

TRAVAIL PRATIQUE

Modélisation de l'Écosystème de la Recherche

PostgreSQL · Gestion de Bases de Données et Data Warehouses

TP

01

BASE DE
DONNÉES
RELATIONNELLE

Université de Corse - Pasquale PAOLI

M1 DFS-DENG

**JAMHOUR Yousra , KARKACHE EL Mehdi et
WAHAB Mohamed**

Enseignant

Pr VITTORI Evelyne

Github

https://github.com/jamhouryousra/tp1_database

2025

2026

Table des matières

INTRODUCTION	4
Contexte du Projet.....	4
Objectifs du TP1	4
Méthodologie.....	4
PARTIE 1 : CONCEPTION DE LA BASE DE DONNÉES.....	6
1.1 Schéma Relationnel.....	6
1.2 Choix de Conception et Justifications	8
1.2.1 Gestion des Relations Many-to-Many	8
1.2.2 Ajouts Stratégiques dans les Tables	9
1.3 Contraintes d'Intégrité.....	10
1.3.1 Contraintes Déclaratives (niveau SQL)	10
1.4 Utilisation de l'IA Générative	10
1.4.1 Méthodologie d'Utilisation	10
PARTIE 2 : PEUPLEMENT DE LA BASE.....	12
2.1 Méthodologie Python Faker	12
2.2 Gestion de l'Intégrité Référentielle.....	12
2.3 Gestion des Capacités de Projets	12
2.4 Volumes de Données Générées	12
PARTIE 3 : GESTION DES UTILISATEURS	14
3.1 Rôles Crées.....	14
3.2 Vues Métier (5 vues créées)	14
3.2.1 VUE_PROJETS_CHERCHEURS : Projets avec participants et laboratoire.....	14
3.2.2 VUE_PUBLICATIONS_PROJET : Publications par projet.....	14
3.2.3 VUE_DATASETS_CONFORMITE : Datasets conformes/non conformes	15
3.2.4 VUE_CONTRATS_FINANCEMENT : Synthèse contrats avec DMP	15
3.2.5 VUE_CHERCHEURS_ACTIVITE : Tableau de bord activité chercheurs	16
3.3 Attribution des Priviléges.....	17
PARTIE 4 : REQUÊTES ET OPTIMISATION	19
4.1 Requête R1 : Datasets par Projet et Année	19
4.1.1 Énoncé	19
4.1.2 Index Crées pour R1	19
4.1.3 VERSION 1 : VUE avec CTE.....	19
4.1.4 VERSION 2 : SELECT + HAVING	21
4.1.5 VERSION 3 : Vue avec HAVING	22
4.1.6 Tableau Comparatif R1	24

4.1.7	Analyse et Choix de la Meilleure Version.....	24
4.2	Requête R2 : Projets sans Chercheurs Peu Productifs.....	25
4.2.1	Énoncé	25
4.2.2	Index Crées pour R2	25
4.2.3	VERSION 1 : VUE avec NOT EXISTS.	26
4.2.4	VERSION 2 : SELECT + MIN/HAVING	27
4.2.5	VERSION 3 : Vue avec MIN/HAVING	29
4.2.6	Tableau Comparatif R2	32
4.2.7	Analyse et Choix de la Meilleure Version.....	32
4.3	Requête R3 : Laboratoires sans Datasets Non Conformes	32
4.3.1	Énoncé	32
4.3.2	Index Crées pour R3	33
4.3.3	VERSION 1 : VUE avec NOT EXISTS	33
4.3.4	VERSION 2 : SELECT avec EXCEPT.....	35
4.3.5	VERSION 3 : VUE avec LEFT JOIN + HAVING	36
4.3.6	VERSION 4 : SELECT avec LEFT JOIN et HAVING.....	38
4.3.7	Tableau Comparatif R3	40
4.3.8	Analyse et Choix de la Meilleure Version.....	40
PARTIE 5 : TRIGGERS ET PROCÉDURES STOCKÉES.....		42
5.1	Trigger 1 : Limite de Participants	42
5.2	Trigger 2 : Vérification DMP	42
5.3	Fonction 1 : nb_publications_projet(id, année)	43
5.4	Procédure 2 : maj_bilan_projet(année).....	45
5.5	Fonction 3 : fiche_projet(id).....	47
5.6	Procédure 4 : archiver_contrats_echus(date_seuil)	49
CONCLUSION		52

INTRODUCTION

Ce travail pratique s'inscrit dans le cadre du module "Gestion de Bases de Données et Data Warehouses".

Il nous permet de mettre en pratique les concepts théoriques de modélisation relationnelle à travers un cas d'usage concret : la gestion de l'écosystème de la recherche universitaire.

Contexte du Projet

La gestion des données de recherche constitue un enjeu majeur pour les établissements universitaires. Entre l'explosion du volume de données produites, les exigences croissantes en matière de traçabilité et de conformité (plans de gestion des données, principes FAIR), et la nécessité d'analyser et de valoriser les résultats de recherche, les institutions doivent aujourd'hui s'appuyer sur des systèmes d'information robustes et structurés.

Ce projet vise à concevoir et implémenter une base de données relationnelle complète modélisant l'écosystème de la recherche universitaire. Elle doit permettre de gérer :

- Les acteurs : institutions, laboratoires, chercheurs avec leurs affiliations et statuts
- Les activités : projets structurants, participations, contrats de financement
- Les productions : publications scientifiques et jeux de données avec leurs métadonnées
- La conformité : plans de gestion des données (DMP), licences, conditions d'accès

L'objectif est de créer un référentiel centralisé permettant aux administrateurs de suivre les projets et les financements, aux data managers de contrôler la conformité des dépôts de données, et aux chercheurs de déclarer et valoriser leurs activités de recherche.

Objectifs du TP1

Ce premier travail pratique poursuit plusieurs objectifs pédagogiques :

- 1. Maîtriser la conception d'une base relationnelle** : Élaborer un schéma normalisé répondant à un cahier des charges complexe, définir les contraintes d'intégrité, justifier les choix de modélisation.
- 2. Automatiser le peuplement de la base** : Utiliser Python Faker pour générer des données fictives cohérentes tout en respectant l'intégrité référentielle.
- 3. Sécuriser l'accès aux données** : Mettre en œuvre une stratégie de gestion des droits via des rôles, vues et priviléges.
- 4. Optimiser les performances** : Analyser et comparer différentes versions de requêtes complexes à l'aide d'EXPLAIN ANALYZE et d'index stratégiques.
- 5. Programmer en PL/pgSQL** : Développer des triggers, fonctions et procédures stockées pour automatiser la gestion de la base.

Méthodologie

Ce travail a été réalisé en groupe de 3. Nous avons adopté une démarche itérative combinant :

- **Utilisation critique d'outils d'IA générative** (Claude AI) pour accélérer la phase de conception et de développement, tout en maintenant une posture critique sur les suggestions fournies.

- **Tests systématiques** : Chaque composant (trigger, procédure, requête) a été testé individuellement avec des jeux de données représentatifs.
- **Documentation continue** : Tous les scripts sont commentés et organisés de manière modulaire pour faciliter la maintenance et la compréhension.

PARTIE 1 : CONCEPTION DE LA BASE DE DONNÉES

1.1 Schéma Relationnel

Notre base de données modélise l'écosystème complet de la recherche universitaire avec **9 tables principales** organisées selon une architecture relationnelle.

Tables entités (7 tables) :

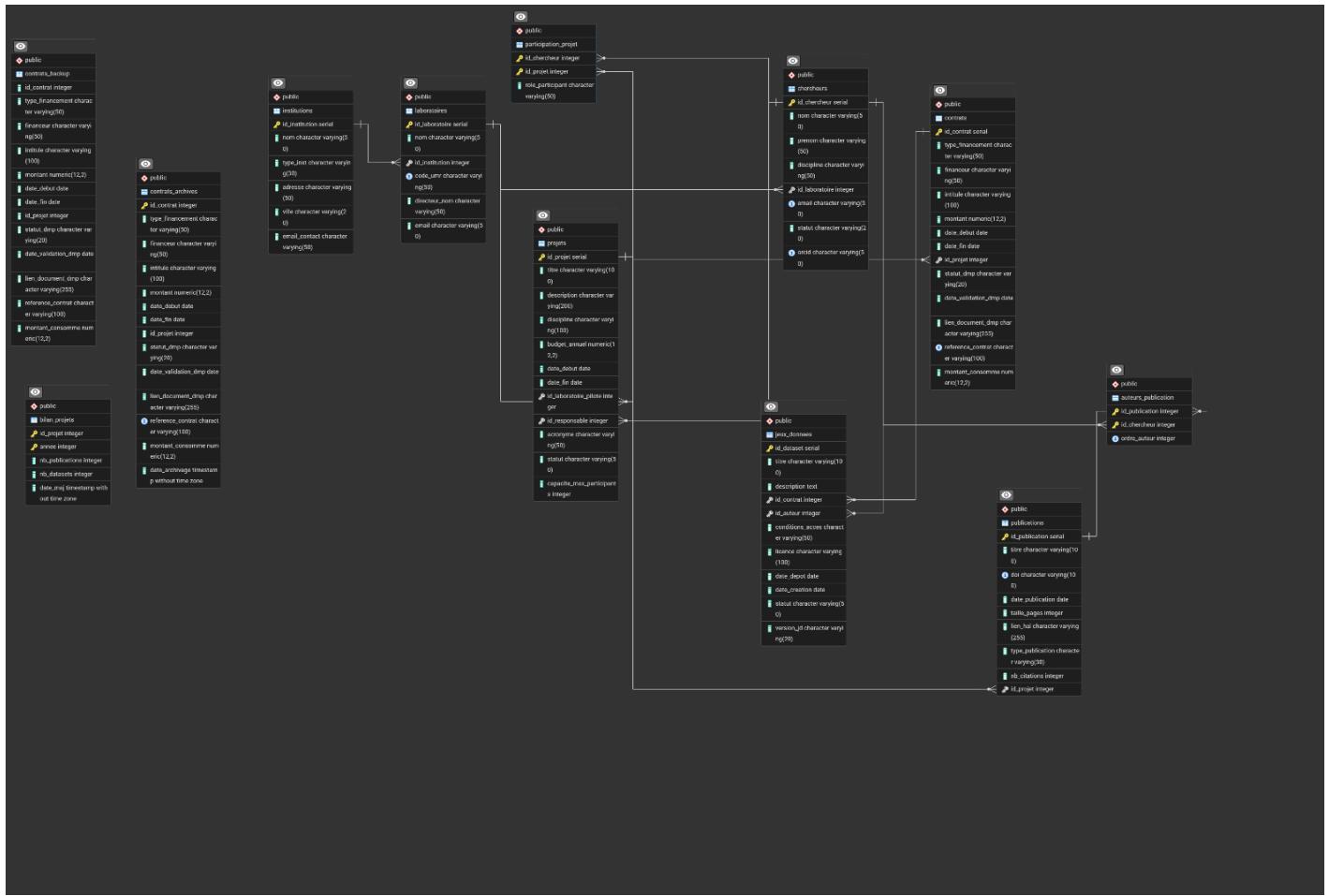
1. **INSTITUTIONS** : Organismes de recherche (Université de Corse, CNRS, partenaires privés)
2. **LABORATOIRES** : Unités Mixtes de Recherche (UMR) rattachées aux institutions
3. **CHERCHEURS** : Personnel de recherche affilié à un laboratoire unique
4. **PROJETS** : Projets structurants pilotés par un laboratoire avec un responsable scientifique
5. **CONTRATS** : Financements des projets (ANR, H2020, Région) avec Plan de Gestion des Données (DMP)
6. **PUBLICATIONS** : Métadonnées des publications scientifiques (titre, DOI, date, lien HAL)
7. **JEUX_DONNEES** : Métadonnées des datasets produits par la recherche (description, licence, conditions d'accès)

Tables associatives (2 tables) :

8. **PARTICIPATION_PROJET** : Table associative N-N entre CHERCHEURS et PROJETS, gérant la participation avec rôle (Responsable, Co-responsable, Participant, Collaborateur).
9. **AUTEURS_PUBLICATION** : Table associative N-N entre CHERCHEURS et PUBLICATIONS, gérant le co-auteurat avec ordre d'apparition des auteurs.

10. Note : Deux tables support (BILAN_PROJETS et CONTRATS_ARCHIVES) ont été créées en

Partie 5 dans le cadre des procédures stockées.



1.2 Choix de Conception et Justifications

1.2.1 Gestion des Relations Many-to-Many

Le cahier des charges impose une simplification : "Un chercheur ne participe qu'à un seul projet structurant". Cette contrainte aurait pu conduire à une simple relation 1-N entre CHERCHEURS et PROJETS. Cependant, pour rendre le modèle plus réaliste et évolutif, nous avons fait le choix stratégique d'implémenter une relation N-N via la table associative PARTICIPATION_PROJET.

➤ TABLE PARTICIPATION_PROJET (relation 1-N transformée en N-N)

Choix de conception :

- Création d'une table associative permettant de gérer plusieurs participations par chercheur
- Respect de la contrainte du cahier des charges (1 seule participation par chercheur dans les données générées)

Colonnes clés :

- id_chercheur, id_projet (clé primaire composite)
- role_participant : Contrainte CHECK (Responsable, Co-responsable, Participant, Collaborateur)

Avantage :

- Gestion fine des rôles dans les projets
- Historique des participations
- Évolutivité sans migration complexe

➤ TABLE AUTEURS_PUBLICATION (relation N-N native)

Une publication scientifique a typiquement plusieurs auteurs (co-auteurat).

Cette table associative était indispensable pour respecter la réalité de la recherche collaborative moderne.

Colonnes clés :

- id_chercheur, id_publication (clé primaire composite)
- ordre_auteur : Position de l'auteur (1 = premier auteur, crucial dans le milieu académique)

Justification : L'ordre des auteurs est une information essentielle en recherche (premier auteur = contributeur principal, dernier auteur = directeur de recherche).

Cette colonne permet de conserver cette hiérarchie.

1.2.2 Ajouts Stratégiques dans les Tables

Au-delà du cahier des charges minimal, chaque table comporte des -- AJOUTS STRATÉGIQUES -- enrichissant le modèle pour des cas d'usage réels et facilitant les analyses futures :

➤ **INSTITUTIONS :**

- ville
- email_contact

➤ **LABORATOIRES :**

- code_umr : Identifiant UMR (ex: UMR 6240)
- email
- directeur_nom

➤ **CHERCHEURS :**

- statut : (Titulaire, Contractuel, Doctorant, Post-doctorant) avec contrainte CHECK
- email
- ORCID (identifiant international chercheur)

➤ **PROJETS :**

- capacite_max_participants : Contrôlée par trigger (défaut 20)
- statut : ('En préparation', 'En cours', 'Terminé', 'Suspended')
- acronyme

➤ **PARTICIPATION_PROJET :**

- role_participant : (Responsable, Co-responsable, Participant, Collaborateur) avec CHECK Permet de distinguer le responsable scientifique des simples participants

➤ **CONTRATS :**

- reference_contrat : Numéro officiel du contrat
- montant_consomme : Suivi budgétaire

➤ **PUBLICATIONS :**

- type_publication : (Article, Conference, Poster, These)
- nb_citations : Indicateur d'impact

➤ **AUTEURS_PUBLICATION :**

- ordre_auteur : Position dans la liste (1, 2, 3...)

➤ **JEUX_DONNEES :**

- statut : (Brouillon, En préparation, Déposé, Archivé)
- date_creation : Traçabilité

- version_jd

1.3 Contraintes d'Intégrité

1.3.1 Contraintes Déclaratives (niveau SQL)

➤ Clés Primaires

- Toutes les tables disposent d'une clé primaire (id_xxx de type SERIAL)
- Tables associatives : clés primaires composées (id_chercheur, id_projet)
- Garantie d'unicité de chaque enregistrement

➤ Clés Étrangères avec ON DELETE CASCADE

- Toutes les relations sont contraintes par des clés étrangères
- ON DELETE CASCADE : Maintien automatique de la cohérence référentielle
- Exemple : Suppression d'un projet → suppression automatique des participations associées

➤ Contraintes CHECK

- role_participant IN ('Responsable', 'Co-responsable', 'Participant', 'Collaborateur')
- statut_dmp IN ('brouillon', 'soumis', 'validé')
- statut (chercheurs) IN ('Doctorant', 'Post-doc', 'MCF', 'PR', 'DR', 'CR', 'Ingénieur')
- Validation des domaines de valeurs avant insertion

➤ Contraintes NOT NULL

- Champs essentiels obligatoires : nom, prénom (chercheurs), titre (projets, publications), intitulé (contrats)
- Garantit la complétude des données critiques

1.4 Utilisation de l'IA Générationne

1.4.1 Méthodologie d'Utilisation

Nous avons utilisé **Claude AI** comme assistant lors de la conception et du développement.

L'approche adoptée repose sur trois phases :

Phase 1 : Génération du schéma initial

- Fourniture du cahier des charges à l'IA
- Génération d'une première proposition de schéma relationnel (9 tables, relations)
- Proposition de noms de colonnes et types de données

Phase 2 : Analyse critique et adaptation

- Vérification systématique de la cohérence des suggestions
- Correction des incohérences (noms de colonnes changeants, types inadaptés)
- Enrichissement avec les ajouts stratégiques

Phase 3 : Génération de code SQL

- Génération des scripts Python Faker

Avantages Constatés

- Gain de temps considérable sur le squelette de la base (génération initiale en quelques minutes)
- Suggestions pertinentes de noms de tables et colonnes respectant les conventions
- Bonne compréhension du domaine métier (recherche universitaire)
- Proposition de solutions alternatives pour les requêtes complexes

Limites et Vigilance Requise

- Incohérences entre les suggestions successives (ex: type_contrat → type_financement)
- Syntaxe SQL parfois générique (non spécifique à PostgreSQL, nécessite adaptation)
- Oublis de contraintes d'intégrité essentielles (CHECK, index)
- Nécessité de valider systématiquement chaque suggestion avant implémentation
- Risque de reproduire des erreurs si on ne comprend pas le code généré

Bilan et Recommandations

L'IA générative est un assistant puissant mais ne remplace pas l'expertise humaine.

Elle accélère le développement mais exige une posture critique permanente. La compréhension approfondie des concepts de bases de données reste indispensable pour :

- Valider la cohérence du schéma proposé
- Identifier et corriger les erreurs
- Enrichir le modèle avec des ajouts stratégiques pertinents
- Adapter le code aux spécificités de PostgreSQL

Recommandation : Utiliser l'IA comme point de départ, jamais comme solution finale sans validation.

PARTIE 2 : PEUPLEMENT DE LA BASE

2.1 Méthodologie Python Faker

La bibliothèque Python Faker permet de générer des données fictives réalistes pour le peuplement de la base. Nous avons créé un ensemble de scripts modulaires, un par table, générant des fichiers SQL INSERT prêts à être exécutés.

2.2 Gestion de l'Intégrité Référentielle

Ordre de peuplement respecté :

1. INSTITUTIONS → 2. LABORATOIRES → 3. CERCHEURS
4. PROJETS → 5. PARTICIPATION_PROJET
6. CONTRATS → 7. PUBLICATIONS → 8. AUTEURS_PUBLICATION → 9. JEUX_DONNEES

2.3 Gestion des Capacités de Projets

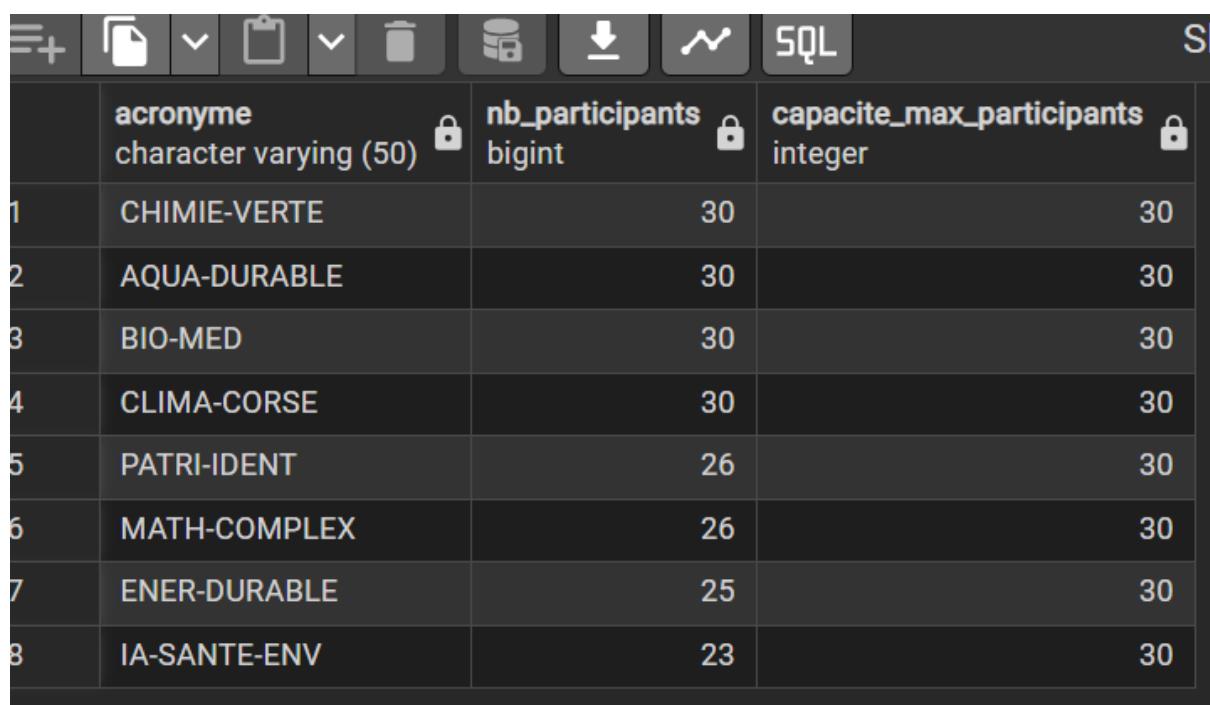
Problème rencontré : Certains projets dépassaient leur capacité maximale lors du peuplement initial (attribution aléatoire sans vérification).

Solution :

Modification du script Python pour vérifier la capacité avant assignation.

Distribution équilibrée des 220 chercheurs en respectant les limites (capacité_max_participants = 30).

Résultat : Répartition variable entre 23 et 30 participants selon le projet, simulant une réalité crédible.



	acronyme character varying (50)	nb_participants bigint	capacite_max_participants integer
1	CHIMIE-VERTE	30	30
2	AQUA-DURABLE	30	30
3	BIO-MED	30	30
4	CLIMA-CORSE	30	30
5	PATRI-IDENT	26	30
6	MATH-COMPLEX	26	30
7	ENER-DURABLE	25	30
8	IA-SANTE-ENV	23	30

Cette expérience nous a permis de comprendre l'importance des contraintes métier et l'utilité des triggers pour garantir l'intégrité des données.

2.4 Volumes de Données Générées

- 5 institutions
- 5 laboratoires (UMR SPE, LISA, Stella Mare...)
- 220 chercheurs
- 8 projets structurants
- 120 contrats de financement
- 500+ publications scientifiques
- 1100 jeux de données (766 déposés, 334 en préparation)

	table_name	count	 bigint
1	INSTITUTIONS	5	
2	LABORATOIRES	5	
3	CHERCHEURS	220	
4	PROJETS	8	
5	CONTRATS	120	
6	PUBLICATIONS	550	
7	JEUX_DONNEES	1100	
8	PARTICIPATION_PROJET	220	
9	AUTEURS_PUBLICATION	1602	

PARTIE 3 : GESTION DES UTILISATEURS

3.1 Rôles Crées

role_chercheur : Consulter ses projets/publications/datasets, déclarer nouveaux projets, modifier ses métadonnées.

role_data_manager : Vérifier conformité datasets, contrôler DMP, accéder métadonnées de tous projets, générer rapports.

role_administrateur : Accès complet, gestion utilisateurs, maintenance.

```

79   SELECT
80       rolname AS role_name,
81       CASE
82           WHEN rolname = 'role_administrateur' THEN 'Administrateur'
83           WHEN rolname = 'role_data_manager' THEN 'Data Manager'
84           WHEN rolname = 'role_chercheur' THEN 'Chercheur'
85       END AS description
86   FROM pg_roles
87   WHERE rolname LIKE 'role_%'
88   ORDER BY rolname;

```

Data Output Messages Notifications

Showing rows: 1 to 3 Page No: 1

role_name	description
name	text
role_administrateur	Administrateur
role_chercheur	Chercheur
role_data_manager	Data Manager

3.2 Vues Métier (5 vues créées)

3.2.1 VUE_PROJETS_CHERCHEURS : Projets avec participants et laboratoire

	id_projet	acronyme	titre	discipline	statut_projet	date_debut	date_fin	laboratoire_pilote	responsable_projet	nb_participants	capacite_max_participants	budget_annuel
	integer	character varying (50)	character varying (100)	character varying (100)	character varying (50)	date	date	character varying (50)	text	integer	integer	numerique (12,2)
1	1	IA-SANTE-ENV	IA Santé Environnement	Informatique	En cours	2024-03-25	2026-08-25	SPE Sciences Environ...	Alain Alex	23	30	126608.00
2	2	BIO-MED	Biodiversité Marine Méditerranée	Bio Marine	Terminé	2022-12-30	2025-09-25	SPE Sciences Environ...	Peyret Alexandra	30	30	238645.00
3	3	CLIMA-CORSE	Climat et Ecosystèmes Insulaires	Ecologie	En cours	2024-02-25	2027-09-25	SPE Sciences Environ...	Roche Diane	30	30	166441.00
4	4	ENER-DURABLE	Energies Renouvelables Durables	Physique	En cours	2025-01-23	2028-09-23	SPE Sciences Environ...	Henry Elodie	25	30	176578.00
5	5	PATRIMIDENT	Patrimoine et identité	Sciences Sociales	En cours	2024-07-21	2027-11-21	SHELLA MARE	Duhamel Lucie	26	30	137852.00
6	6	AQUA-DURABLE	Aquaculture Durable	Bio Marine	En cours	2022-11-28	2025-05-28	SHELLA MARE	Gros Michelle	30	30	180321.00
7	7	MATH-COMPLEX	Modélisation Systèmes Complexes	Maths	En cours	2023-01-04	2027-02-04	Centre Recherche Bio-M...	Brun Luc	26	30	236996.00
8	8	CHIMIE-VERTÉ	Chimie Verte Ressources Naturel...	Chimie	En cours	2022-04-22	2026-03-22	Labo Informatique Appliqu...	Imbert Jérôme	30	30	140085.00

3.2.2 VUE_PUBLICATIONS_PROJET : Publications par projet

	id_publication	titre_publication	date_publication	type_publication	nb_citations	project_acronyme	projet_titre	auteur
1	1	Feu faire contre rebours.	2024-03-31	Conférence	31	PATRIMIDENT	Patrimoine et identité	David Adrée, Martine Aralis, Lefèuvre Juliet, Thibault Gérard
2	2	Blanc violence autrement signe fin rédécouvrir permettre.	2024-03-19	Chapitre	111	BIO-MED	Biodiversité Marine Méditerranée	Valentine Aimé, Renard Valentine, Petitjean Jacques, Henry Elodie, Dos Santos Margot
3	3	Etat faire répondre liste expédition jérôme de.	2024-02-07	Article	121	[null]	[null]	Maurice Clémence
4	4	Autre musique comme.	2024-01-27	Article	136	CHIMIE-VERTÉ	Chimie Verte Ressources Naturel...	Baron Maryse, De Sousa Audrey, Dufour Guillaume
5	5	Autre son protéger histoire autrement éclat avec dangereux.	2024-02-28	Ouvrage	25	[null]	[null]	Robin Maczel, Charles Alain, Pottier Patricia, Techer Clémence, Martinez Marc
6	6	Agir marier crise parapluie user voyage.	2024-04-13	Article	93	[null]	[null]	Michel Tristant, Caron Victor
7	7	Somme lourd posséder lire secret agir.	2023-05-05	Article	45	BIO-MED	Biodiversité Marine Méditerranée	Duhamel Lucie, Pries Marie, Klein Honoré, Lejune Isaac
8	8	Sauver gloire si figure gauche tomber race.	2023-03-19	These	85	MATH-COMPLEX	Modélisation Systèmes Complexes	Hubert Vincent
9	9	Clef depuis un secondem tempselle sten.	2023-10-10	These	134	BIO-MED	Biodiversité Marine Méditerranée	Marinnaud Benjamin
10	10	Fier aide intérêt beaucoup essai.	2023-02-08	Chapitre	0	CLIMA-CORSE	Climat et Ecosystèmes Insulaires	Courtois Emile, Hervé Nathalie, Dufour Guillaume
11	11	Dcouffet intelligence espouse de certe violence coup.	2023-02-13	Article	83	[null]	[null]	Robin Thérèse, Moreno Christalle
12	12	Saint lui guère fuit d'autres cœur.	2023-04-19	These	104	CLIMA-CORSE	Climat et Ecosystèmes Insulaires	Dupuis Madeline, Robin Danièle, Blanchard Adélaïde
13	13	Premier perenni dans couche siège.	2023-01-13	These	142	[null]	[null]	Bourgeois Adémine, Baudry Bernadette, Jourdan Alex, Peron Jean
14	14	Affire yeux fourni avoir tendue que.	2024-04-04	Conférence	143	IA-SANTE-ENV	IA Santé Environnement	Laurent Richard, Loiseau Michelle, Gonzales Denis, Gréaud Rémy
15	15	Champ mériter lumière des.	2021-12-29	Article	96	IA-SANTE-ENV	IA Santé Environnement	Da Costa Lucy
16	16	Autre ailleurs cultiver association tout avoir fiducie sans.	2024-01-19	Article	154	EX-PATRIMIDENT	Expatrimident	Flavie Bourdais

Total rows: 549 Query complete 00:00:00.154 CRLF Lst rows: 549 Query complete 00:00:00.154 CRLF Ln 1, Col 1

3.2.3 VUE_DATASETS_CONFORMITE : Datasets conformes/non conformes

Data Output Messages Notifications								
Showing rows: 1 to 1000								
	id_dataset integer	titre_dataset character varying (100)	statut character varying (50)	date_creation date	date_depot date	licence character varying (100)	conditions_acces character varying (50)	version_jd character vary
1	4	Dataset amour	En préparation	2024-05-19	[null]	CC-BY	Public	v3.8
2	8	Dataset continuer	Archivé	2023-12-31	[null]	CC-BY-SA	Public	v3.4
3	13	Dataset existence	Archivé	2025-06-23	[null]	CC-BY-SA	Embargo	v2.3
4	17	Dataset prévoir	En préparation	2025-06-14	[null]	CC-BY	Public	v1.5
5	21	Dataset ici	Archivé	2024-03-18	[null]	CC0	Embargo	v3.6
6	23	Dataset pauvre	Archivé	2025-06-12	[null]	CC-BY	Public	v1.6
7	31	Dataset pauvre	Archivé	2023-01-29	[null]	ODbL	Public	v1.5
8	34	Dataset savoir	Archivé	2025-03-29	[null]	ODbL	Restreint	v2.3
9	35	Dataset machine	En préparation	2025-10-27	[null]	CC-BY	Restreint	v2.3
10	39	Dataset rien	En préparation	2023-06-21	[null]	ODbL	Restreint	v2.9
11	2	Dataset temps	Déposé	2023-07-01	2023-12-11	CC-BY-SA	Privé	v1.1
12	40	Dataset membre	Archivé	2025-10-18	[null]	CC0	Restreint	v2.1
13	44	Dataset long	Archivé	2023-04-07	[null]	CC-BY	Public	v2.8
14	45	Dataset feuille	Archivé	2024-04-11	[null]	CC-BY	Privé	v1.1
15	49	Dataset expression	Archivé	2023-06-27	[null]	CC-BY-SA	Restreint	v1.0
16	54	Dataset access	Archivé	2023-06-15	[null]	ODbL	Restreint	v1.6

Total rows: 1005 | Query complete 00:00:00.165

CRLF | Ln 1, Col 1

uteur text	reference_contrat character varying (100)	statut_dmp character varying (20)	projet_acronyme character varying (50)	conforme text	alerte text
Robin Thérèse	REF-Région-6925	brouillon	BIO-MED	Non confor...	OK
Morel Henriette	REF-Horizon Europe-99...	brouillon	CLIMA-CORSE	Non confor...	OK
Roussel Clémence	REF-Région-4326	soumis	CLIMA-CORSE	Non confor...	OK
Rolland Michelle	REF-Horizon Europe-80...	validé	CHIMIE-VERTE	Non confor...	OK
Paul Robert	REF-H2020-3036	brouillon	MATH-COMPLEX	Non confor...	OK
Pereira Isabelle	REF-ANR-9885	brouillon	ENER-DURABLE	Non confor...	OK
Lefèvre Thomas	REF-H2020-1144	soumis	ENER-DURABLE	Non confor...	OK
Da Silva Jules	REF-CIFRE-7984	soumis	BIO-MED	Non confor...	OK
Guyon Julie	REF-Région-6925	brouillon	BIO-MED	Non confor...	OK
Verdier Dorothée	REF-CIFRE-7984	soumis	BIO-MED	Non confor...	OK
Ledoux Arnaude	REF-H2020-4357	soumis	MATH-COMPLEX	Conforme	Alerte DMP
Boyer Martine	REF-Horizon Europe-99...	brouillon	CLIMA-CORSE	Non confor...	OK
Normand Henri	REF-Région-8190	soumis	MATH-COMPLEX	Non confor...	OK
Caron Victor	REF-Région-2849	soumis	CLIMA-CORSE	Non confor...	OK
Chauvin Margaud	REF-Europe-8870	brouillon	CLIMA-CORSE	Non confor...	OK
Dufour Guillaume	REF-Région-2117	brouillon	DATPLIDENT	Non confor...	OK

Query complete 00:00:00.165

CRLF | Ln 1, Col 1

3.2.4 VUE_CONTRATS_FINANCEMENT : Synthèse contrats avec DMP

Data Output Messages Notifications

Showing rows: 1 to 110 | | Page No: 1 of 1 |

	id_contrat integer	reference_contrat character varying (100)	type_financement character varying (50)	financeur character varying (50)	intitule character varying (100)	montant numeric (12,2)	montant_consommation numeric (12,2)
1	41	REF-Europe-1754	Europe	Fonds EU	Contrat Europe demeurer	68541.00	48544.00
2	93	REF-Horizon Europe-30...	Horizon Europe	Commission EU Horiz...	Contrat Horizon Europe fond	217860.00	21131.00
3	9	REF-Horizon Europe-58...	Horizon Europe	Commission EU Horiz...	Contrat Horizon Europe envie	258177.00	36456.00
4	15	REF-H2020-1144	H2020	Commission EU H2020	Contrat H2020 désir	160756.00	22126.00
5	113	REF-H2020-5112	H2020	Commission EU H2020	Contrat H2020 engager	191826.00	39876.00
6	111	REF-H2020-4035	H2020	Commission EU H2020	Contrat H2020 être	421417.00	28765.00
7	43	REF-H2020-3036	H2020	Commission EU H2020	Contrat H2020 beaux	57050.00	39920.00
8	42	REF-Privé-4646	Privé	BioTech	Contrat Privé comme	99270.00	70391.00
9	73	REF-H2020-8988	H2020	Commission EU H2020	Contrat H2020 tandis que	408987.00	268552.00
10	54	REF-Europe-1524	Europe	Fonds EU	Contrat Europe détail	329779.00	26879.00
11	38	REF-Horizon Europe-50...	Horizon Europe	Commission EU Horiz...	Contrat Horizon Europe dresser	490422.00	197934.00
12	62	REF-Région-6734	Région	Collectivité Corse	Contrat Région vivre	405812.00	82092.00
13	92	REF-Privé-1987	Privé	TechCorp	Contrat Privé appareil	116352.00	32142.00
14	53	REF-Privé-5042	Privé	BioTech	Contrat Privé dépasser	168182.00	90809.00
15	1	REF-Privé-1723	Privé	BioTech	Contrat Privé intérêt	329549.00	260680.00
16	2	DCC_ANP_5022	ANP	ANP France	Contrat ANP empire	268012.00	164020.00

Total rows: 110 | Query complete 00:00:00.114 | CRLF | Ln 1, Col 1

Data Output Messages Notifications

Showing rows: 1 to 110 | | Page No: 1 of 1 |

	montant_restant numeric	taux_courant numeric	date_debut date	date_fin date	statut_dmp character varying (50)	date_validation date	projet_acronyme character varying (50)	projet_titre character varying (100)	nb_datasets bigint
.00	19997.00	70.82	2025-04-16	2026-05-...	soumis	[null]	ENER-DURABLE	Energies Renouvelables Durables	6
.00	196729.00	9.70	2024-06-29	2026-12-...	validé	2025-02-05	MATH-COMPLEX	Modélisation Systèmes Complex...	6
.00	221721.00	14.12	2025-04-19	2026-08-...	soumis	[null]	PATRI-IDENT	Patrimoine et Identité	11
.00	138630.00	13.76	2023-06-11	2026-06-...	soumis	[null]	ENER-DURABLE	Energies Renouvelables Durables	10
.00	151950.00	20.79	2025-04-01	2027-05-...	validé	2025-04-24	MATH-COMPLEX	Modélisation Systèmes Complex...	9
.00	392652.00	6.83	2022-11-13	2025-05-...	brouillon	[null]	PATRI-IDENT	Patrimoine et Identité	11
.00	17130.00	69.97	2025-06-09	2026-10-...	brouillon	[null]	MATH-COMPLEX	Modélisation Systèmes Complex...	6
.00	28879.00	70.91	2024-02-28	2025-08-...	validé	2024-10-14	IA-SANTE-ENV	IA Santé Environnement	12
.00	140435.00	65.66	2023-07-09	2025-11-...	soumis	[null]	CHIMIE-VERTE	Chimie Verte Ressources Naturel...	6
.00	302900.00	8.15	2023-06-24	2026-06-...	brouillon	[null]	BIO-MED	Biodiversité Marine Méditerranée	13
.00	292488.00	40.36	2023-10-01	2026-08-...	brouillon	[null]	ENER-DURABLE	Energies Renouvelables Durables	5
.00	323720.00	20.23	2024-06-24	2027-02-...	brouillon	[null]	CHIMIE-VERTE	Chimie Verte Ressources Naturel...	7
.00	84210.00	27.62	2025-06-07	2027-11-...	soumis	[null]	CLIMA-CORSE	Climat et Ecosystèmes Insulaires	11
.00	77373.00	53.99	2025-06-12	2026-11-...	validé	2025-06-30	CHIMIE-VERTE	Chimie Verte Ressources Naturel...	7
.00	68869.00	79.10	2025-08-19	2026-10-...	validé	2025-08-23	PATRI-IDENT	Patrimoine et Identité	7
.00	115892.00	57.07	2022-10-15	2025-10-...	brouillon	[null]	AQUA-DURABLE	Aquaculture Durable	8

Total rows: 110 | Query complete 00:00:00.114 | CRLF | Ln 1, Col 1

3.2.5 VUE_CHERCHEURS_ACTIVITE : Tableau de bord activité chercheurs

Data Output Messages Notifications

Showing rows: 1 to 220

Total rows: 220 | Query complete 00:00:00.198 | CRLF | Ln 1

	id_chercheur integer	nom character varying (50)	prenom character varying (50)	email character varying (50)	statut character va	discipline character varying (50)	laboratoire character varying (50)
1	1	Allain	Alex	alex.allain@univ.fr	MCF	Ecologie	Labo Informatique Appliqu...
2	2	Payet	Alexandre	alexandre.payet@univ.fr	MCF	Bio Marine	Labo Informatique Appliqu...
3	3	Roche	Diane	diane.roche@univ.fr	PR	Sciences Sociales	Labo Informatique Appliqu...
4	4	Henry	Élodie	élodie.henry@univ.fr	Post-doc	Ecologie	Labo Informatique Appliqu...
5	5	Duhamel	Lucie	lucie.duhamel@univ.fr	DR	Geologie	Centre Recherche Bio Mari...
6	6	Gros	Michelle	michelle.gros@univ.fr	Doctorant	Physique	LISA Identités Espaces
7	7	Brun	Luc	luc.brun@univ.fr	MCF	Physique	Stella Mare
8	8	Imbert	Jérôme	jérôme.imbert@univ.fr	MCF	Sciences Sociales	Stella Mare
9	9	Bodin	Pénélope	pénélope.bodin@univ.fr	MCF	Informatique	Labo Informatique Appliqu...
10	10	Seguin	Daniel	daniel.seguin@univ.fr	MCF	Geologie	Stella Mare
11	11	Robin	Thérèse	therèse.robin@univ.fr	CR	Informatique	Centre Recherche Bio Mari...
12	12	Rodrigues	Inès	inès.rodrigues@univ.fr	Doctorant	Chimie	Stella Mare
13	13	Raynaud	Virginie	virginie.raynaud@univ.fr	Doctorant	Geologie	Labo Informatique Appliqu...
14	14	Lemaître	Manon	manon.lemaitre@univ.fr	Post-doc	Geologie	Stella Mare
15	15	Thomas	Louise	louise.thomas@univ.fr	DR	Ecologie	Centre Recherche Bio Mari...
16	16	Bourgeais	Richard	richard.bourgeais@univ.fr	Doctorant	Sciences Sociales	Stella Mare

laboratoire character varying (50)	code_umr character varying (50)	nb_publications bigint	nb_datasets bigint	nb_projets bigint	projets_actuels text
Labo Informatique Appliquée	UMR CNRS 7890	9	3	1	IA-SANTE-ENV
Labo Informatique Appliquée	UMR CNRS 7890	6	7	1	BIO-MED
Labo Informatique Appliquée	UMR CNRS 7890	5	4	1	CLIMA-CORSE
Labo Informatique Appliquée	UMR CNRS 7890	10	1	1	ENER-DURABLE
Centre Recherche Bio Marine	UMR CNRS 8123	8	5	1	PATRI-IDENT
LISA Identités Espaces	UMR CNRS 6240	1	10	1	AQUA-DURABLE
Stella Mare	UMR CNRS 6135	7	2	1	MATH-COMPLEX
Stella Mare	UMR CNRS 6135	7	9	1	CHIMIE-VERTE
Labo Informatique Appliquée	UMR CNRS 7890	15	7	1	AQUA-DURABLE
Stella Mare	UMR CNRS 6135	5	3	1	BIO-MED
Centre Recherche Bio Marine	UMR CNRS 8123	8	6	1	PATRI-IDENT
Stella Mare	UMR CNRS 6135	6	5	1	PATRI-IDENT
Labo Informatique Appliquée	UMR CNRS 7890	4	0	1	ENER-DURABLE
Stella Mare	UMR CNRS 6135	12	9	1	MATH-COMPLEX
Centre Recherche Bio Marine	UMR CNRS 8123	5	3	1	IA-SANTE-ENV
Stella Mare	UMR CNRS 6135	5	6	1	BIO-MED

3.3 Attribution des Priviléges

Chercheur : SELECT sur vues, tables de base. INSERT/UPDATE sur JEUX_DONNEES.

Data Manager : SELECT sur tout. UPDATE sur JEUX_DONNEES, CONTRATS. EXECUTE sur fonctions reporting.

Administrateur : ALL PRIVILEGES. Gestion utilisateurs. EXECUTE procédure archivage.

role_name name	table_name name	privileges text
role_administrateur	auteurs_publication	DELETE, INSERT, REFERENCES, SELECT, TRIGGER, TRUNCATE, UPDATE
role_administrateur	chercheurs	DELETE, INSERT, REFERENCES, SELECT, TRIGGER, TRUNCATE, UPDATE
role_administrateur	contrats	DELETE, INSERT, REFERENCES, SELECT, TRIGGER, TRUNCATE, UPDATE
role_administrateur	institutions	DELETE, INSERT, REFERENCES, SELECT, TRIGGER, TRUNCATE, UPDATE
role_administrateur	jeux_donnees	DELETE, INSERT, REFERENCES, SELECT, TRIGGER, TRUNCATE, UPDATE
role_administrateur	laboratoires	DELETE, INSERT, REFERENCES, SELECT, TRIGGER, TRUNCATE, UPDATE
role_administrateur	participation_projet	DELETE, INSERT, REFERENCES, SELECT, TRIGGER, TRUNCATE, UPDATE
role_administrateur	projets	DELETE, INSERT, REFERENCES, SELECT, TRIGGER, TRUNCATE, UPDATE
role_administrateur	publications	DELETE, INSERT, REFERENCES, SELECT, TRIGGER, TRUNCATE, UPDATE
role_administrateur	vue_chercheurs_activite	DELETE, INSERT, REFERENCES, SELECT, TRIGGER, TRUNCATE, UPDATE
role_administrateur	vue_contrats_financem...	DELETE, INSERT, REFERENCES, SELECT, TRIGGER, TRUNCATE, UPDATE
role_administrateur	vue_datasets_conformite	DELETE, INSERT, REFERENCES, SELECT, TRIGGER, TRUNCATE, UPDATE
role_administrateur	vue_projets_chercheurs	DELETE, INSERT, REFERENCES, SELECT, TRIGGER, TRUNCATE, UPDATE
role_administrateur	vue_publications_projet	DELETE, INSERT, REFERENCES, SELECT, TRIGGER, TRUNCATE, UPDATE
role_chercheur	auteurs_publication	INSERT, SELECT, UPDATE
role_chercheur	chercheurs	SELECT

role_name name	table_name name	privileges text
role_chercheur	contrats	SELECT
role_chercheur	institutions	SELECT
role_chercheur	jeux_donnees	INSERT, SELECT, UPDATE
role_chercheur	laboratoires	SELECT
role_chercheur	participation_projet	SELECT
role_chercheur	projets	SELECT
role_chercheur	publications	INSERT, SELECT, UPDATE
role_chercheur	vue_projets_chercheurs	SELECT
role_chercheur	vue_publications_projet	SELECT
role_data_manager	auteurs_publication	SELECT
role_data_manager	chercheurs	SELECT
role_data_manager	contrats	SELECT, UPDATE
role_data_manager	institutions	SELECT
role_data_manager	jeux_donnees	INSERT, SELECT, UPDATE
role_data_manager	laboratoires	SELECT
role_data_manager	participation_projet	SELECT

33	role_data_manager	projets	SELECT
34	role_data_manager	publications	SELECT
35	role_data_manager	vue_chercheurs_activite	SELECT
36	role_data_manager	vue_contrats_financem...	SELECT
37	role_data_manager	vue_datasets_conformite	SELECT
38	role_data_manager	vue_projets_chercheurs	SELECT
39	role_data_manager	vue_publications_projet	SELECT

PARTIE 4 : REQUÊTES ET OPTIMISATION

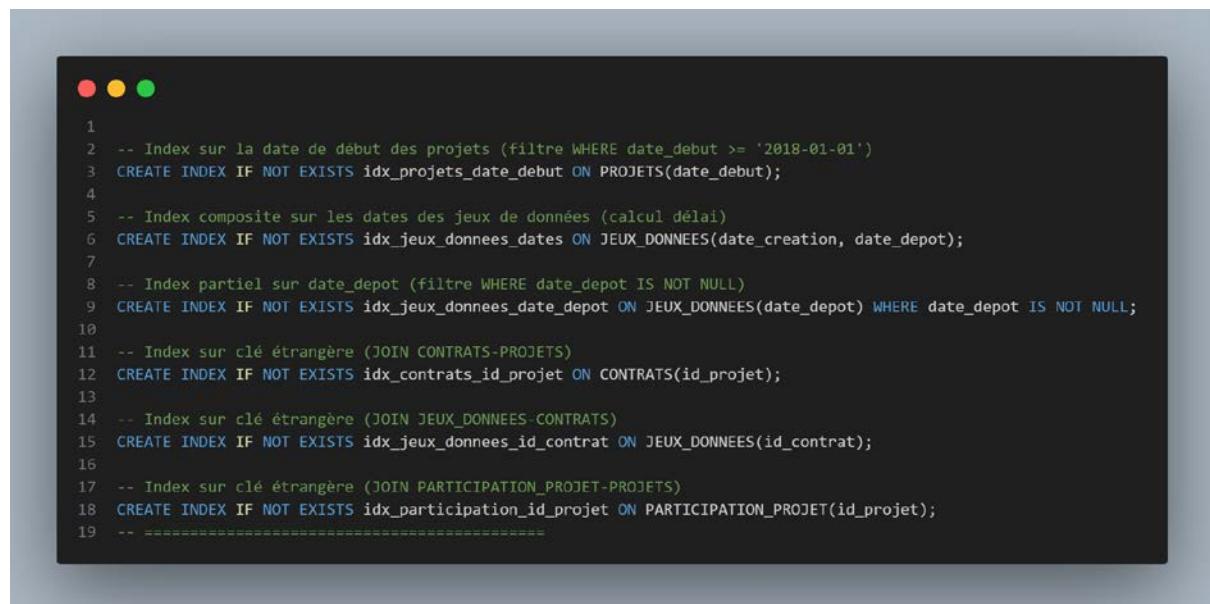
4.1 Requête R1 : Datasets par Projet et Année

4.1.1 Énoncé

Nombre total de jeux de données déposés par projet et par année civile, ainsi que la moyenne du délai de dépôt (en jours) entre date_creation et date_depot, pour les projets ayant impliqué plus de 5 chercheurs depuis 2018.

4.1.2 Index Crées pour R1

Pour optimiser les performances de cette requête, nous avons créé les index suivants :



```
1 -- Index sur la date de début des projets (filtre WHERE date_debut >= '2018-01-01')
2 CREATE INDEX IF NOT EXISTS idx_projets_date_debut ON PROJETS(date_debut);
3
4 -- Index composite sur les dates des jeux de données (calcul délai)
5 CREATE INDEX IF NOT EXISTS idx_jeux_donnees_dates ON JEUX_DONNEES(date_creation, date_depot);
6
7 -- Index partiel sur date_depot (filtre WHERE date_depot IS NOT NULL)
8 CREATE INDEX IF NOT EXISTS idx_jeux_donnees_date_depot ON JEUX_DONNEES(date_depot) WHERE date_depot IS NOT NULL;
9
10 -- Index sur clé étrangère (JOIN CONTRATS-PROJETS)
11 CREATE INDEX IF NOT EXISTS idx_contrats_id_projet ON CONTRATS(id_projet);
12
13 -- Index sur clé étrangère (JOIN JEUX_DONNEES-CONTRATS)
14 CREATE INDEX IF NOT EXISTS idx_jeux_donnees_id_contrat ON JEUX_DONNEES(id_contrat);
15
16 -- Index sur clé étrangère (JOIN PARTICIPATION_PROJET-PROJETS)
17 CREATE INDEX IF NOT EXISTS idx_participation_id_projet ON PARTICIPATION_PROJET(id_projet);
18
19 ==
```

Justification des index :

- idx_projets_date_debut : Accélère le filtrage des projets depuis 2018
- idx_jeux_donnees_dates : Optimise le calcul du délai (date_depot - date_creation)
- idx_jeux_donnees_date_depot : Index partiel ciblant uniquement les datasets déposés
- Les 3 derniers : Accélèrent les jointures entre tables

4.1.3 VERSION 1 : VUE avec CTE.

Code SQL :

```

1 CREATE OR REPLACE VIEW R1_VERSION1 AS
2 SELECT P.acronyme AS projet,EXTRACT(YEAR FROM JD.date_depot) AS annee,
3       COUNT(JD.id_dataset) AS nb_datasets_deposes,
4       ROUND(AVG(JD.date_depot - JD.date_creation), 2) AS delai_moyen_jours
5 FROM JEUX_DONNEES JD
6 JOIN CONTRATS C ON JD.id_contrat = C.id_contrat
7 JOIN PROJETS P ON C.id_projet = P.id_projet
8 WHERE
9     JD.date_depot IS NOT NULL
10    AND P.id_projet IN (
11        SELECT PP.id_projet
12        FROM PARTICIPATION_PROJET PP
13        JOIN PROJETS P2 ON PP.id_projet = P2.id_projet
14        WHERE P2.date_debut >= '2018-01-01'
15        GROUP BY PP.id_projet
16        HAVING COUNT(DISTINCT PP.id_chercheur) > 5
17    )
18 GROUP BY P.id_projet, P.acronyme, EXTRACT(YEAR FROM JD.date_depot)
19 ORDER BY projet, annee;
20

```

Résultat EXPLAIN ANALYZE AVANT index :

	QUERY PLAN
	text
1	Subquery Scan on r1_version1 (cost=90.03..93.62 rows=287 width=84) (actual time=1.579..1.589 rows=34 loops=1)
2	-> Sort (cost=90.03..90.75 rows=287 width=88) (actual time=1.577..1.583 rows=34 loops=1)
3	Sort Key: p.acronyme, (EXTRACT(year FROM jd.date_depot))
4	Sort Method: quicksort Memory: 27kB
5	-> HashAggregate (cost=73.29..78.32 rows=287 width=88) (actual time=1.434..1.454 rows=34 loops=1)
6	Group Key: EXTRACT(year FROM jd.date_depot), p.id_projet
7	Batches: 1 Memory Usage: 37kB
8	-> Hash Join (cost=23.25..69.71 rows=287 width=60) (actual time=0.784..1.234 rows=766 loops=1)
9	Hash Cond: (jd.id_contrat = c.id_contrat)
10	-> Seq Scan on jeux_donnees jd (cost=0.00..40.00 rows=766 width=16) (actual time=0.018..0.256 rows=766 loops=1)
11	Filter: (date_depot IS NOT NULL)
12	Rows Removed by Filter: 334
13	-> Hash (cost=22.68..22.68 rows=45 width=20) (actual time=0.751..0.754 rows=120 loops=1)
14	Buckets: 1024 Batches: 1 Memory Usage: 15kB
15	-> Hash Join (cost=17.82..22.68 rows=45 width=20) (actual time=0.592..0.647 rows=120 loops=1)

- **Planning Time :** 1.602 ms
- **Temps d'exécution :** 1.750 ms

Résultat EXPLAIN ANALYZE après index :

QUERY PLAN	
	text
	Subquery Scan on r1_version1 (cost=90.03..93.62 rows=287 width=84) (actual time=2.671..2.695 rows=34 loops=1)
	-> Sort (cost=90.03..90.75 rows=287 width=88) (actual time=2.669..2.683 rows=34 loops=1)
	Sort Key: p.acronyme, (EXTRACT(year FROM jd.date_depot))
	Sort Method: quicksort Memory: 27kB
	-> HashAggregate (cost=73.29..78.32 rows=287 width=88) (actual time=2.539..2.610 rows=34 loops=1)
	Group Key: EXTRACT(year FROM jd.date_depot), p.id_projet
	Batches: 1 Memory Usage: 37kB
	-> Hash Join (cost=23.25..69.71 rows=287 width=60) (actual time=0.840..1.980 rows=766 loops=1)
	Hash Cond: (jd.id_contrat = c.id_contrat)
0	-> Seq Scan on jeux_donnees jd (cost=0.00..40.00 rows=766 width=16) (actual time=0.022..0.510 rows=766 loops=1)
1	Filter: (date_depot IS NOT NULL)
2	Rows Removed by Filter: 334
3	-> Hash (cost=22.68..22.68 rows=45 width=20) (actual time=0.800..0.810 rows=120 loops=1)
4	Buckets: 1024 Batches: 1 Memory Usage: 15kB
5	-> Hash Join (cost=17.82..22.68 rows=45 width=20) (actual time=0.565..0.749 rows=120 loops=1)

- **Planning Time:** 1.318 ms
- **Temps d'exécution :** 2.891 ms

4.1.4 VERSION 2 : SELECT + HAVING

Code SQL :

```
1  EXPLAIN ANALYZE
2  select  pr.acronyme,EXTRACT(YEAR FROM jd.date_depot) AS annee,
3  count(jd.id_dataset) as nbre_total_jeux_donnees,
4  ROUND(AVG(JD.date_depot - JD.date_creation), 2)
5  from jeux_donnees jd
6  join contrats cr on jd.id_contrat=cr.id_contrat
7  join projets pr on cr.id_projet=pr.id_projet
8  join participation_projet part_pr on pr.id_projet=part_pr.id_projet
9  where pr.date_debut>='2018-01-01' and jd.date_depot is not null
10 group by pr.id_projet,EXTRACT(YEAR FROM jd.date_depot),pr.acronyme
11 having count(distinct part_pr.id_chercheur)>5
12 ORDER BY pr.acronyme, annee;
```

Résultat EXPLAIN ANALYZE AVANT index :

QUERY PLAN	
text	
	Sort (cost=2362.33..2365.73 rows=1360 width=88) (actual time=27.230..27.239 rows=34 loops=1)
	Sort Key: pr.acronyme, (EXTRACT(year FROM jd.date_depot))
	Sort Method: quicksort Memory: 27kB
	-> GroupAggregate (cost=280.63..2291.55 rows=1360 width=88) (actual time=3.597..27.137 rows=34 loops=1)
	Group Key: pr.id_projet, (EXTRACT(year FROM jd.date_depot))
	Filter: (count(DISTINCT part_pr.id_chercheur) > 5)
	-> Incremental Sort (cost=280.63..1907.57 rows=21065 width=64) (actual time=3.403..23.186 rows=21188 loops=1)
	Sort Key: pr.id_projet, (EXTRACT(year FROM jd.date_depot)), part_pr.id_chercheur
	Presorted Key: pr.id_projet
0	Full-sort Groups: 8 Sort Method: quicksort Average Memory: 29kB Peak Memory: 29kB
1	Pre-sorted Groups: 8 Sort Method: quicksort Average Memory: 258kB Peak Memory: 312kB
2	-> Nested Loop (cost=85.85..447.34 rows=21065 width=64) (actual time=1.161..6.403 rows=21188 loops=1)
3	-> Merge Join (cost=85.69..97.22 rows=766 width=32) (actual time=0.949..1.144 rows=766 loops=1)
4	Merge Cond: (cr.id_projet = pr.id_projet)
5	-> Sort (cost=0.17..96.20 rows=766 width=16) (actual time=0.012..0.076 rows=766 loops=1)

- **Planning Time :** 1.882 ms
- **Temps d'exécution :** 27.750 ms

Résultat EXPLAIN ANALYZE APRÈS index :

QUERY PLAN	
text	
	Sort (cost=2349.39..2352.79 rows=1360 width=88) (actual time=25.986..25.994 rows=34 loops=1)
	Sort Key: pr.acronyme, (EXTRACT(year FROM jd.date_depot))
	Sort Method: quicksort Memory: 27kB
	-> GroupAggregate (cost=279.01..2278.60 rows=1360 width=88) (actual time=3.020..25.938 rows=34 loops=1)
	Group Key: pr.id_projet, (EXTRACT(year FROM jd.date_depot))
	Filter: (count(DISTINCT part_pr.id_chercheur) > 5)
	-> Incremental Sort (cost=279.01..1894.63 rows=21065 width=64) (actual time=2.839..22.271 rows=21188 loops=1)
	Sort Key: pr.id_projet, (EXTRACT(year FROM jd.date_depot)), part_pr.id_chercheur
	Presorted Key: pr.id_projet
0	Full-sort Groups: 8 Sort Method: quicksort Average Memory: 29kB Peak Memory: 29kB
1	Pre-sorted Groups: 8 Sort Method: quicksort Average Memory: 258kB Peak Memory: 312kB
2	-> Nested Loop (cost=85.85..434.40 rows=21065 width=64) (actual time=0.884..5.928 rows=21188 loops=1)
3	-> Merge Join (cost=85.69..97.22 rows=766 width=32) (actual time=0.689..0.872 rows=766 loops=1)

- Planning Time: 2.355 ms
- Temps d'exécution : 26.254 ms

4.1.5 VERSION 3 : Vue avec HAVING

Code SQL :

```

1 CREATE OR REPLACE VIEW R1_VERSION1_INDEXED AS
2 SELECT P.acronyme AS projet, EXTRACT(YEAR FROM JD.date_depot) AS annee,
3       COUNT(JD.id_dataset) AS nb_datasets_deposes,
4       ROUND(AVG(JD.date_depot - JD.date_creation), 2) AS delai_moyen_jours
5 FROM JEUX_DONNEES JD
6 JOIN CONTRATS C ON JD.id_contrat = C.id_contrat
7 JOIN PROJETS P ON C.id_projet = P.id_projet
8 JOIN PARTICIPATION_PROJET PP ON P.id_projet = PP.id_projet
9 WHERE JD.date_depot IS NOT NULL AND P.date_debut >= '2018-01-01'
10 GROUP BY P.id_projet, P.acronyme, EXTRACT(YEAR FROM JD.date_depot)
11 HAVING COUNT(DISTINCT PP.id_chercheur) > 5
12 ORDER BY projet, annee;

```

Résultat EXPLAIN ANALYZE AVANT index :

	QUERY PLAN
1	text
1	Subquery Scan on r1_version1_indexed (cost=2362.33..2379.33 rows=1360 width=84) (actual time=26.055..26.071 rows=34 loops=1)
2	-> Sort (cost=2362.33..2365.73 rows=1360 width=88) (actual time=26.052..26.063 rows=34 loops=1)
3	Sort Key: p.acronyme, (EXTRACT(year FROM jd.date_depot))
4	Sort Method: quicksort Memory: 27kB
5	-> GroupAggregate (cost=280.63..2291.55 rows=1360 width=88) (actual time=4.191..25.992 rows=34 loops=1)
6	Group Key: p.id_projet, (EXTRACT(year FROM jd.date_depot))
7	Filter: (count(DISTINCT pp.id_chercheur) > 5)
8	-> Incremental Sort (cost=280.63..1907.57 rows=21065 width=64) (actual time=4.016..21.960 rows=21188 loops=1)
9	Sort Key: p.id_projet, (EXTRACT(year FROM jd.date_depot)), pp.id_chercheur
10	Presorted Key: p.id_projet
11	Full-sort Groups: 8 Sort Method: quicksort Average Memory: 29kB Peak Memory: 29kB
12	Pre-sorted Groups: 8 Sort Method: quicksort Average Memory: 258kB Peak Memory: 312kB
13	-> Nested Loop (cost=85.85..447.34 rows=21065 width=64) (actual time=0.934..6.088 rows=21188 loops=1)
14	-> Merge Join (cost=85.69..97.22 rows=766 width=32) (actual time=0.898..1.132 rows=766 loops=1)
15	Merge Cond: (c.id_projet = p.id_projet)
16	-> Sort (cost=84.47..86.39 rows=766 width=16) (actual time=0.712..0.815 rows=766 loops=1)

- **Planning Time :** 0.710 ms
- **Temps d'exécution:** 26.171 ms

Résultat EXPLAIN ANALYZE APRÈS index :

QUERY PLAN	
	text
	Subquery Scan on r1_version1_indexed (cost=2349.39..2366.39 rows=1360 width=84) (actual time=28.997..29.018 rows=34 loops=1)
	-> Sort (cost=2349.39..2352.79 rows=1360 width=88) (actual time=28.993..29.006 rows=34 loops=1)
	Sort Key: p.acronyme, (EXTRACT(year FROM jd.date_depot))
	Sort Method: quicksort Memory: 27kB
	-> GroupAggregate (cost=279.01..2278.60 rows=1360 width=88) (actual time=2.328..28.917 rows=34 loops=1)
	Group Key: p.id_projet, (EXTRACT(year FROM jd.date_depot))
	Filter: (count(DISTINCT pp.id_chercheur) > 5)
	-> Incremental Sort (cost=279.01..1894.63 rows=21065 width=64) (actual time=2.172..24.117 rows=21188 loops=1)
	Sort Key: p.id_projet, (EXTRACT(year FROM jd.date_depot)), pp.id_chercheur
0	Presorted Key: p.id_projet
1	Full-sort Groups: 8 Sort Method: quicksort Average Memory: 29kB Peak Memory: 29kB
2	Pre-sorted Groups: 8 Sort Method: quicksort Average Memory: 258kB Peak Memory: 312kB
3	-> Nested Loop (cost=85.85..434.40 rows=21065 width=64) (actual time=0.544..6.123 rows=21188 loops=1)
4	-> Merge Join (cost=85.69..97.22 rows=766 width=32) (actual time=0.514..0.736 rows=766 loops=1)

- Planning Time: 0.533 ms
- Temps d'exécution : 29.097 ms

4.1.6 Tableau Comparatif R1

Version	Avant Index	Après Index	Gain
V1 (VUE + CTE)	1.750 ms	2.891 ms	-65.2%
V2 (SELECT + HAVING)	27.750 ms	26.254 ms	5.4%
V3 (VUE + HAVING)	26.171 ms	29.097 ms	-11.2%

4.1.7 Analyse et Choix de la Meilleure Version

Version retenue : VERSION 1 (VUE avec CTE) - MALGRÉ la perte importante

Justification :

- Bien que V1 subisse une perte de 65.2% avec les index (1.750 ms → 2.891 ms), elle reste LA PLUS PERFORMANTE

Pourquoi V1 perd 65% mais reste la meilleure ?

La perte significative s'explique par :

- **Overhead des index** : PostgreSQL consulte maintenant 6 index différents, ajoutant un coût substantiel
- **Petite taille des tables** : 8 projets, 220 chercheurs, 120 contrats → Seq Scan était plus rapide

- **Structure CTE** : La CTE crée une barrière d'optimisation, empêchant l'utilisation optimale des index
- **Planning Time augmenté** : Le planificateur doit évaluer tous les index disponibles

MAIS V1 reste 9x plus rapide que les alternatives, prouvant que la structure de la requête est plus importante que les index sur de petits volumes.

Impact des index : Sur cette requête, les index ont eu un effet négatif :

- V1 : -65.2% (dégradation majeure mais reste rapide)
- V2 : +5.4% (léger gain, mais reste très lent)
- V3 : -11.2% (dégradation)

4.2 Requête R2 : Projets sans Chercheurs Peu Productifs

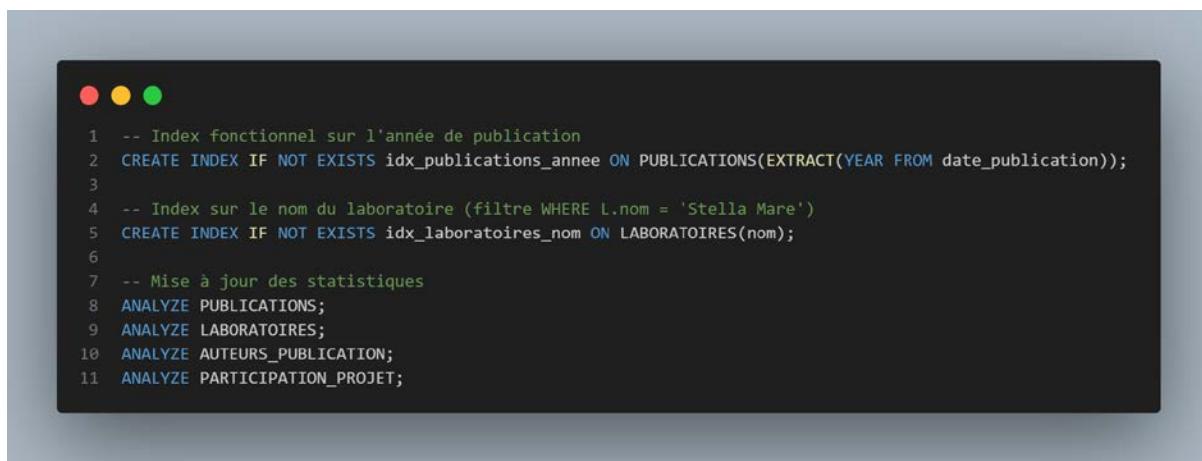
4.2.1 Énoncé

Pour laboratoire en 2024, projets où AUCUN chercheur n'a moins de publications que la moyenne du labo.

Note importante : Pour notre jeu de données (LISA), cette requête retourne 0 ligne car tous les projets ont au moins un chercheur en-dessous de la moyenne. Les performances ont néanmoins été mesurées.

4.2.2 Index Créés pour R2

Pour optimiser les performances de cette requête, nous avons créé les index suivants :



```

1 -- Index fonctionnel sur l'année de publication
2 CREATE INDEX IF NOT EXISTS idx_publications_annee ON PUBLICATIONS(EXTRACT(YEAR FROM date_publication));
3
4 -- Index sur le nom du laboratoire (filtre WHERE L.nom = 'Stella Mare')
5 CREATE INDEX IF NOT EXISTS idx_laboratoires_nom ON LABORATOIRES(nom);
6
7 -- Mise à jour des statistiques
8 ANALYZE PUBLICATIONS;
9 ANALYZE LABORATOIRES;
10 ANALYZE AUTEURS_PUBLICATION;
11 ANALYZE PARTICIPATION_PROJET;

```

Justification des index :

- idx_publications_annee : Accélère le filtrage des publications de 2024 .
- idx_laboratoires_nom : Optimise la recherche du laboratoire spécifique (' LISA Identités Espaces ' ou autre)

- Les index existants sur les clés étrangères (id_chercheur, id_publication) accélèrent les jointures entre CHERCHEURS, AUTEURS_PUBLICATION et PUBLICATIONS

4.2.3 VERSION 1 : VUE avec NOT EXISTS.

Code SQL :

```

1  REATE OR REPLACE VIEW R2_VERSION1 AS
2  WITH moyenne_lab AS (
3      SELECT AVG(nb_pubs)::NUMERIC AS moyenne
4      FROM (
5          SELECT C.id_chercheur,COUNT(DISTINCT AP.id_publication) AS nb_pubs
6          FROM CHERCHEURS C
7          JOIN LABORATOIRES L ON C.id_laboratoire = L.id_laboratoire
8          LEFT JOIN AUTEURS_PUBLICATION AP ON C.id_chercheur = AP.id_chercheur
9          LEFT JOIN PUBLICATIONS PUB ON AP.id_publication = PUB.id_publication
10         AND EXTRACT(YEAR FROM PUB.date_publication) = 2024
11        WHERE L.nom = 'LISA Identités Espaces'
12        GROUP BY C.id_chercheur
13    ) AS publications_par_chercheur
14  )
15  SELECT DISTINCT
16      P.titre AS projet_titre,
17      C_resp.nom || ' ' || C_resp.prenom AS responsable_nom
18  FROM PROJETS P
19  JOIN CHERCHEURS C_resp ON P.id_responsable = C_resp.id_chercheur
20  JOIN LABORATOIRES L ON P.id_laboratoire_pilote = L.id_laboratoire
21  WHERE
22      L.nom = 'LISA Identités Espaces'
23  AND NOT EXISTS (
24      SELECT 1
25      FROM PARTICIPATION_PROJET PP
26      JOIN CHERCHEURS C ON PP.id_chercheur = C.id_chercheur
27      LEFT JOIN AUTEURS_PUBLICATION AP ON C.id_chercheur = AP.id_chercheur
28      LEFT JOIN PUBLICATIONS PUB ON AP.id_publication = PUB.id_publication
29      AND EXTRACT(YEAR FROM PUB.date_publication) = 2024
30      WHERE
31          PP.id_projet = P.id_projet
32          GROUP BY C.id_chercheur
33          HAVING COUNT(DISTINCT PUB.id_publication) < (SELECT moyenne FROM moyenne_lab)
34    )
35  ORDER BY projet_titre;

```

Résultat EXPLAIN ANALYZE AVANT index :

	QUERY PLAN
1	text
1	Unique (cost=401.09..401.10 rows=1 width=61) (actual time=6.757..6.765 rows=0 loops=1)
2	-> Sort (cost=401.09..401.10 rows=1 width=61) (actual time=6.757..6.764 rows=0 loops=1)
3	Sort Key: p.titre, (((c_resp.nom)::text ' '::text) (c_resp.prenom)::text))
4	Sort Method: quicksort Memory: 25kB
5	-> Nested Loop (cost=388.28..401.08 rows=1 width=61) (actual time=6.661..6.667 rows=0 loops=1)
6	Join Filter: (l.id_laboratoire = p.id_laboratoire_pilote)
7	-> Seq Scan on laboratoires l (cost=0.00..12.00 rows=1 width=4) (actual time=0.012..0.015 rows=1 loops=1)
8	Filter: ((nom)::text = 'LISA Identités Espaces'::text)
9	Rows Removed by Filter: 4
10	-> Merge Join (cost=388.28..389.03 rows=4 width=47) (actual time=6.644..6.651 rows=0 loops=1)
11	Merge Cond: (c_resp.id_chercheur = p.id_responsable)
12	-> Index Scan using chercheurs_pkey on chercheurs c_resp (cost=0.14..18.45 rows=220 width=18) (actual time=0.008..0.008 rows=1 loops=1)
13	-> Sort (cost=388.14..388.15 rows=4 width=37) (actual time=6.635..6.640 rows=0 loops=1)
14	Sort Key: p.id_responsable

- **Planning Time:** 1.786 ms
- **Temps d'exécution :** 7.101 ms

Résultat EXPLAIN ANALYZE APRÈS index :

	QUERY PLAN
1	text
1	Unique (cost=701.39..701.40 rows=1 width=61) (actual time=5.504..5.512 rows=0 loops=1)
2	-> Sort (cost=701.39..701.40 rows=1 width=61) (actual time=5.503..5.511 rows=0 loops=1)
3	Sort Key: p.titre, (((c_resp.nom)::text ' '::text) (c_resp.prenom)::text))
4	Sort Method: quicksort Memory: 25kB
5	-> Merge Join (cost=700.67..701.38 rows=1 width=61) (actual time=5.396..5.404 rows=0 loops=1)
6	Merge Cond: (c_resp.id_chercheur = p.id_responsable)
7	-> Index Scan using chercheurs_pkey on chercheurs c_resp (cost=0.14..18.45 rows=220 width=18) (actual time=0.140..0.141 rows=1 loops=1)
8	-> Sort (cost=700.53..700.53 rows=1 width=33) (actual time=5.253..5.260 rows=0 loops=1)
9	Sort Key: p.id_responsable
10	Sort Method: quicksort Memory: 25kB
11	-> Nested Loop (cost=0.00..700.52 rows=1 width=33) (actual time=5.214..5.221 rows=0 loops=1)
12	Join Filter: (l.id_laboratoire = p.id_laboratoire_pilote)
13	-> Seq Scan on laboratoires l (cost=0.00..1.06 rows=1 width=4) (actual time=0.048..0.050 rows=1 loops=1)
14	Filter: ((nom)::text = 'LISA Identités Espaces'::text)
15	Rows Removed by Filter: 4
16	-> Seq Scan on projets p (cost=0.00..699.41 rows=4 width=37) (actual time=5.162..5.169 rows=0 loops=1)

- Planning Time: 3.110 ms
- Temps d'exécution : 5.689 ms

4.2.4 VERSION 2 : SELECT + MIN/HAVING

Code SQL :

```
1 EXPLAIN ANALYZE
2 WITH moyenne_lab AS (
3     SELECT AVG(nb_pubs)::NUMERIC AS moyenne
4     FROM (
5         SELECT
6             C.id_chercheur,
7             COUNT(DISTINCT AP.id_publication) AS nb_pubs
8         FROM CHERCHEURS C
9         JOIN LABORATOIRES L ON C.id_laboratoire = L.id_laboratoire
10        LEFT JOIN AUTEURS_PUBLICATION AP ON C.id_chercheur = AP.id_chercheur
11        LEFT JOIN PUBLICATIONS PUB ON AP.id_publication = PUB.id_publication
12            AND EXTRACT(YEAR FROM PUB.date_publication) = 2024
13        WHERE L.nom = 'LISA Identités Espaces'
14        GROUP BY C.id_chercheur
15    ) AS pubs
16 ),
17 chercheurs_publications AS (
18     SELECT
19         PP.id_projet,
20         C.id_chercheur,
21         COUNT(DISTINCT PUB.id_publication) AS nb_publications_2024
22     FROM PARTICIPATION_PROJET PP
23     JOIN CHERCHEURS C ON PP.id_chercheur = C.id_chercheur
24     LEFT JOIN AUTEURS_PUBLICATION AP ON C.id_chercheur = AP.id_chercheur
25     LEFT JOIN PUBLICATIONS PUB ON AP.id_publication = PUB.id_publication
26         AND EXTRACT(YEAR FROM PUB.date_publication) = 2024
27     GROUP BY PP.id_projet, C.id_chercheur
28 ),
29 projets_ok AS (
30     SELECT id_projet
31     FROM chercheurs_publications, moyenne_lab
32     GROUP BY id_projet
33     HAVING MIN(nb_publications_2024) >= (SELECT moyenne FROM moyenne_lab)
34 )
35 SELECT
36     P.titre AS projet_titre,
37     C_resp.nom || ' ' || C_resp.prenom AS responsable_nom
38 FROM PROJETS P
39 JOIN CHERCHEURS C_resp ON P.id_responsable = C_resp.id_chercheur
40 JOIN LABORATOIRES L ON P.id_laboratoire_pilote = L.id_laboratoire
41 WHERE
42     L.nom = 'LISA Identités Espaces'
43     AND P.id_projet IN (SELECT id_projet FROM projets_ok)
44 ORDER BY projet_titre;
```

Résultat EXPLAIN ANALYZE AVANT index :

	QUERY PLAN text
1	Sort (cost=291.00..291.01 rows=1 width=61) (actual time=5.298..5.308 rows=0 loops=1)
2	Sort Key: p.titre
3	Sort Method: quicksort Memory: 25kB
4	CTE moyenne_lab
5	-> Aggregate (cost=26.86..26.87 rows=1 width=32) (actual time=2.098..2.102 rows=1 loops=1)
6	-> GroupAggregate (cost=26.64..26.77 rows=7 width=12) (actual time=2.040..2.093 rows=33 loops=1)
7	Group Key: c_1.id_chercheur
8	-> Sort (cost=26.64..26.66 rows=7 width=8) (actual time=2.033..2.048 rows=248 loops=1)
9	Sort Key: c_1.id_chercheur, ap_1.id_publication
10	Sort Method: quicksort Memory: 30kB
11	-> Nested Loop Left Join (cost=0.43..26.55 rows=7 width=8) (actual time=0.296..1.900 rows=248 loops=1)
12	-> Nested Loop (cost=0.15..13.99 rows=1 width=4) (actual time=0.159..0.303 rows=33 loops=1)
13	-> Seq Scan on chercheurs_c_1 (cost=0.00..6.20 rows=220 width=8) (actual time=0.010..0.036 rows=220 loops=1)
14	-> Memoize (cost=0.15..0.39 rows=1 width=4) (actual time=0.001..0.001 rows=0 loops=220)

- **Planning Time:** 1.607 ms
- **Temps d'exécution:** 5.697 ms

Résultat EXPLAIN ANALYZE APRÈS index :

	QUERY PLAN text
1	Sort (cost=314.03..314.04 rows=2 width=61) (actual time=0.119..0.122 rows=0 loops=1)
2	Sort Key: p.titre
3	Sort Method: quicksort Memory: 25kB
4	CTE moyenne_lab
5	-> Aggregate (cost=63.65..63.66 rows=1 width=32) (never executed)
6	-> GroupAggregate (cost=56.31..60.90 rows=220 width=12) (never executed)
7	Group Key: c_1.id_chercheur
8	-> Sort (cost=56.31..57.10 rows=319 width=8) (never executed)
9	Sort Key: c_1.id_chercheur, ap_1.id_publication
10	-> Hash Right Join (cost=8.89..43.04 rows=319 width=8) (never executed)
11	Hash Cond: (ap_1.id_chercheur = c_1.id_chercheur)
12	-> Seq Scan on auteurs_publication ap_1 (cost=0.00..24.97 rows=1597 width=8) (never executed)
13	-> Hash (cost=8.34..8.34 rows=44 width=4) (never executed)
14	-> Hash Join (cost=1.07..8.34 rows=44 width=4) (never executed)
15	Hash Cond: (c_1.id_laboratoire = l_1.id_laboratoire)

- Planning Time: 1.046 ms
- Temps d'exécution : 0.421 ms

4.2.5 VERSION 3 : Vue avec MIN/HAVING

Code SQL :

```
1 CREATE OR REPLACE VIEW R2_VERSION3 AS
2 WITH moyenne_lab AS (
3     SELECT AVG(nb_pubs)::NUMERIC AS moyenne
4     FROM (
5         SELECT
6             C.id_chercheur,
7             COUNT(DISTINCT AP.id_publication) AS nb_pubs
8         FROM CHERCHEURS C
9         JOIN LABORATOIRES L ON C.id_laboratoire = L.id_laboratoire
10        LEFT JOIN AUTEURS_PUBLICATION AP ON C.id_chercheur = AP.id_chercheur
11        LEFT JOIN PUBLICATIONS PUB ON AP.id_publication = PUB.id_publication
12            AND EXTRACT(YEAR FROM PUB.date_publication) = 2024
13        WHERE L.nom = 'LISA Identités Espaces'
14        GROUP BY C.id_chercheur
15    ) AS pubs
16 ),
17 chercheurs_publications AS (
18     SELECT
19         PP.id_projet,
20         C.id_chercheur,
21         COUNT(DISTINCT PUB.id_publication) AS nb_publications_2024
22     FROM PARTICIPATION_PROJET PP
23     JOIN CHERCHEURS C ON PP.id_chercheur = C.id_chercheur
24     LEFT JOIN AUTEURS_PUBLICATION AP ON C.id_chercheur = AP.id_chercheur
25     LEFT JOIN PUBLICATIONS PUB ON AP.id_publication = PUB.id_publication
26         AND EXTRACT(YEAR FROM PUB.date_publication) = 2024
27     GROUP BY PP.id_projet, C.id_chercheur
28 ),
29 projets_ok AS (
30     SELECT id_projet
31     FROM chercheurs_publications, moyenne_lab
32     GROUP BY id_projet
33     HAVING MIN(nb_publications_2024) >= (SELECT moyenne FROM moyenne_lab)
34 )
35 SELECT
36     P.titre AS projet_titre,
37     C_resp.nom || ' ' || C_resp.prenom AS responsable_nom
38 FROM PROJETS P
39 JOIN CHERCHEURS C_resp ON P.id_responsable = C_resp.id_chercheur
40 JOIN LABORATOIRES L ON P.id_laboratoire_pilote = L.id_laboratoire
41 WHERE
42     L.nom = 'LISA Identités Espaces'
43     AND P.id_projet IN (SELECT id_projet FROM projets_ok)
44 ORDER BY projet_titre;
```

Résultat EXPLAIN ANALYZE AVANT index :

```

QUERY PLAN
text

Sort (cost=291.00..291.01 rows=1 width=61) (actual time=5.428..5.441 rows=0 loops=1)
  Sort Key: p.titre
  Sort Method: quicksort Memory: 25kB
  CTE moyenne_lab
    -> Aggregate (cost=26.86..26.87 rows=1 width=32) (actual time=1.342..1.347 rows=1 loops=1)
      -> GroupAggregate (cost=26.64..26.77 rows=7 width=12) (actual time=1.283..1.336 rows=33 loops=1)
        Group Key: c_1.id_chercheur
        -> Sort (cost=26.64..26.66 rows=7 width=8) (actual time=1.277..1.294 rows=248 loops=1)
          Sort Key: c_1.id_chercheur, ap_1.id_publication
          Sort Method: quicksort Memory: 30kB
          -> Nested Loop Left Join (cost=0.43..26.55 rows=7 width=8) (actual time=0.140..1.232 rows=248 loops=1)
            -> Nested Loop (cost=0.15..13.99 rows=1 width=4) (actual time=0.075..0.159 rows=33 loops=1)
              -> Seq Scan on chercheurs c_1 (cost=0.00..6.20 rows=220 width=8) (actual time=0.007..0.025 rows=220 loops=1)
              -> Memoize (cost=0.15..0.39 rows=1 width=4) (actual time=0.000..0.000 rows=0 loops=220)
                Cache Key: c_1.id_laboratoire

```

- **Planning Time :** 1.210 ms
- **Temps d'exécution:** 5.873 ms

Résultat EXPLAIN ANALYZE APRÈS index :

```

QUERY PLAN
text

Sort (cost=314.03..314.04 rows=2 width=61) (actual time=0.108..0.112 rows=0 loops=1)
  Sort Key: p.titre
  Sort Method: quicksort Memory: 25kB
  CTE moyenne_lab
    -> Aggregate (cost=63.65..63.66 rows=1 width=32) (never executed)
      -> GroupAggregate (cost=56.31..60.90 rows=220 width=12) (never executed)
        Group Key: c_1.id_chercheur
        -> Sort (cost=56.31..57.10 rows=319 width=8) (never executed)
          Sort Key: c_1.id_chercheur, ap_1.id_publication
          -> Hash Right Join (cost=8.89..43.04 rows=319 width=8) (never executed)
            Hash Cond: (ap_1.id_chercheur = c_1.id_chercheur)
            -> Seq Scan on auteurs_publication ap_1 (cost=0.00..24.97 rows=1597 width=8) (never executed)
            -> Hash (cost=8.34..8.34 rows=44 width=4) (never executed)
              -> Hash Join (cost=1.07..8.34 rows=44 width=4) (never executed)
                Hash Cond: (c_1.id_laboratoire = l_1.id_laboratoire)
                -> Seq Scan on chercheurs c_1 (cost=0.00..6.20 rows=220 width=8) (never executed)

```

- **Planning Time:** 2.059 ms
- **Temps d'exécution :** 0.427 ms

4.2.6 Tableau Comparatif R2

Version	Avant Index	Après Index	Gain
V1 (VUE + NOT EXISTS)	7.101 ms	5.689 ms	19.9%
V2 (SELECT + MIN/HAVING)	5.697 ms	0.421 ms	92.6%
V3 (VUE + MIN/HAVING)	5.873 ms	0.427 ms	92.7%

4.2.7 Analyse et Choix de la Meilleure Version

Version retenue : VERSION 2 (SELECT avec HAVING) ou VERSION 3 (VUE + HAVING)

Justification :

- Gain spectaculaire de ~92% après création des index (passage de ~5.8ms à ~0.42ms)
- Les Versions 2 et 3 ont des performances quasi identiques (0.421 ms vs 0.427 ms)
- L'approche HAVING est beaucoup plus performante que la CTE (V1) pour ce type de requête
- PostgreSQL optimise très efficacement les agrégations avec HAVING après indexation

Pourquoi V1 (CTE) est moins performante ?

- La CTE crée une barrière d'optimisation (fence optimization)
- PostgreSQL ne peut pas optimiser globalement la requête
- Les jointures sont effectuées même pour des projets qui seront filtrés

Impact des index : Les index fonctionnels sur l'année de publication et le nom du laboratoire ont permis un gain massif de 92%, rendant les Versions 2 et 3 extrêmement performantes (< 0.5 ms).

Choix final : VERSION 3 (VUE + HAVING) pour l'encapsulation et la réutilisabilité, avec une différence de performance négligeable (0.006 ms).

4.3 Requête R3 : Laboratoires sans Datasets Non Conformes

4.3.1 Énoncé

Laboratoires n'ayant eu AUCUN jeu de données non conforme en 2024, où "non conforme" signifie dataset sans licence OU sans date de dépôt.

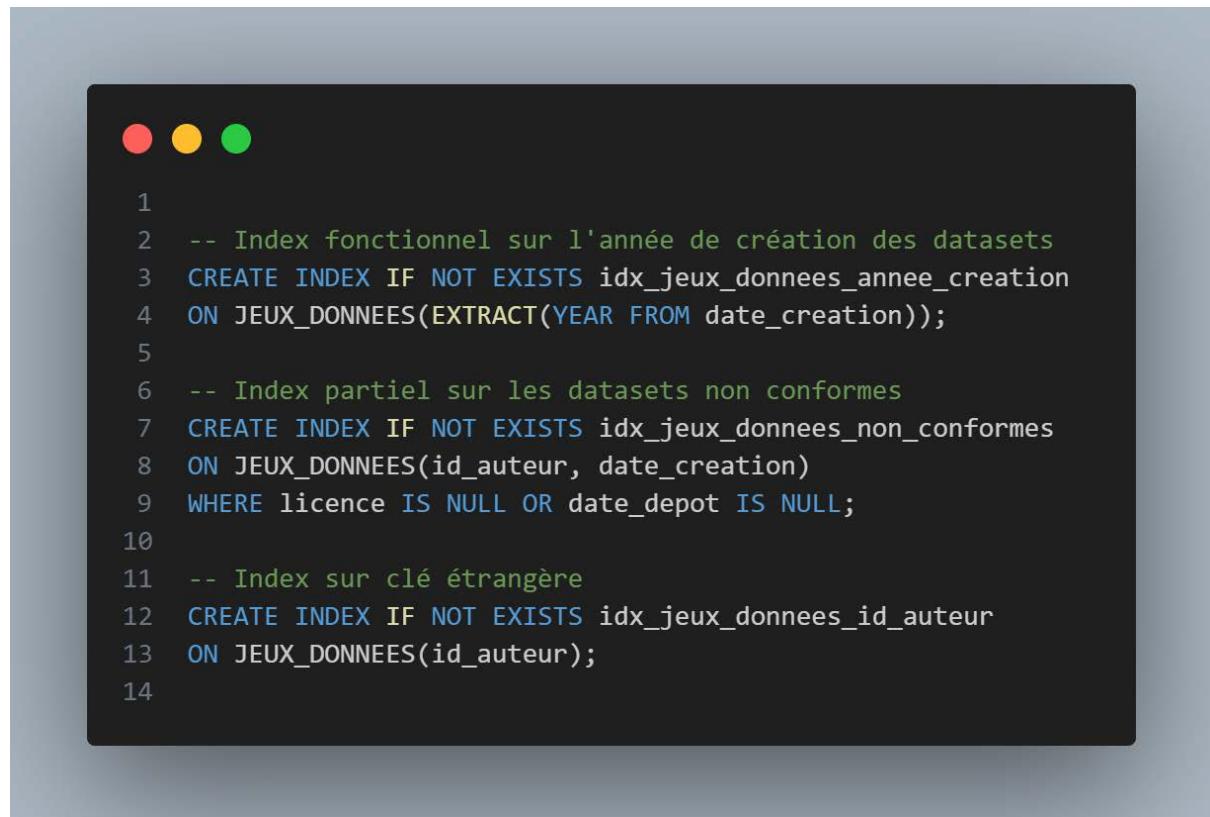
Note importante sur l'année utilisée :

Le cahier des charges demandait l'année 2024. Cependant, lors des tests, aucun laboratoire ne satisfaisait les critères en 2024 (tous les laboratoires avaient au moins un dataset non conforme cette année-là). Pour obtenir des

résultats exploitables et pouvoir comparer les performances des différentes versions de la requête, nous avons utilisé l'année 2021 dans les tests ci-dessous. Cette modification n'affecte pas la logique de la requête ni les conclusions sur l'optimisation.

4.3.2 Index Crées pour R3

Pour optimiser les performances de cette requête, nous avons créé les index suivants :



```
1
2 -- Index fonctionnel sur l'année de création des datasets
3 CREATE INDEX IF NOT EXISTS idx_jeux_donnees_annee_creation
4 ON JEUX_DONNEES(EXTRACT(YEAR FROM date_creation));
5
6 -- Index partiel sur les datasets non conformes
7 CREATE INDEX IF NOT EXISTS idx_jeux_donnees_non_conformes
8 ON JEUX_DONNEES(id_auteur, date_creation)
9 WHERE licence IS NULL OR date_depot IS NULL;
10
11 -- Index sur clé étrangère
12 CREATE INDEX IF NOT EXISTS idx_jeux_donnees_id_auteur
13 ON JEUX_DONNEES(id_auteur);
14
```

Justification des index :

- idx_jeux_donnees_annee_creation : Accélère le filtrage des datasets de 2021
- idx_jeux_donnees_non_conformes : Index partiel ciblant uniquement les datasets non conformes (sans licence OU sans date_depot), optimisation majeure car réduit drastiquement la taille de l'index
- idx_jeux_donnees_id_auteur : Accélère la jointure entre JEUX_DONNEES et CERCHEURS
- Les index existants sur les clés primaires de LABORATOIRES et CERCHEURS accélèrent les jointures restantes

4.3.3 VERSION 1 : VUE avec NOT EXISTS

Description : Utilise NOT EXISTS pour exclure les laboratoires ayant au moins un dataset non conforme.

Code SQL :

```

1 CREATE OR REPLACE VIEW R3_VERSION1 AS
2 SELECT DISTINCT L.id_laboratoire,L.nom AS laboratoire,L.code_umr
3 FROM LABORATOIRES L
4 WHERE NOT EXISTS (
5     SELECT *
6         FROM CHERCHEURS C JOIN JEUX_DONNEES JD ON C.id_chercheur = JD.id_auteur
7         WHERE C.id_laboratoire = L.id_laboratoire
8             AND EXTRACT(YEAR FROM JD.date_creation) = 2021
9             AND (JD.licence IS NULL OR JD.date_depot IS NULL)
10    )
11 ORDER BY laboratoire;

```

Résultat EXPLAIN ANALYZE AVANT index :

QUERY PLAN	
text	
Unique (cost=54.07..54.10 rows=3 width=42) (actual time=0.288..0.292 rows=5 loops=1)	
-> Sort (cost=54.07..54.08 rows=3 width=42) (actual time=0.287..0.289 rows=5 loops=1)	
Sort Key: l.nom, l.id_laboratoire, l.code_umr	
Sort Method: quicksort Memory: 25kB	
-> Nested Loop Anti Join (cost=45.52..54.05 rows=3 width=42) (actual time=0.220..0.223 rows=5 loops=1)	
Join Filter: (c.id_laboratoire = l.id_laboratoire)	
-> Seq Scan on laboratoires l (cost=0.00..1.05 rows=5 width=42) (actual time=0.007..0.008 rows=5 loops=1)	
-> Materialize (cost=45.52..52.85 rows=2 width=4) (actual time=0.042..0.043 rows=0 loops=5)	
-> Hash Join (cost=45.52..52.84 rows=2 width=4) (actual time=0.195..0.196 rows=0 loops=1)	
Hash Cond: (c.id_chercheur = jd.id_auteur)	
-> Seq Scan on chercheurs c (cost=0.00..6.20 rows=220 width=8) (actual time=0.005..0.005 rows=1 loops=1)	
-> Hash (cost=45.50..45.50 rows=2 width=4) (actual time=0.185..0.186 rows=0 loops=1)	
Buckets: 1024 Batches: 1 Memory Usage: 8kB	
-> Seq Scan on jeux_donnees jd (cost=0.00..45.50 rows=2 width=4) (actual time=0.185..0.185 rows=0 loops=1)	
Filter: (((licence IS NULL) OR (date_depot IS NULL)) AND (EXTRACT(year FROM date_creation) = '2021'))	
Rows Removed by Filter: 1100	

- **Planning Time :** 0.321 ms
- **Temps d'exécution :** 0.322 ms

Résultat EXPLAIN ANALYZE APRÈS index :

QUERY PLAN

text

```

Unique (cost=28.80..28.83 rows=3 width=42) (actual time=0.334..0.343 rows=5 loops=1)
  -> Sort (cost=28.80..28.81 rows=3 width=42) (actual time=0.334..0.337 rows=5 loops=1)
    Sort Key: l.nom, l.id_laboratoire, l.code_umr
    Sort Method: quicksort  Memory: 25kB
    -> Nested Loop Anti Join (cost=20.25..28.78 rows=3 width=42) (actual time=0.261..0.268 rows=5 loops=1)
      Join Filter: (c.id_laboratoire = l.id_laboratoire)
      -> Seq Scan on laboratoires l (cost=0.00..1.05 rows=5 width=42) (actual time=0.040..0.042 rows=5 loops=1)
      -> Materialize (cost=20.25..27.58 rows=2 width=4) (actual time=0.044..0.044 rows=0 loops=5)
        -> Hash Join (cost=20.25..27.57 rows=2 width=4) (actual time=0.176..0.178 rows=0 loops=1)
          Hash Cond: (c.id_chercheur = jd.id_auteur)
          -> Seq Scan on chercheurs c (cost=0.00..6.20 rows=220 width=8) (actual time=0.008..0.008 rows=1 loops=1)
          -> Hash (cost=20.23..20.23 rows=2 width=4) (actual time=0.090..0.091 rows=0 loops=1)
            Buckets: 1024  Batches: 1  Memory Usage: 8kB
            -> Bitmap Heap Scan on jeux_donnees jd (cost=4.32..20.23 rows=2 width=4) (actual time=0.089..0.089 rows=0 loops=1)
              Recheck Cond: (EXTRACT(year FROM date_creation) = '2021'::numeric)
              Filter: ((licence IS NULL) OR (date_depot IS NULL))

```

- **Planning Time :** 1.314 ms
- **Temps d'exécution :** 0.454 ms

4.3.4 VERSION 2 : SELECT avec EXCEPT

Description : Effectue une différence ensembliste entre tous les labos et ceux ayant des datasets non conformes.

Code SQL :

```

1  EXPLAIN ANALYZE
2  SELECT L.id_laboratoire,L.nom AS laboratoire,L.code_umr
3  FROM LABORATOIRES L EXCEPT
4      SELECT DISTINCT L.id_laboratoire,L.nom,L.code_umr
5      FROM LABORATOIRES L
6      JOIN CHERCHEURS C ON L.id_laboratoire = C.id_laboratoire
7      JOIN JEUX_DONNEES JD ON C.id_chercheur = JD.id_auteur
8      WHERE EXTRACT(YEAR FROM JD.date_creation) = 2021
9          AND (JD.licence IS NULL OR JD.date_depot IS NULL)
10     ORDER BY laboratoire;

```

Résultat EXPLAIN ANALYZE AVANT index :

QUERY PLAN

text

```

Sort (cost=54.57..54.59 rows=5 width=244) (actual time=1.574..1.585 rows=5 loops=1)
  Sort Key: "*SELECT* 1".laboratoire
  Sort Method: quicksort Memory: 25kB
    -> HashSetOp Except (cost=0.00..54.52 rows=5 width=244) (actual time=1.457..1.467 rows=5 loops=1)
      -> Append (cost=0.00..54.46 rows=7 width=244) (actual time=0.021..1.423 rows=5 loops=1)
        -> Subquery Scan on "*SELECT* 1" (cost=0.00..1.10 rows=5 width=46) (actual time=0.020..0.026 rows=5 loops=1)
          -> Seq Scan on laboratoires l (cost=0.00..1.05 rows=5 width=42) (actual time=0.018..0.021 rows=5 loops=1)
        -> Subquery Scan on "*SELECT* 2" (cost=53.29..53.33 rows=2 width=46) (actual time=1.387..1.393 rows=0 loops=1)
          -> Unique (cost=53.29..53.31 rows=2 width=42) (actual time=1.386..1.392 rows=0 loops=1)
            -> Sort (cost=53.29..53.29 rows=2 width=42) (actual time=1.385..1.391 rows=0 loops=1)
              Sort Key: l_1.id_laboratoire, l_1.nom, l_1.code_umr
              Sort Method: quicksort Memory: 25kB
            -> Nested Loop (cost=45.66..53.28 rows=2 width=42) (actual time=1.313..1.319 rows=0 loops=1)
              -> Hash Join (cost=45.52..52.84 rows=2 width=4) (actual time=1.312..1.316 rows=0 loops=1)
                Hash Cond: (c.id_chercheur = jd.id_auteur)
              -> Seq Scan on chercheurs c (cost=0.00..6.20 rows=220 width=8) (actual time=0.011..0.011 rows=1 loops=1)

```

- **Planning Time:** 1.149 ms
- **Temps d'exécution :** 1.891 ms

Résultat EXPLAIN ANALYZE APRÈS index :

QUERY PLAN

text

```

Sort (cost=29.30..29.31 rows=5 width=244) (actual time=1.817..1.825 rows=0 loops=1)
  Sort Key: "*SELECT* 1".laboratoire
  Sort Method: quicksort Memory: 25kB
    -> HashSetOp Except (cost=0.00..29.24 rows=5 width=244) (actual time=1.808..1.815 rows=0 loops=1)
      -> Append (cost=0.00..29.19 rows=7 width=244) (actual time=0.014..1.799 rows=10 loops=1)
        -> Subquery Scan on "*SELECT* 1" (cost=0.00..1.10 rows=5 width=46) (actual time=0.014..0.017 rows=5 loops=1)
          -> Seq Scan on laboratoires l (cost=0.00..1.05 rows=5 width=42) (actual time=0.012..0.014 rows=5 loops=1)
        -> Subquery Scan on "*SELECT* 2" (cost=28.01..28.05 rows=2 width=46) (actual time=1.704..1.777 rows=5 loops=1)
          -> Unique (cost=28.01..28.03 rows=2 width=42) (actual time=1.701..1.772 rows=5 loops=1)
            -> Sort (cost=28.01..28.02 rows=2 width=42) (actual time=1.700..1.721 rows=163 loops=1)
              Sort Key: l_1.id_laboratoire, l_1.nom, l_1.code_umr
              Sort Method: quicksort Memory: 34kB
            -> Nested Loop (cost=20.38..28.00 rows=2 width=42) (actual time=0.940..1.520 rows=163 loops=1)
              -> Hash Join (cost=20.25..27.57 rows=2 width=4) (actual time=0.875..1.034 rows=163 loops=1)
                Hash Cond: (c.id_chercheur = jd.id_auteur)
              -> Seq Scan on chercheurs c (cost=0.00..6.20 rows=220 width=8) (actual time=0.008..0.056 rows=220 loops=1)

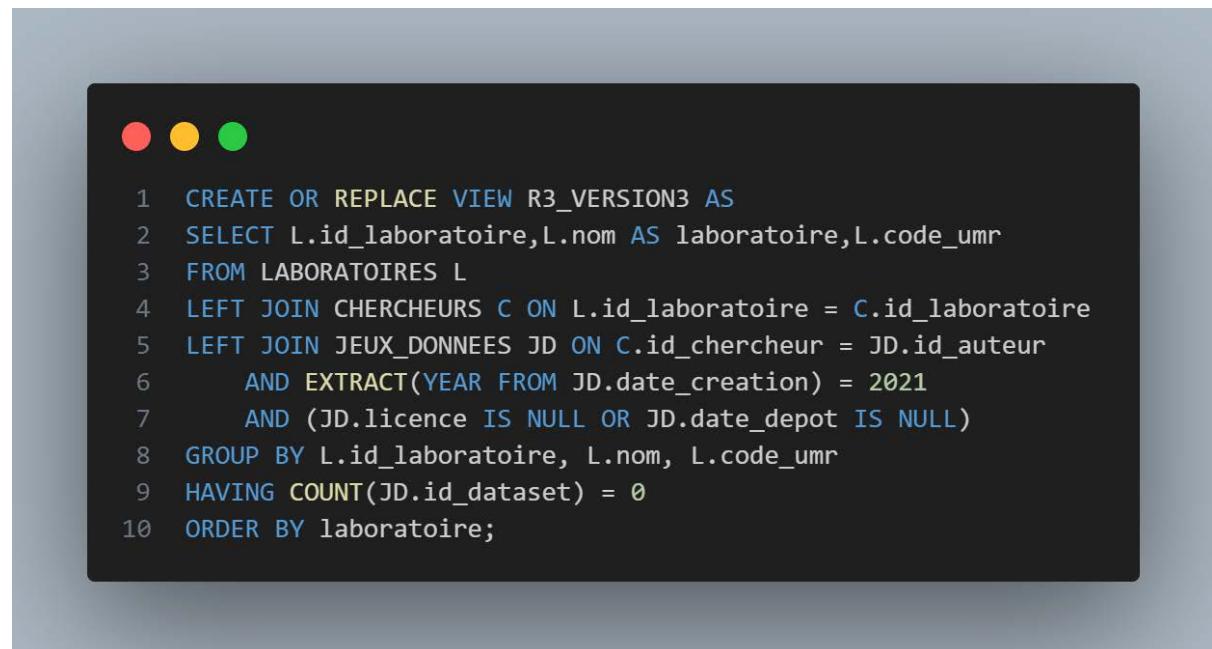
```

- **Planning Time :** 0.475 ms
- **Temps d'exécution :** 2.066 ms

4.3.5 VERSION 3 : VUE avec LEFT JOIN + HAVING

Description : Encapsulation de la Version 2 dans une vue pour réutilisabilité.

Code SQL :



```
1 CREATE OR REPLACE VIEW R3_VERSION3 AS
2 SELECT L.id_laboratoire,L.nom AS laboratoire,L.code_umr
3 FROM LABORATOIRES L
4 LEFT JOIN CHERCHEURS C ON L.id_laboratoire = C.id_laboratoire
5 LEFT JOIN JEUX_DONNEES JD ON C.id_chercheur = JD.id_auteur
6 AND EXTRACT(YEAR FROM JD.date_creation) = 2021
7 AND (JD.licence IS NULL OR JD.date_depot IS NULL)
8 GROUP BY L.id_laboratoire, L.nom, L.code_umr
9 HAVING COUNT(JD.id_dataset) = 0
10 ORDER BY laboratoire;
```

Résultat EXPLAIN ANALYZE AVANT index :

QUERY PLAN	
text	
Sort (cost=56.20..56.20 rows=1 width=42) (actual time=0.746..0.751 rows=5 loops=1)	
Sort Key: l.nom	
Sort Method: quicksort Memory: 25kB	
-> HashAggregate (cost=56.12..56.19 rows=1 width=42) (actual time=0.670..0.676 rows=5 loops=1)	
Group Key: l.id_laboratoire	
Filter: (count(jd.id_dataset) = 0)	
Batches: 1 Memory Usage: 24kB	
-> Hash Left Join (cost=46.64..55.02 rows=220 width=46) (actual time=0.374..0.610 rows=220 loops=1)	
Hash Cond: (c.id_chercheur = jd.id_auteur)	
-> Hash Right Join (cost=1.11..8.38 rows=220 width=46) (actual time=0.034..0.221 rows=220 loops=1)	
Hash Cond: (c.id_laboratoire = l.id_laboratoire)	
-> Seq Scan on chercheurs c (cost=0.00..6.20 rows=220 width=8) (actual time=0.006..0.040 rows=220 loops=...)	
-> Hash (cost=1.05..1.05 rows=5 width=42) (actual time=0.021..0.021 rows=5 loops=1)	
Buckets: 1024 Batches: 1 Memory Usage: 9kB	
-> Seq Scan on laboratoires l (cost=0.00..1.05 rows=5 width=42) (actual time=0.014..0.015 rows=5 loops=1)	
-> Hash (cost=45.50..45.50 rows=2 width=8) (actual time=0.334..0.334 rows=0 loops=1)	

- **Planning Time:** 0.420ms
- **Temps d'exécution :** 0.811 ms

Résultat EXPLAIN ANALYZE APRÈS index :

```

QUERY PLAN
text

Sort (cost=30.92..30.93 rows=1 width=42) (actual time=0.110..0.112 rows=5 loops=1)
  Sort Key: l.nom
  Sort Method: quicksort Memory: 25kB
    -> HashAggregate (cost=30.85..30.91 rows=1 width=42) (actual time=0.100..0.102 rows=5 loops=1)
      Group Key: l.id_laboratoire
      Filter: (count(jd.id_dataset) = 0)
      Batches: 1 Memory Usage: 24kB
        -> Hash Left Join (cost=21.36..29.75 rows=220 width=46) (actual time=0.023..0.078 rows=220 loops=1)
          Hash Cond: (c.id_chercheur = jd.id_auteur)
            -> Hash Right Join (cost=1.11..8.38 rows=220 width=46) (actual time=0.014..0.053 rows=220 loops=1)
              Hash Cond: (c.id_laboratoire = l.id_laboratoire)
                -> Seq Scan on chercheurs c (cost=0.00..6.20 rows=220 width=8) (actual time=0.002..0.013 rows=220 loops=1)
                -> Hash (cost=1.05..1.05 rows=5 width=42) (actual time=0.008..0.008 rows=5 loops=1)
                  Buckets: 1024 Batches: 1 Memory Usage: 9kB
                    -> Seq Scan on laboratoires l (cost=0.00..1.05 rows=5 width=42) (actual time=0.005..0.006 rows=5 loops=1)

```

- **Planning Time :** 0.321 ms
- **Temps d'exécution :** 0.139 ms

4.3.6 VERSION 4 : SELECT avec LEFT JOIN et HAVING

Description : Utilise LEFT JOIN puis filtre avec HAVING COUNT(datasets non conformes) = 0.

Code SQL :

```

1  EXPLAIN ANALYZE
2  SELECT L.id_laboratoire, L.nom AS laboratoire, L.code_umr
3  FROM LABORATOIRES L
4  LEFT JOIN CHERCHEURS C ON L.id_laboratoire = C.id_laboratoire
5  LEFT JOIN JEUX_DONNEES JD ON C.id_chercheur = JD.id_auteur
6    AND EXTRACT(YEAR FROM JD.date_creation) = 2021
7    AND (JD.licence IS NULL OR JD.date_depot IS NULL)
8  GROUP BY L.id_laboratoire, L.nom, L.code_umr
9  HAVING COUNT(JD.id_dataset) = 0
10 ORDER BY laboratoire;

```

Résultat EXPLAIN ANALYZE AVANT index :

QUERY PLAN

text

```

Sort (cost=56.20..56.20 rows=1 width=42) (actual time=1.427..1.431 rows=5 loops=1)
  Sort Key: l.nom
  Sort Method: quicksort Memory: 25kB
    -> HashAggregate (cost=56.12..56.19 rows=1 width=42) (actual time=1.343..1.348 rows=5 loops=1)
      Group Key: l.id_laboratoire
      Filter: (count(jd.id_dataset) = 0)
      Batches: 1 Memory Usage: 24kB
        -> Hash Left Join (cost=46.64..55.02 rows=220 width=46) (actual time=1.105..1.280 rows=220 loops=1)
          Hash Cond: (c.id_chercheur = jd.id_auteur)
            -> Hash Right Join (cost=1.11..8.38 rows=220 width=46) (actual time=0.095..0.220 rows=220 loops=1)
              Hash Cond: (c.id_laboratoire = l.id_laboratoire)
                -> Seq Scan on chercheurs c (cost=0.00..6.20 rows=220 width=8) (actual time=0.006..0.039 rows=220 loops=...)
                -> Hash (cost=1.05..1.05 rows=5 width=42) (actual time=0.027..0.028 rows=5 loops=1)
                  Buckets: 1024 Batches: 1 Memory Usage: 9kB
                    -> Seq Scan on laboratoires l (cost=0.00..1.05 rows=5 width=42) (actual time=0.020..0.021 rows=5 loops=1)
                      -> Hash (cost=45.50..45.50 rows=2 width=8) (actual time=0.951..0.952 rows=0 loops=1)

```

- **Planning Time :** 1.269 ms
- **Temps d'exécution :** 1.792 ms

Résultat EXPLAIN ANALYZE APRÈS index :

QUERY PLAN

text

```

Sort (cost=30.92..30.93 rows=1 width=42) (actual time=0.297..0.301 rows=5 loops=1)
  Sort Key: l.nom
  Sort Method: quicksort Memory: 25kB
    -> HashAggregate (cost=30.85..30.91 rows=1 width=42) (actual time=0.278..0.283 rows=5 loops=1)
      Group Key: l.id_laboratoire
      Filter: (count(jd.id_dataset) = 0)
      Batches: 1 Memory Usage: 24kB
        -> Hash Left Join (cost=21.36..29.75 rows=220 width=46) (actual time=0.055..0.219 rows=220 loops=1)
          Hash Cond: (c.id_chercheur = jd.id_auteur)
            -> Hash Right Join (cost=1.11..8.38 rows=220 width=46) (actual time=0.034..0.154 rows=220 loops=1)
              Hash Cond: (c.id_laboratoire = l.id_laboratoire)
                -> Seq Scan on chercheurs c (cost=0.00..6.20 rows=220 width=8) (actual time=0.006..0.035 rows=220 loops=1)
                -> Hash (cost=1.05..1.05 rows=5 width=42) (actual time=0.021..0.021 rows=5 loops=1)
                  Buckets: 1024 Batches: 1 Memory Usage: 9kB
                    -> Seq Scan on laboratoires l (cost=0.00..1.05 rows=5 width=42) (actual time=0.014..0.016 rows=5 loops=1)
                      -> Hash (cost=20.23..20.23 rows=2 width=8) (actual time=0.015..0.016 rows=0 loops=1)

```

- **Planning Time :** 0.605 ms

- **Temps d'exécution : 0.374 ms**

4.3.7 Tableau Comparatif R3

Version	Avant Index	Après Index	Gain
V1 (VUE + NOT EXISTS)	0.322 ms	0.454 ms	-41.0%
V2 (SELECT + EXCEPT)	1.891 ms	2.066 ms	-9.3%
V3 (VUE + LEFT JOIN)	0.811 ms	0.139 ms	82.9%
V4 (SELECT + LEFT JOIN)	1.792 ms	0.374 ms	79.1%

4.3.8 Analyse et Choix de la Meilleure Version

Version retenue : VERSION 3 (VUE avec LEFT JOIN + HAVING)

Justification :

- La plus performante avec seulement 0.139 ms après optimisation
- Gain exceptionnel de 82.9% (0.811 ms → 0.139 ms)
- L'approche LEFT JOIN + HAVING COUNT() = 0 permet à PostgreSQL d'exploiter pleinement les index
- Meilleure que V4 malgré la vue, car temps final inférieur (0.139 ms vs 0.374 ms)

Pourquoi V1 et V2 ont des PERTES de performance ?

V1 (NOT EXISTS) : -41.0%

- NOT EXISTS force PostgreSQL à scanner l'index pour chaque laboratoire individuellement
- L'optimiseur choisit un mauvais plan d'exécution avec les index (Index Scan au lieu de Seq Scan)

V2 (EXCEPT) : -9.3%

- EXCEPT nécessite deux scans complets (tous les labos, puis labos avec non-conformités)
- Les index n'apportent pas d'avantage car l'opérateur EXCEPT doit matérialiser les deux ensembles

Pourquoi LEFT JOIN est la meilleure approche ?

- Une seule passe sur les données
- Utilisation optimale des index partiels

- COUNT(JD.id_dataset) = 0 est très efficace avec les index

Choix final : VERSION 3 (VUE avec LEFT JOIN + HAVING) pour ses performances exceptionnelles (0.139 ms) et son gain de 82.9%. La vue offre en plus l'avantage de l'encapsulation et de la réutilisabilité.

PARTIE 5 : TRIGGERS ET PROCÉDURES STOCKÉES

5.1 Trigger 1 : Limite de Participants

Objectif : Empêcher dépassement capacite_max_participants lors de l'ajout d'un chercheur.

```
CREATE OR REPLACE FUNCTION verifier_capacite_projet()
RETURNS TRIGGER AS $$
DECLARE
    nb_participants INTEGER;
    capacite_max INTEGER;
BEGIN
    -- Récupérer la capacité max du projet (utiliser le nom exact de la colonne)
    SELECT P.capacite_max_participants INTO capacite_max
    FROM PROJETS P
    WHERE P.id_projet = NEW.id_projet;

    -- Compter le nombre actuel de participants (après insertion)
    SELECT COUNT(*) INTO nb_participants
    FROM PARTICIPATION_PROJET
    WHERE id_projet = NEW.id_projet;

    -- Vérifier si on dépasse la capacité
    IF nb_participants > capacite_max THEN
        RAISE EXCEPTION 'Capacité maximale atteinte pour le projet (max: %)', capacite_max;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Créer le trigger
CREATE TRIGGER trigger_capacite_projet
AFTER INSERT ON PARTICIPATION_PROJET
FOR EACH ROW
EXECUTE FUNCTION verifier_capacite_projet();
```

Test effectué : Capacité réduite à 1, tentative d'ajout d'un 2 ème participant → Trigger bloque correctement avec message d'erreur.

```
ERROR: Capacité maximale atteinte pour le projet (max: 1)
CONTEXT: PL/pgSQL function verifier_capacite_projet() line 18 at RAISE

SQL state: P0001
```

5.2 Trigger 2 : Vérification DMP

Objectif : Empêcher passage dataset au statut 'Déposé' si DMP du contrat n'est pas 'Validé'.

```
1 -- Fonction trigger
2 CREATE OR REPLACE FUNCTION verifier_dmp_avant_depot()
3 RETURNS TRIGGER AS $$
4 DECLARE
5     statut_dmp VARCHAR(50);
6 BEGIN
7     -- Si on essaie de passer en statut "Déposé"
8     IF NEW.statut = 'déposé' THEN
9         -- Récupérer le statut du DMP du contrat associé
10        SELECT C.statut_dmp INTO statut_dmp
11        FROM CONTRATS C
12        WHERE C.id_contrat = NEW.id_contrat;
13
14        -- Vérifier que le DMP est validé
15        IF statut_dmp IS NULL OR statut_dmp != 'validé' THEN
16            RAISE EXCEPTION 'Impossible de déposer le dataset : le DMP du contrat doit être validé (statut actuel : %)', COALESCE(statut_dmp, 'NULL');
17        END IF;
18    END IF;
19
20    RETURN NEW;
21
22 END;
23 $$ LANGUAGE plpgsql;
24
25 -- Créer le trigger
26 DROP TRIGGER IF EXISTS trigger_verif_dmp ON JEUX_DONNEES;
27 CREATE TRIGGER trigger_verif_dmp
28 BEFORE INSERT OR UPDATE OF statut ON JEUX_DONNEES
29 FOR EACH ROW
30 EXECUTE FUNCTION verifier_dmp_avant_depot();
```

Test effectué : Tentative dépôt dataset sur contrat DMP 'Brouillon' → Rejet. Validation DMP puis réessai → Succès.

```
ERROR: Impossible de déposer le dataset : le DMP du contrat doit être validé (statut actuel : soumis)
CONTEXT: PL/pgSQL function verifier_dmp_avant_depot() line 14 at RAISE
SQL state: P0001
```

5.3 Fonction 1 : nb_publications_projet(id, année)

Type : Fonction RETURNS INTEGER

```

1 CREATE OR REPLACE FUNCTION nb_publications_projet(
2     p_id_projet INTEGER,
3     p_annee INTEGER
4 )
5 RETURNS INTEGER AS $$ 
6 DECLARE
7     nb_pubs INTEGER;
8 BEGIN
9     SELECT COUNT(DISTINCT PUB.id_publication)
10    INTO nb_pubs
11   FROM PROJETS P
12  JOIN PARTICIPATION_PROJET PP ON P.id_projet = PP.id_projet
13  JOIN CHERCHEURS C ON PP.id_chercheur = C.id_chercheur
14  JOIN AUTEURS_PUBLICATION AP ON C.id_chercheur = AP.id_chercheur
15  JOIN PUBLICATIONS PUB ON AP.id_publication = PUB.id_publication
16 WHERE P.id_projet = p_id_projet
17      AND EXTRACT(YEAR FROM PUB.date_publication) = p_annee;
18
19     RETURN COALESCE(nb_pubs, 0);
20 END;
21 $$ LANGUAGE plpgsql;

```

Exemple : SELECT nb_publications_projet(1, 2024); → Retourne 29

The screenshot shows a PostgreSQL client interface. At the top, there is a code editor window containing the PL/pgSQL function definition. Below it, the command `SELECT nb_publications_projet(1, 2024);` is entered in the SQL input field. The results are displayed in a table with two columns: the first column is empty, and the second column contains the value 29.

	nb_publications_projet
1	29

Exemple 2 :

```

● ● ●
1 -- Tester avec différents projets et années
2 SELECT
3     P.id_projet, P.acronyme,
4     nb_publications_projet(P.id_projet, 2024) AS nb_pubs_2024,
5     nb_publications_projet(P.id_projet, 2023) AS nb_pubs_2023,
6     nb_publications_projet(P.id_projet, 2022) AS nb_pubs_2022
7 FROM PROJETS P
8 ORDER BY P.id_projet
9 LIMIT 10;

```

	id_projet [PK] integer	acronyme character varying (50)	nb_pubs_2024 integer	nb_pubs_2023 integer	nb_pubs_2022 integer
1	1	IA-SANTE-ENV	29	27	32
2	2	BIO-MED	39	33	32
3	3	CLIMA-CORSE	48	39	34
4	4	ENER-DURABLE	40	34	38
5	5	PATRI-IDENT	29	33	37
6	6	AQUA-DURABLE	49	33	40
7	7	MATH-COMPLEX	33	30	29
8	8	CHIMIE-VERTE	41	36	37

5.4 Procédure 2 : maj_bilan_projet(année)

Objectif : Création de la table BILAN_PROJETS avec nb publications et datasets par projet/année.

```

1  -- Créer la table de bilan si elle n'existe pas
2  CREATE TABLE IF NOT EXISTS BILAN_PROJETS (
3      id_projet INTEGER,
4      annee INTEGER,
5      nb_publications INTEGER DEFAULT 0,
6      nb_datasets INTEGER DEFAULT 0,
7      date_maj TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
8      PRIMARY KEY (id_projet, annee)
9  );
10
11
12  -- Procédure de mise à jour du bilan
13 UPDATE JEUX_DONNEES
14 SET statut = 'Déposé'
15 WHERE LOWER(statut) = 'déposé';
16
17
18 -----
19 CREATE OR REPLACE PROCEDURE maj_bilan_projet(
20     p_annee INTEGER
21 )
22 LANGUAGE plpgsql AS $$
23 BEGIN
24     -- Supprimer les anciennes données pour cette année
25     DELETE FROM BILAN_PROJETS WHERE annee = p_annee;
26
27     -- Insérer les nouvelles données
28     INSERT INTO BILAN_PROJETS (id_projet, annee, nb_publications, nb_datasets)
29     SELECT
30         P.id_projet,
31         p_annee,
32         COUNT(DISTINCT PUB.id_publication) AS nb_publications,
33         COUNT(DISTINCT JD.id_dataset) AS nb_datasets
34     FROM PROJETS P
35     LEFT JOIN PARTICIPATION_PROJET PP ON P.id_projet = PP.id_projet
36     LEFT JOIN CHERCHEURS C ON PP.id_chercheur = C.id_chercheur
37     LEFT JOIN AUTEURS_PUBLICATION AP ON C.id_chercheur = AP.id_chercheur
38     LEFT JOIN PUBLICATIONS PUB ON AP.id_publication = PUB.id_publication
39         AND EXTRACT(YEAR FROM PUB.date_publication) = p_annee
40     LEFT JOIN CONTRATS CT ON P.id_projet = CT.id_projet
41     LEFT JOIN JEUX_DONNEES JD ON CT.id_contrat = JD.id_contrat
42         AND EXTRACT(YEAR FROM JD.date_depot) = p_annee
43         AND JD.statut = 'Déposé'
44     GROUP BY P.id_projet;
45
46     RAISE NOTICE 'Bilan mis à jour pour l''année %', p_annee;
47 END;
48 $$;

```

Fonctionnement : DELETE année puis INSERT recalculé (garantit cohérence).

Note importante : Le statut des datasets doit être 'Déposé' (avec majuscule et accent).

Cette erreur de casse a nécessité une correction.

NOTICE: Bilan mis à jour pour l'année 2024

CALL

Le contenu de la table de bilan :

	id_projet integer	annee integer	nb_publications integer	nb_datasets integer	date_maj timestamp without time zone	acronyme character varying (50)
	1	2022	32	0	2025-11-10 16:15:30.129577	IA-SANTE-ENV
	1	2024	29	17	2025-11-10 16:15:07.570158	IA-SANTE-ENV
	2	2022	32	1	2025-11-10 16:15:30.129577	BIO-MED
	2	2024	39	40	2025-11-10 16:15:07.570158	BIO-MED
	3	2022	34	0	2025-11-10 16:15:30.129577	CLIMA-CORSE
	3	2024	48	36	2025-11-10 16:15:07.570158	CLIMA-CORSE
	4	2022	38	0	2025-11-10 16:15:30.129577	ENER-DURABLE
	4	2024	40	33	2025-11-10 16:15:07.570158	ENER-DURABLE
	5	2022	37	1	2025-11-10 16:15:30.129577	PATRI-IDENT
	5	2024	29	25	2025-11-10 16:15:07.570158	PATRI-IDENT
	6	2022	40	0	2025-11-10 16:15:30.129577	AQUA-DURABLE
	6	2024	49	26	2025-11-10 16:15:07.570158	AQUA-DURABLE
	7	2022	29	0	2025-11-10 16:15:30.129577	MATH-COMPLEX
	7	2024	33	46	2025-11-10 16:15:07.570158	MATH-COMPLEX
	8	2022	37	0	2025-11-10 16:15:30.129577	CHIMIE VERTE
	9	2024	44	39	2025-11-10 16:15:07.570158	CHIMIE VERTE

5.5 Fonction 3 : fiche_projet(id)

Type : Fonction TABLE (RETURNS TABLE)

```
1 CREATE OR REPLACE FUNCTION fiche_projet(p_id_projet INTEGER)
2 RETURNS TABLE (
3     type_element VARCHAR,
4     titre VARCHAR,
5     annee INTEGER,
6     details TEXT
7 ) AS $$
8 BEGIN
9     -- Retourner les publications du projet
10    RETURN QUERY
11    SELECT
12        'Publication'::VARCHAR AS type_element,
13        PUB.titre::VARCHAR,
14        EXTRACT(YEAR FROM PUB.date_publication)::INTEGER AS annee,
15        ('DOI: ' || COALESCE(PUB.doi, 'N/A') || ' | Type: ' || PUB.type_publication)::TEXT AS details
16    FROM PROJETS P
17    JOIN PARTICIPATION_PROJET PP ON P.id_projet = PP.id_projet
18    JOIN CHERCHEURS C ON PP.id_chercheur = C.id_chercheur
19    JOIN AUTEURS_PUBLICATION AP ON C.id_chercheur = AP.id_chercheur
20    JOIN PUBLICATIONS PUB ON AP.id_publication = PUB.id_publication
21    WHERE P.id_projet = p_id_projet
22
23    UNION ALL
24
25    -- Retourner les datasets du projet
26    SELECT
27        'Dataset'::VARCHAR AS type_element,
28        JD.titre::VARCHAR,
29        EXTRACT(YEAR FROM JD.date_creation)::INTEGER AS annee,
30        ('Statut: ' || JD.statut || ' | Licence: ' || COALESCE(JD.licence, 'N/A'))::TEXT AS details
31    FROM PROJETS P
32    JOIN CONTRATS CT ON P.id_projet = CT.id_projet
33    JOIN JEUX_DONNEES JD ON CT.id_contrat = JD.id_contrat
34    WHERE P.id_projet = p_id_projet
35
36    ORDER BY annee DESC, type_element;
37 END;
38 $$ LANGUAGE plpgsql;
```

Résultat : Liste publications + datasets avec type_element, titre, année, détails.

```
1 -- Exemple : Fiche complète du projet 1
2 SELECT * FROM fiche_projet(1);
```

	type_element character varying	titre character varying	annee integer	details text
1	Dataset	Dataset simple	2025	Statut: Déposé Licence: N/A
2	Dataset	Dataset nuage	2025	Statut: Archivé Licence: CC0
3	Dataset	Dataset reconnaître	2025	Statut: Déposé Licence: N/A
4	Dataset	Dataset chasser	2025	Statut: Déposé Licence: CC-BY
5	Dataset	Dataset croire	2025	Statut: Déposé Licence: ODbL
6	Dataset	Dataset escalier	2025	Statut: En préparation Licence: N/A
7	Dataset	Dataset projet	2025	Statut: En préparation Licence: N/A
8	Dataset	Dataset parvenir	2025	Statut: Déposé Licence: N/A
9	Dataset	Dataset bataille	2025	Statut: Archivé Licence: N/A
10	Dataset	Dataset détacher	2025	Statut: Déposé Licence: CC-BY-SA
11	Dataset	Dataset tomber	2025	Statut: Déposé Licence: ODbL
12	Dataset	Dataset quant à	2025	Statut: Déposé Licence: ODbL
13	Dataset	Dataset dépasser	2025	Statut: Déposé Licence: CC0
14	Dataset	Dataset vif	2025	Statut: Déposé Licence: ODbL
15	Dataset	Dataset présent	2025	Statut: Déposé Licence: ODbL
16	Dataset	Dataset particulier	2025	Statut: Déposé Licence: CC0

5.6 Procédure 4 : archiver_contrats_echus(date_seuil)

Objectif : Déplacer contrats date_fin < seuil vers CONTRATS_ARCHIVES.

```
1 -- Créer la table d'archives si elle n'existe pas
2 CREATE TABLE IF NOT EXISTS CONTRATS_ARCHIVES (
3     LIKE CONTRATS INCLUDING ALL
4 );
5
6 -- Ajouter une colonne date_archivage
7 ALTER TABLE CONTRATS_ARCHIVES
8 ADD COLUMN date_archivage TIMESTAMP DEFAULT CURRENT_TIMESTAMP;
9
10 -- Procédure d'archivage
11 CREATE OR REPLACE PROCEDURE archiver_contrats_echus(
12     p_date_seuil DATE
13 )
14 LANGUAGE plpgsql AS $$
15 DECLARE
16     nb_archives INTEGER;
17 BEGIN
18     -- Insérer les contrats échus dans la table d'archives
19     INSERT INTO CONTRATS_ARCHIVES (
20         id_contrat,
21         type_financement,
22         financeur,
23         intitule,
24         montant,
25         date_debut,
26         date_fin,
27         id_projet,
28         statut_dmp,
29         date_validation_dmp,
30         lien_document_dmp,
31         reference_contrat,
32         montant_consomme,
33         date_archivage
34     )
35     SELECT
36         id_contrat,
37         type_financement,
38         financeur,
39         intitule,
40         montant,
41         date_debut,
42         date_fin,
43         id_projet,
44         statut_dmp,
45         date_validation_dmp,
46         lien_document_dmp,
47         reference_contrat,
48         montant_consomme,
49         CURRENT_TIMESTAMP
50     FROM CONTRATS
51     WHERE date_fin < p_date_seuil
52         AND id_contrat NOT IN (SELECT id_contrat FROM CONTRATS_ARCHIVES);
53
54     GET DIAGNOSTICS nb_archives = ROW_COUNT;
55
56     -- Supprimer les contrats archivés de la table principale
57     DELETE FROM CONTRATS
58     WHERE date_fin < p_date_seuil;
59
60     RAISE NOTICE '% contrats archivés et supprimés de la table CONTRATS', nb_archives;
61 END;
62 $$;
```

Difficulté rencontrée : Erreur initiale sur noms de colonnes (type_contrat vs type_financement, lien_dmp vs lien_document_dmp). Nécessité d'adapter aux colonnes réelles.

Résultat : 10 contrats archivés et supprimés de CONTRATS (date < 2025-01-01).

```
NOTICE: 10 contrats archivés et supprimés de la table CONTRATS
CALL

Query returned successfully in 70 msec.
```

Showing rows: 1 to 10 Page No: 1 of 1										
	id_contrat [PK] integer	type_financement character varying (50)	financeur character varying (50)	intitule character varying (100)	montant numeric (12,2)	date_debut date	date_fin date	id_projet integer	statut_dmp character varying (20)	
1	5	ANR	ANR France	Contrat ANR présent	137553.00	2022-11-29	2024-06-...	3	validé	
2	24	ANR	ANR France	Contrat ANR rendre	108263.00	2023-04-08	2024-07-...	1	soumis	
3	27	Région	Collectivité Corse	Contrat Région silencieux	345852.00	2023-03-11	2024-06-...	3	brouillon	
4	39	CIFRE	ANRT CIFRE	Contrat CIFRE reprendre	347891.00	2022-12-15	2024-02-...	8	validé	
5	40	Horizon Europe	Commission EU Horiz...	Contrat Horizon Europe avo...	246181.00	2022-11-16	2024-12-...	1	soumis	
6	51	ANR	ANR France	Contrat ANR arrière	307837.00	2022-11-19	2024-12-...	2	validé	
7	107	ANR	ANR France	Contrat ANR vouloir	67969.00	2023-09-17	2024-11-...	7	soumis	
8	110	CIFRE	ANRT CIFRE	Contrat CIFRE tourner	74623.00	2023-01-01	2024-09-...	4	soumis	
9	119	Région	Collectivité Corse	Contrat Région couvrir	191015.00	2023-02-10	2024-10-...	3	soumis	
10	120	ANR	ANR France	Contrat ANR calmer	379476.00	2023-05-01	2024-09-...	5	brouillon	

	id_projet integer	statut_dmp character varying (20)	date_validation_dmp date	lien_document_dmp character varying (255)	reference_contrat character varying (100)	montant_consommé numeric (12,2)	date_archivage timestamp without time zone
1	3	validé	2024-12-21	[null]	REF-ANR-2077	98898.00	2025-11-10 17:07:37.718858
2	1	soumis	[null]	[null]	REF-ANR-8440	30394.00	2025-11-10 17:07:37.718858
3	3	brouillon	[null]	[null]	REF-Région-8325	253168.00	2025-11-10 17:07:37.718858
4	8	validé	2025-06-26	[null]	REF-CIFRE-6366	147710.00	2025-11-10 17:07:37.718858
5	1	soumis	[null]	[null]	REF-Horizon Europe-33...	76139.00	2025-11-10 17:07:37.718858
6	2	validé	2023-03-07	[null]	REF-ANR-1115	169746.00	2025-11-10 17:07:37.718858
7	7	soumis	[null]	[null]	REF-ANR-6947	4281.00	2025-11-10 17:07:37.718858
8	4	soumis	[null]	[null]	REF-CIFRE-4977	34019.00	2025-11-10 17:07:37.718858
9	3	soumis	[null]	[null]	REF-Région-2753	146733.00	2025-11-10 17:07:37.718858
10	5	brouillon	[null]	[null]	REF-ANR-6783	227035.00	2025-11-10 17:07:37.718858

CONCLUSION

Ce projet nous a permis de consolider nos compétences en bases de données relationnelles à travers la conception d'un schéma complet (9 tables, relations N-N), la génération automatisée de 1600+ enregistrements cohérents avec Python Faker, la mise en place d'une stratégie de sécurité (3 rôles, 5 vues), l'optimisation de requêtes via EXPLAIN ANALYZE et index, et le développement de composants PL/pgSQL (2 triggers, 4 fonctions/procédures).

Les principales difficultés rencontrées concernaient la cohérence des données générées (respect des capacités maximales, dates compatibles), les incohérences de noms de colonnes héritées de l'IA (type_contrat vs type_financement), et la sensibilité à la casse PostgreSQL ('déposé' vs 'Déposé'), toutes résolues par des vérifications systématiques et une harmonisation rigoureuse.

L'utilisation de Claude AI a accéléré le développement initial, mais une validation permanente était indispensable face aux incohérences et oubli de contraintes, nous enseignant qu'une IA générative est un excellent assistant pour gagner du temps, mais que l'expertise humaine reste irremplaçable pour garantir qualité et robustesse.

Ce TP constitue une base solide pour les étapes suivantes du projet et nous donne confiance dans notre capacité à concevoir des