Examen de Bases de Données (CS443)

IR&C, 2021/2022

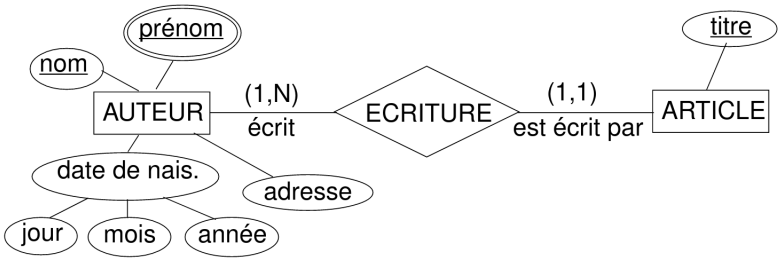
Documents interdits, une annexe est fournie

Instructions :

1. On justifiera les réponses
2. Une annexe vous est fournie.

1 Conception

On notera les schémas E/A en suivant la même convention que sur sur le schéma ci dessous, dans lesquels les auteurs écrivent des articles (les losanges sont les associations, les carrés dénotent des entités possédant des attributs (ronds)).



Question #1

Dans le dessin ci dessus, une des deux cardinalités exprime la contrainte que tous les auteurs doivent écrire au moins un article. Quelle autre contrainte est exprimée par ces cardinalités ?

Une colonie de fourmis consiste en plusieurs fourmilières. Chaque fourmilière loge un groupe (ensemble) de fourmis et chaque fourmi a une couleur et un prénom. En plus la colonie maintient une liste des tâches (comme par exemple, la défense, la récolte de la nourriture etc.) et les tâches sont attribuées aux fourmis. La liste de tâches n'est pas fixe mais peut évoluer avec le temps.

Question #2

Proposer un schéma Entités/relations pour modéliser cette situation. On commentera les choix effectués et les hypothèses supplémentaires faites lors de la modélisation.

2 RétroConception

Nous sommes architectes d'un site Web hambook.org pour les personnes enthousiastes des hamsters. Ce site stocke les informations privées sur ses utilisateurs ainsi que leurs hamsters et permet de publier un flux des messages publics contenant des nouvelles des hamsters. Les informations sont stockées dans une base de données relationnelle construite de la manière suivante :

```
CREATE TABLE USER(
ID INT PRIMARY KEY,
PSEUDO TEXT NOT NULL UNIQUE,
NAME TEXT NOT NULL,
EMAIL TEXT NOT NULL UNIQUE,
DOB DATE
);

CREATE TABLE SPECIES(
ID INT PRIMARY KEY,
NAME TEXT NOT NULL,
DESCRIPTION TEXT
);

CREATE TABLE HAMSTER(
ID INT PRIMARY KEY,
NAME TEXT NOT NULL,
DOB TEXT,
SPECIES_ID INT REFERENCES SPECIES(ID),
OWNER_ID INT NOT NULL REFERENCES USER(ID)
);

CREATE TABLE POST(
HAMSTER_ID INT REFERENCES HAMSTER(ID),
DATE DATE,
CONTENT TEXT NOT NULL,
PRIMARY KEY(HAMSTER_ID,DATE)
);

CREATE TABLE COMMENT(
USER_ID INT REFERENCES USER(ID),
POST_HAMSTER_ID INT,
POST_DATE DATE,
COMMENT TEXT NOT NULL,
FOREIGN KEY (POST_HAMSTER_ID,POST_DATE)
    REFERENCES POST(HAMSTER_ID,DATE)
);
```

Question #3

En vous reportant à l'annexe, expliquez en français les lignes de construction de la table `COMMENT`.

Question #4

Proposer un schéma Entités/Relations pour cette base de données.

3 Requêtes en algèbre relationnelle

On considère la BD suivante de stocks de vélos dans un magasin de réparation :

VÉLOS (idV, *étatV*, *annéeService*)
RÉPARER (#idV, #idT, dateR, *duréeR*)

idV	étatV	annéeService
1	bon	2018
2	bon	2019
3	crevé	2019
4	cassé	2020

Table VÉLOS

idV	idT	dateR	duréeR
1	123	2020/06/01	60
1	456	2020/07/12	30
2	123	2020/09/15	75
2	789	2020/09/15	20
4	789	2020/10/21	30

Table RÉPARER

On rappelle que les requêtes en algèbre relationnelle utilisent les opérateurs de projection (π), sélection (σ) et renommage (ρ), ainsi que (des variantes de) jointure ($\bowtie$), et enfin les opérations ensemblistes classiques.

Question #5

Décrire en français et donner le résultat attendu des requêtes suivantes :

1. $\Pi_{\text{étatV}}(\sigma_{\text{annéeService} < 2020}(\text{VÉLOS}))$
2. $\rho_{\text{vélosOK/idV}}(\sigma_{\text{étatV = 'bon'}}(\Pi_{\text{idV}}(\text{VÉLOS})))$
3. $\Pi_{\text{idV, annéeService, dateR}}(\sigma_{\text{étatV = 'cassé'}}\ (\text{RÉPARER} \bowtie_{\text{idV}} \text{VÉLOS}))$

Question #6

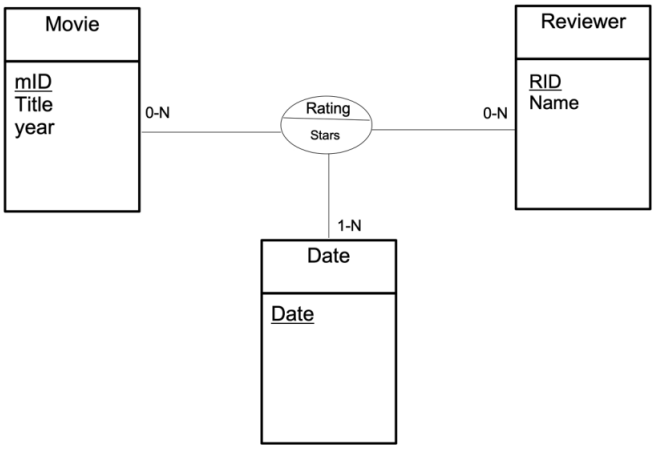
Construire des requêtes pour répondre aux questions suivantes.

1. Identifiant des vélos en bon état ou mis en service après 2019
2. Date de réparation des vélos mis en service en 2019
3. Identifiant et année des vélos qui ne sont pas en bon état ou qui ont été réparés plus de 60 minutes
4. Identifiant des vélos qui n'ont jamais été réparés
5. Identifiant des vélos mis en service la même année que le vélo 2
6. Paires de vélos réparés par la même technicienne

4 Quelques requêtes SQL

Pour concevoir un site d'appréciation de films, on construit un système de gestion dans lequel les films sont identifiés par un numéro, ont un titre, une date de réalisation et un réalisateur. Les rapporteurs ont un identifiant unique et obligatoirement un nom. Chaque évaluation est définie par le rapporteur, le film et la date d'évaluation ; elle donne lieu à une note (nombre d'étoiles). Il ne peut pas y avoir d'évaluation sans note.

Pour simplifier, le nom du réalisateur sera une simple information pour chaque film et ne représentera pas une entité de l'application.
Le schéma Entité/Association (avec les conventions du cours) ci-dessous est proposé pour représenter ces données.



Après traduction du diagramme EA et simplification (on a supprimé la relation 'Date' qui n'apportait rien), on obtient le schéma relationnel suivant :

— Movie(mID, title, year , director)

— Reviewer (rID, name)

— Rating(rID, mID, ratingDate, stars)

Question #7

Expliquer comment la troisième relation Rating est obtenue à partir du schéma E/A.

Question #8

Écrire sous la forme d'une formule booléenne une formule pour la relation Movie permettant de vérifier que la même année, deux films ne peuvent pas avoir le même titre [1].

Question #9

Exprimer en SQL les requêtes permettant de retrouver les informations suivantes :

1. Le nom du relecteur 205.
2. Les films réalisés par Steven Spielberg.
3. Les titres de films dont le réalisateur n'est pas renseigné.
4. Le nom des personnes qui ont noté le film 'Gone with the Wind'.

Les fonctions agrégatives calculent une valeur unique à partir d'un ensemble de tuples, par exemple : $max(x), min(x), count(x), avg(x) \ldots$. Elles le font sur l'ensemble des tuples retournés par les clauses "FROM ... WHERE", ou bien sur chaque groupe formé par l'utilisation de "GROUP BY". On précise ici qu'une condition d'un "WHERE" ne peut porter sur une fonction agrégée, on utilise alors le mot-clé "HAVING".

Question #10

Un peu plus difficile :

1. Pour chaque film, trouver la meilleure note reçue. Retourner le titre du film et le nombre d'étoiles, trier par rapport au titre de film (ordre alphabétique)
2. En utilisant une ou plusieurs fonctions agrégatives, trouver le nom de tous les examinateurs qui ont fait au moins 3 évaluations.

1. On pourrait ajouter cette contrainte avec une commande ALTER TABLE en SQL.