Documentation OLAP

1. Description des collections NoSQL

A. Collection : clickstream_events

Utilité : analyse comportementale, scoring, personalisation, détection anomalie navigation.

Structure d'un document


```
{
  "event_id": "uuid",
  "user_id": "12345",
  "session_id": "sess-987",
  "event_type": "page_view",
  "page": "/checkout",
  "timestamp": "2025-11-18T10:14:02Z",
  "device": {
    "os": "iOS",
    "type": "mobile",
    "browser": "Safari"
  },
  "geo": {
    "country": "FR",
    "ip": "192.168.0.1"
  }
}
```


Index recommandés

- { user_id, timestamp }
- { session_id }
- { event_type }

B. Collection : user_sessions

Utilité : regrouper les sessions, utile pour ML, churn, analyse d'audience.

Structure


```
{
  "session_id": "sess-987",
  "user_id": "12345",
  "start_at": "2025-11-18T10:00:00Z",
  "end_at": "2025-11-18T10:23:00Z",
  "device": { ... },
  "metadata": {
    "session_quality_score": 0.82
  }
}
```


Stratégie

- Embedding des données utiles dans la session (device)
- Référencement vers clickstream_events

C. Collection : system_logs

Utilité : logs d'accès, erreurs, monitoring sécurité, audit PCI-DSS.

Exemple de document


```
{
  "log_id": "uuid",
  "timestamp": "2025-11-18T10:14:02Z",
  "level": "ERROR",
  "service": "payment-api",
  "message": "Timeout on external bank API",
  "request_id": "req-554",
```

"ip": "10.0.4.12"

}

Index

- { timestamp }
- { level }
- { service }

D. Collection : ml_features

Utilité : Feature Store NoSQL temps réel
Source : OLTP (CDC), Clickstream, Logs, Modèles ML

Document type

{

"user_id": "12345",

"last_update": "2025-11-18T10:15:00Z",

"features": {

"velocity_1h": 12,

"avg_tx_amount_7d": 32.5,

"device_risk": 0.4,

"session_depth": 14,

"fraudulent_interactions": 0

}

}

Stratégies

- TTL sur les features périssables
- Aggregation pipeline (Kafka → ML → NoSQL)

E. Collection : fraud_events

Utilité : scores ML, alertes, décisions en temps réel.

```
{
 "event_id": "uuid",
 "transaction_id": "tx_678",
 "user_id": "12345",
 "fraud_score": 0.97,
 "model_version": "fraud_v8",
 "timestamp": "2025-11-18T10:14:55Z",
 "explanations": {
  "velocity_1h": 0.9,
  "ip_anomaly": 0.4
 }
}
```


Index

- { transaction_id }
- { user_id, timestamp }
- { fraud_score}

F. Collections miroir : customer_ref & transaction_ref

Utilité :

- Éviter d’appeler OLTP pour chaque accès
- Garder des info "non sensibles" disponibles pour ML/logs

Exemple customer_ref


```
{
 "customer_id": "12345",
 "country": "FR",
 "created_at": "2020-02-10",
 "segment": "pro"
}
```

Stratégie

- Maintenues via CDC
- Données minimales (RGPD)

---

2. Patterns de modélisation utilisés

✓ **Embedding** pour éviter les JOIN (device, geo, mini-features)
✓ **Referencing** pour tout ce qui change rapidement (transaction → fraud event)
✓ **Time-series collections** pour :

- logs
- clickstreams
- fraud events

✓ **TTL** pour :

- features trop anciennes
- sessions terminées
- logs non critiques

✓ **Sharding** basé sur :

- user_id (distribution large)
- timestamp (time-series partitionnement)

---

3. Relations logiques entre collections

Collection	Relation	Type	Justification
user_sessions → clickstream	1:n	referencing	haute cardinalité
customer_ref → ml_features	1:1	embedding partiel	lookup rapide
transaction_ref → fraud_events	1:n	referencing	volume élevé
clickstream → ml_features	n:1	materialized features	pipeline ML

---

4. Fonctionnement avec les pipelines (intégration)

- Kafka → NoSQL pour ingestion en streaming
- Airflow → NoSQL pour traitements batch ML
- CDC OLTP → collections miroir
- ML inference → écrit dans fraud_events
- Clickstream → enrichit ml_features