Bases de données (Data Base : DB)

Introduction

Mise en situation

Chaque jour nous générons des grandes quantités de données. Cela peut être par internet (Achats de produits en ligne, utilisation des réseaux sociaux, etc…) ou par des actions sur des réseaux locaux (Saisies d'écritures comptables, inscriptions à des activités associatives, gestion des stocks, etc…).Pour conserver et exploiter ces données il faut leur donner une structure organisées qui permette d'enregistrer facilement de nouvelles données de les modifier et de dégager celles qui nous intéressent.

Certaines données doivent être saisies par des personnes et doivent être accessibles par d'autres personnes ou des systèmes de gestion de données. Toutes ces saisies et restitutions d'informations doivent se faire simultanément et en conservant la stabilité, l'accessibilité et la confidentialité de la base de données avec une rapidité suffisante.

Tout cela est assuré par des logiciels performants appelés Système de Gestion de Bases de Données : SGBD. Pour les plus connus on a Access, Oracle, DB2, SQL Server, MySQL ou PostgreSQL.

Définition – Structure - Fonctionnement

La création, l'utilisation, la gestion et la maintenance de ces bases de données (DB) nécessite de multiples compétences qui font appel à plusieurs métiers de l'informatique. Il faut pour cela :

- ☞ Identifier le cahier des charges des utilisateurs
- ☞ Créer la structure de cette base de données
- ☞ Créer les fichiers de données
- ☞ Mettre en place les machines et réseaux pour les échanges d'informations
- ☞ Créer les interfaces d'alimentation en données et de restitution des données.
- ☞ Alimenter en données ces bases de données.
- ☞ Extraire des données en interrogeant les bases de données par des requêtes SQL.

Structure des bases de données

Les Bases de Données relationnelles sont constituées d'un ensemble de Tables en relation les unes avec les autres. Chaque table est en relation avec au moins une autre table de la DB, mais pas toutes.

Ces tables sont des tableaux à deux entrées.

Alimentation de la base de données

Cet ensemble de tables qui est le cœur de la DB doit être alimenté.

Pour cela il faut concevoir des formulaires. Ces interfaces hommes machines sont des logiciels installés sur les machines d'un réseau local pour les petites DB ou sont des serveurs PHP qui récupèrent ces données via des formulaires HTML pour des DB alimentées en ligne sur le Web.

Dans certains cas ces DB peuvent aussi être alimentées automatiquement par des logiciels qui alimentent la base grâce à des capteurs qui mesurent des grandeurs physiques.

Interrogation des bases de données

Ensuite l'interrogation de ces bases de données se fait par des requêtes SQL qui en suivant une syntaxe propre au DB retournera d'autre tableaux, listes, ou résultats de calculs sur les données stockées. Ces interrogations de la DB peuvent se faire soit via des formulaires (HTML pour les formulaires internet). Soit manuellement en écrivant les requêtes SQL.

Le programme de CPGE n'aborde qu'une très infime partie des compétences nécessaires. Il se limite à l'interrogation des bases de données relationnelles par l'écriture de requêtes SQL.

Ces requêtes SQL et leur syntaxe sont le seul point au programme de CPGE.

Confidentialité, sécurité et stabillité des bases de données

Chacun des accès à la DB (alimentation ou récupération des données) est soumis à des règles autorisant ou non les utilisateurs qui ont potentiellement accès à cette DB.

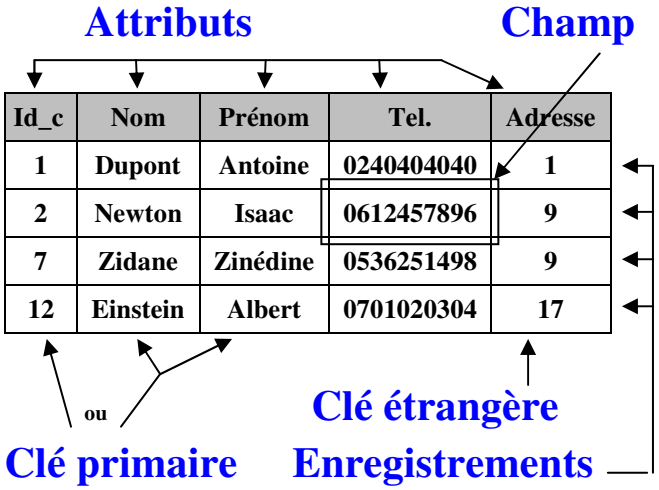
Le stockage des données doit être sécurisé et doit si nécessaire faire l'objet de sauvegardes régulières qui pourront être récupérées pour assurer la continuité du service qu'elles assurent.

Enfin si les données contiennent des données personnelles elles doivent respecter le RGPD : Règlement Général sur la Protection des Données.

Tables : Vocabulaire - Définitions

- ☞ Chaque colonne est un attribut de la table.
- ☞ Chaque ligne est un enregistrement de la table.
- ☞ Chaque case d'une table à l'intersection d'un attribut et d'un enregistrement est un champ.
- ☞ Tous les champs d'un même attribut sont du même type.

  Exemples : Entier, Flottant, Caractère, Chaine de caractère, Booléen, Date, Heure, etc…



Les lignes des enregistrements doivent absolument être distinctes. Cette différence entre les enregistrements se fait soit sur un attribut unique soit sur un N-uplets d'attributs.

Cet attribut ou N-uplet d'attributs, unique, est la clé primaire de la table.

Le principe des bases de données relationnelles est que les tables sont en relations les unes avec les autres. Cette mise en relation se fait par une valeur dans les champs d'un même attribut d'une table qui font appel à un enregistrement dans une autre table.

Cet attribut est une clé étrangère de la table.

Exemple : Dans une DB d'un site de vente en ligne. Un attribut de la table des commandes est une clé étrangère (l'attribut « client ») qui fait appel aux coordonnées (« Nom », « Prénom », etc…) d'un unique client par l'attribut correspondant à la clé primaire de la table de clients (« Id_c »)

Exemple de Base de Données Relationnelle : « DB - Centre de formation »

Un centre de formation professionnelle accueille des stagiaires pour divers stages. Pour chacun de ces stages on a un responsable et un certain nombre de formateurs qui interviennent. Le responsable est un formateur mais qui n'intervient pas forcément dans le stage dont il est responsable.

Chaque intervention de formateur, il y en a plusieurs pour chaque stage, intervient dans une salle à une date et sur un créneau horaire défini (2 le matin : 9h et 10h30 et deux l'après midi 14h et 15h30). Chaque formateur peut intervenir une ou plusieurs fois sur un même stage.

Pour chaque stage le responsable du stage doit définir sa désignation, le nombre maximum de stagiaire, les dates de début et de fin du stage. Choisir pour tous les créneaux du stage les formateurs et la salle dans laquelle intervient le formateur. Chaque salle a ses propres caractéristiques (Bâtiment, Etage, Capacité d'accueil en nombre de stagiaires, Nombre de postes informatiques disponibles, Disponibilité ou non d'un vidéo projecteur et/ou d'une imprimante). Les stagiaires s'inscrivent en ligne sur les stages (pour suivre toutes les interventions).

La DB doit permettre la gestion des stages, des salles, des formateurs et des stagiaires. Pour cela sa structure comporte 7 tables dont on donne ci-dessous les deux premiers enregistrements :

Table des stages **Stages**

Id	Designation	Responsable	Date_deb	Date_fin	Capacite
1	Comptabilité analytique	4	2022-10-10	2022-10-14	12
2	Marketing international	7	2022-10-24	2022-10-28	24

Table des salles **Salles**

Id	Batiment	Etage	Capacite	Nbr_Info	Video_proj	Imprimante
A201	A	2	12	12	1	1
B103	B	1	24	0	1	0

Table des stagiaires **Etudiants**

Id	Nom	Prenom	Adres_mail	Tel
4	Marquaille	Jean-Luc	jl.marq@domaine1.fr	0605040302
12	James	Félicie	f.james@domaine1.fr	0689875456

Table des formateurs **Formateurs**

Id	Nom	Prenom	Adre_mail	Tel
4	Coicaud	Frédéric	f.coicaud@domaine1.fr	0614253669
7	Nerrière	Stéphanie	s.nerriere@domaine3.fr	0696857441

Table d'inscription des formateurs **Ins_form**

Id	Stage	Formateur
4	1	4
7	2	7

Table d'inscription des stagiaires **Ins_etud**

Id	Stage	Stagiaire
1	1	4
5	1	12

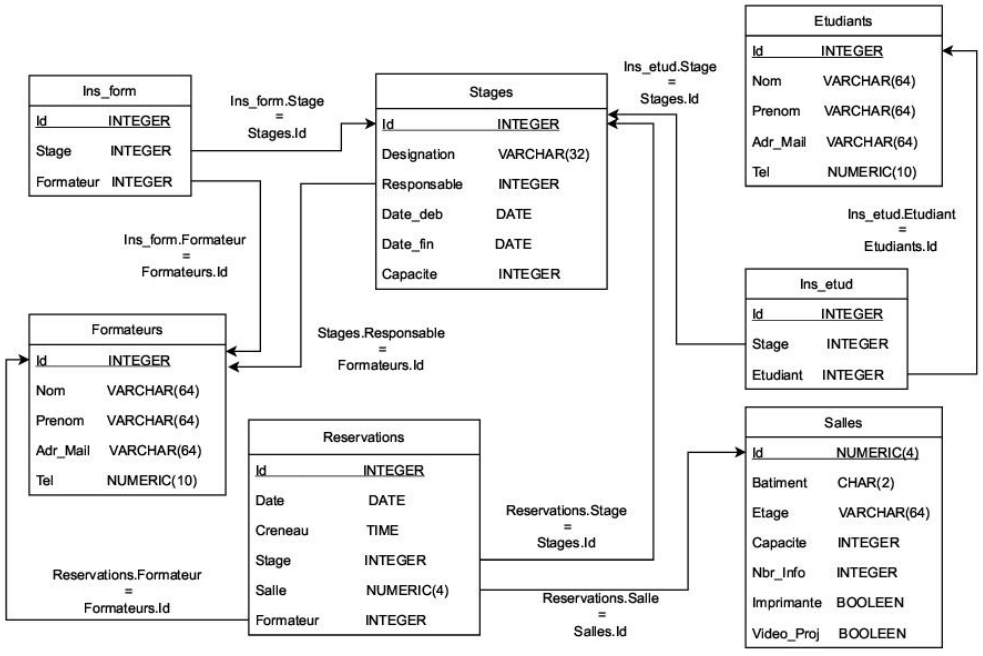
Table des réservations Salles - Créneaux – Formateurs **Reservations**

Id	Date	Creneau	Stage	Salle	Formateur
24	2022-10-12	09:00:00	1	1201	4
128	2022-10-27	15:30:00	2	2103	7

Remarque : On peut voir suivant les champs plusieurs types, différents de ceux de la programmation Python : « Id_sta » (INTEGER) - « Tel » (NUMERIC(10)) - . On voit également des types: Date (DATE) - Creneau (TIME) - « Batiment » (CHAR(2)) – « Nom » (VARCHAR(64)).

La structure d'une DB est souvent présentée sous la forme d'un schéma relationnel comme ci-dessous

Schéma relationnel de la base de donnée « DB – Centre de formation »



Requêtes SQL

Définition d'une requête

Une requête SQL permet de retourner un ou plusieurs attributs d'une table. Voir un ou plusieurs calculs fait à partir des champs d'attributs d'une table. La sélection des attributs se fait avec le mot clef « SELECT » et le choix de la table dans laquelle on fait la sélection se fait avec le mot clef « FROM ».

Donc toute requête SQL comporte au moins un « SELECT » et un « FROM ».

Remarques : Les requêtes écrites ci-dessous utilisent l'exemple de DB ci-dessus.

Requêtes simples

Sélection de tous les champs d'un ou plusieurs attributs

Sélection de tous les champs des attributs « Nom » et « Prenom » dans la table « Formateurs ».

SELECT *Nom,Prenom* FROM *Formateurs*

Cela retourne un tableau avec les étiquettes des attributs qui seront « Nom » et « Prenom ». On peut si on le souhaite renommer ces attributs (Changer les étiquettes) avec le mot clef « AS » :

SELECT *Nom* AS *'Nom de famille', Prenom* AS *Prénom*
FROM *Formateurs*

Sélection de certains champs d'un ou plusieurs attributs avec une ou plusieurs conditions

Sélection des champs des attributs « Batiment », « Etage » et « Id » dans la table « Salles » mais uniquement pour les salles ayant une capacité de 12 stagiaires : mot clé : « WHERE ».

SELECT *Batiment,Etage,Id* FROM *Salles* WHERE *Capacite=12*

Pour le complémentaire on peut utiliser le mot clé « NOT ». Exemple pour la même requête

SELECT *Batiment,Etage,Id* FROM *Salles* WHERE NOT *Capacite<>12*

Types de conditions : Egalité : **=** Différence : **<>** ⚠ différent de Python : « == » et « != »

Supérieur à (strictement) : **>** Inférieur à (strictement) : **<**

Supérieur ou égal à : **>=** Inférieur ou égal à : **<=**

La condition peut aussi être l'appartenance (ou non) à une liste : **IN** ou : **NOT IN**

Liste des noms, prénoms et numéros de téléphone des formateurs qui ont un homonyme dans la liste des stagiaires. La liste est définie à l'aide d'une sous requête écrite entre parenthèses. Pour que ce soit bien le doublet dont on teste la présence dans la liste, il est mis entre parenthèses.

SELECT *Nom,Prenom,Tel* FROM *Formateurs*
WHERE (*Nom,Prenom*) IN (SELECT *Nom,prenom* FROM *Etudiants*)

Pour plusieurs conditions, on utilise les mots clé « AND » et « OR » suivant l'algèbre booléenne.

Exemple : Liste des salles (« Batiment », « Etage » et « N° de salle ») pouvant accueillir au moins 20 stagiaires et qui de plus ont une imprimante ou un vidéo-projecteur.

SELECT *Batiment,Etage,Id* AS *'N° de salle'* FROM *Salles*
WHERE *Capacite>=20* AND (*Imprimante* OR *Video_proj*)

⚠ **L'opérateur « AND » est prioritaire sur l'opérateur « OR » (D'où ici les parenthèses)**

Ordonnancement des résultat

Sélection des champs des attributs « Etage », « Id » et « capacite » dans la table « Salles » mais uniquement pour des salles du « batiment » 'A' . les résultats étant ordonnés dans l'ordre croissant des capacités : mot clé : « ORDER BY ».

SELECT *Etage,Id,Capacite* FROM *Salles*
WHERE *Batiment='A'* ORDER BY *Capacite*

Ordre décroissant : Ajouter après « ORDER BY Attribut » le mot clé : **« DESC »**

Limiter en nombre les résultats : Pour seulement n résultats, ajouter **« LIMIT n »**

Ne pas commencer au 1ier (ou dernier) : Pour commencer au k^{ième} ajouter **« OFFSET k »**

Exemple : Sélection des champs des attributs « Etage », « Id » et « Capacite » dans la table « Salles » mais pour des salles du « batiment » 'A'. On ne présentera que les 2, 3, 4 et 5^{ième} salles par ordre décroissant de capacité.

SELECT *Etage,Id,Capacite* FROM *Salles* WHERE *Batiment='A'*
ORDER BY *Capacite,Etage* DESC LIMIT 4 OFFSET 1

Requêtes d'agrégation

Sélection sans redondances

Lister, dans l'ordre croissant de capacité et du nombre de postes informatique, les différentes configurations de salles dont dispose le centre de formation. La configuration d'une salle étant définie par sa capacité et son nombre de postes d'informatique. On utilise pour cela le mot clé « DISTINCT ».

SELECT DISTINCT *Capacite,Nbr_info* FROM *Salles*
ORDER BY *Capacite,Nbr_info*

Comptage d'un nombre d'éléments

Comptage du nombre de salles.

SELECT COUNT(*) AS *'Nombre de salles'* FROM *Salles*

Cette requête ne renvoie pas une liste mais simplement un nombre entier correspondant au nombre de lignes de la requête : « SELECT * FROM Salles »

Listing d'un comptage par regroupement

Comptage du nombre de salles par bâtiment. On utilise alors le mot clé « GROUP BY » :

SELECT *Batiment*,COUNT(*) AS *'Nombre de salles'*
FROM *Salles* GROUP BY *Batiment*

Cette requête renvoie une liste où chaque ligne est un comptage.

Comptage d'un nombre d'attributs différents

Comptage du nombre de bâtiments. On utilise alors le mot clé « DISTINCT » à l'intérieur de la commande de comptage « COUNT » :

SELECT COUNT(DISTINCT *Batiment*)
AS *'Nombre de bâtiments'* FROM *Salles*

Comptage d'un nombre d'attributs différents par regroupement sur un autre attribut

Comptage du nombre d'étages par bâtiments. On utilise alors les mots clé « DISTINCT » et « COUNT » pour le comptage d'attributs différents et le mot clé « GROUP BY » pour regrouper ce comptage par bâtiments différents.

```
SELECT Batiment,COUNT(DISTINCT(Etage))
  AS 'Nombre d'étages' FROM Salles
  GROUP BY Batiment
```

Opérations sur les attributs

Il est possible de créer des lignes qui n'existent pas dans la table en effectuant des opérations sur les attributs de la table.

Listing des différentes salles (bâtiment, étage et Numéro de salle) et de leur taux d'équipement en postes informatiques (en %) classées par bâtiments et étages :

```
SELECT Batiment,Etage,Id AS Numéro, 100*Nbr_info/Capacite
  AS 'Taux d'équipement informatique en %' FROM Salles
  ORDER BY Batiment,Etage
```

Remarque : Les attributs « Nbr_info » et « Capacite » sont du type « INTEGER » (entier) donc le résultat du calcul retourné est un entier qui est l'arrondi à l'entier inférieur. Donc si on ne met pas le « 100* » les résultats seront soit « 0 » (pour $0 \leq$ taux $<$ 100%) soit « 1 » (taux $\geq$ 100%)

Il convient donc pour les opérations d'être vigilant sur le type des attributs :

« INTEGER » pour entier « DOUBLE » pour un flottant « DECIMAL(n,m) » pour un nombre décimal de n chiffres et m décimales.

Autres mots clés pour des opérations :
- **CEIL(x)** pour l'arrondi de x à l'entier supérieur
- **ABS(x)** pour la valeur absolue de x
- **POWER(x,n)** pour le résultat x^n

Certains mots clés s'appliquent à des chaines de caractère :

« CHAR_LENGTH(ch) » renvoie le nombre de caractère de la chaine « ch »

« Ch1||ch2 » renvoie la concaténation des chaines de caractère « ch1 » et « ch2 »

Opérations sur des regroupements

Certaine opérations ne s'appliquent que sur un groupement de plusieurs enregistrements. On a la possibilité de sélectionner dans un regroupement :

- ☞ la valeur maximale mot clé : **« MAX »**
- ☞ la valeur minimale mot clé : **« MIN »**
- ☞ la valeur moyenne mot clé : **« AVG »**
- ☞ la somme des valeurs mot clé : **« SUM »**

Listing de la capacité d'accueil en stagiaires de chaque étage ordonné par bâtiment et étage.

```
SELECT Batiment,Etage,SUM(Capacite) FROM Salles
  GROUP BY Batiment,Etage ORDER BY Batiment,Etage
```

Sélection de résultats d'opérations sur les attributs

Il est possible d'effectuer une sélection sur les résultats des opérations par regroupement. Pour cela on utilise le mot clé « HAVING ». Ce mot clé est l'équivalent du mot clé « WHERE » qui ne s'applique lui que sur les attributs.

Listing des étages (bâtiments, étages) dont la plus grande salle peut accueillir au moins 20 stagiaires.

```
SELECT Batiment,Etage FROM Salles
  GROUP BY Batiment,Etage HAVING MAX(Capacite) >= 20
```

Listing des étages (bâtiments et étages) avec une moyenne de capacité d'accueil des stagiaires au moins égale à 20.

```
SELECT Batiment,Etage,AVG(Capacite) AS Moyenne
  FROM Salles  GROUP BY Batiment,Etage HAVING Moyenne>=20
```

Requêtes avec jointure

Produit cartésien

Le produit cartésien de deux tables donne une table dont les attributs sont concaténés et dont les enregistrements sont toutes les combinaisons possibles entre les différents enregistrements des tables.

Ce produit cartésien s'obtient en écrivant les 2 noms des tables séparés par une virgule.

Exemple on a 2 tables : « Etudiants » (2 attributs et 2 enregistrements) une clé primaire « Id_e »

Et : « Copies » (**3 attributs et 3 enregistrements**) une clé primaire « Id_c »

Et une clé étrangère « Etudiant » de la table « Etudiants »

Etudiants	
Id_e	Prenom
1	Sophie
2	Luc

Copies		
Id_c	Etudiant	Note
1	1	15
2	1	18
3	2	16

Le produit cartésien « Etudiant,Copies » donne la table :

Etudiants,Copies				
Id_e	Prenom	Id_c	Etudiant	Note
1	Sophie	1	1	15
1	Sophie	2	1	18
1	Sophie	3	2	16
2	Luc	1	1	15
2	Luc	2	1	18
2	Luc	3	2	16

Remarque sur les dimension du produit cartésien : Pour une table 1 de a1 attributs et e1 enregistrements et une table 2 de a2 attributs et e2 enregistrements le produit cartésien donne une table de (a1+a2) attributs et (e1×e2) enregistrements.

Jointure entre deux tables

En général on fait le produit cartésien de deux tables dont un attribut (clé étrangère) de l'une contient des valeurs correspondant à des valeurs d'un attribut de l'autre (clé primaire).

Dans notre exemple la clé étrangère de la table « Copies » : (« Etudiant ») correspond à la clé primaire de la table « Etudiants » : (« Id_e »). C'est-à-dire que toutes les valeurs des champs de l'attribut « Etudiant » de la table « Copies » sont présents dans l'un au moins des champs de l'attribut « Id_e » de la table « Etudiants ». Chaque note peut donc être attribuée à un étudiant qui a un prénom.

En ne sélectionnant que les enregistrements où la clé étrangère des copies : « Copies.Etudiant »

est égale à la clé primaire des Etudiants : « Etudiants.Id_e »

On fait la jointure entre les tables « Etudiants » et « Copies ».

La requête SQL : **SELECT Etudiants.Prenom,Copies.Note**
FROM Etudiants,Copies
WHERE Etudiants.Id_e=Copies.Etudiant

Permet d'obtenir la liste des notes avec les prénoms de chacune des copies. Ainsi cette requête renvoi la liste (ou table) ci-contre :

Prenom	Note
Sophie	15
Sophie	18
Luc	16

Remarque : le préfixe (nom de la table) devant le nom de l'attribut n'est pas obligatoire mais devient indispensable lorsque les deux tables ont des noms d'attributs identiques.

En général on préfère écrire cette requête avec une autre syntaxe utilisant les mots clés « JOIN » qui fait le produit cartésien et le mot clé « ON » qui fait la sélection permettant la jointure :

```
SELECT E.Prenom,C.Note FROM Etudiants AS E
  JOIN Copies AS C ON E.Id_e=C.Etudiant
```

Exemple de Jointure Simple

Liste des désignations des stages avec les noms et prénoms et adresses mail des Responsables.

```
SELECT S.Designation,F.Nom, F.Prenom, F.Adr_Mail
  FROM Stages AS S JOIN Formateurs AS F
  ON S.Responsable=F.Id
```

Liste des dates et créneaux réservés par la formatrice Nerrière Stéphanie.

```
SELECT R.Date,R.Creneau FROM Reservations AS R
  JOIN Formateurs AS F ON R.Formateur=F.Id
  WHERE (F.Nom,F.Prenom)=('Nerrière','Stéphanie')
```

Exemples de Jointure Multiples

Liste des stagiaires (Nom, Prénom) inscrits au Stage « Comptabilité analytique »

```
SELECT E.Nom,E.Prenom FROM Etudiants AS E
  JOIN Ins_etud AS IE ON E.Id=IE.Etudiant
  JOIN Stages AS S ON IE.Stage=S.Id
  WHERE S.Designation='Comptabilité analytique'
```

Félicie pourra-t-elle utiliser une imprimante le 12 Octobre 2022 ?

```
SELECT Imprimante FROM Etudiants AS E
  JOIN Ins_etud AS IE ON E.Id=IE.Etudiant
  JOIN Reservations AS R ON IE.Stage=R.Stage
  JOIN Salles AS S ON R.salle=S.Id
  WHERE Prenom='Félicie' AND R.Date='2022-10-12'
```

Dans combien de salles différentes, la formatrice Nerrière Stéphanie interviendra-t-elle pour le stage du 24/10/2022 au 28/10/2022.

```
SELECT COUNT(DISTINCT(R.Salle)) FROM Formateurs AS F
  JOIN Reservations AS R ON R.Formateur=F.Id
  JOIN Stages AS S ON R.Stage= S.Id
  WHERE F.Nom='Nerrière' AND F.Prenom='Stéphanie'
  AND S.Date_deb='2022-10-24' AND S.Date_fin='2022-10-28'
```

Requêtes par combinaisons

Pour extraire des attributs de plusieurs tables on peut utiliser des constructions ensemblistes qui permettent de les combiner. Pour cela on a trois mots clé possibles.

- ☞ **« UNION »** qui renvoie la concaténation des enregistrements des 2 listes
- ☞ **« EXCEPT »** qui renvoie les enregistrements de la 1ière liste qui ne sont pas dans la 2nd.
- ☞ **« INTERSECT »** qui renvoie uniquement les enregistrements présents dans les deux listes.

Pour une telle requête on écrit la requête générant la première liste, puis le mot clé « UNION » ou « EXCEPT » ou « INTERSECT » et enfin la requête générant la seconde liste.

Remarque :

Les 2 listes doivent impérativement avoir le même nombre d'attributs.

Exemple : Liste des identifiants des formateurs qui ne sont pas responsable d'un stage.

```
SELECT Id FROM Formateurs
EXCEPT
SELECT Responsable FROM Stages
```

Exemple de combinaison simple

Liste des Noms et prénoms des formateurs qui sont (ou ont été) aussi des stagiaires. On suppose qu'il n'y a pas d'homonynes.

```
SELECT Nom,Prenom FROM Etudiants
INTERSECT
SELECT Nom,Prenom FROM Formateurs
```

Exemple de combinaison avec conditions

Listes des Numéros de Salles qui ont une capacité d'au moins 20 stagiaires et qui sont libres toute la semaine du 24 au 28 Octobre 2022.

```
SELECT Id FROM Salles WHERE Capacite>=20
EXCEPT
SELECT Salle FROM Reservations
  WHERE Date>='2022-10-24' AND Date<='2022-10-28'
```

Exemple de combinaison avec jointures

Listes des noms et prénoms des personnes ayant fréquenté le centre formation la semaine du 24 au 28 Octobre 2022.

```
SELECT Nom,Prenom FROM Formateurs AS F
  JOIN Reservations AS R ON F.Id=R.Formateur
  WHERE Date>='2022-10-24' AND Date<='2022-10-28'
UNION
SELECT Nom,Prenom FROM Etudiants AS E
  JOIN Ins_etud AS IE ON IE.Etudiant=E.Id
  JOIN Reservations AS R ON IE.Stage=R.Stage
  WHERE Date>='2022-10-24' AND Date<='2022-10-28'
```