ITC 1

Bases de données

Sommaire

1 Les bases de l'algèbre relationnelle

1.1 Les limites de la représentation plane

On appelle *représentation plane* la présentation de données sous la forme d'un tableau à deux entrées. Prenons pour exemple une liste de véhicules ci-dessous.

Marque	Modèle	Carburant	Cylindrée (cm^3)
Citroën	C_4 Picasso	Diesel	1997
	C_4 Picasso	Essence	1598
Volkswagen	Jetta	Essence	1197
	Jetta	Diesel	1598
Porsche	911 Carrera	Essence	3436
	911 GT_3 RS	Essence	3996

En bscPython, on pourrait représenter cette liste sous la forme d'une liste de listes.

```
voitures=[
["Citroen",
      ["C4 Picasso", "Diesel", "1997"],
      ["C4 Picasso", "Essence", "1598"]],
["Volkswagen",
      ["Jetta", "Essence", "1197"],
      ["Jetta", "Diesel", "1598"]],
["Porsche",
      ["911 Carrera 4S", "Essence","3800"],
      ["911 GT3 RS", "Essence","3996"]]
]
```

Application

ITC1.1 : Liste des véhicules-constructeur

Écrire une fonction permettant d'afficher la liste des véhicules d'un constructeur donné.

Application

ITC1.2 : Liste des véhicules-carburant

Écrire une fonction permettant d'afficher la liste des véhicules utilisant un carburant donné.

Pour simplifier cette dernière fonction, on aurait pu choisir une liste dans laquelle les voitures étaient triées par carburant et pas par marque. En revanche, le listing par constructeur devenait alors plus complexe.
La représentation plane nous a forcé à donner la priorité à une des caractéristiques (ici, le constructeur) des voitures considérées.

1.2 Définitions

Dans le modèle relationnel, on fait abstraction de toute priorité de caractéristique. On représente chaque voiture par un n-uplet :

(Marque, Modèle, Carburant, Cylindrée)

Une telle représentation est appelée un **schéma relationnel** noté S.
Les différents éléments de cette relation (marque, modèle,...) sont appelés les **attributs** A_i de la relation et sont distincts deux à deux. Le schéma relationnel peut ainsi se mettre sous la forme :

$$S = (A_1, \dots, A_n)$$

Remarque importante

Les différents attributs peuvent prendre leurs valeurs dans des ensembles appelés **domaines** des attributs notes $\text{dom}(A_i)$. Il est d'usage de représenter un schéma relationnel en rappelant le domaine de chaque attribut :

$$S = ((A_1, \text{dom}(A_1)), \dots, (A_n, \text{dom}(A_n))$$

Ainsi, dans le cas des listes de véhicules, le schéma relationnel serait :

$$S = ((\text{Marque}, \text{texte}), \dots, (\text{Cylindrée}, \mathbb{N}))$$

On notera $B \in S$ pour signifier que B est un des attributs de S.
De même, on notera $X \subset S$ pour signifier que X est un sous-n-uplet de S.

Exemple

On considère le schéma relationnel :

$$S = (\text{Marque, Modèle, Carburant, Cylindrée})$$

- $B = (\text{Modèle})$ est un attribut de S,
- $X = (\text{Marque, Modèle})$ est un sous-n-uplet de S.

On appelle **relation**, ou **table**, associée à un schéma relationnel S, un ensemble de n-uplets correspondant aux différentes valeurs prises par les attributs. Cette relation est notée $R(S)$.

Exemple

Dans le schéma relationnel S donné ci-dessus, la relation $R(S)$ associée au tableau des voitures est représentable par :

voitures1

marque	modèle	carburant	cylindrée
Citroën	C4 Picasso	Diesel	1997
Citroën	C4 Picasso	Essence	1598
Volkswagen	Jetta	Essence	1197
Volkswagen	Jetta	Diesel	1598
Porsche	911 Carrera	Essence	3436
Porsche	911 GT3 RS	Essence	3996

On notera $e \in R(S)$ un élément de la relation $R(S)$ et $e.A_i$ la valeur de l'attribut A_i associé à l'élément e.

Exemple

Dans la relation définie précédemment, si on numérote les 6 lignes de 1 à 6, on a :

$$3.\text{cylindrée} = 1197$$

On notera $e(X)$ l'opération particulière qui consiste à exprimer tous les attributs de l'élément e pris dans le sous-ensemble X des attributs :

$$X = (B_1, \dots, B_m) \subset S, \quad e(X) = (e.B_1, \dots, e.B_m)$$

2 Commandes simples

2.1 Projection

2.1.1 Formalisme relationnel

La projection d'une relation consiste à ne conserver que certains attributs pour tous les éléments. Soit $R(S)$ une relation de schéma S et $X \subset S$. On appelle **projection** de R selon X la relation :

$$\pi_X(R) = \{e(X) | e \in R\}$$

On voit, qu'une projection ne contient pas forcément autant de valeurs de la relation de départ : certains éléments sont fusionnés si leurs valeurs après projection sont identiques.

2.1.2 Commande SQL

Un **s**ystème de **g**estion de **b**ase de **d**onnées (S.G.B.D.) est un logiciel (ou un ensemble de logiciels) permettant de manipuler les données d'une base de données. Manipuler, c'est-à-dire sélectionner et afficher des informations tirées de cette base, modifier des données, en ajouter ou en supprimer (ce groupe de quatre opérations étant souvent appelé *"CRUD"*, pour *Create, Read, Update, Delete*).

Un langage spécifique a été développé pour interagir avec les bases de données. Il s'agit du langage SQL (*Structured Query Language*). Il permet de traduire simplement les opérateurs de l'algèbre relationnelle en utilisant des mots clefs explicites. Il est utilisé par de nombreux S.G.B.D. (*MySQL, MariaDB, PostgreSQL, SQLite,...*).

Point d'attention

Toutes les requêtes SQL commencent par le mot clef `SELECT` et terminent par un point virgule.

La projection consiste à sélectionner certains attributs d'une relation, ce qui se traduit en langage SQL par :

Il est possible d'obtenir tous les éléments d'une table avec $*$:

Application

ITC1.3 : Projection

Projeter la relation *voitures1* selon les attributs marque et modèle, et comparer au résultat de la requête SQL correspondante

2.2 Sélection simple

2.2.1 Formalisme relationnel

La sélection consiste à choisir les éléments de la relation vérifiant une certaine condition. Soit $R(S)$ une relation de schéma S, $A \in S$ et a une valeur possible de A. On appelle **sélection** de R selon $A = a$ la relation :

$$\sigma_{A=a}(R) = \{e \in R | e.A = a\}$$

Remarque importante

Si le domaine de A le permet (nombres entiers ou flottants), la sélection peut également porter sur une inégalité :

$$\sigma_{A \leqslant a}(R) = \{e \in R | e.A \leqslant a\}$$

Application

ITC1.4 : Sélection

Dans la relation *voitures*, que donne la sélection des voitures diesel ? De même pour la sélection des voitures de plus d'1,7 L de cylindrée.

2.2.2 Commande SQL

On utilise encore la commande `SELECT` associée au mot clef `WHERE` pour imposer la (ou les) condition(s).

Point d'attention

Le mot clef `SELECT` permet de réaliser à la fois les projections et les sélections.

Application

ITC1.5 : Sélection-SQL

Écrire les requêtes SQL correspondant aux deux sélections effectuées dans l'application précédente.

On peut alors combiner aisément la sélection avec une projection, pour afficher par exemple la liste des modèles existant en version diesel :

```
SELECT modèle FROM voitures WHERE carburant="Diesel";
```

2.3 Opérateurs ensemblistes usuels

2.3.1 Formalisme relationnel

Si deux relations ont le même schéma relationnel, il est possible de leur appliquer des opérateurs ensemblistes. Parmi ceux-ci, on aura principalement recours à l'union, l'intersection et la différence. Pour cette partie, on considère deux relations $vendeur_1$ et $vendeur_2$ contenant les listes de véhicules proposés par deux concessionnaires.

$vendeur_1$

marque	modèle	carburant	cyl.
Mercedes	C200	Diesel	2143
Subaru	WRX STI	Essence	2457
Bugatti	Chiron	Essence	7993

$vendeur_2$

marque	modèle	carburant	cyl.
Mercedes	C63 AMG	Essence	6208
Subaru	WRX STI	Essence	2457
Fiat	500	Essence	1242

- L'union $vendeur_1 \cup vendeur_2$ est une relation comprenant l'ensemble des éléments appartenant à $vendeur_1$ **ou** $vendeur_2$:

$vendeur_1 \cup vendeur_2$

marque	modèle	carburant	cyl.
Mercedes	C63 AMG	Essence	6208
Mercedes	C200	Diesel	2143
Subaru	WRX STI	Essence	2457
Bugatti	Chiron	Essence	7993
Fiat	500	Essence	1242

- L'intersection $vendeur_1 \cap vendeur_2$ est une relation comprenant l'ensemble des éléments appartenant à $vendeur_1$ **et** $vendeur_2$:

$vendeur_1 \cap vendeur_2$

marque	modèle	carburant	cyl.
Subaru	WRX STI	Essence	2457

- La différence $vendeur_1 - vendeur_2$ est une relation comprenant l'ensemble des éléments appartenant à $vendeur_1$ mais pas à $vendeur_2$:

$vendeur_1 - vendeur_2$

marque	modèle	carburant	cyl.
Mercedes	C200	Essence	1796
Bugatti	Chiron	Essence	7993

2.3.2 Commande SQL

- L'union est réalisée avec le mot clef `UNION` :

- L'intersection est réalisée avec le mot clef `INTERSECT` :

- La différence est réalisée avec le mot clef `EXCEPT` :

2.3.3 Application à la sélection composée

Comme chaque sélection donne une nouvelle relation, on peut utiliser les opérations ensemblistes sur les sélections pour exprimer des conditions complexes.

Application

ITC1.6 : Opérateurs ensemblistes

On souhaite obtenir la liste des véhicules essence de cylindrée inférieure à 1,5 L proposés par le $vendeur_1$.
Donner deux requêtes SQL permettant d'effectuer cette opération : l'une à l'aide d'opérateurs ensemblistes, et l'autre à l'aide d'opérateurs logiques.

2.4 Renommage

Le renommage sert à changer **temporairement** le nom d'un attribut. Ce renommage n'affecte que la requête en cours (en vue de l'affichage) mais ne modifie pas la structure relationnelle.

Exemple

Pour afficher la liste des véhicules proposés par le vendeur 1 avec des noms de colonne en anglais. La requête suivante donnerait :

```
SELECT marque AS brand, modèle AS model, carburant AS fuel FROM vendeur1;
```

$vendeur_1$

brand	model	fuel	cyl.
Mercedes	C200	Diesel	2143
Subaru	WRX STI	Essence	2457
Bugatti	Chiron	Essence	7993

2.5 Filtrage des résultats

À la toute fin d'une requête il est possible de trier les résultats et de n'en renvoyer qu'une partie. Ces opérations se notent :

- `ORDER BY a ASC` pour trier suivant l'attribut `a` par ordre croissant (on utilise le mot-clef `DESC` pour un tri par ordre décroissant) ;
- `LIMIT n` pour limiter la sortie à `n` enregistrements ;
- `OFFSET` pour débuter à partir du `n`-ième enregistrement.

Si l'on souhaite utilise simultanément ces fonctions de filtrage, elles doivent apparaître dans cet ordre.

Application

ITC1.7 : Filtrage

Q.1. Écrire une requête permettant d'obtenir l'ensemble des véhicules diesel dans la table *voiture1* triés par ordre alphabétique croissant de la marque.

Q.2. Écrire une requête permettant d'afficher les 3 voitures ayant la plus grosse cylindrée triées par ordre décroissant

Il est possible d'effectuer un tri sur plusieurs attributs en les séparant par une virgule :
`SELECT * FROM relation ORDER BY attr1,attr2 ASC;`
Dans ce cas, les données sont d'abord triées selon l'attribut *attr1*, les cas d'égalité sont triés avec l'attribut *attr2*.

3 Clefs

3.1 Définition

Une **clef** pour une relation est un ensemble d'attributs qui permet d'identifier chaque élément de la relation de manière unique.
Soit $R(S)$ une relation de schéma S et $K \subset S$. On dit que K est une clef pour R si et seulement si pour tous éléments $e_1, e_2 \in R$, on a :

$$e_1(K) = e_2(K) \Rightarrow e_1 = e_2$$

Exemple

Considérons la relation *voitures1* ci-dessous.
L'attribut *marque* ou l'ensemble (*marque, modèle*) ne constituent pas des clefs pour la relation car deux voitures ont pour attributs (*Volkswagen, Jetta*).

voitures1

marque	modèle	carburant	cylindrée
Citroën	C4 Picasso	Diesel	1997
Citroën	C4 Picasso	Essence	1598
Volkswagen	Jetta	Essence	1197
Volkswagen	Jetta	Diesel	1598
Porsche	911 Carrera	Essence	3436
Porsche	911 GT3 RS	Essence	3996

En revanche, l'ensemble (*modèle, carburant*) est bien une clef pour la relation.

Remarque importante

La relation ci-dessus pourrait être complétée en ajoutant les différentes finitions disponibles pour un même modèle (et même motorisation).

voitures

marque	modèle	carburant	cylindrée	finition
Citroën	C4 Picasso	Diesel	1997	Business
Citroën	C4 Picasso	Diesel	1997	Confort
Citroën	C4 Picasso	Diesel	1560	Business
Citroën	C4 Picasso	Diesel	1560	Confort
Citroën	C4 Picasso	Essence	1598	Confort
Volkswagen	Jetta	Essence	1197	Trendline
Volkswagen	Jetta	Diesel	1598	Confrotline

L'ensemble (*modèle, carburant*) ne suffit plus et il faudrait considérer, par exemple, l'ensemble (*modèle, carburant, cylindrée, finition*) pour obtenir une clef.
Pour éviter d'utiliser des clefs à rallonge, on ajoute usuellement attribut d'identification (noté *Id*) dont la valeur est un entier auto-incrémenté. Cet entier constitue une clef à lui seul.

3.2 Clef primaire et clef étrangère

Une **clef primaire** pour une relation est un seul attribut qui permet d'identifier chaque élément de la relation de manière unique.
Soit $R(S)$ une relation de schéma S et $A \in S$ un attribut de S. On dit que A est une clef primaire pour R si et seulement si pour tous éléments $e_1, e_2 \in R$, on a :

$$e_1.A = e_2.A \Rightarrow e_1 = e_2$$

Exemple

voitures2

Id	marque	modèle	carburant	cylindrée
1	Citroën	C4 Picasso	Diesel	1997
2	Citroën	C4 Picasso	Diesel	1997
3	Citroën	C4 Picasso	Diesel	1560
4	Citroën	C4 Picasso	Diesel	1560
5	Citroën	C4 Picasso	Essence	1598
6	Volkswagen	Jetta	Essence	1197
7	Volkswagen	Jetta	Diesel	1598

Dans la relation ci-dessus, l'attribut *Id* est bien une clef primaire pour la relation.

L'intérêt d'une clef primaire est de faciliter les liens entre deux relations d'une même base de données.

Exemple

véhicules

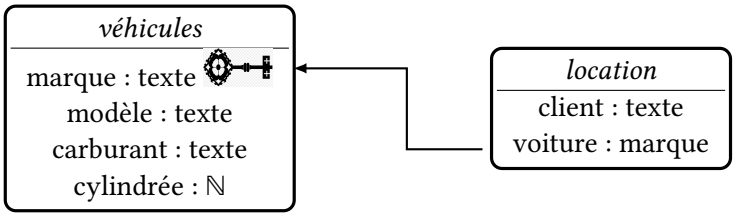
marque	modèle	carburant	cylindrée
Ariel	Atom	Essence	1998
Mercedes	C200	Diesel	2143
Subaru	WRX STI	Essence	2457
Bugatti	Chiron	Essence	7993

location

client	voiture
Alpha	Mercedes
Beta	Bugatti

L'attribut *marque* de la table *véhicules* constitue une clef primaire. Ainsi, en faisant simplement référence à la marque dans la table *location*, on peut avoir les caractéristiques détaillées de la voiture louée par chaque personne.

En revanche, dans l'exemple cité ci-dessus, rien n'empêche de créer un élément dans la table *location* qui emprunte une voiture qui n'existe pas. Une solution existe : c'est la **clef étrangère**.



Dans l'exemple, l'attribut *marque* constitue une clef primaire pour la relation *véhicules*. Si on impose que le domaine de l'attribut *voiture* de la table *location* soit constitué par les marques apparaissant dans la table *véhicules*, alors *marque* est une clef étrangère pour la table *location*. Il sera alors impossible d'insérer une voiture empruntée qui ne fasse pas partie de la liste des véhicules disponibles.

3.3 Associations

L'introduction de la clé étrangère a été l'occasion de présenter une première *association* entre deux tables : au lieu de mettre toutes les informations «Individu / Véhicule loué» dans une seule table, on sépare les informations dans deux tables distinctes.
Considérons une nouvelle table un peu plus remplie :

cinéma

titre	annee	nom_realisateur	prenom_realisateur
Alien	1979	Scott	Ridley
Vertigo	1958	Hitchcock	Alfred
Psychose	1960	Hitchcock	Alfred
Pulp Fiction	1995	Tarantino	Quentin
Titanic	1997	Cameron	James
Inglourious Basterds	2009	Tarantino	Quentin
Reservoir Dogs	1992	Tarantino	Quentin
Avatar	2009	Cameron	James

Comme on peut le voir, certaines informations (nom et prénom du réalisateur) sont répétées plusieurs fois pour les réalisateurs prolifiques. Pour alléger la base de donnée, on peut séparer cette table en deux :

films

titre	annee	id_rea (Clé étrangère)
Alien	1979	1
Vertigo	1958	2
Psychose	1960	2
Pulp Fiction	1995	5
Titanic	1997	6
Inglourious Basterds	2009	5
Reservoir Dogs	1992	5
Avatar	2009	6

réalisateurs

id (Clé primaire)	nom	prenom
1	Scott	Ridley
2	Hitchcock	Alfred
5	Tarantino	Quentin
6	Cameron	James

L'attribut `id` constitue une **clé primaire** de la table *réalisateurs*. Au contraire, l'attribut `id_rea` n'est pas une clé primaire de la table *films*. Mais comme cet attribut fait référence à la clé primaire d'une autre table, `id_rea` est une **clé étrangère** (référençant l'attribut `id` de la table *films*). Dans ce schéma relationnel, un réalisateur peut être associé à plusieurs films : on dit qu'on a une **association 1-***.
Plus compliqué maintenant : certains films ont deux réalisateurs. On doit donc faire correspondre plusieurs films à plusieurs réalisateurs : c'est une **association** *-*.
Une première solution à ce problème serait d'ajouter une seconde colonne `id_rea2` pour mettre éventuellement l'identifiant du second réalisateur… Et le jour où on doit ajouter un film ayant trois réalisateurs, il faut tout recommencer.
Seconde solution, dupliquer les films ayant plusieurs réalisateurs comme sur le schéma ci-dessous. Cette solution oblige à multiplier les entrées pour certains films. On a séparé la table *cinéma* en deux pour éviter les répétitions des réalisateurs, ce n'est pas pour se mettre à répéter les films…

films

titre	annee	id_rea
Alien	1979	1
Vertigo	1958	2
Psychose	1960	2
Pulp Fiction	1995	5
Titanic	1997	6
Inglourious Basterds	2009	5
Reservoir Dogs	1992	5
Avatar	2009	6
Matrix	1999	7
Matrix	1999	8

réalisateurs

id	nom	prenom
1	Scott	Ridley
2	Hitchcock	Alfred
5	Tarantino	Quentin
6	Cameron	James
7	Wachowski	Lilly
8	Wachowski	Lana

La solution est de couper cette association *-* en deux associations 1-* à l'aide d'une table intermédiaire comme illustré sur la figure ci-dessous.

associations

id_film	id_rea
1	1
2	2
3	2
4	3
5	4
6	5
7	6
8	5
9	5
10	6
11	7
11	8

films

id	titre	annee
1	Alien	1979
2	Vertigo	1958
3	Psychose	1960
4	Kagemusha	1980
5	Volte-face	1997
6	Pulp Fiction	1995
7	Titanic	1997
8	Inglourious Basterds	2009
9	Reservoir Dogs	1992
10	Avatar	2009
11	Matrix	1999

réalisateurs

id	nom	prenom
1	Scott	Ridley
2	Hitchcock	Alfred
3	Kurosawa	Akira
4	Woo	John
5	Tarantino	Quentin
6	Cameron	James
7	Wachowski	Lilly
8	Wachowski	Lana

Dans cette situation, les attributs `id` des tables *films* et *réalisateurs* sont deux clés primaires, et la table d'association contient les deux clés étrangères référençant les éléments qu'elle relie. Un seul film peut être lié à plusieurs éléments de la table intermédiaire : c'est bien une association 1-*. De même, un seul réalisateur peut être relié à plusieurs éléments de la table intermédiaire.

4 Opérations avancées

Les opérations définies jusqu'à présent n'étaient applicables qu'à une seule relation ou à deux relations de même schéma relationnel. Les opérateurs suivants permettent de croiser les informations présentes dans plusieurs tables de formats différents.

4.1 Produit cartésien

4.1.1 Formalisme relationnel

Le produit cartésien $R \times R'$ de deux relations R et R', de schémas quelconques, consiste à associer tous les éléments de R avec tous les éléments de R'.

élèves

nom_élève
Alpha
Beta
Gamma

profs

nom_prof
Aleph
Beth

élèves×profs

nom_élève	nom_prof
Alpha	Aleph
Alpha	Beth
Beta	Aleph
Beta	Beth
Gamma	Aleph
Gamma	Beth

Remarque importante

La division cartésienne existe en algèbre relationnelle existe pour des raisons de complétude mais elle est absente des langages de requête.

4.1.2 Commande SQL

Le produit cartésien est obtenu très simplement en langage SQL en utilisant une projection de deux relations :

4.2 Jointure

4.2.1 Formalisme relationnel

La jointure sert à recoller deux relations différentes ayant au moins un attribut en commun.
Soient $R(S)$ et $R'(S')$ deux relations de schémas relationnels disjoints.
Soient $A \in S$ et $A' \in S'$ tels que $\text{dom}(A) = \text{dom}(A')$.
L'opération :

$$R[A = A']R' = \{e \in R \times R' | e.A = e.A'\}$$

est appelée **jointure symétrique** de R et R'. Elle peut également être notée $R \bowtie R'$, mais l'attribut servant à faire la jointure n'est pas précisé avec cette notation.

Exemple

Considérons les deux relations suivantes :

voitures

marque	modèle	carburant	cyl.
Mercedes	C200	Essence	1796
Subaru	Impreza	Diesel	1998
Bugatti	Chiron	Essence	7993
Fiat	500	Diesel	1248

constructeurs

constructeur	pays	groupe
Citroën	France	PSA
Porsche	Allemagne	Porsche
Mercedes	Allemagne	Daimler
Bugatti	France	Volkswagen AG
Subaru	Japon	Subaru

L'attribut `marque` de la table *voitures* correspond à l'attribut `constructeur` de la table *constructeurs*. On va alors réaliser la jointure symétrique selon (`marque = constructeur`) qui donnera une nouvelle relation.

voitures [marque = constructeur] *constructeurs*

marque	modèle	carburant	cylindrée	constructeur	pays	groupe
Mercedes	C200	Essence	1796	Mercedes	Allemagne	Daimler
Subaru	Impreza	Essence	1994	Subaru	Japon	Subaru
Bugatti	Chiron	Essence	7993	Bugatti	France	Volkswagen AG

Remarque importante

Les attributs servant à la jointure sont dupliqués. Une projection bien placée saura résoudre ce problème

Point d'attention

La jointure étudiée est ici une jointure symétrique, aussi appelée **jointure interne** en langage SQL. Elle exige qu'il y ait des données de part et d'autre de la jointure, c'est-à-dire que l'élément e sur lequel on fait la jointure doit exister dans les deux relations R et R'.
Si un élément de R n'a pas de correspondance dans R' (et réciproquement), cet élément n'apparaîtra pas dans la jointure.

4.2.2 Commande SQL

La jointure est obtenue avec le mot clef `JOIN ... ON`. La jointure de l'exemple ci-dessus s'écrit :

Remarque importante

Lorsqu'on les attributs servant à joindre deux tables ont le même nom, on peut utiliser le mot clef `USING` au lieu de `ON`.

Si les deux relations *constructeurs* et *voitures* étaient sous cette forme :

voitures

marque	modèle	carburant	cyl.
Mercedes	C63 AMG	Essence	6208
Subaru	WRX STI	Essence	2457
Bugatti	Chiron	Essence	7993

constructeurs

marque	pays	groupe
Mercedes	Allemagne	Daimler
Bugatti	France	Volkswagen AG
Subaru	Japon	Subaru

On pourrait réaliser la jointure sur la marque avec : `SELECT * FROM voitures JOIN constructeurs USING marque;`

4.2.3 Autojointure

Même si l'intérêt n'apparaît pas de prime abord, il peut être intéressant de réaliser la jointure d'un table avec elle-même.
La syntaxe est la même que précédemment à l'exception de la nécessité de renommer les copies de la table pour différencier les attributs.

Remarque importante

La syntaxe d'une autojointure est la suivante : `SELECT * FROM table [AS] table1 JOIN table [AS] table2 ON condition`
le mot-clef `[AS]` *est indiqué entre crochet car il est ici facultatif.*

Application

ITC1.8 : Autojointures

On se donne la base de données `windsor` suivante.

windsor

id	nom	mere	pere	conjoint
1	Elisabeth II			2
2	Philip			1
3	Charles III	1	2	5
4	Diana			
5	Camilla			3
6	William	4	3	7
7	Kate			6
8	George	7	6	
9	Charlotte	7	6	
10	Henry	4	3	
11	Anne	1	2	
12	Andrew	1	2	
13	Edward	1	2	
14	Meghan			10
15	Archie	14	10	
16	Louis	7	6	
17	Lilibet	14	10	

Écrire des requêtes SQL qui permettent d'afficher :

Q.1. le nom de chaque membre de la famille royale ainsi que celui de sa mère

Q.2. les enfants de Diana

Q.3. les parents de William

Q.4. les frères et sœur de Charles

4.3 Agrégation

Les opérations étudiées jusqu'à présent sont des fonctions scalaires : elles s'appliquent à chaque ligne indépendamment. Les fonctions d'agrégation regroupent les lignes pour effectuer une opération sur ce regroupement.

4.3.1 Formalisme relationnel

Pour toutes les explications qui suivent, on va travailler avec la relation suivante :

voitures3

marque	modèle	carburant	cylindrée
Mercedes	C63 AMG	Essence	6208
Mercedes	C200	Essence	1796
Mercedes	C200	Diesel	2143
Subaru	WRX STI	Essence	2457
Subaru	Impreza	Essence	1994
Subaru	Impreza	Diesel	1998
Bugatti	Chiron	Essence	7993
Renault	Twingo	Essence	1149
Abarth	500	Essence	1368
Fiat	500	Essence	1242
Fiat	500	Diesel	1248

Lors de la réalisation d'une agrégation, on doit distinguer deux opérations distinctes :

L'application de la fonction d'agrégation Les fonctions d'agrégation classiques sont :

- comptage(A) : compte le nombre d'éléments d'attribut A,
- max(A) (resp. min(A)) : renvoi le plus grand (resp. le plus petit) élément de A,
- somme(A) : donne la somme des éléments de A,
- moyenne(A) : donne la valeur moyenne des éléments de A.

L'agrégation de la relation R en appliquant la fonction f à l'attribut A se note de la manière suivante :

$$\gamma_{f(A)}(R)$$

Il est possible de créer, par agrégation, une relation qui résulte de l'application de plusieurs fonctions d'agrégation. Soient $B_1, \dots, B_n \in S$ et $f_1, \dots, f_n$ des fonctions d'agrégation. La relation obtenue par application des fonctions $f_1, \dots, f_n$ aux attributs $B_1, \dots, B_n \in S$ est notée :

$$\gamma_{f_1(B_1),\dots,f_n(B_n)}(R)$$

Exemple

Si on cherche la plus grande cylindrée, ou les cylindrées extrémales dans la relation *voitures*, on peut écrire :

$\gamma_{\max(\text{cylindrée})}(voitures)$

max(*cylindrée*)
7993

$\gamma_{\max(\text{cylindrée}),\min(\text{cylindrée})}(voitures)$

max(*cylindrée*)	min(*cylindrée*)
7993	1149

Le regroupement des valeurs Au lieu d'appliquer la fonction d'agrégation à la totalité de la table, il est possible de regrouper les éléments selon un ou plusieurs attributs pour ensuite appliquer la fonction à chacun des regroupements.
Commençons par un exemple pour voir ce que peut donner un tel regroupement.
Soient $A_1, \dots, A_m, B_1, \dots, B_n \in S$ et $f_1, \dots, f_n$ des fonctions d'agrégation. L'application des fonctions $f_1, \dots, f_n$ aux attributs $B_1, \dots, B_n \in S$ regroupés par rapport aux attributs $A_1, \dots, A_m$ est notée :

$$A_1,\dots,A_m\gamma_{f_1(B_1),\dots,f_n(B_n)}(R)$$

Exemple

Si on cherche les cylindrées moyennes pour chaque marque, il faut donc appliquer la calcul de la moyenne aux éléments ayant comme attribut commun la *marque* :

$_{\text{marque}}\gamma_{\text{moyenne(cylindrée)}}(\textit{voitures})$

marque	moyenne(*cylindrée*)
Mercedes	3382,33
Subaru	2149,67
Bugatti	7793
Renault	1149
Abarth	1368
Fiat	1245

Remarque importante

Il est tout à fait possible de composer l'agrégation avec une sélection :

- sélection en amont : $_{A_1,\dots,A_m}\gamma_{f_1(B_1),\dots,f_n(B_n)}(R) \circ \sigma_P$ La sélection σ_P est effectuée **avant** l'agrégation. Elle porte donc sur la relation R et la condition P porte sur les attributs de R.
- sélection en aval : $\sigma_P \circ_{A_1,\dots,A_m} \gamma_{f_1(B_1),\dots,f_n(B_n)}(R)$ La sélection σ_P est effectuée **après** l'agrégation. Elle porte donc sur la relation γ et la condition P porte sur les attributs de γ.

4.3.2 Commandes SQL

La traduction de $\gamma_{f(A)}(R)$ en langage SQL se fait avec :

où les fonctions `f` sont :

fonction d'agrégation	langage SQL
comptage	`COUNT`
maximum	`MAX`
minimum	`MIN`
moyenne	`AVG`
somme	`SUM`

L'expression de ${}_{A_1,\dots,A_m}\gamma_{f_1(B_1),\dots,f_n(B_n)}(R)$ est légèrement plus complexe : il faut recourir au mot clef `GROUP BY` pour réaliser le regroupement.

Application

ITC1.9 : Fonctions d'agrégation

Écrire les requêtes SQL correspondant aux exemples donnés dans la partie précédente (cylindrée maximale, cylindrées extrémales et cylindrée moyenne).

Remarque importante

- La sélection en amont se fait comme une sélection classique, c'est-à-dire avec le mot clef `WHERE`.

  $${}_{A_1,\dots,A_m}\gamma_{f_1(B_1),\dots,f_n(B_n)}(R) \circ \sigma_{P_1}$$

  s'obtiendrait avec :

- La sélection en aval se fait avec le mot clef `HAVING`.

  $$\sigma_{P_2} \circ {}_{A_1,\dots,A_m}\gamma_{f_1(B_1),\dots,f_n(B_n)}(R)$$

  s'obtiendrait avec :

Point d'attention

Attention à l'ordre des mots clefs. `WHERE` porte sur la relation R. Il doit donc être juste après l'instruction `FROM R` et avant l'instruction de regroupement.
Au contraire, `HAVING` porte sur la relation agrégée. Il doit donc être placé à la fin de l'instruction.

Application

ITC1.10 : Sélections en aval et en amont

Q.1. Écrire une requête SQL qui permet de connaître le nombre de voitures par marque dont la cylindrée est supérieure à 2,0 L.

Q.2. Écrire une requête qui retourne les valeurs moyennes de cylindrées par marque lorsqu'elles sont supérieures à 2,5 L.