REQUÊTES
DANS LES BASES DE DONNÉES

1. Présentation succincte des bases de données

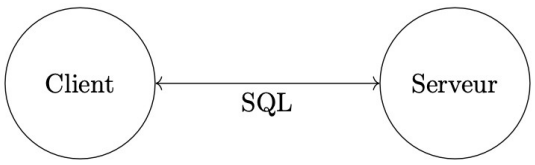
1.1 Rôle des bases de données

Les Systèmes de Gestion de Bases de Données (SGBD) sont des logiciels permettant entre autres :

- d'avoir une structure de données efficace (modification des données sans modifications des structures d'interrogation, une rapidité d'accès, d'éviter les informations redondantes,...)
- une sauvegarde des modifications
- une gestion des pannes
- une gestion des conflits si plusieurs utilisateurs modifient la base en même temps
- protection des données
- d'éviter à l'utilisateur d'avoir à s'interroger sur la manière dont sont stockées les données
- ….

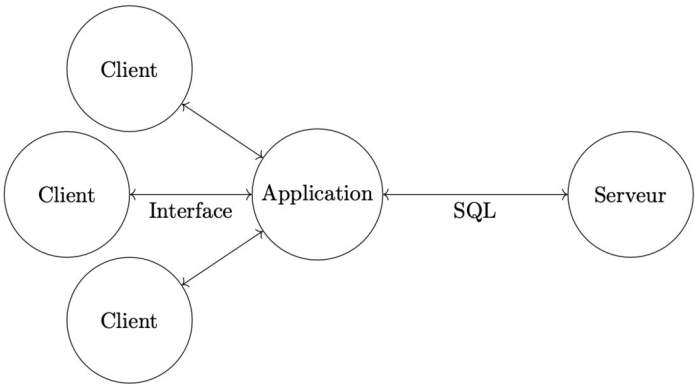
1.2 Architecture client-serveur

Les bases de données sont articulées sous la forme client-serveur : l'utilisateur travaille sur un poste (client) qui peut être éloigné de l'ordinateur qui gère les données (serveur).
Il est nécessaire de permettre un dialogue entre ces deux parties.



Une normalisation a permis d'unifier le langage utilisé dans les bases de données : **SQL** (Structured Query Language).
Du point de vue des utilisateurs, la syntaxe est la même, en tout cas pour les fonctionnalités de base ce qui permet de pouvoir utiliser la grande majorité des logiciels de base de données. Par contre, la programmation en interne de ces logiciels dépend de l'éditeur (Oracle, IBM, Microsoft,…).

En pratique, le langage SQL n'est en général pas visible pour les usagers. En effet, l'utilisation des bases de données se fait usuellement à travers une architecture « trois-tiers ».
Entre l'usager et la base de données se trouve un serveur applicatif qui traduit les demandes de l'utilisateur (en général via une interface graphique) au gestionnaire de bases de données.



Par exemple, lorsque l'on se connecte à Pronote, tout le côté « bases de données » est invisible pour l'utilisateur, seule l'application Pronote est disponible. C'est elle qui notamment gère les droits d'accès : un élève ne peut consulter les notes d'un autre, tandis qu'un professeur de mathématiques ne peut rentrer des notes de physique-chimie…

1.3 Vocabulaire des bases de données

Dans ce cours, nous allons étudier quelques bases de données simples (pouvant être représentées par des tableaux à double entrées).
Essentiellement ici, on verra comment interroger une base de données, sans en créer ni en modifier.

1.3.1 Modélisation en tableau

Ce modèle est très simple : c'est celui d'un tableau à deux dimensions. Celui-ci se rencontre souvent lorsque l'on traite de données, par exemple :

- un répertoire (nom, téléphone, adresse, …)
- une fiche de bibliothèque (auteur, titre, année,…)
- un carnet de commande (client, article, quantité, date, prix,…)
- Un lot de pommes (numéro, masse, couleur)
- Un parc de voiture (immatriculation, pays d'immatriculation, couleur, nombre de places, propriétaire)

On représente ces données dans un tableau, par exemple dans un tableur. Par convention, dans ce cours, les données d'un élément (une fiche de bibliothèque, une commande, une pomme, une voiture...) seront sur une même ligne, les colonnes donnant les différents attributs (auteur, titre, année, par exemple).

Exemples

- Un panier de pommes

identifiant	masse	diamètre	couleur
1	151 g	8.3 cm	rouge
2	169 g	9.1 cm	jaune
3	134 g	8.0 cm	jaune

- Un parc de voitures

immatriculation	pays_immatriculation	couleur	nombre_places	propriétaire
2983-AA	France	bleue	5	Luc
1923-BD	France	verte	7	Lucia
PLT-28-190	Portugal	bleue	7	Gaël
2383-ZN	France	noire	9	Leïla
1209-NQ	France	noire	5	Leïla
2634-OS	France	bleue	5	Gaël

- **Attributs**
  - Les différents titres de colonnes sont appelés attributs.
  - On les notera formellement A_1 , A_2 , … , A_p .
  - Les attributs forment un ensemble : il n'y a donc pas d'attribut en double.
  - L'ordre des attributs n'est pas fixé : on ne parle pas du premier attribut mais par exemple de l'attribut nombre_places.

- **Domaine**

  L'ensemble des valeurs que peut prendre un attribut A est son domaine qu'on pourra noter $Dom(A)$.
  Par exemple, pour la table ci-dessus, certains attributs peuvent avoir pour domaine des chaînes de caractères, mais on peut imaginer que le domaine des pays d'immatriculation est réduit à l'ensemble des noms de pays, le nombre de places est un entier (qu'on peut imposer dans l'intervalle $[1 ; 7]$).

  Remarque Tout cela a été fixé au préalable, à la création de la base.

- **Schéma relationnel**

  On appelle schéma relationnel un ensemble d'attributs, vérifiant toujours la contrainte que les attributs sont distincts deux à deux.

  Dans ce cours on notera généralement S un schéma relationnel.

- **Tuple, p-uplet, enregistrement**

  Chaque ligne s'appelle un p-uplet (ou tuple en anglais) ou un enregistrement (parfois on parle d'instance de relation)
  Chaque enregistrement peut donc être représentée par un élément de :
  $Dom(A_1) \times Dom(A_2) \times \ldots \times Dom(A_p)$.

- **Relation**

  On appelle relation (ou table) un ensemble fini de p-uplets, d'enregistrements de $Dom(A_1) \times Dom(A_2) \times \ldots \times Dom(A_p)$, qu'on notera souvent R dans ce cours.

- **Base de données relationnelle**

  Une base de données relationnelles est un ensemble de relations.

Résumé panier de pommes

Pomme

identifiant	masse	diamètre	couleur	nom_variété
14	142 g	7.3 cm	rouge	Ariane
25	182 g	7.5 cm	rouge	Gala
16	140 g	7.9 cm	rouge	Ariane
[...]	[...]	[...]	[...]	[...]

Variete

libellé	prix_au_kilo	maturation	goût
Ariane	3.19	tardive	sucré/acidulé
Gala	3.49	précoce	sucré
Reinette	3.19	mi-saison	sucré
Boskoop	2.99	mi-saison	acidulé
[...]	[...]	[...]	[...]

1.4 Contraintes

Elles sont nombreuses mais on retiendra en particulier :

- **On ne peut avoir d'attribut en double**.
  Il n'en est en effet pas question ! Ils risqueraient d'être affectés de valeurs différentes dans des tuples.
  (Par exemple 2 immatriculations différentes pour un même véhicule)
- **On ne peut avoir deux enregistrements égaux dans le tableau** (ce qui serait d'ailleurs synonyme de redondance des données)
  Autrement dit : **deux enregistrements doivent différer d'au moins un attribut.**
- L'ordre des lignes n'est pas fixé.
  Bien entendu une présentation des données dans un tableau aura un ordre des attributs et un ordre des tuples mais ces ordres de présentations sont dus à l'ordre physique des données, ou le résultat d'un traitement particulier lors d'une requête. En particulier, il n'est pas garanti a priori par le SGBD.

1.5 Clés

1.5.1 Clés primaires

Comme on ne peut avoir deux enregistrements identiques, pour garantir leur non-répétition , les bases de données utilisent le concept de clé (qui doit être pensé dès la conception des bases de données et indiqué à la création).

Exemple 1 base de données relative à un panier de pommes

identifiant	masse	diamètre	couleur
1	151 g	8.3 cm	rouge
2	169 g	9.1 cm	jaune
3	134 g	8.0 cm	jaune
4	134 g	8.0 cm	jaune

Comme deux pommes peuvent avoir des attributs identiques (ici les pommes 3 et 4), on les différencie grâce à leur identifiant.
L'attribut "identifiant" permet donc d'identifier justement un enregistrement de manière unique.

Exemple 2 base de données relative à un parc de voitures

immatriculation	pays_immatriculation	couleur	nombre_places	propriétaire	prop_num_secu	prop_num_voiture
2983-AA	France	bleue	5	Luc	1134212	1
1923-BD	France	verte	7	Lucia	2021726	1
PLT-28-190	Portugal	bleue	7	Gaël	1012612	1
2383-ZN	France	noire	9	Leïla	2125312	2
1209-NQ	France	noire	5	Leïla	2125312	1
2634-OS	France	bleue	5	Gaël	1162732	1

où :

- **prop_num_secu** : numéro de sécurité sociale du propriétaire (chaque personne de nationalité française possède un numéro de sécurité sociale unique)
- **prop_num_voiture** : si un propriétaire possède plusieurs voitures, il peut les numéroter pour les repérer. S'il n'en possède qu'une, prop_num_voiture vaut 1. Par exemple, Leïla numérote ici ses voitures de cette manière : " 1209-NQ est ma première voiture, et 2383-ZN est ma voiture numéro 2 ".

i) L'attribut "immatriculation" permet-il d'identifier un enregistrement de la relation "parc de voitures" ?

ii) Et le groupe d'attributs [immatriculation ; pays_immatriculation] ?

iii) Et le groupe [immatriculation ; pays_immatriculation ; couleur] ?

Iv) Et le groupe [prop_num_secu] ?

v) Et le groupe [prop_num_secu ; prop_num_voiture] ?

Définition

- Quand plusieurs attributs/groupe d'attributs sont possibles permettent d'identifier les enregistrements d'une relation, on les appelle des **clés candidates**.
- Parmi les clés candidates, on en choisit une (et une seule) minimale qui servira de référence : c'est la **clé primaire**.
- Si aucune clé candidate n'est pas simple ni intelligible, on crée une **clé artificielle**, souvent appelée **identifiant**.

Remarques

- Une clé primaire ne peut avoir de valeurs nulles.
- Lors de la création d'une base de données, on ne peut donc avoir deux clés primaires pour une relation
- Une clé est dite unique si elle permet d'identifier de manière unique un enregistrement mais peut accepter une valeur nulle. À la différence d'une clé primaire, on peut en définir plusieurs pour une même relation.

1.5.2 Clé étrangère

Exemple

Pomme

identifiant	masse	diamètre	couleur	nom_variété
14	142 g	7.3 cm	rouge	Ariane
25	182 g	7.5 cm	rouge	Gala
16	140 g	7.9 cm	rouge	Ariane
[...]	[...]	[...]	[...]	[...]

Variete

libellé	prix_au_kilo	maturation	goût
Ariane	3.19	tardive	sucré/acidulé
Gala	3.49	précoce	sucré
Reinette	3.19	mi-saison	sucré
Boskoop	2.99	mi-saison	acidulé
[...]	[...]	[...]	[...]

i) Donner une clé primaire de la relation "variété".

ii) L'attribut "nom_variété" est-il une clé candidate de la relation "pomme" ?

iii) Quel est le prix au kg de la pomme 16 ?

iv) Que permet l'attribut "nom_variété" de la relation "pomme" ?

Définition (dans le cadre du programme)

Une **clé étrangère** d'une relation R_1 est un attribut qui est lié à une clé primaire d'une autre relation R_2.

Dans l'exemple précédent, "nom_variété" est une clé étrangère de la relation "pomme" référençant l'attribut "libellé" qui est une clé primaire de la relation "variété".

Une clé étrangère sert donc à lier des relations entre elles.

2 ALGÈBRE RELATIONNELLE SIMPLE

L'algèbre relationnelle sert à formaliser les questions que l'on peut poser à une relation ou, comme plus tard, à un ensemble de relations. Elle exprime ce que l'on fait ; le calcul relationnel exprime ce que l'on veut.

Définition
L'algèbre relationnelle est un ensemble d'opérateurs que l'on peut appliquer à des relations, et dont le résultat est une relation.

Comme le résultat est toujours une relation on pourra combiner les opérateurs : on forme ainsi, à partir d'opérateurs élémentaires, des requêtes élaborées. L'objectif est de pouvoir exprimer toute manipulation raisonnable par une expression algébrique des opérateurs élémentaires.

Exemples

ÉLÈVES 2018-2019

prénom	nom	filière	numero	lycée_origine	note_bac
Mathilde	Dufour	PCSI	2	Calmette	18
Léa	Dupond	MPSI	2	Massena	14
Paul	Dugommier	PCSI	1	Massena	12
Mathilde	Dugommier	MPSI	1	Calmette	15
Clément	Durand	PCSI	1	Parc impérial	13

ÉLÈVES 2019-2020

prénom	nom	filière	numero	lycée_origine	note_bac
Thomas	Arbois	MPSI	2	Parc impérial	14
Julie	Bazin	PCSI	2	Massena	15
Paul	Carrier	PCSI	1	Parc impérial	13
Léna	Durand	PCSI	1	Calmette	19
Philippe	Moreau	MPSI	1	Parc impérial	12

Voici des exemples de questions qu'on peut demander au serveur de bases de données.

- Combien d'élèves de Masséna en 2018-2019 ? pour les deux années scolaires ?
- Quelle est la moyenne au baccalauréat des élèves de Masséna sur les deux années scolaires ?
- Qui sont ceux qui ont obtenu une mention ?
- Etc...

2.1 Opérateurs ensemblistes

Un premier type d'opérateur combine 2 relations qui ont le même schéma ; on les utilisera surtout pour assembler les résultats d'autres requêtes. R et R' sont deux relations ayant le même schéma (c'est-à-dire les mêmes attributs).

- **Union R∪R'**

  est la relation de même schéma dont les tuples sont ceux qui appartiennent à R ou à R'.

  Exemple

ÉLÈVES 2018-2019

prénom	nom	filière	numero	lycée_origine	note_bac
Mathilde	Dufour	PCSI	2	Calmette	18
Léa	Dupond	MPSI	2	Massena	14
Paul	Dugommier	PCSI	1	Massena	12
Mathilde	Dugommier	MPSI	1	Calmette	15
Clément	Durand	PCSI	1	Parc impérial	13

ÉLÈVES 2019-2020

prénom	nom	filière	numero	lycée_origine	note_bac
Thomas	Arbois	MPSI	2	Parc impérial	14
Julie	Bazin	PCSI	2	Massena	15
Paul	Carrier	PCSI	1	Parc impérial	13
Léna	Durand	PCSI	1	Calmette	19
Philippe	Moreau	MPSI	1	Parc impérial	12

ÉLÈVES

prénom	nom	filière	numero	lycée_origine	note_bac
Mathilde	Dufour	PCSI	2	Calmette	18
Léa	Dupond	MPSI	2	Massena	14
Paul	Dugommier	PCSI	1	Massena	12
Mathilde	Dugommier	MPSI	1	Calmette	15
Clément	Durand	PCSI	1	Parc impérial	13
Thomas	Arbois	MPSI	2	Parc impérial	14
Julie	Bazin	PCSI	2	Massena	15
Paul	Carrier	PCSI	1	Parc impérial	13
Léna	Durand	PCSI	1	Calmette	19
Philippe	Moreau	MPSI	1	Parc impérial	12

- **Intersection R∩R'**

  est la relation de même schéma dont les tuples sont ceux qui appartiennent à R et à R'.

  Exemple

Carnet d'adresses 1

nom	prénom	numero	mail
Dufour	Mathilde	06 65 34 45	Jdufour@orange.fr
Dupond	Léa	07 55 41 00	LeaD@gmail.com
Dugommier	Paul	04 45 21 23	polo@orange.fr
Dugommier	Mathilde	01 45 96 75	Mathil75@free.fr
Durand	Clément	06 67 87 54	DClem04@free.fr

Carnet d'adresses 2

nom	prénom	numero	mail
Arbois	Thomas	03 67 94 45	Tharbois@yahoo.fr
Dupond	Léa	07 55 41 00	LeaD@gmail.com
Carrier	Paul	06 44 65 43	paul.carrier@laposte.net
Dugommier	Mathilde	01 45 96 75	Mathil75@free.fr
Moreau	Philippe	07 86 35 28	DClem04@free.fr

Carnet d'adresses

nom	prénom	numero	mail
Dupond	Léa	07 55 41 00	LeaD@gmail.com
Dugommier	Mathilde	01 45 96 75	Mathil75@free.fr

- **Différence R \ R'**

  est la relation de même schéma dont les tuples sont ceux qui appartiennent à R mais pas à R' .

  Exemple

  Carnet adresses 1 \ Carnet adresses 2

Carnet d'adresses

nom	prénom	numero	mail
Dufour	Mathilde	06 65 34 45	Jdufour@orange.fr
Dugommier	Paul	04 45 21 23	polo@orange.fr
Durand	Clément	06 67 87 54	DClem04@free.fr

2.2 Opérateurs spécifiques

Les opérateurs étudiés ici sont ceux qui utilisent la structure des relations.

Par exemple la question « Quels sont les élèves en MPSI 1 ? » se traduit par : donner les noms et prénoms (projection) des élèves de Masséna étant dans la classe 831 (sélection). On reprend notre table suivante ici :

ÉLÈVES 2018-2019

prénom	nom	filière	numero	lycée_origine	note_bac
Mathilde	Dufour	PCSI	2	Calmette	18
Léa	Dupond	MPSI	2	Massena	14
Paul	Dugommier	PCSI	1	Massena	12
Mathilde	Dugommier	MPSI	1	Calmette	15
Clément	Durand	PCSI	1	Parc impérial	13

- **Projection**

  La projection consiste à ne garder qu'une partie des attributs.

  Exemple La projection de la relations élèves suivant les attributs nom et note_bac donne :

nom	note_bac
Dufour	18
Dupond	14
Dugommier	12
Dugommier	15
Durand	13

Remarque

Rappelons que dans une relation, les tuples forment un ensemble. Il se peut donc que la projection ait moins d'éléments que la table initiale. (En fait le nombre d'éléments est conservé si et seulement si l'ensemble d'attributs X est une clé candidate de la relation).

Par exemple, la projection de la relations élèves suivant les attributs nom donne :

nom
Dufour
Dupond
Dugommier
Durand

- **Sélection**

La sélection (ou restriction) consiste à ne garder que les tuples qui vérifient une propriété.

Exemple La sélection de la relation élèves venant du lycée Massena donne :

ÉLÈVES **2018-2019**					
prénom	**nom**	**filière**	**numero**	**lycée_origine**	**note_bac**
Léa	Dupond	MPSI	2	Massena	14
Paul	Dugommier	PCSI	1	Massena	12

Sélection et projection vont souvent ensemble : si on s'intéresse aux noms et prénoms des élèves de MPSI, on obtient :

nom	filière
Dupond	MPSI
Dugommier	MPSI

- **Renommage**

Le renommage consiste à renommer un attribut. Ce sera utile lors de produits de tables lorsque deux tables ont des attributs portant le même nom mais correspondent à des données différentes.

3. Interroger une base de données : effectuer une requête en SQL

2.1 Introduction

La présentation des bases de données via l'algèbre relationnelle est abstarit : dans la pratique chaque éditeur d'un logiciel de gestion de base de données (deezer, pronote, ...) organise les données afin d'optimiser leur accès, mais ceci est transparent pour l'utilisateur. Voir 1. 2.

Pour la communication entre le serveur applicatif et le serveur de bases de données, il s'est produit un événement rare en informatique : une norme universelle a été établie, qui permet d'écrire des requêtes de la même façon quel que soit le logiciel. Le langage utilisé est :

S Q L pour **S**tructured **Q**uery **L**angage.

Ce langage reprend la structure de l'algèbre relationnelle en y ajoutant des moyens de calculs et autres améliorations (ordonnancement des résultats, par exemple). Ce langage est très proche du langage humain (du moins l'anglais).

2.2 Syntaxe

2.2.1 Les requêtes de base

Les représentations des relations se font avec le modèle du tableau.
En SQL on parlera de :

- tables à la places de relations ;
- colonnes à la places d'attributs ;
- lignes à la places de tuples.

2.2.1.1 La requête SELECT

La forme générale d'une requête en SQL est

```
SELECT ... FROM table ;
```

Les mots-clés de SQL sont usuellement écrits en majuscule mais ce n'est pas obligatoire. Le mot-clef principal d'une requête dans une base de données est **SELECT**, qu'on retrouvera en tête de toutes nos requêtes SQL.
Les requêtes se terminent par un point-virgule (cela dépend di logiciel utilisé).

2.2.1.2 Projection

Basiquement, **SELECT** permet de faire une projection (attention à ne pas confondre avec la sélection). Il suffit d'indiquer les attributs que l'on veut garder juste après **SELECT** :

SELECT A1,...,Ap **FROM** table ;

Les attributs (colonnes) à garder sont énumérés et séparés par une virgule. Si on ne veut pas effectuer de projection (c'est à dire garder tous les attributs), on peut utiliser * au lieu d'énumérer tous les attributs un à un.

SELECT * **FROM** table ;

Le mot-clef **FROM** permet de spécifier le nom de la table à utiliser.

Attention : en SQL, les résultats d'une requête ne forment pas une table car les doublons résultant de projections ne sont pas supprimés. Il faut utiliser le mot clef **DISTINCT** pour supprimer les doublons.

2.2.1.3 Sélection

La sélection se fait avec le mot-clef **WHERE**, placé après le nom de la table.
Comme en Python, on utilise les comparateurs = et != pour l'égalité et la différence.
Si le domaine de l'attribut le permet, on peut utiliser d'autres comparateurs (>, <, >=, <=) et même des fonctions arithmétiques. Une condition complexe peut être exprimée à l'aide de conditions plus simples et des des connecteurs logiques ET, OU, et NON (en anglais en SQL : AND, OR, NOT)

2.2.1.4 Renommage

On peut renommer un ou plusieurs attributs avec AS ou même en juxtaposant le nouveau nom à droite de l'ancien. Ceci sera particulièrement utile lorsqu'on utilisera plusieurs tables dont les noms d'attributs sont les mêmes, ou lorsqu'on fera des sous-requêtes. La syntaxe générale est :

SELECT A1 **AS** B1,..,Ai **AS** Bi, C1,..,Cj **FROM** table ;

Le **AS** est facultatif, mais c'est plus lisible.

2.2.2 Agrégats et fonctions d'agrégation

Les résultats d'une requête seront souvent utilisés ensuite, par exemple à des fins statistiques. SQL contient la possibilité de faire quelques uns de ces calculs. Les fonctions disponibles (de base) sont, parmi d'autres :

- le comptage, c'est-à-dire le nombre de lignes : **COUNT**
- le maximum des éléments dans une colonne : **MAX**
- le minimum des éléments dans une colonne : **MIN**
- la somme des éléments d'une colonne : **SUM**
- la moyenne des éléments d'une colonne (sum/count) : **AVG**

Le résultat est une table avec une unique ligne et une unique colonne, que l'on peut utiliser comme valeur.
Si f est une de ces fonctions on l'emploie sous la forme

SELECT f(attribut) **FROM** table **WHERE** condition ;

Par exemple, avec la relation eleve, **SELECT** AVG(notebac) **FROM** eleves produit 14.4.
Le mot-clef **GROUP BY** sert à indiquer sur quels attributs sont effectués les regroupements.

Remarque deux petites subtilités :

- on peut spécifier si l'on veut le résultat dans l'ordre croissant ou décroissant à l'aide des mots clés **ASC** et **DESC** (pour ascendant et descendant) juste après l'expression.
- si on met plusieurs expressions après **ORDER BY**, séparés par des virgules (typiquement plusieurs arguments), la relation est triée pour l'ordre lexicographique (on compare la première expression, puis en cas d'égalité la deuxième, etc.)

2.3 Requêtes sur plusieurs tables

Les requêtes SQL décrites dans cette section permettent de manipuler simultanément plusieurs tables et d'exprimer les opérations binaires de l'algèbre relationnelle : jointure, produit cartésien, union, intersection, différence.

2.3.1 Jointures

La jointure est une des opérations les plus utiles (et donc une des plus courantes) puisqu'elle permet d'exprimer des requêtes portant sur des données réparties dans plusieurs tables. La syntaxe pour exprimer des jointures avec SQL est une extension directe de celle étudiée précédemment dans le cas des sélections simples : on donne simplement la liste des tables concernées dans la clause **FROM**, et on exprime les critères de rapprochement entre ces tables dans la clause **WHERE**.

Exemple On donne le schéma relationnel suivant :

idClient	station	début	nbPlaces
10	Passac	1998-07-01	2
30	Santalba	1996-08-14	5
20	Santalba	1998-08-03	4
30	Passac	1998-08-15	3
30	Venusa	1998-08-03	3
20	Venusa	1998-08-03	6
30	Farniente	1999-06-24	5
10	Farniente	1998-09-05	3

La table *Séjour*

id	nom	prénom	ville	région	solde
10	Fogg	Phileas	Londres	Europe	12465
20	Pascal	Blaise	Paris	Europe	6763
30	Kerouac	Jack	New York	Amérique	9812

La table *Client*

Requête donner le nom des clients avec le nom des stations où ils ont séjourné.

Le nom du client est dans la table Client, l'information sur le lien client/station dans la table Séjour.

Deux tuples de ces tables peuvent être joints s'ils concernent le même client, ce qui peut s'exprimer à l'aide de l'identifiant du client. On obtient la requête :

SELECT nom, station **FROM** Client, Sejour **WHERE** id = idClient

On peut remarquer qu'il n'y a pas dans ce cas d'ambiguïté sur les noms des attributs : nom et id viennent de la table Client, tandis que station et idClient viennent de la table Séjour. En réalité, il arrive fréquemment qu'un même nom d'attribut soit partagé par plusieurs tables impliquées dans une jointure. Dans ce cas on résout l'ambiguïté en préfixant l'attribut par le nom de la table.

Exemple

nomStation	capacité	lieu	région	tarif
Venusa	350	Guadeloupe	Antilles	1200
Farniente	200	Seychelles	Océan Indien	1500
Santalba	150	Martinique	Antilles	2000
Passac	400	Alpes	Europe	1000

La table *Station*

NomStation	Libellé	Prix
Venusa	Voile	150
Venusa	Plongée	120
Farniente	Plongée	130
Passac	Ski	200
Passac	Piscine	20
Santalba	Kayac	50

La table *Activité*

Requête : afficher le nom d'une station, son tarif, ses activités et leurs prix.

```
SELECT Station.nomStation, Activite.nomStation, tarif, libelle, prix
FROM   Station, Activite
WHERE  Station.nomStation = Activite.nomStation
```

Remarques

- `Activite.nomStation` est une clé étrangère.
- Ici, on différencie les attributs nomStation en indiquant les relations auxquelles ils apprtiennent
- Comme il peut être fastidieux de répéter intégralement le nom d'une table, on peut alors les renommer. La requête précédente devient par exemple :

```
SELECT S.nomStation, A.nomStation, tarif, libelle, prix
FROM   Station as S, Activite as A
WHERE  S.nomStation = A.nomStation
```

2.3.2 Union, intersection et différence

L'expression de ces trois opérations ensemblistes en SQL est très proche de l'algèbre relationnelle. On construit deux requêtes dont les résultats ont même arité (même nombre de colonnes et mêmes types d'attributs), et on les relie par un des mot-clé **UNION**, **INTERSECT** ou **EXCEPT**.

Trois exemples suffiront pour illustrer ces opérations.

Requête 1: Donnez tous les noms de région dans la base.

```
SELECT region FROM Station
UNION
SELECT region FROM Client
```

Requête 2: Donnez les régions où l'on trouve à la fois des clients et des stations.

```
SELECT region FROM Station
INTERSECT
SELECT region FROM Client
```

Requête 3: Quelles sont les régions où l'on trouve des stations mais pas des clients ?

```
SELECT region FROM Station
EXCEPT
SELECT region FROM Client
```

Une autre méthode pour joindre deux tables : JOIN … ON …

Une analyse des besoins des entraîneurs d'athlètes pratiquant le 100 m, le 200 m, le 400 m ou le 800 m a conduit a la définition d'une base de données constituée des trois tables ci-dessous :

- coureurs (id , nom , prenom)
- epreuves (id , nom , date , distance)
- performances (id_coureur, id_epreuve , temps)

Requête SQL renvoyant les noms et prénoms des coureurs, les noms et les dates des épreuves et les temps réalisés pour tous les 100 m de la base.

```
SELECT   c.nom, c.prenom, e.nom, e.date, p.temps   FROM coureurs as c
JOIN   p   ON   p.id_coureur = c.id
JOIN   e   ON   p.id_epreuve = e.id
WHERE   e.distance = 100
```