+1/1/60+

Contrôle terminal de BDBIO - session 1

UCBL - Département informatique (2024/2025)

Pour assurer l'anonymat, n'écrivez pas votre nom ou numéro étudiant sur la copie.
Une feuille papier A4 autorisée. Durée : 1h30.
Remplissez les cases sur la dernière feuille, de préférence au stylo noir. Les questions avec un symbole ♣ ont 0, 1 ou plusieurs bonnes réponses. Les autres ont une unique bonne réponse.

Catégorie (id_cat, *nom*)
Description (#*nom_série*, #*id_cat*)
Dessinatrice (nom, prénom, *expérience*, *style*)
Éditeur (id_éditeur, *nom*, *nb_employés*)
Festival (id_festival, *nom*, *ville*, #*id_éditeur*)
Production (#*id_éditeur*, #*nom_série*, début, fin, *description*)
Scénariste (nom, prénom, *expérience*, *thème*)
Série (nom_série)
Tome (#*nom_série*, numéro, *titre*)

Source 1 (schéma relationnel)

```
[
  {
    "_id": <string>,
    "éditeur": <string>,
    "employés": [
      {
        "id": <integer>,
        "nom": <string>
      },
      ...
    ]
  },
  ...
]
```

Source 2 (schéma collection JSON)

```
<collection>
  <bd id="b1" numero="2">
    <titre>Fini de rire</titre>
    <serie nom="Calvin et Hobbes" />
    <auteur role="dessin">Bill Watterson</auteur>
  </bd>
  <bd id="b3" numero="1">
    <titre>Idées noires</titre>
  </bd>
</collection>
```

Source 3 (XML)

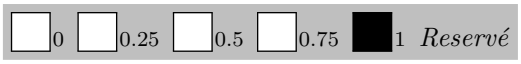
```
bd:id_entier rdfs:domain bd:Categorie .
bd:id_entier rdfs:range xsd:integer .
bd:subcat rdfs:domain bd:Categorie .
bd:subcat rdfs:range bd:Categorie .
bd:noms rdfs:domain bd:Categorie .
bd:noms rdfs:range rdfs:Resource .
rdfs:label rdfs:domain rdfs:Resource .
rdfs:label rdfs:range xsd:string .
bd:acronyme rdfs:domain rdfs:Resource .
bd:acronyme rdfs:range xsd:string .
```

Source 4 (description RDF)

1 Modélisation (6 points)

Dans les questions suivantes, on souhaite compléter et améliorer le schéma relationnel. Ajoutez ou modifiez des relations pour répondre aux demandes.

Question 1 Une ou plusieurs scénaristes et une ou plusieurs dessinatrices contribuent à l'élaboration d'un tome. Notez qu'une personne peut parfois être à la fois scénariste et dessinatrice.



Dessiner(#nom_série, #numéro, #nom, #prénom)
Scénariser(#nom_série, #numéro, #nom, #prénom)

Question 2 Les informations sur les scénaristes et les dessinatrices sont assez redondantes. Proposez une solution pour limiter cette redondance.

☐0 ☐0.25 ☐0.5 ☐0.75 ■1 *Reservé*

Auteure(<u>nom, prénom</u>, expérience)
Dessinatrice(<u>#nom, #prénom</u>, style)
Scénariste(<u>#nom, #prénom</u>, thème)

Question 3 Pour chaque édition annuelle d'un festival, on veut stocker les scénaristes et les dessinatrices invitées ainsi que le nombre de visiteurs.

☐0 ☐0.25 ☐0.5 ☐0.75 ■1 *Reservé*

Edition(<u>#id_festival, année</u>, nb_visiteurs)
Inviter(<u>#id_festival, #année, #nom, #prénom</u>, stand)

Question 4 Enfin, on souhaite pouvoir définir des types de relation (e.g., parodie, *reboot*) entre 2 séries.

☐0 ☐0.25 ☐0.5 ☐0.75 ■1 *Reservé*

TypeRelation(<u>idt</u>, libellé)
Relier(<u>#nom_série1, #nom_série2, #idt</u>)

Question 5 Écrire une DTD qui permet de valider les données XML.

☐0 ☐0.25 ☐0.5 ☐0.75 ☐1 ☐1.25 ☐1.5 ☐1.75 ■2 *Reservé*

```
<!DOCTYPE collection [
<!ELEMENT collection (bd*)>
<!ELEMENT bd (titre, serie?, auteur*)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT serie EMPTY>
<!ELEMENT auteur (#PCDATA)>
<!ATTLIST bd id ID #REQUIRED>
<!ATTLIST bd numero CDATA #IMPLIED>
<!ATTLIST serie nom CDATA #REQUIRED>
<!ATTLIST auteur role CDATA  #IMPLIED>
]>
```

2 Interrogation (3 points)

Question 6 Le nom des séries qui sont produites par un éditeur qui a organisé plus de 3 festivals (SQL).

☐ 0 ☐ 0.25 ☐ 0.5 ☐ 0.75 ☐ 1 ☐ 1.25 ■ 1.5 *Reservé*

```sql
SELECT nom_série FROM Production
WHERE id_éditeur IN ( SELECT id_éditeur
  FROM  Festival
  GROUP BY id_éditeur
  HAVING COUNT(*) > 3
)
```

Question 7 Le nombre de tomes par série (XQuery).

☐ 0 ☐ 0.25 ☐ 0.5 ☐ 0.75 ☐ 1 ☐ 1.25 ■ 1.5 *Reservé*

```xquery
for $bd in //bd
let $s := $bd/serie/@nom
let $nb := count($bd/@id)
group by $s
return <span>{$s} : {$nb}</span>
```

3 Intégration de données (11 points)

Question 8 Une table TEMP(id, name, city, id_ed) possède des centaines d'instances de la forme suivante : `<(1, 'festival A (gratuit)', 'paris', 3), (7, 'festival B (plein air)', 'lyon', 9), ...>`. Écrivez **une requête SQL** qui permet de migrer les données de TEMP vers la table FESTIVAL. La table FESTIVAL contient actuellement 25 instances numérotées de 1 à 24. Vous nettoierez le nom. Enfin, votre requête ne doit pas générer d'erreur.

☐ 0 ☐ 0.5 ☐ 1 ☐ 1.5 ■ 2 *Reservé*

```sql
INSERT INTO Festival
SELECT id, SUBSTR(name, 1, STRPOS(name, '(')-1), city, id_ed
FROM Temp JOIN Editeur ON id_éditeur = id_ed WHERE id > 24;
```

Pour répondre aux questions suivantes, vous utiliserez les langages de requête appropriés en fonction des sources. Le reste du programme informatique sera codé en pseudo-langage (syntaxe libre, mais suffisamment explicite pour que le programme soit implémentable). Pour le niveau de détail, utilisez des appels de fonctions pour simplifier le code (e.g., si vous devez trier un tableau, écrivez `tab_sorted = sort(tab)` accompagné d'un commentaire mais n'écrivez pas un algorithme complet de tri de tableau !). Utilisez des commentaires, par exemple pour expliquer comment vous résolvez les conflits.

Question 9 Écrivez tout d'abord un programme pour migrer les données JSON dans la BD relationnelle. Considérez que vous avez un objet Python pour la source de données JSON. La clé primaire de la table EDITEUR est auto-incrémentée. Vous vérifierez que vous n'insérez pas un éditeur déjà existant dans la table.

☐ 0 ☐ 0.5 ☐ 1 ☐ 1.5 ■ 2 *Reservé*

```python
bdrel = connexion(source1)
json = connexion(source2.json)

for doc in json:
  nom = doc['éditeur']
  nb_emp = len(doc['employés'])
  # vérification déjà existant
  req = f"select * from éditeur where nom = '{nom}'"
  res = bdrel.executer(req)
  if not res:  # non existant, donc insertion
    req = f"insert into éditeur values (NULL, '{nom}', {nb_emp})"
    bdrel.executer(req)
```

Question 10 Écrivez un programme pour migrer des données RDF sur les catégories dans la BD relationnelle. Ces données RDF sont conformes à la description de la source 4.

☐ 0 ☐ 0.5 ☐ 1 ☐ 1.5 ☐ 2 ☐ 2.5 ■ 3 *Reservé*

```
bdrel = connexion(source1)
bdrdf = connexion(source4.rdf)

res = bdrdf.executer('''select * where {
    _:c rdf:type bd:Categorie .
    _:c bd:id_entier ?id .
    _:c bd:noms _:n .
    _:n rdfs:label ?n .
  }''')

for id, n in res:
  # vérification déjà existant
  req = f"select * from catégorie where nom = '{nom}' or id_cat = {id}"
  res2 = bdrel.executer(req)
  if not res2:  # non existant, donc insertion
    req = f"insert into catégorie values ({id}, '{nom}')"
    bdrel.executer(req)
```

Question 11 Écrivez un programme pour migrer les données XML dans la BD relationnelle. Vous vérifierez que les instances n'existent pas avant d'insérer.

☐ 0 ☐ 0.5 ☐ 1 ☐ 1.5 ☐ 2 ☐ 2.5 ☐ 3 ☐ 3.5 ■ 4 *Reservé*

```
bdrel = connexion(source1)
bdxml = connexion(source3)

res = bdxml.executer("//bd")
for bd in res:
  num = bd.get("numéro")
  titre = bd.executer("./titre/text()")
  noms = bd.executer("data(./serie/@nom)")
  auteur = bd.executer("./auteur")
  role = auteur.get("role")
  noma = auteur.text.split()

  res2 = bdrel.exécuter(f"select * from série where nom_série = '{noms}'")
  if not res2:  # non existant, donc insertion
    bdrel.executer(f"insert into série values ('{noms}')")

  res2 = bdrel.exécuter(f"select * from tome where nom_série = '{noms}' and numéro={num}")
  if not res2:  # non existant, donc insertion
    bdrel.executer(f"insert into tome values ('{noms}', {num}, '{titre}')")

  if role == 'dessin':
    res2 = bdrel.exécuter(f"select * from dessinatrice where nom = '{noma[1]}' and
    ↪  prénom='{noma[0]}'")
    if not res2:  # non existant, donc insertion
      bdrel.executer(f"insert into dessinatrice values ('{noma[1]}', {noma[0]}, NULL, NULL)")
  # idem pour scénariste
```