

Projet Migration

1. Création et alimentation des tables avec PostgreSQL

Table Département

Query	Query History	Scratch Pad X
1	<code>SELECT * FROM departments;</code>	
2		
Data Output	Messages	Notifications
Showing rows: 1 to 9 Page No: 1 of 1		
dept_no	dept_name	
[PK] character (4)	character varying (40)	
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

Table Employée

Query

Query History

Scratch

1

2

SELECT * FROM employees;

Data Output

Messages

Notifications

Showing rows: 1 to 1000

Page No: 1

	emp_no [PK] integer	birth_date date	first_name character varying (14)	last_name character varying (16)	gender gender	hire_date 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

Tables relation entre Employée et Département

Query

Query History

1

2

SELECT * FROM dept_emp;

Data Output

Messages

Notifications

≡+

▼

▼

SQL

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

Query History

1

2

SELECT

*

FROM

titles;

Data Output

Messages

Notifications

≡+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

SQL

Showing rows: 1 to 6

	<div>emp_no</div> <div>[PK] integer</div>	<div>title</div> <div>[PK] character varying (50)</div>	<div>from_date</div> <div>[PK] date</div>	<div>to_date</div> <div>date</div>
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

departments

dept_emp

dept_manag

employees

salaries

titles

Trigger Functio

Types (1)

gender

Views

Subscriptions

Data Output

Messages

Notifications

Showing rows: 1 to 1000

Page No: 1

of 444

	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-...

SQL 18 > Databases > employees > Schemas > public > Tables > titles 666

CRLF Ln 2, Col 47

</

2. Cout des jointures en SQL

2.1. Requête SQL de jointure employees + salaries + titles

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

2.3. Une vue matérialisée

Query	Query History
<pre> 76 77 -- 2.3 - Une vue matérialisée sur cette jointure 78 CREATE MATERIALIZED VIEW vue_mat_employes_salaires_titres AS 79 SELECT 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 </pre>	
Data Output	Messages
SELECT 4638507	
Query returned successfully in 20 secs 485 msec.	

2.4. Temps d'accès à la vue matérialisée

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 X Notifications

Total rows: 2844047 Query complete 00:00:04.864

Data Output Messages Graph Visualiser X Notifications

≡+ 📄 ▼ 📋 ▼ 🗑️ 🗄️ ⬇️ 📈 SQL

	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...	
10	{"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

The screenshot displays a PostgreSQL query editor window at the top, showing a SQL script for exporting data to JSON. The script includes comments and four COPY commands for different tables: employees, departments, dept_emp, and salaries. Below the query editor, a Windows File Explorer window is open, showing the 'pg_exports' directory. It contains a table of files with columns for Name, Modified Date, Type, and Size.

```
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 salaries ORDER BY emp_no, dept_no, from_date) AS ligne
185 ) TO 'C:/pg_exports/salaries.json';
```

Nom	Modifié le	Type	Taille
departments	30/12/2025 05:20	Fichier source JSON	1 Ko
dept_emp	30/12/2025 05:20	Fichier source JSON	27 105 Ko
dept_manager	30/12/2025 05:21	Fichier source JSON	2 Ko
employees	30/12/2025 05:14	Fichier source JSON	37 231 Ko
salaries	30/12/2025 05:21	Fichier source JSON	227 004 Ko
titles	30/12/2025 05:21	Fichier source JSON	38 084 Ko

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
PS C:\Users\ibodi> mongoimport --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> mongoimport --db employees --collection 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> mongoimport --db employees --collection dept_emp --file "C:\pg_exports\dept_emp.json" --jsonArray --drop
connected to: mongodb://localhost/
dropping: employees.dept_emp
331603 document(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\ibodi> mongoimport --db employees --collection dept_manager --file "C:\pg_exports\dept_manager.json" --jsonArray --drop
connected to: mongodb://localhost/
dropping: employees.dept_manager
24 document(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\ibodi> mongoimport --db employees --collection 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%)
443308 document(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\ibodi> mongoimport --db employees --collection salaries --file "C:\pg_exports\salaries.json" --jsonArray --drop
connected to: mongodb://localhost/
dropping: employees.salaries
[#####] employees.salaries 22.4MB/222MB (10.1%)
[#####] employees.salaries 36.1MB/222MB (16.3%)
[#####] employees.salaries 49.5MB/222MB (22.3%)
[#####] employees.salaries 63.6MB/222MB (28.7%)
[#####] employees.salaries 77.8MB/222MB (35.1%)
[#####] employees.salaries 102MB/222MB (46.1%)
[#####] employees.salaries 130MB/222MB (58.6%)
[#####] employees.salaries 152MB/222MB (68.6%)
[#####] employees.salaries 168MB/222MB (75.8%)
[#####] employees.salaries 192MB/222MB (86.8%)
[#####] employees.salaries 222MB/222MB (100.0%)
[#####] 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
PS C:\Users\ibodi> mongosh
Current Mongosh Log ID: 69535de1fda4643062cbea3
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/mongosh-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
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Installez la dernière version de PowerShell pour de nouvelles fonctionnalités et améliorations ! https://aka.ms/PSWindows

PS C:\Users\ibodi> mongosh
Current Mongosh Log ID: 695368a8e4450ab8f5cebea3
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> 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)

```
mongosh mongodb://127.0.0.1
employees> 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('69535d5ecef988cbb7d5f02b'),
        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',

```

...

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

```

mongosh mongodb://127.0.0.1
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: 'BB98BAF92',
    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");
%: %s 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.