Bases de données - SQL

Avant-propos

L'administration, les banques, les assurances, les secteurs de la finance utilisent des **bases de données**, systèmes d'informations qui stockent dans des fichiers les données nombreuses qui leur sont nécessaires.

Une base de données relationnelle permet d'organiser, de stocker, de mettre à jour et d'interroger des données structurées volumineuses utilisées simultanément par différents programmes ou différents utilisateurs. Un logiciel, le **système de gestion de bases de données** (SGBD), est utilisé pour la gestion (lecture, écriture, cohérence, actualisation...) des fichiers dans lesquels sont stockées les données.

L'accès aux données d'une base de données relationnelle s'effectue en utilisant un langage informatique qui permet de sélectionner des données spécifiées par des formules de logique, appelées **requêtes** d'interrogation et de mise à jour.
L'objectif est de présenter une description applicative des bases de données en langage de requêtes **SQL** (*Structured Query Language*).

I. Vocabulaire des bases de données

I.1. Tables de relation

L'objet de base que nous manipulerons tout au long de ce cours est appelé *relation*.

Définition

- Une **relation** est un tableau de données (penser à un fichier Excel). Chaque ligne représente un objet et tous les objets sont de même nature.
- Une **base de donnée** (parfois abrégée en BDD) est un ensemble de relations.

Remarque

Même si le tableur Excel est la bonne image à garder en tête, nous verrons plus loin que, dans la pratique, les données sont stockées dans un fichier texte `.csv`, plus léger et moins riche graphiquement que le tableur.

Exemple
Construisons la table de relation des étudiants de cette prépa, en indiquant comme caractéristiques les dates de naissance, la ville de provenance et le sexe et la filière.

Les étudiants de Mme Duroux				
identifiant	**date de naissance**	**ville de provenance**	**F ou G**	**PCSI ou TPC2**
1	16/02/2004	Vincennes	F	TPC2
2	31/10/2006	Esbly	G	PCSI
3	26/10/2005	Nogent-sur-Marne	F	TPC2
4	15/09/2005	Créteil	G	PCSI
5	25/11/2003	Saint-Maur-des-fossés	G	TPC2
6	30/08/2006	Courbevoie	G	PCSI
7	03/06/2005	Saint-Maur-des-fossés	F	TPC2
8	30/08/2006	Paris $5^{\text{ème}}$	F	PCSI
[...]	[...]	[...]	[...]	[...]

Remarque

- Dans une relation, l'ordre des lignes n'a pas d'importance.
- Il est très fréquent de mélanger les termes définis par le modèle relationnel et les termes propres au vocabulaire des bases de données relationnelles.

Définition

- Une *relation* est aussi appelée une **table**.
- Les termes *ligne*, *n-uplet*, *enregistrement* ou *vecteur* sont tous synonymes.
- De même les colonnes sont souvent appelées des **attributs**.
- Le **schéma** est l'ensemble des attributs d'une relation.
- Le *domaine* désigne l'ensemble des valeurs que peuvent prendre un attribut.

I.2. Les clés

Reprenons l'exemple précédent des étudiants de notre prépa. Il est tout à fait possible que deux étudiants aient exactement la même date de naissance et prennent les mêmes options. Dans une table, il est très important d'identifier de **manière unique** chaque ligne. C'est pour résoudre ce problème d'identification qu'interviennent les clés.

Définition
Une **clé** est un groupe minimum d'attributs caractérisant chaque n-uplet de manière unique.

Exemple
Dans notre exemple précédent, l'attribut **identifiant** est une clé (« l'étudiant qui a pour identifiant 2 » permet de désigner un étudiant sans aucune ambiguïté). En revanche, la **date de naissance** n'est pas une clé car il y a deux étudiants qui sont nés le 30/08/2006. Le groupe d'attributs {**date de naissance**, **F/G**} est une clé.

Exercice 1
On considère la relation **conserves** décrite par la table ci-dessous.

code barre	produit	marque	prix unitaire	quantité	peremption
3083680020763	Maïs	Budget+	1,19	12	2026
3017800207901	Maïs	GourmetBio	2,69	9	2025
3038359008504	Tomates	PizzaLovers	2,99	6	2025
303087439871	Haricots Verts	Budget+	1,89	10	2025
3093439818758	Pêches	GourmetBio	2,99	6	2030
3023249825326	Pois chiches	Budget+	0,99	4	2026

1. Le groupe d'attributs {**code barre**} est-il une clé pour notre relation ?
2. Le groupe d'attributs {**code barre**, **produit**} est-il une clé pour notre relation ?
3. Le groupe d'attributs {**produit**, **marque**, **prix**} est-il une clé pour notre relation ?
4. Le groupe d'attributs {**produit**} est-il une clé pour notre relation ?
5. Le groupe d'attributs {**marque**, **prix**} est-il une clé pour notre relation ?

Remarque
Plusieurs attributs peuvent être des clés. Or, dans une base de données, nous voulons nous mettre d'accord une fois pour toute sur l'une d'entre elles.

Définition
- Dans une relation, les groupes d'attributs qui constituent des clés sont appelés des *clés candidates*.
- Dans la pratique, il sera indispensable d'en choisir une et nous l'appellerons la **clé primaire**.
  En anglais, clé primaire se dit *Primary Key* et s'abrège en **PK**.

Remarque
Dans le logiciel de gestion de base de données que nous utiliserons, nous verrons que lors de la création d'une table de données dans un fichier .csv, il est indispensable de désigner au préalable une clé primaire.

Méthodo : Comment choisir la clé primaire ?
Le choix d'une clé primaire pourrait très bien être arbitraire (parmi les clés candidates). Cependant, on peut prendre en compte certains critères. En effet, une clé primaire est généralement choisie de façon à ce qu'elle soit simple, c'est-à-dire qu'elle ne contienne le moins d'attributs possibles.
De plus, on préfère généralement les attributs « basiques » (par exemple des entiers ou des chaînes de caractères courtes).

Remarque
Parfois aucune des clés candidates ne contient un nombre raisonnable d'attributs et ne semble « simple ». On peut alors fabriquer une clé simple.

Définition
Une **clé artificielle** est un attribut que l'on ajoute à la relation. Cet attribut n'a pas de réelle signification dans le domaine que l'on modélise et sa seule fonction est d'identifier de manière unique les n-uplets de la relation.

Exemple
Reprenons encore une fois la liste des étudiants en prépa ici. Le premier attribut **identifiant** est une clé artificielle.

Remarque
Pour des questions de performance (en termes de temps de calcul), les clés non artificielles ne sont souvent pas optimisées lorsqu'il est demandé au SGBDD de retrouver une ligne dans une table.
Un bon programmeur prend donc l'habitude d'utiliser comme clés primaires des clés artificielles. Il y a (au moins) deux raisons à ça.

Une base de données est en fait bien souvent constituée de plusieurs tables ; et ces tables ne sont pas indépendantes.

Dans notre exemple fil rouge de la base de données des étudiants de notre prépa, nous pouvons nous intéresser à la ville de provenance des étudiants et renseigner dans une relation séparée certaines caractéristiques des villes qui y décrivent l'ambiance de travail. Même si cette seconde table est d'un intérêt indépendant, ces deux relations sont liées car chaque étudiant provient d'une des villes décrites dans le seconde table.

Étudier dans les villes autour de d'Arsonval			
ville (PK)	**temps de trajet**	**facilité de travail**	**difficulté de travail**
Saint-Maur-des-fossés	20	Proche	
Créteil	40	Calme	Temps de transport
Vincennes	45	Joli cadre de vie	Problèmes de RER fréquents
Esbly	90	Facile de faire du sport	Temps de transport
[...]	[...]	[...]	[...]

Ainsi, si on veut connaître le temps de trajet quotidien d'un étudiant, on doit déjà connaître sa ville de provenance, puis regarder dans la table des villes, quel est le temps de trajet pour venir au lycée d'Arsonval.

Remarque
Retrouver une ligne dans une table, c'est justement le rôle des clés !

Définition
Un attribut d'une table qui est une clé pour une autre table de la base de données s'appelle une **clé étrangère**.
En anglais, clé étrangère se dit *Foreign Key* et s'abrège en **FK**.

Exemple
Dans notre liste d'étudiant, l'attribut **ville de provenance** n'est pas une clé (plusieurs étudiants peuvent venir de la même ville). Cependant, cet attribut fait quand même référence à la clé primaire d'une autre table de la base de données, celle qui décrit les conditions de travail dans les villes voisines du lycée d'Arsonval. La ville de provenance est donc une clé étrangère dans la relation des étudiants.

Méthodo : Comment choisir les attributs à regrouper dans une relation ?
C'est en général une mauvaise idée de regrouper tous les attributs dans une même table. En effet :

- × Si le temps de trajet pour aller au lycée d'Arsonval à une ville voisine change, il faudrait modifier l'information dans toutes les lignes correspondantes de la table (pour tous les étudiants qui habitent dans cette ville). On risque d'en oublier et de perdre la cohérence de la base de données. On cherche à éviter la **redondance des données**.
- × Si on souhaite ajouter les caractéristiques d'une nouvelle ville proche de l'établissement, mais qu'aucun étudiant n'habite dans cette ville, on ne peut pas.

Pour éviter la redondance, on peut suivre la règle suivante :

Si un attribut A dépend uniquement d'un groupe d'attributs G et que G est minimal (c'est-à-dire que si on enlève un attribut à G, alors A ne dépend plus uniquement que des attributs restants), alors il est possible de créer une nouvelle table qui ne contient que les attributs A et G. G sera d'ailleurs une clé candidate pour la nouvelle table et tout autre attribut B qui ne dépend que de G peut être déplacé aussi dans la nouvelle relation.

Exemple
Dans notre exemple récurrent, nous avons appliqué ce principe dans la table des villes voisines : le temps de trajet, les facilités et difficultés à travailler ne dépendent que de la ville et nous les avions placé dans la nouvelle table.

Définition
Le processus qui consiste à créer de nouvelles tables pour éviter la redondance s'appelle la **normalisation**.

Dans la suite de ce cours, nous nous intéresserons uniquement à la *manipulation* des données, et nous n'entrerons pas dans les détails de la *création* de données et de leur *stockage*. Les stratégies de normalisation sont donc hors programme en TPC.

I.3. Les tables d'association

Considérons la situation suivante. On dispose de la relation des **écoles** ci-dessous.

Écoles d'ingénieur		
Identifiant (PK)	**Nom de l'école**	**Taux d'admis année $n-1$**
1	Chimie ParisTech	25%
2	ENSI	18%
3	CPE	7%
4	ECPM	6%
5	ENSCL	6%
$[\cdots]$	$[\cdots]$	$[\cdots]$

(Observons que nous avons pris pour **PK** une clé articifielle, le taux d'admis n'étant pas une bonne **PK** car il peut y avoir des ex-aequo.)

C'est maintenant la fin de l'année dans notre prépa et nous voulons ajouter à notre base de données les informations sur les différentes admissibilités de nos étudiants dans les différentes écoles. Nous pouvons donc ajouter une colonne dans notre table d'étudiants avec l'attribut **admissible à** qui est une clé étrangère vers la table **écoles**.

Et si un étudiant est admissible dans 2 écoles, comment fait-on ?

On pourrait mettre deux colonnes avec les attributs **admissible à 1** et **admissible à 2** qui seraient toutes les deux des clés étrangères vers la table **écoles**. Mais ce raisonnement n'est pas très bon : on ne peut pas savoir à l'avance dans combien d'écoles un étudiant va être admissible.

La stratégie inverse ne fonctionne pas non plus. On pourrait envisager de mettre dans la table **écoles** un attribut **étudiant admissible** qui serait une clé étrangère vers la table **étudiants**. Mais cela voudrait dire qu'une école ne peut sélectionner qu'un seul étudiant admissible !

Ce problème motive la définition suivante.

Définition
Une **table d'association** est une relation qui contient comme attributs au moins deux clés étrangères vers d'autres relations de la base de données.

Exemple

Construisons donc la table d'association des étudiants et des écoles dans lesquelles ils sont admissibles.

Nous pouvons renseigner efficacement les différentes admissibilités des étudiants de la prépa, sans nous limiter, ni en nombre d'écoles par étudiant, ni en nombre d'étudiants admissibles par école.

Admissibilités	
Étudiant	**École**
1	1
1	2
1	4
2	4
2	5
...	...

Chacun des attributs de cette table est FK mais la PK est constituée des deux attributs {**Étudiant, École**}.

Remarque
Rien ne nous empêche d'ajouter des attributs dans notre table d'association (par exemple la date de passage pour les oraux dans la situation précédente) : tous les attributs de la table d'association ne sont donc pas nécessairement des clés étrangères pour une relation de la base de données.
Cela pose d'ailleurs la question du choix de la clé primaire dans une table d'association. Mais c'est une autre histoire.

II. Algèbre relationnelle

Dans la partie précédente, nous avons vu comment représenter des données. Nous passons maintenant à la partie manipulation. Comme convenu au début de ce cours, nous décrivons tout d'abord de manière abstraite les différentes opérations que l'on peut appliquer à une base de données, puis nous verrons comment cela se concrétise dans la section suivante.

II.1. Projection et restriction

Définition
La **projection** consiste à sélectionner les attributs que l'on souhaite et éliminer les autres.

Exemple
Si l'on s'intéresse uniquement aux dates de naissance des étudiants de la prépa, on peut faire une projection pour ne conserver que l'attribut **date de naissance** et on obtient la table suivante.

identifiant	Date de naissance
1	16/02/2004
2	31/10/2006
3	26/10/2005
[...]	[...]

Nous voudrions maintenant faire la même opération mais sur les lignes, c'est-à-dire ne conserver que certaines des lignes de notre relation. C'est légèrement plus compliqué car les colonnes de notre table portent un nom mais pas les lignes. Les lignes d'une base de données sont amenées à varier car les informations contenues dans la table varient au cours du temps.
Comme les lignes à garder ou à enlever sont dynamiques, il nous faut donc une condition qui nous permette de restreindre les lignes.

Définition
Une **restriction** dans une table de données est une condition binaire (de type vrai ou faux) qui porte sur un ou plusieurs des attributs de la relation.

Exemple

1. Le résultat de la restriction de la table des étudiants sous la condition date de naissance $<$ 01/10/2005 produit la table suivante.

Les étudiants de Mme Duroux				
identifiant	**date de naissance**	**Ville de provenance**	**F ou G**	**PCSI ou TPC2**
1	16/02/2004	Vincennes	F	TPC2
4	15/09/2005	Créteil	G	PCSI
5	25/11/2003	Saint-Maur-des-fossés	G	TPC2
7	03/06/2005	Saint-Maur-des-fossés	F	TPC2

2. La restriction de la table des étudiants avec la condition F ou G = 'G' produit la table suivante.

Les étudiants de Dumas				
identifiant	**date de naissance**	**Ville de provenance**	**F ou G**	**PCSI ou TPC2**
1	16/02/2004	Vincennes	F	TPC2
3	26/10/2005	Nogent-sur-Marne	F	TPC2
7	03/06/2005	Saint-Maur-des-fossés	F	TPC2
8	30/08/2006	Paris $5^{\text{ème}}$	F	PCSI

Exercice 2
Quelle table obtient-on à partir de la table **conserves** de l'Exercice 1 après restriction sous la condition peremption $<$ 2026 ?

II.2. Union, différence, intersection

Définition
L'**union** de deux relations R_1 et R_2 de même schéma est une nouvelle relation, également de même schéma, qui contient l'ensemble des lignes de R_1 et R_2.

Attention, dans la pratique avec le langage SQL, l'union de deux tables qui contiennent une ligne en commun produira une table avec la même ligne répétée.
Mais ce n'est pas très grave, il existe une commande pour éliminer les doublons d'une table.

Exemple
Si on a dans deux relations différentes des renseignements sur les étudiants de la prépa, par exemple :

identifiant	date de naissance	ville de provenance	F ou G	PCSI ou TPC2
1	16/02/2004	Vincennes	F	TPC2
2	31/10/2006	Esbly	G	PCSI
3	26/10/2005	Nogent-sur-Marne	F	TPC2
4	15/09/2005	Créteil	G	PCSI

et

identifiant	date de naissance	ville de provenance	F ou G	PCSI ou TPC2
16	27/08/2004	Lyon	G	TPC2
25	01/10/2005	Belfort	G	TPC2
32	25/06/2004	Varsovie	F	TPC2

alors la réunion des deux relations produit la table :

identifiant	date de naissance	ville de provenance	F ou G	PCSI ou TPC2
1	16/02/2004	Vincennes	F	TPC2
2	31/10/2006	Esbly	G	PCSI
3	26/10/2005	Nogent-sur-Marne	F	TPC2
4	15/09/2005	Créteil	G	PCSI
16	27/08/2004	Lyon	G	TPC2
25	01/10/2005	Belfort	G	TPC2
32	25/06/2004	Varsovie	F	TPC2

Définition
La **différence** de deux relations de même schéma R_1 et R_2 donne une relation R_3 de même schéma qui contient toutes les lignes de R_1 qui ne sont pas dans R_2.

Attention, si l'opération de réunion est commutative, ce n'est pas le cas de l'opération de différence : l'ordre compte !

Définition
L'**intersection** de deux relations de même schéma R_1 et R_2 donne une relation R_3 de même schéma qui contient toutes les lignes qui sont à la fois dans R_1 et dans R_2.

II.3. Produit cartésien

Au conseil de classe, chaque professeur doit donner une appréciation sur chaque étudiant de la prépa. Pour cela, nous disposons d'une relation qui contient les informations sur les professeurs de la classe.

identifiant prof	Nom du professeur
prof1	M. Neveu
prof2	M. Blanchard
prof3	M. Acquier
prof4	Mme Cerf
prof5	M. Espitallier
prof6	Mme Duroux

et (une projection et restriction du) tableau des étudiants

identifiant étudiant
1
2
3

Pour pouvoir indiquer toutes les appréciations dans une relation, nous devons former le *produit cartésien* des deux tables des enseignants et des étudiants.

Définition
Le **produit cartésien** de deux relations R_1 et R_2 est une table qui contient toutes les combinaisons possibles des lignes de R_1 et des lignes de R_2 et qui contient les colonnes de R_1 ainsi que les colonnes de R_2.

Exemple
Dans notre exemple, le produit cartésien des relations professeurs et étudiants renvoie la table suivante :

identifiant prof	Nom du professeur	identifiant étudiant
prof1	M. Neveu	1
prof1	M. Neveu	2
prof1	M. Neveu	3
prof2	M. Blanchard	1
prof2	M. Blanchard	2
prof2	M. Blanchard	3
prof3	M. Acquier	1
prof3	M. Acquier	2
prof3	M. Acquier	3
prof4	Mme Cerf	1
prof4	Mme Cerf	2
prof4	Mme Cerf	3
prof5	M. Espitallier	1
prof5	M. Espitallier	2
prof5	M. Espitallier	3
prof6	Mme Duroux	1
prof6	Mme Duroux	2
prof6	Mme Duroux	3

à laquelle il est ensuite facile d'ajouter un attribut **appréciation**.

II.4. Jointure

La jointure est le concept fondamental de l'algèbre relationnelle. Jusqu'à présent, nous avons vu que les clés étrangères servent à lier des relations. Mais nous ne savons pas encore comment exploiter ces liaisons. C'est justement le but de l'opération de jointure.
Si nous voulons connaître le temps de trajet pour venir au lycée d'Arsonval d'un étudiant de la prépa, nous devons regarder d'abord dans la table des étudiants sa ville de provenance, puis aller chercher le temps de trajet dans la table des villes voisines. La jointure sert à coller les deux tables pour faire apparaître l'information sur une seule table.

Définition

- On considère deux relations R_1 et R_2 et on suppose que dans la relation R_1, le groupe d'attributs G est une clé externe pour la relation R_2. La **jointure** de R_1 et R_2 selon G est la table qui contient le même nombre de lignes que R_1 et les attributs de R_1 et de R_2. Chaque ligne est construite en commençant par la ligne de R_1, puis en ajoutant à sa suite la ligne de R_2 caractérisée par la valeur de sa clé dans G.
- Le groupe d'attributs G s'appelle **la condition de jointure**.

Exemple

Reprenons une restriction de notre relation préférée **étudiants**.

identifiant (PK)	date de naissance	ville de provenance (FK)	F ou G	PCSI ou TPC2
1	16/02/2004	Vincennes	F	TPC2
2	31/10/2006	Esbly	G	PCSI
4	15/09/2005	Créteil	G	PCSI
5	25/11/2003	Saint-Maur-des-fossés	G	TPC2
7	03/06/2005	Saint-Maur-des-fossés	F	TPC2

et une relation **villes** (obtenue après une projection d'une relation ci-avant) :

ville (PK)	temps de trajet
Saint-Maur-des-fossés	20
Créteil	40
Vincennes	45
Esbly	90

La jointure de ces deux relations selon la condition ville de provenance = ville donne la table suivante.

etudiant. identifiant	etudiant. date de naissance	etudiant. ville de provenance	etudiant. F ou G	etudiant. PCSI ou TPC2	ville. nom de ville	ville. temps de trajet
1	16/02/2004	Vincennes	F	TPC2	Vincennes	45
2	31/10/2006	Esbly	G	PCSI	Esbly	90
4	15/09/2005	Créteil	G	PCSI	Créteil	40
25/11/2003	Saint-Maur-des-fossés	G	TPC2	Saint-Maur-des-fossés	20	
7	03/06/2005	Saint-Maur-des-fossés	F	TPC2	Saint-Maur-des-fossés	20

Exercice 3

Montrer que la jointure est l'enchaînement d'un produit cartésien et d'une restriction.

II.5. Agrégation

L'agrégation (qui n'est en fait pas une opération de l'algèbre relationnelle) est utilisée lorsqu'on veut faire un calcul qui porte sur plusieurs ligne d'une table. Elle sert par exemple à répondre à la question : *Quelle est le temps de trajet moyen des étudiants pour chaque filière ?*
Il s'agit donc de calculer une valeur pour chaque choix de filière (PCSI ou TPC2).

L'agrégation est donc une opération en deux étapes : une étape de partitionnement et une étape de calcul où on applique une fonction à chacun des blocs de la partition.

Définition

- On considère une table et un groupe d'attributs.
  Un **agrégat** est une partition des lignes d'une table selon les différentes valeurs que peuvent prendre les attributs.
- Dans cette situation, on dit que les attributs sont des **attributs de partitionnement**.

Définition
Une **fonction d'agrégation** est une fonction définie sur les éléments de la partition (elle rend une unique valeur pour chaque bloc de la partition).

Remarque
Les exemples classiques de fonctions d'agrégation que nous manipulerons dans la pratique sont les fonctions de décompte (compter le nombre d'éléments d'un bloc), des fonctions de moyenne, de minimum, de maximum.