Interrogation d'informatique
2ème année - Novembre 2023

Durée totale : *1h30*
Documents autorisés : *Aucun.*

INSA

— Le barème est sur 20 points.
— Le sujet est sur 7 pages - il y a 3 exercices.

Exercice 1 : Requêtes SQL (11 pts)

Le département FIMI organise un cross, auquel peuvent participer les élèves de $1^{ère}$ et $2^{ème}$ année. Les élèves s'inscrivent en équipe. Une équipe peut être composée d'élèves d'années différentes ($1^{ère}$ et $2^{ème}$ année). Un parcours est composé d'étapes distinctes. Une même étape peut faire partie de plusieurs parcours. Les étapes composant un parcours ont chacune un numéro d'ordre dans le parcours. Chaque équipe choisit de participer sur un seul parcours. La course a une durée fixe. Une équipe fait courir successivement un de ses membres sur les étapes du parcours. L'équipe qui gagne est celle qui aura parcouru la distance la plus grande dans le temps imparti de la course. Ainsi durant le temps imparti, une équipe peut faire plusieurs fois le parcours choisi. La dernière fois qu'une équipe entame le parcours choisi, peut conduire à ne valider qu'un sous-ensemble de ses étapes. Chaque fois qu'un participant finit de courir entièrement une étape, la date de début et le temps passé à faire l'étape sont mémorisés dans la base de données.

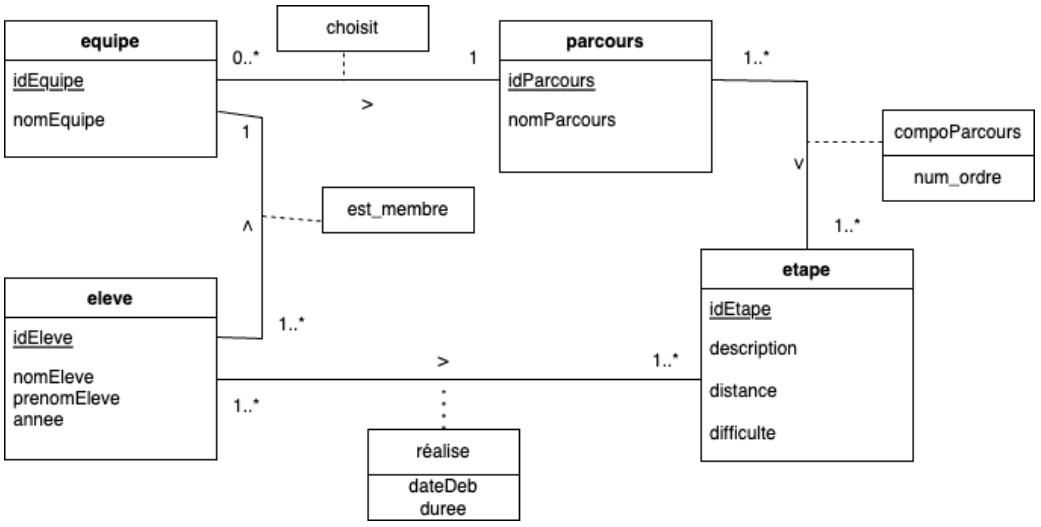

FIGURE 1 – Schéma E/A de la base de données

Le département FIMI a créé une base de données dont le schéma relationnel est décrit ci-dessous. Sa version graphique est représentée en figure 3 et le modèle Entité/Association correspondant en figure 1 Les attributs formant la clé d'une relation sont soulignés. Pour simplifier, les types des attributs ne sont pas indiqués dans les relations. Les attributs `distance`, `difficulte`, `numOrdre`, `annee` et `duree` sont des nombres. L'attribut `dateDeb` et de type date. Tous les autres attributs sont des chaines de caractères.

`parcours(idParcours, nomParcours)`
Chaque parcours est identifié de manière unique par son identifiant. On mémorise aussi son nom.

etape(idEtape, description, distance, difficulte)
Chaque étape est identifiée de manière unique par un identifiant. On mémorise sa description, la distance à parcourir et son niveau de difficulté allant de 1 à 4.
compoParcours(idParcours,idEtape, numOrdre)
Un tuple est créé dans cette table pour chaque étape faisant partie d'un parcours. Les attributs `idParcours` et `idEtape` sont des clés étrangères respectivement sur les relations `parcours` et `etape`. Un parcours étant une succession d'étapes, on mémorise aussi le numéro d'ordre de l'étape dans ce parcours.
eleve(idEleve, nomEleve, prenomEleve, annee, idEquipe)
Chaque élève possède un identifiant unique. Tous les élèves du FIMI sont représentés dans cette table. Le champ `idEquipe` est une clé étrangère sur la table `equipe`. Il est renseigné uniquement pour les élèves qui participent au cross au sein d'une équipe (sinon il est indéfini).
equipe(idEquipe, nomEquipe, idParcours)
Chaque tuple dans cette table, identifié de manière unique (idEquipe), représente une des équipes participantes. On mémorise le nom de l'équipe ainsi que le parcours auquel elle est inscrite. L'attribut `idParcours` est une clé étrangère référençant la relation `parcours`.
tempsRealises(idEleve, idEtape, dateDeb, duree) Une ligne de cette relation mémorise qu'un élève a parcouru une étape en `duree` minutes, en commençant à la date `dateDeb`. Seuls les cas où l'étape a été complètement courrue sont enregistrés. Les attributs `idEleve` et `idEtape` sont des clés étrangères référençant respectivement les relations `eleve` et `etape`.

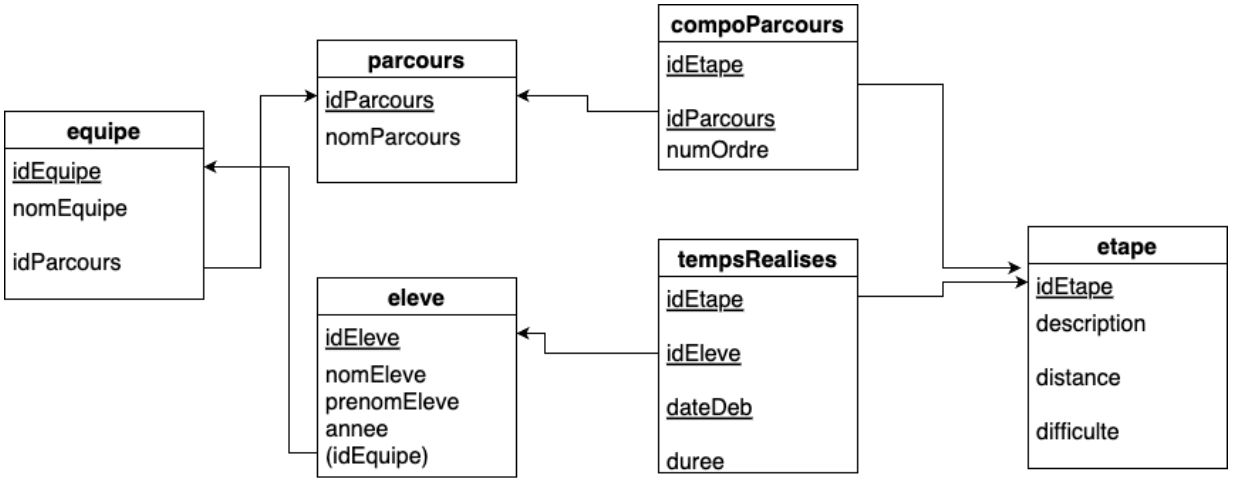

FIGURE 2 – Représentation graphique du schéma relationnel de la base de données

Question : Écrivez les requêtes SQL correspondant aux recherches suivantes. Prenez soin d'éliminer les doublons lorsque le résultat ne doit pas en comporter. Pour chaque requête, utilisez le minimum de tables possible.

1. l'INSA souhaite calculer le nombre de petites bouteilles d'eau à acheter pour les participants. Elle prévoit un achat de 10 bouteilles par participant. Exprimez la requête permettant de calculer le nombre total de bouteilles à commander.

```
SELECT COUNT(*)*10 as qte_bouteilles
```

```
FROM eleves
WHERE idEquipe is not NULL
```

2. Donnez tous les identifiants de parcours ayant au moins 3 étapes de difficulté supérieur à 2.

```
SELECT IdParcours
FROM compoParcours, etapes
WHERE etapes.ipEtape = compoParcours.idEtape
AND difficulte >= 2
GROUP BY idParcours
HAVING count(*) >=3
```

3. Donnez le classement des équipes par ordre décroissant des distances parcourues. Affichez l'équipe, la distance totale parcourue, ainsi que la durée totale. En cas d'égalité des distances, les résultats sont classés par ordre croissant suivant le temps total réalisé.

```
SELECT idEquipe, sum(distance) as distance-totale, sum(duree) as sDuree
FROM etapes, eleves, tempsRealises
WHERE etapes.idEtape = tempsRealises.idEtape
and tempsRealises.idEleve = eleves.idEleve
GROUP BY idEquipe
ORDER BY distance-totale desc, sDuree asc
```

4. Donnez le nom, le prénom et l'année d'étude des participants de l'équipe "sans cardio united" n'ayant jamais couru l'étape d'identifiant ETP5 .

```
SELECT Nom, prenom , annee
FROM eleve, equipe
WHERE eleve.idEquipe = equipe.idEquipe
and nomEquipe = "sans cardio united"
and idEleve not in (
select idEleve from tempsRealises
where idEtape = 'ETP5')
```

5. Donnez le nom des élèves, l'année d'étude et le numéro d'équipe, des élèves qui ont effectué l'étape d'identifiant 'ETP5' avec une durée inférieure à la moyenne des durées réalisées sur cette étape.

```
SELECT nom, annee , idEquipe
from eleves e, tempsRealises t
where e.idEleve = t.idEleve
and idEtape = 'ETP5'
and duree < (
select avg(duree) from tempsRealises where idEtape = 'ETP5')
```

6. Donnez par ordre alphabétique la composition de l'équipe à laquelle participe l'étudiant Antoine Dupont. On suppose qu'il n'y a pas deux personnes s'appelant Antoine Dupont dans la base de données.

```
select e2.Nom, e2.Prenom , e2.annee
FROM eleve e1, eleve e2
where e1.nom = 'Dupont'
and e1.prenom='Antoine'
and e1.idEquipe = e2.IdEquipe
and e2.idEleve != e1.idEleve
order by e2.Nom, e2.Prenom
```

7. L'équipe "Les lutins agiles" s'est inscrite à un parcours. Affichez les étapes du parcours qu'elle doit réaliser, avec pour chaque étape, son identifiant, sa description, sa distance et son niveau. De plus, les étapes doivent être affichées selon leur numéro d'ordre (en premier celle qui a le numéro 1 . . .etc).

```
select e.idEtape, e.description, e.distance, e.difficulte
from equipe q, compoParcours c, etapes e
where q.idEquipe = 'Les lutins agiles'
and q.idParcours = c.idParcours
and c.idEtape = e.idEtape
order by c.numOrdre
```

Exercice 2 : Modélisation (5 points)

2.1 Questions sur le schéma E/A initial

1. En vous aidant du schéma E/A de la figure 1, indiquez de quelle association la relation `tempsRealises` est issue. Précisez pourquoi la clé est formée de 3 attributs.

`tempsRealises` est issue de l'association `réalise`, de type "many to many". Selon la règle 3 de transformation du cours, elle donne lieu à une relation, idEleve et idEtape faisant partie de sa clé. On ajoute dataDeb à la clé car une étape peut être courrue plusieurs fois par la même personne.

2. Une équipe peut-elle être inscrite à plusieurs parcours ? Justifiez.

Non, du fait de la cardinalité 1 près de l'entité `parcours` pour l'association `choisit`.

2.2 Étapes avec points d'intérêt

On dispose des informations supplémentaires suivantes : Un *point d'intérêt* (POI) a un identifiant, un nom, une latitude et une longitude. Une étape peut comporter des points d'intérêts (POI). Tous les points d'intérêt répertoriés appartiennent à au moins une étape. Chaque point d'intérêt propose un défi et un seul. Un défi a un identifiant et une description, par exemple, *faire 20 flexions*. Un même défi peut être proposé par plusieurs points d'intérêt. Lorsqu'un ou une élève relève un défi, on mémorise le moment où il-elle le fait et le temps que cela lui prend.

1. Proposez une extension du schéma E/A de la figure 1 qui tienne compte de ces nouvelles informations.

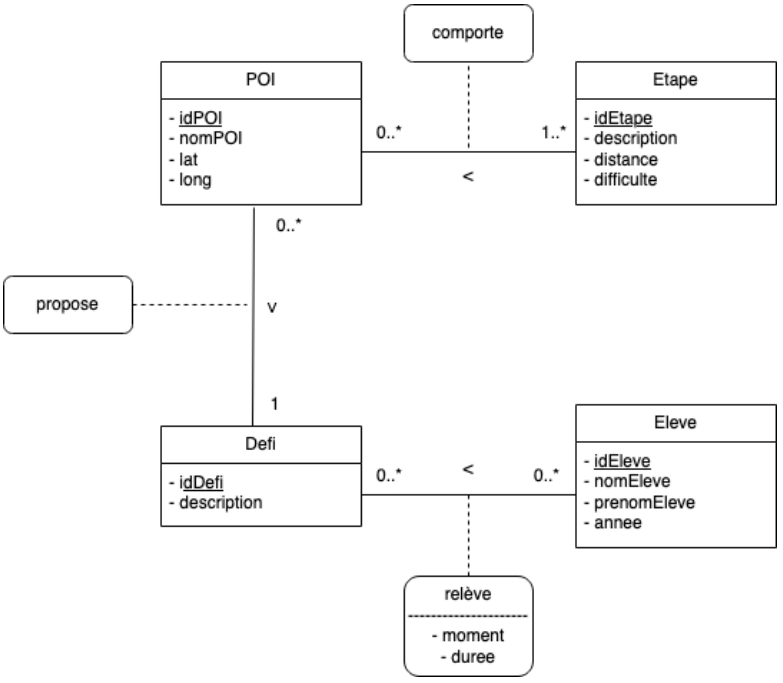

FIGURE 3 – Extension schéma E/A

2. Indiquez les modifications à apporter au modèle relationnel suite à la transformation de votre schéma E/A.

poi(idPoi, nomPoi, lat, long, idDefi)
idDefi est une clé étrangère référençant Defi
defi(idDefi, descriptionDefi)
releveDefis(idDefi, idEleve, momentDef, dureeDef)
idDefi est une clé étrangère référençant Defi ; idEleve est une clé étrangère référençant Eleve
comporte(idPoi, idEtape) nomPoi est une clé étrangère référençant poi ; idEtape est une clé étrangère référençant Etape.
Les relations définies auparavant sont inchangées.

Exercice 3 : Dictionnaires (4pts)

Afin de répondre aux questions de ses lecteurs sur les articles publiés, un journal national mémorise, pour chaque article écrit, le nom et le numéro de téléphone d'une personne que les lecteurs peuvent contacter pour poser leurs questions.

Un programme est réalisé pour automatiser des traitements. Il utilise un dictionnaire qui associe à chaque identifiant d'article, une liste comportant le nom de la personne contact et son numéro de téléphone.

Étant donnée la variable `articles`, définie dans le code ci-dessous, répondez aux différentes questions.

```
articles = {
'GraphèmeCerveauHumain231023' :  ['Jean Dupont' , '0666666666'],
'PerfUltraEndu241023' : ['Alain Durand' , '0666666667'],
'MortSubiteNourrisson241023' : ['Jean Dupont' , '0666666666'],
'ChantAmourMoucheVinaigre' : ['Alain Durand' , '0666666667'],
'ChatGPTExpMultiMedia211023' : [ 'Lina Romero', '0677777777'],
'GrippeAviaireFouBassan201023': ['Alain Durand' , '0666666667']
}
```

(3.1) Écrivez un code python permettant d'afficher le nom et le numéro de téléphone de la personne contact de l'article d'identifiant 'PerfUltraEndu241023'.

L'affichage se fera sous la forme :
`Nom : Alain Durand, tél : 0666666667`

```
print(f"Nom : {articles['PerfUltraEndu241023'][0]}, tél : {articles['PerfUltraEndu241023'][1]}")
```

(3.2) Écrivez le code python d'une fonction `est_contact_de(articles, nom_p)`, qui retourne la liste des identifiants des articles qui ont comme contact la personne de nom `nom_p`

Exemple : l'appel de la fonction avec le dictionnaire mémorisé dans la variable *articles* ci-dessus et comme nom 'Alain Durand' retournera la liste :

```
['PerfUltraEndu241023', 'ChantAmourMoucheVinaigre', 'GrippeAviaireFouBassan201023']
```

```
def est_contact_de(articles, nom_p) :
    res = []
    for cle, pers in articles.items() :
        if pers[0] == nom_p:
            res.append(cle)
    return res
```

(3.3) Écrivez le code python d'une fonction `genere_dico_noms(articles)`, qui retourne un dictionnaire associant à chaque personne la liste des identifiants d'articles pour lesquels elle est contact.

```python
def genere_dico_noms(articles):
    res = dict()
    for cle, pers in articles.items() :
        if pers[0] not in res.keys():
            res[pers[0]] = [cle]
        else:
            res[pers[0]].append(cle)
    return res
```

```python
#solution avec un get
def genere_dico_noms(articles):
    res = dict()
    for cle, pers in articles.items() :
        #res[pers[0]] = res.get(pers[0],[]).append(cle) pose pb si None
        res[value[0]] = res.get(value[0],[]) + [key]
    return res
```

```python
def genere_dico_noms3(articles):
    res = dict()
    for pers in articles.values():
        if pers[0] not in res.keys():
            res[pers[0]] = est_contact_de(articles, pers[0])
    return res
```

Remarque : un dictionnaire `d` peut être créé par le code : `d = dict()`.