PSI* 2021/22 Exos d'info 4

# Bases de données aux concours

**Proposition de "top 3" :** ce sont les plus riches, donc aussi parmi les plus difficiles sans doute.

— Centrale 2020

— X 2019

— X 2016

*Remarque : pas de base de données in X2021 ; pas de sujet X2020 du tout (épreuve annulée).*

## Sujet 1. Centrale 2021 -

*Je n'ai pas de source LaTeX - voir le sujet original*

## Sujet 2. Mines 2021 - Marchons, marchons, marchons

Lors de la préparation d'une randonnée, une accompagnatrice doit prendre en compte les exigences des participants. Elle dispose d'informations rassemblées dans deux tables d'une base de données :

— La table `Rando` décrit les randonnées possibles - la clef primaire entière `rid`, le nom du parcours, son niveau de difficulté (entier de 1 à 5), le dénivelé en mètres, la durée moyenne en minutes

| `rid` | `rnom` | `diff` | `deniv` | `duree` |
|---|---|---|---|---|
| 1 | La belle des champs | 1 | 20 | 30 |
| 2 | Lac de Castellagne | 4 | 650 | 150 |
| 3 | Le tour du mont | 2 | 200 | 120 |
| 4 | Les crêtes de la mort | 5 | 1200 | 360 |
| 5 | Yukon ho ! | 3 | 700 | 210 |
| ... | ... | ... | ... | ... |

— La table `Participant` décrit les randonneurs - la clef primaire entière `pid`, le nom du randonneur, son année de naissance, le niveau de difficulté maximum de ses randonnées

| `pid` | `pnom` | `ne` | `diff_max` |
|---|---|---|---|
| 1 | Calvin | 2014 | 2 |
| 2 | Hobbes | 2015 | 2 |
| 3 | Susie | 2014 | 2 |
| 4 | Rosalyn | 2001 | 4 |
| ... | ... | ... | ... |

Donner une requête SQL de cette base pour

**Q1.** Compter le nombre de participants nés entre 1999 et 2003 inclus

**Q2.** Calculer la durée moyenne des randonnées pour chaque niveau de difficulté.

**Q3.** Extraire le nom des participants pour lesquels la randonnée n°42 est trop difficile.

**Q4.** Extraire les clés primaires des randonnées qui ont un ou des homonymes (nom identique et clé primaire distincte), sans redondance.

## Sujet 3. Centrale 2020 - Photomosaïque

*Je n'ai pas de source LaTeX, voir la dernière page de cette fiche d'exos, rajoutée à la main.*

## Sujet 4. Mines 2020 - Images de vagues et de structures

*Je n'ai pas de source LaTeX, voir l'énoncé originel p.4*

## Sujet 5. X 2019 - TetriX

On souhaite utiliser une base de données pour stocker les résultats obtenus par une communauté de joueurs. On suppose que l'on dispose d'une base de données comportant les tables JOUEURS(id_j, nom, pays) et PARTIES(id_p, date, duree, score, id_joueur) où :

**id_j** , de type entier, est la clé primaire de la table JOUEURS,

**nom** est une chaîne de caractères donnant le nom du joueur,

**pays** est une chaîne de caractères donnant le pays du joueur,

**id_p** , de type entier, est la clé primaire de la table PARTIES,

**date** est la date (AAAAMMJJ) de la partie,

**duree** , de type entier, est la durée en secondes de la partie,

**score** , de type entier, est le nombre de points marqués au cours de la partie,

**id_joueur** est un entier qui identifie le joueur de la partie.

**Question 16.** Étant donné une chaîne de caractères cc contenant le nom d'un joueur, écrire une requête SQL qui renvoie la date, la durée et le score de toutes les parties jouées par le joueur cc, listées par ordre chronologique (au choix, croissant ou décroissant).

**Question 17.** Étant donné un entier s (le score que vient de réaliser une joueuse nommée Alice), écrire une requête SQL qui renvoie la position qu'aura le score s dans le classement des parties par ordre de score (on suppose que la dernière partie d'Alice n'a pas encore été insérée dans la table des parties). En cas d'ex aequo pour le score s (une ou plusieurs parties déjà présentes ayant le score s), le rang sera le même que s'il n'y avait qu'une seule partie avec le score s. Par exemple, la requête renverra 1 (le score d'Alice est "1er") si aucun score n'est meilleur que s. Autre exemple, si la base de données contient 6 parties dont les scores sont 87 ; 75 ; 75 ; 63 ; 60 ; 60, alors le rang de s = 75 sera 2, le rang de s = 70 sera 4 et le rang de s = 60 sera 5.

**Question 18.** Écrire une requête SQL qui renvoie le record de France de Tetris couleur, c'est-à-dire le meilleur score réalisé par un joueur dont le pays est la France.

**Question 19.** Étant donné une chaîne de caractères cc contenant le nom d'un joueur (ayant déjà joué au moins une partie de Tetris couleur), écrire une requête SQL qui renvoie le rang du joueur cc, c'est-à-dire sa position dans le classement des joueurs par ordre de leur meilleur score dans une partie de Tetris couleur (on traitera les ex aequo de la même manière qu'à la question 17).

## Sujet 6. Mines 2019 - Performance des algorithmes pour les nombres premiers

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 'a grande échelle pour évaluer les performances réelles du code qui a été développé. Pour ce faire on exécute 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 exécutés pour différentes fonctions en cours de développement. Ses attributs sont :

• `id INTEGER` l'identifiant du test exécuté.

• `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 exécuté.

• `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 |

**Q.25.** Expliquer pourquoi il n'est pas possible d'utiliser l'attribut nom comme clé primaire de la table fonctions.

**Q.26.** 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 exécuté et la puissance du PC.

## Sujet 7. X 2018 - base de données

Le sujet portait totalement sur les bases de données, mais l'objectif était de programmer en Python les instructions classiques du langage SQL. A titre d'utilisation il y avait notamment 5 requêtes SQL à traduire avec les fonctions Python que les candidats avaient écrites.

## Sujet 8. Centrale 2018 - cinétique des gaz parfaits

On dispose d'une version plus générale de la fonction `simulation` pour laquelle toutes les particules ne sont plus nécessairement identiques. Cette fonction enregistre ses résultats dans une base de données dont la structure est donnée figure 4.

| SIMULATION | |
|---|---|
| SI_NUM | integer |
| SI_DEB | datetime |
| SI_DUR | float |
| SI_DIM | integer |
| SI_N | integer |
| SI_L | float |

| REBOND | |
|---|---|
| SI_NUM | integer |
| RE_NUM | integer |
| PA_NUM | integer |
| RE_T | float |
| RE_DIR | integer |
| RE_VIT | float |
| RE_P | float |

| PARTICULE | |
|---|---|
| PA_NUM | integer |
| PA_NOM | varchar(100) |
| PA_M | float |
| PA_R | float |

**Figure 4** Structure physique de la base de données des résultats de simulation

Cette base comporte les trois tables suivantes :

— la table SIMULATION donne les caractéristiques de chaque simulation effectuée. Elle contient les colonnes
- SI_NUM numéro d'ordre de la simulation (clef primaire)
- SI_DEB date et heure du lancement du programme de simulation
- SI_DUR durée (en secondes) de la simulation (il ne s'agit pas du temps d'exécution du programme, mais du temps simulé)
- SI_DIM nombre de dimensions de l'espace de simulation
- SI_N nombre de particules pour cette simulation
- SI_L (en mètres) taille du récipient utilisé pour la simulation

— la table PARTICULE des types de particules considérées. Elle contient les colonnes
- PA_NUM numéro (entier) identifiant le type de particule (clef primaire)
- PA_NOM nom de ce type de particule
- PA_M masse de la particule (en grammes)
- PA_R rayon (en mètres) de la particule

— la table REBOND, de clef primaire (SI_NUM, RE_NUM), liste les chocs des particules avec les parois du récipient. Elle contient les colonnes
- SI_NUM numéro d'ordre de la simulation ayant généré ce rebond
- RE_NUM numéro d'ordre du rebond au sein de cette simulation
- PA_NUM numéro du type de particule concernée par ce rebond
- RE_T temps de simulation (en secondes) auquel ce rebond est arrivé
- RE_DIR paroi concernée : entier non nul de l'intervalle [-SI_DIM, SI_DIM] donnant la direction de la normale à la paroi. Ainsi $-2$ désigne la paroi située en $y = 0$ alors que 1 désigne la paroi située en $x = L$
- RE_VIT norme de la vitesse de la particule qui rebondit (en $\mathrm{m} \cdot \mathrm{s}^{-1}$)
- RE_VP valeur absolue de la composante de la vitesse normale à la paroi (en m·s )

**Q.34.** Ecrire une requête SQL qui donne le nombre de simulations effectuées pour chaque nombre de dimensions de l'espace de simulation.

**Q.35.** Ecrire une requête SQL qui donne, pour chaque simulation, le nombre de rebonds enregistrés et la vitesse moyenne des particules qui frappent une paroi.

**Q.36.** Ecrire une requête SQL qui, pour une simulation $n$ donnée, calcule, pour chaque paroi, la variation de quantité de mouvement due aux chocs des particules sur cette paroi tout au long de la simulation. On se rappellera que lors du rebond d'une particule sur une paroi la composante de sa vitesse normale à la paroi est inversée, ce qui correspond à une variation de quantité de mouvement de $2m|v_\perp|$ où $m$ désigne la masse de la particule et $v_\perp$ la composante de sa vitesse normale à la paroi.

## Sujet 9. Mines 2018 - Mesure de la houle par des capteurs sur des bouées

Plusieurs indicateurs sont couramment considérés pour définir l'état de la mer. Parmi eux, on note :

— $H_{max}$ : la hauteur de la plus grande vague observée sur l'intervalle d'enregistrement $[0, T]$ ;

— $H_{1/3}$ : la valeur moyenne des hauteurs du tiers supérieur des plus grandes vagues observées sur $[0, T]$ ;

— $T_{H_{1/3}}$ : la valeur moyenne des périodes du tiers supérieur des plus grandes vagues observées sur $[0, T]$.

On dispose d'une base de données relationnelle `Vagues`.

La première table est `Bouee`. On se limite aux attributs suivants : le numéro d'identification `idBouee`, le nom du site `nomSite`, le nom de la mer ou de l'océan `localisation`, le type du capteur `typeCapteur` et la fréquence d'échantillonnage `frequence`.

`Bouee`

| idBouee | nomSite | localisation | typeCapteur | frequence |
|---|---|---|---|---|
| 831 | Porquerolles | Mediterranee | Datawell non directionnelle | 2.00 |
| 291 | Les pierres noires | Mer d'iroise | Datawell directionnelle | 1.28 |
| ... | ... | ... | ... | ... |

La seconde table est `Campagne`. On se limite aux attributs suivants : le numéro d'identification `idCampagne`, le numéro d'identification de la bouée `idBouee`, la date de début `debutCampagne` et la date de fin `finCampagne`.

`Campagne`

| idCampagne | idBouee | debutCampagne | finCampagne |
|---|---|---|---|
| 08301 | 831 | 01/01/2010 00h00 | 15/01/2010 00h00 |
| 02911 | 291 | 15/10/2005 18h30 | 18/10/2005 08h00 |
| ... | ... | ... | ... |

La troisième table est `Tempete`. Les informations fournies relatives à un évènement « tempête »sont les suivantes :

— date de début et fin de tempête ;

— évolution des paramètres $H_{1/3}$ et $H_{max}$ en fonction du temps

— Le détail de certains paramètre non définis ici, obtenus au pic de tempête.

On se limite aux attributs suivants : le numéro d'identification de la tempête `idTempete`, le numéro d'identification de la bouéee `idBouee`, la date de début `debutTempete`, la date de fin `finTempete`, la valeur maximale de hauteur de vague `Hmax`.

`Tempete`

| idTempete | idBouee | debutTempete | finTempete | Hmax |
|---|---|---|---|---|
| 083010 | 831 | 07/01/2010 20h00 | 09/01/2010 15h30 | 5.3 |
| 029012 | 291 | 16/10/2005 08h30 | 18/10/2005 09h00 | 8.5 |
| ... | ... | ... | ... | ... |

Le schéma de la base de donnée est donc : `Vagues=Bouee,Campagne,Tempete`.

**Q19.** Formuler les requêtes SQL permettant de répondre aux questions suivantes :

— « Quels sont le numéro d'identification et le nom de site des bouées localisées en Méditerrannée ? »

— « Quel est le numéro d'identification des bouées où il n'y a pas eu de tempêtes ? »

— « Pour chaque site, quelle est la hauteur maximale enregistrée lors d'une tempête ? »

## Sujet 10. X 2017 - ensemble de points

On suppose que l'on représente des points à coefficients entiers dans le plan à l'aide d'une base de données. Cette base comporte deux tables. La tables `POINTS` contient trois colonnes :

- `id` (clé primaire) qui est un entier naturel unique représentant le point ;
- `x` qui est un entier naturel représentant son abscisse ;
- `y` qui est un entier naturel représentant son ordonnée.

On suppose qu'il n'existe pas deux points d'identifiants distincts et de mêmes coordonnées.

La relation d'appartenance à un ensemble de points estreprésentée par la table `MEMBRE` à deux colonnes :

- `idpoint`, un entier naturel qui identifie un point ;
- `idensemble`, un entier naturel qui identifie un ensemble de points.

4. Ecrire une requête SQL qui renvoie les identifiants des ensembles auxquels appartient le point de coordonnées $(a, b)$.
5. Ecrire une requête SQL qui renvoie les coordonnées des points qui appartiennent à l'intersection des ensembles d'identifiants $i$ et $j$.
6. Ecrire une requête SQL qui renvoie les identifiants des points appartenant à au moins un des ensembles auxquels appartient le point de coordonnées $(a, b)$.

## Sujet 11. Centrale 2017 - Exploration martienne

*Je n'ai pas de source LaTeX, voir le sujet originel p.3-4-5.*

## Sujet 12. Mines 2017 - Trafic routier

On modélise ici 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.

**Q26.** Écrire la requête SQL qui renvoie les identifiants des croisements atteignables en utilisant une seule voie à partir du croisement ayant l'identifiant c.

**Q27.** Écrire la requête SQL qui renvoie les longitudes et latitudes des croisements atteignables en utilisant une seule voie, à partir du croisement c.

**Q28.** Que renvoie la requête SQL suivante ?

```
SELECT V2.id_croisement_fin
FROM Voie as V1
JOIN Voie as V2
ON V1.id_croisement_fin = V2.id_croisement_debut
WHERE V1.id_croisement_debut = c
```

## Sujet 13. X MP PC 2016 - réseaux sociaux

Une représentation simplifiée, réduite à deux tables, de la base de données d'un réseau social est donnée dans la figure suivante

*INDIVIDUS*

| *id* | *nom* | *prenom* |
|---|---|---|
| 1 | *Potter* | *Harry* |
| 2 | *Granger* | *Hermione* |
| | ... | |

*LIENS*

| *id*1 | *id*2 |
|---|---|
| 1 | 2 |
| 2 | 1 |
| ... | ... |

La table INDIVIDUS répertorie les individus et contient les colonnes
- id (clé primaire), un entier identifiant chaque individu ;
- nom, une chaîne de caractères donnant le nom de famille de l'individu ;
- prenom, une chaîne de caractères donnant le prénom de l'individu.

La table LIENS répertorie les liens d'amitiés entre individus et contient les colonnes
- id1, entier identifiant le premier individu du lien d'amitié ;
- id2, entier identifiant le second individu du lien d'amitié.

On supposera par ailleurs que pour tout couple (x,y) dans la table LIENS, le couple (y,x) est également présent dans la table.

Q16. Ecrire une requête SQL qui renvoie les identifiants des amis de l'individu d'identifiant x.

Q17. Ecrire une requête SQL qui renvoie les nom et prénom de chacun des amis de l'individu d'identifiant x.

Q18. Ecrire une requête SQL qui renvoie les identifiants des individus qui sont amis avec au moins un ami de l'individu d'identifiant x.

## Sujet 14. Centrale 2016 : prévenir les collisions aériennes

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 (chaine de caractères) ;
- `depart` : code de l'aéroport de départ (chaine de caractères) ;
- `arrivee` : code de l'aéroport d'arrivée (chaine 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 (chaine de caractères) ;
- `ville` : principale ville desservie (chaine de caractères) ;
- `pays` : pays dans lequel se situe l'aéroport (chaine 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 chaine de caractères correspondant à leur représentation externe (`aaaa-mm-jj` ou `hh:mi`).

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

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

**I.C** - Que fait la requête suivante ?

```
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'
```

**I.D** - 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 ($Id_1$ , $Id_2$) des identifiants des vols dans cette situation.

## Sujet 15. Mines 2016 : propagation d'épidémies

Pour suivre la propagation des épidémies, de nombreuses données sont recueillies par les institutions internationales comme l'OMS. Par exemple, pour le paludisme, on dispose de deux tables :

— La table `palu` recense le nombre de nouveaux cas confirmés et le nombre de décès liés au paludisme ; certaines lignes de cette table sont données en exemple (on précise que `iso` est un identifiant unique pour chaque pays) :

| nom | iso | annee | cas | deces |
|---|---|---|---|---|
| Bresil | BR | 2009 | 309 316 | 85 |
| Bresil | BR | 2010 | 334 667 | 76 |
| Kenya | KE | 2010 | 898 531 | 26 017 |
| Mali | ML | 2011 | 307 035 | 2 128 |
| Ouganda | UG | 2010 | 1 581 160 | 8 431 |
| ... | | | | |

— la table `demographie` recense la population totale de chaque pays ; certaines lignes de cette table sont données en exemple :

| pays | periode | pop |
|---|---|---|
| BR | 2009 | 193 020 000 |
| BR | 2010 | 194 946 000 |
| KE | 2010 | 40 909 000 |
| ML | 2011 | 14 417 000 |
| UG | 2010 | 33 987 000 |
| ... | | |

**Q5.** Au vu des données présentées dans la table `palu`, parmi les attributs `nom, iso` et `annee`, quels attributs peuvent servir de clé primaire ? Un couple d'attributs pourrait-il servir de clé primaire ? (on considère qu'une clé primaire peut posséder plusieurs attributs). Si oui, en préciser un.

**Q6.** Écrire une requête en langage SQL qui récupère depuis la table `palu` toutes les données de l'année 2010 qui correspondent à des pays où le nombre de décès dus au paludisme est supérieur ou égal à 1 000.

On appelle *taux d'incidence d'une épidémie* le rapport du nombre de nouveaux cas pendant une période donnée sur la taille de la population-cible pendant la même période. Il s'exprimer généralement en « nombre de nouveaux cas pour 100 000 personnes par année ». Il s'agit d'un des critères les plus importants pour évaluer la fréquence et la vitesse d'apparition d'une épidémie.

**Q7.** Écrire une requête en langage SQL qui détermine le taux d'incidence du paludisme en 2011 pour les différents pays de la table `palu`.

**Q8.** Écrire une requête en langage SQL permettant de déterminer le nom du pays ayant eu le deuxième plus grand nombre de nouveaux cas de paludisme en 2010 (on pourra supposer qu'il n'y a pas de pays *ex æquo* pour les nombres de cas).

## Sujet 16. Centrale 2015 - Autour de la dynamique gravitationnelle

À partir de mesures régulièrement effectuées par différents observatoires, une base de données des caractéristiques et des états des corps célestes de notre Système solaire est maintenue à jour. L'objectif de cette partie est d'extraire de cette base de données les informations nécessaires à la mise en ?uvre des fonctions développées dans les parties précédentes, puis de les utiliser pour prévoir les positions futures des différentes planètes. Les données à extraire sont les masses des corps étudiés et leurs états (position et vitesse) à l'instant $t_{min}$ du début de la simulation. Une version simplifiée, réduite à deux tables, de la base de données du Système solaire est donnée figure 3. Les masses sont exprimées en kilogrammes, les distances en unités astronomiques (1 au = $1,5.10^{11}$ m) et les vitesses en kilomètres par seconde. Le référentiel utilisé pour exprimer les composantes des positions et des vitesses est galiléen, orthonormé et son centre est situé à proximité du Soleil.

La table CORPS répertorie les corps étudiés, elle contient les colonnes

— id_corps (clé primaire) entier identifiant chaque corps ;

— nom, chaine de caractères, désigne le nom usuel du corps ;

— masse de type flottant, contient la masse du corps.

La table ETAT rassemble l'historique des états successifs (positions et vitesses) des corps étudiés. Elle est constituée de huit colonnes :

— id_corps de type entier, identifie le corps concerné ;

— datem est la date de la mesure, sous forme d'un entier donnant le nombre de secondes écoulées depuis un instant d'origine ;

— trois colonnes de type flottant pour les composantes de la position x, y, z ;

— trois colonnes de type flottant pour les composantes de la vitesse vx, vy, vz.

**IV.A** Écrire une requête SQL qui renvoie la liste des masses de tous les corps étudiés.

**IV.B** Les états des différents corps ne sont pas forcément tous déterminés exactement au même instant. Nous allons assimiler l'état initial (à la date $t_{min}$) de chaque corps à son dernier état connu antérieur à $t_{min}$.

Dans toute la suite, on supposera que la valeur de $t_{min}$, sous le format utilisé dans la table ETAT, est accessible à toute requête SQL via l ?expression tmin().

**IV.B1)** On souhaite d'abord vérifier que tous les corps étudiés disposent d'un état connu antérieur à tmin().

Le nombre de corps présents dans la base est obtenu grâce à la requête SELECT count(*) FROM corps. Écrire une requête SQL qui renvoie le nombre de corps qui ont au moins un état connu antérieur à tmin().

**IV.B2)** Écrire une requête SQL qui renvoie, pour chaque corps, son identifiant et la date de son dernier état antérieur à tmin().

**IV.B3)** Le résultat de la requête précédente est stocké dans une nouvelle table date_mesure à deux colonnes :

— id_corps de type entier, contient l'identifiant du corps considéré ;

— date_der de type entier, correspond à la date du dernier état connu du corps considéré, antérieur à tmin().

Pour simplifier la simulation, on décide de négliger l'influence des corps ayant une masse strictement inférieure à une valeur fixée masse_min() et de ne s'intéresser qu ?aux corps situés dans un cube, centré sur l'origine du référentiel de référence et d'arête arete() donnée. Les faces de ce cube sont parallèles aux plans formés par les axes du référentiel de référence.

Écrire une requête SQL qui renvoie la masse et l'état initial (sous la forme masse, x, y, z, vx, vy, vz) de chaque corps retenu pour participer à la simulation. Classez les corps dans l'ordre croissant par rapport à leur distance à l'origine du référentiel.

## Sujet 17. Mines 2015 - Tests de validation d'une imprimante

Une représentation simpliée de deux tables de la base de données qu'on souhaite utiliser est donnée ci-dessous :

`testfin`

| nSerie | dateTest | ... | Imoy | Iec | ... | fichierMes |
|---|---|---|---|---|---|---|
| 230-588ZX2547 | 2012-04-22 14-25-45 | | 0.45 | 0.11 | | mesure31025.csv |
| 230-588ZX2548 | 2012-04-22 14-26-57 | | 0.43 | 0.12 | | mesure41026.csv |
| ... | ... | ... | ... | ... | ... | ... |

`production`

| Num | nSerie | dateProd | type |
|---|---|---|---|
| 20 | 230-588ZX2547 | 2012-04-22 15-52-12 | JETDESK-1050 |
| 21 | 230-588ZX2549 | 2012-04-22 15-53-24 | JETDESK-3050 |
| ... | ... | ... | ... |

Après son assemblage et avant les différents tests de validation, un numéro de série unique est attribué à chaque imprimante. A la fin des tests de chaque imprimante, les résultats d'analyse ainsi que le fichier contenant l'ensemble des mesures réalisées sur l'imprimante sont rangées dans la table testfin. Lorsqu'une imprimante satisfait les critères de validation, elle est enregistrée dans la table production avec son numéro de série, la date et l'heure de sortie de production ainsi que son type.

**Q8.** Rédiger une requête SQL permettant d'obtenir les numéros de série des imprimantes ayant une valeur de Imoy comprise strictement entre deux bornes Imin et Imax.

**Q9.** Rédiger une requête SQL permettant d'obtenir les numéros de série, la valeur de l'écart type et le fichier de mesures des imprimantes ayant une valeur de Iec strictement inférieure à la valeur moyenne de la colonne Iec.

**Q10.** Rédiger une requête SQL qui permettra d'extraire à partir de la table testfin le numéro de série et le fichier de mesures correspondant aux imprimantes qui n'ont pas été validées en sortie de production.