Proposition de correction de l'examen du cours *Conception et traitement des données*

Claire Medrala (MESR), Fabien Coelho (Mines Paris – PSL)

Janvier 2026

Bravo Hobbes ! 20/20

1 Contraintes *3*/3

Voici l'expression des contraintes sur le schéma :

1. Les valeurs possibles des attributs `caccepte` et `cconfirme` de la relation `Choix` sont liées par la progression du processus de candidature. Version longue et explicite :

```
ALTER TABLE Choix ADD CONSTRAINT coherence_etats_des_choix_1
  CHECK (caccepte IS NULL AND cconfirme IS NULL OR
         caccepte IS NOT NULL AND cconfirme IS NULL OR
         caccepte IS TRUE AND cconfirme IS NOT NULL);
```

   et version courte et astucieuse :

```
ALTER TABLE Choix ADD CONSTRAINT coherence_etat_des_choix_2
  CHECK (cconfirme IS NULL OR caccepte IS TRUE);
```

2. Pour une année donnée, un étudiant ne peut s'inscrire qu'à un seul diplôme.

```
CREATE UNIQUE INDEX une_seule_inscription
  ON Choix(sid, annee)
  WHERE cconfirme IS TRUE;
```

3. Un étudiant ne peut avoir que trois candidatures actives simultanées, de priorités distinctes.

```
CREATE UNIQUE INDEX trois_candidatures_actives_prio_distinctes
  ON Choix(sid, annee, cpriorite)
  WHERE caccepte IS NULL OR caccepte AND cconfirme IS NULL;
```

2 Requêtes *6*/6

1. Quelles sont les choix de l'étudiante *Susie* pour l'année scolaire *2025-2026* concernant des diplômes de type *BUT* dans l'académie de *Saint-Martin* ayant au moins *100* places offertes ?

```
SELECT c.*
FROM Etudiant AS s
JOIN Choix AS c USING (sid)
JOIN Annee AS an USING (annee)
JOIN Diplome AS d USING (did)
JOIN Etablissement AS e USING (eid)
JOIN Academie AS ac ON ac.aid = e.aid
JOIN TypeDiplome AS t USING (tid)
JOIN Place AS p USING (annee, did)
WHERE s.snom = 'Susie'
  AND an.annee_universitaire = '2025-2026'
  AND ac.anom = 'Saint-Martin'
  AND t.tnom = 'BUT'
  AND p.pcal >= 100;
```

2. En partant du nom de l'étudiante, un seul index est nécessaire pour naviguer entre les tables et retrouver les informations demandées.

```
-- Etudiant snom -> sid
CREATE INDEX etudiant_nom ON Etudiant(snom);
-- Choix sid -> cid : unique(did, annee, sid) n'est pas utilisable
CREATE INDEX choix_sid ON Choix(sid);
-- Annee : annee est PK
-- Diplome : did est PK
-- Academie : aid est PK
-- TypeDiplome : tid est PK
-- Place (annee, did) : unique(annee, did) est utilisable
```

3. Comment trouver les INE et les noms des étudiants qui ont exactement une seule candidature **active** pour l'année universitaire *2026-2027*, par ordre des INE ?

```
SELECT s.sine, s.snom
FROM Etudiant AS s
JOIN Choix AS c USING (sid)
JOIN Annee AS a USING (annee)
WHERE a.annee_universitaire = '2026-2027'
  AND (c.caccepte IS NULL OR (c.caccepte IS TRUE AND c.cconfirme IS NULL))
GROUP BY 1, 2
HAVING COUNT(*) = 1
ORDER BY 1;
```

4. Quels étudiants (INE et nom) ayant exprimé un choix n'ont été acceptés par aucun diplôme pour l'année universitaire *2026-2027* ?

   Voici 5 approches pour répondre à la question : les deux suivantes avec `HAVING` astucieux, la dernière avec une sous requête :

   Avec un `EXCEPT` direct :

```
-- les étudiants avec des choix 2026
SELECT sine, snom
  FROM Etudiant
  JOIN Choix USING (sid)
  JOIN Annee USING (annee)
 WHERE annee_universitaire = '2026-2027'
EXCEPT
-- les étudiants acceptés en 2026
SELECT sine, snom
  FROM Etudiant
  JOIN Choix USING (sid)
  JOIN Annee USING (annee)
 WHERE caccepte IS TRUE  -- caccepte IS NOT NULL AND caccepte
   AND annee_universitaire = '2026-2027'
ORDER BY sine;
```

Avec un EXCEPT sur un WITH :

```sql
-- précalcule des étudiants avec des choix en 2026...
WITH
  etudiants_choix_2026 AS (
    SELECT sine, snom, caccepte
      FROM Etudiant
      JOIN Choix USING (sid)
      JOIN Annee USING (annee)
     WHERE annee_universitaire = '2026-2027'
  )
-- puis différence
SELECT sine, snom FROM etudiants_choix_2026
EXCEPT
SELECT sine, snom FROM etudiants_choix_2026
WHERE caccepte IS TRUE  -- caccepte IS NOT NULL AND caccepte
ORDER BY sine;
```

Avec un HAVING astucieux :

```sql
-- having astucieux : count compte les valeurs non NULL
SELECT sine, snom
FROM Etudiant
JOIN Choix USING (sid)
JOIN Annee USING (annee)
WHERE annee_universitaire = '2026-2027'
GROUP BY sine, snom
HAVING COUNT(caccepte IS TRUE OR NULL) = 0
ORDER BY sine;
```

Avec un HAVING et un FILTER :

```sql
-- having count et filter
SELECT sine, snom
FROM Etudiant
JOIN Choix USING (sid)
JOIN Annee USING (annee)
WHERE annee_universitaire = '2026-2027'
GROUP BY sine, snom
HAVING COUNT(*) FILTER (WHERE caccepte IS TRUE) = 0
ORDER BY sine;
```

Avec une sous-requête sur un WITH :

```sql
-- précalcule des étudiants avec des choix en 2026...
WITH
  etudiants_choix_2026 AS (
    SELECT sine, snom, caccepte
      FROM Etudiant
      JOIN Choix USING (sid)
      JOIN Annee USING (annee)
     WHERE annee_universitaire = '2026-2027'
  )
-- puis sous requête de comptage des acceptations
SELECT DISTINCT sine, snom
FROM etudiants_choix_2026 AS e1
WHERE (SELECT COUNT(*)
       FROM etudiants_choix_2026 AS e2
       WHERE e2.sine = e1.sine
         AND e2.caccepte IS TRUE) = 0
ORDER BY sine;
```

5. Pour chaque année universitaire et chaque numéro de diplôme, donner le nombre de choix, le nombre d'acceptation par l'établissement et le nombre de confirmation par l'étudiant, par ordre des années universitaires les plus récentes et des numéros de diplômes.

```sql
SELECT
  a.annee_universitaire, d.dinf,
  COUNT(*) AS "demandes",
  COUNT(*) FILTER (WHERE c.caccepte IS TRUE) AS "acceptes",
  COUNT(*) FILTER (WHERE c.cconfirme IS TRUE) AS "confirmes"
FROM Choix AS c
JOIN Annee AS a USING (annee)
JOIN Diplome AS d USING (did)
GROUP BY 1, 2
ORDER BY 1 DESC, 2 ASC;
```

6. Pour **toutes** les académies et **tous** les types de diplômes, donner le pourcentage des diplômes de ce type par rapport à l'ensemble des diplômes de l'académie, par ordre des académies et des types.

   Version avec CTE *(Common Table Expression)* :

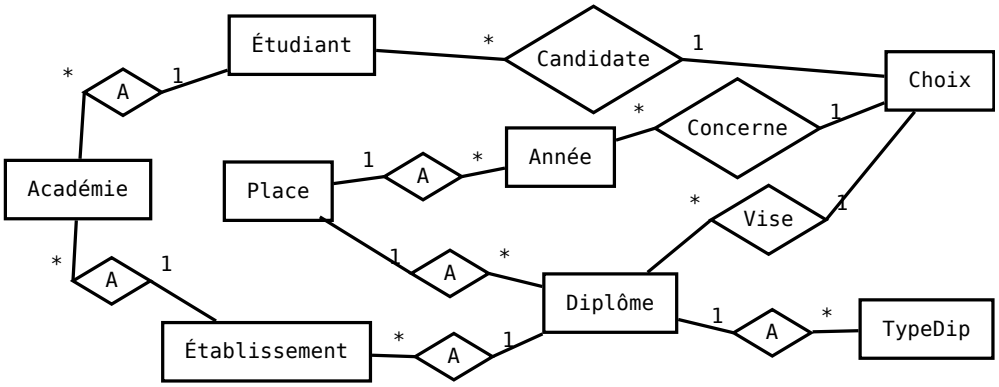
```sql
WITH NbDiplomeAcademie AS (
  SELECT a.aid, a.anom, COUNT(*) AS nb
  FROM Academie AS a
  JOIN Etablissement AS etab USING (aid)
  JOIN Diplome AS d USING (eid)
  GROUP BY 1, 2
)
SELECT
  a.anom AS "académie",
  t.tnom AS "type",
  ROUND(100.0 * COUNT(d.did) / a.nb, 1) AS "%"
FROM NbDiplomeAcademie AS a
CROSS JOIN TypeDiplome AS t
JOIN Etablissement AS etab USING (aid)
LEFT JOIN Diplome AS d USING (eid, tid)
GROUP BY 1, 2, a.nb
ORDER BY 1, 2;
```

   Version avec des fonctions de fenêtrage :

```sql
SELECT DISTINCT
  a.anom AS "académie",
  t.tnom AS "type",
  ROUND(100.0 * COUNT(d.did) OVER (PARTITION by a.anom, t.tnom) /
        COUNT(d.did) OVER (PARTITION BY a.anom), 1) AS "%"
FROM Academie AS a
CROSS JOIN TypeDiplome AS t
-- l'établissement porte son académie
JOIN Etablissement AS etab USING (aid)
-- les diplômes de ce type et de l'établissement
LEFT JOIN Diplome AS d USING (eid, tid)
ORDER BY 1, 2;
```

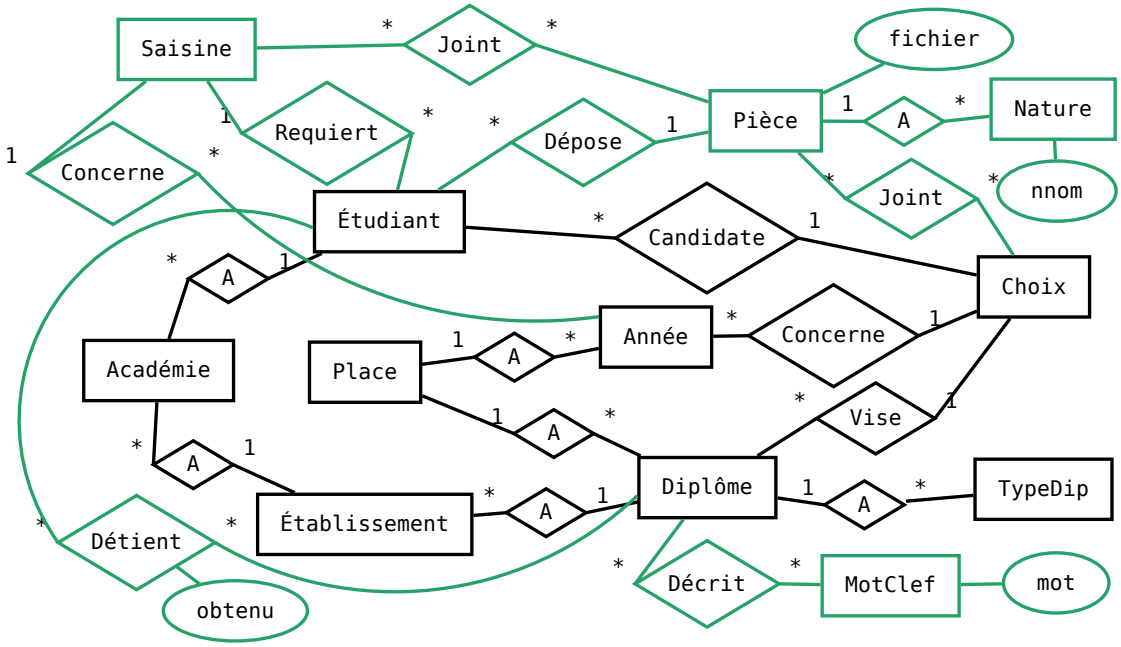
3 Modélisation E/A et traduction relationnelle 7/7

1. Voici une proposition de modèle E/A pour *SeFormer* :



Remarque : le vocabulaire utilisé pour les associations est assez pauvre...

2. Voici une proposition de modèle E/A pour étendre *SeFormer* :



L'académie de la saisine n'est accessible que via l'étudiant, ce qui est discutable si on a besoin d'y accéder par ce biais mais évite une petite redondance. La saisine est sous-modélisée : on ne dit pas quand elle a lieu, comment le recteur peut répondre...

```
CT Détient(sid INNR Étudiant, did INNR Diplôme, PK(sid, did), obtenu DNN)
CT Nature (nid SPK, nnom TUNN)
CT Pièce(pid SPK, sid INNR Étudiant, fichier TUNN, nid INNR Nature)
CT PièceChoix(pid INNR Pièce, cid INNR Choix, PK(pid, cid))
CT MotClef(mid SPK, mot TUNN)
CT Décrit(did INNR Diplôme, mid INNR MotClef, PK(did, mid))
CT Saisine(said SPK, annee INNR Annee, sid INNR Étudiant, U(sid, annee))
CT PièceSaisine(pid INNR Pièce, said INNR Saisine, PK(pid, said))
```

4 Cours *4/4*

***Thème** : ACID (Advanced Chanting In Databases)*

Ce thème est trés passionnant, j'ai appris plein de choses super intéressantes qui éclairent parfaitement le fonctionnement de cet aspect des bases de données à la fois théorique et pratique, et me permettent de bien comprendre les caractéristiques générales et particulières des multiples facettes de ce thème dans une perspective transversale de mise en application concrète du modèle relationnel sur des problèmes réels, tout en gardant, au delà du simple niveau fonctionnel, une maîtrise généraliste des aspects opérationnels sur Postgres qui gére le stockage à un prix compatible avec toutes les bourses, une bonne nouvelle pour nos budgets !

Les deux principaux contributeurs à l'invention du modèle relationnel sont *David Child* et *Edgard (Ted) Codd*.