Solutions de l'examen UE Bases de données – 1ère session 2016-2017

Licence 2ème Année

Schéma de la base de données

- **Régate**(Num, NomR, Date)
- **Participant**(Num, IdB, IdE)
- **Bateau**(IdB, Nom, Catégorie)
- **Marin**(IdM, Nom, Prénom, Nationalité)
- **Equipage**(IdE, IdM)

A. Algèbre relationnelle et SQL sans imbrication ni agrégats

A-1. Quels sont les marins (noms) français ?

Algèbre relationnelle :

$$\pi_{Nom}(\sigma_{Nationalit='France'}(Marin))$$

SQL :

```
SELECT Nom
FROM Marin
WHERE Nationalite = 'France';
```

Explication : On filtre d'abord l'ensemble des marins dont la nationalité vaut 'France', puis on projette sur le nom.

A-2. Quels sont les bateaux (identifiants et noms) de catégorie "dériveur" ?

Algèbre relationnelle :

$$\pi_{IdB,Nom}(\sigma_{Catgorie='driveur'}(Bateau))$$

SQL :

```
SELECT IdB, Nom
FROM Bateau
WHERE Categorie = 'deriveur';
```

Explication : On impose la condition de categorie puis on affiche les colonnes d'identifiant et de nom.

A-3. Quelles sont les régates (identifiants) qui ont eu lieu le 10 Mai 2017 ?

Algèbre relationnelle :

$$\pi_{Num}(\sigma_{Date='2017-05-10'}(Rgate))$$

SQL :

```
SELECT Num
FROM Regate
WHERE Date = '2017-05-10';
```

Explication : Dans la table Régate, on sélectionne les tuples dont la date est le 10 mai 2017, puis on projette l'identifiant.

A-4. Quels sont les bateaux (identifiants et noms) de catégorie "dériveur" participant à des régates du 10 Mai 2017 ?

Algèbre relationnelle :

$$\pi_{B.IdB,B.Nom}\big(\sigma_{B.Categorie='driveur'}(Bateau) \bowtie B.IdB = P.IdB \\ \bowtie P.Num = R.Num \bowtie \sigma_{R.Date='2017-05-10'}(Rgate)\big)$$

SQL :

```
SELECT B.IdB, B.Nom
FROM Bateau AS B
JOIN Participant AS P ON B.IdB = P.IdB
JOIN Regate   AS R ON P.Num = R.Num
WHERE B.Categorie = 'deriveur'
  AND R.Date = '2017-05-10';
```

Explication : On joint les trois tables selon les clefs correspondantes, filtre sur la catégorie et la date, puis on projette.

A-5. Marins (noms) ayant participé à des régates du 10 Mai 2017 sur des dériveurs, avec bateau navigué

Algèbre relationnelle :

$$\pi_{M.Nom,B.Nom}\big(Marin\ M \bowtie Equipage\ E \bowtie P.IdE = E.IdE \\ \bowtie P.IdB = B.IdB \bowtie \sigma_{B.Categorie='driveur'}(Bateau) \\ \bowtie P.Num = R.Num \bowtie \sigma_{R.Date='2017-05-10'}(Rgate)\big)$$

SQL :

```
SELECT M.Nom, B.Nom AS Bateau
FROM Marin   AS M
JOIN Equipage AS E ON M.IdM = E.IdM
JOIN Participant AS P ON E.IdE = P.IdE
JOIN Bateau    AS B ON P.IdB = B.IdB
JOIN Regate    AS R ON P.Num = R.Num
WHERE B.Categorie = 'deriveur'
  AND R.Date = '2017-05-10';
```

Explication : On construit la chaîne de jointures de Marin à Régate en passant par Equipage et Participant, filtre la catégorie et la date, puis on affiche le nom du marin et du bateau.

A-6. Marins (identifiants et noms) qui n'ont jamais navigué

Algèbre relationnelle :

$$\pi_{IdM,Nom}(Marin) - \pi_{E.IdM,M2.Nom}\big(Marin\ M2 \bowtie Equipage\ E\big)$$

SQL :

```
SELECT IdM, Nom
FROM Marin
WHERE IdM NOT IN (
  SELECT IdM
  FROM Equipage
);
```

Explication : On retire de l'ensemble des marins ceux dont l'Id figure dans Equipage.

A-7. Marins (identifiants et noms) membres d'au moins deux équipages différents

Algèbre relationnelle :

$$\pi_{IdM,Nom}\Big(\sigma_{c\geq 2}\big(\gamma_{IdM;\mathrm{count}(IdE)\to c}(Equipage)\big) \bowtie Marin\Big)$$

(note : on exprime le comptage en algèbre étendue)

SQL :

```
SELECT M.IdM, M.Nom
FROM Marin AS M
JOIN (
  SELECT IdM
  FROM Equipage
  GROUP BY IdM
  HAVING COUNT(DISTINCT IdE) >= 2
) AS T ON M.IdM = T.IdM;
```

Explication : On regroupe Equipage par marin, on retient ceux avec au moins deux équipages distincts, puis on joint pour obtenir le nom.

B. SQL avec agrégats et sous-requêtes dans `WHERE`

B-1. Combien de régates ont eu lieu en Mai 2017 ?

```
SELECT COUNT(*) AS nb_regates
FROM Regate
WHERE Date BETWEEN '2017-05-01' AND '2017-05-31';
```

Explication : On compte toutes les lignes de Régate dont la date est dans l'intervalle complet de mai.

B-2. Marins (identifiants et noms) faisant partie de strictement plus de 3 équipages

```
SELECT M.IdM, M.Nom
FROM Marin AS M
WHERE (
  SELECT COUNT(DISTINCT IdE)
  FROM Equipage
  WHERE IdM = M.IdM
) > 3;
```

Explication : Pour chaque marin, la sous-requête calcule le nombre d'équipages, puis on filtre.

B-3. Nombre de marins par régate (id et nom de la régate)

```
SELECT R.Num, R.NomR,
       (
         SELECT COUNT(DISTINCT E.IdM)
         FROM Participant AS P
         JOIN Equipage AS E ON P.IdE = E.IdE
         WHERE P.Num = R.Num
       ) AS nb_marins
FROM Regate AS R;
```

Explication : Pour chaque régate, on compte via une sous-requête le nombre de marins distincts dans les équipages participants.

B-4. Équipages ayant participé à toutes les régates appelées "Vendée Globe"

```
SELECT IdE
FROM Participant
WHERE Num IN (
  SELECT Num
  FROM Regate
  WHERE NomR = 'Vendee␣Globe'
)
GROUP BY IdE
HAVING COUNT(DISTINCT Num) = (
  SELECT COUNT(*)
  FROM Regate
  WHERE NomR = 'Vendee␣Globe'
);
```

Explication : On sélectionne les équipages apparaissant dans les régates nommées "Vendée Globe", on groupe par équipage et on ne retient que ceux dont le nombre de participations égale le nombre total de régates "Vendée Globe".

C. Contraintes et effets des mises à jour

Schéma et instance initiale

```
CREATE TABLE LABEL (
  lab_Id VARCHAR PRIMARY KEY,
  Val INT
);
CREATE TABLE LIEN (
  Deb VARCHAR,
  Fin VARCHAR,
  FOREIGN KEY (Deb) REFERENCES LABEL(lab_Id)
    ON UPDATE CASCADE,
  FOREIGN KEY (Fin) REFERENCES LABEL(lab_Id)
    ON DELETE SET NULL
    ON UPDATE SET NULL
);
-- Instance initiale:
-- LABEL:

-- LIEN:
--  (L23,L24), (L24,L25), (L26,L23)
```

1. `INSERT INTO LIEN VALUES ('L23','L56')`

Après l'insertion, la contrainte de clef 'etrangère sur `Fin` (référant LABEL.lab_Id) viole car 'L56' n'existe pas dans LABEL. **Résultat :** L'insertion échoue, l'état ne change.

2. `UPDATE LABEL SET lab_Id='o54' WHERE lab_Id='L24'`

L'òperation modifie la clef primaire L24→o54. Par `ON UPDATE CASCADE` sur `LIEN(Deb)`, tout lien dont Deb='L24' devient Deb='o54'. Par `ON UPDATE SET NULL` sur `LIEN(Fin)`, tout lien dont Fin='L24' devient Fin=NULL.

État intermédiaire : LABEL :

labId	Val
L23	12
o54	16
L25	18
L26	11

LIEN :

Deb	Fin	
L23	NULL	(anciennement L23→L24)
o54	L25	(anciennement L24→L25)
L26	L23	

3. `DELETE FROM LABEL WHERE lab_Id='L23'`

Suppression de L23 de LABEL. Les contraintes :

- `ON DELETE CASCADE` ? Non défini pour Deb, donc RESTRICT par défaut → suppression échoue car L23 est référencé par LIEN(Fin) et LIEN(Deb).
- `ON DELETE SET NULL` ? Défini pour `LIEN(Fin)`, mais la suppression est déjà bloquèe par le premier.

Résultat : La suppression échoue, l'état reste celui de l'étape 2.

D. Formes normales et décomposition

D-1. Avantages d'une décomposition $(R_1, F_1), (R_2, F_2)$

- **Élimination des redondances :** Moins de duplication d'attributs, stockage moindre.
- **Amélioration de la cohérence :** Les anomalies de mise à jour (insertions, suppressions) sont réduites.
- **Requêtes plus ciblées :** Les jointures limitées aux relations concernées peuvent être plus efficaces.
- **Maintenance facilitée :** Compréhension et gestion des responsabilités de chaque relation.

D-2. Inconvénients d'une décomposition

- **Coût des jointures :** Chaque requête multi-relations nécessite une jointure, pouvant être coûteuse.
- **Complexité accrue :** Logique des transactions et contraintes réparties sur plusieurs tables.
- **Problèmes de performance :** Nombre élevé de jointures peut ralentir l'accès pour certaines requêtes.
- **Gestion des contraintes :** Le respect des dépendances requires vérifications croisées.

E. Algorithmes de décomposition et formes normales

Soit $R(VWXYZ)$ et $F = \{XY \to Z, ZX \to W, YZ \to X, W \to V\}$.

E-1. Clés de (R, F)

Pour trouver les clés, on calcule la fermeture de combinaisons minimales :
- $WX^+ = \{W, X, V\}$ (puis V), pas Y, Z
- $XZ^+ = \{X, Z, W, V\}$, pas Y
- $XY^+ = \{X, Y, Z, W, V\}$, contient tous $\to$ $\{X, Y\}$ est clé candidate.
- $YZ^+ : Y, Z \to X$ donne X, puis $X, Y \to Z$ redondant, puis $ZX \to W$, puis $W \to V \to$ tout. Donc $\{Y, Z\}$ est clé.
- Aucune singleton ne couvre tous $\to$ clés : $\{X, Y\}$ et $\{Y, Z\}$.

E-2. R est-elle en 3FN ?

On teste chaque DF dans F :
- $XY \to Z$: XY est super-clé $\to$ OK.
- $ZX \to W$: ZX contient clé $\{Y, Z\}$? Non, Z, X non super-clé $\to$ W **non-prime** ? W n'appartient à aucune clé $\to$ violation.
- $YZ \to X$: YZ est clé $\to$ OK.
- $W \to V$: W non super-clé et V non prime $\to$ violation.

Donc R n'est pas en 3FN.

E-3. Décomposition en 3FN

On choisit relations pour chaque DF violant :

$$\begin{aligned} R_1 &= (Z, X, W) && \text{pour } ZX \to W, \\ R_2 &= (W, V) && \text{pour } W \to V, \\ R_3 &= (X, Y, Z) && \text{pour } XY \to Z \text{ et } YZ \to X. \end{aligned}$$

On note que R_3 couvre les DF entre X, Y, Z, et R_1, R_2 éliminent les dépendances transverses. Ces relations sont maintenant en 3FN, seule clef dans R_1 est ZX, dans R_2 W, et dans R_3 XY (ou YZ).