

Projet Migration

1. Création et alimentation des tables avec PostgreSQL

Table Département

The screenshot shows a PostgreSQL query interface with the following details:

- Query Tab:** Contains the SQL command: `SELECT * FROM departments;`
- Data Output Tab:** Displays the results of the query in a grid format.
- Table Structure:** The 'departments' table has two columns:
 - dept_no:** [PK] character (4)
 - dept_name:** character varying (40)
- Data Rows:** 9 rows are shown, labeled 1 through 9.

	dept_no	dept_name
1	d001	Marketing
2	d002	Finance
3	d003	Human Resources
4	d004	Production
5	d005	Development
6	d006	Quality Management
7	d007	Sales
8	d008	Research
9	d009	Customer Service
- Page Navigation:** Shows "Showing rows: 1 to 9" and "Page No: 1 of 1".

Table Employée

The screenshot shows a PostgreSQL query interface with the following details:

- Query Tab:** Contains the SQL command: `SELECT * FROM employees;`
- Data Output Tab:** Displays the results of the query in a grid format.
- Table Structure:** The 'employees' table has seven columns:
 - emp_no:** [PK] integer
 - birth_date:** date
 - first_name:** character varying (14)
 - last_name:** character varying (16)
 - gender:** gender
 - hire_date:** date
- Data Rows:** 7 rows are shown, labeled 1 through 7.

	emp_no	birth_date	first_name	last_name	gender	hire_date
1	10001	1953-09-02	Georgi	Facello	M	1986-06-26
2	10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
3	10003	1959-12-03	Parto	Bamford	M	1986-08-28
4	10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
5	10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
6	10006	1953-04-20	Anneke	Preusig	F	1989-06-02
7	10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
- Page Navigation:** Shows "Showing rows: 1 to 1000" and "Page No: 1".

Tables relation entre Employée et Département

Query Query History

```
1  SELECT * FROM dept_emp;
2
```

Data Output Messages Notifications

Showing rows: 1 to 100

	emp_no [PK] integer	dept_no [PK] character (4)	from_date date	to_date date
1	10001	d005	1986-06-26	9999-01-...
2	10002	d007	1996-08-03	9999-01-...
3	10003	d004	1995-12-03	9999-01-...
4	10004	d004	1986-12-01	9999-01-...
5	10005	d003	1989-09-12	9999-01-...
6	10006	d005	1990-08-05	9999-01-...
7	10007	d008	1989-02-10	9999-01-...
8	10008	d005	1998-03-11	2000-07-...
9	10009	d006	1985-02-18	9999-01-...

Query History

```
1 SELECT * FROM dept_manager;
2
```

Data Output Messages Notifications

Showing rows: 1 to 2

	dept_no [PK] character	emp_no integer	from_date date	to_date date
1	d001	110022	1985-01-01	1991-10-...
2	d001	110039	1991-10-01	9999-01-...
3	d002	110085	1985-01-01	1989-12-...
4	d002	110114	1989-12-17	9999-01-...
5	d003	110183	1985-01-01	1992-03-...
6	d003	110228	1992-03-21	9999-01-...
7	d004	110303	1985-01-01	1988-09-...
8	d004	110344	1988-09-09	1992-08-...

Table Salarie

Query History

```
1 SELECT * FROM salaries;
2
```

Data Output Messages Notifications

	emp_no [PK] integer	salary integer	from_date [PK] date	to_date date
1	10001	60117	1986-06-26	1987-06-...
2	10001	62102	1987-06-26	1988-06-...
3	10001	66074	1988-06-25	1989-06-...
4	10001	66596	1989-06-25	1990-06-...
5	10001	66961	1990-06-25	1991-06-...
6	10001	71046	1991-06-25	1992-06-...
7	10001	74333	1992-06-24	1993-06-...

Query Query History

```
1  SELECT * FROM titles;  
2
```

Data Output Messages Notifications

The screenshot shows a SQL query results interface. At the top, there are several icons for file operations (New, Open, Save, Print, Copy, Paste, Find, Delete, Import, Export, Refresh, Help) and a SQL button. To the right of the buttons, it says "Showing rows: 1 to 6". Below the toolbar is a table with the following data:

	emp_no [PK] integer	title [PK] character varying (50)	from_date [PK] date	to_date date
1	10001	Senior Engineer	1986-06-26	9999-01-...
2	10002	Staff	1996-08-03	9999-01-...
3	10003	Senior Engineer	1995-12-03	9999-01-...
4	10004	Engineer	1986-12-01	1995-12-...
5	10004	Senior Engineer	1995-12-01	9999-01-...
6	10005	Senior Staff	1996-09-12	9999-01-...

Table Title

The screenshot shows a database management system interface. On the left, there is a tree view of database objects: departments, dept_emp, dept_manag, employees, salaries, titles (which is selected), Trigger Functions, Types (1), gender, Views, and Subscriptions. The main area displays the results of a query on the titles table. The results grid has the following structure:

	emp_no [PK] integer	title [PK] character varying (50)	from_date [PK] date	to_date date
1	10001	Senior Engineer	1986-06-26	9999-01-...
2	10002	Staff	1996-08-03	9999-01-...
3	10003	Senior Engineer	1995-12-03	9999-01-...
4	10004	Engineer	1986-12-01	1995-12-...
5	10004	Senior Engineer	1995-12-01	9999-01-...
6	10005	Senior Staff	1996-09-12	9999-01-...

At the bottom, there is a toolbar with various icons (File, Edit, View, Insert, Delete, etc.) and some status information: CRLF, Ln 2, Col 47, FRA US, 12:00, and 11/12/2025.

2. Cout des jointures en SQL

2.1. Requête SQL de jointure employees + salaries + titles

The screenshot shows a MySQL Workbench interface with a query editor and a data output viewer.

Query Editor:

```

54 -- 2.1 - Requête SQL de jointure employees + salaries + titles
55
56     employe.emp_no,
57     employe.first_name,
58     employe.last_name,
59     salaire.salary,
60     salaire.from_date    AS salaire_date_debut,
61     salaire.to_date      AS salaire_date_fin,
62     titre.title,
63     titre.from_date     AS titre_date_debut,
64     titre.to_date       AS titre_date_fin
65   FROM employees AS employe
66 JOIN salaries   AS salaire ON salaire.emp_no = employe.emp_no
67 JOIN titles     AS titre  ON titre.emp_no   = employe.emp_no;
68
69

```

Data Output:

	emp_no	first_name	last_name	salary	salaire_date_debut	salaire_date_fin	title	titre_date_debut	titre_date_fin
1	10005	Kyoichi	Maliniak	78228	1989-09-12	1990-09-12	Staff	1989-09-12	1996-09-12
2	10005	Kyoichi	Maliniak	78228	1989-09-12	1990-09-12	Senior Staff	1996-09-12	9999-01-01
3	10005	Kyoichi	Maliniak	82621	1990-09-12	1991-09-12	Staff	1989-09-12	1996-09-12
4	10005	Kvoichi	Maliniak	82621	1990-09-12	1991-09-12	Senior Staff	1996-09-12	9999-01-01

Total rows: 4638507 Query complete 00:01:16.967 CRLF Ln 66, Col 9

2.2. Total rows: 4638507 || 16 secs 967 msec

2.3. Une vue matérialisée

The screenshot shows a MySQL Workbench interface with a query editor and a data output viewer.

Query Editor:

```

75
76
77 -- 2.3 - Une vue matérialisée sur cette jointure
78 CREATE MATERIALIZED VIEW vue_mat_employees_salaires_titres AS
79
80     employe.emp_no,
81     employe.first_name,
82     employe.last_name,
83     salaire.salary,
84     salaire.from_date    AS salaire_date_debut,
85     salaire.to_date      AS salaire_date_fin,
86     titre.title,
87     titre.from_date     AS titre_date_debut,
88     titre.to_date       AS titre_date_fin
89   FROM employees AS employe
90 JOIN salaries   AS salaire ON salaire.emp_no = employe.emp_no
91 JOIN titles     AS titre  ON titre.emp_no   = employe.emp_no;
92

```

Data Output:

SELECT 4638507

Query returned successfully in 20 secs 485 msec.

2.4. Temps d'accès à la vue matérialisée

```

-- 2.4 -- Temps d'accès à la vue matérialisée
EXPLAIN (ANALYZE, BUFFERS)
SELECT COUNT(*)
FROM vue_mat_employes_salaires_titres;

```

Showing rows: 1 to 12 | Page No: 1 of 1 | < << << >> >> >

QUERY PLAN

text

Finalize Aggregate (cost=73253.70..73253.71 rows=1 width=8) (actual time=513.807..532.074 rows=1.00 loops=1)

Buffers: shared hit=2868 read=45221

> Gather (cost=73253.49..73253.70 rows=2 width=8) (actual time=513.529..532.063 rows=3.00 loops=1)

Workers Planned: 2

Workers Launched: 2

Buffers: shared hit=2868 read=45221

> Partial Aggregate (cost=72253.49..72253.50 rows=1 width=8) (actual time=448.331..448.332 rows=1.00 loops=3)

Buffers: shared hit=2868 read=45221

> Parallel Seq Scan on vue_mat_employes_salaires_titres (cost=0.00..67420.59 rows=1933159 width=0) (actual time=1.347..288.454 rows=1546169.00 loops=3)

Planning Time: 0.101 ms

Execution Time: 532.128 ms

Total rows: 12 | Query complete 00:00:00.767 | CRLF | Ln 95, Col 45

Résultats chiffrés à reporter

Étape	Requête mesurée	Planning Time	Execution Time	Buffers (shared hit / read)
2.2	COUNT(*) sur la jointure employees ⚡ titles ⚡ salaries	14.418 ms	1460.525 ms (≈ 1.461 s)	7610 / 13116
2.4	COUNT(*) sur la vue matérialisée vue_mat_employes_salaires_titres	0.101 ms	532.128 ms (≈ 0.532 s)	2868 / 45221

3. Export de données PostgreSQL vers fichiers JSON

3.1. Sortie “1 document JSON par ligne” avec row_to_json

Query Query History

```

113 -- 3 Export de données PostgreSQL vers fichiers JSON
114 -- employees
115 SELECT row_to_json(ligne) AS document_json
116 FROM (SELECT * FROM employees ORDER BY emp_no) AS ligne;
117
118 -- departments
119 SELECT row_to_json(ligne) AS document_json
120 FROM (SELECT * FROM departments ORDER BY dept_no) AS ligne;
121
122 -- dept_emp
123 SELECT row_to_json(ligne) AS document_json
124 FROM (SELECT * FROM dept_emp ORDER BY emp_no, dept_no, from_date) AS ligne;
125
126 -- dept_manager
127 SELECT row_to_json(ligne) AS document_json
128 FROM (SELECT * FROM dept_manager ORDER BY dept_no, emp_no, from_date) AS ligne;
129
130 -- titles
131 SELECT row_to_json(ligne) AS document_json
132 FROM (SELECT * FROM titles ORDER BY emp_no, from_date, title) AS ligne;
133
134 -- salaries
135 SELECT row_to_json(ligne) AS document_json
136 FROM (SELECT * FROM salaries ORDER BY emp_no, from_date) AS ligne;
137

```

Data Output Messages Graph Visualiser Notifications

Total rows: 2844047 Query complete 00:00:04.864

Data Output Messages Graph Visualiser Notifications



	document_json json	🔒
1	{"emp_no":10001,"salary":60117,"from_date":"1986-06-26","to_date":"1987-06-2...	
2	{"emp_no":10001,"salary":62102,"from_date":"1987-06-26","to_date":"1988-06-2...	
3	{"emp_no":10001,"salary":66074,"from_date":"1988-06-25","to_date":"1989-06-2...	
4	{"emp_no":10001,"salary":66596,"from_date":"1989-06-25","to_date":"1990-06-2...	
5	{"emp_no":10001,"salary":66961,"from_date":"1990-06-25","to_date":"1991-06-2...	
6	{"emp_no":10001,"salary":71046,"from_date":"1991-06-25","to_date":"1992-06-2...	
7	{"emp_no":10001,"salary":74333,"from_date":"1992-06-24","to_date":"1993-06-2...	
8	{"emp_no":10001,"salary":75286,"from_date":"1993-06-24","to_date":"1994-06-2...	
9	{"emp_no":10001,"salary":75994,"from_date":"1994-06-24","to_date":"1995-06-2...	
1n	{"emp_no":10001,"salary":76884,"from_date":"1995-06-24","to_date":"1996-06-2...	

Total rows: 2844047 Query complete 00:00:04.864

3.2. — Agréger en “tableau JSON” avec json_agg

employees/postgres@PostgreSQL 18

No limit

Query History

```

139 --3.2 – Agréger en “tableau JSON” avec json_agg
140 -- employees
141 SELECT json_agg(row_to_json(ligne)) AS tableau_json
142 FROM (SELECT * FROM employees ORDER BY emp_no) AS ligne;
143
144 -- departments
145 SELECT json_agg(row_to_json(ligne)) AS tableau_json
146 FROM (SELECT * FROM departments ORDER BY dept_no) AS ligne;
147
148 -- dept_emp
149 SELECT json_agg(row_to_json(ligne)) AS tableau_json
150 FROM (SELECT * FROM dept_emp ORDER BY emp_no, dept_no, from_date) AS ligne;
151
152 -- dept_manager
153 SELECT json_agg(row_to_json(ligne)) AS tableau_json
154 FROM (SELECT * FROM dept_manager ORDER BY dept_no, emp_no, from_date) AS ligne;
155
156 -- titles
157 SELECT json_agg(row_to_json(ligne)) AS tableau_json
158 FROM (SELECT * FROM titles ORDER BY emp_no, from_date, title) AS ligne;
159
160 -- salaries
161 SELECT json_agg(row_to_json(ligne)) AS tableau_json
162 FROM (SELECT * FROM salaries ORDER BY emp_no, from_date) AS ligne;

```

employees/postgres@PostgreSQL 18

No limit

Query History

```

139 --3.2 – Agréger en “tableau JSON” avec json_agg
140 -- employees
141 SELECT json_agg(row_to_json(ligne)) AS tableau_json
142 FROM (SELECT * FROM employees ORDER BY emp_no) AS ligne;
143
144 -- departments
145 SELECT json_agg(row_to_json(ligne)) AS tableau_json
146 FROM (SELECT * FROM departments ORDER BY dept_no) AS ligne;
147
148 -- dept_emp
149 SELECT json_agg(row_to_json(ligne)) AS tableau_json
150 FROM (SELECT * FROM dept_emp ORDER BY emp_no, dept_no, from_date) AS ligne;
151
152 -- dept_manager
153 SELECT json_agg(row_to_json(ligne)) AS tableau_json

```

Data Output Messages Graph Visualiser Notifications

	tableau_json
1	[{"emp_no":10001,"birth_date":"1953-09-02","first_name":"Georgi","last_name":"Facello","gender":"M","hire_date":"1986-06-26"}, {"emp_no":10002,"birth_date":"1964-06-02","first_name":"Bezalel",...]

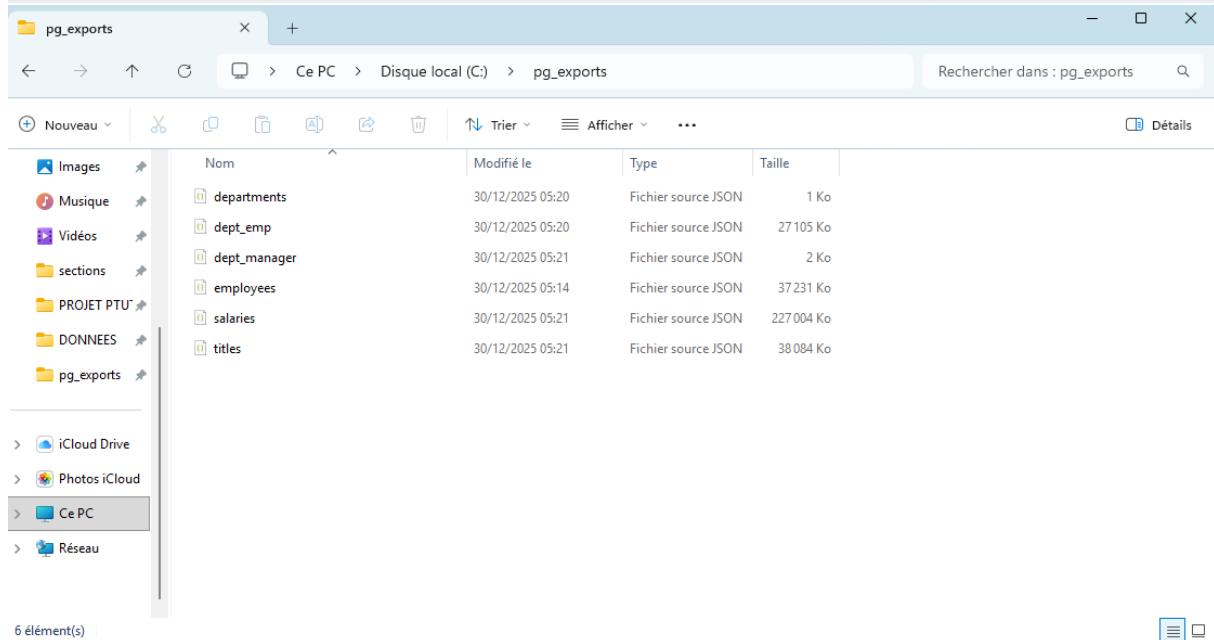
3.3. Export en JSON

```

166 -- 3.3 Export en JSON
167 COPY (
168   SELECT json_agg(row_to_json(ligne))
169   FROM (SELECT * FROM employees ORDER BY emp_no) AS ligne
170 ) TO 'C:/pg_exports/employees.json';
171
172 COPY (
173   SELECT json_agg(row_to_json(ligne))
174   FROM (SELECT * FROM departments ORDER BY dept_no) AS ligne
175 ) TO 'C:/pg_exports/departments.json';
176
177 COPY (
178   SELECT json_agg(row_to_json(ligne))
179   FROM (SELECT * FROM dept_emp ORDER BY emp_no, dept_no, from_date) AS ligne
180 ) TO 'C:/pg_exports/dept_emp.json';
181
182 COPY (
183   SELECT json_agg(row_to_json(ligne))
184   FROM (SELECT * FROM titles ORDER BY title_id) AS ligne
185 ) TO 'C:/pg_exports/titles.json';

```

Data Output Messages Graph Visualiser × Notifications



4. Import dans MongoDB des données des fichiers JSON

- Je lance mongod d'abord

4.1. Commandes mongoimport dans le CMD

```

mongosh mongoDB://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.8
PS C:\Users\ibodi> mongorestore --db employees --collection employees --file "C:\pg_exports\employees.json" --jsonArray --drop
connected to: mongoDB://localhost/
dropping: employees.employees
[#####] employees.employees 26.2MB/36.4MB (71.9%)
[#####] employees.employees 36.4MB/36.4MB (100.0%)
300024 document(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\ibodi> mongorestore --db departments --file "C:\pg_exports\departments.json" --jsonArray --drop
connected to: mongoDB://localhost/
dropping: employees.departments
9 document(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\ibodi> mongorestore --db dept_emp --file "C:\pg_exports\dept_emp.json" --jsonArray --drop
connected to: mongoDB://localhost/
dropping: employees.dept_emp
24 documents(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\ibodi> mongorestore --db titles --file "C:\pg_exports\titles.json" --jsonArray --drop
connected to: mongoDB://localhost/
dropping: employees.titles
331603 document(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\ibodi> mongorestore --db dept_manager --file "C:\pg_exports\dept_manager.json" --jsonArray --drop
connected to: mongoDB://localhost/
dropping: employees.dept_manager
31 document(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\ibodi> mongorestore --db salaries --file "C:\pg_exports\salaries.json" --jsonArray --drop
connected to: mongoDB://localhost/
dropping: employees.salaries
443308 document(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\ibodi> mongorestore --db titles --file "C:\pg_exports\titles.json" --jsonArray --drop
connected to: mongoDB://localhost/
dropping: employees.titles
[#####] employees.titles 25.1MB/37.2MB (67.3%)
[#####] employees.titles 37.2MB/37.2MB (100.0%)
2025-12-30T06:04:34.377+0100
443308 document(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\ibodi> mongorestore --db salaries --file "C:\pg_exports\salaries.json" --jsonArray --drop
connected to: mongoDB://localhost/
dropping: employees.salaries
2025-12-30T06:04:34.377+0100
[#####] employees.salaries 22.4MB/222MB (10.1%)
[#####] employees.salaries 36.1MB/222MB (16.3%)
2025-12-30T06:05:02.271+0100
[#####] employees.salaries 49.5MB/222MB (22.3%)
2025-12-30T06:05:05.272+0100
[#####] employees.salaries 63.6MB/222MB (28.7%)
2025-12-30T06:05:08.271+0100
[#####] employees.salaries 77.8MB/222MB (35.1%)
2025-12-30T06:05:11.271+0100
[#####] employees.salaries 102MB/222MB (46.1%)
2025-12-30T06:05:14.272+0100
[#####] employees.salaries 130MB/222MB (58.6%)
2025-12-30T06:05:17.271+0100
[#####] employees.salaries 152MB/222MB (68.6%)
2025-12-30T06:05:20.271+0100
[#####] employees.salaries 168MB/222MB (75.8%)
2025-12-30T06:05:23.271+0100
[#####] employees.salaries 192MB/222MB (86.8%)
2025-12-30T06:05:26.271+0100
[#####] employees.salaries 222MB/222MB (100.0%)
2025-12-30T06:05:26.279+0100
[#####] employees.salaries 222MB/222MB (100.0%)
2844047 document(s) imported successfully. 0 document(s) failed to import.

```

- Puis Je Vérifier que l'import est OK (avec mongosh)

```

mongosh mongoDB://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.8
PS C:\Users\ibodi> mongosh
Current Mongosh Log ID: 69535defda4643062cebea3
Connecting to: mongoDB://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.8
Using MongoDB: 8.2.0
Using Mongosh: 2.5.8
mongosh 2.5.10 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

The server generated these startup warnings when booting
2025-12-18T08:26:01.246+01:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
_____
test> use employees
switched to db employees
employees> show collections
departments
dept_emp
dept_manager
employees
salaries
titles
employees> db.employees.countDocuments()
300024
employees> db.departments.countDocuments()
9
employees> db.dept_emp.countDocuments()
331603
employees> db.dept_manager.countDocuments()
24
employees> db.titles.countDocuments()
443308
employees> db.salaries.countDocuments()
2844047
employees> db.employees.findOne()
{
  _id: ObjectId('69535ce0fa52c8f9a9fe8bfe'),
  emp_no: 10001,
  birth_date: '1953-09-02',
  first_name: 'Georgi',
  last_name: 'Facello',
  gender: 'M',
  hire_date: '1986-06-26'
}
employees>

```

5. Import dans MongoDB des données des fichiers JSON

- 5.0 — Je vais créer des index pour accélérer les \$lookup

```

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.8
Using MongoDB: 8.2.0
Using Mongosh: 2.5.8
mongosh 2.5.10 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/
-----
The server generated these startup warnings when booting
2025-12-18T08:26:01.246+01:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----
test> use employees
switched to db employees
employees> db.titles.createIndex({ emp_no: 1 })
emp_no_1
employees> db.salaries.createIndex({ emp_no: 1 })
emp_no_1
employees>

```

5.1. Jointure employees ↔ titles (1 \$lookup)

```

db.employees.aggregate([{$lookup: {from: "titles", localField: "emp_no", foreignField: "emp_no", as: "titles"} }])
[ {
  _id: ObjectId('69535ce0fa52c8f9a9fe8bfe'),
  emp_no: 10001,
  birth_date: '1953-09-02',
  first_name: 'Georgi',
  last_name: 'Facello',
  gender: 'M',
  hire_date: '1986-06-26',
  titles: [
    {
      _id: ObjectId('69535d5ecef988cbb7d5f02b'),
      emp_no: 10001,
      title: 'Senior Engineer',
      from_date: '1986-06-26',
      to_date: '9999-01-01'
    }
  ],
  _id: ObjectId('69535ce0fa52c8f9a9fe8bff'),
  emp_no: 10007,
  birth_date: '1957-05-23',
  first_name: 'Tzvetan',
  last_name: 'Zielinski',
  gender: 'F',
  hire_date: '1989-02-10',
  titles: [
    {
      _id: ObjectId('69535d5ecef988cbb7d5f024'),
      emp_no: 10007,
      title: 'Staff',
      from_date: '1989-02-10',
      to_date: '1996-02-11'
    },
    {
      _id: ObjectId('69535d5ecef988cbb7d5f025'),
      emp_no: 10007,
    }
  ]
},
...

```

5.2. Pipeline à 2 étapes : jointure employees ↔ titles ↔ salaries

```

employees> const pipeline_jointure_3 = [{$lookup: {from: "titles", localField: "emp_no", foreignField: "emp_no", as: "titles"}}, ... {$lookup: {from: "salaries", localField: "emp_no", foreignField: "emp_no", as: "salaries"}}]
employees> db.employees.aggregate([{$match: {emp_no: 10001}}, ...pipeline_jointure_3]).toArray()
[ {
    _id: ObjectId('69535ce0fa52c8f9a9fe8bfe'),
    emp_no: 10001,
    birth_date: '1953-09-02',
    first_name: 'Georgi',
    last_name: 'Facello',
    gender: 'M',
    hire_date: '1986-06-26',
    titles: [
        {
            _id: ObjectId('69535d5ecef988ccb7d5f02b'),
            emp_no: 10001,
            title: 'Senior Engineer',
            from_date: '1986-06-26',
            to_date: '9999-01-01'
        }
    ],
    salaries: [
        {
            _id: ObjectId('69535d75229d94d11c125a0f'),
            emp_no: 10001,
            salary: 60117,
            from_date: '1986-06-26',
            to_date: '9999-01-01'
        }
    ]
},
...

```

5.3. Mesurer le temps d'exécution de la jointure (MongoDB)

```

mongosh mongodb://127.0.0.1:27017
employees> db.employees .explain("executionStats") .aggregate(pipeline_jointure_3, { allowDiskUse: true })
{
  explainVersion: '2',
  queryPlanner: {
    namespace: 'employees.employees',
    parsedQuery: {},
    indexFilterSet: false,
    queryHash: 'CDC0091B',
    planCacheShapeHash: 'CDC0091B',
    planCacheKey: 'BB9BAF92',
    optimizationTimeMillis: 1,
    optimizedPipeline: true,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      queryPlan: {
        stage: 'EQ_LOOKUP',
        planNodeId: 3,
        foreignCollection: 'employees.salaries',
        localField: 'emp_no',
        foreignField: 'emp_no',
        asField: 'salaries',
        strategy: 'IndexedLoopJoin',
        indexName: 'emp_no_1',
        indexKeyPattern: { emp_no: 1 },
        scanDirection: 'forward',
        inputStage: {
          stage: 'EQ_LOOKUP',
          planNodeId: 2,
          foreignCollection: 'employees.titles',
          localField: 'emp_no',
          foreignField: 'emp_no',
          asField: 'titles',
          strategy: 'IndexedLoopJoin',
          indexName: 'emp_no_1',
          indexKeyPattern: { emp_no: 1 },
          scanDirection: 'forward',
          inputStage: {
            stage: 'COLLSCAN',
            planNodeId: 1,
            filter: {}
          }
        }
      }
    }
  }
}
.....

```

5.4. Dénormalisation + \$project (supprimer doublons emp_no et _id)

```

employees> let pipeline_denormalisation = [
...   ...pipeline_jointure_3, {$project: { _id: "$emp_no", birth_date: 1, first_name: 1, last_name: 1,
...     gender: 1, hire_date: 1,
...     titles: {$map: {input: "$titles", as: "t", in: {title: "$$t.title", from_date: "$$t.from_date",
...       to_date: "$$t.to_date"}},},
...     salaries: {$map: {input: "$salaries", as: "s", in: {salary: "$$s.salary",
...       from_date: "$$s.from_date", to_date: "$$s.to_date"}}}}},
... ]
employees> db.employees.aggregate([{$match: { emp_no: 10001 } }, ...pipeline_denormalisation], { allowDiskUse: true }).toArray()
[
  {
    birth_date: '1953-09-02',
    first_name: 'Georgi',
    last_name: 'Facello',
    gender: 'M',
    hire_date: '1986-06-26',
    _id: 10001,
    titles: [
      {
        title: 'Senior Engineer',
        from_date: '1986-06-26',
        to_date: '9999-01-01'
      }
    ],
    salaries: [
      { salary: 60117, from_date: '1986-06-26', to_date: '1987-06-26' },
      { salary: 75994, from_date: '1994-06-24', to_date: '1995-06-24' },
      { salary: 76884, from_date: '1995-06-24', to_date: '1996-06-23' },
      { salary: 80013, from_date: '1996-06-23', to_date: '1997-06-23' },
      { salary: 81025, from_date: '1997-06-23', to_date: '1998-06-23' },
      { salary: 81097, from_date: '1998-06-23', to_date: '1999-06-23' },
      { salary: 62102, from_date: '1987-06-26', to_date: '1988-06-25' },
      { salary: 84917, from_date: '1999-06-23', to_date: '2000-06-22' },
      { salary: 85097, from_date: '2001-06-22', to_date: '2002-06-22' },
      { salary: 85112, from_date: '2000-06-22', to_date: '2001-06-22' },
      ...
    ]
  }
]

```

5.5. Sauvegarder le résultat dans une nouvelle collection (\$merge ou \$out)

```

employees> console.time("creation_employees_denormalises");
employees> db.employees.aggregate([
...   ...pipeline_denormalisation,
...   {$merge: {into: "employees_denormalises", on: "_id", whenMatched: "replace", whenNotMatched: "insert"}},
...   { allowDiskUse: true }).toArray();
[]
employees> console.timeEnd("creation_employees_denormalises");
%: % creation_employees_denormalises 2:13.200 (m:ss.mmm)
employees> |

```

5.6. Mesurer le délai d'accès “complet” via la nouvelle collection

```

employees> db.employees_denormalises.explain("executionStats").find({ _id: 10001 }).limit(1);
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'employees.employees_denormalises',
    parsedQuery: { _id: { '$eq': 10001 } },
    indexFilterSet: false,
    queryHash: 'B774D27B',
    planCacheShapeHash: 'B774D27B',
    planCacheKey: '7635556C',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'EXPRESS_IXSCAN',
      keyPattern: '{ _id: 1 }',
      indexName: '_id_'
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 1,
    executionTimeMillis: 0,
    totalKeysExamined: 1,
    totalDocsExamined: 1,
    executionStages: {
      isCached: false,
      stage: 'EXPRESS_IXSCAN',
      keyPattern: '{ _id: 1 }',
      indexName: '_id_',
      keysExamined: 1,
    }
  }
}

```

6. Conclusions

La dénormalisation proposée (un document employee contenant deux tableaux titles et salaries) est opportune lorsque ces informations sont quasi systématiquement lues ensemble, car elle évite des \$lookup fréquents et améliore les performances de lecture.

Elle est particulièrement adaptée aux charges read-heavy avec peu de mises à jour sur les données dupliquées (sinon la maintenance et la cohérence deviennent coûteuses).

Enfin, elle reste pertinente tant que le document final (employé + tableaux) reste d'une taille raisonnable et ne s'approche pas de la limite de 16 MiB par document BSON ; au-delà, un modèle par références est préférable.