Bases de données relationnelles

G. Dewaele

28 mai 2018

Les bases de données

L'un des objectifs de l'informatique est le traitement de grandes quantité d'informations.

Une solution efficace pour organiser une grande quantité d'informations est la *base de données relationnelle*.

Les outils permettant d'interagir avec ces bases de données sont appelés *systèmes de gestion de bases de données (SGBD)*.

Il en existe de nombreux, beaucoup d'entre eux utilisant, pour interagir avec les données, le langage *SQL (Simple Query Language)*.

Exemples : MySQL, Oracle, PostGreSQL, SQLite...

Les tables

Une *base de données* regroupe un ensemble de *tables*, lesquelles peuvent être représentées sous la forme de tableaux bi-dimensionnels.

TABLE Avions :

Modèle	Fabricant	Moteurs	Places	Autonomie
737-900	Boeing	2	215	5926
777-200	Boeing	2	440	14307
A321-200	Airbus	2	220	5926
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705
CRJ-100	Bombardier	2	50	1574

Les tables

Une base de données comprend généralement plus d'une table, et les systèmes de gestions de base de données permettent de *croiser* les données de plusieurs tables.

TABLE Compagnies :

Code	Compagnie	Modèle	Nombre
AF	Air France	777-200	25
AF	Air France	A321-200	15
AF	Air France	A330-200	15
AF	Air France	A340-300	12
AC	Air Canada	777-200	6
AC	Air Canada	A321-200	10
AC	Air Canada	CRJ-100	51

Attributs et domaines

Chaque colonne d'une table correspond à un *attribut*.

L'ensemble des valeurs possibles pour un attribut est appelé *domaine* de l'attribut (chaîne de caractères, valeur numérique, booléen, etc)

TABLE Avions :

Modèle	Fabricant	Moteurs	Places	Autonomie
737-900	Boeing	2	215	5926
777-200	Boeing	2	440	14307
A321-200	Airbus	2	220	5926
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705
CRJ-100	Bombardier	2	50	1574

Attributs manquants

Certaines « cases » d'une table peuvent être vides.

(En pratique, elles contiennent une valeur spéciale, `NULL`, ne faisant pas partie du domaine de l'attribut en question).

TABLE Avions :

Modèle	Fabricant	Moteurs	Places	Autonomie
737-900	Boeing	2	215	5926
777-200	Boeing	2	440	14307
A321-200	Airbus	2	220	
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705
CRJ-100	Bombardier	2		1574

Enregistrements

Les lignes de la tables sont généralement appelées *enregistrements*.

TABLE Avions :

Modèle	Fabricant	Moteurs	Places	Autonomie
737-900	Boeing	2	215	5926
777-200	Boeing	2	440	14307
A321-200	Airbus	2	220	5926
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705
CRJ-100	Bombardier	2	50	1574

Objectifs

Objectifs d'une base de données

Les bases de données permettent de stocker une grande quantité de données de différentes natures, et de les mettre en relation afin de répondre à des questions variées et complexes :

Quel est le bi-moteur encore en service avec la plus grande capacité ?

Combien de modèles d'Airbus contient la flotte d'Air France ?

Quels sont les fabricants d'avions qui proposent un modèle permettant d'effectuer Paris - Pékin sans escale ?

Quelle est la dix-septième compagnie dont le nom ne commance pas par « A » ayant le plus d'avions pouvant transporter entre quatre-vingt-sept et cent onze passagers ?

Et globalement, toutes ces questions qui vous empêchent de dormir la nuit.

Principe des requêtes

Le langage SQL permet de transcrire ces questions sous la forme de *requêtes* qui pourront être comprises par la base de donnée.

Le résultat d'une requête **doit impérativement avoir la forme d'une table** (qui peut-être réduite à une seule case).

Quels sont les fabricants d'avions ont fourni des appareils biréacteurs actuellement en service dans la flotte d'Air France ?

```
SELECT DISTINCT A.Fabricant
    FROM Avions AS A JOIN Compagnies AS C
    ON A.Modèle = C.Modèle
    WHERE A.Moteurs = 2 AND C.Compagnie = "Air France"
```

Projections

On peut sélectionner certains attributs en effectuant une *projection* grâce au mot-clé SELECT :

SELECT Modèle, Moteurs FROM Avions

Modèle	Fabricant	Moteurs	Places	Autonomie
737-900	Boeing	2	215	5926
777-200	Boeing	2	440	14307
A321-200	Airbus	2	220	5926
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705
CRJ-100	Bombardier	2	50	1574

Suppression des doublons

Le mot-clé DISTINCT, placé derrière le mot-clé SELECT, permet d'éliminer les doublons dans le résultat :

SELECT DISTINCT Fabricant, Moteurs FROM Avions

Modèle	Fabricant	Moteurs	Places	Autonomie
737-900	Boeing	2	215	5926
777-200	Boeing	2	440	14307
A321-200	Airbus	2	220	5926
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705
CRJ-100	Bombardier	2	50	1574

Calculs

Il est possible de créer temporairement des attributs supplémentaires en utilisant des expressions :

SELECT Modèle, Places/Moteurs FROM Avions

Modèle	Fabricant	Moteurs	Places	Autonomie	Pl./Mot.
737-900	Boeing	2	215	5926	107.5
777-200	Boeing	2	440	14307	220
A321-200	Airbus	2	220	5926	110
A330-200	Airbus	2	380	13890	190
A340-300	Airbus	4	440	13705	110
CRJ-100	Bombardier	2	50	1574	25

Sélection sur critère booléen

On peut également effectuer des *sélections*, c'est-à-dire ne conserver que les enregistrements vérifiant une expression booléenne grâce au mot-clé WHERE :

SELECT Modèle, Fabricant, Moteurs, Planes, Autonomie FROM Avions

WHERE Places $>$ 250

Modèle	Fabricant	Moteurs	Places	Autonomie
737-900	Boeing	2	215	5926
777-200	Boeing	2	440	14307
A321-200	Airbus	2	220	5926
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705
CRJ-100	Bombardier	2	50	1574

Sélection sur critère booléen

Pour obtenir la totalité des attributs, on peut utiliser « * » :

SELECT * FROM Avions

WHERE Places $>$ 250

Modèle	Fabricant	Moteurs	Places	Autonomie
737-900	Boeing	2	215	5926
777-200	Boeing	2	440	14307
A321-200	Airbus	2	220	5926
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705
CRJ-100	Bombardier	2	50	1574

Expressions booléennes utiles

Pour savoir si un attribut figure parmi un ensemble de valeurs, on dispose du mot-clé IN :

SELECT * FROM table WHERE nom IN (val1, val2, val3)

Pour identifier les « cases vides » (attribut non présent dans un enregistrement), on utilisera IS NULL (ou IS NOT NULL) :

SELECT * FROM table WHERE nom IS NOT NULL

Pour faire des recherches sur une partie d'une chaîne, on peut utiliser LIKE, le caractère % pouvant remplacer n'importe quelle suite de caractères :

SELECT * FROM table WHERE nom LIKE "% Dupont"

Opérateurs booléens

Les opérateurs booléens OR, AND et NOT permettent de combiner les expressions booléennes :

SELECT * FROM Avions

WHERE Places $>$ 250 AND Moteurs $=$ 2

Modèle	Fabricant	Moteurs	Places	Autonomie
737-900	Boeing	2	215	5926
777-200	Boeing	2	440	14307
A321-200	Airbus	2	220	5926
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705
CRJ-100	Bombardier	2	50	1574

Sélection et projection

Il est bien entendu possible de combiner sélection et projection :

SELECT Modèle, Moteurs FROM Avions

WHERE Places $>$ 250 AND Moteurs $=$ 2

Modèle	Fabricant	Moteurs	Places	Autonomie
737-900	Boeing	2	215	5926
777-200	Boeing	2	440	14307
A321-200	Airbus	2	220	5926
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705
CRJ-100	Bombardier	2	50	1574

Tri des résultats

Pour ordonner les résultats, on peut utiliser le mot-clé ORDER BY :

(on peut utiliser une expression, et les attributs n'ont pas besoin d'être sélectionnés)

SELECT Modèle, Moteurs FROM Avions

ORDER BY Places

Modèle	Fabricant	Moteurs	Places	Autonomie
CRJ-100	Bombardier	2	50	1574
737-900	Boeing	2	215	5926
A321-200	Airbus	2	220	5926
A330-200	Airbus	2	380	13890
777-200	Boeing	2	440	14307
A340-300	Airbus	4	440	13705

Tri des résultats

Le mot-clé DESC permet d'obtenir les résultats dans l'ordre décroissant :

SELECT Modèle, Moteurs FROM Avions

ORDER BY Places DESC

Modèle	Fabricant	Moteurs	Places	Autonomie
777-200	Boeing	2	440	14307
A340-300	Airbus	4	440	13705
A330-200	Airbus	2	380	13890
A321-200	Airbus	2	220	5926
737-900	Boeing	2	215	5926
CRJ-100	Bombardier	2	50	1574

Limitation du nombre de résultats

On peut ne conserver que les n premiers résultats avec le mot-clé LIMIT :

SELECT Modèle, Moteurs FROM Avions

ORDER BY Places

LIMIT 3

Modèle	Fabricant	Moteurs	Places	Autonomie
CRJ-100	Bombardier	2	50	1574
737-900	Boeing	2	215	5926
A321-200	Airbus	2	220	5926
A330-200	Airbus	2	380	13890
777-200	Boeing	2	440	14307
A340-300	Airbus	4	440	13705

Décalage

On peut ignorer les n premiers résultats avec le mot-clé OFFSET :

SELECT Modèle, Moteurs FROM Avions

ORDER BY Places

OFFSET 2 LIMIT 3

Modèle	Fabricant	Moteurs	Places	Autonomie
CRJ-100	Bombardier	2	50	1574
737-900	Boeing	2	215	5926
A321-200	Airbus	2	220	5926
A330-200	Airbus	2	380	13890
777-200	Boeing	2	440	14307
A340-300	Airbus	4	440	13705

Compter

La fonction COUNT permet d'effectuer des décomptes dans le résultat de la requête :

SELECT COUNT(*) FROM Avions

Modèle	Fabricant	Moteurs	Places	Autonomie	COUNT
737-900	Boeing	2	215	5926	6
777-200	Boeing	2	440	14307	
A321-200	Airbus	2	220	5926	
A330-200	Airbus	2	380	13890	
A340-300	Airbus	4	440	13705	
CRJ-100	Bombardier	2	50	1574	

Compter

Les « cases » vides (contenant NULL) ne sont pas comptées.

COUNT(Moteurs) compte les enregistrements pour lesquels l'attribut Moteur n'est pas vide.

COUNT(Moteurs, Places) compte les enregistrements pour lesquels les attributs Moteur et Places ne sont pas *tous les deux* vides.

COUNT(*) compte tous les enregistrements.

Compter les valeurs uniques

Il est possible de dénombrer les différentes valeurs prises par l'attribut, grâce à DISTINCT :

SELECT COUNT(DISTINCT Fabricant) FROM Avions

Modèle	Fabricant	Moteurs	Places	Autonomie	COUNT
737-900	Boeing	2	215	5926	3
777-200	Boeing	2	440	14307	
A321-200	Airbus	2	220	5926	
A330-200	Airbus	2	380	13890	
A340-300	Airbus	4	440	13705	
CRJ-100	Bombardier	2	50	1574	

Compter avec condition

Si l'on combine une sélection et un décompte, seuls sont comptés les enregistrements respectant la condition :

SELECT COUNT(*) FROM Avions

WHERE Moteurs = 2

Modèle	Fabricant	Moteurs	Places	Autonomie	COUNT
737-900	Boeing	2	215	5926	5
777-200	Boeing	2	440	14307	
A321-200	Airbus	2	220	5926	
A330-200	Airbus	2	380	13890	
A340-300	Airbus	4	440	13705	
CRJ-100	Bombardier	2	50	1574	

Valeur maximale

Il est également possible de déterminer la plus grande valeur d'un attribut grâce au mot-clé MAX :

SELECT MAX(Places) FROM Avions

WHERE Autonomie $<$ 10000

Modèle	Fabricant	Moteurs	Places	Autonomie	MAX(Pl.)
737-900	Boeing	2	215	5926	220
777-200	Boeing	2	440	14307	
A321-200	Airbus	2	220	5926	
A330-200	Airbus	2	380	13890	
A340-300	Airbus	4	440	13705	
CRJ-100	Bombardier	2	50	1574	

Autres fonctions

Diverses fonctions sont disponibles :

- MAX : retourne le plus grand élément
- MIN : retourne le plus petit élément
- SUM : retourne la somme des éléments
- AVG : retourne la moyenne des éléments
- VARIANCE : retourne l'écart-type des éléments

...

La disponibilité de ces fonctions peut varier d'un système de gestion de bases de données à l'autre.

Mélanger attributs et fonctions

Le résultat devant être une table, mélanger attributs et fonctions peut donner une erreur, ou un résultat partiellement « aléatoire ».

La requête suivante est admise avec SQLite, mais pas avec Oracle SQL :

SELECT Modèle, MAX(Places) FROM Avions

Modèle	Fabricant	Moteurs	Places	Autonomie	MAX(Pl.)
737-900	Boeing	2	215	5926	440
777-200	Boeing	2	440	14307	
A321-200	Airbus	2	220	5926	
A330-200	Airbus	2	380	13890	
A340-300	Airbus	4	440	13705	
CRJ-100	Bombardier	2	50	1574	

Grouper

Il est possible de regrouper les valeurs identiques d'un attribut, grâce au mot-clé GROUP BY :

SELECT Fabricant FROM Avions

GROUP BY Fabricant

Modèle	Fabricant	Moteurs	Places	Autonomie
737-900	Boeing	2	215	5926
777-200		2	440	14307
A321-200	Airbus	2	220	5926
A330-200		2	380	13890
A340-300		4	440	13705
CRJ-100	Bombardier	2	50	1574

Grouper

Le résultat doit rester une table, aussi sélectionner des attributs non-groupés ne fonctionne pas toujours :

SELECT Modèle, Fabricant FROM Avions

GROUP BY Fabricant

Modèle	Fabricant	Moteurs	Places	Autonomie
737-900	Boeing	2	215	5926
777-200		2	440	14307
A321-200	Airbus	2	220	5926
A330-200		2	380	13890
A340-300		4	440	13705
CRJ-100	Bombardier	2	50	1574

COUNT, MIN, MAX... et groupes

Les fonctions telles que COUNT ou MAX agissent sur les groupes :

SELECT Fabricant, COUNT(*) FROM Avions

GROUP BY Fabricant

Modèle	Fabricant	Moteurs	Places	Autonomie	COUNT
737-900	Boeing	2	215	5926	2
777-200		2	440	14307	
A321-200	Airbus	2	220	5926	3
A330-200		2	380	13890	
A340-300		4	440	13705	
CRJ-100	Bombardier	2	50	1574	1

Groupes sur plusieurs critères

Il est possible de regrouper les enregistrements sur plusieurs attributs :

SELECT Fabricant, MIN(Places) FROM Avions

GROUP BY Fabricant, Moteurs

Modèle	Fabricant	Moteurs	Places	Autonomie	MIN(Pl.)
737-900	Boeing	2	215	5926	215
777-200			440	14307	
A321-200	Airbus	2	220	5926	220
A330-200			380	13890	
A340-300	Airbus	4	440	13705	440
CRJ-100	Bombardier	2	50	1574	50

Filtrer les groupes

De la même façon que l'on peut sélectionner des enregistrements, on peut filtrer les groupes grâce au mot-clé HAVING :

SELECT Fabricant, MIN(Places) FROM Avions

GROUP BY Fabricant, Moteurs

HAVING MIN(Places) $>$ 215

Modèle	Fabricant	Moteurs	Places	Autonomie	MIN(Pl.)
737-900	Boeing	2	215	5926	215
777-200			440	14307	
A321-200	Airbus	2	220	5926	220
A330-200			380	13890	
A340-300	Airbus	4	440	13705	440
CRJ-100	Bombardier	2	50	1574	50

Double filtrage

WHERE et HAVING peuvent être utilisés dans une même requête (WHERE agit avant le regroupement, HAVING après) :

SELECT Fabricant, MIN(Places) FROM Avions

WHERE Autonomie $>$ 10000

GROUP BY Fabricant, Moteurs

HAVING MIN(Places) $>$ 215

Modèle	Fabricant	Moteurs	Places	Autonomie	MIN(Pl.)
737-900	Boeing	2	215	5926	440
777-200			440	14307	
A321-200	Airbus	2	220	5926	380
A330-200			380	13890	
A340-300	Airbus	4	440	13705	440
CRJ-100	Bombardier	2	50	1574	50

Ordre des éléments

Il convient de conserver les éléments de la requête dans un ordre logique, correspondant à la chronologie des opérations :

- WHERE avant GROUP BY
- HAVING après GROUP BY
- ORDER BY après les opérations de sélection
- OFFSET / LIMIT en dernier

La projection (SELECT) est la dernière opération effectuée, elle n'est placée en tête que par commodité pour la lecture.

Exemple de requête combinant tous les éléments

SELECT Fabricant, MIN(Places) FROM Avions

WHERE Autonomie $>$ 10000

GROUP BY Fabricant, Moteurs

HAVING MIN(Places) $>$ 215

ORDER BY MIN(Places) DESC

OFFSET 2 LIMIT 1

Modèle	Fabricant	Moteurs	Places	Autonomie	MIN(Pl.)
737-900	Boeing	2	215	5926	440
777-200			440	14307	
A321-200	Airbus	2	220	5926	380
A330-200			380	13890	
A340-300	Airbus	4	440	13705	440
CRJ-100	Bombardier	2	50	1574	50

Sous-requêtes

Le résultat d'une requête peut servir dans une autre requête. On place la sous-requêtre entre parenthèses.

Pour une sous-requête renvoyant une valeur, dans une expression :

SELECT * FROM Avions

WHERE Places > (SELECT AVG(Places) FROM Avions) :

Modèle	Fabricant	Moteurs	Places	Autonomie
737-900	Boeing	2	215	5926
777-200	Boeing	2	440	14307
A321-200	Airbus	2	220	5926
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705
CRJ-100	Bombardier	2	50	1574

Sous-requêtes

Pour une sous-requête renvoyant une « liste » (table réduite à un seul attribut), dans une expression booléenne, avec le mot-clé IN :

SELECT Modèles FROM Avions

WHERE Fabricant IN

(SELECT DISTINCT Fabricant FROM Avions

WHERE Places $>$ 400)

Modèle	Fabricant	Moteurs	Places	Autonomie
737-900	Boeing	2	215	5926
777-200	Boeing	2	440	14307
A321-200	Airbus	2	220	5926
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705
CRJ-100	Bombardier	2	50	1574

Sous-requêtes

Enfin, une sous-requête renvoyant une « table » peut servir de table :

```
SELECT AVG(TMin) FROM
    (SELECT Fabricant, MIN(Places) AS TMin
     FROM Avions GROUP BY Fabricant)
```

Modèle	Fabricant	Mot.	Places	Auton.	TMin	AVG(TM.)
737-900	Boeing	2	215	5926	215	161.33
777-200		2	440	14307		
A321-200	Airbus	2	220	5926	220	
A330-200		2	380	13890		
A340-300		4	440	13705		
CRJ-100	Bombardier	2	50	1574	50	

On remarquera le AS pour nommer une colonne de la sous-requête.

Combiner les tables

Le plein intérêt des systèmes de gestion de bases de données ne se révèle que lorsque l'on croise les informations de plusieurs tables.

Ils permettent la création de tables temporaires regroupant les informations de plusieurs tables.

Il existe plusieurs façons de regrouper les tables, la plus simple étant un produit cartésien :

A	a
B	b

$\times$

1	x
2	y
3	z

$\mapsto$

A	a	1	x
B	b	1	x
A	a	2	y
B	b	2	y
A	a	3	z
B	b	3	z

Combiner les tables

TABLE Avions :

Modèle	Fabricant	Moteurs	Places	Autonomie
737-900	Boeing	2	215	5926
777-200	Boeing	2	440	14307
A321-200	Airbus	2	220	5926
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705
CRJ-100	Bombardier	2	50	1574

TABLE Compagnies :

Code	Compagnie	Modèle	Nombre
AF	Air France	777-200	25
AF	Air France	A321-200	15
AF	Air France	A330-200	15
AF	Air France	A340-300	12
AC	Air Canada	777-200	6
AC	Air Canada	A321-200	10
AC	Air Canada	CRJ-100	51

Produit cartésien de deux tables

TABLE Avions, Compagnies :

Avions					Compagnies			
Modèle	Fabricant	Moteurs	Places	Autonomie	Code	Compagnie	Modèle	Nombre
737-900	Boeing	2	215	5926	AF	Air France	777-200	25
777-200	Boeing	2	440	14307	AF	Air France	777-200	25
A321-200	Airbus	2	220	5926	AF	Air France	777-200	25
A330-200	Airbus	2	380	13890	AF	Air France	777-200	25
A340-300	Airbus	4	440	13705	AF	Air France	777-200	25
CRJ-100	Bombardier	2	50	1574	AF	Air France	777-200	25
737-900	Boeing	2	215	5926	AF	Air France	A321-200	15
777-200	Boeing	2	440	14307	AF	Air France	A321-200	15
A321-200	Airbus	2	220	5926	AF	Air France	A321-200	15
A330-200	Airbus	2	380	13890	AF	Air France	A321-200	15
A340-300	Airbus	4	440	13705	AF	Air France	A321-200	15
CRJ-100	Bombardier	2	50	1574	AF	Air France	A321-200	15
737-900	Boeing	2	215	5926	AF	Air France	A330-200	15
777-200	Boeing	2	440	14307	AF	Air France	A330-200	15
A321-200	Airbus	2	220	5926	AF	Air France	A330-200	15
A330-200	Airbus	2	380	13890	AF	Air France	A330-200	15
A340-300	Airbus	4	440	13705	AF	Air France	A330-200	15
CRJ-100	Bombardier	2	50	1574	AF	Air France	A330-200	15
...	...	...	...	...	...	...	...	...

(et 24 autres)

Rendre cohérent produit cartésien

Puisque les tables peuvent avoir des attributs portant le même nom, on fait précéder le nom de l'attribut par le nom de la table pour les distinguer (Avions.Modèle, Compagnie.Code, etc.)

La table obtenue par le produit cartésien de deux tables n'a fréquemment guère de sens.

En effet, Si un même attribut est présent dans les deux tables (ici Avions.Modèle et Compagnie.Modèle), il est plus logique de ne s'intéresser qu'aux lignes où les deux attributs sont les mêmes.

Rendre cohérent le produit cartésien

SELECT * FROM Avions, Compagnies

WHERE Avions.Modèle = Compagnie.Modèle

Avions					Compagnies			
Modèle	Fabricant	Moteurs	Places	Autonomie	Code	Compagnie	Modèle	Nombre
737-900	Boeing	2	215	5926	AF	Air France	777-200	25
777-200	Boeing	2	440	14307	AF	Air France	777-200	25
A321-200	Airbus	2	220	5926	AF	Air France	777-200	25
A330-200	Airbus	2	380	13890	AF	Air France	777-200	25
A340-300	Airbus	4	440	13705	AF	Air France	777-200	25
CRJ-100	Bombardier	2	50	1574	AF	Air France	777-200	25
737-900	Boeing	2	215	5926	AF	Air France	A321-200	15
777-200	Boeing	2	440	14307	AF	Air France	A321-200	15
A321-200	Airbus	2	220	5926	AF	Air France	A321-200	15
A330-200	Airbus	2	380	13890	AF	Air France	A321-200	15
A340-300	Airbus	4	440	13705	AF	Air France	A321-200	15
CRJ-100	Bombardier	2	50	1574	AF	Air France	A321-200	15
737-900	Boeing	2	215	5926	AF	Air France	A330-200	15
777-200	Boeing	2	440	14307	AF	Air France	A330-200	15
A321-200	Airbus	2	220	5926	AF	Air France	A330-200	15
A330-200	Airbus	2	380	13890	AF	Air France	A330-200	15
A340-300	Airbus	4	440	13705	AF	Air France	A330-200	15
CRJ-100	Bombardier	2	50	1574	AF	Air France	A330-200	15
...	...	...	...	...	...	...	...	...

(et 24 autres)

Jointures

On préférera effectuer une *jointure interne* des tables grâce au mot-clé JOIN (ou INNER JOIN)

SELECT * FROM Avions

JOIN Compagnies ON Avions.Modèle = Compagnie.Modèle

Avions					Compagnies			
Modèle	Fabricant	Moteurs	Places	Autonomie	Code	Compagnie	Modèle	Nombre
777-200	Boeing	2	440	14307	AF	Air France	777-200	25
A321-200	Airbus	2	220	5926	AF	Air France	A321-200	15
A330-200	Airbus	2	380	13890	AF	Air France	A330-200	15
A340-300	Airbus	4	440	13705	AF	Air France	A340-300	12
777-200	Boeing	2	440	14307	AC	Air Canada	777-200	6
A321-200	Airbus	2	220	5926	AC	Air Canada	A321-200	10
CRJ-100	Bombardier	2	50	1574	AC	Air Canada	CRJ-100	51

Jointures

Une jointure interne donne le même résultat qu'un filtrage sur un produit cartésien, mais généralement plus rapidement.

On peut joindre plus de deux tables :

SELECT * FROM table1 JOIN table2 ON table1.x = table2.y
JOIN table3 ON table2.z = table3.t

Il existe d'autres jointures, notamment pour les cas où les éléments d'une table n'ont pas de correspondance dans l'autre table.

Le produit cartésien est un type de jointure (CROSS JOIN)

Alias

On peut utiliser AS pour définir des alias pour les noms des tables :

SELECT * FROM Avions AS a

JOIN Compagnies AS c ON a.Modèle = c.Modèle

Avions / a					Compagnies / c			
Modèle	Fabricant	Moteurs	Places	Autonomie	Code	Compagnie	Modèle	Nombre
777-200	Boeing	2	440	14307	AF	Air France	777-200	25
A321-200	Airbus	2	220	5926	AF	Air France	A321-200	15
A330-200	Airbus	2	380	13890	AF	Air France	A330-200	15
A340-300	Airbus	4	440	13705	AF	Air France	A340-300	12
777-200	Boeing	2	440	14307	AC	Air Canada	777-200	6
A321-200	Airbus	2	220	5926	AC	Air Canada	A321-200	10
CRJ-100	Bombardier	2	50	1574	AC	Air Canada	CRJ-100	51

Requis si l'on joint plusieurs fois la même table.

Combiner les requêtes

SELECT * FROM Avions WHERE Places $>$ 250

Modèle	Fabricant	Moteurs	Places	Autonomie
737-900	Boeing	2	215	5926
777-200	Boeing	2	440	14307
A321-200	Airbus	2	220	5926
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705
CRJ-100	Bombardier	2	50	1574

SELECT * FROM Avions WHERE Fabricant $=$ Airbus

Modèle	Fabricant	Moteurs	Places	Autonomie
737-900	Boeing	2	215	5926
777-200	Boeing	2	440	14307
A321-200	Airbus	2	220	5926
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705
CRJ-100	Bombardier	2	50	1574

Union de requêtes ($\mathrm{Req}_1 \cup \mathrm{Req}_2$)

777-200	Boeing	2	440	14307
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705
A321-200	Airbus	2	220	5926
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705

SELECT * FROM Avions WHERE Places $>$ 250

UNION

SELECT * FROM Avions WHERE Fabricant $=$ Airbus

777-200	Boeing	2	440	14307
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705
A321-200	Airbus	2	220	5926

Conditions à respecter

Il doit y avoir le même nombre de colonnes, et les colonnes doivent être de même type, mais pas nécessairement contenir les mêmes attributs.

Les lignes apparaîssant dans les deux requêtes ne sont retournées qu'une seule fois (utiliser UNION ALL sinon).

Intersection de requêtes $(\mathrm{Req}_1 \cap \mathrm{Req}_2)$

777-200	Boeing	2	440	14307
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705

A321-200	Airbus	2	220	5926
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705

SELECT * FROM Avions WHERE Places $>$ 250

INTERSECT

SELECT * FROM Avions WHERE Fabricant $=$ Airbus

A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705

Différence de requêtes $(\mathrm{Req}_1 - \mathrm{Req}_2)$

777-200	Boeing	2	440	14307
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705

A321-200	Airbus	2	220	5926
A330-200	Airbus	2	380	13890
A340-300	Airbus	4	440	13705

SELECT * FROM Avions WHERE Places $>$ 250

MINUS (ou EXCEPT)

SELECT * FROM Avions WHERE Fabricant $=$ Airbus

777-200	Boeing	2	440	14307

Création d'une table

Le mot-clé CREATE permet de créer une table.

On spécifie la liste des attributs et leur domaine

(INTEGER pour un entier, VARCHAR pour une chaîne de caractères, FLOAT pour un réel, BOOLEAN pour une valeur booléenne...)

```
CREATE TABLE IF NOT EXISTS Avions (
    Modèle      VARCHAR(6) NOT NULL PRIMARY KEY,
    Fabricant   VARCHAR(12),
    Moteurs     INTEGER,
    Places      INTEGER,
    Autonomie   FLOAT );
```

Taille et précision des attributs

Certains types ont des arguments pour préciser la longueur maximale, par exemple les chaînes de caractères.

VARCHAR(6) correspond à des chaînes d'au plus 6 caractères.

Pour les entiers, le chiffre n'indique que le nombre de caractères lors de l'affichage, mais pas les valeurs admises.

INTEGER(2) peut contenir des entiers de plus de deux chiffres (selon le SGBD), mais certains *affichages* seront réduits à deux chiffres.

DECIMAL(5,2) ou NUMERIC(5,2) désigne un nombre a virgule avec au plus 5 chiffres, dont deux après la virgule (donc de -999.99 à $+999.99$).

Clés primaires

Il est fréquent que l'on ait besoin de pouvoir faire référence à un enregistrement dans une table *de manière unique*.

On peut définir une *clé primaire* qui est un attribut (ou un groupe d'attributs) qui ne peut (peuvent) pas prendre plusieurs fois la (les) même(s) valeur(s) dans la table. Cette unicité est garantie par le SGBD.

Pour une clé primaire constituée d'un unique attribut :

```
Modèle VARCHAR(6) NOT NULL PRIMARY KEY
```

Pour une clé primare consistant en un groupe d'attributs :

```
Modèle VARCHAR(6) NOT NULL,
Fabricant VARCHAR(12) NOT NULL,
PRIMARY KEY(Modèle, Fabricant)
```

Clés primaires

Il y a *au plus* une clé primaire (*primary key*) par table.

Si aucun attribut ou groupe d'attributs n'est satisfaisant pour cet usage, on utilise souvent un entier à cet effet (sans sens particulier, mais unique pour chaque enregistrement).

Un enregistrement ne peut avoir « NULL » pour un attribut faisant partie d'une clé primaire.

Autres clés

Il n'y a pas de contrainte particulière sur les valeurs des autres attributs d'un enregistrement, excepté :

- on peut empêcher que les enregistrements puisse ne pas avoir de valeur pour un attribut donné en indiquant NOT NULL dans la définition de l'attribut ;
- on peut limiter les valeurs possibles d'un attribut aux valeurs présentes dans une autre table en définissant une relation de *clé étrangère*.

Clés étrangères

Il est fréquent que certains attributs apparaissent dans plusieurs tables pour permettre des jointures (pas nécessairement sous le même nom !)

Pour *diminuer les risques de données incorrectes*, on *peut* déclarer ces relations, en spécifiant qu'un attribut d'une table fait référence à un attribut (souvent une clé primaire) d'une autre table.

C'est ce que l'on appelle *clé étrangère* (*foreign key*), définie par :

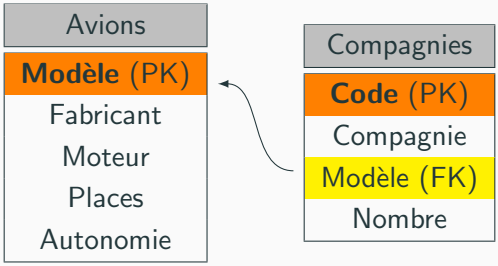
FOREIGN KEY (attribut) REFERENCES autre_table(attribut)

Seules les valeurs effectivement prises par l'attribut dans l'autre table seront acceptées pour l'attribut concerné (ou NULL).

Clés étrangères

Par exemple, pour la table Compagnie :

```
CREATE TABLE IF NOT EXISTS Compagnie (
    Code          VARCHAR(6) NOT NULL PRIMARY KEY,
    Compagnie     VARCHAR(18),
    Modèle        VARCHAR(6),
    Nombre        INTEGER,
    FOREIGN KEY   (Modèle) REFERENCES Avion(Modèle) );
```



Clés étrangères et jointures

Les clés étrangères suggèrent une jointure possible :

```
CREATE TABLE IF NOT EXISTS Compagnie (
    ...
    FOREIGN KEY (Modèle) REFERENCES Avion(Modèle) );

FROM Compagnie JOIN Avion ON Compagnie.Modèle = Avion.Modèle
```

Modification d'une table

Ajouter ou supprimer des attributs

On peut ajouter des attributs :

ALTER TABLE nom_de_la_table

    ADD nom_nouvel_attribut type_nouvel_attribut

Ou bien en supprimer :

ALTER TABLE nom_de_la_table

    DROP nom_attribut

Le renommage est possible, mais la syntaxe dépend du SGBD.

(Très vraisemblablement non-exigible)

Ajouter des enregistrements

Pour ajouter une série d'enregistrements :

INSERT INTO nom_de_la_table (attribut$_1$, attribut$_2$, attribut$_3$)

VALUES

(valeur$_1$_A, valeur$_2$_A, valeur$_3$_A),

(valeur$_1$_B, valeur$_2$_B, valeur$_3$_B),

(valeur$_1$_C, valeur$_2$_C, valeur$_3$_C);

Si l'on ne précise pas de liste d'attributs, il faut tous les préciser, dans l'ordre.

Modifier/supprimer des enregistrements

Pour modifier une série d'enregistrements :

UPDATE nom_de_la_table

    SET attribut = nouvelle_valeur

    WHERE expression_booléenne

Pour supprimer une série d'enregistrements :

DELETE FROM nom_de_la_table

    WHERE expression_booléenne

La condition WHERE ... permet de choisir quels enregistrements seront mis à jour ou supprimés.

Sans condition, tous les enregistrements sont mis à jour/supprimés !

Principe de l'algèbre relationnelle

SQL n'est pas le seul langage pour manipuler des bases de données.

Pour décrire des opérations sur des bases de données de façon générale (et démontrer des propriétés sur ces bases), on dispose d'une *algèbre relationnelle*.

Dans ce cadre, plutôt que de tables, on parle de *relations* entre les données (chaque enregistrement d'une table constitue un ensemble de données mises en relation).

Notations

On dispose de notations pour les différentes opérations :

Projection d'une relation R sur des attributs a_1, a_2 $\pi_{a_1,a_2}(R)$
SELECT DISTINCT a1, a2 FROM R

Sélection dans R selon une condition cond. $\sigma_{\text{cond.}}(R)$
SELECT * FROM R WHERE cond.

Produit cartésien de R_1 et R_2 $R_1 \times R_2$
R1, R2 ou R1 CROSS JOIN R2

Jointure de R_1 et R_2 sur cond. $R_1 \bowtie_{\text{cond.}} R_2$
R1 JOIN R2 ON cond.

Notations (suite)

On dispose de notations pour les différentes opérations :

Union de deux relation R_1 et R_2	$R_1 \cup R_2$
Intersection de deux relation R_1 et R_2	$R_1 \cap R_2$
Différence entre deux relation R_1 et R_2	$R_1 - R_2$

Par ailleurs, dans les expressions booléennes, on note généralement

- $\wedge$ le ET logique,
- $\vee$ le OU logique,
- $\neg$ le NON logique.

Utilisation

Par exemple, la notation :

$$\pi_{\text{C.Compagnie,A.Modele}} (\sigma_{\text{A.Autonomie}>10000} (\text{C} \bowtie_{\text{C.Modele=A.Modele}} \text{A}))$$

où C et A représentent les relations liées aux tables Compagnie et Avion

correspond, en langage SQL, à la requête :

SELECT Compagnie.Compagnie, Avion.Modèle

FROM Compagnie AS C JOIN Avion AS A

ON Compagnie.Modèle = Avion.Modèle

WHERE Avion.Autonomie > 10000

Utilisation

On a par exemple justifié qu'une jointure interne était équivalente à un produit cartésien suivi d'une sélection.

Cela s'écrira formellement :

$$\mathrm{R}_1 \bowtie_{\text{cond.}} \mathrm{R}_2 \;=\; \sigma_{\text{cond.}} \left(\mathrm{R}_1 \times \mathrm{R}_2\right)$$

L'algèbre relationnelle permet de démontrer de tels résultats.

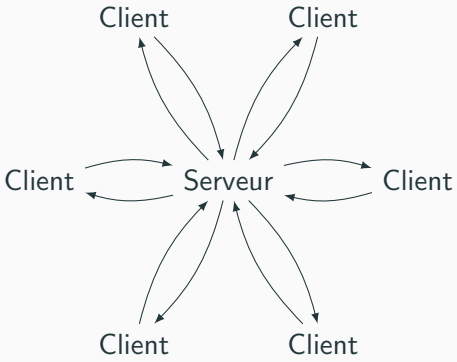
Architecture « monolithique »

L'utilisateur interagit avec programme qui effectue, seul, tous les calculs, et gère ses données, et retourne les résultats qu'il a obtenu.

C'est ce qui se passe avec le module `SQLite3` de Python.

Architecture « client-serveur »

L'utilisateur interagit avec un client « mince » qui transmet la demande à un serveur, récupère le résultat, et le communique à l'utilisateur.



Architecture « client-serveur »

Client et serveur sont des programmes séparés.

Ils peuvent tourner sur la même machine physique ou non.

Dans ce second cas, les échanges se font généralement par un réseau.

Avantages : mise en commun des données, sécurisation, déploiement des clients sur des plates-formes différentes, etc.

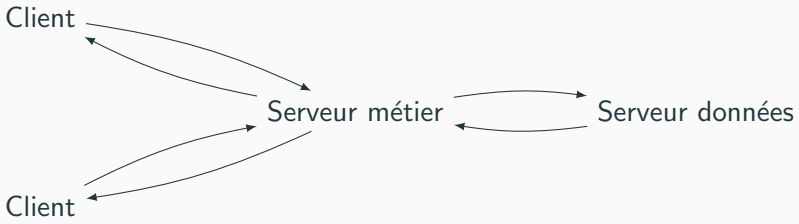
On peut par exemple installer un serveur MySQL et communiquer avec lui via un client écrit en Python avec le module `mysql`

Architecture « trois couches »

Appelée « 3-tiers » en anglais.

Cette fois, trois intervenants : le client (couche de présentation) communique avec un premier serveur (couche métier) qui interprète les demandes du clients, mais ne gère pas les données.

Le premier serveur transmet ses besoins en données à un second serveur qui obtient les données souhaitées et les retourne au premier serveur, lequel les met en forme avant de les retourner au client.



Architecture « trois couches »

Chacun des trois éléments peut ou non se trouver sur la même machine physique que les autres.

Avantages de cette architecture : davantage de sécurisation et de souplesse ; plusieurs serveurs métier peuvent partager les mêmes données.

C'est le cas de l'accès à la base de données *Mondial* en ligne :

le navigateur (client) communique avec un serveur web (couche métier) qui lui-même extrait ses données d'un serveur de bases de données Oracle.