

I - Bases de données et terminologie

Le terme *Big Data* désigne le domaine de l'informatique consacré à la gestion des très grandes quantités de données. Il est associé à l'ensemble des méthodes qui permettent non seulement de stocker rationnellement les données mais surtout de les restituer à la demande selon des critères déterminés.

En termes de volume, une entreprise de taille moyenne peut être amenée à stocker autant de données que la librairie du Congrès américain considérée comme la plus grande bibliothèque du monde. De plus, sur le plan de la variété des données, on voit cohabiter des données interpersonnelles (communications électroniques, e-mails, ...), des données d'interaction homme-machine (recouvrant diverses transactions comme les archives des cartes bancaires, des historiques de navigation web,...) et les données inter-machine (celles issues d'une communication entre machines, par exemple les GPS, les caméras de surveillance, la géolocalisation,...). Ce sont ces dernières qui vont probablement augmenter le plus, à travers le développement exponentiel des objets connectés.

Enfin, le rythme de renouvellement et de défilement des données s'est lui aussi considérablement accru : chaque jour, 45 milliards d'e-mails sont envoyés, 50 millions de tweets sont postés dans le monde, pas moins de 165 millions de transactions bancaires sont réalisées quotidiennement dans la zone euro, des milliers de touristes stockent leurs millions photos de vacances sur le cloud, etc.

Pour gérer de telles masses de données, il est nécessaire de disposer de méthodes de stockage permettant notamment un temps de restitution satisfaisant. On fait alors appel au concept de *base de données*.

Considérons quelques exemples de bases de données.

Exemple

Une première solution envisageable pour stocker des données est l'utilisation d'un tableau. Prenons par exemple une base de données contenant la liste des aéroports du monde.

Nom	Ville	Pays	Continent	Type
Biarritz-Anglet-Bayonne Airport	Biarritz/Anglet/Bayonne	France	Europe	medium_airport
Milhaud Heliport	Toulon	France	Europe	heliport
Toulon-Hyères Airport	Toulon/Hyères/Le Palyvestre	France	Europe	medium_airport
Lake Hood Seaplane Base	Anchorage	États-Unis	Amérique du Nord	seaplane_base
Ted Stevens Anchorage International Airport	Anchorage	États-Unis	Amérique du Nord	large_airport
Mandalay International Airport	Mandalay	Myanmar	Asie	large_airport

Notons que plusieurs informations sont redondantes (par exemple, l'information «France» est stockée à de multiples reprises) et des couples d'informations sont redondants (le couple (Pays, Continent) sera toujours identique pour un pays donné, on pourrait ne stocker qu'une seule fois le fait que les États-Unis sont en Amérique du Nord).

Dans le cas d'une telle table, des requêtes simples sont aisées. Ainsi, faire la liste de tous les aéroports français ne pose pas de problème. Faire la liste de tous les héliports français est un peu plus difficile.

Exemple

On souhaite recouper une liste de films avec leur metteur en scène et leurs acteurs, avec les séances dans différents cinémas ainsi que les coordonnées des cinémas.

Dans ce cas on peut imaginer que chaque semaine est fait un tableau unique comportant plusieurs colonnes

1	2	3	4	5	6	7
film	réalisateur	acteurs	cinéma	séance	adresse	n° tél.

Un tel tableau aura un très grand nombre de lignes et

- un même film sera répété plusieurs fois;
- un même cinéma sera répété plusieurs fois.

De plus il faudra actualiser chaque semaine.

Toutefois certaines données pourraient être conservées dans un tableau (colonnes 1,2,3, ou colonnes 4,6,7).

Et chaque semaine on rajouterait des données en colonnes 1,2,3, mais on en créerait aussi en colonne 1,4,5, avec le problème de la colonne 1 qui est commune.

Une base de données est un *conteneur* servant à stocker des renseignements bruts tels que des chiffres, des dates ou des mots. Ces données, retraitées par des moyens informatiques, permettent de produire une information : par exemple, des chiffres et des noms assemblés et triés formeront un annuaire téléphonique.

La base de données est la pièce centrale d'un dispositif informatique dit *système de base de données* qui régit la collecte, le stockage, le retraitement et l'utilisation de données. Ce dispositif, en plus de la base de données elle-même, comporte également un logiciel-moteur, le *système de gestion de base de données* (SGBD ou DBMS en anglais pour database management system), des logiciels applicatifs, et un ensemble de règles relatives à l'accès et l'utilisation des informations.

Le SGBD est une suite de programmes qui manipulent la structure de la base et dirigent l'accès aux données qui y sont stockées. Tout accès aux données passe par le SGBD, qui sert d'intermédiaire entre la base de données et ses usagers. Ses tâches sont multiples :

- il reçoit des demandes de manipulation de contenu (rechercher, ajouter ou supprimer des enregistrements par exemple) et effectue les opérations nécessaires sur les fichiers adéquats. Il cache ainsi la complexité réelle des opérations et offre une vue synthétique pour l'utilisateur ;
- il permet à plusieurs usagers de manipuler simultanément le contenu et peut donc offrir différentes *vues* sur un même ensemble de données ;
- il assure la cohérence, la confidentialité et la pérennité des données. Le logiciel refusera qu'un usager modifie ou supprime une information s'il n'y a pas été préalablement autorisé ; il refusera aussi de stocker une information qui n'est pas conforme aux règles de cohérence associées aux bases. De plus, chaque opération est inscrite dans un journal, ce qui permet d'annuler ou de terminer l'opération même en cas de crash informatique garantissant ainsi la cohérence du contenu.

Actuellement, les principaux systèmes de gestion de base de données sur le marché sont, au niveau commercial, *IBM DB2*, *Oracle Database*, *Microsoft SQL Server* et dans le domaine libre, *MySQL* et *PostgreSQL*.

II - Organisation d'une base de données relationnelle

La très grande majorité des bases de données est fondée sur une organisation déduite du *modèle relationnel* élaboré en 1970 par Edgar F. CODD. Ces bases sont qualifiées de *bases de données relationnelles*.

Dans une base de données relationnelle, l'information est répartie dans des tableaux à deux dimensions appelés *tables* ou *relations*. Une base de données consiste en une ou plusieurs tables.

Exemple

On considère une base de données CINEMA constituée de quatre tables dénommées PERSONNAGE, FILM, LICENCE et JOUEDANS.

<u>Id</u>	Nom	Espèce	Sexe	Côté
1	Harry Potter	humain	M	C
2	Hermione Granger	humain	F	C
3	Dobby	elfe	M	C
4	Voldemort	esprit	M	O
...
14	R2D2	robot	NULL	C
15	Chewbacca	wookie	M	C
16	Yoda	NULL	M	C
17	Dark Vador	humain	M	O
18	Obiwan Kenobi	humain	M	C
...
154	Bilbon	hobbit	M	C
155	Frodon	hobbit	M	C
156	Gandalf le Gris	NULL	M	C
157	Gimli	nain	M	C
158	Arwenn	elfe	F	C
159	Sauron	esprit	M	O
...

Table PERSONNAGE

<u>Id</u>	Nom	Année	Univers
1	Harry Potter à l'école des sorciers	2001	Harry Potter
2	Harry Potter et la chambre des secrets	2002	Harry Potter
3	Harry Potter et le prisonnier d'Azkabhan	2004	Harry Potter
...
9	La guerre des étoiles	1977	Star Wars
10	L'empire contre-attaque	1980	Star Wars
11	Le retour du Jedi	1983	Star Wars
12	La menace Fantôme	1999	Star Wars
...
21	La communauté de l'Anneau	2001	Le seigneur des anneaux
22	Les deux tours	2002	Le seigneur des anneaux
23	Le retour du roi	2003	Le seigneur des anneaux

Table FILM

<u>Nom</u>	Créateur	Genre
Harry Potter	J.K Rowling	F
Star Wars	G. Lucas	SF
Le Seigneur des Anneaux	J.R.R Tolkien	MF
...

Table LICENCE

<u>Qui</u>	<u>Dans</u>
1	1
1	2
...	...
3	2
...	...
16	10
16	12
...	...

Table JOUEDANS

Chaque colonne de chaque table correspond à un **attribut**, identifié par son nom. L'ensemble des valeurs autorisées pour les éléments de la colonne d'attribut a est appelé **domaine de a** .

En pratique, les domaines sont des ensembles correspondant à des types informatiques simples (booléens, entiers sur 32 bits, chaînes de caractères de taille fixe, de taille variable, nombres en virgule flottante, données de type date/heure, etc.).

L'ensemble des associations constituées par les attributs d'une table et leurs domaines respectifs est appelé **schéma relationnel de la table**.

Pour une table T d'attributs a_1, \dots, a_n variant respectivement dans des domaines $\mathcal{D}_1, \dots, \mathcal{D}_n$, on exprime son schéma relationnel sous la forme :

$$T \{a_1 : \mathcal{D}_1; \dots; a_n : \mathcal{D}_n\}$$

Il n'y a *a priori* pas d'ordre sur les attributs dans le schéma.

Exemple

Dans l'exemple précédent, la table PERSONNAGE comporte cinq attributs : Id, Nom, Espèce, Sexe, Côté.

Son schéma relationnel est le suivant :

```
PERSONNAGE { Id : int(64) ; Nom : string(100) ; Espece : string(20) ; Sexe : string(1) ; Cote : string(1) }
```

De même, on a :

```
FILM { Id : int(64) ; Nom : string(100) ; Annee : int(16), Licence : string(100) }
```

```
LICENCE { Nom : string(100) ; Createur : string(100) ; Genre : string(2) }
```

```
JOUEDANS { Qui : int(64) ; Dans : int(64) }
```

Résumons la terminologie principale :

Relation ou Table	Tableau avec des colonnes et des lignes
Attribut	Une colonne nommée de la table
Domaine	Ensemble ou type de valeurs admissibles pour un attribut
Schéma relationnel	Ensemble des paires {attribut, domaine} définissant la table
n-uplet ou enregistrement	Une ligne de la table
Clé primaire	Attribut (ou ensemble minimum d'attributs) qui permet d'identifier de manière unique un n-uplet dans une table. Elle est choisie par l'utilisateur parmi toutes les clés candidates.
Requêtes	Interactions avec une base de données formulées dans un langage
Base de donnée (BDD)	Ensemble des tables munies de leurs schémas relationnels : elle est stockée dans des fichiers
Système de Gestion de Base de Données (SGBD)	Permet à des utilisateurs de créer des BDD, d'y accéder et de les modifier (ex : SQLite, MySQL...)
Interface graphique	Facilite les interactions entre l'utilisateur et le SGBD (ex : SQLiteStudio, phpMyAdmin...)
SQL	Langage informatique commun aux SGBD permettant de formuler des requêtes sur une BDD.

Exemple

Considérons la relation (ou table) nommée élève ci-dessous :

élève			
Id	Nom	Prénom	Classe
1	Meyer	Zoé	ECG11
2	Michel	Florent	ECG11
3	Benoit	Marie	ECG21
4	Michel	Zoé	ECG22

- Exemple d'attribut : **Id** représentant le numéro d'identifiant de l'élève.
- Exemple de domaine : \mathbb{N} est le domaine associé à l'attribut **Id**.
- Exemple de n -uplet : 2, Michel, Florent, ECG11
- Clé : l'attribut **Id** est le seul permettant d'identifier de manière unique un n -uplet. Aucun autre attribut ne peut désigner une clé. Il peut donc être choisi comme clé primaire.
- Schéma relationnel :

$$\{ (\mathbf{Id}, \mathbb{N}), (\mathbf{Nom}, \text{Texte}), (\mathbf{Prénom}, \text{Texte}), (\mathbf{Classe}, \text{Texte}) \}$$

III - Requêtes

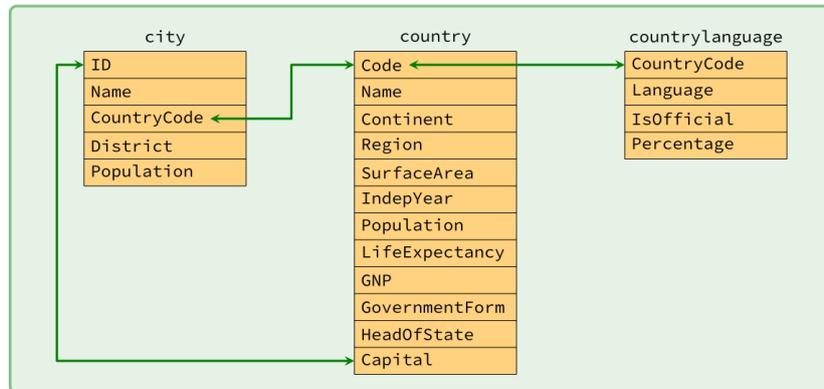
III.1 - Sélection des attributs (colonnes) et des enregistrements (lignes)

Commençons par quelques exemples.

Si certains attributs d'une table ne nous intéressent pas, on peut ne considérer que certaines colonnes.

Exemple

On manipule une base de données **world** (disponible sur le site) se présentant ainsi :



La table pays contient des informations sur les pays du monde. Pour sélectionner la table en entier en langage SQL :

```
SELECT * FROM pays
```

nom	region	surface	population	pib
Afghanistan	South Asia	652225	26000000	NULL
Albania	Europe	28728	3200000	6656000000
Algeria	Africa	2400000	32900000	75012000000
...				

Pour ne considérer que certaines colonnes :

```
SELECT nom, region FROM pays
```

nom	region
Afghanistan	South Asia
Albania	Europe
Algeria	Africa
Andorra	Europe
Angola	Africa
...	

Exemple

Considérons la table Films suivante :

film	réalisateur	acteur
Le discours d'un roi	Tom Hooper	Colin Firth
Le discours d'un roi	Tom Hooper	Geoffrey Rush
Kingsman	Matthew Vaughn	Colin Firth
Kingsman	Matthew Vaughn	Taron Egerton
Imitation Game	Morten Tyldum	Benedict Cumberbatch
X-Men : Le commencement	Matthew Vaughn	James McAvoy
X-Men : Le commencement	Matthew Vaughn	Kevin Bacon
...		

La requête :

```
SELECT film, réalisateur FROM Films
```

donne :

film	réalisateur
Le discours d'un roi	Tom Hooper
Le discours d'un roi	Tom Hooper
Kingsman	Matthew Vaughn
Kingsman	Matthew Vaughn
Imitation Game	Morten Tyldum
X-Men : Le commencement	Matthew Vaughn
X-Men : Le commencement	Matthew Vaughn
...	

Pour éviter les doublons on utilise :

```
SELECT DISTINCT film, réalisateur FROM Films
```

film	réalisateur
Le discours d'un roi	Tom Hooper
Kingsman	Matthew Vaughn
Imitation Game	Morten Tyldum
X-Men : Le commencement	Matthew Vaughn
...	

Parmi l'ensemble des enregistrements d'une table, on peut sélectionner les lignes qui nous intéressent en imposant des contraintes. Ces dernières peuvent s'exprimer avec des connecteurs logiques (AND, OR,...).

Exemple

Une contrainte peut-être par exemple la vérification d'une égalité (Nom=France) ou d'une inégalité. Par exemple :

```
SELECT * FROM pays WHERE population/surface>200
```

nom	region	surface	population	pib
Bahrain	Middle East	717	754000	9357140000
Bangladesh	South Asia	143998	152600000	67144000000
Barbados	Americas	430	272000	2518720000
Belgium	Europe	30528	10300000	319609000000
Burundi	Africa	27816	7300000	NULL
...				

On peut coupler cette démarche avec le choix de colonnes :

```
SELECT nom, population/surface, pib FROM pays WHERE population/surface>200
```

nom	population/surface	pib
Bahrain	1051	9357140000
Bangladesh	1059	67144000000
Barbados	632	2518720000
Belgium	337	319609000000
Burundi	262	NULL
...		

Exemple

À partir d'une table `periodic`, la requête demandant les éléments de rayon atomique $< 100\text{pm}$ et de masse volumique $< 1\text{gcm}^{-3}$, s'écrit :

```
SELECT Z,symb,nom,masse_vol,r_at FROM periodic WHERE r_at<100 AND masse_vol<1
```

Z	symb	nom	masse_vol (gcm ⁻³)	r_atmique (pm)
1	H	Hydrogen	0.0708	79
2	He	Helium	0.147	0
7	N	Nitrogen	0.808	92

On a vu les premières instructions, les plus fondamentales, dans les exemples précédents :

```
1 SELECT ... FROM ... WHERE ...
```

La condition donnée par WHERE peut utiliser :

- ▷ les opérateurs de comparaison (=, >, <, <=, >=, <> pour différent) ;
- ▷ les opérateurs logiques AND, OR, NOT.

D'autres opérateurs usuels semblent ne pas être au programme (IN qui teste l'appartenance, LIKE qui compare des chaînes de caractères d'une certaine forme).

On peut également considérer le *produit cartésien* de deux tables table1 et table2 où chaque élément de table1 est couplé à chaque élément de table2.

Exemple

On dispose de deux tables etudiants et villes :

Nom	Année_de_naissance	Ville	NomV	Département
Alice	1997	Toulouse	Toulouse	31
Bob	1998	Paris	Paris	75
Eve	1995	Toulouse		

Le produit cartésien etudiants×villes fait appel à toutes les colonnes des deux tables :

```
SELECT * FROM etudiants , villes
```

Nom	Année_de_naissance	Ville	NomV	Département
Alice	1997	Toulouse	Toulouse	31
Alice	1997	Toulouse	Paris	75
Bob	1998	Paris	Toulouse	31
Bob	1998	Paris	Paris	75
Eve	1995	Toulouse	Toulouse	31
Eve	1995	Toulouse	Paris	75

Le produit cartésien a peu de sens ici (et en général) car certains étudiants se retrouvent associés à deux villes différentes. Outre ce problème d'interprétation, la taille du résultat pose également problème (produit des cardinaux des deux tables).

III.4 - Jointures

La *jointure* est une opération consistant à recoller deux relations table1 et table2 en se fondant sur la correspondance de valeurs entre deux colonnes :

```
SELECT * FROM table1 JOIN table2 ON expression_logique
```

Exemple

Reprenons l'exemple précédent. On peut réaliser une jointure selon (Ville, NomV) entre les tables etudiants et villes. Cela revient à faire un produit cartésien des deux tables suivi d'une sélection sur les lignes où il y a correspondance entre les deux colonnes.

Nom	Année_de_naissance	Ville	NomV	Département
Alice	1997	Toulouse	Toulouse	31
Alice	1997	Toulouse	Paris	75
Bob	1998	Paris	Toulouse	31
Bob	1998	Paris	Paris	75
Eve	1995	Toulouse	Toulouse	31
Eve	1995	Toulouse	Paris	75

La requête suivante :

```
SELECT * FROM etudiants JOIN villes ON Ville=NomV
```

donne :

III.5 - Fonctions d'agrégation

Exemple

On dispose de la relation `relevé` ci-dessous à gauche. Un exemple d'agrégation consiste à calculer la moyenne des notes sur chaque classe, ce qui produit la relation ci-dessous à droite.

relevé				
classe	eleve	note	classe	moyenne
PC	Meyer	17,5	PC	13,38
PSI	Martin	7,75	PSI	11,08
PC	Bernard	9,25		
PSI	Robert	14,0		
PSI	Dubois	11,5		

L'agrégation sert à regrouper les lignes d'une table et à évaluer une fonction f sur ces regroupements. Les cinq fonctions à connaître sont :

Fonction	Syntaxe SQL	Valeur renvoyée
<i>comptage</i>	COUNT	nombre d'éléments
<i>moyenne</i>	AVG	moyenne des éléments
<i>somme</i>	SUM	somme des éléments
<i>min</i>	MIN	le plus petit des éléments
<i>max</i>	MAX	le plus grand des éléments

Exemple

Considérons la table `utilisateur` donnant la liste des utilisateurs d'un site web :

id	nom	ville	date_inscription	nombre_achat	id_dernier_achat
1	Marie	Paris	2010-04-22	5	24
2	Léon	Marseille	2011-08-18	3	36
3	Paul	Lyon	2011-11-02	0	NULL
4	Léon	Paris	2012-09-01	1	7
5	Paul	Nantes	2013-01-10	0	NULL

Pour compter le nombre d'utilisateurs qui ont effectué au moins un achat :

```
SELECT COUNT(*) AS nb_user FROM utilisateur WHERE nombre_achat > 0
```

nb_user

3

Pour calculer le nombre total d'achats effectués sur le site par les personnes se nommant Paul :

```
SELECT SUM(nombre_achat) AS achat_tot FROM utilisateur WHERE nom = 'Paul'
```

achat_tot

0

Il est aussi possible de regrouper les enregistrements d'une table par agrégation à l'aide du mot-clef `GROUP BY`. Ce regroupement permet d'appliquer la fonction à chacun des groupes et le résultat de la requête est l'ensemble des valeurs prises par la fonction sur chacun des regroupements.

La syntaxe est (où `a` et `b` sont des attributs et `f` est la fonction d'agrégation) :

```
SELECT f(a) FROM table WHERE expression_logique GROUP BY b
```

Exemple

Poursuivons le même exemple que ci-dessus. Pour calculer le nombre d'achats effectués par chaque nom, on regroupe les lignes :

```
SELECT nom, SUM(nombre_achat) AS achat_tot FROM utilisateur GROUP BY nom
```

nom	achat_tot
Marie	5
Léon	4
Paul	0

Exemple

Reprenons l'exemple initial de ce paragraphe : une requête permettant de calculer les moyennes des notes sur chaque classe s'écrit :

```
SELECT classe, AVG(note) AS moyenne FROM relevé GROUP BY classe
```

classe	moyenne
PC	13,38
PSI	11,08

Lorsque l'on impose une contrainte (par exemple une sélection) **après** avoir utilisé une fonction d'agrégation, en particulier après `GROUP BY`, il faut utiliser le mot-clé `HAVING` et non `WHERE`.

Exemple

Considérons la base de donnée **world**.

Pour obtenir la population de chacun des continents (dans l'ordre décroissant), on réalise la requête :

```
SELECT Continent, SUM(Population) AS Pop FROM country GROUP BY Continent ORDER BY Pop DESC
```

Si l'on souhaite maintenant se restreindre aux continents comportant plus de 40 pays, on écrit :

```
SELECT Continent, SUM(Population), COUNT(*) AS NombreDePays FROM country
GROUP BY Continent HAVING NombreDePays > 40
```

IV - Des exercices issus de sujets de concours

IV.1 - MP-PC-PSI Mines-Ponts 2017

On modélise un réseau routier par un ensemble de croisements et de voies reliant ces croisements. Les voies partent d'un croisement et arrivent à un autre croisement. Ainsi, pour modéliser une route à double sens, on utilise deux voies circulant en sens opposés. La base de données du réseau routier est constituée des relations suivantes :

- Croisement(id, longitude, latitude)
- Voie(id, longueur, id_croisement_debut, id_croisement_fin)

Dans la suite on considère c l'identifiant (id) d'un croisement donné.

Question 1 - Écrire la requête SQL qui renvoie les identifiants des croisements atteignables en utilisant une seule voie à partir du croisement ayant l'identifiant c .

Question 2 - Écrire la requête SQL qui renvoie les longitudes et latitudes des croisements atteignables en utilisant une seule voie, à partir du croisement c .

Question 3 - Que renvoie la requête SQL suivante?

```

1 SELECT V2.id_croisement_fin
2 FROM Voie as V1 JOIN Voie as V2
3 ON V1.id_croisement_fin = V2.id_croisement_debut
4 WHERE V1.id_croisement_debut = c

```

IV.2 - MP-PC-PSI Centrale-Supélec 2016

Ce problème s'intéresse à différents aspects relatifs à la sécurité aérienne et plus précisément au risque de collision entre deux appareils.

Afin d'éviter les collisions entre avions, les altitudes de vol en croisière sont normalisées. Dans la majorité des pays, les avions volent à une altitude multiple de 1000 pieds (un pied vaut 30,48 cm) au-dessus de la surface isobare à 1013,25 hPa. L'espace aérien est ainsi découpé en tranches horizontales appelées niveaux de vol et désignées par les lettres « FL » (*flight level*) suivies de l'altitude en centaines de pieds : « FL310 » désigne une altitude de croisière de 31000 pieds au-dessus de la surface isobare de référence.

EUROCONTROL est l'organisation européenne chargée de la navigation aérienne, elle gère plusieurs dizaines de milliers de vol par jour. Toute compagnie qui souhaite faire traverser le ciel européen à un de ses avions doit soumettre à cet organisme un plan de vol comprenant un certain nombre d'informations : trajet, heure de départ, niveau de vol souhaité, etc. Muni de ces informations, Eurocontrol peut prévoir les secteurs aériens qui vont être surchargés et prendre des mesures en conséquence pour les désengorger : retard au décollage, modification de la route à suivre, etc. Nous modélisons (de manière très simplifiée) les plans de vol gérés par EUROCONTROL sous la forme d'une base de données comportant deux tables :

- la table `vol` qui répertorie les plans de vol déposés par les compagnies aériennes ; elle contient les colonnes
 - `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 (de type date, affiché au format `aaaa-mm-jj`) ;
 - `heure` : heure de décollage souhaitée (de type time, affiché au format `hh :mi`) ;
 - `niveau` : niveau de vol souhaité (entier).

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

Figure 1 Extrait de la table `vol` : vols de la compagnie Air France entre les aéroports Charles-de-Gaule (Paris) et Léonard-de-Vinci à Fiumicino (Rome)

- la table `aeroport` qui répertorie les aéroports européens ; elle contient les colonnes
 - `id_aero` : code de l'aéroport (chaîne de caractères) ;
 - `ville` : principale ville desservie (chaîne de caractères) ;

- pays : pays dans lequel se situe l'aéroport (chaîne de caractères).

id_aero	ville	pays
CDG	Paris	France
ORY	Paris	France
MRS	Marseille	France
FCO	Rome	Italie

Figure 2 Extrait de la table aeroport

Les types SQL `date` et `time` permettent de mémoriser respectivement un jour du calendrier grégorien et une heure du jour. Deux valeurs de type `date` ou de type `time` peuvent être comparées avec les opérateurs habituels(=, <, <=, etc.). La comparaison s'effectue suivant l'ordre chronologique. Ces valeurs peuvent également être comparées à une chaîne de caractères correspondant à leur représentation externe ('aaaa-mm-jj' ou 'hh :mi').

Question 1 - Écrire une requête SQL qui fournit le nombre de vols qui doivent décoller dans la journée du 2 mai 2016 avant midi.

Question 2 - Écrire une requête SQL qui fournit la liste des numéros de vols au départ d'un aéroport desservant Paris le 2 mai 2016

Question 3 - Que fait la requête suivante ?

```

1 SELECT id_vol
2 FROM vol JOIN aeroport AS d ON d.id_aero = depart
3         JOIN aeroport AS a ON a.id_aero = arrivee
4 WHERE
5     d.pays = 'France' AND
6     a.pays = 'France' AND
7     jour = '2016-05-02'
```

Question 4 - Certains vols peuvent engendrer des conflits potentiels : c'est par exemple le cas lorsque deux avions suivent un même trajet, en sens inverse, le même jour et à un même niveau. Écrire une requête SQL qui fournit la liste des couples (Id₁ , Id₂) des identifiants des vols dans cette situation.

IV.3 - PC Mines Ponts 2019

Chiffrer les données est nécessaire pour assurer la confidentialité lors d'échanges d'informations sensibles. Dans ce domaine, les nombres premiers servent de base au principe de clés publique et privée qui permettent, au travers d'algorithmes, d'échanger des messages chiffrés. La sécurité de cette méthode de chiffrement repose sur l'existence d'opérations mathématiques peu coûteuses en temps d'exécution mais dont l'inversion (c'est-à-dire la détermination des opérandes de départ à partir du résultat) prend un temps exorbitant. On appelle ces opérations "fonctions à sens unique". Une telle opération est, par exemple, la multiplication de grands nombres premiers. Il est aisé de calculer leur produit. Par contre, connaissant uniquement ce produit, il est très difficile de déduire les deux facteurs premiers.

[...]

Les questions portant sur les bases de données sont à traiter en langage SQL.

Au cours du développement des fonctions nécessaires à la manipulation des nombres premiers on s'aperçoit que le choix des algorithmes pour évaluer chaque fonction est primordial pour garantir des performances acceptables. On souhaite donc mener des tests à grande échelle pour évaluer les performances réelles du code qui a été développé. Pour ce faire on effectue un grand nombre de tests sur une multitude d'ordinateurs. Les données sont ensuite centralisées dans une base de données composée de deux tables.

La première table est `ordinateurs` et permet de stocker des informations sur les ordinateurs utilisés pour les tests. Ses attributs sont :

- nom TEXT, clé primaire, le nom de l'ordinateur.
- gflops INTEGER la puissance de l'ordinateur en milliards d'opérations flottantes par seconde.
- ram INTEGER la quantité de mémoire vive de l'ordinateur en Go.

Exemple du contenu de cette table :

nom	gflops	ram
nyarlathotep114	69	32
nyarlathotep119	137	32
...		
shubniggurath42	133	16
azathoth137	85	8

La seconde table est `fonctions` et stocke les informations sur les tests effectués pour différentes fonctions en cours de développement. Ses attributs sont :

- `id` INTEGER l'identifiant du test effectué.
- `nom` TEXT le nom de la fonction testée (par exemple `li`, `Ei`, etc).
- `algorithme` TEXT le nom de l'algorithme qui permet le calcul de la fonction testée (par exemple `BBS` si on teste une fonction de génération de nombres aléatoires).
- `teste_sur` TEXT le nom du PC sur lequel le test a été effectué.
- `temps_exec` INTEGER le temps d'exécution du test en millisecondes.

Exemple du contenu de cette table :

id	nom	algorithme	teste_sur	temps_exec
1	li	rectangles	nyarlathotep165	2638
2	li	rectangles	shubniggurath28	736
3	li	trapezes	nyarlathotep165	4842
...				
2154	Ei	puiseux	nyarlathotep145	2766
2155	aleatoire	BBS	azathoth145	524

Question 1 Expliquer pourquoi il n'est pas possible d'utiliser l'attribut `nom` comme clé primaire de la table `fonctions`.

Question 2 Écrire des requêtes SQL permettant de :

1. Connaître le nombre d'ordinateurs disponibles et leur quantité moyenne de mémoire vive.
2. Extraire les noms des PC sur lesquels l'algorithme `rectangles` n'a pas été testé pour la fonction nommée `li`.
3. Pour la fonction nommée `Ei`, trier les résultats des tests du plus lent au plus rapide. Pour chaque test retenir le nom de l'algorithme utilisé, le nom du pc sur lequel il a été effectué et la puissance du PC.