

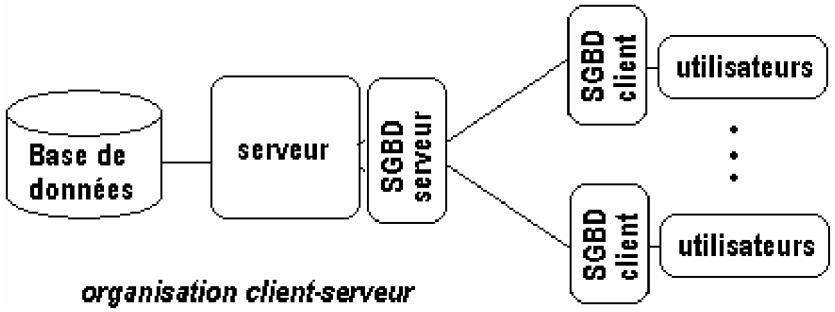
On organise actuellement les SGBD selon deux modes :

L'organisation locale selon laquelle le SGBD réside sur la machine où se trouve la base de données :



organisation locale

L'organisation client-serveur selon laquelle sur le SGBD est réparti entre la machine serveur locale supportant la BD (partie SGBD serveur) et les machines des utilisateurs (partie SGBD client). Ce sont ces deux parties du SGBD qui communiquent entre elles pour assurer les transactions de données :



organisation client-serveur

Le caractère généraliste de la structuration des données induit une description abstraite de l'objet BD (Base de données). Les applications étant indépendantes des données, ces dernières peuvent donc être manipulées et changées indépendamment du programme qui y accédera en implantant les méthodes générales d'accès aux données de la base, conformément à sa structuration abstraite.

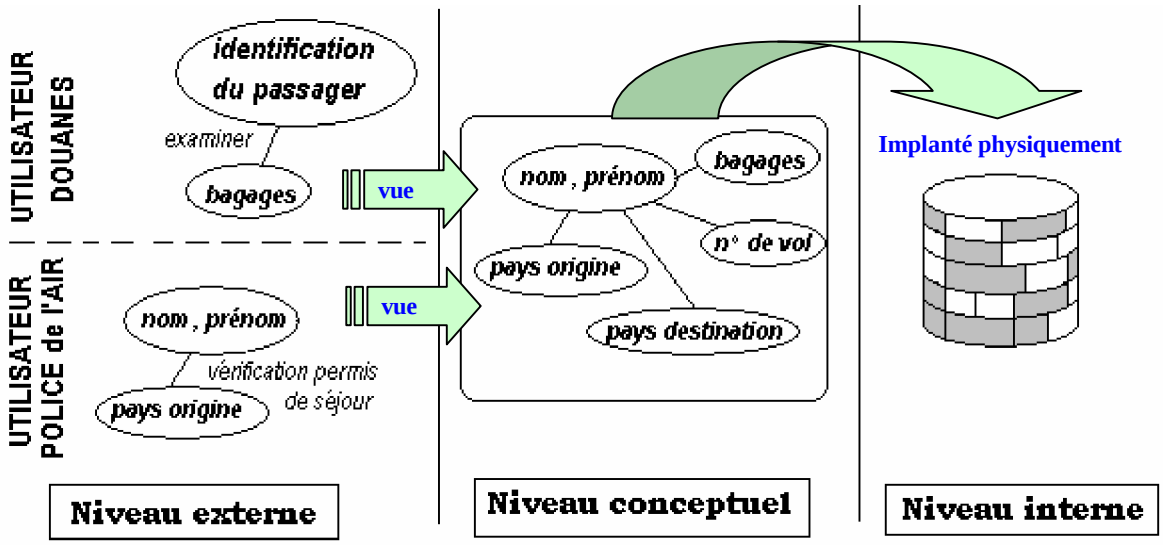
Une Base de Données peut être décrite de plusieurs points de vue, selon que l'on se place du côté de l'utilisateur ou bien du côté du stockage dans le disque dur du serveur ou encore du concepteur de la base.

Il est admis de nos jours qu'une **BD** est décrite en trois niveaux d'abstraction : un seul niveau a une existence matérielle physique et les deux autres niveaux sont une explication abstraite de ce niveau matériel.

Les 3 niveaux d'abstraction définis par l'ANSI depuis 1975

- **Niveau externe** : correspond à ce que l'on appelle une vue de la BD ou la façon dont sont perçues au niveau de l'utilisateur les données manipulées par une certaine application (vue abstraite sous forme de schémas)
- **Niveau conceptuel** : correspond à la description abstraite des composants et des processus entrant dans la mise en œuvre de la BD. Le niveau conceptuel est le plus important car il est le résultat de la traduction de la description du monde réel à l'aide d'expressions et de schémas conformes à un modèle de définition des données.
- **Niveau interne** : correspond à la description informatique du stockage physique des données (fichiers séquentiels, indexages, tables de hachage,…) sur le disque dur.

Figurons pour l'exemple des passagers d'un avion , stockés dans une base de données de la compagnie aérienne, sachant qu'en plus du personnel de la compagnie qui a une vue externe commerciale sur les passagers, le service des douanes peut accéder à un passager et à ses bagages et la police de l'air peut accéder à un passager et à son pays d''embarquement.

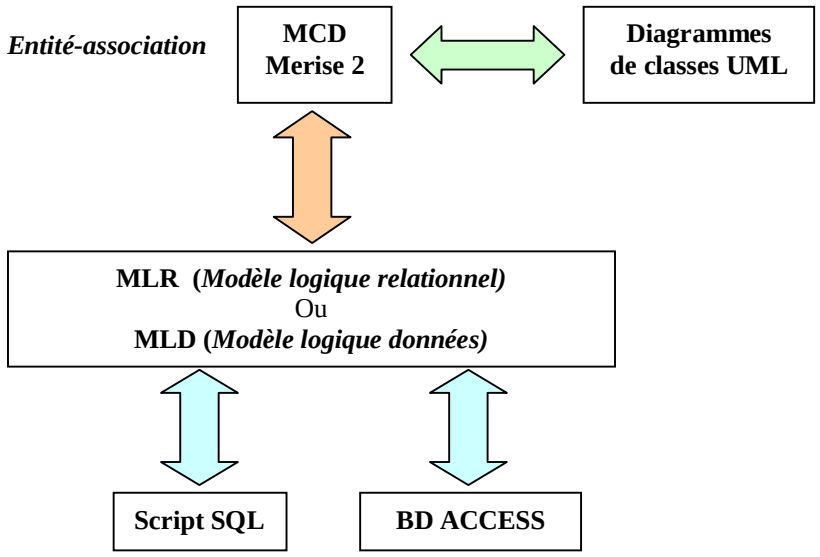


Le niveau conceptuel forme l'élément essentiel d'une BD et donc d'un SGBD chargé de gérer une BD, il est décrit avec un modèle de conception de données MCD avec la méthode française Merise qui est très largement répandu, ou bien par le formalisme des diagrammes de classes UML qui prend une part de plus en plus grande dans le formalisme de description conceptuelle des données (rappelons qu'UML est un langage de modélisation formelle, orienté objet et graphique ; Merise2 a intégré dans Merise ces concepts mais ne semble pas beaucoup être utilisé). Nous renvoyons le lecteur intéressé par cette partie aux très nombreux ouvrages écrits sur Merise ou sur UML.

Dans la pratique actuelle les logiciels de conception de BD intègrent à la fois la méthode Merise 2 et les diagrammes de classes UML. Ceci leur permet surtout la génération automatique et semi-automatique (paramétrable) de la BD à partir du modèle conceptuel sous forme de scripts (programmes simples) SQL adaptés aux différents SGBD du marché (ORACLE, SYBASE, MS-SQLSERVER,…) et les différentes versions de la BD ACCESS.

Les logiciels de conception actuels permettent aussi la rétro-génération (ou reverse engeneering) du modèle à partir d'une BD existante, cette fonctionnalité est très utile pour reprendre un travail mal documenté.

En résumé pratique :



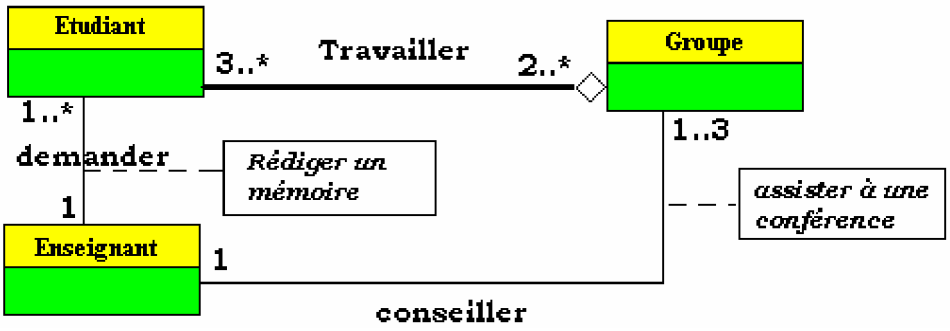
C'est en particulier le cas du logiciel français WIN-DESIGN dont une version démo est disponible à www.win-design.com et de son rival POWER-AMC (ex AMC-DESIGNOR).

Signalons enfin un petit logiciel plus modeste, très intéressant pour débuter avec version limitée seulement par la taille de l'exemple : CASE-STUDIO chez CHARONWARE. Les logiciels basés uniquement sur UML sont, à ce jour, essentiellement destinés à la génération de code source (Java, Delphi, VB, C++,…), les versions **Community** (versions logicielles libres) de ces logiciels ne permettent pas la génération de BD ni celle de scripts SQL.

Les quelques schémas qui illustreront ce chapitre seront décrits avec le langage UML.

L'exemple ci-après schématise en UML le mini-monde universitaire réel suivant :

- un enseignant pilote entre 1 et 3 groupes d'étudiants,
- un enseignant demande à 1 ou plusieurs étudiants de rédiger un mémoire,
- un enseignant peut conseiller aux groupes qu'il pilote d'aller assister à une conférence,
- un groupe est constitué d'au moins 3 étudiants,
- un étudiant doit s'inscrire à au moins 2 groupes.



Si le niveau conceptuel d'une BD est assis sur un modèle de conceptualisation de haut niveau (Merise, UML) des données, il est ensuite fondamentalement traduit dans le Modèle Logique de représentation des Données (MLD). Ce dernier s'implémentera selon un modèle physique des données.

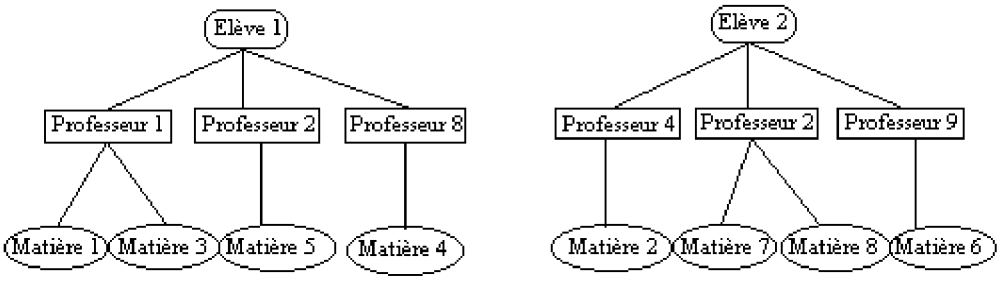
Il existe plusieurs MLD **M**odèles **L**ogiques de **D**onnées et plusieurs modèles physiques, et pour un même MLD, on peut choisir entre plusieurs modèles physiques différents.

Il existe 5 grands modèles logiques pour décrire les bases de données.

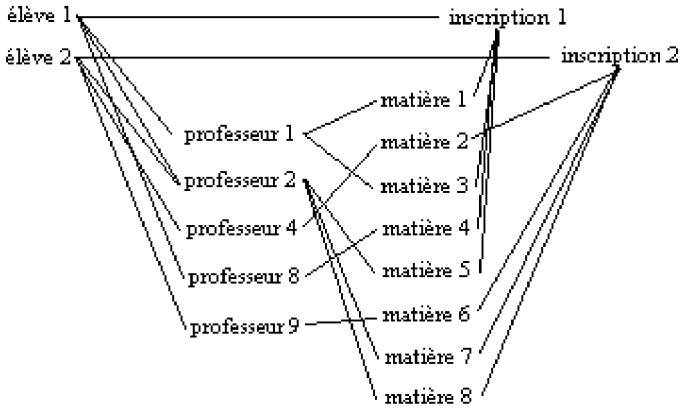
Les modèles de données historiques

(Prenons un exemple comparatif où des élèves ont des cours donnés par des professeurs leur enseignant certaines matières (les enseignants étant pluridisciplinaires)

- **Le modèle hiérarchique:** l'information est organisée de manière arborescente, accessible uniquement à partir de la racine de l'arbre hiérarchique. Le problème est que les points d'accès à l'information sont trop restreints.



- **Le modèle réseau:** toutes les informations peuvent être associées les unes aux autres et servir de point d'accès. Le problème est la trop grande complexité d'une telle organisation.



- **Le modèle relationnel**: toutes les relations entre les objets contenant les informations sont décrites et représentées sous la forme de tableaux à 2 dimensions.

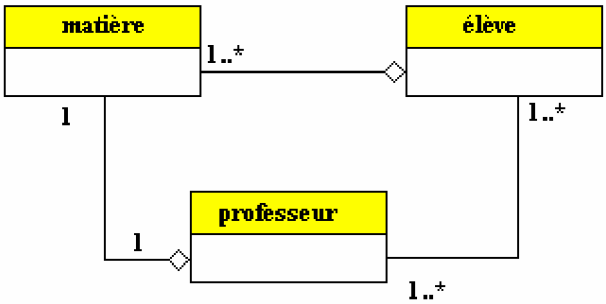
élève 1	matière 1
élève 1	matière 3
élève 1	matière 4
élève 1	matière 5
élève 2	matière 2
élève 2	matière 6
élève 2	matière 7
élève 2	matière 8

professeur 1	matière 1
professeur 1	matière 3
professeur 2	matière 5
professeur 2	matière 7
professeur 2	matière 8
professeur 4	matière 2
professeur 8	matière 4
professeur 9	matière 6

Dans ce modèle, la gestion des données (insertion, extraction,…) fonctionne selon la théorie mathématique de l'algèbre relationnelle. C'est le modèle qui allie une grande indépendance vis à vis des données à une simplicité de description.

- **Le modèle par déduction :** comme dans le modèle relationnel les données sont décrites et représentées sous la forme de tableaux à 2 dimensions. La gestion des données (insertion, extraction,…) fonctionne selon la théorie mathématique du calcul dans la logique des prédicats. Il ne semble exister de SGBD commercial directement basé sur ce concept. Mais il est possible de considérer un programme Prolog (programmation en logique) comme une base de données car il intègre une description des données. Ce sont plutôt les logiciels de réseaux sémantiques qui sont concernés par cette approche (cf. logiciel AXON).

- **Le modèle objet :** les données sont décrites comme des classes et représentées sous forme d'objets, un modèle **relationnel-objet** devrait à terme devenir le modèle de base.



L'expérience montre que le modèle relationnel s'est imposé parce qu'il était le plus simple en terme d'indépendance des données par rapport aux applications et de facilité de représenter les données dans notre esprit. C'est celui que nous décrirons succinctement dans la suite de ce chapitre.

2. Le modèle de données relationnelles

Défini par EF Codd de la société IBM dès 1970, ce modèle a été amélioré et rendu opérationnel dans les années 80 sous la forme de SBGD-R (SGBD Relationnels). Ci-dessous une liste non exhaustive de tels SGBD-R :

Access de Microsoft,
Oracle,
DB2 d'IBM,
Interbase de Borland,
SQL server de microsoft,
Informix,
Sybase,
MySQL,
PostgreSQL, ….

Nous avons déjà vu dans un précédent chapitre, la notion de relation binaire : une relation binaire R est un sous-ensemble d'un produit cartésien de deux ensembles finis E et F que nous nommerons domaines de la relation R :

$$R \subseteq E \times F$$

Cette définition est généralisable à n domaines, nous dirons que **R** est une relation n-aire sur les domaines E_1 , E_2 , … , E_n si et seulement si :

$$R \subseteq E_1 \times E_2 \ldots \times E_n$$

Les ensembles E_k peuvent être définis comme en mathématiques : en extension ou en compréhension :

E_k = { 12 , 58 , 36 , 47 } *en extension*

E_k = { x / (x est entier) **et(** $x \in [1, 20]$) } *en compréhension*

Notation

si nous avons: $R = \{ (v_1 , v_2 \ldots , v_n) \}$,
Au lieu d'écrire : $(v_1 , v_2 \ldots , v_n) \in R$, on écrira $R(v_1 , v_2 , \ldots , v_n)$

Exemple de déclarations de relations :

Passager (nom, prénom, n° de vol, nombre de bagages) , cette relation contient les informations utiles sur un passager d'une ligne aérienne.

Personne (nom, prénom) , cette relation caractérise une personne avec deux attributs

Enseignement (professeur, matière) , cette relation caractérise un enseignement avec le nom de la matière et le professeur qui l'enseigne.

Schéma d'une relation

On appelle schéma de la relation R : $R(a_1 : E_1, a_2 : E_2 ,\ldots, a_n : E_n)$
Où $(a_1 , a_2 \ldots , a_n)$ sont appelés les **attributs**, chaque attribut a_k indique comment est utilisé dans la relation R le domaine E_k , chaque attribut prend sa valeur dans le domaine qu'il définit, nous notons $val(a_k)= v_k$ où v_k est un élément (une valeur) quelconque de l'ensemble E_k (domaine de l'attribut a_k).

Convention : lorsqu'il n'y a pas de valeur associée à un attribut dans un n-uplet, on convient de lui mettre une valeur spéciale notée **null**, indiquant l'absence de valeur de l'attribut dans ce n-uplet.

Degré d'une relation

On appelle degré d'une relation, le nombre d'attributs de la relation.

Exemple de schémas de relations :

Passager (nom : chaîne, prénom : chaîne, n° de vol : entier, nombre de bagages : entier) relation de degré 4.

Personne (nom : chaîne, prénom : chaîne) relation de degré 2.

Enseignement (professeur : ListeProf, matière : ListeMat) relation de degré 2.

*Attributs : prenons le schéma de la relation **Enseignement***

Enseignement (professeur : ListeProf, matière : ListeMat). C'est une relation binaire (degré 2) sur les deux domaines ListeProf et ListeMat. L'attribut professeur joue le rôle d'un paramètre formel et le domaine ListeProf celui du type du paramètre.

Supposons que :

ListeProf ={ Poincaré, Einstein, Lavoisier, Raimbault , Planck }
ListeMat = { mathématiques, poésie , chimie , physique }

L'attribut professeur peut prendre toutes valeurs de l'ensemble ListeProf :

Val(professeur) = Poincaré, …. , Val(professeur) = Raimbault

Si l'on veut dire que le poste d'enseignant de chimie n'est pas pourvu on écrira :

Le couple (**null** , chimie) est un couple de la relation **Enseignement.**

Enregistrement dans une relation

Un n-uplet $(val(a_1), val(a_2) \ldots , val(a_n)) \in R$ est appelé un enregistrement de la relation R. Un enregistrement est donc constitué de valeurs d'attributs.

Dans l'exemple précédent (Poincaré , mathématiques), (Raimbault , poésie) , (**null** , chimie) sont trois enregistrements de la relation **Enseignement**.

Clef d'une relation

Si l'on peut caractériser d'une façon **bijective** tout n-uplet d'attributs $(a_1 , a_2 \ldots , a_n)$ avec seulement un sous-ensemble restreint $(a_{k1} , a_{k2} \ldots , a_{kp})$ avec $p < n$, de ces attributs, alors ce sous-ensemble est appelé une **clef** de la relation. Une relation peut avoir plusieurs clefs, nous choisissons l'une d'elle en la désignant comme **clef primaire de la relation**.

Clef minimale d'une relation

On a intérêt à ce que la clef **primaire soit minimale** en nombre d'attributs, car il est clair que si un sous-ensemble à p attributs $(a_{k1} , a_{k2} \ldots , a_{kp})$ est une clef, tout sous-ensemble à p+m attributs dont les p premiers sont les $(a_{k1} , a_{k2} \ldots , a_{kp})$ est aussi une clef :
$(a_{k1} , a_{k2} \ldots , a_{kp} , a_0 , a_1)$
$(a_{k1} , a_{k2} \ldots , a_{kp} , a_{10} , a_5 , a_9 , a_2)$ sont aussi des clefs etc…

Il n'existe aucun moyen méthodique formel général pour trouver une clef primaire d'une relation, il faut observer attentivement. Par exemple :

- Le code Insee est une clef primaire permettant d'identifier les personnes.
- Si le couple (nom, prénom) peut suffire pour identifier un élève dans une classe d'élèves, et pourrait être choisi comme clef primaire, il est insuffisant au niveau d'un lyçée où il est possible que l'on trouve plusieurs élèves portant le même nom et le même premier prénom ex: (martin, jean).

Convention : on souligne dans l'écriture d'une relation dont on a déterminé une clef primaire, les attributs faisant partie de la clef.

Clef secondaire d'une relation

Tout autre clef de la relation qu'une clef primaire (minimale) , exemple :
Si $(a_{k1} , a_{k2} \ldots , a_{kp})$ est un clef primaire de R
$(a_{k1} , a_{k2} \ldots , a_{kp} , a_0 , a_1)$ et $(a_{k1} , a_{k2} \ldots , a_{kp} , a_{10} , a_5 , a_9 , a_2)$ sont des clefs secondaires.

Clef étrangère d'une relation

Soit $(a_{k1} , a_{k2} \ldots , a_{kp})$ un p-uplet d'attributs d'une relation R de degré n. [$R(a_1 : E_1, a_2 : E_2 ,\ldots, a_n : E_n)$]

Si $(a_{k1} , a_{k2} \ldots , a_{kp})$ est une clef primaire d'une autre relation Q on dira que $(a_{k1} , a_{k2} \ldots , a_{kp})$ est une clef étrangère de R.

Convention : on met un # après chaque attribut d'une clef étrangère.

Exemple de clef secondaire et clef étrangère :

Passager (nom# : chaîne, prénom# : chaîne , n° de vol : entier, nombre de bagages : entier, n° client : entier) relation de degré 5.

Personne (nom : chaîne, prénom : chaîne , âge : entier, civilité : Etatcivil) relation de degré 4.

n° client est une clef primaire de la relation **Passager**.
(nom, n° client) est une clef secondaire de la relation **Passager**.
(nom, n° client , n° de vol) est une clef secondaire de la relation **Passager**….etc

(nom , prénom) est une clef primaire de la relation **Personne**, comme (nom# , prénom#) est aussi un couple d'attributs de la relation **Passager**, c'est une clef étrangère de la relation **Passager**.

On dit aussi que dans la relation **Passager**, le couple d'attributs (nom# , prénom#) réfère à la relation **Personne**.

Règle d'intégrité référentielle

Toutes les valeurs d'une clef étrangère (v_{k1} , v_{k2} … , v_{kp}) se retrouvent comme valeur de la clef primaire de la relation référée (ensemble des valeurs de la clef étrangère est **inclus** au sens large dans l'ensemble des valeurs de la clef primaire)

Reprenons l'exemple précédent

(nom , prénom) est une clef étrangère de la relation **Passager**, c'est donc une clef primaire de la relation **Personne** : les domaines (liste des noms et liste des prénoms associés au nom doivent être strictement les mêmes dans **Passager** et dans **Personne**, nous dirons que la clef étrangère respecte la contrainte d'intégrité référentielle.

Règle d'intégrité d'entité

Aucun des attributs participant à une clef primaire ne peut avoir la valeur **null**.

Nous définirons la valeur **null**, comme étant une valeur spéciale n'appartenant pas à un domaine spécifique mais ajoutée par convention à tous les domaines pour indiquer qu'un champ n'est pas renseigné.

Représentation sous forme tabulaire

Reprenons les relations Passager et Personne et figurons un exemple pratique de valeurs des relations.

Passager (nom# : chaîne, prénom# : chaîne, n° de vol : entier, nombre de bagages : entier, n° client : entier).

Personne (nom : chaîne, prénom : chaîne, âge : entier, civilité : Etatcivil) relation de degré 4.

Nous figurons les tables de valeurs des deux relations

Passager :

Clef étrangère

nom#	prénom#	N°vol	Nbr bagages	N° client
Einstein	Albert	45622	2	154565
Lavoisier	Antoine	45622	1	785154
Raimbault	Arthur	12896	2	544552
Poincaré	Henri	45644	3	781201
Einstein	Albert	75906	**null**	858547
Lavoisier	Antoine	45644	4	235002

Personne:

Clef primaire

Clef primaire

nom	prénom	âge	Civilité
Einstein	Albert	45	marié
Lavoisier	Antoine	41	marié
Planck	Max	52	veuf
Raimbault	Arthur	25	célibataire
Poincaré	Henri	45	marié

Nous remarquons que la compagnie aérienne attribue un numéro de client unique à chaque personne, c'est donc un bon choix pour une clef primaire.

Les deux tables (relations) ont deux colonnes qui portent les mêmes noms colonne **nom** et colonne **prénom**, ces deux colonnes forment une clef primaire de la table **Personne**, c'est donc une clef étrangère de **Passager** qui réfère **Personne.**

En outre, cette clef étrangère respecte la contrainte d'intégrité référentielle : la liste des valeurs de la clef étrangère dans **Passager** est incluse dans la liste des valeurs de la clef primaire associée dans **Personne**.

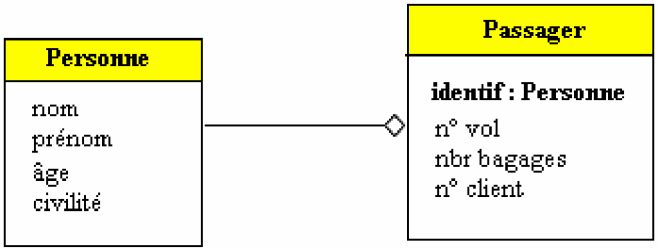


Diagramme UML modélisant la liaison Passager-Personne

Ne pas penser qu'il en est toujours ainsi, par exemple voici une autre relation **Passager2** dont la clef étrangère ne respecte pas la contrainte d'intégrité référentielle :

Clef étrangère , réfère Personne

Passager2 :

nom	prénom	N°vol	Nbr bagages	N° client
Einstein	Albert	45622	2	154565
Lavoisier	Antoine	45644	1	785154
Raimbault	Arthur	12896	2	544552
Poincaré	Henri	45644	3	781201
Einstein	Albert	75906	**null**	858547
Picasso	Pablo	12896	5	458023

En effet, le couple (Picasso , Pablo) n'est pas une valeur de la clef primaire dans la table **Personne**.

Principales règles de normalisation d'une relation

1ère forme normale :

Une relation est dite en première forme normale si, chaque attribut est représenté par un identifiant unique (les valeurs ne sont pas des ensembles, des listes,…) .Ci-dessous une relation qui n'est pas en 1ère forme normale car l'attribut **n° vol** est multivalué (il peut prendre 2 valeurs) :

nom	prénom	N°vol	Nbr bagage	N° client
Einstein	Albert	45622 , 75906	2	154565
Lavoisier	Antoine	45644 , 45622	1	785154
Raimbault	Arthur	12896	2	544552
Poincaré	Henri	45644	3	781201
Picasso	Pablo	12896	5	458023

En pratique, il est très difficile de faire vérifier automatiquement cette règle, dans l'exemple proposé on pourrait imposer de passer par un masque de saisie afin que le N°vol ne comporte que 5 chiffres.

2ème forme normale :

Une relation est dite en deuxième forme normale si, elle est **en première forme normale** et si chaque attribut qui n'est pas une clef, dépend entièrement de tous les attributs qui composent la clef primaire. La relation **Personne** (nom : chaîne, prénom : chaîne , age : entier , civilité : Etatcivil) est en deuxième forme normale :

nom	**prénom**	**âge**	**Civilité**
Einstein	Albert	45	marié
Lavoisier	Antoine	41	marié
Planck	Max	52	veuf
Raimbault	Arthur	25	célibataire
Poincaré	Henri	45	marié

Car l'attribut âge ne dépend que du nom et du prénom, de même pour l'attribut civilité.

La relation **Personne3** (nom : chaîne, prénom : chaîne , age : entier , civilité : Etatcivil) qui a pour clef primaire (nom , âge) n'est pas en deuxième forme normale :

nom	**prénom**	**âge**	**Civilité**
Einstein	Albert	45	marié
Lavoisier	Antoine	41	marié
Planck	Max	52	veuf
Raimbault	Arthur	25	célibataire
Poincaré	Henri	45	marié

Car l'attribut Civilité ne dépend que du nom et non pas de l'âge ! Il en est de même pour le prénom, soit il faut changer de clef primaire et prendre (nom, prénom) soit si l'on conserve la clef primaire (nom , âge) , il faut décomposer la relation **Personne3** en deux autres relations **Personne31** et **Personne32** :

nom	**âge**	**Civilité**
Einstein	45	marié
Lavoisier	41	marié
Planck	52	veuf
Raimbault	25	célibataire
Poincaré	45	marié

Personne31(nom : chaîne, age : entier , civilité : Etatcivil) : **2ème forme normale**

nom	**âge**	**prénom**
Einstein	45	Albert
Lavoisier	41	Antoine
Planck	52	Max
Raimbault	25	Arthur
Poincaré	45	Henri

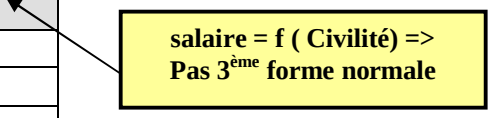
Personne32(nom : chaîne, age : entier , prénom : chaîne) : **2ème forme normale**

En pratique, il est aussi très difficile de faire vérifier automatiquement la mise en deuxième forme normale. Il faut trouver un jeu de données représentatif.

3ème forme normale :

Une relation est dite en troisième forme normale si chaque attribut qui ne compose pas la clef primaire, dépend directement de son identifiant et à travers une dépendance fonctionnelle. Les relations précédentes sont toutes en forme normale. Montrons un exemple de relation qui n'est pas en forme normale. Soit la relation **Personne4** (nom : chaîne, age : entier, civilité : Etatcivil, salaire : monétaire) où par exemple le salaire dépend de la clef primaire et que l'attribut civilité ne fait pas partie de la clef primaire (nom , âge) **:**

nom	âge	Civilité	salaire
Einstein	45	marié	1000
Lavoisier	41	marié	1000
Planck	52	veuf	1800
Raimbault	25	célibataire	1200
Poincaré	45	marié	1000



L'attribut salaire dépend de l'attribut civilité, ce que nous écrivons salaire = f(civilité), mais l'attribut civilité ne fait pas partie de la clef primaire clef = (nom , âge), donc **Personne4** n'est pas en 3ème forme normale **:**

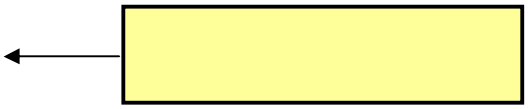
Il faut alors décomposer la relation **Personne4** en deux relations **Personne41** et **Personne42** chacune en troisième forme normale:

Personne41 :

nom	âge	Civilité#
Einstein	45	marié
Lavoisier	41	marié
Planck	52	veuf
Raimbault	25	célibataire
Poincaré	45	marié

Personne42 :

Civilité	salaire
marié	1000
veuf	1800
célibataire	1200



En pratique, il est également très difficile de faire contrôler automatiquement la mise en troisième forme normale.

Remarques pratiques importantes pour le débutant :

- Les spécialistes connaissent deux autres formes normales. Dans ce cas le lecteur intéressé par l'approfondissement du sujet, trouvera dans la littérature, de solides références sur la question.

- Si la clef primaire d'une relation n'est composée que d'un seul attribut (choix conseillé lorsque cela est possible, d'ailleurs on trouve souvent des clefs primaires sous forme de numéro d'identification client, Insee,…) automatiquement, la relation est en 2ème forme normale, car chaque autre attribut non clef étrangère, ne dépend alors que de la valeur unique de la clef primaire.

- Penser dès qu'un attribut est fonctionnellement dépendant d'un autre attribut qui n'est pas la clef elle-même à décomposer la relation (créer deux nouvelles tables).

- En l'absence d'outil spécialisé, il faut de la pratique et être très systématique pour contrôler la normalisation.

Base de données relationnelles BD-R:

Ce sont des données structurées à travers :

- Une famille de domaines de valeurs,
- Une famille de relations n-aires,
- Les contraintes d'intégrité sont respectées par toute clef étrangère et par toute clef primaire.
- Les relations sont en 3ème forme normale. (à minima en 2ème forme normale)

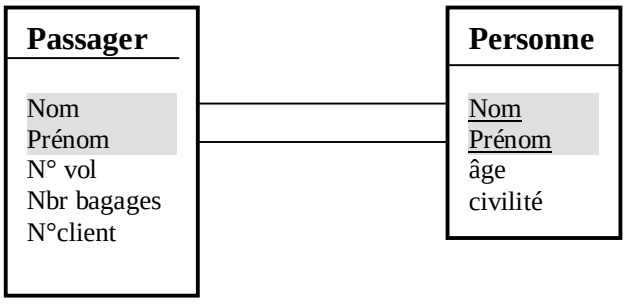
Les données sont accédées et manipulées grâce à un langage appelé **langage d'interrogation** ou **langage relationnel** ou **langage de requêtes**

Système de Gestion de Base de Données relationnel :

C'est une famille de logiciels comprenant :

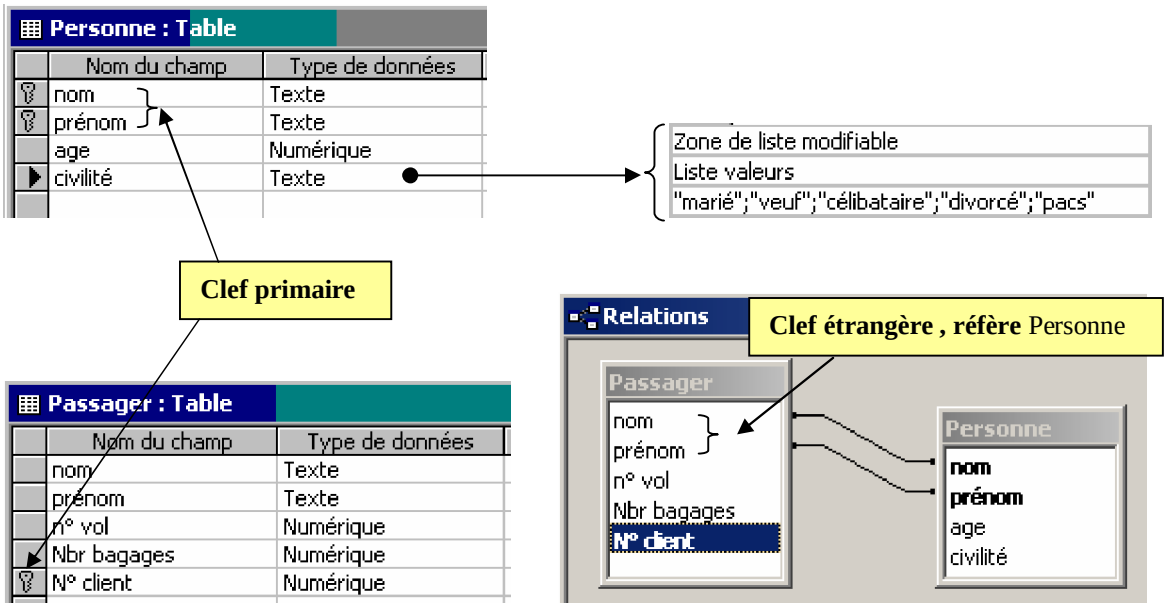
- Une BD-R.
- Un langage d'interrogation.
- Une gestion en interne des fichiers contenant les données et de l'ordonnancement de ces données.
- Une gestion de l'interface de communication avec les utilisateurs.
- La gestion de la sécurité des accès aux informations contenues dans la BD-R.

Le schéma relation d'une relation dans une BD relationnelle est noté graphiquement comme ci-dessous :

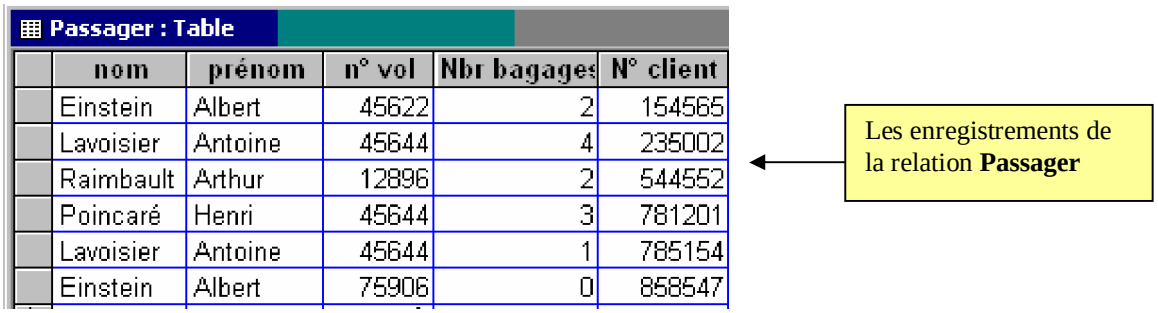


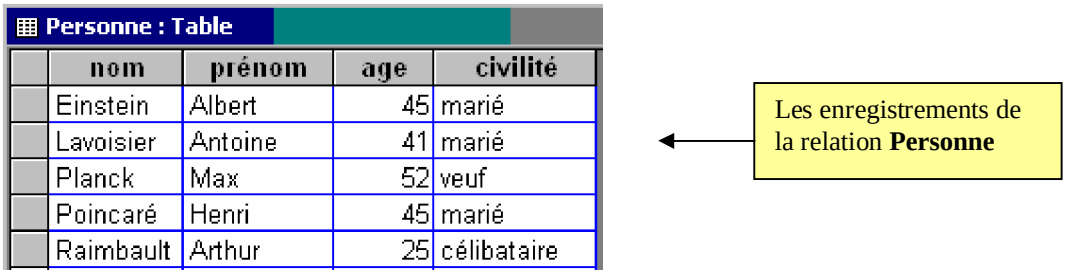
Les attributs reliés entre ces deux tables indiquent une liaison entre les deux relations.

Voici dans le **SGBD-R Access**, la représentation des schémas de relation ainsi que la liaison sans intégrité des deux relations précédentes **Passager** et **Personne** :



Access et la représentation des enregistrements de chaque table :





Les besoins d'un utilisateur d'une base de données sont classiquement ceux que l'on trouve dans tout ensemble de données structurées : insertion, suppression, modification, recherche avec ou sans critère de sélection. Dans une BD-R, ces besoins sont exprimés à travers un langage d'interrogation. Historiquement deux classes de langages relationnels équivalentes en puissance d'utilisation et de fonctionnement ont été inventées : les langages **algébriques** et les langages des **prédicats**.

Un langage relationnel n'est pas un langage de programmation : il ne possède pas les structures de contrôle de base d'un langage de programmation (condition, itération, …). **Très souvent il doit être utilisé comme complément à l'intérieur de programmes Delphi, Java, C#, ...**

Les langages d'interrogation prédicatifs sont des langages fondés sur la logique des prédicats du 1er ordre, le plus ancien s'appelle **Q**uery **B**y **E**xample QBE.

Ce sont les langages algébriques qui sont de loin les plus utilisés dans les SGBD-R du commerce, le plus connu et le plus utilisé dans le monde se dénomme le **S**tructured **Q**uery **L**anguage ou SQL. Un tel langage n'est qu'une implémentation en anglais d'opérations définies dans une algèbre relationnelle servant de modèle mathématique à tous les langages relationnels.

3. Principes fondamentaux d'une l'algèbre relationnelle

Une algèbre relationnelle est une famille d'opérateurs binaires ou unaires dont les opérandes sont des **relations**. Nous avons vu que l'on pouvait faire l'union, l'intersection, le produit cartésien de relations binaires dans un chapitre précédent, comme les relations n-aires sont des ensembles, il est possible de définir sur elle une algèbre opératoire utilisant les opérateurs classiques ensemblistes, à laquelle on ajoute quelques opérateurs spécifiques à la manipulation des données.

Remarque pratique :

> La phrase "**tous les n-uples sont distincts, puisqu'éléments d'un même ensemble nommé relation**" se transpose en pratique en la phrase **" toutes les lignes d'une même table nommée relation, sont distinctes** (même en l'absence de clef primaire explicite)".

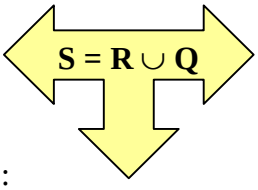
Nous exhibons les opérateurs principaux d'une algèbre relationnelle et nous montrerons pour chaque opération, un exemple sous forme.

Union de 2 relations

> Soient R et Q deux relations de même domaine et de même degré on peut calculer la nouvelle relation S = R ∪ Q de même degré et de même domaine contenant les enregistrements différents des deux relations R et Q :

R :

nom	âge	Civilité
Einstein	45	marié
Lavoisier	41	marié
Planck	52	veuf



Q :

nom	âge	Civilité
Lupin	45	célibataire
Planck	52	veuf
Mozart	34	veuf
Gandhi	64	célibataire

S :

nom	âge	Civilité
Einstein	45	marié
Lavoisier	41	marié
Planck	52	veuf

Lupin	45	célibataire
Mozart	34	veuf
Gandhi	64	célibataire

Remarque : (Planck, 52, veuf) ne figure qu'une seule fois dans la table R ∪ Q.

Intersection de 2 relations

Soient R et Q deux relations de même domaine et de même degré on peut calculer la nouvelle relation S = R ∩ Q de même degré et de même domaine contenant les enregistrements communs aux deux relations R et Q :

R :

nom	âge	Civilité
Einstein	45	marié
Lavoisier	41	marié
Planck	52	veuf

S = R ∩ Q

Q :

nom	âge	Civilité
Einstein	45	marié
Planck	52	veuf
Mozart	34	veuf
Gandhi	64	célibataire

S :

nom	âge	Civilité
Einstein	45	marié
Planck	52	veuf

Différence de 2 relations

Soient R et Q deux relations de même domaine et de même degré on peut calculer la nouvelle relation S = R – Q de même degré et de même domaine contenant les enregistrements qui sont présents dans R mais qui ne sont pas dans Q (on exclut de R les enregistrements qui appartiennent à R ∩ Q) :

R :

nom	âge	Civilité
Einstein	45	marié
Lavoisier	41	marié
Planck	52	veuf

S = R – Q

Q :

nom	âge	Civilité
Einstein	45	marié
Planck	52	veuf
Mozart	34	veuf
Gandhi	64	célibataire

S :

nom	âge	Civilité
Lavoisier	41	marié

Produit cartésien de 2 relations

Soient R et Q deux relations de domaine et de degré quelconques (degré(R)=n, degré(Q)=p), avec Domaine(R) ∩ Domaine(Q) = ∅ (pas d'attributs en communs).

On peut calculer la nouvelle relation S = R x Q de degré n + p et de domaine égal à l'union des domaines de R et de Q contenant tous les couples d'enregistrements à partir d'enregistrements présents dans R et d'enregistrements présents dans Q :

R :

nom	âge	Civilité
Einstein	45	marié
Lavoisier	41	marié
Planck	52	veuf

S = R x Q

Q :

ville	km
Paris	874
Rome	920

S :

nom	âge	Civilité	ville	km
Einstein	45	marié	Paris	874
Einstein	45	marié	Rome	920
Lavoisier	41	marié	Paris	874
Lavoisier	41	marié	Rome	920
Planck	52	Veuf	Paris	874
Planck	52	Veuf	Rome	920

Selection ou Restriction d'une relation

Soit R une relation, soit $R(a_1 : E_1, a_2 : E_2 ,\ldots, a_n : E_n)$ le schéma de cette relation. Soit Cond(a_1, a_2 , … , a_n) une expression booléenne classique (expression construite sur les attributs avec les connecteurs de l'algèbre de Boole et les opérateurs de comparaison < , > , = , >= , <= , <>)

On note S = **select** (Cond(a_1, a_2 , … , a_n) , R), la nouvelle relation S construite ayant le même schéma que R soit $S(a_1 : E_1, a_2 : E_2 ,\ldots, a_n : E_n)$, qui ne contient que les enregistrements de R qui satisfont à la condition booléenne Cond(a_1, a_2 , … , a_n).

R :

nom	âge	Civilité	ville	km
Einstein	45	marié	Paris	874
Mozart	32	marié	Rome	587
Gandhi	64	célibataire	Paris	258
Lavoisier	41	marié	Rome	124
Lupin	42	Veuf	Paris	608
Planck	52	Veuf	Rome	405

Cond(a_1, a_2 , … , a_n) = { âge > 42 **et** ville=Paris }

S :

nom	âge	Civilité	ville	km
Einstein	45	marié	Paris	874
Gandhi	64	célibataire	Paris	258

Select ({ âge > 42 et ville=Paris }, R) signifie que l'on ne recopie dans S que les enregistrements de R constitués des personnes ayant séjourné à Paris et plus âgées que 42 ans.

Projection d'une relation

Soit R une relation, soit $R(a_1 : E_1, a_2 : E_2 ,\ldots, a_n : E_n)$ le schéma de cette relation. On appelle S = **proj**(a_{k1} , a_{k2} … , a_{kp}) la projection de R sur un sous-ensemble restreint (a_{k1} , a_{k2} … , a_{kp}) avec p < n , de ses attributs, la relation S ayant pour

schéma le sous-ensemble des attributs $S(a_{k1} : E_{k1}, a_{k2} : E_{k2}, \ldots, a_{kp} : E_{kp})$ et contenant les enregistrements différents obtenus en ne considérant que les attributs $(a_{k1}, a_{k2} \ldots, a_{kp})$.

Exemple

R :

nom	âge	Civilité	ville	km
Einstein	45	marié	Paris	874
Mozart	32	marié	Rome	587
Lupin	42	Veuf	Paris	464
Einstein	45	marié	Venise	981
Gandhi	64	célibataire	Paris	258
Lavoisier	41	marié	Rome	124
Lupin	42	Veuf	Paris	608
Planck	52	Veuf	Rome	405

S1 = **proj**(nom , civilité)

nom	Civilité
Einstein	marié
Mozart	marié
Lupin	Veuf
Gandhi	célibataire
Lavoisier	marié
Planck	Veuf

S2 = **proj**(nom , âge)

nom	âge
Einstein	45
Mozart	32
Lupin	42
Gandhi	64
Lavoisier	41
Planck	52

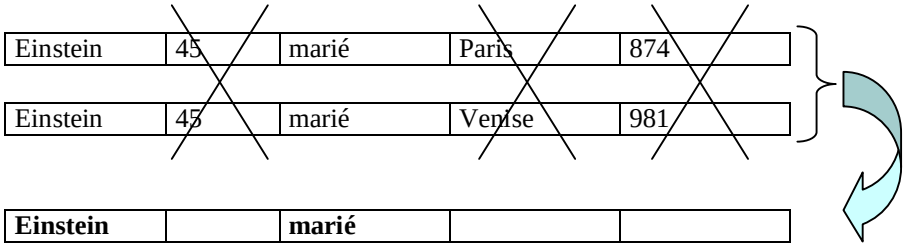
S3 = **proj**(nom , ville)

nom	ville
Einstein	Paris
Mozart	Rome
Lupin	Paris
Einstein	Venise
Gandhi	Paris
Lavoisier	Rome
Planck	Rome

S4 = **proj**(ville)

ville
Paris
Rome
Venise

Que s'est-il passé pour Mr Einstein dans S2 ?



Lors de la recopie des enregistrements de R dans S2 on a ignoré les attributs âge, ville et km, le couple (Einstein , marié) ne doit se retrouver qu'une seule fois car une relation est un ensemble et ses éléments sont tous distincts.

Jointure de deux relations

Soit Soient R et Q deux relations de domaine et de degré quelconques (degré(R) = n, degré(Q) = p), avec Domaine(R) ∩ Domaine(Q) = ∅ (pas d'attributs en communs).

soit R x Q leur produit cartésien de degré n + p et de domaine D union des domaines de R et de Q.
Soit un ensemble $(a_1, a_2, \ldots, a_{n+p})$ d'attributs du domaine D de R x Q.

La relation **joint**(R,Q) = **select** (Cond(a_1, a_2 , … , a_{n+p}) , R x Q), est appelée jointure de R et de Q (c'est donc une sélection de certains attributs sur le produit cartésien).

Une jointure couramment utilisée en pratique, est celle qui consiste en la sélection selon une condition d'égalité entre deux attributs, les personnes de "l'art relationnel" la dénomment alors l'**équi-jointure**.

Exemples de 2 jointures :

R :

nom	âge	Civilité
Einstein	45	marié
Lavoisier	41	marié
Planck	52	veuf

joint(R,Q)

Q :

ville	km
Paris	874
Rome	920

1°) S = **select** (km < 900 , R x Q)

nom	âge	Civilité	ville	km
Einstein	45	marié	Paris	874
Lavoisier	41	marié	Paris	874
Planck	52	veuf	Paris	874

2°) S = **select** (km < 900 **et** Civilité = veuf, R x Q)

nom	âge	Civilité	ville	km
Planck	52	veuf	Paris	874

Nous nous plaçons maintenant du point de vue pratique, non pas de l'administrateur de BD mais de l'utilisateur uniquement concerné par l'extraction des informations contenues dans une BD-R.

Un SGBD permet de gérer une base de données. A ce titre, il offre de nombreuses fonctionnalités supplémentaires à la gestion d'accès simultanés à la base et à un simple interfaçage entre le modèle logique et le modèle physique : il sécurise les données (en cas de coupure de courant ou autre défaillance matérielle), il permet d'accéder aux données de manière confidentielle (en assurant que seuls certains utilisateurs ayant des mots de passe appropriés, peuvent accéder à certaines données), il ne permet de mémoriser des données que si elles sont du type abstrait demandé : on dit qu'il vérifie leur intégrité (des données alphabétiques ne doivent pas être enregistrées dans des emplacements pour des données numériques,...)

Actuellement, une base de données n'a pas de raison d'être sans son SGBD. Aussi, on ne manipule que des bases de données correspondant aux SGBD qui les gèrent : il vous appartient de choisir le SGBD-R qui vous convient (il faut l'acheter auprès de vendeurs qui généralement, vous le fournissent avec une application de manipulation visuelle, ou bien utiliser les SGBD-R qui vous sont livrés gratuitement avec certains environnements de développement comme Borland studio ou Visual Studio ou encore utiliser les produits gratuits comme mySql).

Lorsque l'on parle d'utilisateur, nous entendons l'application utilisateur, car l'utilisateur final n'a pas besoin de connaître quoique ce soit à l'algèbre relationnelle, il suffit que l'application utilisateur communique avec lui et interagisse avec le SGBD.

Une application doit pouvoir "parler" au SGBD : elle le fait par le moyen d'un langage de manipulation des données. Nous avons déjà précisé que la majorité des SGBD-R utilisait un langage relationnel ou de requêtes nommé SQL pour manipuler les données.

4. SQL et Algèbre relationnelle

Requête

Les requêtes sont des questions posées au SGBD, concernant une recherche de données contenues dans une ou plusieurs tables de la base.

Par exemple, on peut disposer d'une table définissant des clients (noms, prénoms, adresses, n°de client) et d'une autre table associant des numéros de clients avec des numéros de commande d'articles, et vouloir poser la question : quels sont les noms des clients ayant passé des commandes ?

Une requête est en fait, une instruction de type langage de programmation, respectant la norme SQL, permettant de réaliser un tel questionnement. L'exécution d'une requête permet d'extraire des données en provenance de tables de la base de données : ces données réalisent ce que l'on appelle une **projection** de champs (en provenance de plusieurs tables). Le résultat d'exécution d'une requête est une table constituée par les réponses à la requête.

Le SQL permet à l'aide d'instructions spécifiques de manipuler des données à l'intérieur des tables :

Instruction SQL	Actions dans la (les) table(s)
INSERT INTO <…>	Ajout de lignes
DELETE FROM <…>	Suppression de lignes
TRUNCATE TABLE <…>	Suppression de lignes
UPDATE <…>	Modification de lignes
SELECT <…>**FROM** <…>	Extraction de données

Ajout, suppression et modification sont les trois opérations typiques de la **mise à jour** d'une BD. L'extraction concerne la **consultation** de la BD.

Il existe de nombreuses autres instructions de création, de modification, de suppression de tables, de création de clefs, de contraintes d'intégrités référentielles, création d'index, etc… Nous nous attacherons à donner la traduction en SQL des opérateurs principaux de l'algèbre relationnelle que nous venons de citer.

Traduction en SQL des opérateurs relationnels

C'est l'instruction SQL "**SELECT** <…>**FROM** <…>" qui implante tous ces opérateurs. Tous les exemples utiliseront la relation R = TableComplete suivante et l'interpréteur SQL d'**Access** :

TableComplete : Table

nom	âge	civilité	ville	km
Einstein	45	marié	Paris	874
Mozart	32	marié	Rome	587
Lupin	42	Veuf	Paris	464
Einstein	45	marié	Venise	981
Gandhi	64	célibataire	Paris	258
Lavoisier	41	marié	Rome	124
Lupin	42	Veuf	Paris	608
Planck	52	Veuf	Rome	405

La relation initiale : R=TableComplete

Projection d'une relation R

S = **proj**(a_{k1} , a_{k2} … , a_{kp})

SQL : **SELECT DISTINCT** a_{k1} , a_{k2} … , a_{kp} **FROM** R

Instruction SQL version opérateur algèbre :
SELECT DISTINCT nom , civilité **FROM** Tablecomplete

Lancement de la requête SQL :

Requête51 : Requête Sélection

```
SELECT DISTINCT [nom], [civilité]
FROM TableComplete;
```

Le mot DISTINCT assure que l'on obtient bien une nouvelle relation.

Table obtenue après requête :

Requête51 : Requête Sélection

nom	civilité
Einstein	marié
Gandhi	célibataire
Lavoisier	marié
Lupin	Veuf
Mozart	marié
Planck	Veuf

Instruction SQL non relationnelle (tout même redondant) :
SELECT nom , civilité **FROM** Tablecomplete

Lancement de la requête SQL :

Requête51 : Requête Sélection

```
SELECT [nom], [civilité]
FROM TableComplete;
```

Table obtenue après requête :

Requête51 : Requête Séle

nom	civilité
Einstein	marié
Mozart	marié
Lupin	Veuf
Einstein	marié
Gandhi	célibataire
Lavoisier	marié
Lupin	Veuf
Planck	Veuf

Remarquons dans le dernier cas **SELECT** nom , civilité **FROM** Tablecomplete que la table obtenue n'est qu'une extraction de données, mais qu'en aucun cas elle ne constitue une relation puisqu'une relation est un ensemble et que les enregistrements sont tous distincts!

Une autre projection sur la même table :

Instruction SQL version opérateur algèbre :
SELECT DISTINCT nom , ville **FROM** Tablecomplete

Lancement de la requête SQL :

RequêteS2 : Requête Sélection

```
SELECT DISTINCT [nom], [ville]
FROM TableComplete;
```

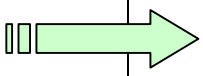


Table obtenue après requête :

RequêteS2 : Requête Sél

	nom	ville
	Einstein	Paris
	Einstein	Venise
	Gandhi	Paris
	Lavoisier	Rome
	Lupin	Paris
	Mozart	Rome
▶	Planck	Rome

Sélection-Restriction

S = **select** (Cond(a_1, a_2 , … , a_n) , R)
SQL : **SELECT * FROM** R **WHERE** Cond(a_1, a_2 , … , a_n)
*Le symbole * signifie toutes les colonnes de la table (tous les attributs du schéma)*

Instruction SQL version opérateur algèbre :
SELECT * FROM Tablecomplete **WHERE** âge > 42 **AND** ville = Paris

Lancement de la requête SQL :

TableComplete Requête : Requête Sélection

```
SELECT * FROM TableComplete
WHERE [âge]>42 AND [ville]="Paris";
```

Table obtenue après requête :

TableComplete Requête : Requête Sélection

	nom	âge	civilité	ville	km
	Einstein	45	marié	Paris	874
	Gandhi	64	célibataire	Paris	258

On a sélectionné toutes les personnes de plus de 42 ans ayant séjourné à Paris.

Combinaison d'opérateur projection-restriction

Projection distincte et sélection :
SELECT DISTINCT nom , civilité, âge **FROM** Tablecomplete **WHERE** âge >= 45

Lancement de la requête SQL :

TableComplete Requête : Requête Sélection

```
SELECT DISTINCT [nom], [civilité], [âge]
FROM TableComplete
WHERE [âge]>=45 ;
```

Table obtenue après requête :

TableComplete Requête : Requête Sé

	nom	civilité	âge
	Einstein	marié	45
	Gandhi	célibataire	64
▶	Planck	Veuf	52

On a sélectionné toutes les personnes d'au moins 45 ans et l'on ne conserve que leur nom, leur civilité et leur âge.

Produit cartésien

S = **R x Q**
SQL : **SELECT * FROM** R , Q

Afin de ne pas présenter un exemple de table produit trop volumineuse, nous prendrons comme opérandes du produit cartésien, deux tables contenant peu d'enregistrements :

Personne : Table

nom	prénom	age	civilité
Einstein	Albert	45	marié
Lavoisier	Antoine	41	marié

Employeur : Table

nom	n°Insee
Université	12112472
Collège	25478550
Institut	54578559

Produit cartésien :
SELECT * FROM Employeur , Personne

Employeur Requête : Requête Sélection

Personne.nom	prénom	age	civilité	Employeur.nom	n°Insee
Einstein	Albert	45	marié	Université	12112472
Lavoisier	Antoine	41	marié	Université	12112472
Einstein	Albert	45	marié	Collège	25478550
Lavoisier	Antoine	41	marié	Collège	25478550
Einstein	Albert	45	marié	Institut	54578559
Lavoisier	Antoine	41	marié	Institut	54578559

Nous remarquons qu'en apparence l'attribut **nom** se retrouve dans le domaine des deux relations ce qui semble contradictoire avec l'hypothèse "Domaine(R) ∩ Domaine(Q) = ∅ (pas d'attributs en communs). En fait ce n'est pas un attribut commun puisque les valeurs sont différentes, il s'agit plutôt de deux attributs différents qui ont la même identification. Il suffit de préfixer l'identificateur par le nom de la relation (Personne.nom et Employeur.nom).

Combinaison d'opérateurs : projection - produit cartésien

SELECT Personne.nom , prénom, civilité, n°Insee **FROM** Employeur , Personne

On extrait de la table produit cartésien uniquement 4 colonnes :

Employeur Requête : Requête Sélection

nom	prénom	civilité	n°Insee
Einstein	Albert	marié	12112472
Lavoisier	Antoine	marié	12112472
Einstein	Albert	marié	25478550
Lavoisier	Antoine	marié	25478550
Einstein	Albert	marié	54578559
Lavoisier	Antoine	marié	54578559

Intersection, union, différence,

S = **R** ∩ **Q**
SQL : **SELECT * FROM** R **INTERSECT SELECT * FROM** Q

S = **R** ∪ **Q**
SQL : **SELECT * FROM** R **UNION SELECT * FROM** Q

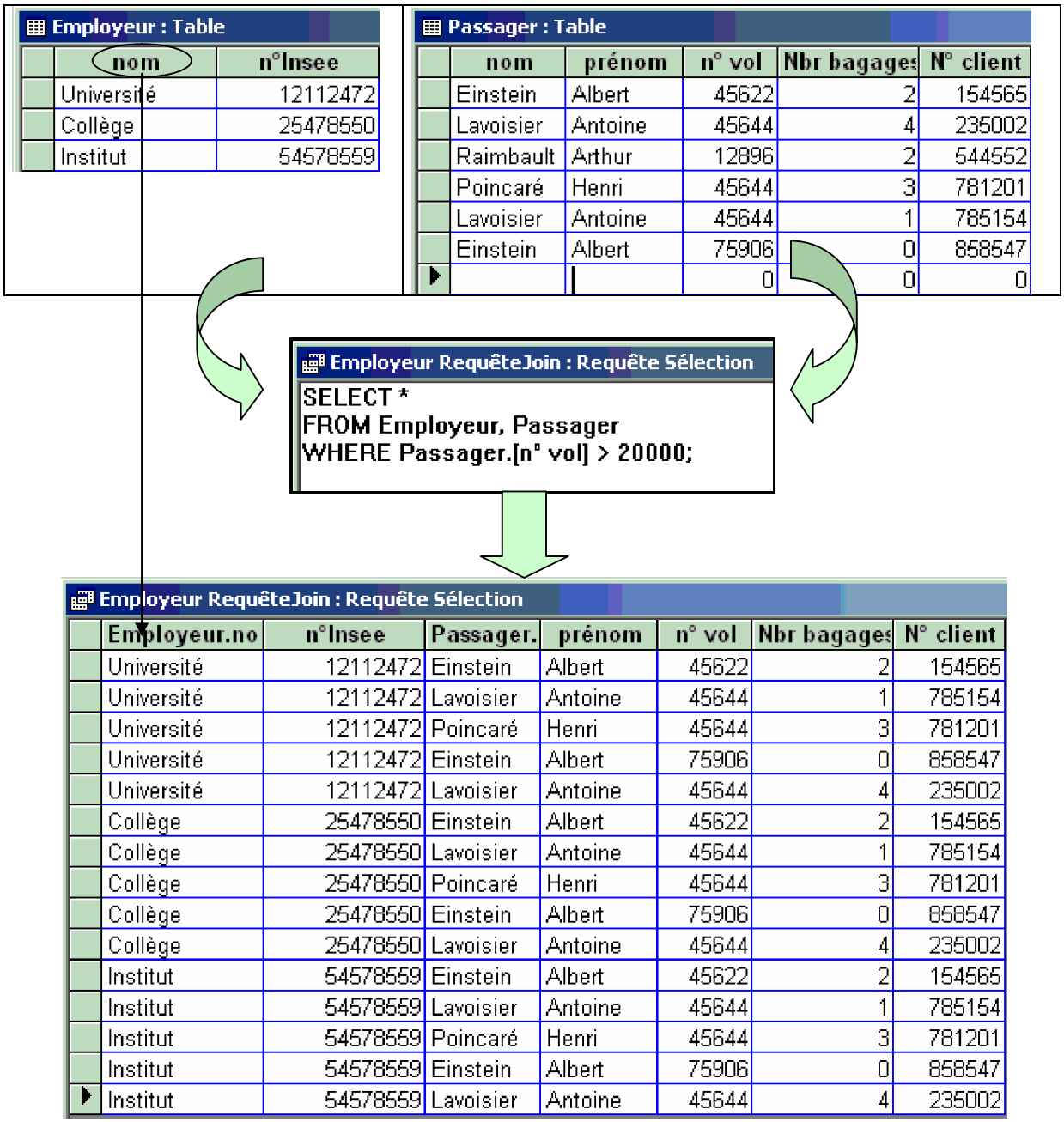
S = **R** – **Q**
SQL : **SELECT * FROM** R **MINUS SELECT * FROM** Q

Jointure de deux relations

Soient R et Q deux relations de domaine et de degré quelconques (degré(R) = n, degré(Q) = p), avec Domaine(R) ∩ Domaine(Q) = ∅ (pas d'attributs en communs).

La jointure **joint**(R,Q) = **select** (Cond(a_1, a_2 , … , a_{n+p}) , R x Q).

SQL : **SELECT** * **FROM** R ,Q **WHERE** Cond(a_1, a_2 , … , a_{n+p})



Remarque pratique importante

Le langage SQL est plus riche en fonctionnalités que l'algèbre relationnelle. En effet SQL intègre des possibilités de calcul (numériques et de dates en particulier).

Soit une table de tarifs de produit avec des prix hors taxe:

Tarifs : Table

Article	PrixHT
chaise	10,00 €
table	100,00 €
Téléviseur	1 000,00 €

1°) Usage d'un opérateur multiplicatif : calcul de la nouvelle table des tarifs TTC abondés de la TVA à 20% sur le prix hors taxe.

Tarifs : Table

Article	PrixHT
chaise	10,00 €
table	100,00 €
Téléviseur	1 000,00 €

Requête SQL : Tarifs TTc

Tarifs TTc : Requête Sélection

```
SELECT Article , PrixHT*1.20 AS PrixTTC
FROM Tarifs;
```

Tarifs TTc : Requête Sélection

Article	PrixTTC
chaise	12
table	120
Téléviseur	1200

2°) Usage de la fonction intégrée SUM : calcul du total des prix HT.

Tarifs : Table

Article	PrixHT
chaise	10,00 €
table	100,00 €
Téléviseur	1 000,00 €

Requête SQL : Total HT

Total HT : Requête Sélection

```
SELECT SUM(PrixHT) AS Total
FROM Tarifs;
```

Total HT : Requête Sélection

Total
1 110,00 €