Matthieu Boyer
Planche A

Question de Cours

On dispose du schéma relationnel suivant :

`Gare(nom TEXT, ville TEXT, creation TEXT)`

`Train(id INTEGER, depart TEXT, arrivee TEXT, horaire_depart INTEGER, horaire_arrivee INTEGER)`

Les champs `depart` et `arrivee` de `Train` sont des clés étrangères vers `Gare.nom`. Les horaires sont des entiers représentant des minutes depuis minuit.

Donnez une requête SQL renvoyant la liste des correspondances réalisables, i.e. des paires de trains de sorte que le premier arrive dans la gare d'où part le second avant le départ d'icelui, ainsi que le temps d'attente entre les trains.

Category of Sets

On s'intéresse dans cet exercice à la bonne définition des opérations SQL vues comme opérations entre des ensembles. On introduit de plus l'ensemble $\mathbb{B} = \{\bot, \top\}$, ainsi que deux tables T_1, T_2.

1. Rappelez les requêtes SQL permettant de renvoyer la table contenant toutes les paires t_1, t_2 d'éléments de T_1 et T_2.
2. Le produit cartésien de deux ensembles A et B est un ensemble noté $A \times B$ tel qu'on ait deux fonctions π_A, π_B envoyant $A \times B$ dans A et B respectivement, et tel que si on a $f : S \to A$ et $g : S \to B$ il existe une unique application $f \times g : S \to A \times B$ telle que $f = \pi_A \circ (f \times g)$ et $g = \pi_B \circ (f \times g)$. Montrer que le produit cartésien de T_1 et T_2 est unique à bijection près, et est l'ensemble renvoyé par la requête SQL de la question précédente.
3. Proposez une définition similaire pour le produit cartésien de n ensembles, et la requête SQL associée.
4. L'égaliseur de deux applications $f, g : A \to B$ est un ensemble E avec une application $eq : E \to A$ telle que $f \circ eq = g \circ eq$, et tel que si O est un ensemble avec une application $m : O \to A$ telle que $f \circ m = g \circ m$, alors il existe une unique application $u : O \to E$ telle que $m = eq \circ u$. Montrer que l'égaliseur de deux applications est unique à bijection près. Exprimer l'égaliseur de $f : n \in \mathbb{N} \mapsto (n \geq 12) \in \mathbb{B}$ et de $g : n \in \mathbb{N} \mapsto \top \in \mathbb{B}$ via une requête SQL sur une table `N(n INTEGER)`.
5. Le produit fibré selon f, g sur C de deux ensembles A et B, noté $A \times_C B$, est un ensemble tel qu'on dispose de deux applications π_A, π_B de sorte que si on a $p : S \to A$ et $q : S \to B$ telles que $f \circ p = g \circ q$ alors il existe une unique application i telle que $p = \pi_A \circ i$ et $q = \pi_B \circ i$. Montrer que le produit fibré de T_1 et T_2 est unique à bijection près, et exprimer celui de T_1, T_2 selon $t \in T_1 \mapsto$ `t.id` $\in \mathbb{N}$ et $t \in T_2 \mapsto$ `t.id` $\in \mathbb{N}$ via une requête SQL.
6. Exprimer le produit fibré selon f, g sur C de A et B à partir des constructions précédentes. Comment traduirait-on cette nouvelle construction en SQL ?
7. Exprimer le coproduit de deux ensembles A et B (un ensemble noté $A \sqcup B$ tel qu'on ait deux fonctions π_A, π_B envoyant A et B dans $A \times B$ respectivement, et tel que si on a $f : A \to S$ et $g : A \to S$ il existe une unique application $f \times g : S \to A \times B$ telle que $f = (f \times g) \circ \pi_A$ et $g = (f \times g) \circ \pi_B$. Montrer que le coproduit de T_1 et T_2 est unique à bijection près, et donnés une requête SQL le calculant.

Matthieu Boyer
Planche A : Corrigés

Solution : Question de Cours

On effectue une auto-jointure de `Train` avec lui-même : le premier exemplaire `T1` représente le train d'arrivée, le second `T2` le train au départ. La condition de correspondance est que le lieu d'arrivée de `T1` coïncide avec le lieu de départ de `T2`, et que l'horaire d'arrivée de `T1` soit strictement antérieur à l'horaire de départ de `T2`.

```
SELECT T1.id AS train_arrivee,
       T2.id AS train_depart,
       T2.horaire_depart - T1.horaire_arrivee AS temps_attente
FROM Train T1, Train T2
WHERE T1.arrivee = T2.depart
      AND T1.horaire_arrivee < T2.horaire_depart;
```

Le champ calculé `temps_attente` donne le nombre de minutes entre l'arrivée du premier train et le départ du second. La condition `T1.id <> T2.id` n'est pas nécessaire ici car un train ne peut pas arriver avant de partir, mais on peut l'ajouter par clarté.

Si l'on souhaite également afficher les noms des gares de départ et d'arrivée de chaque train, on peut compléter la requête par deux jointures supplémentaires avec `Gare`.

Solution : Category of Sets

1. Le produit cartésien de T_1 et T_2 s'obtient par une jointure croisée :

   ```
   SELECT * FROM T1, T2;
   ```
   ou de manière équivalente :
   ```
   SELECT * FROM T1 CROSS JOIN T2;
   ```

2. Soient P_1 et P_2 deux produits cartésiens de T_1 et T_2, munis respectivement de leurs projections (π_A^1, π_B^1) et (π_A^2, π_B^2). En appliquant la propriété universelle de P_1 à $S = P_2$, $f = \pi_A^2$ et $g = \pi_B^2$, il existe une unique application $u : P_2 \to P_1$ telle que $\pi_A^1 \circ u = \pi_A^2$ et $\pi_B^1 \circ u = \pi_B^2$. Symétriquement, il existe une unique $v : P_1 \to P_2$. La composée $u \circ v : P_1 \to P_1$ satisfait $\pi_A^1 \circ (u \circ v) = \pi_A^1$ et $\pi_B^1 \circ (u \circ v) = \pi_B^1$, et l'identité id_{P_1} aussi. P ar unicité, $u \circ v = \mathrm{id}_{P_1}$, et de même $v \circ u = \mathrm{id}_{P_2}$. Donc u est une bijection.

   La jointure croisée réalise bien le produit. Notons $P =$ `SELECT * FROM T1 CROSS JOIN T2` ; ses éléments sont les paires (t_1, t_2), et les projections naturelles sont $\pi_A : (t_1, t_2) \mapsto t_1$ et $\pi_B : (t_1, t_2) \mapsto t_2$. Pour tout ensemble S et toutes applications $f : S \to T_1$, $g : S \to T_2$, l'unique application $f \times g : s \mapsto (f(s), g(s))$ vérifie bien $\pi_A \circ (f \times g) = f$ et $\pi_B \circ (f \times g) = g$, d'où la propriété universelle. L'unicité de $f \times g$ est claire car les deux composantes sont imposées.

3. Le produit cartésien de n ensembles $A_1, \ldots, A_n$ est un ensemble P muni de n projections $\pi_k : P \to A_k$ $(1 \leq k \leq n)$ tel que pour tout ensemble S et toute famille d'applications $f_k : S \to A_k$, il existe une unique application $\langle f_1, \ldots, f_n \rangle : S \to P$ vérifiant $\pi_k \circ \langle f_1, \ldots, f_n \rangle = f_k$ pour tout k. Un tel produit est unique à bijection près par le même argument qu'à la question 2.

4. Unicité à bijection près de l'égaliseur : Soient E_1 (avec eq_1) et E_2 (avec eq_2) deux égaliseurs de $f, g : A \to B$. En appliquant la propriété universelle de E_1 à $O = E_2$ et $m = eq_2$ (qui vérifie bien $f \circ eq_2 = g \circ eq_2$), on obtient une unique $u : E_2 \to E_1$ telle que $eq_1 \circ u = eq_2$. Symétriquement on a $v : E_1 \to E_2$ avec $eq_2 \circ v = eq_1$. Alors $eq_1 \circ (u \circ v) = eq_1$, d'où $u \circ v = \mathrm{id}_{E_1}$ par unicité, et u est une bijection.

   L'égaliseur de f et g est l'ensemble $\{n \in \mathbb{N} \mid f(n) = g(n)\} = \{n \in \mathbb{N} \mid n \geq 12\}$.

   ```
   SELECT n FROM N
   WHERE n >= 12;
   ```

5. On adapte le même argument d'unicité : si P_1 et P_2 sont deux produits fibrés, on obtient des applications réciproques $u : P_2 \to P_1$ et $v : P_1 \to P_2$ en appliquant les propriétés universelles respectives, puis on conclut par unicité que u et v sont des bijections réciproques.

   Le produit fibré de T_1 et T_2 selon $t.id$ est l'ensemble des paires $(t_1, t_2) \in T_1 \times T_2$ telles que $t_1.\texttt{id} = t_2.\texttt{id}$:

```
SELECT *
FROM T1 JOIN T2 ON T1.id = T2.id;
```

6. Le produit fibré $A \times_C B$ peut s'exprimer comme l'égaliseur des deux composées $f \circ \pi_A$ et $g \circ \pi_B$ sur le produit cartésien $A \times B$:

$$A \times_C B = \mathrm{Eq}\big(f \circ \pi_A, g \circ \pi_B : A \times B \to C\big).$$

   En d'autres termes, on prend d'abord le produit cartésien, puis on en extrait la partie sur laquelle $f \circ \pi_A = g \circ \pi_B$. La traduction SQL est immédiate : on effectue un `CROSS JOIN` (produit cartésien) suivi d'une clause `WHERE` (égaliseur), ce qui revient exactement à une jointure naturelle sur l'égalité $f(a) = g(b)$:

```
SELECT *
FROM A CROSS JOIN B
WHERE f(A) = g(B);
```

   C'est bien la forme générale d'un `JOIN` en SQL.

D'un point de vue catégorique, la bonne définition des opérations en SQL équivaut à l'existence de toutes les limites dans Set.

Matthieu Boyer
Planche B

Question de Cours

On dispose du schéma relationnel suivant :

`Film(titre TEXT, realisateur TEXT, acteur TEXT)`

`Personne(nom TEXT, profession TEXT, age INTEGER)`

Donnez une requête SQL renvoyant la liste des titres et des ages des acteurs principaux des films dont le réalisateur est « Tim Burton ».

Produits Tensoriels en SQL

Un tenseur est une généralisation des matrices (tableaux à double entrée) à un nombre arbitraire de dimensions. Un *tenseur d'ordre n* à valeurs dans un corps $\mathbb{K}$ est une application

$$T : I_1 \times I_2 \times \cdots \times I_n \longrightarrow \mathbb{K},$$

où $I_1, \ldots, I_n$ sont des ensembles d'indices finis. L'entier n est appelé l'*ordre* (ou rang) du tenseur. Un scalaire est un tenseur d'ordre 0, un vecteur d'ordre 1, une matrice d'ordre 2.

Pour stocker un tenseur en SQL, on se donne une table dont les colonnes $i_1, i_2, \ldots, i_n$ représentent les coordonnées de chaque entrée non nulle, et une colonne `val` contient la valeur. Par exemple, une matrice A (tenseur d'ordre 2) est stockée selon le schéma (appelé format COO)

`A(i INTEGER, j INTEGER, val REAL)`.

1. Donnez le schéma COO pour un vecteur $u \in \mathbb{K}^n$, et un tenseur d'ordre 3, $T \in \mathbb{K}^{d_1 \times d_2 \times d_3}$. Donnez la table qui représenterait la matrice $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$.
2. Exprimez en SQL le produit scalaire
$$s = \langle u, v \rangle = \sum_i u_i v_i.$$
3. Exprimez en SQL le produit matriciel $C = A \cdot B$, i.e.
$$c_{ij} = \sum_k a_{ik} b_{kj}, .$$
4. Exprimez en SQL les opérations suivantes :
   (a) La *trace* d'une matrice carrée A : $s = \sum_i a_{ii}$.
   (b) Le *produit de Hadamard* (produit terme à terme) : $c_{ij} = a_{ij} b_{ij}$.
   (c) Le *produit extérieur* de deux vecteurs : $c_{ij} = u_i v_j$ (aucun indice contracté).
5. Soit une expression de sommation d'Einstein générale
$$C_{\mathbf{f}} = \sum_{\mathbf{c}} A_{\alpha(\mathbf{f},\mathbf{c})} \cdot B_{\beta(\mathbf{f},\mathbf{c})},$$
où $\mathbf{f} = (f_1, \ldots, f_p)$ désigne les indices libres et $\mathbf{c} = (c_1, \ldots, c_q)$ les indices contractés. Donnez le template SQL général permettant de calculer C à partir de A et B.

Solution : Question de Cours

On effectue une jointure entre `Film` et `Personne` sur l'attribut commun `acteur`/`nom`, puis on filtre sur le réalisateur.

```
SELECT F.titre, P.age
FROM Film F, Personne P
WHERE F.acteur = P.nom
      AND F.realisateur = 'Tim Burton';
```

On peut également écrire la jointure de façon explicite :

```
SELECT F.titre, P.age
FROM Film F
JOIN Personne P ON F.acteur = P.nom
WHERE F.realisateur = 'Tim Burton';
```

Les deux formulations sont équivalentes. La clause `WHERE` filtre les lignes dont le champ `realisateur` vaut exactement la chaîne « Tim Burton », et la jointure sur `acteur = nom` garantit que l'âge renvoyé est bien celui de la personne ayant joué dans le film. Il vaut mieux utiliser la seconde méthode, la jointure explicite évitant des problèmes d'interprétation.

Solution : Produits Tensoriels en SQL

1. Les schémas COO sont les suivants :

```
u(i INTEGER, val REAL)
M(i INTEGER, j INTEGER, val REAL)
T(i INTEGER, j INTEGER, k INTEGER, val REAL)
```

   Les colonnes d'indices forment la clé primaire (on ne stocke que les entrées non nulles).

2. Le produit scalaire $s = \sum_i u_i v_i$ est calculé par :

```
SELECT SUM(u.val * v.val) AS s
FROM u, v
WHERE u.i = v.i;
```

   La condition `WHERE u.i = v.i` réalise une *jointure* sur l'indice contracté i, et `SUM` effectue la *sommation*. Il n'y a pas d'indice libre, donc pas de `GROUP BY` : le résultat est un scalaire (une seule ligne).

3. Le produit matriciel $c_{ij} = \sum_k a_{ik} b_{kj}$ est calculé par :

```
SELECT A.i, B.j, SUM(A.val * B.val) AS val
FROM A, B
WHERE A.j = B.i
GROUP BY A.i, B.j;
```

   La jointure `WHERE A.j = B.i` réalise la contraction sur l'indice k (qui apparaît comme `A.j` dans A et comme `B.i` dans B) ; `SUM` effectue la sommation ; `GROUP BY A.i, B.j` regroupe selon les indices libres i et j du résultat C.

4. (a) $s = \sum_i a_{ii}$ (indice i contracté avec lui-même) :

```
SELECT SUM(val) AS trace
FROM A
WHERE i = j;
```

On filtre les lignes diagonales (`i = j`) et on somme leurs valeurs. Il n'y a qu'un seul tenseur en entrée, donc pas de jointure entre deux tables.

(b) $c_{ij} = a_{ij} b_{ij}$ (pas d'indice contracté, deux indices libres) :

```
SELECT A.i, A.j, A.val * B.val AS val
FROM A, B
WHERE A.i = B.i AND A.j = B.j;
```

Les deux indices i et j sont partagés par les deux opérandes mais sont des indices libres (ils apparaissent dans le résultat), donc il n'y a pas de `GROUP BY` ni de `SUM`.

(c) **Produit extérieur.** $c_{ij} = u_i v_j$ (aucun indice contracté) :

```
SELECT u.i, v.i AS j, u.val * v.val AS val
FROM u, v;
```

Il s'agit d'un simple produit cartésien : chaque entrée de u est multipliée par chaque entrée de v, sans condition de jointure ni agrégation.

5. Le patron SQL général pour $C_{\mathbf{f}} = \sum_{\mathbf{c}} A_\alpha \cdot B_\beta$ est :

```
SELECT A.<f1_in_A>, ..., B.<f1_in_B>, ..., SUM(A.val * B.val) AS val
FROM A, B
WHERE A.<c1_in_A> = B.<c1_in_B>
      AND ...
GROUP BY A.<f1_in_A>, ..., B.<f1_in_B>, ...;
```

Plus précisément :

— Le `FROM A, B` forme le produit cartésien des deux tables.

— Le `WHERE` impose l'égalité des colonnes correspondant à chaque indice contracté c_k (jointure).

— Le `SELECT` liste les colonnes des indices libres f_l, en préfixant chacune par le nom de la table dans laquelle cet indice apparaît.

— Le `GROUP BY` regroupe sur ces mêmes indices libres et `SUM` effectue la sommation.

Ce patron s'étend immédiatement à plus de deux tenseurs en entrée en ajoutant des termes au `FROM` et au `WHERE`.

6. La requête cherche les couples (x, z) tels qu'il existe y vérifiant $(x, p_1, y) \in T$ et $(y, p_2, z) \in T$. On effectue une auto-jointure de `T` sur $T_1.o = T_2.s$ (le nud intermédiaire y), en filtrant sur les prédicats p_1 et p_2 :

```
SELECT T1.s AS x, T2.o AS z
FROM T T1, T T2
WHERE T1.p = p1
      AND T2.p = p2
      AND T1.o = T2.s;
```

Interprétation tensorielle. On peut définir les matrices de prédicat $M_{so}^{(p)} = T_{s,p,o}$ pour un prédicat p fixé. La requête calcule alors

$$R_{xz} = \sum_y M_{xy}^{(p_1)} M_{yz}^{(p_2)},$$

qui est exactement un produit matriciel entre $M^{(p_1)}$ et $M^{(p_2)}$ en notation d'Einstein. La jointure sur $y = T_1.o = T_2.s$ réalise la contraction tensorielle, et le résultat R est la table de tous les couples (x, z) atteignables par le chemin de longueur 2 de prédicats p_1, p_2 dans le graphe de connaissances.