Mission DBA

Tutoriel interactif Catrust / CatrustDB

7 niveaux pour maîtriser le moteur de données catégorielle

Scénario : vous êtes le nouveau DBA de la startup *WidgetCorp*

Catrust — documentation didactique

Avril 2026 — v1.3

« Une base de données est un foncteur.
Une migration est une transformation naturelle.
Mais aujourd'hui, c'est vous qui décidez. »

Table des matières

Prologue — Bienvenue chez WidgetCorp

Contexte de la mission

Nous sommes le 10 avril 2026.

Vous venez d'être recruté comme DBA senior chez **WidgetCorp**, une startup spécialisée dans la gestion de personnel. La société possède une base de données existante sous PostgreSQL mais cherche à migrer vers un système plus rigoureux, basé sur les *schémas catégoriels*.

Votre outil : **Catrust / CatrustDB** — un moteur de données en Rust qui traite les schémas comme des catégories et les migrations comme des foncteurs.

Votre mission, si vous l'acceptez, se déroule en **7 niveaux** :

1. Définir le schéma de WidgetCorp
2. Peupler la base avec les employés
3. Interroger et analyser les données
4. Sécuriser le serveur TCP
5. Migrer vers une nouvelle version du schéma
6. Protéger les données personnelles (RGPD)
7. Audit de sécurité final (RSSI)

Comment utiliser ce tutoriel

Chaque niveau suit la même structure :

Briefing Le contexte métier du problème à résoudre.

Théorie Le concept Catrust mis en jeu (en option, pour les curieux).

Pratique Les commandes exactes à taper, avec les sorties attendues.

Exercice Un défi supplémentaire pour consolider.

Résultat attendu Comment vérifier que vous avez réussi.

Objectif

Ce tutoriel est disponible dans deux éditions :

— **Community (gratuit)** — suivez les instructions « Option A »ci-dessous.

— **Enterprise (licence)** — suivez la même progression mais consultez le chapitre A pour activer les fonctionnalités avancées (RBAC, WAL chiffré, RGPD complet).

Option A — Binaire pré-compilé (Community, recommandé) :

```
# Linux
curl -sSfL https://catrust.eu/releases/install.sh | sh

# macOS
curl -sSfL https://catrust.eu/releases/install.sh | sh

# Windows
# T l charger l'installeur : https://catrust.eu/download.html
# -> catrust-v1.0.0-x86_64-windows.msi (ajoute catrust au PATH)

# V rification
catrust --version      # catrust 1.0.0 (community)
```

Option B — Compilation depuis les sources (Rust stable $\geq$ 1.77) :

```
git clone https://github.com/sirgudu/Catrust
cd Catrust
cargo install --path . --locked
catrust --version      # catrust 1.0.0
```

Option C — Docker (aucun prérequis) :

```
docker run -p 7878:7878 catrustdb/catrust:latest
```

Deux binaires sont disponibles :

— `catrust` — CLI de gestion (parse, validate, migrate, viz, serve)

— `catrustdb` (Enterprise) — serveur autonome TCP/NDJSON, port 7474

Chapitre 1

Niveau 1 — Définir le schéma de WidgetCorp

Difficulté : Débutant ● ○ ○ ○ ○

1.1 Briefing

L'équipe RH vous remet une liste sur papier :

> *« Nous avons des employés. Chaque employé appartient à un département. Un employé a un nom et un salaire. Un département a un nom et un budget. »*

Votre première tâche : modéliser cette réalité en CQL (*Categorical Query Language*).

1.2 Concepts clés

Un **schéma CQL** contient trois éléments :

`entities` Les *nœuds* de votre graphe de données (tables en SQL, sommets en théorie des graphes).

`foreign_keys` Les *arêtes* — une FK est une fonction totale d'une entité vers une autre.

`attributes` Les *étiquettes* — chaque attribut est une fonction totale d'une entité vers un type de base (`String`, `Float`, `Integer`, `Bool`).

Coulisse : la théorie des catégories

En théorie des catégories, un schéma CQL est une **petite catégorie** $\mathcal{C}$:

- Les entités sont les *objets* de $\mathcal{C}$.
- Les FK et attributs sont les *morphismes* de $\mathcal{C}$.
- Les équations de chemins (`path_equations`) sont les relations de commutativité de $\mathcal{C}$.

Une *instance* de ce schéma est alors un foncteur $I : \mathcal{C} \to \mathbf{Set}$.

1.3 Mise en pratique

Créez le fichier `widgetcorp.cql` :

```
schema WidgetCorp {
  entities
    Employee
    Department

  foreign_keys
    dept : Employee -> Department

  attributes
    emp_name  : Employee   -> String
    salary    : Employee   -> Float
    dept_name : Department -> String
    budget    : Department -> Float
}
```

Listing 1.1 – Schéma initial de WidgetCorp

Validez le schéma :

```
catrust validate widgetcorp.cql
```

Résultat attendu

```
[OK] Schema 'WidgetCorp' valide : 2 entites, 1 FK, 4 attributs
```

1.4 Explorer le schéma

Générez le DDL SQL pour vérifier que la structure est correcte :

```
catrust generate sql --schema widgetcorp.cql --dialect postgres
```

Catrust produit automatiquement les clés primaires (`_id BIGINT`) et les contraintes `NOT NULL` sur les FK.

Exercice

Le DRH vous informe qu'il faut aussi stocker la **date d'embauche** de chaque employé.

1. Ajoutez un attribut `hire_date : Employee -> String` (format ISO 8601).
2. Revalidez.
3. Regénérez le DDL — observez la nouvelle colonne.

Piège courant

CQL ne distingue pas encore `Date` de `String` au niveau du schéma. Cependant, le *moteur de données* (CatrustDB) valide le format ISO 8601 `YYYY-MM-DD` à l'insertion grâce au type `Val::Date`. Utilisez `hire_date : Employee -> String` dans le schéma, et passez des chaînes `"2024-03-15"` dans les données.

Chapitre 2

Niveau 2 — Peupler la base de données

Difficulté : Débutant ●●○○○

2.1 Briefing

Le schéma est validé. L'équipe RH vous envoie un tableur Excel avec 5 employés et 3 départements. Votre mission : charger ces données dans le moteur CatrustDB.

2.2 Le format instance (.cqli)

Une *instance* CQL est un ensemble de lignes qui respectent le schéma :

```
instance ProdData : WidgetCorp {

  // Les departements en premier (les employes referencent leurs IDs)
  Department {
    d1  dept_name="Engineering"  budget=150000.0
    d2  dept_name="Marketing"    budget=80000.0
    d3  dept_name="Research"     budget=200000.0
  }

  Employee {
    e1  dept=d1  emp_name="Alice"    salary=90000.0
    e2  dept=d1  emp_name="Bob"      salary=75000.0
    e3  dept=d2  emp_name="Charlie"  salary=70000.0
    e4  dept=d2  emp_name="Diana"    salary=82000.0
    e5  dept=d3  emp_name="Eve"      salary=95000.0
  }
}
```

Listing 2.1 – Instance widgetcorp.cqli

Les symboles `d1`, `d2`, `e1`...sont des *étiquettes locales* : ils servent uniquement à relier les lignes via les FK. Ils n'apparaissent pas dans les données finales.

2.3 Démarrer le serveur CatrustDB

Ouvrez un premier terminal et démarrez le serveur :

```
catrustdb --port 7474 --token "mon-secret"
```

Dans un second terminal, chargez le schéma et l'instance :

```
echo '{"cmd":"load","auth":"mon-secret",
      "schema_cql":"schema␣WidgetCorp␣{␣entities␣Employee␣Department
␣␣␣␣␣␣␣␣foreign_keys␣dept:Employee->Department
␣␣␣␣␣␣␣␣attributes␣emp_name:Employee->String␣salary:Employee->Float
␣␣␣␣␣␣␣␣dept_name:Department->String␣budget:Department->Float␣}",
      "instance_cql":"instance␣ProdData␣:␣WidgetCorp␣{
␣␣␣␣␣␣␣␣Department␣{␣d1␣dept_name=\"Engineering\"␣budget=150000.0
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣d2␣dept_name=\"Marketing\"␣␣␣budget=80000.0␣}
␣␣␣␣␣␣␣␣Employee␣␣␣{␣e1␣dept=d1␣emp_name=\"Alice\"␣salary=90000.0
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣e2␣dept=d1␣emp_name=\"Bob\"␣␣␣salary=75000.0␣}␣}"}'
    \
  | nc localhost 7474
```

Listing 2.2 – Requête NDJSON — chargement

Résultat attendu

```
{"ok":true,"entities":2,"rows":4}
```

Vérifiez les statistiques :

```
echo '{"cmd":"stats","auth":"mon-secret"}' | nc localhost 7474
```

Exercice

Ajoutez un troisième département `"Research"` et deux employés supplémentaires liés à ce département en envoyant une nouvelle commande `load` complète (schema + instance mise à jour). Vérifiez avec `stats` que vous avez bien 5 employés et 3 départements.

Coulisse : la théorie des catégories

Un foncteur $I : \mathcal{C} \to \mathbf{Set}$ associe :

- à chaque entité E un ensemble $I(E)$ (les lignes),
- à chaque morphisme $f : A \to B$ une fonction totale $I(f) : I(A) \to I(B)$ (les FK et attributs).

Le fait que chaque employé ait *exactement un* département (FK totale) est une propriété catégorielle fondamentale : $I(\texttt{dept})$ est une fonction, pas une relation.

Chapitre 3

Niveau 3 — Interroger et analyser les données

Difficulté : Intermédiaire ●●●○○

3.1 Briefing

La direction demande un rapport mensuel :

— La masse salariale par département.

— La liste des employés triés par salaire décroissant.

— Le département ayant le budget moyen par employé le plus élevé.

3.2 Requêtes CQL

CatrustDB dispose d'un langage de requête déclaratif. La syntaxe suit le modèle `from / where / select`.

3.2.1 Requête simple — liste des employés

```
query AllEmployees : WidgetCorp {
  from e : Employee
  select e.emp_name  e.salary  e.dept.dept_name
  order_by e.salary desc
}
```

Listing 3.1 – Tous les employés par salaire décroissant

```
{"cmd":"query","auth":"mon-secret","cql_query":"
␣␣query␣AllEmployees␣:␣WidgetCorp␣{
␣␣␣␣from␣e␣:␣Employee
␣␣␣␣select␣e.emp_name␣␣e.salary␣␣e.dept.dept_name
␣␣␣␣order_by␣e.salary␣desc␣}"}
```

Résultat attendu


```
{"ok":true,"name":"AllEmployees",
 "columns":["e.emp_name","e.salary","e.dept.dept_name"],
 "rows":[["Eve","95000","Research"],
         ["Alice","90000","Engineering"],
         ["Diana","82000","Marketing"],
         ["Bob","75000","Engineering"],
         ["Charlie","70000","Marketing"]],
 "row_count":5}
```


3.2.2 GROUP BY — masse salariale par département

```
query SalaryByDept : WidgetCorp {
  from e : Employee
  group_by e.dept.dept_name
  select e.dept.dept_name  SUM(e.salary)  COUNT(*)  AVG(e.salary)
}
```

Listing 3.2 – Agrégation par département

Exercice

1. Écrivez une requête qui liste les départements dont la *masse salariale totale* dépasse `100000`. *Indice :* utilisez `where` avant `select` pour filtrer sur un attribut.
2. Trouvez le département avec le budget par employé le plus élevé. *Indice :* `AVG(e.salary)` vous donne la moyenne salariale — comparez avec `e.dept.budget`.
3. Quels employés ont un salaire supérieur à la moyenne du département Engineering ?

3.2.3 Traversée de FK

Un des super-pouvoirs de CQL : la **traversée de FK en un point**. Le chemin `e.dept.dept_name` signifie littéralement : *« suivre la FK `dept` depuis `e`, puis lire l'attribut `dept_name` ».*

```
query Engineering : WidgetCorp {
  from e : Employee
  where e.dept.dept_name = "Engineering"
  select e.emp_name  e.salary
}
```

Listing 3.3 – Employés du département Engineering

Coulisse : la théorie des catégories

Une requête CQL $Q : \mathcal{C}$ est évaluée par composition de morphismes :

$$\texttt{e.dept.dept\_name} \equiv I(\texttt{dept\_name}) \circ I(\texttt{dept}) : I(E) \to \textbf{String}$$

Le chemin est une composition de fonctions — c'est exactement la définition d'un morphisme dans la catégorie $\mathcal{C}$.

Chapitre 4

Niveau 4 — Sécuriser le serveur

Difficulté : Intermédiaire ●●●○○

4.1 Briefing

Le RSSI de WidgetCorp vous convoque :

« Le serveur de base de données est exposé sur le réseau interne. Nous avons besoin d'authentification par rôle, de TLS, et d'un audit log. Et si quelqu'un nous vole la machine, les données WAL ne doivent pas être lisibles. »

4.2 Panorama des mesures de sécurité disponibles

Mesure	Flag	Effet
Auth Bearer	`-token <secret>`	Token unique, niveau admin
RBAC	`-token-ro / -token-rw / -token-admin`	3 rôles : lecture seule, lecture+écriture, administration
TLS	`-tls-cert <pem> -tls-key <pem>`	Connexion chiffrée TLS 1.3 (rustls)
WAL chiffré	`-wal-key <passphrase>`	AES-256-GCM par ligne de WAL
Audit log	`-audit-log <path>`	NDJSON : `{ts, peer, cmd, ok, duration_ms}`
Rotation logs	`-audit-log-max-mb <n>`	Archive automatique quand la taille limite est dépassée
Lecture seule	`-read-only`	Refuse toutes les commandes d'écriture
Rate limiting	`-max-connections <n>`	Semaphore tokio, défaut 128

Table 4.1 – Paramètres de sécurité de CatrustDB

4.3 Configuration de production

```
# Generez un certificat auto-signe pour les tests
openssl req -x509 -newkey rsa:4096 -keyout server.key \
        -out server.crt -days 365 -nodes \
        -subj "/CN=widgetcorp-db"

# Demarrez avec toutes les mesures actives
catrustdb \
  --port 7474 \
  --tls-cert server.crt \
  --tls-key  server.key \
  --token-ro    "lecteur-42xZ" \
  --token-rw    "editeur-78aB" \
  --token-admin "admin-9kLmN" \
  --wal-key     "passphrase-AES256" \
  --audit-log   /var/log/catrustdb/audit.log \
  --audit-log-max-mb 50 \
```

```
  --max-connections 64 \
  --data-dir    /var/lib/catrustdb
```

Listing 4.1 – Démarrage sécurisé

4.4 Vérifier la posture de sécurité

La commande `security_report` vous donne un tableau de bord instantané :

```
{"cmd":"security_report","auth":"admin-9kLmN"}
```

Résultat attendu

```
{
  "ok":                 true,
  "tls":                true,
  "auth_mode":          "rbac",
  "roles_configured":   ["ro","rw","admin"],
  "wal_encryption":     true,
  "audit_log":          true,
  "read_only":          false,
  "rate_limit":         64,
  "max_payload_bytes":  67108864
}
```

4.5 Rotation de clé WAL

Si la passphrase WAL est compromise, vous pouvez la changer à chaud sans redémarrer le serveur :

```
{"cmd":"rotate_wal_key",
 "auth":"admin-9kLmN",
 "path":"/var/lib/catrustdb/prod.wal",
 "new_key":"nouvelle-passphrase-rotee"}
```

Exercice

1. Démarrez le serveur avec uniquement `-token-ro "lecture"`.
2. Essayez d'envoyer une commande `load` avec ce token. Que se passe-t-il ?
3. Ajoutez `-token-rw "edition"` et relancez. Avec quel token pouvez-vous désormais charger des données ?
4. Essayez `rotate_wal_key` avec le token rw — est-ce autorisé ?

Coulisse : la théorie des catégories

RBAC (Role-Based Access Control) dans CatrustDB respecte une **hiérarchie** :

$$\texttt{ReadOnly} \leq \texttt{ReadWrite} \leq \texttt{Admin}$$

Chaque token est résolu par comparaison en temps constant (*constant-time equality*) pour résister aux attaques par timing. La résolution est *fail-secure* : un token inconnu retourne `"unauthorized"` sans révéler si le token existe.

Chapitre 5

Niveau 5 — Migrer vers une nouvelle version du schéma

Difficulté : Avancé ●●●●○

5.1 Briefing

Six mois ont passé. WidgetCorp rachète une autre startup dont la base de données utilise un schéma différent — `OldHR`. Votre mission : migrer toutes les données de `OldHR` vers `WidgetCorp` en préservant l'intégrité des FK.

Le schéma source :

```
schema OldHR {
  entities
    Person
    Team

  foreign_keys
    team : Person -> Team

  attributes
    full_name  : Person -> String
    annual_pay : Person -> Float
    team_name  : Team   -> String
    headcount  : Team   -> Integer
}
```

Listing 5.1 – Schéma source OldHR

5.2 Définir le mapping (foncteur Σ)

Un mapping CQL est une correspondance *structurelle* entre deux schémas. Ce n'est pas un ETL — c'est la définition d'un **foncteur** $F : \mathcal{C}_{\text{OldHR}} \to \mathcal{C}_{\text{WidgetCorp}}$.

```
mapping Fusion : OldHR -> WidgetCorp {
```

```
  entities
    Person -> Employee
    Team   -> Department

  foreign_keys
    team -> dept

  attributes
    full_name  -> emp_name
    annual_pay -> salary
    team_name  -> dept_name
    headcount  -> budget    // approximation : on le recalculera apres
}
```

Listing 5.2 – Mapping OldHR -> WidgetCorp

Piège courant

L'attribut `headcount : Team -> Integer` est mappé vers `budget : Department -> Float`. CatrustDB effectue la conversion de type automatiquement, mais vous perdez la sémantique originale. En production, créez un attribut `budget` séparé avec la vraie valeur.

5.3 Exécuter la migration

5.3.1 Validation préalable (dry-run)

```
{"cmd":"sigma_validate","auth":"admin-9kLmN",
 "target_cql":"<schema␣WidgetCorp␣...>",
 "mapping_cql":"<mapping␣Fusion␣...>"}
```

```
{"ok":true,"valid":true,"warnings":[]}
```

5.3.2 Migration complète (Σ)

```
{"cmd":"sigma","auth":"admin-9kLmN",
 "target_cql":"<schema␣WidgetCorp␣...>",
 "mapping_cql":"<mapping␣Fusion␣...>"}
```

Résultat attendu

```
{"ok":true,"schema":"WidgetCorp",
 "entities":[{"name":"Department","count":3,...},
             {"name":"Employee","count":5,...}],
 "duration_ms":2}
```

5.3.3 Migration incrémentale ($\Sigma\Delta$)

Si la source reçoit de nouveaux employés régulièrement, la migration complète redevient coûteuse. Utilisez `sigma_delta` pour ne traiter que les nouvelles lignes :

```
{"cmd":"sigma_delta","auth":"admin-9kLmN",
 "target_cql":"<schema␣WidgetCorp...>",
 "mapping_cql":"<mapping␣Fusion...>"}
// Retourne le checkpoint (dernier row_id traite)
// Le serveur le memorise -- prochain appel repart de ce point
```

5.4 Registre de migrations (Flyway-style)

Pour les migrations planifiées, organisez vos fichiers CQL en versions :

```
migrations/
  V001__init_widgetcorp.cql
  V002__ajout_hire_date.cql
  V003__ajout_manager.cql
```

```
{"cmd":"migrate","auth":"admin-9kLmN","dir":"/migrations"}
// Applique uniquement les versions non encore appliquees
```

```
{"cmd":"migrate_status","auth":"admin-9kLmN","dir":"/migrations"}
// Liste : version | nom | status (applied/pending)
```

Exercice

1. Créez un schéma `OldHR` minimal (2 entités, 1 FK, 4 attributs).
2. Chargez-le dans le serveur avec 3 personnes dans 2 équipes.
3. Créez le mapping vers `WidgetCorp`.
4. Exécutez `sigma_validate` puis `sigma`.
5. Vérifiez avec `lineage` que l'événement est enregistré.

Coulisse : la théorie des catégories

La migration Σ_F est un **foncteur image directe** : étant donné un foncteur de schémas $F : \mathcal{C} \to \mathcal{D}$ et une instance $I : \mathcal{C} \to \mathbf{Set}$, on calcule $\Sigma_F(I) : \mathcal{D} \to \mathbf{Set}$ par colimite. Propriété clef : Σ_F est **exact à gauche** (préserve les morphismes injectifs). C'est pourquoi les intégrités référentielles sont toujours préservées après migration — c'est une *garantie mathématique*, pas une contrainte implémentée.

Chapitre 6

Niveau 6 — Protéger les données personnelles (RGPD)

Difficulté : Avancé ●●●●○

6.1 Briefing

Le DPO (Data Protection Officer) vous envoie un formulaire :

> *« Alice Martin vient de quitter la société et exerce son droit à l'effacement (Art. 17 RGPD). Bob Dupont demande une copie de ses données (Art. 20 RGPD — portabilité). De plus, le salaire de Diana a été modifié suite à une promotion. »*

6.2 Le soft-delete : effacement sans perte d'audit

CatrustDB implémente un **soft-delete** : les lignes supprimées sont marquées comme inactives mais conservées dans le WAL pour l'audit. Elles n'apparaissent plus dans les requêtes ni dans les exports.

6.2.1 Trouver l'identifiant d'Alice

```
{"cmd":"query","auth":"admin-9kLmN","cql_query":"
␣␣query␣FindAlice␣:␣WidgetCorp␣{
␣␣␣␣from␣e␣:␣Employee
␣␣␣␣where␣e.emp_name␣=␣\"Alice\"
␣␣␣␣select␣e.emp_name␣␣e.salary␣}"}
```

Les `row_id` sont retournés dans chaque ligne de résultat. Supposons qu'Alice a le `row_id = 0`.

6.2.2 Droit à l'effacement (Art. 17 RGPD)

```
{"cmd":"delete","auth":"admin-9kLmN",
 "entity":"Employee","row_id":0}
```

Résultat attendu

```
{"ok":true,"entity":"Employee","row_id":0}
```

Alice n'apparaît plus dans aucune requête ni aucun export.

6.2.3 Suppression par critère (`delete_where`)

Pour supprimer tous les employés d'un département fermé :

```
{"cmd":"delete_where","auth":"admin-9kLmN",
 "entity":"Employee","attr":"dept_name",
 "eq":"Marketing"}
```

6.2.4 Mise à jour — promotion de Diana (Art. 16 RGPD)

```
{"cmd":"update","auth":"admin-9kLmN",
 "entity":"Employee","row_id":3,
 "attrs":{"salary":95000.0}}
```

6.3 Portabilité des données (Art. 20 RGPD)

6.3.1 Export CSV

```
{"cmd":"export_csv","auth":"admin-9kLmN",
 "entity":"Employee"}
```

La réponse contient la clé `"csv"` avec le contenu CSV complet (uniquement les lignes *actives* — les lignes supprimées sont exclues).

6.3.2 Export JSON Lines

```
{"cmd":"export_jsonl","auth":"admin-9kLmN",
 "entity":"Employee"}
```

6.4 Transactions pour les opérations complexes

Plusieurs modifications atomiques :

```
{"cmd":"begin","auth":"admin-9kLmN"}
{"cmd":"delete","auth":"admin-9kLmN","entity":"Employee","row_id":0}
{"cmd":"update","auth":"admin-9kLmN","entity":"Employee","row_id":3,
 "attrs":{"salary":95000.0}}
{"cmd":"commit","auth":"admin-9kLmN"}
// En cas d'erreur : {"cmd":"rollback","auth":"admin-9kLmN"}
```

Exercice

1. Bob Dupont demande ses données. Chargez la base, effectuez un export JSONL de l'entité `Employee`, puis filtrez manuellement la ligne de Bob.
2. Diana est promue chef de département. Mettez à jour son salaire à `95000` et le budget du département Marketing à `120000`. Faites-le dans une transaction (begin/commit).
3. Vérifiez avec `stats` que le nombre d'employés actifs a bien diminué après la suppression d'Alice.

Piège courant

La commande `stats` retourne le nombre total de lignes *incluant* les lignes supprimées. Pour compter uniquement les lignes actives, utilisez une requête :

```
{"cmd":"query","auth":"admin-9kLmN","cql_query":"
␣␣query␣Count␣:␣WidgetCorp␣{
␣␣␣␣from␣e␣:␣Employee
␣␣␣␣select␣COUNT(*)␣}"}
```

Chapitre 7

Niveau 7 — Persister et surveiller la base

Difficulté : Avancé ●●●●○

7.1 Briefing

Le serveur tourne depuis 3 jours. Vous devez :

1. Sauvegarder l'état courant de la base sur disque.
2. Configurer le rechargement automatique au redémarrage.
3. Mettre en place un mode de sauvegarde rapide (format binaire).

7.2 Le WAL : Write-Ahead Log

Chaque modification dans CatrustDB est enregistrée dans un fichier WAL (JSON Lines, avec checksum xxh3). C'est la source de vérité primaire.

7.2.1 Persister l'état courant (WAL)

```
{"cmd":"persist","auth":"admin-9kLmN",
 "path":"/var/lib/catrustdb/widgetcorp.wal"}
```

Résultat attendu

```
{"ok":true,"path":"/var/lib/catrustdb/widgetcorp.wal",
 "encrypted":true}
```

Le fichier est chiffré avec AES-256-GCM si `-wal-key` a été fourni.

7.2.2 Compacter le WAL

Au fil du temps, le WAL accumule des entrées. La compaction ne conserve que les lignes actives :

```
{"cmd":"compact","auth":"admin-9kLmN",
 "path":"/var/lib/catrustdb/widgetcorp.wal"}
```

7.2.3 Recharger depuis le WAL au redémarrage

```
{"cmd":"from_wal","auth":"admin-9kLmN",
 "path":"/var/lib/catrustdb/widgetcorp.wal"}
```

7.3 Snapshot V2 — sauvegarde haute performance

Le format binaire V2 (disponible depuis v0.8) offre des performances nettement supérieures grâce au lazy indexing :

Opération	Format V2	WAL JSON
Persist (1 M lignes)	**93 ms**	~1 250 ms
Chargement (1 M lignes)	**~1,8 ms**	~1 250 ms

7.3.1 Via CLI

```
# Cr er un snapshot V2
catrust snapshot --db /var/lib/catrustdb/widgetcorp.db \
                 --out /var/lib/catrustdb/snap.v2

# Restaurer depuis un snapshot V2
catrust rollback --db /var/lib/catrustdb/widgetcorp.db \
                 --snap /var/lib/catrustdb/snap.v2
```

7.3.2 Via protocole JSON (pgwire/Flight)

```
// Sauvegarde V2 (13,5x plus rapide que WAL)
{"cmd":"persist_v2","auth":"admin-9kLmN",
 "path":"/var/lib/catrustdb/snapshot.v2"}

// Rechargement V2 avec lazy indexing (~700x plus rapide)
{"cmd":"from_v2","auth":"admin-9kLmN",
 "path":"/var/lib/catrustdb/snapshot.v2"}
```

Les fichiers V2 commencent par le magic header `CATRUSTDBV2\x00`. Les index sont construits à la première requête (lazy) plutôt qu'au chargement.

Note

Compatibilité ascendante. Les commandes `persist_bin` / `from_bin` (V1) restent fonctionnelles ; V2 est le format recommandé pour les nouvelles installations.

7.4 Multi-bases simultanées

CatrustDB peut gérer plusieurs bases nommées en mémoire :

```
// Sauvegarder l'etat actuel sous le nom "production"
{"cmd":"db_save","auth":"admin-9kLmN","name":"production"}

// Charger un autre schema (dev/test)
{"cmd":"load","auth":"admin-9kLmN","schema_cql":"..."}
{"cmd":"db_save","auth":"admin-9kLmN","name":"dev"}

// Lister les bases disponibles
{"cmd":"db_list","auth":"admin-9kLmN"}
// -> {"ok":true,"databases":["production","dev"],"count":2}

// Revenir sur production
{"cmd":"db_use","auth":"admin-9kLmN","name":"production"}
```

Exercice

Simulez une procédure de backup quotidien :

1. Créez un snapshot V2 : `catrust snapshot` dans `/tmp/backup-YYYYMMDD.v2`.
2. Sauvegardez aussi en WAL chiffré : `persist` dans `/tmp/backup-YYYYMMDD.wal`.
3. Simulez un redémarrage (tuer le serveur, relancer) et restaurez depuis le snapshot V2 avec `catrust rollback`.
4. Vérifiez avec `stats` que toutes les données sont là.

Chapitre 8

Niveau 8 — Boss final — Audit de sécurité RSSI

Difficulté : Expert ●●●●●

8.1 Briefing

Le RSSI (Responsable Sécurité des Systèmes d'Information) débarque avec sa checklist ISO 27001 et vous demande de justifier chaque point :

A.1 Chiffrement des données en transit

A.2 Chiffrement des données au repos

A.3 Contrôle d'accès par rôle

A.4 Journalisation des accès

A.5 Rotation des secrets

A.6 Mode dégradé (lecture seule)

A.7 Protection contre la surcharge

A.8 Confinement des accès fichiers

8.2 Réponse point par point

Point	Mesure	Commande de vérification
A.1 — Transit	TLS 1.3 via rustls	`-tls-cert server.crt -tls-key server.key`
A.2 — Repos	AES-256-GCM sur le WAL	`-wal-key "passphrase"` → `security_report.wal_encryption=true`

Point	Mesure	Commande de vérification
A.3 — RBAC	3 niveaux ro/rw/admin	`-token-ro / -token-rw / -token-admin`
A.4 — Audit	NDJSON {ts, peer, cmd, ok, ms}	`-audit-log /var/log/catrustdb.log -audit-log-max-mb 50`
A.5 — Rotation	Rotation clé WAL à chaud	`{"cmd":"rotate_wal_key","new_key":"..."}`
A.6 — Dégradé	Mode lecture seule	`-read-only` → `security_report.read_only=true`
A.7 — Surcharge	Semaphore tokio	`-max-connections 64 -max-payload-bytes 10485760`
A.8 — Fichiers	Path whitelist canonicalisé	`-data-dir /var/lib/catrustdb`

Table 8.1 – Checklist RSSI — CatrustDB v1.1

8.3 Le scénario d'attaque

Le RSSI simule 4 scénarios d'attaque. Résolvez-les.

8.3.1 Attaque 1 — Vol du disque

Un attaquant vole le disque dur et récupère le fichier WAL. *Quel est l'impact ?*

Résultat attendu

Avec `-wal-key`, chaque ligne est chiffrée en AES-256-GCM avec un nonce aléatoire. Sans la passphrase, le fichier est illisible.

8.3.2 Attaque 2 — Token compromis

Le token `rw` est compromis. *Quelles opérations sont désormais possibles ?*

Résultat attendu

Le token `rw` peut lire, écrire et migrer des données. Il ne peut *pas* : effectuer `rotate_wal_key` ni modifier la configuration du serveur. **Action corrective :** redémarrez le serveur avec un nouveau `-token-rw`.

8.3.3 Attaque 3 — Path traversal

Un attaquant envoie `"path":"../../etc/passwd"` dans une commande `persist`.

Résultat attendu

La fonction `check_path()` canonicalise le chemin et vérifie qu'il est sous le préfixe `-data-dir`. Réponse du serveur :

```
{"ok":false,"error":"path ’../../etc/passwd’ hors du repertoire
    autorise"}
```

8.3.4 Attaque 4 — Timing attack sur le token

Un attaquant essaie de deviner le token en mesurant les temps de réponse.

Résultat attendu

La comparaison de tokens utilise `constant_time_eq()` :

```
fn constant_time_eq(a: &[u8], b: &[u8]) -> bool {
    if a.len() != b.len() { return false; }
    a.iter().zip(b.iter())
        .fold(0u8, |acc, (x, y)| acc | (x ^ y)) == 0
}
```

Le temps de réponse est identique quel que soit le token fourni : aucune information n'est divulguée par le canal temporel.

8.4 Exercice final — le rapport RSSI

Défi Boss

1. Démarrez le serveur avec la configuration de production complète (TLS + RBAC + WAL chiffré + audit log).
2. Envoyez `security_report` avec le token admin.
3. Vérifiez que *tous* les champs sont à `true` ou cohérents avec votre configuration.
4. Simulez la compromission du token rw :
   - Essayez `rotate_wal_key` avec le token rw — cela doit échouer.
   - Essayez `persist` avec le token rw — cela doit réussir.
5. Activez le mode `-read-only` (relancez le serveur). Essayez `delete` — cela doit échouer avec `"serveur en lecture seule"`.
6. Inspectez le fichier d'audit log : chaque tentative refusée doit y apparaître avec `"ok":false`.

Épilogue — Félicitations, DBA !

Vous avez parcouru les 7 niveaux de la mission DBA WidgetCorp.

Niveau	Ce que vous maîtrisez	Score
1 — Schéma	CQL, entités, FK, attributs	/20
2 — Instance	.cqli, symboles locaux, load	/20
3 — Requêtes	from/where/select, group_by, FK	/20
4 — Sécurité	TLS, RBAC, WAL chiffré, audit	/20
5 — Migration	Σ, mapping, Flyway, delta	/20
6 — RGPD	soft-delete, update, export	/20
7 — Boss RSSI	ISO 27001, attaques, audit trail	/20
Total		/140

Ce que ce tutoriel n'a pas couvert

Il reste des fonctionnalités avancées à explorer :

`sigma_dry` Simulation de migration sans écriture — parfait pour valider un mapping avant de le lancer en production.

Lineage `lineage` Historique complet des migrations Σ avec statistiques par entité et durée.

catrust CLI Génération SQL/Cypher, visualisation, introspection PostgreSQL, serve pgwire (compatible DBeaver).

MCP routes `GET /mcp/schema-text` et `GET /mcp/tools` pour intégrer Catrust dans un agent LLM (OpenAI/Anthropic).

Multi-bases Gestion de plusieurs bases nommées simultanément (`db_save` / `db_use` / `db_drop`).

La route vers v1.2

Arrow Flight Stream `RecordBatch` vers Python/Polars.

Client Python `pip install catrustdb` — PyO3/maturin.

Studio Web Interface onglets Données / Requêtes / Migrations.

Fuzzing `cargo-fuzz` sur le parseur CQL et le protocole NDJSON.

Δ_F Le pullback (migration inverse) — adjonction $\Delta_F \dashv \Sigma_F$.

Annexe A

Éditions Community et Enterprise

Ce chapitre décrit les différences concrètes entre les deux éditions lors de l'exécution du scénario *Mission DBA*.

A.1 Scénario Community

L'édition Community est **gratuite**, sans clé, et couvre l'intégralité des 7 niveaux du tutoriel avec les contraintes suivantes :

— **5 entités** maximum par base chargée.

— **100 000 lignes** maximum au total.

— Auth par **token unique** (`-token`) — pas de RBAC multi-rôle.

— **Pas de chiffrement WAL** (`-wal-key` ignoré).

— Commandes RGPD limitées : `soft_delete`, `update`, `export_csv`.

Installation Community

```
# Linux / macOS -- installation en une ligne
curl -sSfL https://catrust.eu/releases/install.sh | sh

# Windows -- Telecharger le .msi
# https://catrust.eu/download.html

# Verification de l'edition active
catrust --version
# catrust 1.0.0 (community)
```

Démarrer le serveur en Community

```
catrust serve --port 7878
# ou :
catrustdb --port 7474 --token "mon-secret"
```

Le serveur affiche :

```
[catrustdb] edition community (org: community)
[catrustdb] INFO edition community — certaines fonctionnalites necessitent
            une licence Enterprise
```

Atteindre les limites — message d'erreur

Si le schéma WidgetCorp dépasse 5 entités, la commande `load` retourne :

```
{
  "ok": false,
  "error": "limite community 'max_entities' depassee (6/5)
            — mise a niveau vers Enterprise"
}
```

Fonctionnalités Enterprise non disponibles

Tenter d'utiliser une fonctionnalité Enterprise en Community retourne :

```
{
  "ok": false,
  "error": "fonctionnalite Compliance necessite une licence Enterprise
            — contacter support@catrust.eu"
}
```

A.2 Scénario Enterprise

L'édition Enterprise **lève toutes les limites** et débloque les fonctionnalités avancées de sécurité et de conformité.

Activer la licence

```
# Via variable d'environnement (recommand  en production)
export CATRUST_LICENSE_KEY="<votre-cl  -ed25519>"

# Ou directement au d marrage
catrustdb --port 7474 \
          --license-key "<votre-cl  >" \
          --data-dir ./data
```

Message de démarrage avec licence valide :

```
[catrustdb] edition enterprise (org: WidgetCorp)
[catrustdb] authentification activee (RBAC)
```

Niveau 4 — Sécurité Enterprise

En Enterprise, le niveau 4 se déroule différemment :

```
# RBAC multi-role
catrustdb --port 7474 \
          --license-key "$CATRUST_LICENSE_KEY" \
          --token-admin "$ADMIN_SECRET" \
          --token-rw    "$RW_SECRET" \
          --token-ro    "$RO_SECRET" \
          --wal-key     "passphrase-forte-argon2" \
          --tls-cert    ./certs/server.crt \
          --tls-key     ./certs/server.key \
          --audit-log   ./logs/audit.jsonl \
          --audit-log-max-days 90
```

Les trois rôles possèdent des droits distincts :

`-token-admin` Lecture, écriture, `db_drop`, `rotate_wal_key`.

`-token-rw` Lecture et écriture des données, migrations.

`-token-ro` Lecture seule — idéal pour les rapports / dashboards.

Rotation de clé WAL

```
{"cmd":"rotate_wal_key",
 "old_key":"passphrase-courante",
 "new_key":"nouvelle-passphrase"}
```

Niveau 6 — RGPD Enterprise

Les commandes RGPD étendues ne sont disponibles qu'en Enterprise :

Anonymisation complète (Art. 17 + Art. 25 RGPD)

```
{"cmd":"anonymize",
 "entity":"Employee",
 "row_id":"e1",
 "fields":["emp_name","salary"]}
-> {"ok":true, "anonymized":["emp_name","salary"]}
```

Accès aux données personnelles (Art. 15)

```
{"cmd":"data_subject_access", "subject_id":"e1"}
-> {"ok":true, "entity":"Employee", "row":{...}}
```

Export portabilité (Art. 20)

```
{"cmd":"data_subject_export",
 "subject_id":"e1",
 "format":"json"}
-> {"ok":true, "export":{...}}
```

Rapport de conformité (DPO)

```
{"cmd":"compliance_report"}
-> {
  "ok": true,
  "edition": "enterprise",
  "soft_deletes": 2,
  "anonymizations": 1,
  "wal_encrypted": true,
  "tls_enabled": true,
  "audit_log": "./logs/audit.jsonl"
}
```

Déploiement Enterprise — Docker Compose

```
version: "3.9"
services:
  catrustdb:
    image: catrustdb/catrust:enterprise-1.0.0
    ports: ["7474:7474"]
    environment:
      CATRUST_LICENSE_KEY: "${CATRUST_LICENSE_KEY}"
      CATRUST_TOKEN_ADMIN:  "${CATRUST_TOKEN_ADMIN}"
      CATRUST_TOKEN_RW:     "${CATRUST_TOKEN_RW}"
      CATRUST_TOKEN_RO:     "${CATRUST_TOKEN_RO}"
      CATRUST_WAL_KEY:      "${CATRUST_WAL_KEY}"
    volumes:
      - ./data:/data
      - ./certs:/certs:ro
      - ./logs:/logs
    command: >
      catrustdb --port 7474
                --data-dir /data
                --tls-cert /certs/server.crt
                --tls-key  /certs/server.key
                --audit-log /logs/audit.jsonl
```

```
--audit-log-max-days 90
```

Tableau comparatif des éditions

Fonctionnalité	**Community**	**Enterprise**
Schéma CQL / Migrations ΣΔΠ	✓	✓
Serveur TCP NDJSON	✓	✓
Studio Web (port 7878)	✓	✓
Export CSV / JSONL / Parquet	✓	✓
TLS (rustls)	✓	✓
Arrow Flight / pgwire	✓	✓
Limite d'entités	5	Illimité
Limite de lignes	100 000	Illimité
RBAC multi-rôle	—	✓
WAL chiffré AES-256-GCM	—	✓
Rotation de clé WAL	—	✓
RGPD : `anonymize`, `compliance_report`	—	✓
Support commercial	—	✓
Licence	Gratuite	Clé Ed25519 signée

Licence Enterprise : support@catrust.eu

Catrust — La base de données comme foncteur.