Les Bases de Données ?

Pierre-François Bonnefoi — *bonnefoi@unilim.fr*

Organiser l'information

Exemple d'une société de VPC (Vente Par Correspondance) : elle vend différents produits à des clients

Qu'est ce qu'un client pour cette société ?

C'est une personne qui commande des produits à la société

Quels rapports y-a-t'ils entre un client et cette société ?

Il faut pouvoir envoyer au client les produits qu'il a commandé ainsi que la facture pour le réglement de cette commande

Comment identifier précisement un client ?

- par son nom
  - — Nom et prénom
- par son adresse (pour envoyer une facture ou le contenu d'une commande)
  - — Numéro de rue, Nom de la rue, Code postal, Ville
- par son numéro de téléphone

☛ l'ensemble de ces informations correspondent aux **coordonnées** du client.

✎ La société **doit connaître chaque nouveau client** par un même ensemble d'informations.

Exemple : Jean-Paul Denis; 45 rue de la lune 87000 Limoges; 05 55 12 34 56 *1er client*
Pauline Derue; 37 rue du salut 54000 Nancy; 03 83 12 34 56 *2e client*
etc.

Donner un sens à cette organisation

Un client est connu par ses coordonnées.

↕

Les coordonnées désigne un client.

☛ Il **existe un lien** entre les élements constituant **les coordonnées** et **le concept de client**.
Ce lien établit une **relation** entre les différents élements.

Exemple : Parler de client est équivalent à penser à **[** un nom, une adresse et un numéro de télephone **]** .

☛ Il est possible de **généraliser**.

Exemple : 03 83 12 34 56 est le numéro de téléphone de Pauline Derue.
05 55 12 34 56 est le numéro de téléphone de Jean-Paul Denis.

Généralisation ➜ Les « numéros de téléphones » des Clients sont : 03 83 12 34 56 et 05 55 12 34 56

☛ Il est possible **d'enrichir** ce qui définit un **client**.

Exemple : Ajout de l'adresse de courrier électronique à la notion de client

Jean-Paul Denis; 45 rue de la lune 87000 Limoges; 05 55 12 34 56; jp_denis@monmail.fr
Pauline Derue; 37 rue du salut 54000 Nancy; 03 83 12 34 56; p_derue@moncourrier.fr

☛ Il est possible **de restreindre** ce qui définit un **client**.

Exemple : Envoi d'un courrier promotionnel à chaque client
Jean-Paul Denis; 45 rue de la lune 87000 Limoges
Pauline Derue; 37 rue du salut 54000 Nancy

Définition d'une base de donnée

Une base de donnée permet de gérer un ensemble d'information.

- Ajout de nouvelles informations
- Suppression d'informations
- Consultation d'informations

✎ Ces différentes informations **doivent être organisées de la même manière**.

Exemple : Une base de données de clients **contient un ensemble d'information**.
Chacune de ces informations correspond **aux coordonnées** d'une personne.

Jean-Paul Denis; 45 rue de la lune 87000 Limoges; 05 55 12 34 56 *1er client*
Pauline Derue; 37 rue du salut 54000 Nancy; 03 83 12 34 56 *2e client*
etc.

La base de donnée est donnée sous forme d'**un ensemble de coordonnées**.

Ces **coordonnées** sont données sous la forme de l'**association** d'un nom, d'une adresse et d'un numéro de téléphone.

☛ Définition « abstraite » d'un client : CLIENT(NOM, ADRESSE, TELEPHONE).
Définition « concréte » d'un client : (J-Paul Denis; 45 rue de la lune 87000 Limoges; 05 55 12 34 56).

- Enrichissement de l'informations

Exemple : ajout d'un nouvel élément dans l'association définissant les coordonnées :
L' « adresse électronique »
Pauline Derue; 37 rue du salut 54000 Nancy; 03 83 12 34 56; p_derue@moncourrier.fr

D'où la nouvelle définition du concept de client : CLIENT(NOM, ADRESSE, TELEPHONE, E-MAIL)

Les bases de données

Ce sont des systèmes de gestion de l'information.

Leurs buts sont :
• de permettre à l'utilisateur **d'insérer**, de **modifier** et de **rechercher** de manière efficace des données spécifiques dans une grande masse d'informations.

• d'assurer une indépendance par rapport au matériel utilisé. L'utilisateur n'a pas à savoir où sont stockées les données et comment.

Les données peuvent être directement sur l'ordinateur employé ou bien de l'autre côté de l'Atlantique.

• d'assurer une indépendance par rapport à l'organisation des données. Chaque utilisateur doit pouvoir consulter les informations qu'il désire suivant ses propres besoins.

*Une base de données contenant des informations sur des personnes et des voitures doit pouvoir être consultée aussi bien si on s'interresse seulement aux **personnes possédant une voiture** ou bien seulement **aux voitures achetées dans l'année**.*

• d'être manipulable par des non-informaticiens.

On doit pouvoir décrire ce que l'on **souhaite sans avoir à** décrire **comment** l'obtenir.

Utilisation de SQL (Structured Query Language)

• de minimiser la redondance de l'informations.

Une information ne doit pas être dupliquée dans la base de données.

• d'assurer la cohérence des données.

Par exemple, une voiture doit toujours être saisie avec son numéro de plaque minéralogique, un conducteur doit être agé d'au moins 16 ans etc.

• de permettre d'établir des statistiques.

Combien de véhicules de marque « Renault » ont-il été vendus ?

Les Bases de données relationnelles

Ce sont les plus utilisées **75 % du marché** !

Organisation de l'information et l'accés à cette information :

• Les données sont représentées à l'aide de **relations**
• L'accés à ces données est basé sur l'**algèbre relationnelle**
• Un **langage déclaratif** (on indique seulement ce que l'on veut sans indiquer comment l'obtenir) permet d'organiser ces accés (*SQL Structured Query Language*).
• Un accès suivant une approche simplifiée est possible (*QBE Query by Example*).

Aide à la conception des relations :

• Utilisation de méthodes de conception : *Merise*, *OMT*…

☛ Plusieurs étapes sont nécessaires à la mise en place d'une base de données, dès lors que l'on a précisément défini ses besoins (*ce qui n'est déjà pas chose facile !*) :

• la **création de la structure de la base** sous forme de tables (tableaux de données) reliées entre elles par des données clés,

• la **conception des requêtes** qui permettront d'extraire ou de mettre à jour les informations qu'elle contient,

• **la conception de l'interface homme-machine** (écrans et états) qui rendra plus conviviale la saisie et la restitution des informations.

Gestion d'une assurance :

Pour gérer une assurance, il faut pouvoir disposer d'une liste de **personne** et d'une liste de **véhicules**.
Chacune de ces personnes a souscrit une assurance pour un véhicule donné.

Besoins :
• Informations concernant la **personne**
• Informations concernant le **véhicule**

Comment identifier une personne ?

Une **personne** possède un **nom**, une **adresse** (utile pour envoyer les factures) et un **age** (utile pour calculer la prime d'assurance).

Le **nom** est composé d'un **nom de famille** et d'un **prénom**.

☛ Une **personne** est identifiée par **[** un **nom de famille**, un **prénom**, une **adresse** et un **age]**.

Exemples : **Paul Dupond** habite à **Paris** et est agé de **4 0** ans.
Jean Durand habite également **Paris** et est agé de **2 8** ans.

Comment identifier un véhicule ?

Un véhicule possède une **marque**, une **puissance** exprimée en chevaux, une **couleur** de carrosserie et un **numéro** de plaque minéralogique.

☛ Un véhicule est identifié par **[** une **marque**, une **puissance**, une **couleur** et un **numéro]**

Exemples : Le véhicule immatriculé **45TH78** est de marque **Renault**, de **8** chevaux et de couleur **rouge**.
Le véhicule immatriculé **5647GT49** est de marque **Peugeot**, de **5** chevaux et de couleur **verte**.

Définition de la « Relation » et des « Attributs »

Les différentes informations « Paul Dupond », « Paris » et « 40 » définissent une **personne** au sens de notre gestion d'assurance.

☛ Ces informations sont **liées** et définissent une **relation**.

Exemples : la relation **VEHICULE (Numéro, Marque, Puissance, Couleur)**.
la relation **PERSONNE (Nom, Prénom, Adresse, Age)**.

✎ Une relation est **définie** par une combinaison **d'attributs** ou **d'arguments**.
Une relation est **exprimée** par un ensemble de **combinaisons de valeurs** pour chaque arguments

Exemples : La relation VEHICULE est exprimée par :
VEHICULE(45TH78, Renault, 8, Rouge) et VEHICULE(5647GT49, Peugeot, 5, Vert).
Deux combinaisons de valeurs
La relation PERSONNE est exprimée par :
PERSONNE(Dupond, Paul, Paris, 40) et PERSONNE(Durand, Jean, 28).
Également deux combinaisons de valeurs.

Définition d'une « Table »

Une manière plus pratique d'exprimée une relation est de la donner sous forme d'une **table** :

PERSONNE

Nom	Prénom	Adresse	Age
Dupond	Paul	Paris	40
Dupond	Jacques	Lyon	23
Martin	Georges	Dijon	55
Durand	Jean	Paris	28

VEHICULE

Numéro	Marque	Puissance	Couleur
902 FEE 75	Renault	7	Rouge
434 FR 92	Renault	5	Noir
5647 GT 49	Peugeot	5	Vert
5643 ER 92	Citroën	9	Noir

☛ La table **contient tous les éléments** constituants la relation.
*La table **est** le contenu de la base de donnée.*

Définition du « Domaine » d'un attribut

Dans la base de données :

La puissance d'un véhicule **doit correspondre** à une valeur acceptable pour une puissance.
La couleur d'un véhicule **doit correspondre** à un mot désignant bien une couleur de carrosserie.

☛ Les **choix possibles** de valeurs définissent le **domaine** de l'attribut.

Le domaine est alors **ENTIER** (que des nombres sans virgules) ou **CARACTERES** (n'importe quel mot) ou une combinaison des deux *etc*.

Exemple : L'attribut « Puissance » de la relation VEHICULE est **ENTIER** car il peut prendre une **valeur entière** comme 5, 6, 7… **mais pas** 5.78, Rapide, Lente…

L'attribut « Nom » de la relation PERSONNE est **CARACTERES**.

L'attribut « Couleur » de la relation VEHICULE **peut prendre** les valeurs Rouge, Noir, Jaune… **mais pas** Grand, Petit, Rond, 5, 598… Son domaine est **CARACTERES restreint** à seulement quelques mots.

☛ Le domaine peut être exprimé par un **ensemble de valeurs**.

Exemple : Le domaine de l'attribut « Couleur » de la relation VEHICULE est l'ensemble { Rouge, Noir, Vert }.
L'attribut « Couleur » ne peut prendre comme valeurs que Rouge, Noir ou Vert (mais pas Jaune).

Accéder à une partie seulement d'une table (ou relation)

Pour accéder à certains éléments d'une table (ou d'une relation), il faut **sélectionner une partie** de cette table.

Exemple : Sélection des véhicules de marque « Renault » dans la table VEHICULE
par l'expression : 'Marque = « Renault »'

Ce qui donne la « sous » table suivante :

NOUVELLE_RELATION

Numéro	Marque	Puissance	Couleur
902 FEE 75	Renault	7	Rouge
434 FR 92	Renault	5	Noir

☛ Cette opération s'appelle une **restriction**.
La table VEHICULE est restreinte par **le choix d'une valeur pour un attribut donné**.
Ici, l'attribut « Marque » est restreint à la valeur « Renault ».

✎ L'opération de restriction **conserve le même nombre d'attributs** de la relation initiale.
Les attributs restent (Numéro, Marque, Puissance, Couleur).

Une **nouvelle relation** est obtenue à partir de la relation après restriction.
Il y a juste moins d'éléments dans cette nouvelle relation.

Réduction de la définition d'une relation (ou table)

Les **attributs d'une relation** peuvent être **sélectionnés indépendemment** les uns des autres.

Exemple : Sélection des attributs de « Marque » et de « Puissance » de la relation VEHICULE

Ce qui donne la « sous » table suivante :

NOUVELLE_RELATION

Marque	Puissance
Renault	7
Renault	5
Peugeot	5
Citroën	9

☛ Cette opération s'appelle une **projection**.

Les attributs autres que « Marque » et « Puissance » sont projettés hors de la table.

Explication légérement fantaisiste mais satisfaisante…

✎ L'opération de Projection **conserve le même nombre d'éléments** que dans la relation initiale.

Il y a toujours 4 éléments dans la table.

Une **nouvelle relation** est obtenue à partir de la relation après projection.

Il y a juste moins d'attibuts dans cette nouvelle relation.

Combinaison de la projection et de la restriction d'une relation

Une relation peut subir à la fois une opération de restriction et de projection.
Il n'y a pas d'ordre pour appliquer les deux opérations.

Exemple : Sélection des attributs de « Marque » et de « Puissance » de la relation VEHICULE
Sélection des véhicules de marque « Renault » dans la table VEHICULE

Ce qui donne la « sous » table suivante :

NOUVELLE_RELATION

Marque	Puissance
Renault	7
Renault	5

Le **nombre d'attributs** et le **nombre d'éléments** de la relation ont diminué.

Une **nouvelle relation** est obtenue à partir de la relation après projection et restriction.

Retour sur la gestion d'assurance

Deux tables sont disponibles :

- une table PERSONNE
- une table VEHICULE

Ce qui manque ?

Savoir à qui appartient tel ou tel véhicule

Comment résoudre le problème ?

Associer un véhicule à une personne

Définition de la nouvelle relation CONDUCTEUR

À quelle opération cela correspond il ?

À l'opération de **jointure**

☛ L'opération de **Jointure** correspond à la **combinaison** de **deux relations** pour obtenir **une nouvelle relation.**

✎ Cette opération **n'est possible que** si les deux relations à joindre possédent **un attribut commun.**

Il **est nécessaire de** définir **comment les deux relations vont correspondre.**

*Ici, une **personne** possède un **vehicule**, c'est le **conducteur** du véhicule*

Problème de l'identification d'un élément d'une relation

Comment être sûr d'associer à la bonne personne le bon véhicule ?

Il faut pouvoir **rendre unique chaque élément** de la relation (ou table) PERSONNE.
Deux personnes peuvent s'appeller de la même façon, résider au même endroit…

☛ Un **attribut** doit être ajouté à la relation.
Ses valeurs doivent être données **arbitrairement** et **automatiquement** par la base de donnée lors de l'entrée des différents éléments.

✎ Chaque valeur de ce nouvelle attribut est **différente** pour chaque élément de la table.
Une valeur **identifie sans ambigüité** un élément de la table.
Il est **choisi automatiquement par la base de donnée** et **non par l'utilisateur**.
Ce qui évite les erreurs.

☛ Cet attribut est appelé **Clé** de la relation.

Exemple : La clé est l'attribut « Nom_Unique » qui correspond au « nom » suivi d'un numéro de saisie.

PERSONNE

Nom	Prénom	Adresse	Age	Nom_Unique
Dupond	Paul	Paris	40	DUPOND1
Dupond	Jacques	Lyon	23	DUPOND2
Martin	Georges	Dijon	55	MARTIN1
Durand	Jean	Paris	28	DURAND1

Ajout de référence à une table vers une autre table

Chaque véhicule peut être associé avec une personne de manière unique.

Pour chaque élément de la relation VEHICULE, une **référence est ajoutée**.

La relation VEHICULE est enrichie d'un nouvel attribut « NOM_COND » dont les valeurs correspondent **aux clés** de la relation PERSONNE

Une nouvelle relation VEHICULE est obtenue :

VEHICULE

Numéro	Marque	Puissance	Couleur	Nom_Cond
902 FEE 75	Renault	7	Rouge	DUPOND1
434 FR 92	Renault	5	Noir	MARTIN1
5647 GT 49	Peugeot	5	Vert	MARTIN1
5643 ER 92	Citroën	9	Noir	DURAND1

☛ Un **lien entre deux relations** est défini par l'**ajout d'une référence** dans l'une de ces deux tables.
Une des tables est appelée **relation référençante**. *Ici, la relation VEHICULE*
L'autre table est appelée **relation référençée**. *Ici, la relation PERSONNE.*
L'**attribut de référence** est aussi appelé **clé étrangère**. *Ici, l'attribut « Nom_Cond ».*

✎ Chaque véhicule **ne possède qu'un** seul conducteur.
Un conducteur **peut posséder plusieurs** véhicules.

Opération de jointure

Rappel :

☛ L'opération de **Jointure** correspond à la **combinaison** de **deux relations** pour obtenir **une nouvelle relation.**

✎ Cette opération **n'est possible que** si les deux relations à joindre possédent **un attribut commun.**

Il **est nécessaire de** définir **comment les deux relations vont correspondre.**

Une nouvelle relation, CONDUCTEUR_DE_VEHICULE est obtenue à partir de la relation VEHICULE et PERSONNE à partir de l'expression 'Nom_Cond = Nom_Unique' :

CONDUCTEUR_DE_VEHICULE

Numéro	Marque	Puissance	Couleur	Nom_Cond	Nom	Prénom	Adresse	Age
902 FEE 75	Renault	7	Rouge	DUPOND1	Dupond	Paul	Paris	40
434 FR 92	Renault	5	Noir	MARTIN1	Martin	Georges	Dijon	55
5647 GT 49	Peugeot	5	Vert	MARTIN1	Martin	Georges	Dijon	55
5643 ER 92	Citroën	9	Noir	DURAND1	Durand	Jean	Paris	28

Opération de Semi-Jointure

Cette opération est **similaire à l'opération de jointure**, **sauf qu'elle ne conserve que les attributs** d'une des deux relations intiales.

Exemple : Sélection des personnes conduisant effectivement un véhicule et conservation des attributs de la relation personne.

La nouvelle relation obtenue est :

NOUVELLE_RELATION

Nom	Prénom	Adresse	Age	Nom_Unique
Dupond	Paul	Paris	40	DUPOND1
Dupond	Jacques	Lyon	23	DUPOND2
Martin	Georges	Dijon	55	MARTIN1

L'élément de clé « DURAND1 » n'apparait pas dans la nouvelle relation car il ne possède pas de véhicule.

✎ Cette opération de semi-jointure correspond à la restriction d'une relation par les valeursd'un attribut d'une autre relation.

Sur l'exmple, on a choisit l'attribut « Nom_Cond » pour restreindre la relation PERSONNE.
Ainsi, toute personne ne possédant pas de voiture à été enlevée dans la nouvelle relation.

Gestion simplifiée des étapes du Tour de France 97 :

Le **modèle relationnel** est un modèle d'organisation des données sous forme de **Tables,** (*Tableaux de valeurs*) où chaque Table représente une **Relation**, *au sens mathématique d'Ensemble*.

Exemple présenté : le Tour de France édition 97, où figurent **l'ensemble** des **Équipes**, des **Coureurs**, des **Étapes**, des **Temps réalisés** par les coureurs à chacune des étapes, et enfin l'ensemble des **Pays**.

Description des équipes

CodeEquipe	NomEquipe	Directeur Sportif
BAN	BANESTO	Eusebio UNZUE
COF	COFIDIS	Cyrille GUIMARD
CSO	CASINO	Vincent LAVENU
FDJ	LA FRANCAISE DES JEUX	Marc MADIOT
FES	FESTINA	Bruno ROUSSEL
GAN	GAN	Roger LEGEAY
ONC	O.N.C.E.	Manolo SAIZ
TEL	TELEKOM	Walter GODEFROOT
…	…	…

✎ Les **colonnes** des tables s'appellent des **attributs**.
Les **lignes** des « **n-uplets** »
(où n est le degré de la relation, c'est à dire le nombre d'attributs de la relation).
Un **attribut** ne prend qu'**une seule valeur** pour **chaque n-uplet**.
L'**ordre des lignes** et **des colonnes n'a pas d'importance**.

Clés primaire

☛ Chaque **table** doit avoir **une clé primaire** constituée par un ensemble **minimum d'attributs** permettant de distinguer chaque n-uplet de la Relation par rapport à tous les autres.

Chaque ensemble de valeurs formant la clé primaire d'un n-uplet est donc unique au sein d'une table.

Description des Coureurs

NuméroCoureur	NomCoureur	CodeEquipe	CodePays
8	ULLRICH Jan	TEL	ALL
31	JALABERT Laurent	ONC	FRA
61	ROMINGER Tony	COF	SUI
91	BOARDMAN Chris	GAN	G-B
114	CIPOLLINI Mario	SAE	ITA
151	Olano Abraham	BAN	ESP
…	…	…	…

Exemple : dans la table COUREURS, **chaque coureur** a un NuméroCoureur **différent**.

✎ Dans certains cas, **plusieurs clés primaires** sont possibles pour une seule table.
On parle alors de **clés candidates**. *Il faut alors en choisir une comme clé primaire.*

Description des Étapes

NuméroEtape	DateEtape	VilleDépart	VilleArrivée	NbKm
1	06-jui-97	Rouen	Forge-les-Eaux	192
2	07-jui-97	St-Valéry-en-Caux	Vire	262
3	08-jui-97	Vire	Plumelec	224
…	…	…	…	…

Pays participant

CodePays	Pays
ALL	ALLEMAGNE
AUT	AUTRICHE
BEL	BELGIQUE
DAN	DANEMARK
ESP	ESPAGNE
FRA	FRANCE
G-B	GRANDE BRETAGNE
ITA	ITALIE
P-B	PAYS-BAS
RUS	RUSSIE
SUI	SUISSE
…	…

Liens sémantiques

Les **liens sémantiques** (ou **règles de gestion sur les données**) existants entre les ensembles sont réalisés par l'intermédiaire de **clés étrangères** faisant elles-mêmes **référence à des clés primaires** d'autres tables.

Description des Coureurs

NuméroCoureur	NomCoureur	CodeEquipe	CodePays
8	ULLRICH Jan	TEL	ALL
3 1	JALABERT Laurent	ONC	FRA
6 1	ROMINGER Tony	COF	SUI
9 1	BOARDMAN Chris	GAN	G-B
1 1 4	CIPOLLINI Mario	SAE	ITA
1 5 1	Olano Abraham	BAN	ESP
…	…	…	…

Exemple : Dans la table COUREURS, la **clé étrangère** CodeEquipe (faisant référence à la clé primaire de même nom dans la table EQUIPES) traduit les deux règles de gestion suivantes :

« Un COUREUR appartient à une EQUIPE**†**»
« Une EQUIPE est composée de plusieurs COUREURS »

Différents types de liens sémantiques

Il existe deux grands types de liens :
• « Un - Plusieurs » *Voir exemple précédent.*
• « Plusieurs - Plusieurs ».

La **réalisation de ce dernier type de liens**, un peu plus complexe, passe par l'**utilisation d'une table intermédiaire** dont la **clé primaire** est formée des **clés étrangères** des tables qu'elle relie.

Temps

NuméroCoureur	NuméroEtape	TempsRéalisé
8	3	04:54:33
8	1	04:48:21
8	2	06:27:47
31	3	04:54:33
31	1	04:48:37
31	2	06:27:47
61	1	04:48:24
61	2	06:27:47
91	3	04:54:33
…	…	…

Exemple : la table des TEMPS réalisés à chaque étape par chacun des coureurs exprime les deux règles de gestion suivantes : « Un COUREUR participe à plusieurs ETAPES »
« Une ETAPE fait participer plusieurs COUREURS »

Formalisation algébrique

Le modèle relationnel est le plus souvent décrit sous la forme suivante :

- les clés primaires étant soulignées
- les clés étrangères marquées par un signe distinctif (ici *).

EQUIPES (CodeEquipe, NomEquipe, DirecteurSportif)

COUREURS (NuméroCoureur, NomCoureur, CodeEquipe*, CodePays*)

ETAPES (NuméroEtape, VilleDépart, VilleArrivée, NbKm)

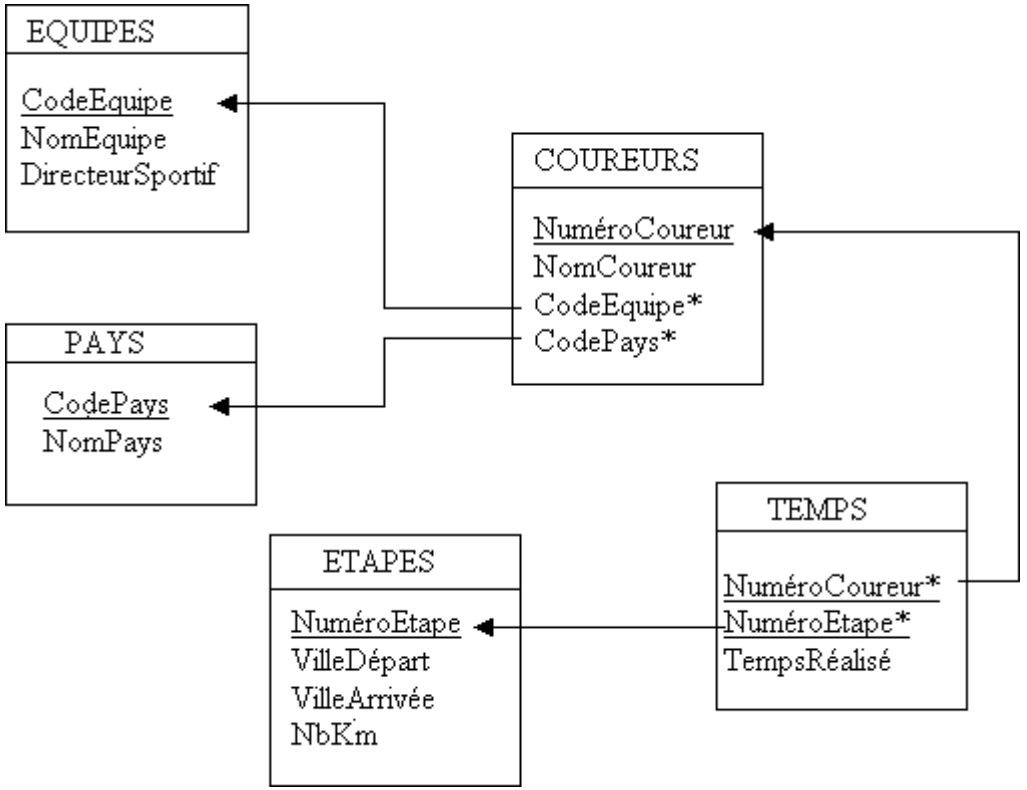
TEMPS (NuméroCoureur*, NuméroEtape*, TempsRéalisé)

PAYS (CodePays, NomPays)

Formalisation graphique

Cette représentation graphique facilite l'analyse et la conception de la base de donnée.

✎ **MCD** (Modèle Conceptuel de Données) est un model abstrait de la méthode **Merise**.



Exemple : Un COUREUR appartient à une EQUIPE
Une EQUIPE est composée de plusieurs COUREURS

Un COUREUR est originaire d'un PAYS
Un PAYS est représenté par plusieurs COUREURS

Un COUREUR participe à plusieurs ETAPES
Une ETAPE fait participer plusieurs COUREURS

Méthodologie

La **conception d'une base de données relationnelle** passe d'abord par **l'identification** :

- des objets de gestion *(Coureurs, Etapes, …)*,
- des règles de gestion du domaine modélisé *(interviews des utilisateurs, étude des documents manipulés, des fichiers existants, …)*.

Une fois **énoncées** et **validées**, ces règles conduisent **automatiquement** à la **structure du modèle relationnel correspondant.**

L'algèbre Relationnelle

• Elle a été définie par Codd en 1970.

• Elle est à l'origine du langage **SQL** (*Structured Query Language*) d'IBM,
langage d'interrogation et de manipulation de tous les SGBDR actuels
(**Oracle**, **Sybase**, **Informix**, **Ingres**, **DB2**, **Access** et tous les autres).

• Elle est également mise en œuvre de manière plus conviviale à travers le langage **QBE** (*Query By Example*) que l'on retrouve seulement sur certains SGBDR (**Access** par exemple).

☛ Une bonne maîtrise de l'algèbre relationnelle permet de concevoir n'importe quelle requête **aussi complexe soit elle** avant de la mettre en œuvre à l'aide du langage QBE ou SQL.

Parmi les opérations de l'algèbre relationnelle, on dispose :

• d'opérations propres (**projection**, **sélection**, **jointure**, **division**),

• d'opérations classiques sur les ensembles (**union**, **intersection**, **différence**, **produit cartésien**),

• des opérations de **calcul**, de **regroupement**, de **comptage** et de **tri**.

Opération PROJECTION

Formalisme : R = PROJECTION (R1, liste des attributs)

Et en langage SQL :

SELECT DISTINCT liste d'attributs FROM table ; *le DISTINCT permet de supprimer les doublons*

SELECT liste d'attributs FROM table ;

Exemples :

Champignons

Espèce	Catégorie	Conditionnement
Rosé des prés	Conserve	Bocal
Rosé des prés	Sec	Verrine
Coulemelle	Frais	Boîte
Rosé des prés	Sec	Sachet plastique

- Cet opérateur ne porte que sur **1 relation**.
- Il permet de ne retenir que **certains attributs** spécifiés d'une relation.
- On obtient tous les n-uplets de la relation à l'exception des doublons.

□

Exemple de Projection

R1 = PROJECTION (CHAMPIGNONS, Espèce)

R1

Espèce
Rosé des prés
Rosé des prés
Coulemelle
Rosé des prés

R2 = PROJECTION (CHAMPIGNONS, Espèce, Catégorie)

R2

Espèce	Catégorie
Rosé des prés	Conserve
Rosé des prés	Sec
Coulemelle	Frais
Rosé des prés	Sec

Exemples en SQL :

```
SELECT DISTINCT Espèce FROM Champignons ;

SELECT DISTINCT Espèce, Catégorie FROM Champignons ;
```

La clause DISTINCT permet d'éliminer les doublons.

Opération Sélection

Formalisme : R = SELECTION (R1, condition)

Et en langage SQL :

SELECT * FROM table WHERE condition ;

Exemple :

R3 = SELECTION (CHAMPIGNONS, Catégorie = "Sec")

R3

Espèce	Catégorie	Conditionnement
Rosé des prés	Sec	Verrine
Rosé des prés	Sec	Sachet plastique

- Cet opérateur porte sur **1 relation**.
- Il permet de ne retenir que les n-uplets répondant à une condition exprimée à l'aide des **opérateurs arithmétiques** (=, >, <, >=, <=, <>) ou **logiques de base** (ET, OU, NON).
- **Tous les attributs** de la relation sont conservés.

Exemple de Sélection en SQL

Exemple en SQL :

SELECT * FROM Champignons WHERE Catégorie="Sec" ;

☛ La condition de sélection exprimée derrière la clause WHERE peut être spécifiée à l'aide :
des opérateurs de comparaison : =, >, <, <=, >=, <>
des opérateurs logiques : AND, OR, NOT
des opérateurs : IN (*appartenance à un ensemble*), BETWEEN (*appartenance à un intervalle*),
LIKE (*présence de certains caractères*).

Autres exemples :

Soit la table ETUDIANT(N°Etudiant, Nom, Age, CodePostal, Ville)

```
SELECT *
FROM ETUDIANT
WHERE Age IN (19, 20, 21, 22, 23) ;
```

```
SELECT *
FROM ETUDIANT
WHERE Age BETWEEN 19 AND 23 ;
```

```
SELECT *
FROM ETUDIANT
WHERE CodePostal LIKE '42%' ;    sous Access : LIKE "42*"
```

```
SELECT *
FROM ETUDIANT
WHERE CodePostal LIKE '42___' ;    sous Access : LIKE "42???"
```

Opération de jointure (équi-jointure)

Formalisme : R = JOINTURE (R1, R2, condition d'égalité entre attributs)

Et en langage SQL :

il est possible d'enchaîner plusieurs jointures dans la même instruction SELECT :

```
SELECT * FROM table1, table2, table3, ...
WHERE  table1.attribut1=table2.attribut1  AND  table2.attribut2=table3.attribut2  AND  ...;
```

Produit

CodeProd	Libellé	Prix unitaire
590A	HD 1.6Go	1615
515J	LBP 660	1820
588J	Scanner HP	1700

Detail_Commande

N°cde	CodeProd	Quantité
97001	590A	2
97002	515J	1
97003	515J	3

- Cet opérateur porte sur **2 relations** qui doivent avoir au moins un attribut défini dans le même domaine (*ensemble des valeurs permises pour un attribut*).
- La condition de jointure peut porter sur **l'égalité d'un** ou **de plusieurs attributs** définis dans le même domaine (*mais n'ayant pas forcément le même nom*).
- Les n-uplets de la relation résultat sont formés par la **concaténation** des n-uplets des relations d'origine qui vérifient la condition de jointure.

Des jointures plus complexes que l'équi-jointure peuvent être réalisées en généralisant l'usage de la condition de jointure à d'autres critères de comparaison que l'égalité (<,>, <=,>=, <>).

Exemple de jointure

R = JOINTURE (PRODUIT, DETAIL_COMMANDE, Produit.CodeProd=Detail_Commande.CodeProd)

R

CodeProd	Libellé	Prix unitaire	N°cde	Quantité
590A	HD 1,6Go	1615	97001	2
515J	LBP 660	1820	97002	1
515J	LBP 660	1820	97003	3

En SQL :

```
SELECT * FROM Produit, Détail_Commande
WHERE Produit.CodePrd=Détail_Commande.CodePrd ;
```

ou en utilisant des **alias** pour les noms des tables :

```
SELECT * FROM Produit A, Détail_Commande B
WHERE A.CodePrd=B.CodePrd ;
```

Opération d'Union

Formalisme : R = UNION (R1, R2)

Et en langage SQL :
Opération UNION

SELECT liste d'attributs FROM table1
UNION
SELECT liste d'attributs FROM table 2 ;

• Cet opérateur porte sur **deux relations** qui doivent avoir le **même nombre d'attributs** définis dans le même domaine (*ensemble des valeurs permises pour un attribut*).

*On parle de relations ayant le même **schéma**.*

• La relation résultat possède les attributs des relations d'origine et les n-uplets de chacune, avec élimination des doublons éventuels.

Exemple :

Etudiants_Info

N°Etudiant	Nom_etudiant
1	Dupont
3	Durand
4	Martin
5	Bertrand

Etudiants_Anglais

N°Etudiant	Nom_etudiant
1	Dupont
4	Martin
6	Michel

Exemple d'union

On désire obtenir l'ensemble des étudiants inscrits en « Info » **ou** en « Anglais ».

Etudiants = UNION (Etudiants_Info, Etudiants_Anglais)

Etudiants

N°Etudiant	Nom_etudiant
1	Dupont
3	Durand
4	Martin
5	Bertrand
6	Michel

Exemple en SQL :

```
SELECT n°etudiant, Nom_etudiant FROM Etudiants_Info
UNION
SELECT n°etudiant, Nom_etudiant FROM Etudiants_Anglais ;
```

Opération d'intersection

Formalisme : R = INTERSECTION (R1, R2)

Et en langage SQL :

```
SELECT attribut1, attribut2, ... FROM table1
INTERSECT
SELECT attribut1, attribut2, ... FROM table2 ;
```

ou

```
SELECT attribut1, attribut2, ... FROM table1
WHERE attribut1 IN (SELECT attribut1 FROM table2) ;
```

Exemple :

On désire obtenir l'ensemble des étudiants inscrits en « Info » **e t** en « Anglais ».

Etudiants = INTERSECTION (Etudiants_Info, Etudiants_Anglais)

Etudiants

N°Etudiant	Nom_etudiant
1	Dupont
4	Martin

- Cet opérateur porte sur **deux relations** de même schéma.
- La relation résultat possède les attributs des relations d'origine et les n-uplets communs à chacune.

Exemple d'Intersection en SQL

```
SELECT n°etudiant, Nom_etudiant FROM Etudiants_Info
INTERSECT
SELECT n°etudiant, Nom_etudiant FROM Etudiants_Anglais ;
```

ou

```
SELECT N°etudiant, Nom_Etudiant FROM Etudiants_Info
      WHERE N°etudiant IN (SELECT n°etudiant FROM Etudiants_Anglais) ;
```

Opération de différence

Formalisme : R = DIFFERENCE (R1, R2)

Exemple :

On désire obtenir l'ensemble des étudiants inscrits en « Info » **mais pas** en « Anglais ».

Etudiants

N°Etudiant	Nom_etudiant
3	Durand
5	Bertrand

Et en langage SQL :

```
SELECT attribut1, attribut2, ... FROM table1
EXCEPT                                          ou MINUS
SELECT attribut1, attribut2, ... FROM table2 ;
```

ou

```
SELECT attribut1, attribut2, ... FROM table1
WHERE attribut1 NOT IN (SELECT attribut1 FROM table2) ;
```

- Cet opérateur porte sur **deux relations** de même schéma.
- La relation résultat possède les attributs des relations d'origine et les n-uplets de la première relation qui n'appartiennent pas à la deuxième.

Attention ! DIFFERENCE (R1, R2) ne donne pas le même résultat que DIFFERENCE (R2, R1)

Opération de Produit Cartésien

Formalisme : R = PRODUIT (R1, R2)

Et en langage SQL :

SELECT * FROM table1, table2 ;

• Cet opérateur porte sur **deux relations**.

• La relation résultat possède les attributs de chacune des relations d'origine et ses n-uplets sont formés par la concaténation de chaque n-uplet de la première relation avec l'ensemble des n-uplets de la deuxième.

Exemple :

Etudiants

N°Etudiant	Nom_etudiant
101	Dupont
102	Martin

Epreuves

Libellé_epreuve	coefficient
Informatique	2
Mathématiques	3
Gestion financière	5

Exemple de Produit Cartésien

Examen = PRODUIT (Etudiants, Epreuves)

Examen

N°Etudiant	Nom_etudiant	Libellé_epreuve	coefficient
101	Dupont	Informatique	2
101	Dupont	Mathématiques	3
101	Dupont	Gestion financière	5
102	Martin	Informatique	2
102	Martin	Mathématiques	3
102	Martin	Gestion financière	5

En SQL :

SELECT * FROM Etudiants, Epreuves ;

Opération de Tri

R = TRI (R0, att1A, att2 A, …)

Et en langage SQL :

```
SELECT attribut1, attribut2, attribut3, …
FROM table
ORDER BY attribut1 ASC, attribut2 DESC, … ;
```

ASC : par ordre croissant (*Ascending*)
DESC : par ordre décroissant (*Descending*)

Exemple :

Champignons

Espèce	Catégorie	Conditionnement
Rosé des prés	Conserve	Bocal
Rosé des prés	Sec	Verrine
Coulemelle	Frais	Boîte
Rosé des prés	Sec	Sachet plastique

R1 = TRI (CHAMPIGNONS, EspèceA, CatégorieA, ConditionnementA)

Champignons_triés

Espèce	Catégorie	Conditionnement
Rosé des prés	Conserve	Bocal
Rosé des prés	Sec	Sachet plastique
Rosé des prés	Sec	Verrine
Coulemelle	Frais	Boîte

- Le tri s'effectue sur un ou plusieurs attributs, dans l'ordre croissant ou décroissant.
- La relation résultat a la même structure et le même contenu que la relation de départ.

Opération de Tri en SQL

Exemple :

```
SELECT Espèce, Catégorie, Conditionnement
FROM Champignons
ORDER BY Espèce DESC, Catégorie ASC, Conditionnement ASC ;
```

Par défaut le tri se fait par ordre croissant si l'on ne précise pas ASC ou DESC.

Principe d'écriture d'une requête

La plupart des **requêtes** (ou *interrogations*) portant sur une base de données relationnelle **ne peuvent pas être réalisées** à partir d'**une seule opération** mais en **enchaînant** successivement plusieurs opérations.

Exemple :

Soient les deux tables (ou *relations*) suivantes :

CLIENT(CodeClient, NomClient, AdrClient, TélClient)
COMMANDE(N°Commande, Date, CodeClient*)

*Les clés primaires sont soulignées et les clés étrangères sont marquées par **

On désire obtenir le code et le nom des clients ayant commandé le 10/06/97 :

R1=SELECTION(COMMANDE, Date=10/06/97)
R2=JOINTURE(R1, CLIENT, R1.CodeClient=CLIENT.CodeClient)
R3=PROJECTION(R2, CodeClient, NomClient)

En SQL :

Une même instruction SELECT permet de combiner Sélections, Projections, Jointures.

```
SELECT DISTINCT CLIENT.CodeClient, NomClient
FROM CLIENT, COMMANDE
WHERE CLIENT.CodeClient=COMMANDE.CodeClient AND Date='10/06/97';
```

*Si l'on ne précisait pas CLIENT.CodeClient au niveau du SELECT, la commande SQL ne pourrait **pas s'exécuter**.*
*En effet il y aurait **ambiguïté** sur le CodeClient à projeter : celui de la table CLIENT ou celui de la table COMMANDE ?*

L'opérateur SELECT de SQL

SELECT *attr1*, *attr2*, ... *attrN*
FROM *nom_table*
[**WHERE** <*conditions*>] **;**

où <*conditions*> est un ensemble de conditions simples séparées par des opérateurs logiques **AND** et **OR**.

Exemples : Soient les relation Etudiants (N°Etudiant, Nom, Age, Ville) et Suit(N°Etudiant, Nom_Module, Moyenne);

sans partie "WHERE" :

—› *Le nom de tous les étudiants :*

```
SELECT Nom FROM Etudiants ;
```

Une condition simple peut porter :

- sur une comparaison avec les opérateurs classiques **>**, **>**, **> =**, **< =**, **=**, **< >**

  —› Numéros des étudiants ayant une moyenne >= 10 en Info.

```
SELECT N°Etudiant
FROM Suit
WHERE Nom_Module = 'Info' AND Moyenne >= 10 ;
```

- sur l'appartenance à un ensemble avec l'opérateur **IN**

  —› Liste des noms des étudiants habitant Limoges, Poitiers ou Brive.

```
SELECT Nom
FROM Etudiants
WHERE Ville in ('Limoges', 'Poitiers', 'Brive') ;
```

- sur l'appartenance à un intervalle avec l'opérateur **BETWEEN**

  —› Noms des étudiants ayant entre 20 et 24 ans

```
SELECT Nom
FROM Etudiants
WHERE Age BETWEEN 20 AND 24 ;
```

• sur la présence de certains caractères dans une chaîne, avec l'opérateur **LIKE 'chaîne'**.

Dans la chaîne de caractères : le caractère ***** remplace n'importe quelle chaîne de caractère.
le caractère **?** remplace n'importe quel caractère.

—› Noms des étudiants dans une ville commençant par un B

```
SELECT Nom
FROM Etudiants
WHERE Ville like 'B*' ;
```

La négation d'une condition se traduit par **not**. Par exemple **NOT IN** ou **NOT LIKE**.

—› Noms et villes des étudiants n'habitant pas Limoges ou Poitiers

```
SELECT Nom, Ville
FROM Etudiants
WHERE Ville NOT IN ('Limoges', 'Poitiers') ;
```

Il est également possible de trier les résultats au moyen de l'opérateur **ORDER BY**

—› Liste des noms et villes des étudiants n'habitant pas Limoges,
ordonnée par ordre alphabétique des villes, puis des noms (*pour les étudiants d'une même ville*)

```
SELECT Nom, Ville
FROM Etudiants
WHERE Ville <> 'Limoges'
ORDER BY Ville, Nom ;
```

Sous-Requêtes

Requêtes imbriquées : une condition peut également contenir des requêtes (*sous-requêtes*)

—› Nom des étudiants habitant la même ville que l'étudiant « Dupond »

```
SELECT Nom
FROM Etudiants
WHERE Ville = (SELECT Ville FROM Etudiants WHERE Nom = 'Dupond') ;
```

Attributs calculés et Renommés

Atributs calculés

Un attribut calculé est un attribut dont les valeurs sont obtenues par des opérations arithmétiques portant sur des attributs de la même relation.
Le calcul est spécifié lors d'une projection ou lors de l'utilisation d'une fonction.

R=PROJECTION(R0, att1, att2, att3, att4, att1*att2, att3/att2)

Attributs renommés

Il est possible de renommer n'importe quel attribut en le faisant précéder de son nouveau nom suivi de ":".

R=PROJECTION(R0, att1, att2, att3, att4, nouvel_att1:att1*att2, nouvel_att2:att3/att2)

Exemple :

Ligne_commande

N°BonCommande	CodeProdu	Quantité	PuHT
96008	A10	10	83
96008	B20	35	32
96009	A10	20	83
96010	A15	4	110
96010	B20	55	32

Exemples d'Attributs calculés et Renommés

R=PROJECTION(LIGNE_COMMANDE, N°BonCommande, CodeProduit, Montant:Quantité*PuHt)

R

N°BonCommande	CodeProduit	PuHT
96008	A10	830
96008	B20	1120
96009	A10	1660
96010	A15	440
96010	B20	1760

Et en langage SQL :

Attributs calculés

```
SELECT N°BonCommande, CodeProduit, Quantité*PuHt
FROM LIGNE_COMMANDE ;
```

Attributs renommés

```
SELECT N°BonCommande, CodeProduit, Quantité*PuHt AS Montant
FROM LIGNE_COMMANDE ;
```

Opération CALCULER

R=CALCULER(R0, fonction1, fonction2, ...) ou N=CALCULER(R0, fonction)

Et en langage SQL :

```
SELECT fonction1(attribut1), fonction2(attribut2), ...
FROM table ;
```

- Les calculs et/ou comptage portent sur la relation R0.
- La relation résultat ne comportera **qu'une ligne** avec autant de colonnes que de résultats demandés ou pourra simplement être considérée comme un nombre N, utilisable ultérieurement en tant que tel, dans le cas où un seul résultat est attendu.

Exemple :

On désire obtenir le chiffre d'affaires total Ht, ainsi que le nombre total de produits commandés :

Chiffre_affaires=CALCULER(LIGNE_COMMANDE, Somme(Quantité*PuHt), Somme(Quantité))

Chiffre_affaires

Somme(Quantité*PuHT)	Somme(Quantité)
5810	124

En SQL :

```
SELECT SUM(Quantité*PuHt), SUM(Quantité)
    FROM  LIGNE_COMMANDE;
```

Opération Regrouper_et_Calculer

R=REGROUPER_ET_CALCULER(R0, att1, att2, ..., fonction1, fonction2, ...)

Et en langage SQL :

SELECT attribut1, attribut2, … , fonction1(attribut3), fonction2(attribut4), …
FROM table
GROUP BY attribut1, attribut2, … ;

- Le regroupement s'effectue sur un **sous ensemble des attributs** de la relation R0.
- La relation résultat comportera **autant de lignes** que de groupes de n-uplets, les fonctions s'appliquant à chacun des groupes séparément.

Exemple :

On désire obtenir le montant total Ht de chaque bon de commande :
Montant_Total=REGROUPER_ET_CALCULER(LIGNE_COMMANDE, N°BonCommande, MontantHt : Somme(Quantité*PuHt))

Montant_total

N°BonCommande	MontantH
96008	1950
96009	1660
96010	2200

Exemple Regrouper_et_Calculer en SQL

```
SELECT N°BonCommande, SUM(Quantité*PuHt)
        FROM  LIGNE_COMMANDE
        GROUP  BY  N°BonCommande  ;
```

Les attributs figurant dans la clause GROUP BY, doivent obligatoirement figurer dans la clause SELECT.

Il est possible de sélectionner des lignes issues d'un regroupement (grâce à la clause **HAVING**).

Exemple :

On souhaite, parmi l'ensemble des commandes, ne retenir que celles dont la montant total hors taxes est supérieur à 10000.

```
SELECT N°BonCommande, SUM(Quantité*PuHt)
        FROM  LIGNE_COMMANDE
        GROUP  BY  N°BonCommande  HAVING  SUM(Quantité*PuHt)>10000 ;
```

Les Fonctions

Les **fonctions** sont utilisées dans les opérateurs **CALCULER** et **REGROUPER_ET_CALCULER**.

Les fonctions de calcul

Les fonctions de calculs portent sur un ou plusieurs groupes de n-uplets et évidemment sur un attribut de type numérique.

Somme(attribut) : *total des valeurs d'un attribut*
Moyenne(attribut) : *moyenne des valeurs d'un attribut*
Minimum(attribut) : *plus petite valeur d'un attribut*
Maximum(attribut) : *plus grande valeur d'un attribut*

La fonction de comptage : Comptage()

La fonction de comptage donne le nombre de n-uplets d'un ou de plusieurs groupes de n-uplets. Il n'est donc pas nécessaire de préciser d'attribut.

Et en langage SQL :

Les fonctions de calcul

SUM(attribut) : *total des valeurs d'un attribut*
AVG(attribut) : *moyenne des valeurs d'un attribut*
MIN(attribut) : *plus petite valeur d'un attribut*
MAX(attribut) : *plus grande valeur d'un attribut*

La fonction de comptage

COUNT(*) : nombre de n-uplets
COUNT(DISTINCT attribut) : nombre de valeurs différentes de l'attribut

Exemple de calculs en SQL

Exemple : Soient les relation Etudiants (N°Etudiant, Nom, Age, Ville) et Suit(N°Etudiant, Nom_Module, Moyenne);

—› Liste des numéros d'étudiant et des moyennes majorées de 1 point pour les étudiants ayant eu moins de 5 en Anglais

```
SELECT N°Etudiant, moyenne + 1
FROM Suit
WHERE Nom_Module = 'Anglais' AND Moyenne < 5 ;
```

—› Plus petite, plus grande moyenne et moyenne générale dans le module Anglais

```
SELECT Min(Moyenne), Max(Moyenne), Avg(Moyenne)
FROM Suit
WHERE Nom_Module = 'Anglais' ;
```

—› Nombre d'étudiants suivant le module Info

```
SELECT Count(*)
FROM Suit
WHERE Nom_module = 'Anglais' ;
```

—› Nom des villes et pour chacune de celles-ci le nombre d'étudiants y habitant

```
SELECT Ville, Count(*)
FROM Etudiants
GROUP BY Ville ;
```

—› Noms des villes où habitent plus de 10 étudiants

```
SELECT Ville, Count(*)
FROM Etudiants
GROUP BY Ville
HAVING Count(*) > 10 ;
```

Utilisation des Sous-Requêtes

Une sous-requête peut fournir un ensemble de réponses.

☛ Il faut savoir si on utilise **tous** les résultats (*ALL*) ou **certains** (*ANY*).

Exemples de sous-requête :

… WHERE Reference > 125
peut être remplacé par une sous-requête :

… WHERE Reference > (SELECT Reference FROM …)
La sous-requête doit renvoyer un seul résultat.

… WHERE Reference IN (SELECT Reference FROM …)
Ici la sous-requête renvoie plusieurs valeurs. Pour obtenir un succès, la valeur testée doit être présente dans le résultat de la sous-requête. La forme NOT IN est aussi autorisée.

… WHERE Reference > ALL (SELECT Reference FROM …)
Ici la sous-requête renvoie plusieurs valeurs. Pour obtenir un succès, la valeur testée doit être supérieure à toutes les valeurs obtenues pour la sous-requête.

… WHERE Reference > ANY (SELECT Reference FROM …)
Ici la sous-requête renvoie plusieurs valeurs. Pour obtenir un succès, la valeur testée doit être supérieure à au moins une des valeurs obtenues pour la sous-requête.

… WHERE EXISTS (SELECT Reference FROM …)
Ici la sous-requête renvoie plusieurs valeurs. Pour obtenir un succès, la sous-requête doit donner un succès. La forme NOT EXISTS est aussi autorisée.

Exemples

Soient les relation Etudiants (N°Etudiant, Nom, Age, Ville) et Suit(N°Etudiant, Nom_Module, Moyenne);

—› Nom des étudiants de Limoges ayant un âge inférieur à l'âge maximum des étudiants de Poitiers.

```
SELECT Nom
FROM Etudiants
WHERE Ville = 'Limoges'
        AND Age < ANY ( SELECT Age
                        FROM Etudiants
                        WHERE Ville = 'Poitiers') ;
```

—› Nom des étudiants de Limoges les plus jeunes

```
SELECT Nom
FROM Etudiants
WHERE Ville = 'Limoges'
        AND Age <= ALL( SELECT Age
                        FROM Etudiant
                        WHERE Ville = 'Limoges') ;
```

—› Nom des étudiants qui suivent au moins au cours

```
SELECT Nom
FROM Etudiants
WHERE EXISTS (  SELECT *
                FROM Suit
                WHERE N°Etudiant = Etudiants.N°Etudiant);
```

Exercices

A — Soit le modèle relationnel suivant relatif à une base de données sur des représentations musicales :

REPRESENTATION (n°représentation, titre_représentation, lieu)
MUSICIEN (nom, n°représentation*)
PROGRAMMER (date, n°représentation*, tarif)

*Les clés primaires sont soulignées et les clés étrangères sont marquées par **

Questions :

1 - Donner la liste des titres des représentations.
2 - Donner la liste des titres des représentations ayant lieu à l'opéra Bastille.
3 - Donner la liste des noms des musiciens et des titres des représentations auxquelles ils participent.
4 - Donner la liste des titres des représentations, les lieux et les tarifs pour la journée du 14/09/96.

B — Soit le modèle relationnel suivant relatif à la gestion des notes annuelles d'une promotion d'étudiants :

ETUDIANT(N°Etudiant, Nom, Prénom)
MATIERE(CodeMat, LibelléMat, CoeffMat)
EVALUER(N°Etudiant*, CodeMat*, Date, Note)

*Les clés primaires sont soulignées et les clés étrangères sont marquées par **

Questions :

1 - Quel est le nombre total d'étudiants ?
2 - Quelles sont, parmi l'ensemble des notes, la note la plus haute et la note la plus basse ?
3 - Quelles sont les moyennes de chaque étudiant dans chacune des matières ?
4 - Quelles sont les moyennes par matière ?
5 - Quelle est la moyenne générale de chaque étudiant ?
6 - Quelle est la moyenne générale de la promotion ?
7 - Quels sont les étudiants qui ont une moyenne générale supérieure ou égale à la moyenne gén. de la promotion ?