Corrigé de l'Examen de Bases de données (Licence 2) – 1re session 2023–2024

Partie 1 : Requêtes – Algèbre & SQL

Exercice A.1

Énoncé. Donnez le nom des équipements situés au repère GPS `345_78S`.

Algèbre relationnelle

$$\pi_{\text{Nom}}\big(\sigma_{\text{GPS}='345\_78S'}(\text{Equip})\big)$$

SQL

```
SELECT Nom
FROM Equip
WHERE GPS = '345_78S';
```

Explication. On commence par sélectionner dans la relation `Equip` toutes les lignes où l'attribut `GPS` vaut `'345_78S'`. Ensuite, on projette sur l'attribut `Nom` pour ne conserver que le nom de l'équipement.

Exercice A.2

Énoncé. Quelles sont les épreuves (identifiants) de ski ?

Algèbre relationnelle

$$\pi_{\text{IdE}}\big(\sigma_{\text{Nom}='ski'}(\mathit{Sport}) \ \bowtie \ \text{Sport.IdS} = \text{Epreuve.IdS} \ \ \mathit{Epreuve}\big)$$

SQL

```
SELECT E.IdE
FROM Epreuve E
JOIN Sport S ON E.IdS = S.IdS
WHERE S.Nom = 'ski';
```

Explication. On effectue une jointure entre `Epreuve` et `Sport` sur l'identifiant du sport `IdS`, puis on sélectionne les lignes où le nom du sport est `'ski'`. Enfin, on projette uniquement l'identifiant d'épreuve `IdE`.

Exercice A.3

Énoncé. Quelles sont les épreuves (identifiants) de sports individuels qui ont eu lieu au repère GPS `345_78S` ?

Algèbre relationnelle

$$\pi_{E.IdE}\big(\sigma_{\text{TP}='individuelle' \,\wedge\, \text{GPS}='345\_78S'}\big(Sport \bowtie_{Sport.IdS=Epreuve.IdS} Epreuve \bowtie_{Epreuve.IdEq=Equip.IdEq} Equip\big)\big)$$

SQL

```
SELECT E.IdE
FROM Epreuve E
JOIN Sport S ON E.IdS = S.IdS
JOIN Equip  Q ON E.IdEq = Q.IdEq
WHERE S.TP = 'individuelle'
  AND Q.GPS = '345_78S';
```

Explication. On réalise une triple jointure : d'abord entre `Epreuve` et `Sport` pour filtrer par type de pratique `'individuelle'`, puis avec `Equip` pour filtrer les équipements situés au `GPS='345_78S'`. On projette enfin l'identifiant `IdE` des épreuves correspondantes.

Exercice A.4

Énoncé. Quels sont les sports (noms) pour lesquels aucune épreuve n'est prévue ?

Algèbre relationnelle

$$\pi_{\text{Nom}}(Sport) \;-\; \pi_{S.Nom}\big(Sport \bowtie_{Sport.IdS=Epreuve.IdS} Epreuve\big)$$

SQL

```
SELECT Nom
FROM Sport
WHERE IdS NOT IN (
  SELECT IdS FROM Epreuve
);
```

Explication. On récupère d'abord tous les noms de sports, puis on soustrait ceux qui apparaissent dans la relation `Epreuve`. En SQL, on utilise une sous-requête pour exclure les sports dont l'identifiant `IdS` figure dans `Epreuve`.

Exercice A.5

Énoncé. Quels sports (nom et TP) ont utilisé pour au moins une de leurs épreuves le même équipement que l'épreuve de ski individuel du 14-02-2014 ?

Algèbre relationnelle

$$\pi_{S2.Nom,S2.TP}\Big((S2 \bowtie_{S2.IdS=E2.IdS} Epreuve\ E2) \cap \pi_{S.IdS}\big(\sigma_{Nom='ski' \wedge TP='individuelle' \wedge Date='2014-02-14'}(Sport \bowtie Epreuve)\big)\Big)$$

SQL

```
SELECT DISTINCT S2.Nom, S2.TP
FROM Sport S2
JOIN Epreuve E2 ON S2.IdS = E2.IdS
WHERE E2.IdEq = (
  SELECT E1.IdEq
  FROM Epreuve E1
  JOIN Sport S1 ON E1.IdS = S1.IdS
  WHERE S1.Nom = 'ski'
    AND S1.TP = 'individuelle'
    AND E1.Date = '2014-02-14'
);
```

Explication.

1. On identifie d'abord l'équipement `IdEq` utilisé par l'épreuve de `ski` individuelle du 14 février 2014 à l'aide d'une sous-requête.
2. Puis on sélectionne tous les sports `S2` qui ont au moins une épreuve utilisant ce même `IdEq`, en joignant `Sport` et `Epreuve`. On élimine les doublons grâce à `DISTINCT`.

Exercice A.6

Énoncé. Quels sont les équipements (noms) qui n'ont jamais été utilisés ?

Algèbre relationnelle

$$\pi_{\text{Nom}}(Equip)\ - \pi_{Q.Nom}\big(Equip \bowtie_{Equip.IdEq=Epreuve.IdEq} Epreuve\big)$$

SQL

```
SELECT Nom
FROM Equip
WHERE IdEq NOT IN (
  SELECT IdEq FROM Epreuve
);
```

Explication. On liste tous les équipements puis on soustrait ceux dont l'identifiant `IdEq` apparaît dans la relation `Epreuve`. Cela donne les équipements jamais utilisés.

Exercice A.7

Énoncé. Quels sont les équipements (identifiants) ayant été utilisés au moins deux fois ?

Algèbre relationnelle

$$\pi_{IdEq}\Big(\sigma_{\text{Count}\geq 2}\big(\gamma_{IdEq;\ \text{COUNT}(IdEq)\to\text{Count}}(Epreuve)\big)\Big)$$

SQL

```
SELECT IdEq
FROM Epreuve
GROUP BY IdEq
HAVING COUNT(*) >= 2;
```

Explication. On groupe les lignes de `Epreuve` par `IdEq`, on calcule le nombre d'occurrences par groupe, puis on ne conserve que ceux dont le compte est supérieur ou égal à 2.

Exercice B.8

Énoncé. Donnez les lieux (GPS) où tous les sports ont été programmés.

Algèbre relationnelle

$$\pi_{GPS}(Equip) \div \pi_{IdS}(Sport)$$

Explication. L'opérateur division permet de sélectionner les `GPS` pour lesquels l'ensemble des sports (tous les `IdS` de `Sport`) ont au moins une épreuve. On divise la relation `Equip` $\bowtie$ *Epreuveprojetesur*`(GPS, Id`

Partie 1 : Requêtes – SQL avec agrégats et sous-requêtes

Exercice C.9

Énoncé. Combien y a-t-il d'équipements situés au repère GPS 345_78S ?

SQL

```
SELECT COUNT(*) AS nb
FROM Equip
WHERE GPS = '345_78S';
```

Explication. On applique une sélection sur l'attribut GPS, puis on utilise COUNT(*) pour compter le nombre de lignes restantes.

Exercice C.10

Énoncé. Quels sont les équipements (identifiants) ayant été utilisés au moins 10 fois ?

SQL

```
SELECT IdEq
FROM Epreuve
GROUP BY IdEq
HAVING COUNT(*) >= 10;
```

Explication. Même raisonnement que l'exercice A.7, mais avec un seuil de 10.

Exercice C.11

Énoncé. Donnez pour chaque sport (identifiant et nom) la durée moyenne des épreuves pour ce sport.

SQL

```
SELECT S.IdS, S.Nom, AVG(E.Durée) AS duree_moyenne
FROM Sport S
JOIN Epreuve E ON S.IdS = E.IdS
GROUP BY S.IdS, S.Nom;
```

Explication. On joint Sport et Epreuve pour associer chaque épreuve à son sport, puis on groupe par sport et on calcule la moyenne des durées.

Exercice C.12

Énoncé. Donnez les lieux (GPS) où tous les sports ont été programmés.

SQL

```
SELECT Q.GPS
FROM Equip Q
WHERE NOT EXISTS (
  SELECT 1
  FROM Sport S
  WHERE NOT EXISTS (
    SELECT 1
    FROM Epreuve E
    WHERE E.IdEq = Q.IdEq
      AND E.IdS = S.IdS
  )
);
```

Explication. On utilise une double sous-requête NOT EXISTS pour garantir qu'aucun sport S ne manque une épreuve au lieu Q.GPS.

Partie 2 : SQL – Agrégats sur l'instance R

Soit la relation R(A,B,C,D) donnée en Figure 1.

A	B	C	D
a1	b1	c1	d1
a1	b2	c1	d1
a2	b2	c1	d1
a1	b2	c3	d3
a1	b1	c1	d2
a1	b2	c4	d4

1. SELECT A, B, COUNT(C) FROM R GROUP BY A, B;

   Résultat :

A	B	COUNT(C)
a1	b1	2
a1	b2	3
a2	b2	1

2. SELECT A, B, COUNT(DISTINCT C) FROM R GROUP BY A, B;

   Résultat :

A	B	COUNT(DISTINCT C)
a1	b1	1
a1	b2	3
a2	b2	1

3. SELECT A, COUNT(C) FROM R GROUP BY A, B;
   Résultat :
   C'est une requête non valide SQL standard, car on ne peut pas projeter A sans inclure B dans le GROUP BY ou dans un agrégat.
4. SELECT A, COUNT(DISTINCT C) FROM R GROUP BY A, B;
   Résultat :
   Même observation que la requête précédente : invalidité due au GROUP BY.
5. SELECT A, B, COUNT(C) FROM R GROUP BY A;
   Résultat :
   Non valide pour la même raison : B n'est pas dans le GROUP BY ni agrégé.
6. SELECT A, B, COUNT(DISTINCT C) FROM R GROUP BY A;
   Résultat :
   Non valide pour la même raison.

Partie 3 : Anomalies de mise à jour

Soit la relation Employé(Emp_Id, Nom, Salle, Extension) de la Figure 2.

Question 3.1 : Types d'anomalies

On identifie les trois anomalies :

Anomalie de suppression

Exemple : Suppression de l'employé Durant (ligne E001, Durant, B3, 4500) conduit à perdre l'information sur la salle B3 et son extension 4500.

Anomalie d'insertion

Exemple : Pour insérer une nouvelle salle C1 avec extension 4600 sans avoir d'employé assigné, on ne peut pas, car la relation impose un Emp_Id.

Anomalie de mise à jour

Exemple : Changer l'extension de la salle A7 de 3142 à 3150 nécessite de modifier deux tuples (E001, Dupont et E001, Petit), risquant incohérence si on oublie l'un.

Question 3.2 : Élimination des anomalies

Pour éliminer ces anomalies, on décompose la relation Employé en deux relations :

$$\texttt{Employe}(Emp\_Id, Nom),$$
$$\texttt{Localisation}(Salle, Extension).$$

Chaque tuple Employe contient maintenant un employé indépendant de la salle, et chaque tuple Localisation mappe une salle à son extension sans redondance.

Question 3.3 : Faut-il toujours éliminer ?

Non, pas systématiquement. Lorsque l'on a besoin de requêtes très fréquentes alliant employé, salle et extension, la décomposition peut nuire aux performances (trop de jointures). Il faut équilibrer coût de maintenance et besoin de cohérence, selon les priorités de l'application.