# Bases de données: Examen

Pierre Senellart (pierre.senellart@ens.fr)

20 mai 2026

Les seuls documents autorisés sont 5 feuilles A4 recto-verso (10 pages), avec le contenu de votre choix, lisible à l'œil nu. Cet examen dure deux heures et est constitué d'un unique problème de 10 questions, noté sur 20 points. La qualité de la rédaction sera prise en compte ; pensez à expliciter brièvement vos notations lorsque celles-ci ne sont pas standard.

On souhaite modéliser le système d'information (volontairement minimal) d'un *serveur de préprints scientifiques* (du type d'arXiv ou HAL). Le système gère un ensemble d'*articles*, d'*auteurs*, et de *citations* entre articles.

Chaque article a un identifiant unique et un titre, et est écrit par un ou plusieurs auteurs. Un auteur a un identifiant unique, un nom, une adresse email, et est rattaché à une unique *institution* courante. Une institution a un identifiant, un nom et un pays.

Un article peut *citer* d'autres articles ; il ne se cite pas lui-même.

1. (2 points) Proposer un *schéma entité–association* pour ce système d'information, en distinguant entités, attributs, relations, cardinalités, et contraintes pertinentes.
2. (2 points) En déduire un *schéma relationnel* et l'exprimer à l'aide d'ordres SQL `CREATE TABLE`, en indiquant les contraintes suivantes lorsque pertinentes : `PRIMARY KEY`, `UNIQUE`, `NOT NULL`, `CHECK`, `FOREIGN KEY`.
3. (1 point) Identifier une contrainte du système d'information qui n'est *pas capturée* par les contraintes SQL standard de la question précédente. L'exprimer sous forme de formule de la logique du premier ordre, et préciser s'il s'agit d'une TGD ou d'une EGD.
4. (3 points) Écrire en *algèbre relationnelle*, en *calcul relationnel*, puis en *SQL* une requête retournant les noms et adresses email des auteurs s'étant *cités eux-mêmes*, c'est-à-dire des auteurs $p$ pour lesquels il existe deux articles $A_1$ et $A_2$, tous deux ayant $p$ parmi leurs auteurs, tels que $A_1$ cite $A_2$.
5. (2 points) Proposer deux *plans d'exécution* distincts pour la requête précédente et discuter de leurs performances respectives en fonction des tailles et des distributions statistiques des relations impliquées. Préciser quel(s) index pourrai(en)t accélérer chacun de ces plans.
6. (2 points) Pour des besoins d'analyse, on dénormalise une partie du schéma dans une unique table
$$R(\mathit{auteur}, \mathit{nom}, \mathit{email}, \mathit{institution}, \mathit{nomInst}, \mathit{paysInst}).$$
On considère l'ensemble de dépendances fonctionnelles
$$F = \{\ \mathit{auteur} \to \mathit{nom}, \mathit{email}, \mathit{institution}\ ;\ \mathit{institution} \to \mathit{nomInst}, \mathit{paysInst}\ \}.$$
   a) Utiliser la *chase* pour décider si la dépendance fonctionnelle $\mathit{auteur} \to \mathit{paysInst}$ est impliquée par $F$.

b) $R$ est-elle en BCNF ? Justifier, et, si ce n'est pas le cas, proposer une décomposition sans perte qui le soit, en précisant les dépendances fonctionnelles préservées.

7. (2 points) On considère la requête booléenne conjonctive

$$Q_\triangle() \leftarrow \mathit{Cite}(x,y),\ \mathit{Cite}(y,z),\ \mathit{Cite}(z,x)$$

qui teste l'existence d'un *cycle de citations* de longueur 3 (on n'impose plus que les sommets soient distincts).

a) Exprimer la borne AGM sur le nombre maximal de triplets satisfaisant $Q_\triangle$, lorsque $|\mathit{Cite}| = N$, comme la solution d'un programme linéaire sur les couvertures fractionnaires d'arêtes du graphe de la requête, puis calculer explicitement cette borne.

b) Comparer le coût asymptotique, dans le pire cas, d'un plan d'exécution à *jointures binaires* (par exemple par jointures de hachage successives) à celui d'un *algorithme de jointure pire-cas optimal* (par exemple Leapfrog Triejoin). Préciser dans quel régime la différence est la plus marquée.

8. (2 points) Exprimer la relation $\mathit{Influence}(A, B)$ des paires d'articles telles que $A$ est atteignable depuis $B$ par une suite (de longueur $\geq 1$) de citations, dans *deux* des trois formalismes suivants au choix :

a) Datalog ;

b) SQL (à l'aide de `WITH RECURSIVE`) ;

c) logique du premier ordre à point fixe inflationniste.

9. (2 points) On souhaite expliquer chaque tuple $(A, B) \in \mathit{Influence}$ produit par la requête précédente en conservant la trace des citations élémentaires qui le justifient. Pour cela, on annote chaque tuple de *Cite* par un identifiant distinct dans un ensemble $\mathcal{X}$, et on calcule la provenance de *Influence* dans un semi-anneau bien choisi.

a) Quel semi-anneau choisir, parmi ceux vus en cours, pour retrouver l'ensemble des sous-ensembles de citations *minimaux* dont chacun suffit à justifier le tuple ? Justifier rapidement qu'il s'agit bien d'un semi-anneau commutatif.

b) À quoi correspondent concrètement les opérations $\oplus$ et $\otimes$ ainsi que leurs éléments neutres dans ce semi-anneau ?

c) Sur l'exemple où $\mathit{Cite} = \{(a,b) : t_1,\ (b,c) : t_2,\ (a,c) : t_3\}$, donner la provenance du tuple $(a, c)$ de *Influence*.

10. (2 points) Le serveur expose une API publique permettant d'accéder aux titres des articles et à leurs auteurs. Cette API ne doit *en aucun cas* laisser fuiter les adresses email des auteurs.

a) Définir, à l'aide d'ordres SQL `CREATE VIEW`, les vues à exposer à l'API. Expliquer brièvement quels droits d'accès doivent être configurés sur ces vues et sur les tables de base pour qu'une application connectée à l'API ne puisse jamais lire d'autres données que celles autorisées.

b) Un·e développeur·euse écrit en Python le code suivant pour rechercher un article par identifiant (le module `psycopg` est utilisé pour l'interface avec PostgreSQL et la connexion est établie avec le rôle `api_public`) :

```
sql = "SELECT titre FROM Article WHERE id = '" + request.args["id"] + "'"
cur.execute(sql)
```

Donner une charge utile (*payload*) que peut soumettre une personne malveillante pour exploiter ce code, en précisant l'effet de l'attaque. Expliquer en quoi le mécanisme de ségrégation d'accès de la question (a) limite la portée de l'attaque, et en particulier protège les adresses email. Réécrire enfin le code de manière à empêcher l'attaque.