

Migrer 10 ans de business de datastore à postgres

Geoffrey Coulaud - API engineer @ Pictarine



984



114



57

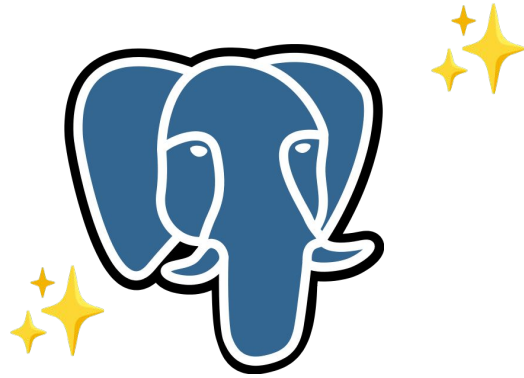


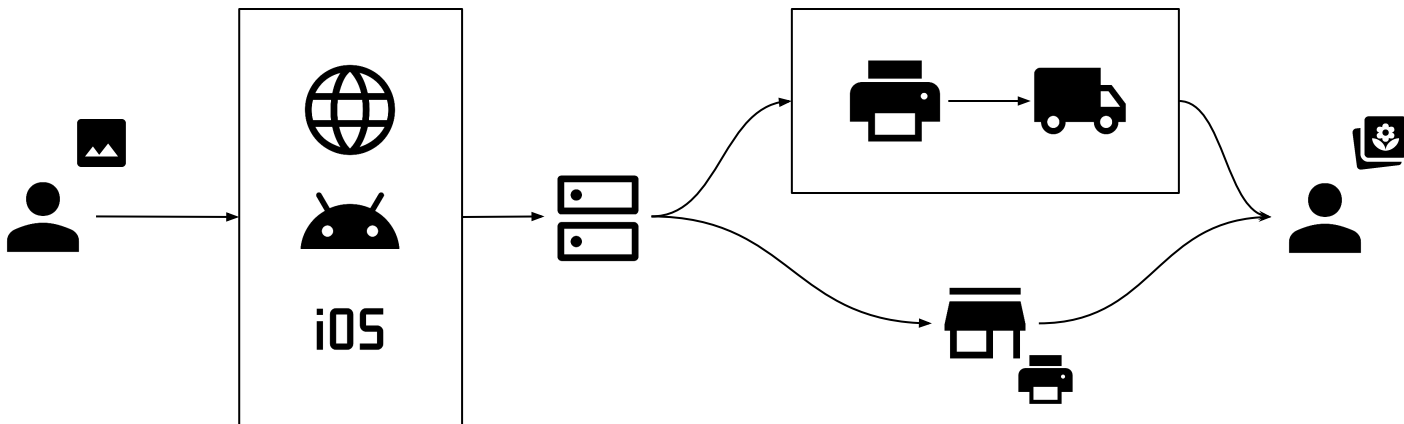
~1

574 jours



~2015 - 2024







1. Migrer 10 ans de données sans downtime
2. Surprises, bonnes et moins bonnes

Migrations

Migrations

Schéma



Insérer une donnée crée la
colonne associée



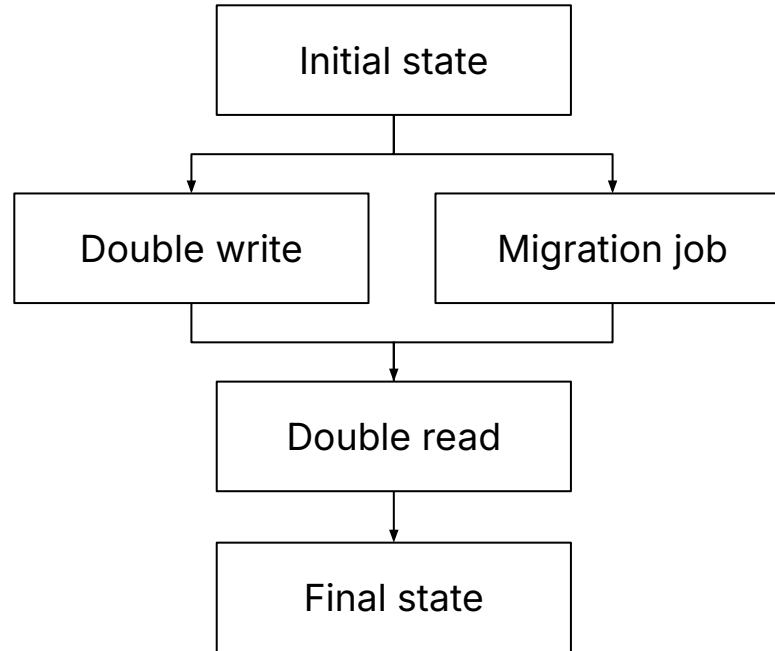
Définition du schéma à la main



Entités définissent le schéma

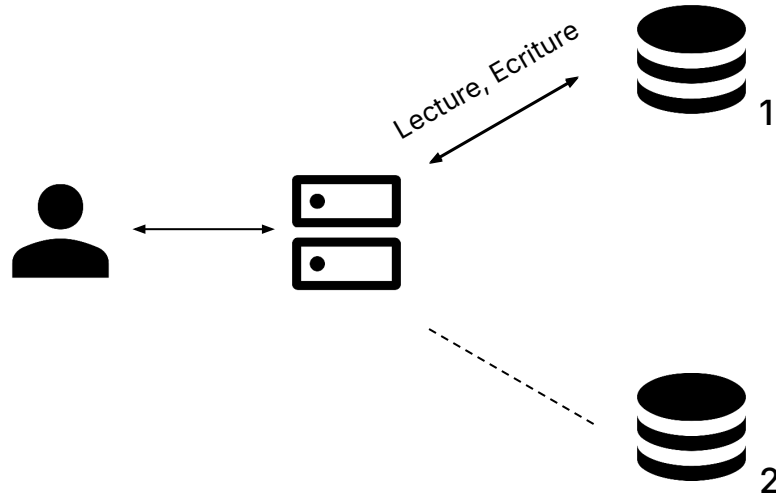
Migrations

Migration Zero Downtime



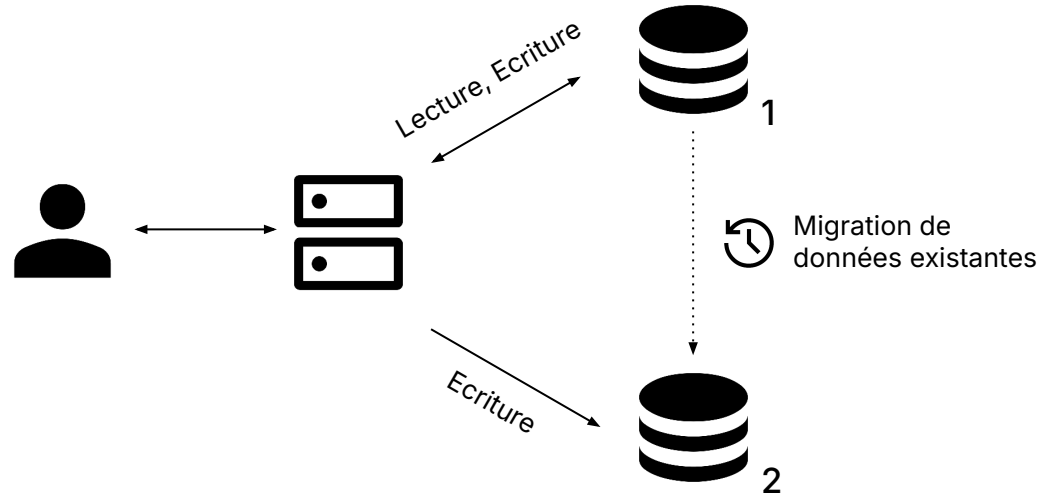
Migrations

Migration Zero Downtime - Initial state



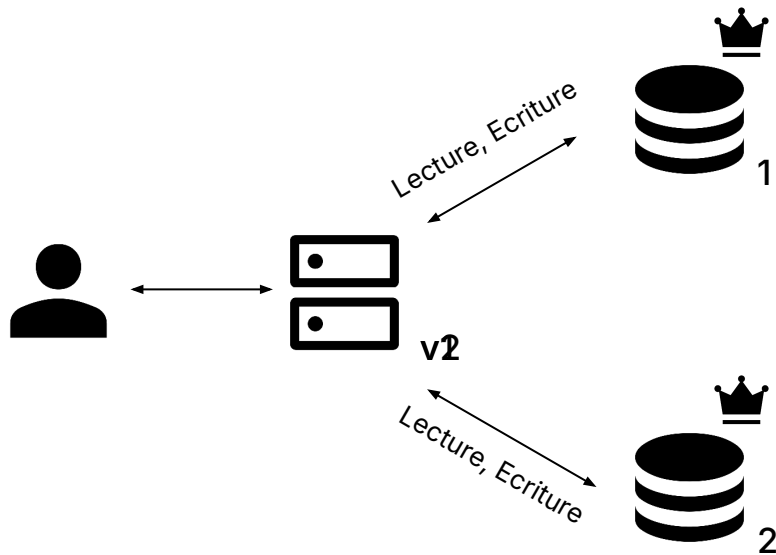
Migrations

Migration Zero Downtime - Double write & job



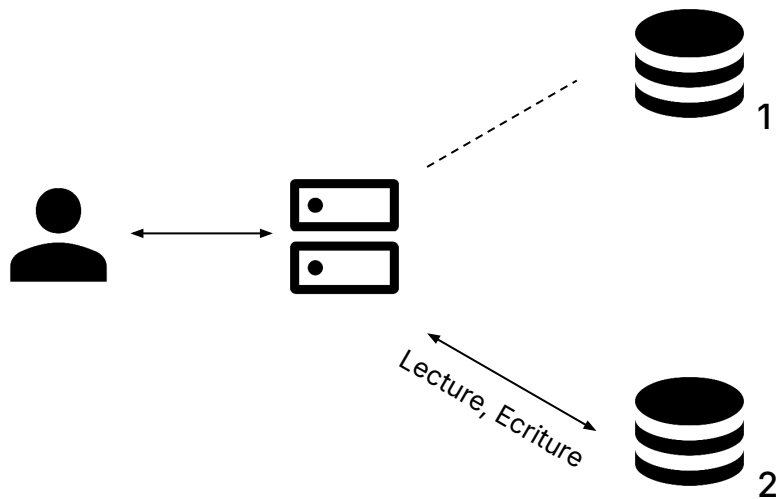
Migrations

Migration Zero Downtime - Double read



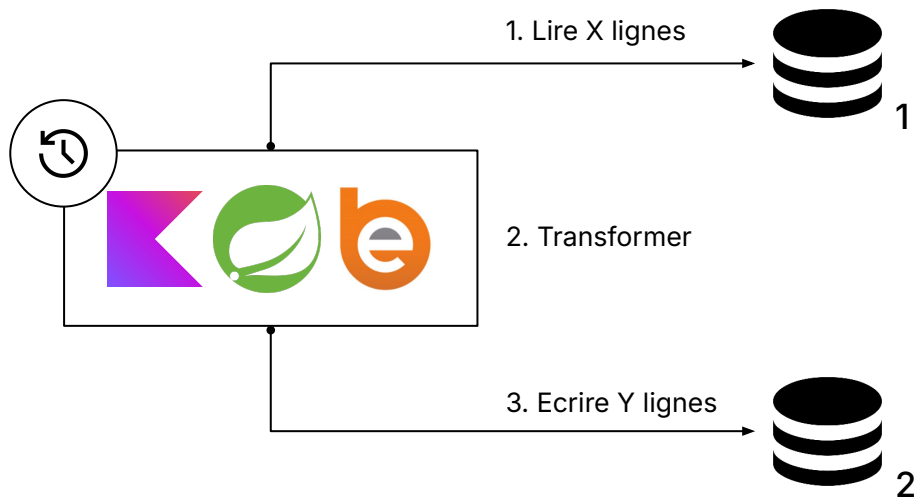
Migrations

Migration Zero Downtime - Double read



Migrations

Données existantes - Cas "job"



Migrations

Données existantes - Cas simple

- Volume moyen, < 50 Go
- Relation de destination à 1 table



Bonus Migrations PG-PG



Lit & Ecrit name



Lit & Ecrit full_name,
Fallback sur name si NULL

+

Ecrit sur name



Lit & Ecrit full_name



USER_V1		
int	id	PK
text	name	



USER_V2		
int	id	PK
text	name	
text	full_name	



SQL UPDATE



USER_V3		
int	id	PK
text	full_name	

Apprentissages

Apprentissages

Typage des enums



ENUM
PG TYPE



CHECK
CONSTRAINT



Apprentissages

Typage des enums - Check constraint

Ajout de "GOOTEN"

```
1 -- drop dependencies
2 alter table orders drop constraint if exists ck_orders_partner;
3 -- apply post alter
4 alter table orders add constraint ck_orders_partner check ( partner in ('CVS',
    'WALGREENS', 'NONE', 'KOSTA_FR', 'DISTRICT_PHOTO', 'GOOTEN', 'FUJIFILM_EU', 'FUJIFILM_US',
    'WALMART') not valid ;
```

ALTER TABLE ...
ADD CONSTRAINT ...
CHECK ...



ALTER TABLE ...
ADD CONSTRAINT ...
CHECK ... NOT VALID;

VALIDATE ... ;



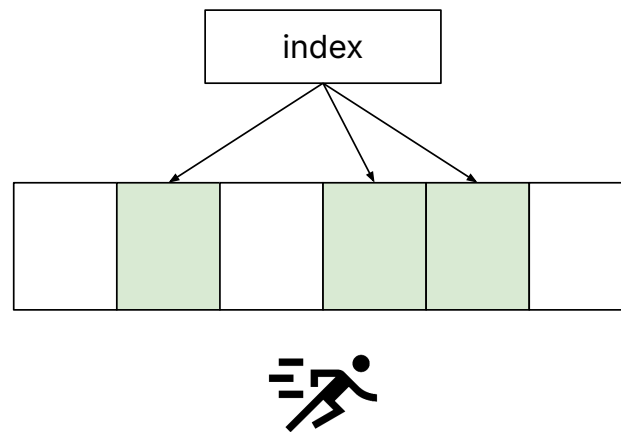
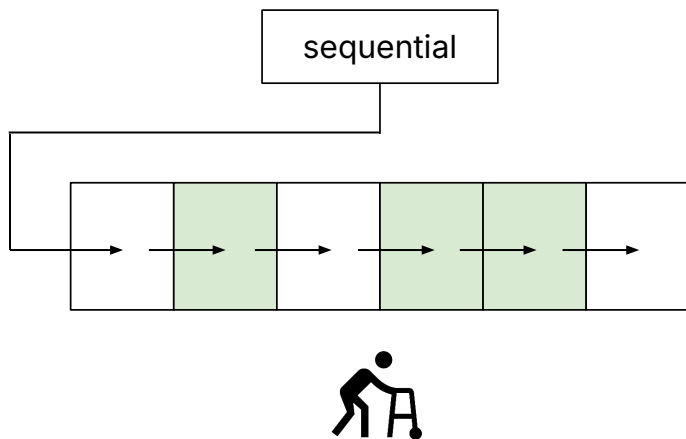
Normally, this form will cause a scan of the table to verify that all existing rows in the table satisfy the new constraint. But if the **NOT VALID** option is used, this potentially-lengthy scan is skipped. The constraint will still be applied against subsequent inserts or updates.

most forms of **ADD table_constraint** require an **ACCESS EXCLUSIVE** lock,

The validation step does not need to lock out concurrent updates only pre-existing rows need to be checked. Hence, validation acquires only a **SHARE UPDATE EXCLUSIVE** lock on the table being altered.

Apprentissages

Query planning - Utilisation des index



Apprentissages

Query planning - Utilisation des index



`random_page_cost` (floating point)

Sets the planner's estimate of the cost of a non-sequentially-fetched disk page. The default is 4.0. Reducing this value relative to `seq_page_cost` will cause the system to prefer index scans; raising it will make index scans look relatively more expensive.

Random access to mechanical disk storage is normally much more expensive than four times sequential access. However, a lower default is used (4.0) because the majority of random accesses to disk, such as indexed reads, are assumed to be in cache.

Storage that has a low random read cost relative to sequential, e.g., solid-state drives, might also be better modeled with a lower value for `random_page_cost`, e.g., 1.1.

Apprentissages

Problème N+1

GET catalog/{name} →



1. SELECT * FROM catalog WHERE name = ...

2. SELECT * FROM section WHERE catalogId = ...

3. SELECT * FROM category WHERE sectionId in ...

...

Catalog		
UUID	id	PK
string	name	
string	description	

has many

Section		
UUID	id	PK
UUID	catalogId	FK
int	rank	
string	name	

has many

Category		
UUID	id	PK
UUID	sectionId	FK
int	rank	
string	name	

has many

ProductPage		
UUID	id	PK
UUID	categoryId	FK
int	rank	
string	name	
string	description	
UUID	design	FK

used by

Design		
UUID	id	PK
string	name	
jsonb	data	
string	previewUrl	

Apprentissages

Problème N+1

GET catalog/{name}

```
1. SELECT * FROM catalog
   WHERE name = ...
   LEFT JOIN ...
```

```
.fetch("sections")
.fetch("sections.categories")
...
```

Catalog		
UUID	id	PK
string	name	
string	description	

has many

Section		
UUID	id	PK
UUID	catalogId	FK
int	rank	
string	name	

has many

Category		
UUID	id	PK
UUID	sectionId	FK
int	rank	
string	name	

has many

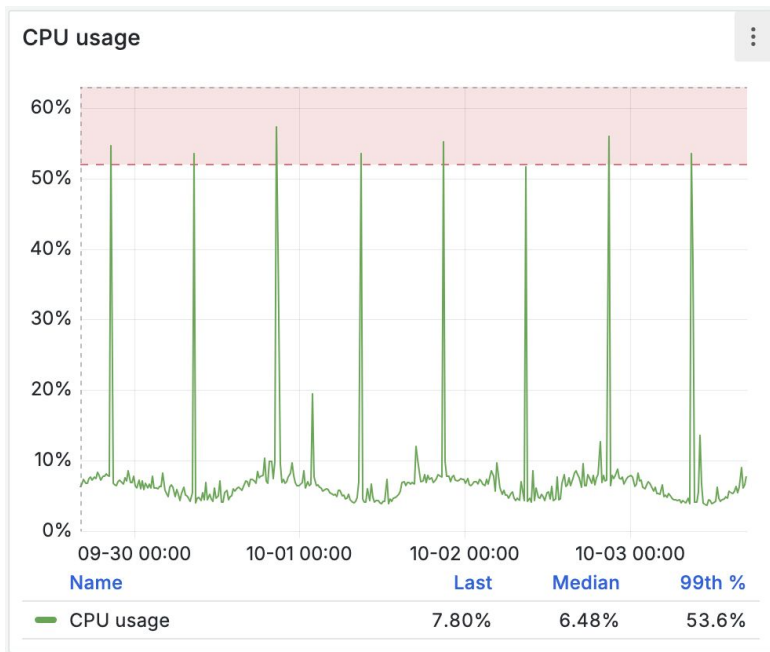
ProductPage		
UUID	id	PK
UUID	categoryId	FK
int	rank	
string	name	
string	description	
UUID	design	FK

Design		
UUID	id	PK
string	name	
jsonb	data	
string	previewUrl	

used by

Apprentissages

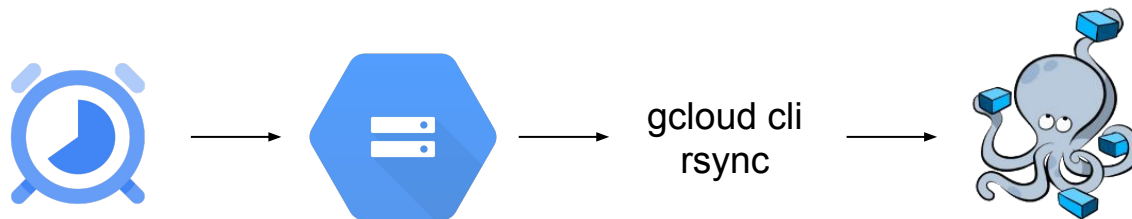
pg_stat_statements



The `pg_stat_statements` module provides a means for tracking planning and execution statistics of all SQL statements executed by a server.

Apprentissages

Dump de la base d'intégration



Apprentissages

Tests de base de données



Apprentissages

Tests de base de données

@Test  Geoffrey Coulaud

```
fun `Text colors should be sorted by rank ascending`() {  
    // Given  
    val textColors : List<TextColor> = buildList { repeat(100) { add(createTextColor()) } }  
    textColorRepository.insertAll(textColors)  
  
    // When  
    val fromDb : List<TextColor> = textColorRepository.findAll()  
  
    // Then  
    val fromDbSorted : List<TextColor> = fromDb.sortedBy { it.rank }  
    assertEquals(fromDbSorted, fromDb)  
}
```

Conclusion

- Migration sans accrocs, pas à pas
- Charge DB tenue sans problème
- Gain de performance API
- Documentation et évolutivité du schéma
- Fonctionnalités nouvelles

brthday



Suggestions

-  Birthday Cards
-  Birthday

[< Previous](#)

[Next >](#)