

Projet BDD : Collections interconnectées dans une chaîne de bibliothèques

Groupe : Collections-interconnect-es-dans-une-cha-ne-de-biblioth-ques

Liens du GitHub : [Github](#)

Axel Derail—Santer
Marius Barbaud
Matthieu Griffonnet

Tables des matières

Projet BDD : Collections interconnectées dans une chaîne de bibliothèques	1
Tables des matières	3
1. Reformulation du sujet :	4
2. Le schéma de la base de données	6
3. L'expression en SQL des requêtes sur la base de données.....	8
Création d'un nouvel 'order' et création ou récupération d'un nouveau 'transfert' .	8
Rendu d'un 'lending' et récupération des livres associés au 'holdings' de la bibliothèque ayant effectué le prêt.	11
4. Questions principales :	14
Question 1 :	14
Question 2 :	16
Question 3 :	17
Question 4 :	18
Question 5 :	18
Question 6 :	19
5. La description du jeu de données utilisé pour tester les différentes requêtes	20
6. Une analyse critique du déroulement du projet	24
7. Vidéo	25

1. Reformulation du sujet :

Gestion Centralisée d'une Chaîne de Bibliothèques

Contexte et Objectifs :

La chaîne de bibliothèques vise à centraliser et harmoniser la gestion de ses activités, notamment les collections d'ouvrages, les abonnés, les événements et les prêts. Ce projet a pour objectif de concevoir une base de données robuste et adaptée aux besoins spécifiques des bibliothèques.

Définition des Bibliothèques :

Chaque bibliothèque est identifiée par un ID unique, un nom, une adresse complète et une région géographique. Elle emploie un ou plusieurs salariés pour assurer son fonctionnement et gère plusieurs exemplaires d'ouvrages.

Les bibliothèques offrent trois principaux services : la commande d'ouvrages entre bibliothèques, le prêt d'ouvrages aux abonnés et l'organisation d'événements culturels.

Gestion des Ouvrages :

Chaque ouvrage est défini par un ISBN unique, un titre, un auteur, un éditeur, une date de publication et une collection éventuelle. Les ouvrages existent en plusieurs exemplaires. Chaque exemplaire est associé à une bibliothèque spécifique et est localisé avec précision grâce à une description de son emplacement comprenant la pièce, l'étagère et le rang.

Gestion des Abonnés :

Un abonné est identifié par une adresse email unique et possède un nom, un prénom et une date de naissance.

La base de données doit permettre de consulter l'historique des prêts pour chaque abonné.

Les abonnés peuvent emprunter des ouvrages pour une durée initiale d'un mois, renouvelable une fois pour deux semaines. Ils doivent sélectionner une bibliothèque pour retirer et retourner les ouvrages empruntés. Les retards de restitution entraînent l'enregistrement d'une infraction.

Transferts d'Ouvrages entre Bibliothèques :

Les transferts peuvent être demandés dans deux cas : pour satisfaire une demande d'un abonné ou pour répondre aux besoins internes d'une bibliothèque.

Les transferts sont organisés une fois par semaine, à condition qu'au moins cinq ouvrages soient prêts pour la même destination. Chaque transfert doit être supervisé par un employé de la bibliothèque d'origine, tandis qu'un employé de la bibliothèque destinataire doit être présent lors de la réception. Aucun transfert ne peut avoir lieu si des événements sont en cours dans l'une des deux bibliothèques concernées.

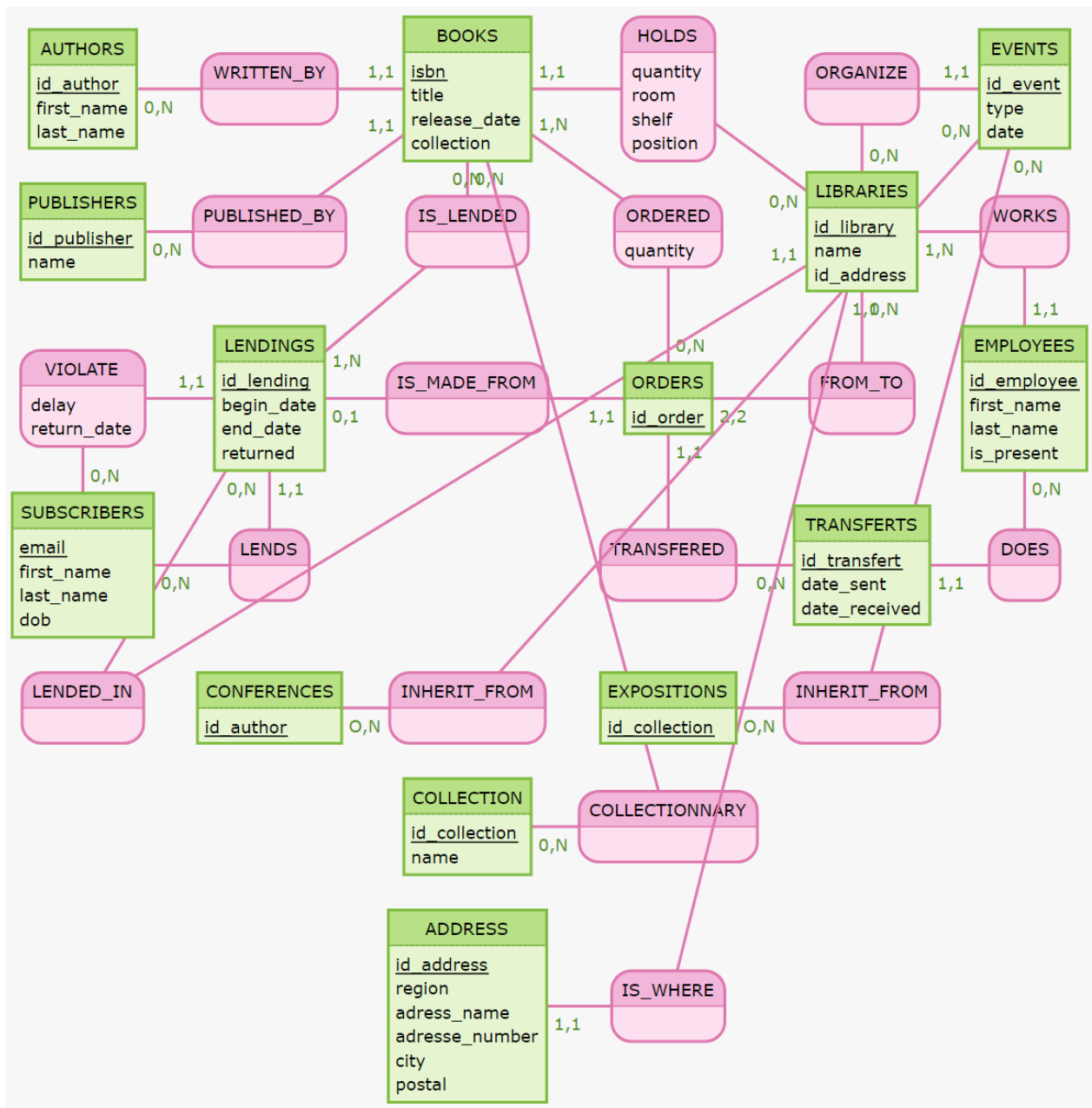
Organisation des Événements :

Chaque bibliothèque peut organiser des événements, identifiés par un ID unique, un type et une date.

Deux types d'événements sont possibles :

- Les conférences, où un auteur présente ses ouvrages.
- Les expositions, qui mettent en valeur des ouvrages autour d'une thématique spécifique.

2. Le schéma de la base de données



Un livre est écrit par un auteur, qui écrit 0 ou plusieurs livres, il est publié par un éditeur, qui publie 0 ou plusieurs livres. Ainsi quand on supprime un livre de la base, on ne supprime pas l'éditeur où l'auteur si c'était le dernier livre lié.

Un livre peut faire partie de 0 ou plusieurs collections.

Ce qu'est une collection est à décider des bibliothécaires. Cela pourrait très bien être les livres sur l'Égypte antiques ou la saga des Harry Potter.

Un livre peut être prêté.

Un prêt est composé d'un livre ou plus.

Un prêt est contracté par un abonné.

Un abonné peut ne pas respecter la date de retour de son prêt, on stocke ces retards dans la relation violate.

Un prêt, si certains livres ne sont pas disponibles dans la bibliothèque ou le prêt a été contracté peut donner lieu à une commande.

Une commande est composée d'un ou plusieurs transferts.

On crée un transfert par bibliothèque dans lesquels on a besoin d'un livre. On choisira les bibliothèques qui contient le plus grand nombre d'exemplaires d'un livre donné.

Un transfert est donc composé d'un ou plusieurs livres.

Un transfert a une bibliothèque de départ et une bibliothèque d'arrivée, qui est la bibliothèque dans laquelle a été contracté le prêt à l'origine de la commande et du transfert.

Un transfert à une date de départ qui est la date courante, et une date d'arrivée qui est la date courante+ 1 semaine

Un transfert est effectué par un employé de la bibliothèque d'où il part.

Une bibliothèque contient 0 ou plusieurs livres. On choisit de ne pas suivre chaque exemplaire individuel. Ainsi on stocke dans la relation holds la quantité d'un livre dans une bibliothèque. On stocke aussi le numéro de la salle, de l'étagère et leur rang sur l'étagère. On considère que tous les exemplaires d'un livre sont côte à côte.

Une bibliothèque a une adresse.

Une bibliothèque peut organiser des événements. Un événement peut être une conférence ou une exposition.

Un événement dure une journée. Il ne peut pas y avoir un événement et un transfert dans une bibliothèque le même jour.

Un employé travaille pour une bibliothèque.

[Lien base de donnée en sql](#)

3. L'expression en SQL des requêtes sur la base de données

Création d'un nouvel 'order' et création ou récupération d'un nouveau 'transfert'

Fait par : Matthieu Griffonnet

Un 'order' est une association entre un 'transferts' et un 'lendings' pour l'insérer il est nécessaire que les deux soit créer auparavant.

Pour ce faire nous avons créé une fonction retournant l'id du prochain transfert disponible.

- La fonction '`findTheClosestTransfer`' est utilisée pour trouver le transfert le plus proche entre deux bibliothèques.
- S'il y a un transfert entre les deux bibliothèques dans la même semaine, elle retournera l'identifiant du transfert.
- S'il n'y a pas de transfert entre les deux bibliothèques la même semaine, la fonction créera un nouveau transfert et retournera l'identifiant du nouveau transfert.
- La fonction prend deux paramètres : l'identifiant de la bibliothèque à partir de laquelle le transfert est envoyé et l'identifiant de la bibliothèque vers laquelle le transfert est envoyé.
- La fonction renvoie l'identifiant du transfert.


```

1 CREATE OR REPLACE FUNCTION findTheClosestTransfer(id_from INTEGER, id_to INTEGER)
2 RETURNS INTEGER AS $findTheClosestTransfer$
3 DECLARE
4     v_nbTransfert INTEGER;
5     v_idTransfert INTEGER;
6     v_maxId INTEGER;
7     v_idEmployee INTEGER;
8     v_newDate DATE;
9 BEGIN
10    -- Check if there is a transfer between the two libraries in the same week
11    SELECT COUNT(*), COALESCE(MIN(id_transfert), 999)
12    INTO v_nbTransfert, v_idTransfert
13    FROM Transferts
14    WHERE date_sent <= (SELECT DATE_TRUNC('week', CURRENT_DATE)::DATE + 6) and id_library_from = id_from and id_library_to = id_to ;
15
16    -- If there is a transfer between the two libraries in the same week, return the id of the transfer
17    IF v_nbTransfert >= 1 THEN
18        RETURN v_idTransfert;
19    ELSE
20        -- Find the next id
21        SELECT COALESCE(MAX(id_transfert), 0) + 1 INTO v_maxId FROM Transferts;
22        -- Find an employee
23        SELECT MIN(id_employee) INTO v_idEmployee FROM employees where id_library = id_from;
24        -- Find the next date
25        SELECT adjust_transfer_date((SELECT DATE_TRUNC('week', CURRENT_DATE)::DATE + 6), id_from) INTO v_newDate;
26        -- Insert the new transfer use an other function explain below
27        INSERT INTO Transferts VALUES (v_maxId, v_newDate, v_newDate + INTERVAL '1 week', v_idEmployee, id_from, id_to);
28        -- Return the id of the new transfer
29        RETURN v_maxId;
30    END IF;
31 END;
32 $findTheClosestTransfer$ LANGUAGE plpgsql;

```


<https://github.com/bdr-si3-2425/Collections-interconnect-es-dans-une-cha-ne-de-biblioth-ques/blob/main/functions/findTheClosestTransfer.sql>

La fonction 'adjust_transfer_date' permet d'avoir la prochaine date disponible, c'est à dire un jour où il n'y a pas d'évènement dans la bibliothèque de départ.

Si la date envoyée n'est pas en conflit, elle renverra la même date.

Si la date envoyée est en conflit, elle renverra le jour suivant qui n'est pas en conflit.

Elle prend en argument une date et un id de Library et renvoie la prochaine date sans conflit à partir du prochain dimanche.



```

1 CREATE OR REPLACE FUNCTION adjust_transfer_date(date_sent DATE, id_library_from INTEGER)
2 RETURNS DATE AS $adjust_transfer_date$
3 DECLARE
4     v_new_date DATE := date_sent;
5     v_p_id_library INTEGER := id_library_from;
6     v_conflict_count INTEGER;
7 BEGIN
8     -- While there is a conflict, keep adding one day
9     LOOP
10         SELECT COUNT(*)
11         INTO v_conflict_count
12         FROM Events
13         WHERE event_date = v_new_date AND id_library = v_p_id_library;
14
15         -- If there is no conflict, return the date
16         IF v_conflict_count = 0 THEN
17             return v_new_date;
18         END IF;
19
20         -- If there is a conflict, add one day and check again
21         v_new_date := v_new_date + INTERVAL '1 day';
22     END LOOP;
23 END;
24 $adjust_transfer_date$ LANGUAGE plpgsql;

```

https://github.com/bdr-si3-2425/Collections-interconnect-es-dans-une-cha-ne-de-biblioth-ques/blob/main/functions/adjust_transfer_date.sql

Rendu d'un 'lending' et récupération des livres associés au 'holdings' de la bibliothèque ayant effectué le prêt.

Fait par : Marius Barbaud

Un 'lendings' de n livres est associé à n 'is_lended' contenant chacun l'id du lending et l'isbn du livre prêté.

Lorsqu'un ou plusieurs lendings sont rendus, un trigger a pour objectifs de placer les livres rendus dans la bibliothèque ayant effectué le prêt.

- Lorsqu'un lendings est rendu, le trigger 'Lending_returned_trigger' s'active et appelle la fonction 'Return_lendings_books'.
- La fonction 'Return_lendings_books' rajoute aux 'holdings' de la bibliothèque chaque livre du prêt.
- Cette fonction appelle la fonction 'add_book' pour l'ajout des livres

```
1  -- This trigger triggers when the 'returned' column of a lending switches from false to true --
2  -- This trigger calls the return_lendings_books function --
3  -- that function returns the books that were lent to the lending library --
4  CREATE TRIGGER Lending_returned_trigger
5  |   AFTER UPDATE
6  |   ON LENDINGS FOR EACH ROW
7  |   WHEN (OLD.returned = FALSE AND NEW.returned = TRUE)
8  |   EXECUTE FUNCTION return_lendings_books();
9
```

https://github.com/bdr-si3-2425/Collections-interconnect-es-dans-une-cha-ne-de-biblioth-ques/blob/main/triggers/Lending_returned_trigger.sql

```
1  --Function called by the trigger Lending_returned_trigger--
2  --updates the quantity of books in holdings when a lending is returned--
3  --Calls add_book function to increase the quantity of books in holdings--
4
5  CREATE OR REPLACE FUNCTION return_lendings_books()
6  RETURNS TRIGGER AS $$
7  DECLARE rec RECORD;
8  BEGIN
9      IF OLD.returned = FALSE AND NEW.returned = TRUE THEN
10         -- We record the lines from is_lended table that corresponds to the lending that is returned
11         FOR rec IN
12             SELECT * FROM is_lended
13             WHERE id_lending = NEW.id_lending
14         LOOP
15             --we add a book to the holdings of the library for every lent book
16             PERFORM add_book(rec.isbn, NEW.id_library, 1);
17         END LOOP;
18     END IF;
19     RETURN NEW;
20 END;
21 $$ LANGUAGE plpgsql;
```

https://github.com/bdr-si3-2425/Collections-interconnect-es-dans-une-cha-ne-de-biblioth-ques/blob/main/triggers/Return_lendings_books.sql

- La fonction 'add_book' est une fonction qui modifie la table holdings d'une bibliothèque donnée afin d'ajouter ou retirer les livres nécessaires.
- Elle prend en paramètre un 'isbn', un 'id_library' et une quantité, cette quantité peut être négative pour retirer des livres.
- Cette fonction n'a pas de valeur de retour.

```

1  -- increases the quantity of a book a library holds --
2  -- number can be negative ---
3  CREATE OR REPLACE FUNCTION add_book(p_isbn VARCHAR(42), p_id_library INTEGER, p_number INTEGER)
4  RETURNS VOID as $$
5  BEGIN
6      UPDATE holdings
7      SET quantity = quantity + p_number
8      WHERE id_library = p_id_library
9      AND isbn = p_isbn;
10
11  END;
12  $$ LANGUAGE plpgsql;

```

https://github.com/bdr-si3-2425/Collections-interconnect-es-dans-une-cha-ne-de-biblioth-ques/blob/main/functions/add_book.sql

Rendu d'un 'lending' et récupération des livres associés au 'holdings' de la bibliothèque ayant effectué le prêt.

Fait par : Axel Derail—Santer

```

1 CREATE OR REPLACE FUNCTION remove_or_order_lended_books()
2 RETURNS TRIGGER AS $$
3 DECLARE rec_is_lended RECORD;
4 DECLARE rec_libraries RECORD;
5 DECLARE books_to_order INTEGER;
6 DECLARE id_transfert INTEGER;
7 DECLARE id_order INTEGER;
8 BEGIN
9     WITH RankedHoldings AS (
10         SELECT isbn, id_library, quantity,
11             ROW_NUMBER() OVER (PARTITION BY isbn ORDER BY quantity DESC, id_library ASC) AS rank
12         FROM holdings
13         WHERE isbn NOT IN (
14             SELECT isbn FROM holdings WHERE id_library = NEW.id_library
15         )
16         AND isbn IN (
17             SELECT isbn FROM is_lended WHERE id_lending = NEW.id_lending
18         )
19     )
20
21     FOR rec_is_lended IN
22     SELECT * FROM is_lended
23     WHERE id_lending = NEW.id_lending
24     LOOP
25         IF NOT EXISTS (SELECT isbn FROM holdings
26             WHERE id_library = NEW.id_library
27             AND isbn = rec_is_lended.isbn
28         )
29             WHERE isbn = rec_is_lended.isbn AND id_library = NEW.id_library)
30         FOR
31         THEN
32             books_to_order := books_to_order + 1;
33         END LOOP;
34     ELSE
35         PERFORM add_book(rec_is_lended.isbn, NEW.id_library, -1);
36     END IF;
37 END LOOP;
38
39 IF books_to_order > 0
40 THEN
41     id_order := id_orders_seq.nextval();
42     INSERT INTO ORDER
43     VALUES(id_order, NEW.id_lending);
44
45     FOR rec_libraries IN
46     SELECT DISTINCT id_library
47     FROM RankedHoldings
48     WHERE rank = 1
49     LOOP
50         id_transfert := create_transfert(id_lending, id_library);
51         INSERT INTO ordered
52         VALUES(id_order, id_transfert);
53     END LOOP;
54 END IF;
55 RETURN NEW;
56 END;
57 $$ LANGUAGE plpgsql;

```

Cette fonction est appelée par un trigger lorsque qu'un prêt est enregistré (i.e: quand il y a un insert sur la table lendings)

RankedHoldings classe les isbn par id_library et quantity. C'est à dire qu'on a dans cette vue temporaire, chaque livre et la bibliothèque qui en compte le plus d'exemplaires. Il ne contient que les livres présent dans le prêt et absent de la bibliothèque ou a été effectué le prêt.


On boucle ensuite sur les nouveaux livres prêtés, si le livre est présent dans la bibliothèque ou a été fait le prêt, on le retire de celle-ci. Sinon on incrémente un compteur de livres a commander.

S'il y a des livres à commander, on va créer un transfert pour chaque bibliothèque concernée, qu'on va associer à une commande.

4. Questions principales :

Question 1 :

Pour qu'un ouvrage soit disponible dans la base il faut qu'il soit dans la table 'books' et aussi dans la table 'holdings' avec une quantité > 0.




```

1  -- Quels ouvrages sont disponibles dans le réseau et peuvent être transférés à une bibliothèque
2  -- donnée pour un abonné spécifique ?
3
4  -- Si on insere un nouveau livre mais sans exemplaire (Holdings) il ne sera pas disponible
5  insert into books values (15, 'Lords of rings', '1954-01-26', 1,1);
6
7  -- Le livre est bien present dans la table books
8  select * from books;
9  -- Mais pas dans holdings
10 select * from holdings;
11
12 -- Mais pas disponible a être transféré
13 SELECT DISTINCT b.*
14 FROM holdings h
15 JOIN books b ON b.isbn = h.isbn
16 WHERE h.quantity > 0;
17
18 --Pour qu'il soit disponible il faut le mettre un exemplaire dans une bibliothèque du réseau
19 insert into holdings values (15, 1, 1, 'A', 'Fantasy', 1);
20
21 -- Il est maintenant disponible dans tout le réseau;
22 SELECT DISTINCT b.*
23 FROM holdings h
24 JOIN books b ON b.isbn = h.isbn
25 WHERE h.quantity > 0;
26

```

<https://github.com/bdr-si3-2425/Collections-interconnect-es-dans-une-cha-ne-de-biblioth-ques/blob/main/questions/q1.sql>

Question 2 :



```
1  -- Comment gérer l'intégration d'une nouvelle bibliothèque au
2  -- réseau et l'attribution de ses ressources ?
3
4  -- Pour ajouter une nouvelle bibliothèque
5
6  SELECT * FROM libraries;
7  SELECT * FROM addresses;
8  SELECT * FROM holdings;
9  SELECT * FROM books;
10 SELECT * from employees;
11
12 -- Insere de l'adresse
13 INSERT INTO addresses VALUES (4, 'PACA', ' Rte des Colles', 930, 'Biot', 06410);
14
15 -- Insere d'une nouvelle bibliothèque
16 INSERT INTO libraries VALUES (4, 'PNS Library', 4)
17
18 -- Insere d'un nouveau employee
19 INSERT INTO employees VALUES (3, 'Matthieu', 'Griffonnet', 'Yes?', 4);
20
21 -- Insere de nouveau exemplaire pour la bibliothèque
22 INSERT INTO holdings VALUES (1, 4, 1, 1, 'Fantasy', 01);
23 INSERT INTO holdings VALUES (2, 4, 5, 1, 'Fantasy', 01);
24 INSERT INTO holdings VALUES (3, 4, 2, 1, 'Fantasy', 01);
25 INSERT INTO holdings VALUES (4, 4, 3, 1, 'Fantasy', 01);
26
27 -- Voici les livres présent dans la nouvelle bibliothèque
28 SELECT * FROM holdings where id_library = 4;
```

<https://github.com/bdr-si3-2425/Collections-interconnect-es-dans-une-cha-ne-de-biblioth-ques/blob/main/questions/q2.sql>

Question 3 :

```
1  -- 3. Quels ouvrages sont fréquemment transférés entre bibliothèques et quels délais sont associés
2  SELECT * FROM transferts;
3  SELECT * FROM orders;
4  SELECT * FROM lendings;
5  SELECT * FROM is_lended;
6
7  -- La requête compte combien de fois chaque ISBN apparaît dans les commandes
8  -- Ensuite, elle calcule le pourcentage de présence de
9  -- chaque ISBN en divisant ce nombre par le total des commandes
10
11 -- En partant du jeu de donnée de base
12 -- Un premier est effectué avec deux livres 4 et 5, ils ont donc 100% d'apparence dans la base
13
14 SELECT
15     i.isbn,
16     -- Compte le nombre de fois que chaque ISBN apparaît dans les commandes
17     COUNT(o.id_order) AS total_orders,
18     -- Calcule le pourcentage de présence de chaque ISBN en pourcentage
19     ROUND((COUNT(o.id_order) * 100.0 / (SELECT COUNT(*) FROM orders)), 2) AS percentage_presence
20 FROM
21     orders o
22 JOIN
23     is_lended i ON i.id_lending = o.id_lending
24 JOIN
25     transferts t ON t.id_transfert = o.id_transfert
26 GROUP BY
27     i.isbn
28 ORDER BY
29     percentage_presence DESC;
30
31 -- On crée un nouvel emprunt
32 INSERT into lendings values (3,1,'subscriber.nb1@gmail.com', '2025-02-02', '2025-02-09',false, null, null);
33 -- Liasont entre le lending et le livre 4
34 INSERT into is_lended values (3,4);
35 -- Liaison en 1 lending et un transfert
36 INSERT into orders values (2,3,1,2,1);
37 -- Maintenant le livre 4 est pris a 100% des fois et le livre 5 50% car present dans 1/2 lendings
```

<https://github.com/bdr-si3-2425/Collections-interconnect-es-dans-une-cha-ne-de-biblioth-ques/blob/main/questions/q3.sql>

Question 4 :

```
1  --4. Quels abonnés ne respectent pas les délais de prêt --
2  --et quelle est leur fréquence d'infraction ? --
3
4  --Returns the number of violation done by a subscriber in the last year--
5  CREATE OR REPLACE FUNCTION violation_frequence_calculator(p_email VARCHAR(42))
6  RETURNS INTEGER AS $$
7  DECLARE frenquency INTEGER;
8  BEGIN
9      SELECT COUNT(delays) INTO frenquency
10     FROM lendings
11     WHERE email = p_email
12     AND delays > 0
13     AND CURRENT_DATE - end_date <= 365;
14     RETURN frenquency;
15 END;
16 $$ LANGUAGE plpgsql;
17
18 --On récupère tous les abonnés ayant des violations --
19 --et on utilise violation_frequence_calculator pour obtenir la fréquence de leur violations--
20
21 --On ajoute quelques données en plus...--
22 INSERT INTO SUBSCRIBERS VALUES ('Abonne.Parfait@yahooMail.com', 'Abonne', 'Parfait', '2002-01-01');
23 INSERT INTO SUBSCRIBERS VALUES ('Pire.Abonne@etu.univ-cotedazur.fr', 'Pire', 'Abonne', '2004-12-12');
24 SELECT * FROM SUBSCRIBERS;
25 DELETE FROM LENDINGS WHERE id_lending > 2;
26 INSERT INTO LENDINGS VALUES (3, 1, 'Abonne.Parfait@yahooMail.com', '2024-01-02', '2024-02-02', true, NULL, '2024-01-03');
27 INSERT INTO LENDINGS VALUES (4, 1, 'Pire.Abonne@etu.univ-cotedazur.fr', '2024-01-01', '2024-02-01');
28 INSERT INTO LENDINGS VALUES (5, 2, 'Pire.Abonne@etu.univ-cotedazur.fr', '2024-10-10', '2024-11-10');
29 INSERT INTO LENDINGS VALUES (6, 2, 'Pire.Abonne@etu.univ-cotedazur.fr', '2025-01-01', '2025-02-01');
30 INSERT INTO LENDINGS VALUES (7, 2, 'Jean.Jacques@gmail.com');
31 SELECT update_lending_delays();
32 SELECT * FROM LENDINGS ORDER BY id_lending;
33
34 --Cette requête renvoie une table contenant tous les abonnés ayant fait une violation, et donne leurs fréquences d'infraction--
35 SELECT S.first_name, S.last_name, violation_frequence_calculator(S.email) as "violation frequence"
36 FROM LENDINGS L JOIN SUBSCRIBERS S ON L.email = S.email
37 WHERE L.delays > 0
38 GROUP BY S.email
39 ORDER BY "violation frequence" DESC;
40 --L'abonné Pire Abonne a fait 2 infractions dans les 365 derniers jours--
41 --Et l'abonné Jean Jacques en a commit 1--
```

<https://github.com/bdr-si3-2425/Collections-interconnect-es-dans-une-cha-ne-de-biblioth-ques/blob/main/questions/q4.sql>

Question 5 :

Quels événements sont programmés dans une bibliothèque et quels abonnés ont déjà participé à des événements similaires ?

Après discussion avec le client, il nous a confirmé qu'il n'existe pas d'historique pour les abonnés lors d'un événement. Par conséquent, cette question ne peut être justifiée.

Question 6 :

```
1  --6. Quels ouvrages ou collections sont les plus
2  --populaires selon les abonnés d'une région spécifique ?
3
4  -- Function: popular_books_by_region(region_name VARCHAR)
5  -- This function returns the most popular books in a given region. It takes a region name as an argument
6  -- returns a table with the 5 most popular books and the total number of loans.
7  CREATE OR REPLACE FUNCTION popular_books_by_region(region_name VARCHAR)
8  RETURNS TABLE(
9      title VARCHAR,
10     total_loans INTEGER
11 ) AS $$
12 BEGIN
13     RETURN QUERY
14     SELECT
15         b.title,
16         COUNT(*)::INTEGER AS total_loans
17     FROM
18         LENDINGS l
19     -- Jointure entre les différentes tables
20     JOIN IS_LENDED il ON l.id_lending = il.id_lending
21     JOIN BOOKS b ON il.isbn = b.isbn
22     JOIN LIBRARIES lib ON l.id_library = lib.id_library
23     JOIN ADDRESSES addr ON lib.id_address = addr.id_address
24     WHERE
25         -- Conditions sur la région
26         addr.region = region_name
27     GROUP BY
28         b.title
29     ORDER BY
30         total_loans DESC
31     LIMIT 5;
32 END;
33 $$ LANGUAGE plpgsql;
34
35
36 SELECT * FROM LENDINGS;
37 SELECT * FROM is_lended;
38 SELECT * FROM BOOKS;
39
40 -- Creation de deux nouveau emprunt
41 insert into lendings values (4, 1, 'subscriber.nb1@gmail.com', '2025-04-01', '2025-04-08', false, null, null);
42 insert into lendings values (5, 1, 'subscriber.nb1@gmail.com', '2025-04-02', '2025-04-09', false, null, null);
43 -- Rajout des livres associé a l'emprunt
44 insert into is_lended (id_lending, isbn) values (4,1);
45 insert into is_lended (id_lending, isbn) values (4,2);
46 insert into is_lended (id_lending, isbn) values (4,3);
47 insert into is_lended (id_lending, isbn) values (5,1);
48
49
50 SELECT popular_books_by_region('PACA');
51 -- Le livre 1 apparaît 2 fois donc il est premier et le livre 2 et 3 apparaissent 1 fois donc 2 et 3ème.
```

<https://github.com/bdr-si3-2425/Collections-interconnect-es-dans-une-cha-ne-de-biblioth-ques/blob/main/questions/q6.sql>

5. La description du jeu de données utilisé pour tester les différentes requêtes

La description du jeu de données utilisé pour tester les différentes requêtes sous des conditions typiques. La taille de ce jeu de données peut être faible mais on justifiera soigneusement le choix de ces données pour la complétude des tests (= validation)

Jeu de donnée par défaut :

Ce jeu de donnée initialise la plupart des données initiale appelées dans les tests : auteur, éditeur, adresse, bibliothèque, collection, livre, abonné, employé et holdings. Ensuite, chaque test insert des données supplémentaires pour effectuer les différents tests.

<https://github.com/bdr-si3-2425/Collections-interconnect-es-dans-une-cha-ne-de-biblioth-ques/blob/main/Donnees.sql>

Test du jeu de données pour les transferts :

```
1  -- Date du test 3 fev 2025
2  select * from lendings;
3
4  --Creation d'un nouveau lending pour le subscriber 3
5  insert into lendings values (99, 3, 'subscriber.nb1@gmail.com', '2025-02-03', '2025-02-10', false ,null ,null);
6  delete from lendings where id_lending = 99;
7
8  -- On ajoute un événement le prochain dimanche et lundi
9  INSERT INTO events values (7,'Conference', '2025-02-09', 2);
10 INSERT INTO events values (8,'Conference', '2025-02-10', 2);
11 --delete from events where id_event = 7;
12 --delete from events where id_event = 8;
13 SELECT * FROM events ORDER BY event_date DESC;
14
15 -- La prochaine date disponible sera le 2025-02-11
16 -- le select nous retourne creer un evenement ce jour la et nous retourne l'id du transfert
17 -- entre la bibliothèque 3 et 2 avec la date la plus proche apres le prochain dimanche
18 SELECT findtheclosesttransfer(2,3);
19 SELECT * FROM transferts where id_transfert = 3;
20
21 -- Transferts(id_transfert, date_sent, date_received, id_employee, id_library_from, id_library_to)
22 -- Obtient bien le transferts 3,"2025-02-11","2025-02-18",2,2,3
23 -- Qui commence bien le 11 apres les deux evenements;
24
25 insert into orders values (3, 99, 3, 2, 3);
26 SELECT * FROM orders;
```

<https://github.com/bdr-si3-2425/Collections-interconnect-es-dans-une-cha-ne-de-biblioth-ques/blob/main/tests/testTriggerOrder.sql>

Test de la fonction popularité :

```
1  -- En partant du jeu de donnée de base
2  -- Les deux premiers livre empruntés sont les livres 4 et 5
3  -- On a donc comme livre les plus populaires les livres 4 et 5
4  SELECT * FROM popular_books_by_region('PACA');
5
6  -- On crée un nouvel emprunt
7  INSERT into lendings values (3,1,'subscriber.nb1@gmail.com', '2025-02-02', '2025-02-09',false, null, null);
8  -- Liasont entre le lending et le livre 4
9  INSERT into is_lended values (3,4);
10
11 -- Le livre 4 est maintenant premier avec 2 emprunts et le livre 5 est deuxième avec 1 emprunt
12 SELECT * FROM popular_books_by_region('PACA');
13
```

<https://github.com/bdr-si3-2425/Collections-interconnect-es-dans-une-cha-ne-de-biblioth-ques/blob/main/tests/testPopularity.sql>

Test trigger 'event' update :

```
1  -- En partant du principe que le transfert entre la bibliothèque le 2 et 3 du 11-fevrier est dans la base
2  insert into transferts(3,'2025-02-11','2025-02-18',2,2,3);
3
4  SELECT * FROM TRANSFERTS;
5  SELECT * FROM EVENTS;
6
7  -- Insertion d'un nouvel exposition
8  insert into EXPOSITIONS values (9, 'Exposition', '2025-02-11', 3, 1);
9  DELETE FROM EXPOSITIONS WHERE id_event = 9;
10
11 -- La nouvelle date est le 2025-02-12
12 SELECT * FROM transferts where id_transfert = 3;
```

<https://github.com/bdr-si3-2425/Collections-interconnect-es-dans-une-cha-ne-de-biblioth-ques/blob/main/tests/testTriggerEvent.sql>

Test du trigger du rendu de livre :

```

1  --test for the lending_returned_trigger trigger--
2  --this trigger calls Return_lendings_books function when triggered--
3  --and returns the lendended books to the library of the lending--
4
5  --we create a new lending and add books to it--
6  INSERT INTO LENDINGS VALUES (3, 1, 'subscriber.nb1@gmail.com');
7  INSERT INTO IS_LENDED VALUES (3, '2');
8  INSERT INTO IS_LENDED VALUES (3, '3');
9
10 SELECT * FROM LENDINGS;
11
12 SELECT * FROM IS_LENDED;
13
14 --There is by default 4 copies of the book with isbn 2 and 3 with isbn 3 --
15 SELECT * FROM HOLDINGS WHERE id_library = 1 ORDER BY isbn;
16
17 --We return the lending--
18 UPDATE LENDINGS SET returned = TRUE WHERE id_lending = 3;
19
20 --The trigger should have update the holdings table--
21 --now there should be 5 copies of isbn 2 and 4 of isbn 3--
22 SELECT * FROM HOLDINGS WHERE id_library = 1 ORDER BY isbn;
23
24 --reset data--
25 UPDATE HOLDINGS SET quantity = 4 WHERE id_library = 1 and isbn = '2';
26 UPDATE HOLDINGS SET quantity = 3 WHERE id_library = 1 and isbn = '3';
27 DELETE FROM IS_LENDED WHERE id_lending = 3;
28 DELETE FROM LENDINGS WHERE id_lending = 3;

```

https://github.com/bdr-si3-2425/Collections-interconnect-es-dans-une-cha-ne-de-biblioth-ques/blob/main/tests/Lending_returned_trigger_test.sql

Test de la fonction qui met à jour le retard des prêts non retournés :

```
1  --Test for the update_lending_delays function--
2  --this function changes the number of days since a book is due --
3  --if the lending isn't returned on time--
4
5  --we create a new lending that is already due--
6  INSERT INTO LENDINGS VALUES (3, 2, 'subscriber.nb2@gmail.com', '2024-12-01', '2025-01-01');
7  --and a new lending that began today (by default begin date = CURRENT_DATE)
8  INSERT INTO LENDINGS VALUES (4, 1, 'subscriber.nb1@gmail.com');
9  --we use the function--
10 SELECT update_lending_delays();
11
12 --now the 'delays' column has been updated --
13 --and the number of days since end_date is written--
14 --on the lending number 3--
15 SELECT * FROM LENDINGS;
16
17 --If we use it again, nothing changes--
18 SELECT update_lending_delays();
19 SELECT * FROM LENDINGS;
20
21 --reset data--
22 DELETE FROM LENDINGS WHERE id_lending = 3;
23 DELETE FROM LENDINGS WHERE id_lending = 4;
```

https://github.com/bdr-si3-2425/Collections-interconnect-es-dans-une-cha-ne-de-biblioth-ques/blob/main/tests/Update_Lending_Delays_Test.sql

Test de la fonction calculant le nombre de violation du rendu de prêt de la dernière année d'un abonné :

```
1  --Test for the violation_frequence_calculator function--
2  --This function takes a subscriber's email in parameter--
3  --and returns the number of violations he's had in the last 365 days--
4
5  --we create a new subscriber and give him some lendings--
6  INSERT INTO SUBSCRIBERS VALUES ('Violation.Subscriber@gmail.com', 'Test', 'Name', '2000-01-01');
7  INSERT INTO LENDINGS VALUES (3, 2, 'Violation.Subscriber@gmail.com', '2024-12-01', '2025-01-15');
8  INSERT INTO LENDINGS VALUES (4, 2, 'Violation.Subscriber@gmail.com');
9  SELECT update_lending_delays();
10
11 --The subscriber now has 1 violation--
12 SELECT violation_frequence_calculator('Violation.Subscriber@gmail.com');
13
14 --we add 2 more violations, but one is dated of more than 1 year--
15 INSERT INTO LENDINGS VALUES (5, 2, 'Violation.Subscriber@gmail.com', '2020-01-01', '2020-02-01');
16 INSERT INTO LENDINGS VALUES (6, 2, 'Violation.Subscriber@gmail.com', '2025-01-01', '2025-02-01');
17 SELECT update_lending_delays();
18
19 --therefore only 2 violations are counted for this subscriber for the last 365 days--
20 SELECT violation_frequence_calculator('Violation.Subscriber@gmail.com');
21
22 --reset data--
23 DELETE FROM LENDINGS WHERE id_lending > 2;
24 DELETE FROM SUBSCRIBERS WHERE email = 'Violation.Subscriber@gmail.com';
```

https://github.com/bdr-si3-2425/Collections-interconnect-es-dans-une-cha-ne-de-biblioth-ques/blob/main/tests/Violation_frequence_calculator_test.sql

6. Une analyse critique du déroulement du projet

Au début du projet, nous avons réparti les responsabilités entre nous pour travailler sur différentes parties de la base de données :

- Axel : Mise en place des fonctionnalités permettant la réservation et le transfert d'ouvrages entre bibliothèques.
- Marius : Suivi des ouvrages disponibles dans chaque bibliothèque, avec l'indication précise de leur emplacement.
- Matthieu : Gestion des abonnés et de leur historique de prêt.

Une fois ces tâches individuelles réalisées, nous avons regroupé nos résultats pour concevoir ensemble le schéma entité-association. Nous avons consacré beaucoup de temps à l'élaboration de ce schéma afin d'éviter de devoir y revenir par la suite.

Une fois le premier schéma validé, nous avons réparti les tâches pour implémenter chacun une partie des tables de la base et rédiger les requêtes SQL correspondantes.

Pour ce faire chaque requête, trigger, fonction, vue et rôle ont été mis dans des issues [GitHub](#).

Axel :

- Contrôle d'accès :
 - Admin : a le contrôle de toute la base de données
 - Gérant (Director) : a le contrôle de la base de données en ce qui concerne sa bibliothèque
 - Employé (Employee) : a les mêmes pouvoirs que le gérant, sauf sur la table employée
 - Abonné (Subscriber) : peut voir et changer ses informations, peut voir ses prêts, violations, peut voir les stocks dans toutes les bibliothèques
- View : Subscriber
- Trigger lending : Quand un lending est fait, diminue le nombre de livre que possède la bibliothèque
- Trigger Lending : Quand un lending est demandé on regarde que le livre est bien dans la bibliothèque sinon un transfert automatique est demandé par la bibliothèque d'origine dans la bibliothèque qui possède le livre avec la plus grande quantité.
S'il y a une commande dans une autre bibliothèque, on ajoute une semaine à la date de rendu du prêt.

Marius :

- Fonction : Update lending delays : Met à jour les delays pour les lendings non rendu en retard.

- Fonction : Extend lending period : Augmente un lending de 2 semaine s'il n'est pas rendu, pas en retard et qu'il n'a pas déjà été étendu.
- Fonction : Calcul fréquence d'infraction : On peut obtenir le nombre d'infraction commis par un abonné lors des 365 derniers jours.
- Trigger : returned lending : Quand un lending est rendu, les livres empruntés sont redonnés à la bibliothèque qui a créé le lending.

Matthieu :

- Fonction : Calcul de la popularité des ouvrages
- Trigger Event : Quand un événement est insérer ou mis à jour, on regarde s'il un transfert à déjà lieu le jour même. S'il est on décale le transfert de un jour.
- Fonction Transfert : Quand un transfert un insérer ou mis à jour, on ne regarde qu'aucun "event" a déjà lieu le même jour. S'il y en a un la date est décalée au lendemain

Au cours de ce projet, nous aurions gagné à nous réunir plus fréquemment en dehors des cours afin de mieux coordonner nos efforts et renforcer notre collaboration. De plus, ajouter des deadlines aux issues aurait permis de structurer davantage notre travail et de progresser de manière plus efficace sur les différentes tâches. Ces ajustements pourraient être de précieux enseignements pour nos futurs projets.

7. Vidéo

La vidéo est disponible sur ce lien : [video.mp4](#)

Autre lien : [youtube](#)