CAPES-INFO 2019–2020 ECRIT 2 BLANC
PROBLÈME DE BASES DE DONNÉES
Durée approximative 1 heure

Jeudi 12 mars 2020

On donne en annexe B.1 le thème *traitement de données en tables* extrait du programme de NSI 1re. On donne également en annexe B.2 le thème *bases de données* extrait du programme de NSI Terminale.

1 Bases de données

Exercise 1 : Analyse de schéma de base de données

On considère le schéma de base de données fourni en annexe A. Il s'agit de celui d'une application de gestion de questionnaires (*quiz*). L'attribut **api_key** est un secret qui permet l'authentification des utilisateurs. Son type `uuid` est un entier de 128 bits représenté par une chaîne de 32 chiffres en base 16, comme `123e4567-e89b-12d3-a456-426655440000`.

1. Proposer une représentation graphique du schéma. Préciser le formalisme employé. Illustrer les principales notions du modèle relationnel sur votre formalisme (dans une autre couleur).
2. On souhaite introduire la notion de contrainte d'intégrité sur ce schéma. Lister tous les types de contraintes employés (faire références aux numéro de lignes) en en proposant une description informelle en langue naturelle.
3. Proposer un jeu d'essai pour ce schéma en le représentant sous formes tabulaire (on ne demande pas les requêtes). Faire en sorte que ce schéma comporte un maximum de cas particuliers.
4. Proposer *cinq* questions fermées (binaires) de compréhension du schéma à proposer aux élèves comme par exemple « *peut-on avoir une question qui ne comporte que des réponses fausses ?* ». Justifier la variété des question en exploitant les cas particuliers identifiés à la question précédente.
5. Proposer *trois* requêtes de difficulté progressive à écrire en SQL à partir de leur description en langue naturelle.
6. Inversement, proposer *trois* requêtes de difficulté progressive à décrire en langue naturelle à partir de leur définition en SQL.
7. Pour réemployer ce schéma en 1re, on souhaite exporter le contenu de la base au format CSV. Donner une requête SQL qui va produire un `unique` tableau contenant *tout le contenu de la base*.
8. Proposer *trois* anomalies sur le tableau précédent.

Exercise 2 : Production d'élève

On considère les questions et réponses apportées à un questionnaire d'une évaluation donnés en figure 1. Cette épreuve porte sur le schéma suivant, où les clefs étrangères sont préfixées du symbole « # » et les attributs des clefs primaires sons soulignés :

$$\begin{aligned}
\text{Certificats}(&\underline{idC}, titreC, duree, garanties, organisme, \#idC_ancien, dateR)\\
\text{Certifications}(&\underline{idF}, \underline{idC}, dateC)\\
\text{Forets}(&\underline{idF}, nomF, gestionnaire, contour, \#idE)\\
\text{Localisations}(&\underline{idF}, \underline{idP}, superficie)\\
\text{Pays}(&\underline{idP}, nomP, contour)\\
\text{Especes}(&\underline{idE}, nomE, description)\\
\text{Provenance}(&\underline{idP}, \underline{idE})
\end{aligned}$$

1. Corriger les trois questions en expliquant les éventuelles erreurs.
2. Justifier des erreurs intégrées dans les distracteurs de la question 3.
3. Donner une définition SQL (`CREATE TABLE`) de la relation Forets.

Question 3 ♣ On souhaite lister les forêts sans contour ou qui ont un certificat. Complétez la requête SQL suivante.

```
SELECT idF …
```

- ☐ `FROM FORÊTS WHERE contour = NULL OR CERTIFICATIONS.idC != NULL`
- ☐ `FROM FORÊTS WHERE contour = NULL`
- ☐ `FROM FORÊTS WHERE contour IS NULL OR CERTIFICATIONS.idC IS NOT NULL`
- ☑ `FROM FORÊTS NATURAL JOIN CERTIFICATIONS WHERE contour = NULL`
- ☑ `INTERSECT SELECT idF FROM FORÊTS NATURAL JOIN Certifications`
- ☐ `FROM FORÊTS WHERE contour IS NULL`
- ☑ `FROM FORÊTS NATURAL JOIN CERTIFICATIONS WHERE contour IS NULL`
- ☑ `UNION SELECT idF FROM FORÊTS NATURAL JOIN Certifications`
- ☐ *Aucune de ces réponses n'est correcte.*

Question 5 ♣ Une collègue a écrit la requête SQL suivante. Sans même la tester, vous pouvez dire...

```
SELECT idE, nomP
FROM PROVENANCE NATURAL JOIN PAYS
WHERE idE > 100
ORDER BY nomP ;
```

- ☐ Il y a une erreur dans la clause `ORDER BY`
- ☐ Il y a une erreur dans la clause `FROM`
- ☐ Il y a une erreur dans la clause `WHERE`
- ☐ Il y a une erreur dans la clause `SELECT`
- ☑ *Bravo, ta requête est juste !*

Question 6 ♣ Un autre collègue a écrit une requête pour retrouver les forêts certifiées en 2019. Sans même la tester, vous pouvez dire...

```
SELECT idF, idC, dateC
FROM FORÊTS INNER JOIN CERTIFICATIONS
WHERE dateC LIKE '*2019*'
ORDER BY dateC DOWN ;
```

- ☑ Il y a une erreur dans la clause `WHERE`
- ☑ Il y a une erreur dans la clause `ORDER BY`
- ☐ Il y a une erreur dans la clause `SELECT`
- ☐ Il y a une erreur dans la clause `FROM`
- ☐ *Bravo, ta requête est juste !*

FIGURE 1 – Extrait de production d'élève

A Annexe : schéma de base de données

```sql
CREATE SCHEMA IF NOT EXISTS lifap5;

DROP TABLE IF EXISTS lifap5.answer;
DROP TABLE IF EXISTS lifap5.proposition;
DROP TABLE IF EXISTS lifap5.question;
DROP TABLE IF EXISTS lifap5.quiz;
DROP TABLE IF EXISTS lifap5.quiz_user;

CREATE TABLE IF NOT EXISTS lifap5.quiz_user (
  user_id TEXT PRIMARY KEY CHECK (char_length(user_id) > 4),
  firstname TEXT NOT NULL,
  lastname TEXT NOT NULL,
  api_key UUID UNIQUE NULL
);

CREATE TABLE IF NOT EXISTS lifap5.quiz (
  quiz_id INTEGER PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
  created_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT CURRENT_TIMESTAMP,
  title TEXT UNIQUE NOT NULL,
  description TEXT NOT NULL,
  owner_id TEXT REFERENCES lifap5.quiz_user(user_id) NOT NULL,
  open BOOLEAN NOT NULL DEFAULT FALSE
);

CREATE TABLE IF NOT EXISTS lifap5.question (
  PRIMARY KEY (quiz_id, question_id),
  quiz_id INTEGER REFERENCES lifap5.quiz,
  question_id INTEGER GENERATED ALWAYS AS IDENTITY,
  content TEXT NOT NULL,
  optional BOOLEAN NOT NULL DEFAULT FALSE,
  weight INTEGER NOT NULL DEFAULT 1
);

CREATE TABLE IF NOT EXISTS lifap5.proposition (
  PRIMARY KEY (quiz_id, question_id, proposition_id),
  FOREIGN KEY (quiz_id, question_id) REFERENCES lifap5.question,
  quiz_id INTEGER,
  question_id INTEGER,
  proposition_id INTEGER GENERATED ALWAYS AS IDENTITY,
  content TEXT NOT NULL,
  correct BOOLEAN NOT NULL
);

CREATE TABLE IF NOT EXISTS lifap5.answer (
  PRIMARY KEY (quiz_id, question_id, user_id),
  FOREIGN KEY (quiz_id, question_id, proposition_id)
    REFERENCES lifap5.proposition,
  quiz_id INTEGER,
  question_id INTEGER,
  user_id TEXT REFERENCES lifap5.quiz_user,
  proposition_id INTEGER NOT NULL,
  answered_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

B Annexe : programmes

B.1 Extrait du programme de NSI 1re

On donne l'extrait ci-dessous, le tableau des compétences est donné en figure 2.

Traitement de données en tables

Les données organisées en table correspondent à une liste de p-uplets nommés qui partagent les mêmes descripteurs. La mobilisation de ce type de structure de données permet de préparer les élèves à aborder la notion de base de données qui ne sera présentée qu'en classe terminale. Il s'agit d'utiliser un tableau doublement indexé ou un tableau de p-uplets, dans un langage de programmation ordinaire et non dans un système de gestion de bases de données.

Contenus	Capacités attendues	Commentaires
Indexation de tables	Importer une table depuis un fichier texte tabulé ou un fichier CSV.	Est utilisé un tableau doublement indexé ou un tableau de p-uplets qui partagent les mêmes descripteurs.
Recherche dans une table	Rechercher les lignes d'une table vérifiant des critères exprimés en logique propositionnelle.	La recherche de doublons, les tests de cohérence d'une table sont présentés.
Tri d'une table	Trier une table suivant une colonne.	Une fonction de tri intégrée au système ou à une bibliothèque peut être utilisée.
Fusion de tables	Construire une nouvelle table en combinant les données de deux tables.	La notion de domaine de valeurs est mise en évidence.

FIGURE 2 – Extrait du programme de NSI 1re sur le traitement de données en tables

B.2 Extrait du programme de NSI Terminale

On donne l'extrait ci-dessous, le tableau des compétences est donné en figure 3.

Bases de données

Le développement des traitements informatiques nécessite la manipulation de données de plus en plus nombreuses. Leur organisation et leur stockage constituent un enjeu essentiel de performance.

Le recours aux bases de données relationnelles est aujourd'hui une solution très répandue. Ces bases de données permettent d'organiser, de stocker, de mettre à jour et d'interroger des données structurées volumineuses utilisées simultanément par différents programmes ou différents utilisateurs. Cela est impossible avec les représentations tabulaires étudiées en classe de première.

Des systèmes de gestion de bases de données (SGBD) de très grande taille (de l'ordre du pétaoctet) sont au centre de nombreux dispositifs de collecte, de stockage et de production d'informations.

L'accès aux données d'une base de données relationnelle s'effectue grâce à des requêtes d'interrogation et de mise à jour qui peuvent par exemple être rédigées dans le langage SQL (Structured Query Language). Les traitements peuvent conjuguer le recours au langage SQL et à un langage de programmation.

Il convient de sensibiliser les élèves à un usage critique et responsable des données.

Contenus	Capacités attendues	Commentaires
Modèle relationnel : relation, attribut, domaine, clef primaire, clef étrangère, schéma relationnel.	Identifier les concepts définissant le modèle relationnel.	Ces concepts permettent d'exprimer les contraintes d'intégrité (domaine, relation et référence).
Base de données relationnelle.	Savoir distinguer la structure d'une base de données de son contenu. Repérer des anomalies dans le schéma d'une base de données.	La structure est un ensemble de schémas relationnels qui respecte les contraintes du modèle relationnel. Les anomalies peuvent être des redondances de données ou des anomalies d'insertion, de suppression, de mise à jour. On privilégie la manipulation de données nombreuses et réalistes.
Système de gestion de bases de données relationnelles.	Identifier les services rendus par un système de gestion de bases de données relationnelles : persistance des données, gestion des accès concurrents, efficacité de traitement des requêtes, sécurisation des accès.	Il s'agit de comprendre le rôle et les enjeux des différents services sans en détailler le fonctionnement.
Langage SQL : requêtes d'interrogation et de mise à jour d'une base de données.	Identifier les composants d'une requête. Construire des requêtes d'interrogation à l'aide des clauses du langage SQL : SELECT, FROM, WHERE, JOIN. Construire des requêtes d'insertion et de mise à jour à l'aide de : UPDATE, INSERT, DELETE.	On peut utiliser DISTINCT, ORDER BY ou les fonctions d'agrégation sans utiliser les clauses GROUP BY et HAVING.

FIGURE 3 – Extrait du programme de NSI terminale sur les bases de données