Bases de données

Fabrice Lembrez - PSI*

Lycée Pierre de Fermat

Plan

Intérêt des bases de données

Intérêt des bases de données

Grosses quantités de données

Critères d'organisation variables

$\neq$ listes (=un type de parcours privilégié)

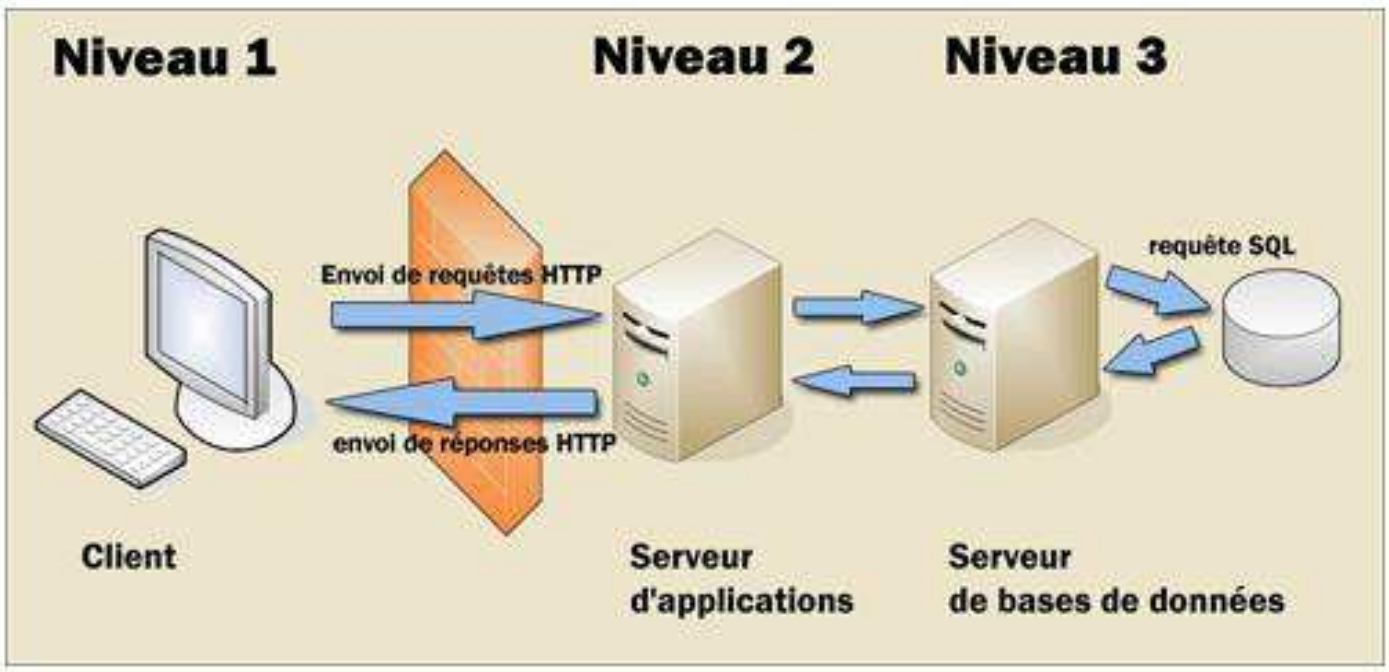
croisements d'informations de natures différentes

Exemples :

- lors d'un rappel, quels sont tous les clients qui ont acheté un produit du lot 18181818XX dans les supermarchés de l'enseigne Prizinique ?
- quels sont les départements de France qui ont vu le nombre d'emplois augmenter le plus entre les dates D et D' dans le secteur médical ?
- ...

Plan

Interroger une base de données



Architecture 3-tiers (tier = couche, étage). Par exemple

- un navigateur communique (par requêtes HTML, par internet)
- avec un serveur de gestion (codé par exemple en PHP)
- qui se charge d'interroger une base de données : le langage est

$$\boxed{\text{SQL} = \text{Structured Query Language.}}$$

Langage normalisé utilisé très largement par les différents logiciels de gestion.

Plan

Rôle du SGBD

Concrètement les données sont organisées sous forme de tables. Il y a de nombreuses fonctions à réaliser.

Organiser la structuration des données (quelles tables, quels liens)

Rôle du SGBD

Concrètement les données sont organisées sous forme de tables. Il y a de nombreuses fonctions à réaliser.

Organiser la structuration des données (quelles tables, quels liens)

Mettre à jour les données (lignes dans les tables)

Rôle du SGBD

Concrètement les données sont organisées sous forme de tables. Il y a de nombreuses fonctions à réaliser.

Organiser la structuration des données (quelles tables, quels liens)

Mettre à jour les données (lignes dans les tables)

Extraire des données en réponse à une requête complexe

Rôle du SGBD

Concrètement les données sont organisées sous forme de tables. Il y a de nombreuses fonctions à réaliser.

Organiser la structuration des données (quelles tables, quels liens)

Mettre à jour les données (lignes dans les tables)

Extraire des données en réponse à une requête complexe

Gérer les autorisations d'accès lors du remplissage

Rôle du SGBD

Concrètement les données sont organisées sous forme de tables. Il y a de nombreuses fonctions à réaliser.

Organiser la structuration des données (quelles tables, quels liens)

Mettre à jour les données (lignes dans les tables)

Extraire des données en réponse à une requête complexe

Gérer les autorisations d'accès lors du remplissage

Gérer les accès concurrents lors du remplissage

Rôle du SGBD

Concrètement les données sont organisées sous forme de tables. Il y a de nombreuses fonctions à réaliser.

Organiser la structuration des données (quelles tables, quels liens)

Mettre à jour les données (lignes dans les tables)

Extraire des données en réponse à une requête complexe

Gérer les autorisations d'accès lors du remplissage

Gérer les accès concurrents lors du remplissage

En prépa on parle un peu du premier point, surtout du troisième

Plan

Exemple : plans de vol pour Eurocontrol

Une base très simple (Centrale 2016), formée de deux tables

Table `plan_vol`

`id_vol`	`depart`	`arrivee`	`jour`	`heure`	`niveau`
AF1204	CDG	FCO	2016-05-02	07:35	300
AF1205	FCO	CDG	2016-05-02	10:25	300
AF1504	CDG	FCO	2016-05-02	10:05	310
AF1505	FCO	CDG	2016-05-02	13:00	310

Table `aeroport`

`id_aero`	`nom`	`ville`	`pays`
CDG	Charles-de-Gaulle	Paris	France
ORY	Orly	Paris	France
MRS	Marignane	Marseille	France
FCO	Fiumcino	Rome	Italie

Exemple : plans de vol pour Eurocontrol

Une base très simple (Centrale 2016), formée de deux tables

Table `plan_vol`

`id_vol`	`depart`	`arrivee`	`jour`	`heure`	`niveau`
AF1204	CDG	FCO	2016-05-02	07:35	300
AF1205	FCO	CDG	2016-05-02	10:25	300
AF1504	CDG	FCO	2016-05-02	10:05	310
AF1505	FCO	CDG	2016-05-02	13:00	310

- `id_vol` : numéro du vol (chaîne de caractères) ;
- `depart` : code de l'aéroport de départ (chaîne de caractères) ;
- `arrivee` : code de l'aéroport d'arrivée (chaîne de caractères) ;
- `jour` : jour du vol (chaîne au format `aaaa-mm-jj`) ;
- `heure` : heure de décollage souhaitée (chaîne au format `hh:mi`) ;
- `niveau` : niveau de vol souhaité (entier).

Exemple : plans de vol pour Eurocontrol

Une base très simple (Centrale 2016), formée de deux tables

Table `aeroport`

`id_aero`	`nom`	`ville`	`pays`
CDG	Charles-de-Gaulle	Paris	France
ORY	Orly	Paris	France
MRS	Marignane	Marseille	France
FCO	Fiumcino	Rome	Italie

- `id_aero` : code de l'aéroport (chaîne de caractères) ;
- `nom` : son nom (chaîne de caractères) ;
- `ville` : principale ville desservie (chaîne de caractères) ;
- `pays` : pays dans lequel se situe l'aéroport (chaîne de caractères).

Plan

Structure et contenu d'une table

- Description de la structure (ou schéma)
  - liste des **attributs = colonnes** (parfois : "champs")
  - chacun de ces attributs a un "domaine" dans lequel il prend ses valeurs
    ex : entier/chaîne/flottant... parfois c'est plus précis (3 lettres)
- Description du contenu : les **enregistrements = lignes** de la table

En réalité il existe une définition formelle de la notion de table. C'est une définition de type mathématique, à l'aide des notions de fonctions et d'ensembles. Vous n'avez pas à le savoir, mais c'est cela qui explique la présence d'un double vocabulaire formel/concret.

Vocabulaire

Formulation concrète	Description formelle
Une table	
Une ligne	
Structure de la table (liste des colonnes et types)	
Colonne	
Type de donnée de la colonne	

Vocabulaire

Formulation concrète	Description formelle
Une table	Une relation
Une ligne	Un enregistrement
Structure de la table (liste des colonnes et types)	
Colonne	
Type de donnée de la colonne	

Vocabulaire

Formulation concrète	Description formelle
Une table	Une relation
Une ligne	Un enregistrement
Structure de la table (liste des colonnes et types)	
Colonne	Attribut A
Type de donnée de la colonne	

Vocabulaire

Formulation concrète	Description formelle
Une table	Une relation
Une ligne	Un enregistrement
Structure de la table (liste des colonnes et types)	
Colonne	Attribut A
Type de donnée de la colonne	Domaine de A (ex. : "chaîne de caractères")

Vocabulaire

Formulation concrète	Description formelle
Une table	Une relation
Une ligne	Un enregistrement
Structure de la table (liste des colonnes et types)	Schéma relationnel ((A1,D(A1)),...(An,D(An)))
Colonne	Attribut A
Type de donnée de la colonne	Domaine de A (ex. : "chaîne de caractères")

Plan

Clefs, clefs primaires

Déf - Clef

Clef = ensemble d'attributs A1, . . . , Ak

1. permettant d'identifier un enregistrement
   e.A1=f.A1 et . . . et e.Ak=f.Ak $\Rightarrow$ e=f
2. minimal : si on enlève un attribut le (1) ne marche plus

Clefs, clefs primaires

Déf - Clef

Clef = ensemble d'attributs A1, . . . , Ak

1. permettant d'identifier un enregistrement
   e.A1=f.A1 et . . . et e.Ak=f.Ak $\Rightarrow$ e=f
2. minimal : si on enlève un attribut le (1) ne marche plus

Clef primaire d'une table = choix de clef privilégiée.

- des codes d'identification comme le numéro de vol `id_vol` ou le code d'aéroport `id_aero` ont été introduits explicitement dans ce but
- dans la classe, selon les années soit (nom), soit (nom,prenom)
- et à l'échelle nationale (nom,prenom) n'est plus une clef
- mais le numéro INSEE est sensé être univoque (pour qui en a un)

S'il n'y a pas une clef primaire "naturelle", on introduit un attribut ad hoc, l'identifiant (id).

Plan

Trois opérations de base

Projection = choix d'attributs/colonnes

Résultat : nouvelle relation, schéma différent.

Trois opérations de base

Projection = choix d'attributs/colonnes

Résultat : nouvelle relation, schéma différent.

Sélection = d'enregistrements vérifiant une condition

La condition porte sur des attributs.

Résultat : même schéma, moins de lignes.

Trois opérations de base

Projection = choix d'attributs/colonnes

Résultat : nouvelle relation, schéma différent.

Sélection = d'enregistrements vérifiant une condition

La condition porte sur des attributs.

Résultat : même schéma, moins de lignes.

Renommage = changement de nom d'un attribut

Résultat : juste un en-tête de colonne modifié.

Requêtes SQL correspondantes

Syntaxe de base

SELECT listecolonnes FROM nomtable WHERE conditionlogique

Remarque : les mots clef de SQL sont en majuscule uniquement pour une question de lisibilité.

Requêtes SQL correspondantes

Syntaxe de base

SELECT listecolonnes FROM nomtable WHERE conditionlogique

Remarque : les mots clef de SQL sont en majuscule uniquement pour une question de lisibilité.
Exemple

SELECT nom,ville FROM aeroport WHERE id_aero = "CDG"

nom	ville
Charles-de-Gaulle	Paris

Requêtes SQL correspondantes

Syntaxe de base

SELECT listecolonnes FROM nomtable WHERE conditionlogique

Remarque : les mots clef de SQL sont en majuscule uniquement pour une question de lisibilité.

Exemple

```
SELECT id_vol FROM plan_vol
    WHERE jour < "2016-05-02" AND jour > "2016-04-25"
```

id_vol
AF1204
AF1205
AF1504
AF1505

Eléments de syntaxe pour la sélection (WHERE)

Opérateurs logiques SQL

Test d'égalité $=$ simple, test de non égalité $<>$.
Les autres opérateurs de comparaison sont comme pour Python :
$>, <, >=, <=$.
Et on a également les opérateurs logiques AND / OR / NOT.

Le programme semble limiter les choses ainsi, je cite pour mémoire
- le test d'appartenance IN
- le test IS NULL quand l'enregistrement n'est pas rempli
- une comparaison souple de chaîne avec LIKE...

Eléments de syntaxe pour la projection (SELECT)

SELECT = projection !!!

Ca choisit les colonnes, pour faire une sélection (de lignes) c'est WHERE...

- Après SELECT mettre la liste des colonnes : col1,col2,col3
- Si on veut tous les attributs : SELECT $*$
- On peut au passage faire des calculs simples sur les colonnes

Eléments de syntaxe pour la projection (SELECT)

SELECT = projection !!!

Ca choisit les colonnes, pour faire une sélection (de lignes) c'est WHERE...

- Après SELECT mettre la liste des colonnes : col1,col2,col3
- Si on veut tous les attributs : SELECT $*$
- On peut au passage faire des calculs simples sur les colonnes

Exemples

SELECT * FROM aeroport

SELECT heure,depart,id_vol FROM plan_vol WHERE niveau=300

SELECT numeroFacture, prixHT+TVA FROM facture

Eléments de syntaxe pour le renommage (AS)

On peut ajouter le renommage des colonnes lors de la projection

SELECT depart AS codeAero FROM plan_vol
WHERE id_vol="AF1204"

SELECT numeroFacture, prixHT+TVA AS montant FROM facture

Eléments de syntaxe pour le renommage (AS)

On peut ajouter le renommage des colonnes lors de la projection

SELECT depart AS codeAero FROM plan_vol
WHERE id_vol="AF1204"

SELECT numeroFacture, prixHT+TVA AS montant FROM facture

En théorie au moins, le renommage ne peut pas être utilisé dans le WHERE car le WHERE se fait avant le SELECT

SELECT numeroFacture, prixHT+TVA AS montant FROM facture
WHERE prixHT+TVA>8000
(l'alias ne sert que pour l'affichage)

Plan

Tris et limites

Ils se mettent à la fin de la commande SQL et s'exécutent à la toute fin.

Trier le résultat

ORDER BY nomcol pour l'ordre croissant

ORDER BY nomcol DESC (décroissant)

On peut utiliser un couple : nom,prenom par exemple

Limiter le nombre d'enregistrements

LIMIT N,p

LIMIT N OFFSET p (c'est pareil)

Que les p enregistrements à partir du numéro N.

Exemple LIMIT 0,10 donne les dix premiers, LIMIT 10 OFFSET 30 les 30 suivants etc...

Tris et limites

Ils se mettent à la fin de la commande SQL et s'exécutent à la toute fin.

Trier le résultat

ORDER BY nomcol pour l'ordre croissant

ORDER BY nomcol DESC (décroissant)

On peut utiliser un couple : nom,prenom par exemple

Limiter le nombre d'enregistrements

LIMIT N,p

LIMIT N OFFSET p (c'est pareil)

Que les p enregistrements à partir du numéro N.

Exemple LIMIT 0,10 donne les dix premiers, LIMIT 10 OFFSET 30 les 30 suivants etc...

SELECT note,nom FROM DS ORDER BY note DESC, nom

Tris et limites

Ils se mettent à la fin de la commande SQL et s'exécutent à la toute fin.

Trier le résultat

ORDER BY nomcol pour l'ordre croissant

ORDER BY nomcol DESC (décroissant)

On peut utiliser un couple : nom,prenom par exemple

Limiter le nombre d'enregistrements

LIMIT N,p

LIMIT N OFFSET p (c'est pareil)

Que les p enregistrements à partir du numéro N.

Exemple LIMIT 0,10 donne les dix premiers, LIMIT 10 OFFSET 30 les 30 suivants etc...

```
SELECT score,nom FROM championnat
    ORDER BY score DESC, LIMIT 3,1
```

Tris et limites

Ils se mettent à la fin de la commande SQL et s'exécutent à la toute fin.

Trier le résultat

ORDER BY nomcol pour l'ordre croissant

ORDER BY nomcol DESC (décroissant)

On peut utiliser un couple : nom,prenom par exemple

Limiter le nombre d'enregistrements

LIMIT N,p

LIMIT N OFFSET p (c'est pareil)

Que les p enregistrements à partir du numéro N.

Exemple LIMIT 0,10 donne les dix premiers, LIMIT 10 OFFSET 30 les 30 suivants etc...

```
SELECT numeroFacture, prixHT+TVA AS montant
    FROM facture ORDER BY montant DESC LIMIT 0,10
```

Ici utiliser le nom "montant" est légitime dans le ORDER BY.

Doublons

Quand on précise **SELECT DISTINCT** on filtre les doublons, en supprmant les lignes identiques.

```
SELECT DISTINCT niveau FROM plan_vol
```

```
SELECT DISTINCT titre,auteur FROM bibliotheque
    ORDER BY auteur,titre
```

Plan

Fonctions d'agrégation et regroupements

Déf - Fonctions d'agrégation

Ce sont les fonctions MIN, MAX, SUM, AVG (moyenne), COUNT

On peut les employer sur toute la table :

SELECT COUNT(*) FROM aeroport

SELECT MAX(niveau) FROM plan_vol

SELECT AVG(prixHT+TVA) FROM facture

Fonctions d'agrégation et regroupements

Mais l'intérêt principal est d'effectuer des **regroupements** par valeurs de certains attributs. Commençons par quelques exemples

- compter le nombre de vols par niveau

  SELECT niveau,COUNT(*) FROM plan_vol GROUP BY niveau

- compter le nombre de vols par jour au niveau 300

Fonctions d'agrégation et regroupements

Mais l'intérêt principal est d'effectuer des **regroupements** par valeurs de certains attributs. Commençons par quelques exemples

- compter le nombre de vols par niveau

```
SELECT niveau,COUNT(*) FROM plan_vol GROUP BY niveau
```

- compter le nombre de vols par jour au niveau 300

```
SELECT jour,COUNT(*) FROM plan_vol
    WHERE niveau=300 GROUP BY jour
```

- chercher les niveaux de vol extrêmes par jour

Fonctions d'agrégation et regroupements

Mais l'intérêt principal est d'effectuer des **regroupements** par valeurs de certains attributs. Commençons par quelques exemples

- compter le nombre de vols par niveau

```
SELECT niveau,COUNT(*) FROM plan_vol GROUP BY niveau
```

- compter le nombre de vols par jour au niveau 300

```
SELECT jour,COUNT(*) FROM plan_vol
    WHERE niveau=300 GROUP BY jour
```

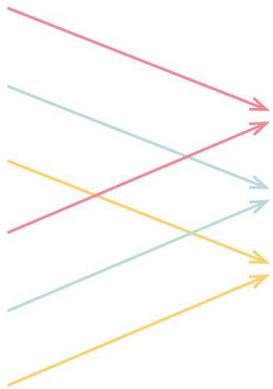
- chercher les niveaux de vol extrêmes par jour

```
SELECT jour,MIN(vol),MAX(vol) FROM plan_vol
    GROUP BY jour
```

Sur ce dernier exemple on ne peut pas demander d'afficher d'autre colonne : pourquoi ?

Fonctions d'agrégation et regroupements

title	genre	qty
book 1	adventure	4
book 2	fantasy	5
book 3	romance	2
book 4	adventure	3
book 5	fantasy	3
book 6	romance	1



genre	total
adventure	7
fantasy	8
romance	3

Trois types d'attributs

- ceux qu'on obtient par une fonction d'agrégat (ex : total)
- ceux qui sont identiques au sein d'un groupe peuvent rester (ex : genre)
- ceux qui sont différents ne doivent plus figurer (ex : title)

La dernière catégorie ne renverra pas d'erreur mais une valeur arbitraire.

Fonctions d'agrégation et regroupements

Syntaxe complète avec regroupement

SELECT attributsRegr, f(attributs) FROM table
WHERE conditionPreRegr
GROUP BY attributsRegr
HAVING conditionPostRegr
ORDER BY ... LIMIT ...

WHERE : condition AVANT le regroupement, pas d'agrégat
HAVING : condition APRES le regroupement, utilisant les agrégats

Ordre d'exécution (qui justifie la différence entre WHERE et HAVING)
- d'abord la sélection (WHERE)
- puis le regroupement (GROUP BY)
- le filtrage après regroupement (HAVING)
- la projection (SELECT) avec le renommage (AS)
- le tri (ORDER BY) : lui seul peut utiliser les alias

Fonctions d'agrégation et regroupements

Syntaxe complète avec regroupement

SELECT attributsRegr, f(attributs) FROM table
WHERE conditionPreRegr
GROUP BY attributsRegr
HAVING conditionPostRegr
ORDER BY ... LIMIT ...

WHERE : condition AVANT le regroupement, pas d'agrégat
HAVING : condition APRES le regroupement, utilisant les agrégats

Exemple montant moyen des factures par carte, jour par jour.

Fonctions d'agrégation et regroupements

Syntaxe complète avec regroupement

SELECT attributsRegr, f(attributs) FROM table
WHERE conditionPreRegr
GROUP BY attributsRegr
HAVING conditionPostRegr
ORDER BY ... LIMIT ...

WHERE : condition AVANT le regroupement, pas d'agrégat
HAVING : condition APRES le regroupement, utilisant les agrégats

Exemple montant moyen des factures par carte, jour par jour.

SELECT date,AVG(montant) FROM factures
WHERE reglement="par carte"
GROUP BY date

Fonctions d'agrégation et regroupements

Syntaxe complète avec regroupement

```
SELECT attributsRegr, f(attributs) FROM table
    WHERE conditionPreRegr
    GROUP BY attributsRegr
    HAVING conditionPostRegr
    ORDER BY ... LIMIT ...
```

WHERE : condition AVANT le regroupement, pas d'agrégat
HAVING : condition APRES le regroupement, utilisant les agrégats

Exemple montant moyen des factures par carte, jour par jour.
On peut affiner en triant les résultats

```
SELECT date,AVG(montant) AS montantmoyen FROM factures
    WHERE reglement="par carte"
    GROUP BY date
    ORDER BY montantmoyen
```

Fonctions d'agrégation et regroupements

Syntaxe complète avec regroupement

SELECT attributsRegr, f(attributs) FROM table
WHERE conditionPreRegr
GROUP BY attributsRegr
HAVING conditionPostRegr
ORDER BY ... LIMIT ...

WHERE : condition AVANT le regroupement, pas d'agrégat
HAVING : condition APRES le regroupement, utilisant les agrégats

Exemple montant moyen des factures par carte, jour par jour.
On peut ne garder que les journées où ce montant moyen dépasse 1000

SELECT date,AVG(montant) AS montantmoyen FROM factures
WHERE reglement="par carte"
GROUP BY date
HAVING AVG(montant)>1000

Fonctions d'agrégation et regroupements

Syntaxe complète avec regroupement

SELECT attributsRegr, f(attributs) FROM table
WHERE conditionPreRegr
GROUP BY attributsRegr
HAVING conditionPostRegr
ORDER BY ... LIMIT ...

WHERE : condition AVANT le regroupement, pas d'agrégat
HAVING : condition APRES le regroupement, utilisant les agrégats

Ordre d'exécution (qui justifie la différence entre WHERE et HAVING)
- d'abord la sélection (WHERE)
- puis le regroupement (GROUP BY)
- le filtrage après regroupement (HAVING)
- la projection (SELECT) avec le renommage (AS)
- le tri (ORDER BY) : lui seul peut utiliser les alias

Plan

Requêtes imbriquées

On peut utiliser des sous-requêtes dans une requête. On les met alors entre parenthèses.

Requêtes imbriquées

On peut utiliser des sous-requêtes dans une requête. On les met alors entre parenthèses.

Ex1 : trouver tous les élèves qui ont eu plus que la moyenne

```
SELECT nomEleve,note FROM listenotes
    WHERE note> (SELECT AVG(note) FROM listenotes)
```

Dans ce cas la sous-requête produit un nombre.

Requêtes imbriquées

On peut utiliser des sous-requêtes dans une requête. On les met alors entre parenthèses.

Exemple 2 : regrouper toutes les factures jour par jour, puis trouver le meilleur résultat journalier

```
SELECT MAX(resultatJournalier) FROM
    (SELECT SUM(montant) AS resultatJournalier
        FROM facture GROUP BY date)
```

Dans ce cas la sous-requête produit une séquence d'enregistrements.

Plan

Opérations ensemblistes

Déf - Opérations ensemblistes

Elles fonctionnent sur des **relations ayant des schémas identiques.**

UNION, INTERSECT = union, intersection

EXCEPT = différence ensembliste

Opérations ensemblistes

Déf - Opérations ensemblistes

Elles fonctionnent sur des **relations ayant des schémas identiques.**

UNION, INTERSECT = union, intersection

EXCEPT = différence ensembliste

Exemple : création d'un annuaire profs+élèves

```
SELECT nom FROM eleves UNION SELECT nom FROM professeurs
```

Et au cas où les en-têtes seraient différents :

```
SELECT nom_e AS nom FROM eleves
    UNION
SELECT nom_p AS nom FROM professeurs
    ORDER BY nom
```

Opérations ensemblistes

Déf - Opérations ensemblistes

Elles fonctionnent sur des **relations ayant des schémas identiques.**

UNION, INTERSECT = union, intersection

EXCEPT = différence ensembliste

Exemple : liste des PSI non internes par croisement des fichiers intendance et scolarité.

```
SELECT INE,nom FROM scolarite WHERE classe="PSIE"
    EXCEPT
SELECT INE,nom FROM internat
```

Plan

Produit cartésien

Déf - Produit cartésien

Il donne toutes les associations possibles de données de deux tables. Le produit cartésien d'une table de p_1 enregistrements avec une table de p_2 enregistrements contient $p_1 p_2$ enregistrements. Si le nombre de colonnes était ℓ_1 et ℓ_2, il y en a $\ell_1 + \ell_2$ dans le résultat.

Exemple : tous les couples candidat/interrogateur imaginables

```
SELECT candidat.nom,interrogateur.nom FROM
candidat,interrogateur
```

Remarque : on utilise la syntaxe table.attribut pour lever l'ambigüité.

Exemple : retour sur Centrale 2016

Que donne le produit cartésien de ces deux tables ?

Table `plan_vol`

id_vol	depart	arrivee	jour	heure	niveau
AF1204	CDG	FCO	2016-05-02	07:35	300
AF1205	FCO	CDG	2016-05-02	10:25	300
AF1504	CDG	FCO	2016-05-02	10:05	310
AF1505	FCO	CDG	2016-05-02	13:00	310

Table `aeroport`

id_aero	nom	ville	pays
CDG	Charles-de-Gaulle	Paris	France
ORY	Orly	Paris	France
MRS	Marignane	Marseille	France
FCO	Fiumcino	Rome	Italie

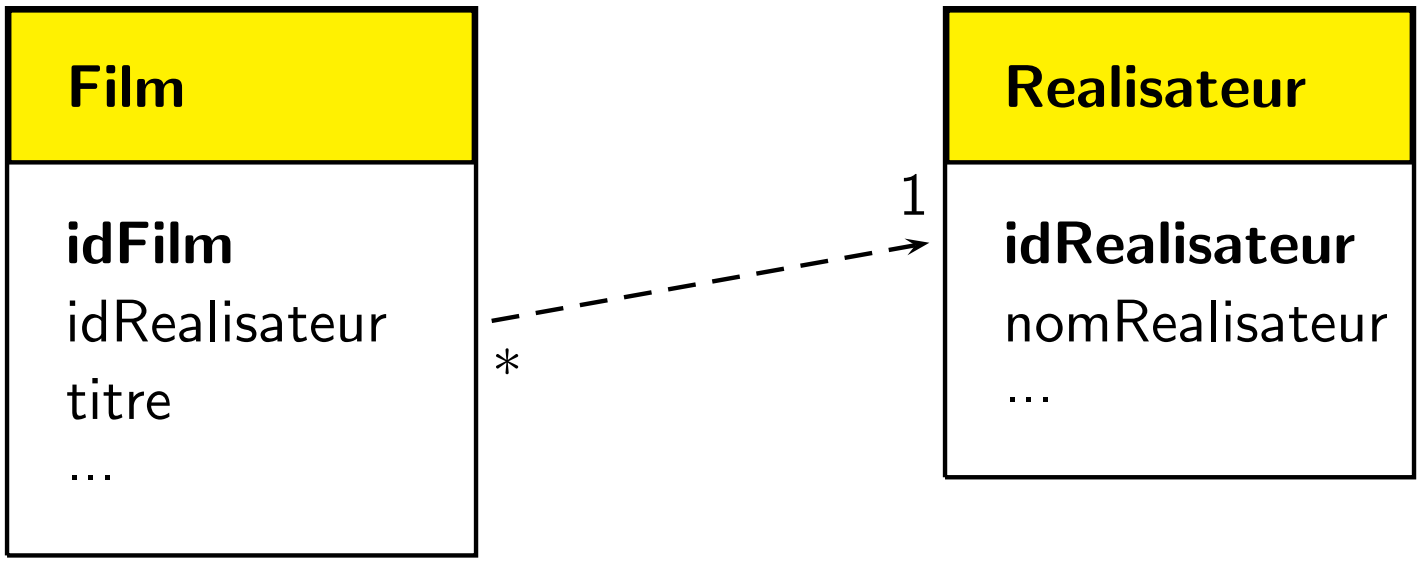
Il y a parfois le cas intéressant où le code de départ (ou celui d'arrivée) vaut `id_aero` : cela nous amène à l'idée de jointure.

Plan

Clefs étrangères

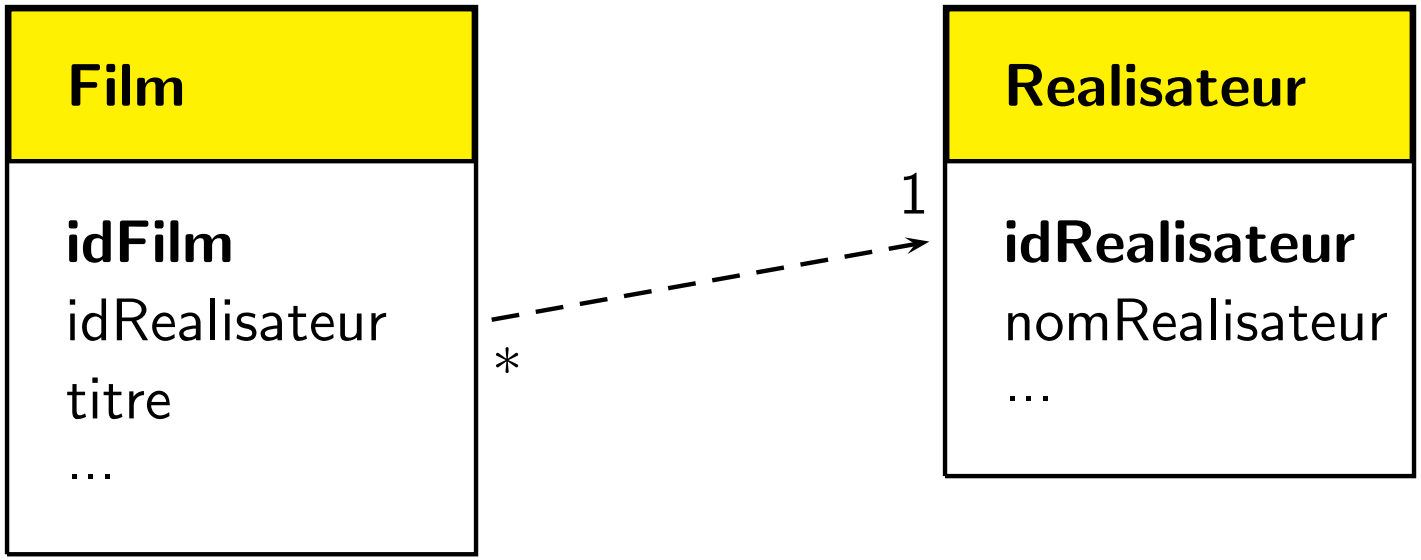
Déf - Clef étrangère

Un (ou plusieurs) attribut de la table R contient des valeurs désignant des valeurs de la clef primaire d'une autre table R'.



Ici les attributs ont même nom, ce n'est pas obligé.

Modèle entité-association



Notation **1** — $*$ (plusieurs à un) : une valeur de idRealisateur est unique dans R' (clef primaire) mais peut apparaître plusieurs fois dans R.

Dans la conception de la base on a ainsi considéré qu'il y avait une "liaison privilégiée" entre certaines tables. On a ainsi préparé le terrain pour certaines opérations ultérieures.

Modèle entité-association, diagrammes

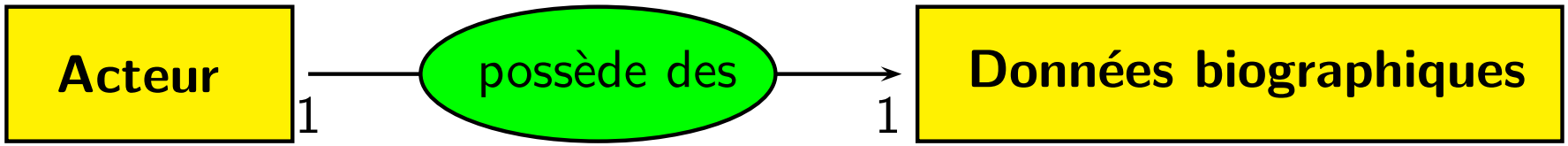
Déf - Modèle entité-association

Ce modèle représente la structure logique de la base de données, qui doit être imaginée avant son implémentation concrète. Les entités sont les données brutes et les associations sont les liens privilégiés qu'on veut établir entre ces données.
Il a des associations $\mathbf{1}$ — $*$ (plusieurs à un), et aussi $\mathbf{1}$ — $\mathbf{1}$ (un à un), $*$ — $*$ (plusieurs à plusieurs).

Modèle entité-association, diagrammes

Déf - Modèle entité-association

Ce modèle représente la structure logique de la base de données, qui doit être imaginée avant son implémentation concrète. Les entités sont les données brutes et les associations sont les liens privilégiés qu'on veut établir entre ces données.
Il a des associations **1** — $*$ (plusieurs à un), et aussi **1** — **1** (un à un), $*$ — $*$ (plusieurs à plusieurs).



Comme pour **1** — $*$, il suffit d'utiliser une clef étrangère dans une table renvoyant à la clef primaire de l'autre table.

Modèle entité-association, diagrammes

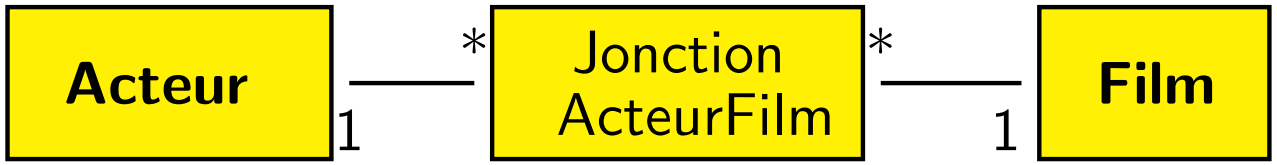
Déf - Modèle entité-association

Ce modèle représente la structure logique de la base de données, qui doit être imaginée avant son implémentation concrète. Les entités sont les données brutes et les associations sont les liens privilégiés qu'on veut établir entre ces données.
Il a des associations **1** — $*$ (plusieurs à un), et aussi **1** — **1** (un à un), $*$ — $*$ (plusieurs à plusieurs).

Cette fois une seule clef étrangère ne suffira pas.

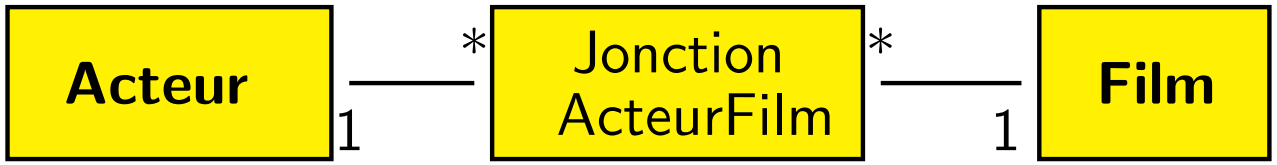
L'association plusieurs à plusieurs

On la découpe en deux associations $\mathbf{1}$ — $*$



L'association plusieurs à plusieurs

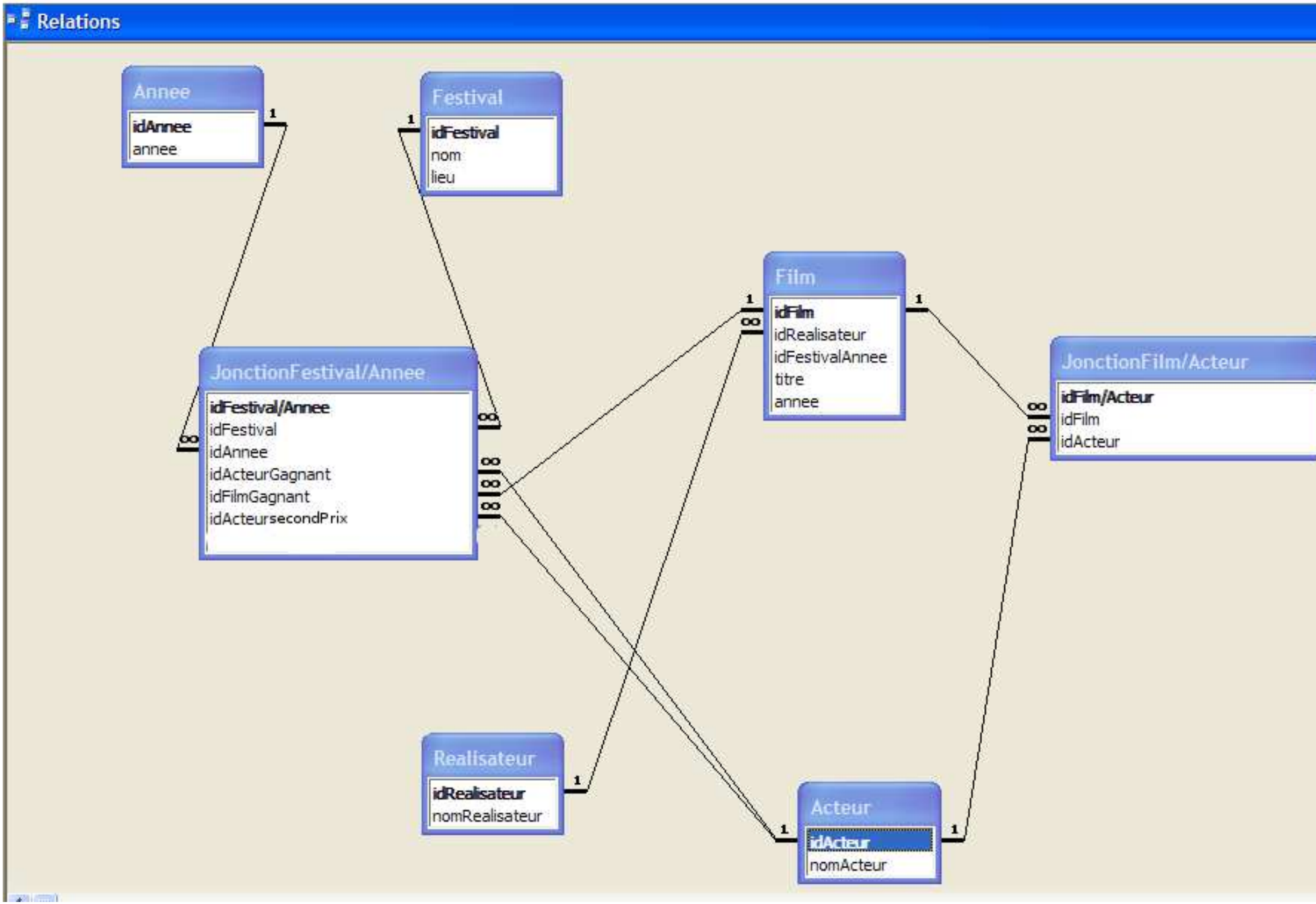
On la découpe en deux associations $\mathbf{1}$ — $*$



La table de jonction (ci-contre) contient des clefs étrangères vers chacune des deux tables.

Ex : l'acteur numéro 12 ($\rightarrow$ Benedict Cumberbatch) joue dans de nombreux films dont le numéro 18 ($\rightarrow$ Imitation Game).

id	acteur_id	film_id
1	12	18
2	12	30
3	15	18
4	12	65
5	6	65
	...	


Relations
Annee
idAnnee
annee
Festival
idFestival
nom
lieu
Film
idFilm
idRealisateur
idFestivalAnnee
titre
annee
JonctionFilm/Acteur
idFilm/Acteur
idFilm
idActeur
JonctionFestival/Annee
idFestival/Annee
idFestival
idAnnee
idActeurGagnant
idFilmGagnant
idActeursecondPrix
Realisateur
idRealisateur
nomRealisateur
Acteur
idActeur
nomActeur

Plan

Jointures (symétriques)

La jointure va rendre compte du lien clef étrangère $\rightarrow$ clef primaire quand on interroge une base de données.

Sur le principe il s'agit de faire deux opérations connues
- un produit cartésien sur les couples de clé
- puis une sélection de la forme R1.clefE=R2.clefP

La machine fait cela de façon optimisée. C'est pourquoi il est bon d'utiliser la syntaxe spécifique pour les jointures symétriques :

Jointure symétrique

```
SELECT R1.Attributs, R2.Attributs
    FROM R1 JOIN R2 ON R1.A1=R2.A2
    [WHERE, GROUP BY, HAVING ...]
```

Jointures (symétriques)

La jointure va rendre compte du lien clef étrangère → clef primaire quand on interroge une base de données.

Sur le principe il s'agit de faire deux opérations connues
- un produit cartésien sur les couples de clé
- puis une sélection de la forme R1.clefE=R2.clefP

La machine fait cela de façon optimisée. C'est pourquoi il est bon d'utiliser la syntaxe spécifique pour les jointures symétriques :

Jointure symétrique

```
SELECT R1.Attributs, R2.Attributs
    FROM R1 JOIN R2 ON R1.A1=R2.A2
    [WHERE, GROUP BY, HAVING ...]
```

Ex

```
SELECT film.titre,realisateur.nom FROM film
    JOIN realisateur ON film.idRealisateur=realisateur.idRealisateur
```

Utilisation

La jointure se fait en premier.

Tout se passe comme si on "commençait par reporter dans le tableau les données pertinentes issues des autres".

Exemple : décortiquer

```
SELECT nomVendeur,SUM(montant) FROM facture
    JOIN vendeur ON facture.idVendeur=vendeur.idVendeur
    GROUP BY facture.idVendeur
```

Jointures - remarques syntaxiques

Points de syntaxe

Jointures successives

.. FROM ra JOIN rb ON ra.a1=rb.a2
JOIN rc ON rb.b2=rc.c3 ...

Attributs multiples

... FROM ra JOIN rb ON ra.a1=rb.a2 AND ra.b1=rb.b2 ...

Autojointure

... FROM r AS r1 JOIN r AS r2 ON r.a1=r.b1 ...

Jointure et requêtes imbriquées

... FROM ra JOIN (SELECT ...) AS rb ON ra.a1=rb.a2 ...

Encore Centrale 2016

Table `plan_vol`

id_vol	depart	arrivee	jour	heure	niveau
AF1204	CDG	FCO	2016-05-02	07:35	300
AF1205	FCO	CDG	2016-05-02	10:25	300
AF1504	CDG	FCO	2016-05-02	10:05	310
AF1505	FCO	CDG	2016-05-02	13:00	310

Table `aeroport`

id_aero	nom	ville	pays
CDG	Charles-de-Gaulle	Paris	France
ORY	Orly	Paris	France
MRS	Marignane	Marseille	France
FCO	Fiumcino	Rome	Italie

Tracer un diagramme entité association

Encore Centrale 2016

Table `plan_vol`

id_vol	depart	arrivee	jour	heure	niveau
AF1204	CDG	FCO	2016-05-02	07:35	300
AF1205	FCO	CDG	2016-05-02	10:25	300
AF1504	CDG	FCO	2016-05-02	10:05	310
AF1505	FCO	CDG	2016-05-02	13:00	310

Table `aeroport`

id_aero	nom	ville	pays
CDG	Charles-de-Gaulle	Paris	France
ORY	Orly	Paris	France
MRS	Marignane	Marseille	France
FCO	Fiumcino	Rome	Italie

Donner la liste des numéros de vols au départ de Paris le 2 mai 2016

Encore Centrale 2016

Table `plan_vol`

id_vol	depart	arrivee	jour	heure	niveau
AF1204	CDG	FCO	2016-05-02	07:35	300
AF1205	FCO	CDG	2016-05-02	10:25	300
AF1504	CDG	FCO	2016-05-02	10:05	310
AF1505	FCO	CDG	2016-05-02	13:00	310

Table `aeroport`

id_aero	nom	ville	pays
CDG	Charles-de-Gaulle	Paris	France
ORY	Orly	Paris	France
MRS	Marignane	Marseille	France
FCO	Fiumcino	Rome	Italie

SELECT id_vol FROM vol
JOIN aeroport AS d ON d.id_aero = depart
JOIN aeroport AS a ON a.id_aero = arrivee
WHERE d.pays = 'France' AND a.pays = 'France' AND jour = '2016-05-02'

Encore Centrale 2016

Table `plan_vol`

id_vol	depart	arrivee	jour	heure	niveau
AF1204	CDG	FCO	2016-05-02	07:35	300
AF1205	FCO	CDG	2016-05-02	10:25	300
AF1504	CDG	FCO	2016-05-02	10:05	310
AF1505	FCO	CDG	2016-05-02	13:00	310

Table `aeroport`

id_aero	nom	ville	pays
CDG	Charles-de-Gaulle	Paris	France
ORY	Orly	Paris	France
MRS	Marignane	Marseille	France
FCO	Fiumcino	Rome	Italie

Fournir la liste des couples (Id_1 , Id_2) des identifiants des vols en conflit potentiel : même trajet, en sens inverse, le même jour et à un même niveau.

Jointures - remarques syntaxiques

Pour information

- Il y a une syntaxe spécifique pour la situation où les clefs ont le même nom (NATURAL JOIN) : c'est la jointure "prévue dès la conception".
- Il existe d'autres types de jointure, disymétriques, notamment pour les cas où un enregistrement n'a pas encore de correspondant (exemple : un interrogateur sans candidat à interroger). On ne les regardera pas de façon spécifique.