Documentations complètes
Python3 : https://docs.python.org/3/
Nympy : https://docs.scipy.org/doc/
Matplotlib : http://matplotlib.org/
SQL : http://sql.sh/

4

Base de données - SQL Partie II

4.1 Un mot sur la conception

Commençons par un mauvais exemple : La gestion d'une vidéothèque grâce à la table :

Vidéo(Titre : `str`, ActeurPrincipal : `str`, Réalisateur : `str`, Année : `int`, NomClient : `str`, PrénomClient : `str`, AdresseClient : `str`, DateEmprunt : `date`)

Le schéma est mal choisi, les données sont mal représentées. En effet :

— Seuls les DVD empruntés existent dans cette base.

— Seuls les clients ayant un emprunt en cours existent.

— Un film peut être co-réalisé.

— Il n'y a pas d'historique des emprunts.

— Si un client emprunte plusieurs DVD on l'enregistre plusieurs fois...

On comprend qu'il faut séparer les clients, le stock et les emprunts, donc avoir plusieurs tables.

Les étapes essentielles de la conception d'une base de données :

1. Observer la réalité de ce qui doit être représenter.
2. Collecter les informations.
3. Modéliser ⇒ Schéma du modèle entités-associations, que l'on présentera plus loin.
4. Générer des tables depuis le schéma du modèle entités-associations par une démarche systématique.

4.2 Les clés dans le modèle relationnel

Une base de données est généralement constituée de plusieurs tables.

Il est donc nécessaire de pouvoir croiser les données différentes tables pour obtenir les informations cherchées, et il faut pouvoir rapidement accéder et distinguer les enregistrements des différentes tables.

Pour ça on utilise des clés.

4.2.1 Clé

Définition 4.1 (Clé)**.**

Une clé pour une relation R de schéma S est un sous-schéma $K \subset S$ qui caractérise entièrement la table : les valeurs de ces attributs caractérisent une unique ligne.

Autrement dit K est une clé si et seulement si :

$$\forall e, e' \in R,\ e(K) = e'(K) \Rightarrow e = e'.$$

Exemples :

car_wish_list			
Marque	Modèle	Couleur	Année
Porsche	918 - Spyder	Gris	2010
Porsche	911 - GT3 Cup	Bleu	2013
Ford	Mustang GT	Bleu	2011
Porsche	550 - Spyder	Gris	1953
Ferrari	GTE	Noir	2015
Ferrari	288 - GTO	Rouge	1984
Porsche	918 - Spyder	Noir	2012
Ferrari	288 - GTO	Jaune	1987

— $\{\text{Marque}, \text{Modèle}\}$ n'est pas une clé.
— $\{\text{Marque}, \text{Couleur}\}$, $\{\text{Année}\}$ et $\{\text{Marque}, \text{Modèle}, \text{Couleur}, \text{Année}\}$ sont des clés.

4.2.2 Clé primaire

Remarque Une relation $R(S)$ admet toujours au moins une clé.

Définition 4.2 (Clé candidates et clé primaires)**.**

Soit $R(S)$ une relation de schéma S.

On appelle clé candidate de R toute clé K qui est minimale. Autrement dit pour tout $K' \subsetneq K$, K' n'est pas une clé.

On appelle clé primaire une clé candidate que l'on choisit pour caractériser la table.

Dans la pratique il est d'usage courant de choisir une clé primaire de cardinal 1, autrement dit réduite à un attribut. Certain SGBD l'impose même, et c'est pourquoi il rajoute un *index* c'est à dire un numéro de ligne.

— Dans la suite on choisira toujours des clés primaires réduites à un attribut. Autrement dit une clé primaire sera un attribut $A \in S$, tel que

$$\forall e, e' \in R,\ e.A = e'.A \Rightarrow e = e'.$$

Dans l'exemple précédent, la seule clé primaire est donc $\{\text{Année}\}$.

— On soulignera une clé primaire dans une relation $R(A_1, \ldots, \mathbf{A_i}, \ldots, A_n)$.
On notera donc

Voiture(Marque,Modèle,Couleur,<u>Année</u>).

4.2.3 Clé étrangère

Reprenons nos tables :

car_wish_list				
id	Marque	Modèle	Couleur	Année
1	Porsche	918 - Spyder	Gris	2010
2	Porsche	911 - GT3 Cup	Bleu	2013
3	Ford	Mustang GT	Bleu	2011
4	Porsche	550 - Spyder	Gris	1953
5	Ferrari	GTE	Noir	2015
6	Ferrari	288 - GTO	Rouge	1984
7	Porsche	918 - Spyder	Noir	2012
8	Ferrari	288 - GTO	Jaune	1987
9	Ford	Mustang GT	Rouge	2014
10	Aston-Martin	DB5	Gris	1963
11	Ford	Shelby GT500	Bleu	2013

specifications			
id	Type	Puissance	Poids
1	918 - Spyder	887	1675
2	911 - GT3 Cup	435	1395
3	Mustang GT	412	1580
4	550 - Spyder	110	760
5	288 - GTO	400	1224
6	DB5	286	1465
7	Shelby GT500	671	1747

— **Type** et **id** sont des clés candidates pour la relation *spécifications*.

— Choisissons **Type** comme clé primaire. Elle va nous permettre de faire un lien avec la table *car_wish_list*, car on peut considérer que le domaine de l'attribut **Modèle** de la table *car_wish_list* est exactement l'ensemble des valeurs de l'attribut **Type** de la table *spécifications*.
On fait un lien entre les deux tables en décidant de dire que le domaine de **Modèle** est **Type**.

Définition 4.3 (Clé étrangère)**.**

Un attribut qui fait référence à une autre table est appelé clé étrangère, c'est en général une clé primaire de la seconde table.

Les clés étrangères ont pour fonction principale la vérification de l'intégrité de la base. Le SGBD les utilise pour vérifier que l'on n'entre pas de données incohérentes dans une table.

4.3 Un mot sur les schémas entités-associations

4.3.1 Les notions de base

Une entité est un objet, une chose concrète ou abstraite qui peut être reconnue distinctement et qui est caractérisée par son unicité.

Par exemple, Karen Spärck Jones[i], le premier livre de ma bibliothèque ou encore la Porsche qui est dans mon sous-sol.

i. Karen Spärck Jones 1935-2007 est une scientifique britannique, chercheuse en informatique.

Un type entité désigne en ensemble d'entités qui possèdent une sémantique et des caractéristiques communes.

Par exemple les personnes, les livres et les voitures sont des type entité. Effectivement les informations qui caractérise une voiture ne change pas d'une voiture à l'autre.

On dit qu'une entité est une occurence de son type entité. Les caractéristiques d'un type entité s'appelle ses attributs, chacun de ses attributs peut prendre des valeurs dans un domaine (entier, chaîne de caractères, booléen, flottant).

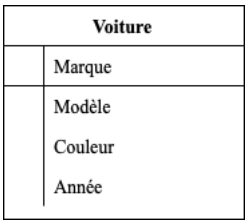


FIGURE 4.1 – Représentation d'un type entité.

Une association est une connexion entre plusieurs entités. Par exemple, Karen Spärck Jones est l'auteur de l'article « A statistical interpretation of term specificity and its application in retrieval », ou encore Paul Smith possède une license.

Un type association désigne un ensemble de relations qui possèdent les mêmes caractéristiques.

Comme les types entité, les types association sont définis à l'aide d'attributs qui prennent leur valeur dans les associations.

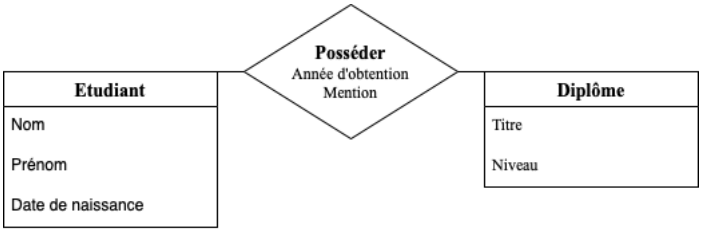


FIGURE 4.2 – Représentation entité-relation.

Une clé primaire d'un type entité ou d'un type association est constitué par une partie de ses attributs qui doivent avoir une valeur unique pour chaque entité ou association de ce type.

Par exemple le numéro de sécurité sociale pour une personne, l'immatriculation pour une voiture ou le code ISBN pour un livre

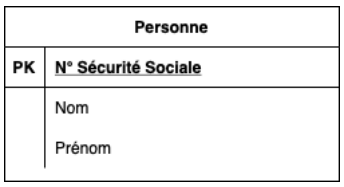


FIGURE 4.3 – Représentation d'une clé primaire.

Lorsqu'un attribut ou des attributs d'une table font référence à une clé primaire d'une autre table on appelle ce ou ces attributs une clé étrangère.

La cardinalité d'un lien reliant un type association et un type entité précise le nombre de fois minimal et maximal d'interventions d'une entité du type entité dans une association du type association. La cardinalité minimale doit être inférieure ou égale à la cardinalité maximale.

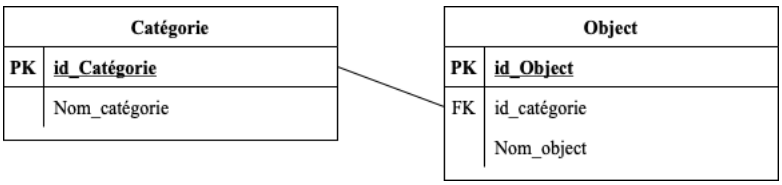


FIGURE 4.4 – Représentation d'une clé étrangère.

Par exemple une personne peut être l'auteur de zéro à un nombre quelconque de livre, mais un livre doit avoir au moins un auteur mais peut aussi en avoir un nombre quelconque. De même, un salarié d'une entreprise ne peut être affecté qu'à un et un seul service, et un service pour exister doit avoir au moins un salarié affecté, mais peut en avoir un nombre quelconque. On représente la cardinalité sur les schémas de la manière suivante :

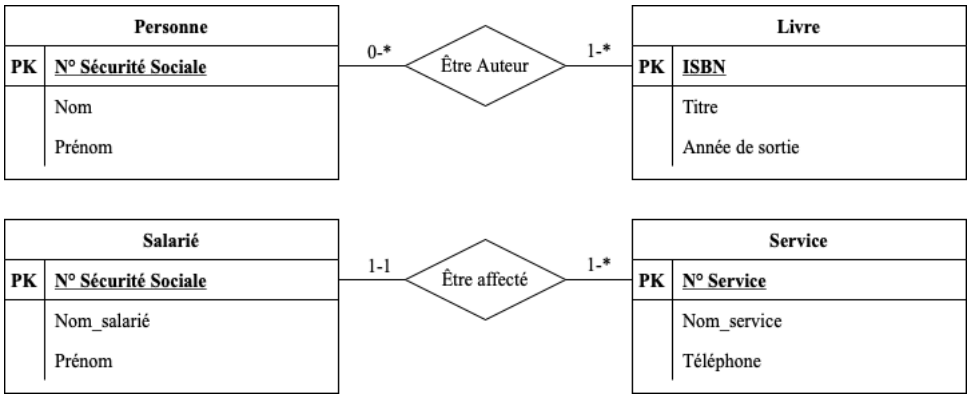


FIGURE 4.5 – Représentation de la cardinalité.

On utilise « * » pour indiquer un nombre quelconque.

Les cardinalités admises sont de quatre types :

Cardinalité	Signification
0-1	une occurrence du type-entité peut exister en étant impliquée soit dans aucune association soit au maximum dans une seule.
0-*	une occurrence du type entité peut exister tout en n'étant impliquée dans aucune association et peut être impliquée, sans limitation, dans plusieurs associations.
1-1	une occurrence du type entité ne peut exister que si elle est impliquée dans exactement (au moins et au plus) une association.
1-*	une occurrence du type entité ne peut exister que si elle est impliquée dans au moins une association.

Les type-associations peuvent être catégorisés en fonction des cardinalités maximales de leurs liens :

Description	Appelation
La cardinalité maximale sur chacune de ses pattes est 1.	Type-association $1-1$
Une cardinalité maximale est à 1 et une cardinalité maximale est à $*$.	Type-association $1-*$
La cardinalité maximale de chacun de ses liens est à $*$.	Type-association $*-*$

4.3.2 Passage du modèle entité-association au modèle relationnel

Nous ne présenterons pas ici toute la démarche systématique du passage du modèle entité-association au modèle relationnel, qui dépasse largement le cadre de ce cours. Mais seulement un bref survol de ce processus.

Règle n°1

1. Un type entité devient une relation c'est-à-dire une table dans la base.
2. La clé primaire du type entité devient la clé primaire de la table.
3. Les attributs du type entité deviennent ceux de la table.

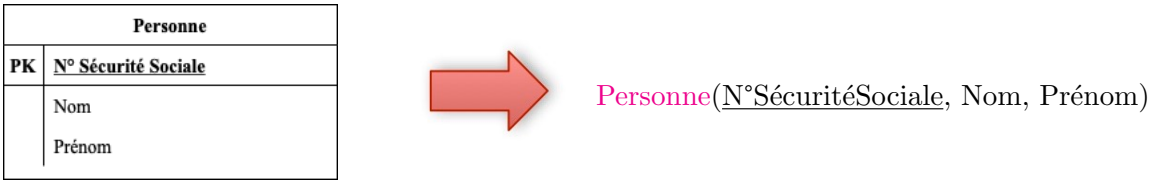


FIGURE 4.6 – Transformation d'un type entité en relation

Règle n°2

Une association de type $1 - *$ se traduit par la création d'une clé étrangère dans la relation correspondante à l'entité côté 1. Cette clé étrangère référence la clé primaire de la relation correspondant à l'autre entité.

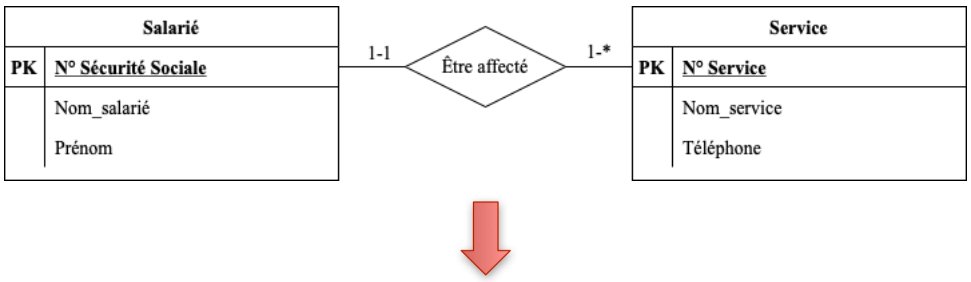


FIGURE 4.7 – Transformation d'un type association $1 - *$

De cette manière même si la base contient deux tables on peut répondre à toutes les questions comme :

— Quels sont les salariés affectés à la comptabilité ?

— Dans quel service travaille Bob Morane ?

Règle n°3

Une association de type $* - *$ se traduit par la création d'une table dont la clé primaire est composée des clés étrangères référençant les relations correspondant aux entités liées par l'association.

Les éventuelles propriétés de l'association deviennent des attributs de la relation.

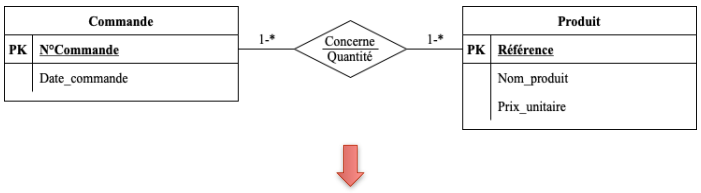


FIGURE 4.8 – Transformation d'un type association $* - *$

L'association « Concerne » a été renommée pour avoir un nom significatif.

Cas particulier des associations $1-1$

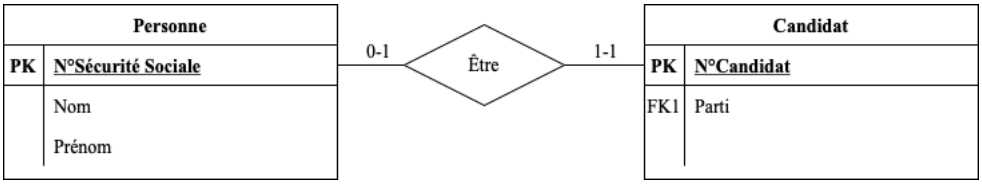


FIGURE 4.9 – Association $1-1$

D'après les règles énoncées précédemment nous devrions traduire ce schéma en deux tables :

Personne(N°SécuritéSociale, Nom, Prénom, N°Candidat)
Candidat(N°Candidat, Parti, N°SécuritéSociale)

Mais comme l'association est du type $1-1$, donc est entièrement matérialisé dans la relation Candidat par l'attribut `N°SécuritéSociale`. Il est donc inutile de la rematérialiser dans la relation Personne. L'attribut N°Candidat dans la relation Personne doit donc être supprimé. D'autre part, dans la relation Candidat, l'attribut N°SécuritéSociale, en plus d'être une clé étrangère, constitue une clé candidate. On peut donc se passer de la clé N°Candidat. On obtient alors :

Personne(N°SécuritéSociale, Nom, Prénom)
Candidat(N°SécuritéSociale, Parti)

4.4 Opérateurs complexes

4.4.1 Le produit cartésien

Définition 4.4 (Produit cartésien)**.**

Soient $R(S)$ et $R'(S')$ deux relations de schémas $S = (A_1, \ldots, A_p)$ et $S' = (B_1, \ldots, B_q)$ disjoints.
On appelle produit cartésien, la relation notée $R \times R'$, telle que :

$$\left\{(v_1, \ldots, v_p, v'_1, \ldots, v'_q) \mid (v_1, \ldots, v_p) \in R \text{ et } (v'_1, \ldots, v'_q) \in R'\right\}.$$

Son schéma est notée $S \biguplus S'$, où :

$$S \biguplus S' = (A_1, \ldots, A_p, B_1, \ldots, B_q).$$

Titre
Ainsi parlait Zarathoustra
Le cas Wagner
Le Traité des cinq roues

Auteur
Frédéric Nietzsche
Musashi Miyamoto

Titre	Auteur
Ainsi parlait Zarathoustra	Frédéric Nietzsche
Le cas Wagner	Frédéric Nietzsche
Le Traité des cinq roues	Frédéric Nietzsche
Ainsi parlait Zarathoustra	Musashi Miyamoto
Le cas Wagner	Musashi Miyamoto
Le Traité des cinq roues	Musashi Miyamoto

Informatique 4.1.

Pour obtenir, en SQL, le produit cartésien des relations $R1$ et $R2$:

```
SELECT *
FROM R1 , R2
```

On peut aussi utiliser :

```
SELECT *
FROM R1 CROSS JOIN R2
```

Il faut faire attention la table obtenue peut rapidement être très très grosse. On peut utiliser `LIMIT` pour spécifier le nombre maximum de résultats que l'ont souhaite obtenir. Cette clause est souvent associé à un `OFFSET`, c'est-à-dire effectuer un décalage sur le jeu de résultat.

Informatique 4.2.

En MySQL la commande suivante affiche les résultats de 6 à 15 d'une table.
Le premier nombre est l'`OFFSET` tandis que le suivant est la limite.

```
SELECT *
FROM table
LIMIT 10 OFFSET 5;
```

Il faut faut faire deux remarques au sujet de `LIMIT` :

1. l'utilisation de `LIMIT` n'a aucune influence sur les performance de la requête car la requête va permettre de récupérer toutes les lignes (donc temps d'exécution identique) puis seulement les résultats définit par `LIMIT` et `OFFSET` seront retournés.
2. la bonne pratique lorsque l'ont utilise `LIMIT` consiste à utiliser également la clause `ORDER BY` que l'on verra plus loin.

4.4.2 La jointure symétrique

Définition 4.5 (Jointure symétrique)**.**

Soient $R(S)$ et $R'(S')$ deux relations de schémas disjoints, $(A, A') \in S \times S'$ tel que $\text{dom}(A) = \text{dom}(A')$.
On appelle jointure symétrique de $R(S)$ et $R(S')$ selon (A, A'), la relation notée $R \bowtie_{A=A'} R'$, ou $R[A = A']R'$, définie par :

$$R \bowtie_{A=A'} R' = \sigma_{A=A'}(R \times R').$$

— Il s'agit donc d'une sélection dans la relation $R \times R'$ suivant la condition $A = A'$.

— Autrement dit on réunit deux tables, en recollant une valeur de l'une avec une valeur de l'autre à condition que les attributs A et A' soient les mêmes.

car		
Marque	Modèle	Couleur
Porsche	918 - Spyder	Gris
Porsche	911 - GT3 Cup	Bleu
Ford	Mustang GT	Bleu

spec		
Type	Puissance	Poids
918 - Spyder	887	1675
Mustang GT	412	1580

$car \times spec$					
Marque	Modèle	Couleur	Type	Puissance	Poids
Porsche	918 - Spyder	Gris	918 - Spyder	887	1675
Porsche	918 - Spyder	Gris	Mustang GT	412	1580
Porsche	911 - GT3 Cup	Bleu	918 - Spyder	887	1675
Porsche	911 - GT3 Cup	Bleu	Mustang GT	412	1580
Ford	Mustang GT	Bleu	918 - Spyder	887	1675
Ford	Mustang GT	Bleu	Mustang GT	412	1580

$car \bowtie_{\text{Modèle = Type}} spec$					
Marque	Modèle	Couleur	Type	Puissance	Poids
Porsche	918 - Spyder	Gris	918 - Spyder	887	1675
Ford	Mustang GT	Bleu	Mustang GT	412	1580

On ne garde que les lignes telles que Modèle = Type.

Informatique 4.3.

En SQL :

```sql
SELECT * FROM car JOIN spec ON Modele = Type
```

Remarque 4.1.

1. Dans notre exemple **Type** est une clé primaire pour la table *spec*, donc nous pouvions prévoir que nous n'allions récupérer qu'un **Type** par **Modèle**.
   Mais on peut faire des jointures sur des attributs qui ne sont pas des clés primaires, et on obtiendra alors des répétitions.
2. On peut élargir la notion de jointure à plusieurs conditions :
$$R \bowtie_{\text{A=A',B=B'}} R'.$$
3. Les SGBD n'effectuent pas la jointure $R \bowtie_{A=A'} R' = \sigma_{A=A'}(R \times R')$ suivant cette formule. La taille d'une BDD étant trop importante. Il utilise des algorithmes en $O(n \log n)$.

4.5 Fonctions d'agrégation

Définition 4.6 (Fonctions d'agrégation)**.**

Les fonctions d'agrégation permettent d'effectuer des opérations statistiques sur un ensemble d'enregistrements.

- AVG() pour calculer la moyenne sur un ensemble d'enregistrements.
- COUNT() pour compter le nombre d'enregistrement sur une table ou une colonne distincte.
- MAX() (resp. MIN()) pour récupérer la valeur maximum (resp.minimum) d'une colonne sur un ensemble de lignes. Cela s'applique à la fois pour des données numériques ou alpha-numériques.
- SUM() pour calculer la somme sur un ensemble d'enregistrements.

On peut utiliser les fonctions d'agrégation sur toute une colonne, ou après avoir effectué des regroupements en sous-groupes que l'on appelle agrégats grâce à `GROUP BY`.

Après un `GROUP BY` on peut souhaiter ajouter une condition à vérifier grâce à `HAVING`.
Prenons la table suivante comme exemple :

Voitures			
Marque	Modèle	Puissance	Poids
Porsche	918 - Spyder	887	1675
Porsche	911 - GT3 Cup	435	1395
Ford	Mustang GT	412	1580
Porsche	550 - Spyder	110	760
Ferrari	F50	520	1230
Porsche	918 - Spyder	900	1675
Ferrari	288 - GTO	400	1224

1. Puissance minimale des voitures de la liste :

```
SELECT MIN(Puissance) FROM voitures
```

2. Poids maximal :

```
SELECT MAX(Poids) FROM voitures
```

3. Somme des poids :

```
SELECT SUM(Poids) FROM voitures
```

4. Puissance moyenne :

```sql
SELECT AVG(Puissance) FROM voitures
```

5. Compter les voitures :

```sql
SELECT COUNT(*) FROM voitures
```

6. Compter les Porsche :

```sql
SELECT COUNT(*) FROM voitures WHERE Marque = "Porsche"
```

7. Puissance moyenne par marque :

```sql
SELECT Marque, AVG(Puissance) FROM voitures GROUP BY Marque
```

8. Les marques dont la puissance moyenne est supérieure à 450 :

```sql
SELECT Marque, AVG(Puissance)
FROM voitures
GROUP BY Marque
HAVING AVG(Puissance) >= 450
```

9. Les marques dont les modèles de moins de 1300 Kg ont une puissance moyenne supérieure à 450 :

```sql
SELECT Marque, AVG(Puissance)
FROM voitures
WHERE Poids < 1300
GROUP BY Marque
HAVING AVG(Puissance) >= 450
```

Si $R(S)$ est une relation, $A \in S$ un attribut, et f une fonction d'agrégation, on note $f(R.A)$ le résultat obtenu en appliquant la fonction f aux valeurs de R pour l'attribut A.

Si $R(S)$ est une relation, $A_1, \ldots, A_n, B_1, \ldots, B_m \in S$ des attributs, et $f_1, \ldots, f_m$ des fonctions d'agrégation, on note :

$$_{A_1,\ldots,A_n}\gamma_{f_1(B_1),\ldots,f_m(B_m)}(R),$$

la relation obtenue par :

1. regroupement des valeurs de R qui sont identiques sur les attributs $A_1, \ldots, A_n$;
2. définition des attributs $f_i(B_i)$ pour ces valeurs regroupées par application de la fonction f_i sur chaque agrégat sur l'attribut B_i.

En résumé :

— En SQL pour obtenir $\gamma_{f(A)}(R)$, où f est une fonction d'agrégation :

```sql
SELECT f(A) FROM R
```

— Pour obtenir $_{A_1,\ldots,A_n}\gamma_{f_1(B_1),\ldots,f_m(B_m)}$:

```sql
SELECT A1, . . . , An , f1(B1) , . . . , fm(Bm) FROM R
GROUP BY A1 , . . .  , An
```

— Pour obtenir $\sigma_{C2} \circ_{A_1,\ldots,A_n} \gamma_{f_1(B_1),\ldots,f_m(B_m)} \circ \sigma_{C1}$:

```sql
SELECT A1, . . . , An , f1(B1) , . . . , fm(Bm) FROM R
WHERE C1
GROUP BY A1 , . . .  , An
HAVING C2
```

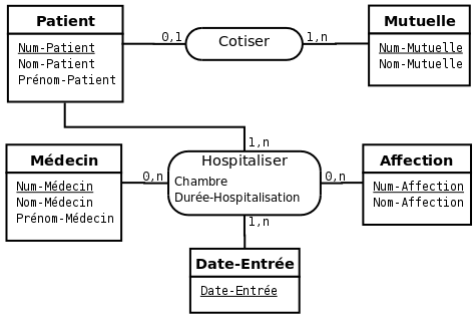
Utiliser `GROUP BY` sur plusieurs attributs permet de différencier les enregistrement qui coïncident sur `A1`, ..., `An-1`

Remarque 4.2.

Les deux sélections σ_{C1} et σ_{C2} ont des rôles différents, la première limite les valeurs prises en considération par l'agrégation, alors que la seconde s'effectue sur les agrégats après application des fonctions.

4.6 Exercices

Exercice 4.1. Proposer les tables de la base de donnée qui représente le schéma suivant :



Exercice 4.2. Une base de données qui recense des informations sur des oeuvres cinématographiques est composée de trois tables :

acteurs (id : `int`, prenom : `text`, nom : `text`)
castings (id_acteur : `int`, id_films : `int`, role : `text`)
films (id : `int`, titre : `text`, annee : `int`, id_genre : `int`)
genres (id : `int`, id_films : `int`, genre : `text`)
films_realisateurs (id_real : `int`, id_films : `int`)
realisateurs (id : `int`, prenom : `text`, nom : `text`)

1. Lister les prénoms et noms de tous les acteurs jouant dans le film Matrix .

Solution.

2. Déterminer les réalisateurs des films du genre « Guerre » tournés une année multiple de 4. La requête devra retourner les prénoms, noms réalisateurs, titre du films et année.

Solution.

3. Déterminer les réalisateurs qui ont tourné plus de 200 films. La requête retournera leurs prénom, nom et nombre de films par ordre décroissant.

Solution.

4. Déterminer les acteurs qui ont joué 5 rôles ou plus dans le même films au cours de l'année 2010. Attention, la base de données peut contenir des erreurs et un acteur peut être rentré plusieurs fois avec le même rôle pour un films. Ici, on souhaite avoir des rôles distincts. La requête doit retourner le prénom et le nom des acteurs, le nom du films, le nombre de rôles distincts joués dans le films.

Solution.

4.7 Corrections

Corrigé 4.1

1. Patient(Num-Patient, Nom-Patient, Prénom-Patient, Num-Mutuelle) ;
2. Mutuelle(Num-Mutuelle, Nom-Mutuelle) ;
3. Médecin(Num-Médecin, Nom-Médecin, Prénom-Médecin) ;
4. Affection(Num-Affection, Nom-Affection) ;
5. Hospitaliser(Num-Patient, Num-Affection, Num-Médecin, Date-Entrée, Chambre, Durée-Hospitalisation).

Corrigé 4.2 Les requêtes :

1.

```
SELECT acteurs.nom, acteurs.prenom FROM acteurs WHERE acteurs.id in (
SELECT castings.id_acteur FROM catsings WHERE castings.id_film = (
SELECT films.id FROM films WHERE films.titre LIKE "Matrix"))
```

2.

3.

4.

```
SELECT nom,prenom,titre, count(distinct role)
FROM catsing JOIN acteurs ON id_acteur=acteurs.id JOIN films ON id_film=films.id WHERE annee = 2010
GROUP BY id_film , id_acteur
HAVING count(distinct role) > 4
```