Le SQL aux concours

1 X 2016

Une représentation simplifiée, réduite à deux tables, de la base de données d'un réseau social est donnée dans la figure 1.

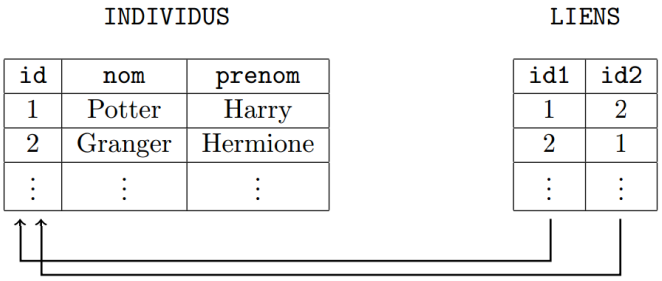

FIGURE 1: Schéma de la base de données du réseau social

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 (contrairement à la partie précédente de cet énoncé).

Question 1. Écrire une requête SQL qui renvoie les identifiants des amis de l'individu d'identifiant x.

Question 2. Écrire une requête SQL qui renvoie les (noms,prénoms) des amis de l'individu d'identifiant x.

Question 3. Écrire une requête SQL qui renvoie les identifiants des individus qui sont amis avec au moins un ami de l'individu d'identifiant x.

2 Centrale 2016

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 (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 2: Extrait de la table `vol` : vols de la compagnie Air France entre les aéroports Charles-de-Gaulle (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 3: 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'`).

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 ?

```
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'
```

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.

3 Mines 2016

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
...		

Question 1. 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.

Question 2. 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.

Question 3. Écrire une requête en langage SQL qui détermine le taux d'incidence du paludisme en 2011 pour les différents pays de la table `palu`.

Question 4. 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).

On considère la requête R qui s'écrit dans le langage de l'algèbre relationnelle :

$$R = \pi_{\texttt{nom,deces}}(\sigma_{\texttt{annee}=2010}(\texttt{palu}))$$

On suppose que le résultat de cette requête a été converti en une liste Python stockée dans la variable `deces2010` et constituée de couples (chaîne,entier).

Question 5. Quelle instruction peut-on écrire en Python pour trier la liste `deces2010` par ordre croissant du nombre de décès dus au paludisme en 2010 ?

4 X 2017

On suppose maintenant que l'on représente les points du problème à l'aide d'une base de données. Cette base de données comporte deux tables. La table `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 d'un point à un ensemble de points est représentée 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

Question 6. Écrire une requête SQL qui renvoie les identifiants des ensembles auxquels appartient le point de coordonnées (a, b).

Question 7. Écrire une requête SQL qui renvoie les coordonnées des points qui appartiennent à l'intersection des ensembles d'identifiants i et j.

Question 8. Écrire une requête SQL qui renvoie les identifiants des points appartenant à au moins un des ensembles auxquels appartient le point de coordonnées (a, b).

5 Centrale 2017

Afin d'assurer son autonomie opérationnelle, le robot dispose localement des informations nécessaires à son fonctionnement quotidien. Ainsi, il enregistre la durée d'utilisation de ses différents instruments embarqués. Il connait également les différents types d'analyses qu'il peut effectuer et, pour chacun de ces types, les instruments à utiliser. Il enregistre la prochaine exploration, c'est-à-dire les différents points d'intérêts qu'il doit visiter et pour chacun la ou les analyses qu'il doit effectuer. D'autre part, il conserve les résultats d'analyses effectuées lors de ses explorations passées. Ces résultats ne sont effacés qu'après confirmation de leur bonne transmission sur Terre. Ces différentes informations sont stockées dans une base de données relationnelle dont le modèle physique est schématisé figure 4

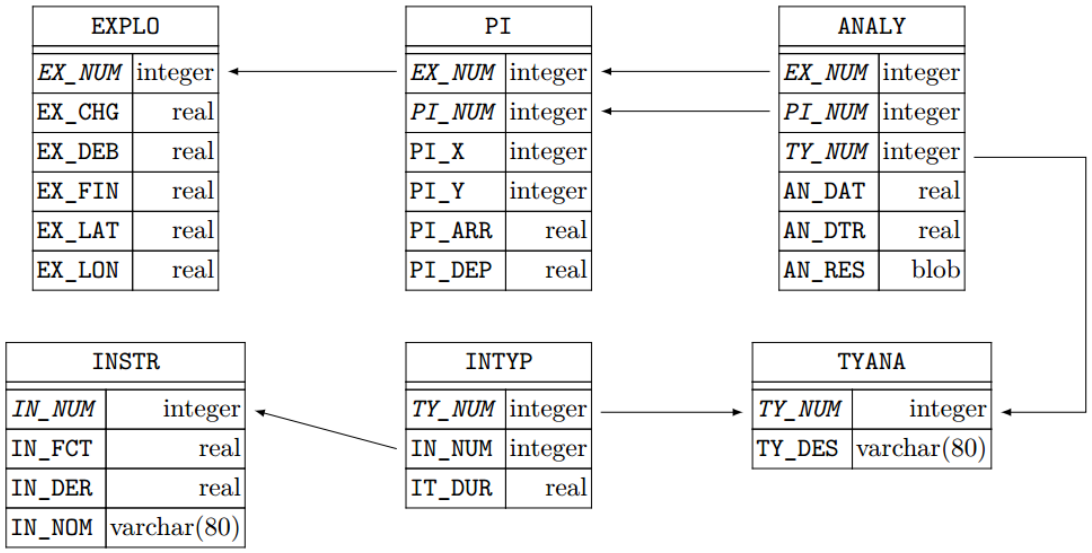

FIGURE 4: Structure physique de la base de données d'un robot

Cette base comporte les six tables suivantes :

— la table `EXPLO` des explorations, avec les colonnes

- `EX_NUM` numéro (entier) de l'exploration (clef primaire)
- `EX_CHG` date de transmission des points d'intérêts de cette exploration
- `EX_DEB` date de début de l'exploration (`NULL` si l'exploration n'est pas encore commencée)
- `EX_FIN` date de fin de l'exploration (`NULL` si l'exploration n'est pas encore terminée)
- `EX_LAT` latitude (en degrés décimaux) du point de coordonnées `(0, 0)` de la zone d'exploration
- `EX_LON` longitude (en degrés décimaux) du point de coordonnées `(0, 0)` de la zone d'exploration

— la table `PI` des points d'intérêts, de clef primaire `(EX_NUM, PI_NUM)`, avec les colonnes

  - `EX_NUM` numéro de l'exploration à laquelle appartient le point d'intérêt
  - `PI_NUM` numéro du point d'intérêt dans l'exploration (au sein d'une exploration les PI sont numérotés en séquence en commençant à 0, ce numéro n'a pas de rapport avec l'ordre dans lequel les PI sont explorés par le robot)
  - `PI_X` l'abscisse du point d'intérêt dans la zone d'exploration (entier positif en millimètres)
  - `PI_Y` l'ordonnée du point d'intérêt dans la zone d'exploration (entier positif en millimètres)
  - `PI_ARR` date d'arrivée du robot au point d'intérêt (`NULL` si ce point n'a pas encore été visité)
  - `PI_DEP` date à laquelle le robot a quitté le point d'intérêt (`NULL` si ce point n'a pas encore été exploré ou si la visite est en cours)
- la table `INSTR` des instruments embarqués, avec les colonnes
  - `IN_NUM` le numéro (entier) de l'instrument (clef primaire)
  - `IN_FCT` la durée pendant lequel l'instrument a déjà été utilisé depuis l'arrivée sur la planète (nombre décimal : fraction de jour martien)
  - `IN_DER` la date de la dernière utilisation de l'instrument
  - `IN_NOM` nom de l'instrument
- la table `TYANA` des types d'analyses à effectuer, avec les colonnes
  - `TY_NUM` le numéro de référence (entier) du type d'analyse (clef primaire)
  - `TY_DES` le nom du type d'analyse
- la table `INTYP` des instruments utilisés pour un type d'analyse, de clef primaire `(TY_NUM, IN_NUM)`, avec les colonnes
  - `TY_NUM` le numéro de référence (entier) du type d'analyse
  - `IN_NUM` le numéro (entier) de l'instrument
  - `IT_DUR` la durée standard d'utilisation de l'instrument dans ce type d'analyse (nombre décimal : fraction de jour martien)
- la table `ANALY` indiquant pour chaque point d'intérêt les types d'analyses à effectuer ou effectuées, de clef primaire `(EX_NUM, PI_NUM, TY_NUM)` et avec les colonnes
  - `EX_NUM` numéro de l'exploration à laquelle appartient le point d'intérêt
  - `PI_NUM` numéro du point d'intérêt dans l'exploration
  - `TY_NUM` type de l'analyse
  - `AN_DAT` date de l'analyse (`NULL` si l'analyse n'a pas été effectuée)
  - `AN_DTR` date de transmission sur Terre des résultats de l'analyse (`NULL` si l'analyse n'a pas été transmise)
  - `AN_RES` résultat de l'analyse (donnée opaque dont la signification dépend du type d'analyse)

Toutes les dates sont stockées sous forme d'un nombre décimal correspondant au nombre de jours martiens depuis l'arrivée du robot sur la planète.

Question 1. Écrire une requête SQL qui donne le numéro de l'exploration en cours, s'il y en a une.

Question 2. Écrire une requête SQL qui donne, pour une exploration dont on connait le numéro, la liste des points d'intérêts de cette exploration avec leurs coordonnées.

Question 3. Écrire une requête SQL qui donne la surface, en mètres carrés, de chaque zone déjà explorée par le robot. La zone d'exploration est définie comme le plus petit rectangle qui englobe l'ensemble des points d'intérêts de l'exploration et dont les bords sont parallèles aux axes de référence (axes des abscisse et des ordonnées).

Question 4. Quelle est la surface maximale d'une zone d'exploration que peut stocker cette base de données ?

Question 5. Écrire une requête SQL qui donne, pour l'exploration en cours, le nombre de fois où chaque instrument doit être utilisé et sa durée d'utilisation théorique (en jours martiens) pour la totalité de l'exploration.

6 Mines 2017

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.

Question 1. Ecrire 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. Ecrire 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 ?

```
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
```

Annexe : rappels

- Voici un rappel de ce à quoi ressemble une requête SQL complète :

```
SELECT attributs FROM table1 JOIN table2 JOIN table3 ...
ON conditions de jointure
WHERE condition (sélection) GROUP BY attributs (agrégation)
HAVING condition (sélection post-agrégation)
ORDER BY quantité1, quantité2,... (ordonnancement, ASC/DESC après l'attribut)
LIMIT n (limiter à au plus n résultats)
OFFSET p (on ignore les p premiers résultats)
```

- Opérateurs booléens : `AND`, `OR`, `NOT`, et comparaisons `<=`, `<`, etc... pour les conditions.
- `SELECT DISTINCT...` pour éliminer les doublons de la liste de résultats.
- Fonctions d'agrégation : `MAX`, `MIN`, `COUNT`, `SUM`, `AVG`.
- Lorsqu'il y a une fonction d'agrégation sans clause GROUP BY, le résultat n'a qu'une ligne, il n'y a pas de sens à faire afficher une colonne.
- Lorsqu'il y a une clause GROUP BY, la clause SELECT doit contenir la (ou les)colonne(s) qui a (ont) servi au regroupement ; les seuls autres résultats qui peuvent être affichés sont des fonctions d'agrégation.
- Le renommage d'attribut ou de table se fait à l'aide de AS ou en juxtaposant simplement le nouveau nom à l'ancien (`SELECT attribut a FROM ...` , `SELECT ... FROM table1 t1 ...`) ;
- Les opérations ensemblistes `UNION`, `INTERSECT`, `EXCEPT` fonctionnenent avec des tables à même schéma (rarement utilisé pour nous) ;
- Requêtes imbriquées : `SELECT ... FROM (SELECT ...)`. Rappel : le résultat d'une requête est une table !
- Cas particuliers :
  — une table à une ligne et une colonne s'identifie à son contenu.
  (ex : `WHERE truc = (SELECT max(machin) FROM bidule)`) ;
  — on peut tester l'appartenance à une table à une colonne avec `IN`.
  (ex : `WHERE truc IN (SELECT machin FROM bidule)`).
  — éventuellement utile : `WITH (requete) AS r SELECT` ... permet d'utiliser facilement le résultat d'une sous-requête dans une requête plus importante.
- Lorsque l'on utilise plusieurs tables dont certaines colonnes ont le même nom, dans le cas d'une jointure, il faut préfixer les noms des colonnes par le nom de la table partout où on utilise une colonne. On doit remplacer le nom de la table par son alias lorsqu'il existe ; il sera donc souvent utile de donner un alias court.
- Jointure particulière : `t1 NATURAL JOIN t2` lorsque `t1` et `t2` ont des colonnes identiques. Le résultat est une table ou chaque colonne en question n'apparaît qu'une fois.