Solutions de l'Examen Bases de Données (Session 1, 2021–2022)

A. Requêtes : Algèbre relationnelle et SQL sans sous-requêtes ni agrégats

1. Marques de bus pouvant transporter plus de 50 voyageurs

Algèbre relationnelle

$$\pi_{\text{Marq}}\Big(\sigma_{\text{Cap}>50}(\text{Bus})\Big)$$

Explication : On sélectionne d'abord les tupples de `Bus` dont `Cap>50`, puis on projette l'attribut `Marq` pour obtenir les marques.

SQL

```
SELECT DISTINCT Marq
FROM Bus
WHERE Cap > 50;
```

Explication : La clause `WHERE` filtre les bus de capacité supérieure à 50, et `DISTINCT` élimine les doublons.

2. Identifiants des trajets effectués en bus de marque `RW`

Algèbre relationnelle

$$\pi_{\text{IdT}}\big(\sigma_{\text{Marq}='RW'}(\text{Bus}) \bowtie \text{Traj}\big)$$

Explication : On réalise la jointure naturelle entre `Bus` et `Traj` pour associer chaque trajet à son bus, on sélectionne les bus de marque `RW`, puis on projette `IdT`.

SQL

```
SELECT T.IdT
FROM Traj T JOIN Bus B ON T.IdB = B.IdB
WHERE B.Marq = 'RW';
```

Explication : `JOIN` lie les deux tables ; la clause `WHERE` filtre sur la marque, et on affiche l'identifiant du trajet.

3. Trajets (IdT) avec bus stationné à Lille et chauffeurs nommés 'Driver'

Algèbre relationnelle

$$\pi_{\text{IdT}}\big(\sigma_{\text{Ville}='Lille'}(\text{Bus}) \bowtie_{B.IdB=Tr.IdB} \sigma_{\text{NomC}='Driver'}(\text{Cond}) \bowtie_{Tr.IdC=C.IdC} \text{Traj}\big)$$

Explication : On restreint d'abord `Bus` à Lille et `Cond` au nom 'Driver', on effectue les jointures successives avec `Traj`, puis on projette `IdT`.

SQL

```
SELECT T.IdT
FROM Traj T
JOIN Bus B   ON T.IdB = B.IdB
JOIN Cond C  ON T.IdC = C.IdC
WHERE B.Ville = 'Lille'
  AND C.NomC = 'Driver';
```

4. Identifiants des bus jamais utilisés sur un trajet

Algèbre relationnelle

$$\pi_{\text{IdB}}(\text{Bus}) \ - \ \pi_{\text{IdB}}(\text{Traj})$$

Explication : On prend l'ensemble des bus puis on retire ceux qui apparaissent dans `Traj`.

SQL

```
SELECT IdB
FROM Bus
WHERE IdB NOT IN (SELECT DISTINCT IdB FROM Traj);
```

Explication : La sous-requête liste les bus utilisés, et `NOT IN` sélectionne les autres.

5. Conducteurs (IdC, NomC) ayant utilisé une marque parmi celles des bus partis de Nice le 14-04-2022

Algèbre relationnelle

$$\pi_{\text{IdC,NomC}}\Big(\big(\pi_{\text{IdC,IdB}}(\sigma_{\text{VD}='Nice' \wedge \text{Date}='2022-04-14'}(\text{Traj})) \bowtie \text{Traj} \bowtie \text{Bus}\big)\Big)$$

Explication : On identifie d'abord les trajets partant de Nice ce jour-là, on récupère les marques des bus correspondants, puis on sélectionne les conducteurs ayant conduit ces bus.

SQL

```
SELECT DISTINCT C.IdC, C.NomC
FROM Traj T
JOIN Bus B  ON T.IdB = B.IdB
JOIN Cond C ON T.IdC = C.IdC
WHERE T.VD = 'Nice'
  AND T.Date = '2022-04-14'
  AND B.Marq IN (
      SELECT DISTINCT B2.Marq
      FROM Traj T2
      JOIN Bus B2 ON T2.IdB = B2.IdB
      WHERE T2.VD = 'Nice'
        AND T2.Date = '2022-04-14'
  );
```

6. Villes rejoignables depuis Nice avec un changement à Lyon le même jour

Algèbre relationnelle

$$\pi_{\text{Date},HD1,HA1,VA2,HA2}$$
$$\Big(\rho_{T1}(\sigma_{\text{VD}='Nice'}(\text{Traj})) \bowtie_{T1.Date=T2.Date \wedge T1.VA='Lyon' \wedge T1.HA<T2.HD} \rho_{T2}(\sigma_{\text{VD}='Lyon'}(\text{Traj}))\Big)$$

Explication : On prend deux instances de `Traj` : `T1` pour le tronçon Nice→Lyon, `T2` pour Lyon→destination, on impose même date et correspondance horaire, puis on projette les attributs demandés.

SQL

```
SELECT T1.Date,
       T1.HD    AS HeureDepartNice,
       T1.HA    AS HeureArriveeLyon,
       T2.VA    AS VilleFinale,
       T2.HA    AS HeureArriveeFinale
FROM Traj T1
JOIN Traj T2 ON T1.Date = T2.Date
             AND T1.VA = 'Lyon'
             AND T1.HA < T2.HD
WHERE T1.VD = 'Nice'
  AND T2.VD = 'Lyon';
```

7. Conducteurs ayant fait au moins deux trajets entre Lille et Nice

Algèbre relationnelle

$$\pi_{\text{IdC}}\Big(\sigma_{\text{VD}='Lille'\wedge\text{VA}='Nice'}(\text{Traj}) \ \bowtie_{IdC} \sigma_{\text{VD}='Nice'\wedge\text{VA}='Lille'}(\text{Traj})\Big)$$

Explication : On réalise la jointure sur `IdC` entre trajets Lille→Nice et Nice→Lille, ce qui garantit au moins deux trajets.

SQL

```
SELECT C.IdC
FROM (
  SELECT IdC
  FROM Traj
  WHERE VD = 'Lille' AND VA = 'Nice'
) AS A
JOIN (
  SELECT IdC
  FROM Traj
  WHERE VD = 'Nice' AND VA = 'Lille'
) AS B ON A.IdC = B.IdC;
```

B. Algèbre relationnelle uniquement

8. Villes où tous les conducteurs ont démarré au moins un trajet

Algèbre relationnelle

$$\pi_{\text{VD}}(\text{Traj}) \ - \pi_{\text{VD}}\big(\pi_{\text{VD}}(\text{Traj}) \times (\pi_{\text{IdC}}(\text{Cond}) \ - \pi_{\text{IdC}}(\text{Traj}))\big)$$

Explication : On énumère d'abord toutes les villes de départ, puis on retire celles pour lesquelles il existe au moins un conducteur n'ayant pas de trajet débutant dans cette ville.

C. SQL avec agrégats et sous-requêtes

9. Nombre de trajets au départ de Nice le 14-04-2022

```
SELECT COUNT(*) AS NbTrajets
FROM Traj
WHERE VD = 'Nice'
  AND Date = '2022-04-14';
```

Explication : `COUNT(*)` compte les lignes satisfaisant la condition.

10. Nombre de trajets par marque (supérieur à 20)

```
SELECT B.Marq, COUNT(*) AS NbTrajets
FROM Traj T
JOIN Bus B ON T.IdB = B.IdB
GROUP BY B.Marq
HAVING COUNT(*) > 20;
```

Explication : `GROUP BY` regroupe par marque, et `HAVING` filtre les groupes de taille > 20.

Normalisation

Schéma :
Agence(IdCl, NomCl, IdApp, AdrApp, DDLoc, DFLoc, Montant, IdProp, NomProp)
Dépendances fonctionnelles :

1. IdCl → NomCl
2. IdProp → NomProp
3. NomProp → IdProp
4. IdCl,IdApp → DDLoc,DFLoc
5. IdApp → AdrApp, Montant, IdProp, NomProp

Q1. Instance illustrant anomalies

IdCl	NomCl	IdApp	AdrApp	DDLoc	DFLoc	Montant	IdProp	NomProp
1	Alice	A1	1 rue X	2022-01-01	2022-02-01	500	P1	Dupont
1	Alice	A2	2 rue Y	2022-03-01	2022-04-01	300	P2	Martin
2	Bob	A1	1 rue X	2022-05-01	2022-06-01	500	P1	Dupont
3	Charlie	A3	3 rue Z	2022-02-15	2022-03-15	400	P3	Durand
3	Charlie	A2	2 rue Y	2022-07-01	2022-08-01	300	P2	Martin

Type 1 : *Insertion*
Problème : Impossible d'ajouter un nouvel appartement A4 sans louer à un client existant (puisqu'AdrApp dépend de IdApp dans la même table).

Type 2 : *Suppression*
Problème : Si on supprime la dernière location de A3, on perd l'adresse de A3 et les informations sur le propriétaire P3.

Type 3 : *Mise à jour*
Problème : Si le propriétaire P2 change de nom (ex. 'Martin' → 'Petit'), il faut mettre à jour plusieurs lignes (pour A2, A2), d'où risque d'incohérence.

Q2. Clés du schéma

Un déterminant minimal qui couvre tous les attributs : {IdCl, IdApp}.
Justification :

- De IdCl,IdApp découlent DDLoc,DFLoc (DF 4).
- De IdCl découle NomCl (DF 1).
- De IdApp découlent AdrApp, Montant, IdProp, NomProp (DF 5), et de IdProp NomProp (DF 2 et 3).

Aucune sous-partie propre n'est clé, donc c'est la clé candidate.

Q3. 3eme forme normale

Non. Par exemple, IdCl → NomCl viole la 3FN car IdCl n'est pas super-clé et NomCl n'est pas partie d'une clé.

Q4. Forme normale de Boyce–Codd

Non plus, car pour IdProp → NomProp, IdProp n'est pas super-clé du schéma complet.

Q5. Décomposition en FNBC

On applique l'algorithme de décomposition :
1. FD IdCl → NomCl donne R1(IdCl,NomCl).
2. FD IdApp → AdrApp,Montant,IdProp,NomProp donne R2(IdApp,AdrApp,Montant,IdProp,NomProp).
3. FD IdCl,IdApp → DDLoc,DFLoc donne R3(IdCl,IdApp,DDLoc,DFLoc).
4. FD IdProp → NomProp donne R4(IdProp,NomProp) (mais NomProp déjà dans R2).
On conserve l'information et chaque schéma est maintenant en BCNF.