

AIA Bloc_02 : stripe

Security & Governance

GDPR | CCPA | PCI-DSS Recommandations Stratégiques

Vue d'Ensemble

Pour une plateforme de paiement internationale comme Stripe, la conformité réglementaire n'est **pas une option** mais un **impératif business**. Trois réglementations majeures structurent notre architecture de données :

Réglementation	Juridiction	Impact Prioritaire	Pénalités Maximales
GDPR	Union Européenne (27 pays)	Protection données personnelles	4% CA global ou €20M
CCPA	Californie (USA)	Transparence & opt-out	\$7,500 par violation
PCI-DSS	Mondial (paiements carte)	Sécurité données bancaires	Retrait licence + amendes

Principe Directeur

"Privacy & Security by Design" : La conformité doit être **intégrée dès la conception** de l'architecture, pas ajoutée après coup.

Comparatif GDPR vs CCPA : les différences

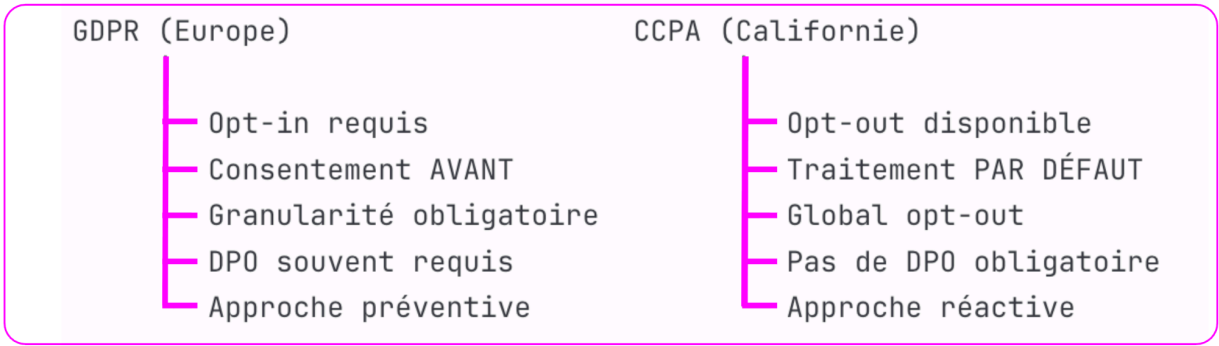




Tableau Synoptique des Exigences

Aspect	GDPR 	CCPA 	Recommandation Stripe
Consentement	Opt-in explicite	Opt-out sur demande	Implémenter les deux selon géolocalisation
Base légale	6 bases légales définies	Notification suffisante	Documenter base légale par traitement
Définition "vente"	Transfert contrôle données	Partage tiers (même gratuit)	Mapper tous les partages tiers
Droits individus	8 droits (accès, rectification, etc.)	5 droits principaux	Unifier dans portail self-service
Délai réponse	1 mois (extensible à 3)	45 jours (extensible à 90)	Target interne : 15 jours
Portée territoriale	Résidents UE partout	Résidents CA (même hors CA)	Détection géographique IP
UI obligatoire	Cookie banner (opt-in)	Lien "Do Not Sell" (opt-out)	Design adaptatif selon région
Amendes	Jusqu'à 4% CA global	\$2,500-\$7,500 par violation	GDPR = risque financier majeur

Recommandations par Composant d'Architecture

1. OLTP (PostgreSQL) - Source de Vérité

Objectif : Stocker PII de manière sécurisée avec capacité GDPR/CCPA

Recommandations Prioritaires

Données Sensibles

- Chiffrement at-rest : tablespaces PostgreSQL chiffrés (AES-256)
- Chiffrement colonnes : **pgcrypto** pour email, phone, adresse
- Tokenization : Cartes bancaires → tokens (jamais PAN complet)
- Row-Level Security : Isolation multi-tenant par merchant_id

Gestion Consentements




- Table dédiée **customer_consent** avec :
 - Détection juridiction automatique
 - Flags GDPR (opt-in granulaire par purpose)
 - Flags CCPA (opt-out global vente)
 - Versioning consentements (traçabilité)
 - Timestamps précis (audit)

Droits Individus




- **Procédures stockées pour :**
 - Export données (Right to Access)
 - Anonymisation (Right to Erasure - attention : pas suppression totale)
 - Rectification (Right to Rectification)
- **Logs de toutes les opérations GDPR/CCPA**

Architecture Données PCI-DSS

Jamais stocker :

-  Numéro carte complet (PAN)
-  CVV/CVV2/CVC2
-  Données piste magnétique

Autorisé à stocker :

-  Token (ex: **tok_visa_4242**)
-  **4 derniers chiffres**
-  **Date expiration**
-  **Fingerprint (hash pour déduplication)**

2. Streaming Layer (Kafka) - Données en Transit

Objectif : Capturer événements en temps réel tout en protégeant PII

Recommandations Prioritaires

Pseudonymisation Avant Kafka

- **Hash emails** → `SHA256(email + salt + year)`
- Mask IP addresses → Conserver subnet uniquement (`185.123.0.0/16`)
- Truncate timestamps → Précision minute (pas milliseconde)
- Suppress device fingerprints → Remplacer par hash

Sécurité Kafka

- SSL/TLS pour tous les brokers
- SASL authentication (SCRAM-SHA-512)
- ACLs par topic (qui peut lire/écrire)
- Logs chiffrés sur disque

Topics de Consentement

- Topic dédié : `customer.consent.changed`
- Propagation temps réel des préférences GDPR/CCPA
- Consommation par tous les services downstream
- Retention : 90 jours (audit)

3. Data Lake (S3) - Stockage Intermédiaire

Objectif : Stocker données brutes/transformées avec zones sécurisées

Recommandations Prioritaires

Zone-Based Security Model

```
S3 Bucket: stripe-datalake
|
|— /raw-zone/
|   |— Encryption: SSE-KMS (clés par région)
|   |— Access: Data Engineers uniquement (IAM strict)
|   |— Retention: 90 jours puis archivage Glacier
|   |— Contient PII pseudonymisées
|
|— /clean-zone/
|   |— Encryption: SSE-KMS
|   |— Access: Data Engineers + Data Scientists
```

```
|
|   | Retention: 1 an
|   | PII hashées, validées
|
|   /curated-zone/
|   | Encryption: SSE-S3 (suffisant car anonymisé)
|   | Access: Analytics Team + BI Tools
|   | Retention: 7 ans (compliance financière)
|   | Données ANONYMISÉES (pas de PII identifiables)
```

✓ Bucket Policies

- Bloquer accès non-TLS (force HTTPS)
- IP whitelisting (VPN corporate uniquement)
- MFA delete (protection suppression accidentelle)
- Versioning activé (récupération en cas erreur)

✓ Anonymisation au Pipeline ETL

- **AVANT écriture dans curated-zone :**
 - Généralisation (âge → tranche d'âge)
 - Agrégation (K-anonymity, $k \geq 5$)
 - Suppression PII directes (email, phone, adresse)
 - Hash irréversible des identifiants

4. OLAP (Snowflake) - Analytics & Reporting

Objectif : Zéro PII identifiables, données 100% anonymisées

Recommandations Prioritaires

Principe Fondamental

⚠ OLAP ne doit JAMAIS contenir de PII réversibles

Contrairement à OLTP qui peut chiffrer, OLAP doit anonymiser irréversiblement.

Tables Dimensions - Données Anonymisées

✗ NE JAMAIS stocker dans OLAP :

- Email complets
- Noms/Prénoms
- Adresses complètes
- IP addresses complètes

- Numéros de téléphone
- Dates de naissance exactes

STOCKER dans OLAP :

- Domaines email (ex: @gmail.com)
- Tranches d'âge (ex: 25-34)
- Régions (ex: Ile-de-France)
- Préfixes codes postaux (ex: 750)
- Hash clients (SHA256(customer_id))
- Années (pas dates complètes)

Contrôles d'Accès Snowflake

- Network policies (VPN uniquement)
- MFA obligatoire (tous les utilisateurs)
- SSO via SAML (Okta/Azure AD)
- Row Access Policies (isolation merchant)
- Audit logs (365 jours minimum)

Pas de Dynamic Masking pour GDPR

- ✗ Le masking dynamique est insuffisant (données en clair en base)
 - Anonymisation irréversible au pipeline ETL
 - Approche conforme : Data minimization dès la conception
-

5. NoSQL (MongoDB) - Flexibilité & Logs

Objectif : Stocker données semi-structurées avec conformité

Recommandations Prioritaires

Field-Level Encryption

- Client-Side Field Level Encryption (CSFLE)
- PII chiffrées : email, phone dans customer_360
- Clés gérées via KMS (AWS KMS / Azure Key Vault)
- Rotation clés automatique (90 jours)

Collections Spécialisées

Logs (Time-Series) :

- TTL automatique (30-90 jours)

- IP addresses masquées
- Pas de PII en clair

Fraud Signals :

- Features anonymisées pour ML
- Customer hash (pas ID direct)
- Scores et décisions seulement

Customer 360 :

- Vue agrégée (pas données brutes)
- Chiffrement colonnes sensibles
- Suppression automatique si GDPR deletion

Sharding & Access Control

- Sharding par customer_hash (distribution)
- RBAC MongoDB (rôles par use case)
- Audit logs activés (tous les accès)
- Connection chiffrée (TLS 1.3)

Consent Management Platform (CMP) - Cœur de la Conformité

Architecture Recommandée

Consent Management Platform (CMP)

1

DÉTECTION JURIDICTION

- Géolocalisation IP (GeoIP2)
- Détermination automatique : EU / CA / UK / Other
- Fallback : Applique règle la plus stricte (EU)

2

INTERFACE ADAPTATIVE

- EU → Cookie banner (opt-in obligatoire)
- CA → Lien "Do Not Sell" (opt-out disponible)
- Multilangue (détection locale)
- Mobile-responsive

3

STOCKAGE CONSENTEMENTS

- Base OLTP (table customer_consent)
- Versioning (historique complet)
- Granularité par purpose (GDPR)
- Global opt-out sale (CCPA)

4

PROPAGATION TEMPS RÉEL

- Event Kafka → Tous les services
- Cache Redis (low-latency checks)
- Notification partenaires externes si opt-out

5

APIS SELF-SERVICE

- GET /privacy/my-data (Right to Access)
- POST /privacy/update-consent (Modification)
- DELETE /privacy/delete-me (Right to Erasure)
- POST /privacy/ccpa/opt-out (CCPA specific)

Solutions CMP Recommandées

Solution	Avantages	Inconvénients	Recommandation
OneTrust	Leader marché, GDPR+CCPA+50+ lois	Coûteux (\$50k+/an)	✅ Entreprise >500M\$ CA
Cookiebot	Simple, bon rapport qualité/prix	Moins de features avancées	✅ PME/Startup
TrustArc	Complet, bon support	Interface complexe	⚠️ Si besoin global compliance
Custom (maison)	Contrôle total, pas de coût licence	Dev/maintenance important	⚠️ Si ressources tech suffisantes

Recommandation Stripe : OneTrust (justifié par volume et criticité)

Droits des Individus - Implémentation Recommandée

Matrice des Droits GDPR + CCPA

Droit	GDPR	CCPA	Délai	Complexité	Priorité
Accès	✅	✅	30j/45j	Moyenne	🔴 Élevée
Rectification	✅	✅ (2023+)	30j	Faible	🟡 Moyenne
Suppression	✅	✅	30j/45j	Élevée	🔴 Élevée
Portabilité	✅	❌	30j	Moyenne	🟡 Moyenne
Opposition	✅	❌	Immédiat	Faible	🟡 Moyenne
Opt-out sale	N/A	✅	Immédiat	Moyenne	🔴 Élevée (CA)
Limitation	✅	❌	30j	Moyenne	🟢 Faible
Décision auto	✅	❌	30j	Faible	🟢 Faible

Workflow Recommandé : Right to Erasure (Exemple)

Processus de Suppression (Right to Erasure / Delete)

1

Requête Client

- Via portail self-service ou email DPO
- Vérification identité (2FA / email confirmation)
- Justification optionnelle

2

Évaluation Exceptions (< 48h)

- Obligations légales ? (ex: 7 ans transactions)
- Litiges en cours ?
- Intérêts légitimes impérieux ?
- Décision : Suppression totale / partielle / refus

3

Planification (si approuvé)

- Job asynchrone (Airflow DAG)
- Ordre : OLAP → NoSQL → Datalake → OLTP (inverse)
- Backups séparés (conformité légale 7 ans)

4

Exécution Multi-Systèmes

- Snowflake : DELETE customer records
- MongoDB : `db.customer_360.deleteOne()`
- S3 : Suppression fichiers (si identifiables)
- PostgreSQL : Anonymisation (pas suppression)
→ email = 'deleted_{id}@anonymized.local'

5

Confirmation & Audit

- Email confirmation client (< 30 jours)
- Log exhaustif (qui, quand, quoi, pourquoi)
- Notification DPO
- Preuve de conformité archivée

⚠ ATTENTION : Suppression ≠ Deletion complète

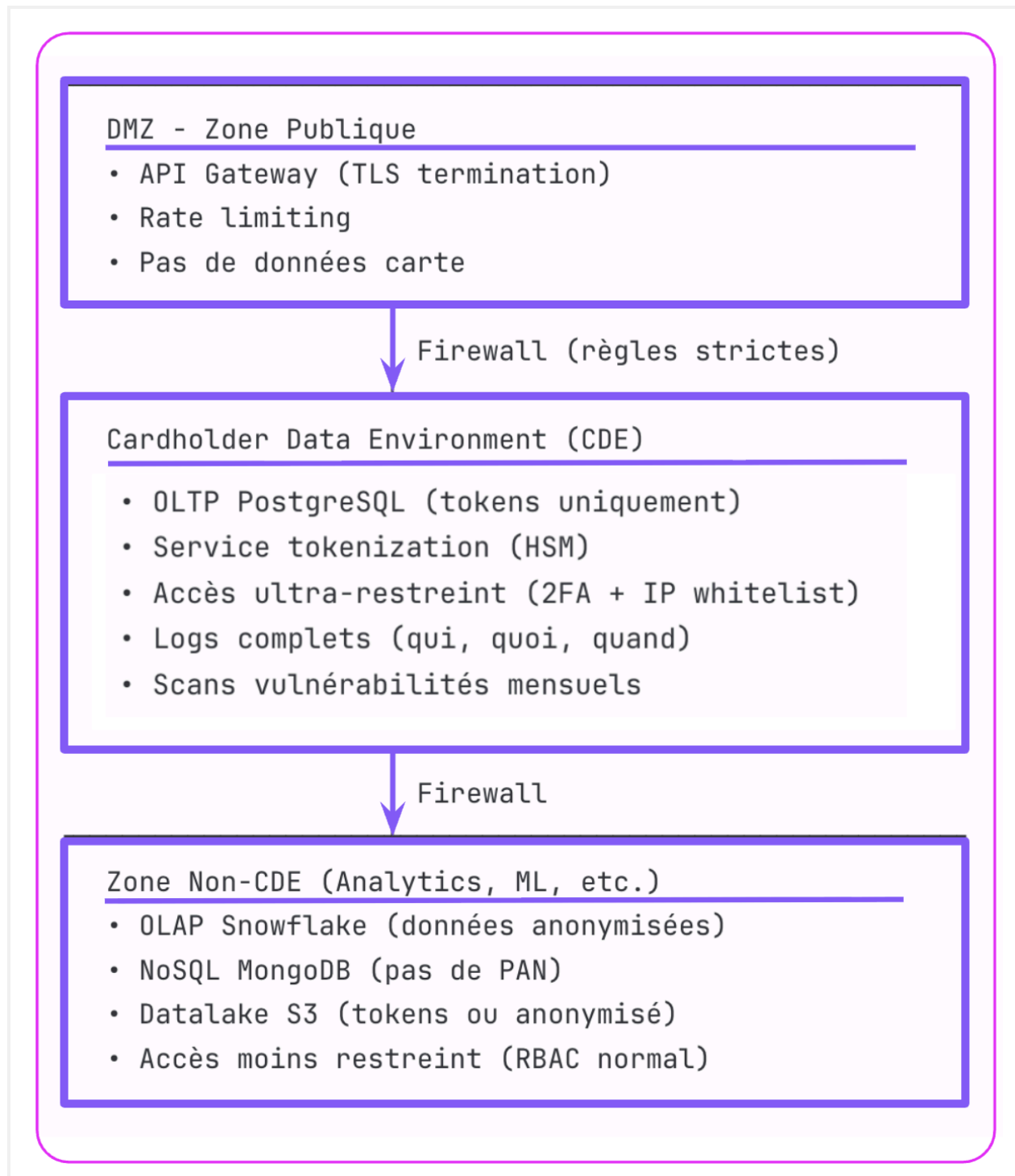
- Données transactionnelles : Anonymisation (pas suppression)
- Obligations légales : Conservation 7 ans (finance)
- Approche : PII supprimées, transactions anonymisées conservées

PCI-DSS - Sécurité Données Bancaires

Les 12 Exigences PCI-DSS (Synthèse)

	Exigence	Impact Architecture Stripe	Priorité
1	Pare-feu et segmentation réseau	VPC privés, pas d'accès Internet direct	●
2	Pas de mots de passe par défaut	Rotation secrets automatique	●
3	Protéger données cartes stockées	Tokenization (pas de PAN)	●
4	Chiffrer transmissions réseaux publics	TLS 1.3 obligatoire	●
5	Antivirus/Malware	Solutions endpoint protection	●
6	Systèmes et apps sécurisés	Scans vulnérabilités réguliers	●
7	Accès données sur besoin métier	RBAC strict, principe moindre privilège	●
8	Authentification forte	MFA obligatoire, SSO	●
9	Accès physique restreint	Datacenters Tier IV certifiés	●
10	Tracer accès ressources réseau	Audit logs 365 jours minimum	●
11	Tests sécurité réguliers	Pentests annuels + bug bounty	●
12	Politique sécurité info	Documentation + formation équipe	●

Architecture PCI-DSS : Network Segmentation



Exemple de Table pour gérer les consentements GDPR + CCPA

```

CREATE TABLE customer_consent (
    consent_id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    customer_id UUID NOT NULL REFERENCES
customer(customer_id),

    -- Détection jurisdiction
    jurisdiction VARCHAR(10) NOT NULL, -- 'EU', 'CA',
'US-OTHER', etc.
    ip_address INET,
    detected_at TIMESTAMP NOT NULL DEFAULT
CURRENT_TIMESTAMP,

    -- GDPR (EU) - Opt-in
    gdpr_consent_given BOOLEAN DEFAULT FALSE,
    gdpr_consent_timestamp TIMESTAMP,
    gdpr_purposes JSONB DEFAULT '{}', -- {"marketing":
true, "analytics": false}

    -- CCPA (California) - Opt-out
    ccpa_opt_out_sale BOOLEAN DEFAULT FALSE,
ccpa_opt_out_timestamp TIMESTAMP,
    ccpa_do_not_sell_requested BOOLEAN DEFAULT FALSE,

    -- Historique et audit
    consent_version VARCHAR(10), -- "v2.1" (version des
CGU)
    consent_method VARCHAR(50), -- "web_banner",
"api_call", "email"
    consent_text TEXT, -- Texte exact présenté

    -- Métadonnées
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

```

```

is_active BOOLEAN DEFAULT TRUE,

-- Constraints
CONSTRAINT chk_jurisdiction CHECK (jurisdiction IN
('EU', 'CA', 'US-OTHER', 'UK', 'OTHER')),
CONSTRAINT chk_gdpr_eu_only CHECK (
(jurisdiction = 'EU' AND gdpr_consent_given IS NOT
NULL) OR
(jurisdiction != 'EU')
),
CONSTRAINT chk_ccpa_ca_only CHECK (
(jurisdiction = 'CA' AND ccpa_opt_out_sale IS NOT
NULL) OR
(jurisdiction != 'CA')
)
);

-- Index
CREATE INDEX idx_consent_customer ON
customer_consent(customer_id, is_active);
CREATE INDEX idx_consent_jurisdiction ON
customer_consent(jurisdiction);
CREATE INDEX idx_consent_ccpa_optout ON
customer_consent(ccpa_opt_out_sale)
WHERE jurisdiction = 'CA';

-- Audit trigger
CREATE TRIGGER consent_audit AFTER UPDATE ON
customer_consent
FOR EACH ROW EXECUTE FUNCTION log_consent_change();

```

Techniques d'Anonymisation Conformes

1. Hashing Irréversible (pour Comptage)

```
-- ✗ Dans OLAP : NE PAS stocker
email: "jean.dupont@example.com"

-- ✔ Dans OLAP : Stocker hash
customer_hash: SHA256("jean.dupont@example.com" + salt)
→ "8f7a3b2c1d9e5f4a3b2c1d9e5f4a3b2c"

-- Usage analytique possible :
SELECT customer_hash, COUNT(*) as transaction_count
FROM fact_transaction
GROUP BY customer_hash; -- Comptage unique OK

-- Objectif : ne pas pouvoir retrouver l'email original !
```

Implementation dbt :

```
-- models/staging/stg_customers.sql
SELECT
  -- ✗ PAS d'email brut
  -- email,

  -- ✔ Hash irréversible
  {{ hash_pii('email') }} as customer_hash,

  -- ✔ Extraction domaine seulement (analytics)
  SPLIT_PART(email, '@', 2) as email_domain,

  -- ✔ Agrégation géographique
  country_code,

  -- ✔ Généralisation âge
  CASE
    WHEN age BETWEEN 18 AND 24 THEN '18-24'
```

```

    WHEN age BETWEEN 25 AND 34 THEN '25-34'
    WHEN age BETWEEN 35 AND 44 THEN '35-44'
    ELSE '45+'
  END as age_group

```

```
FROM {{ source('oltp', 'customer') }}
```

2. Agrégation (K-Anonymity)

--  Données individuelles (identifiables)

customer_id	age	city	salary
-----	-----	-----	-----
uuid_123	32	Paris	45000
uuid_456	28	Lyon	42000

--  Données agrégées (anonymes si $k \geq 3-5$)

age_group	city_region	avg_salary	customer_count
-----	-----	-----	-----
25-34	Ile-de-France	43500	127

Règle K-Anonymity :

- Chaque combinaison doit contenir au minimum **k individus** (généralement $k=5$)
- Si moins de k : **supprimer ou regrouper** davantage

```

-- dbt macro pour k-anonymity
{% macro ensure_k_anonymity(group_by_cols, k=5) %}
HAVING COUNT(*) >= {{ k }}
{% endmacro %}

```

```
-- Utilisation
```



```

SELECT
    age_group,
    country,
    COUNT(*) as customer_count,
    AVG(lifetime_value) as avg_ltv
FROM dim_customer
GROUP BY age_group, country
{{ ensure_k_anonymity(['age_group', 'country'], 5) }}

```

```

-- ❌ Trop précis (identifiable)
address: "123 Rue de Rivoli, 75001 Paris"
birth_date: "1990-03-15"
ip_address: "185.123.45.67"

-- ✅ Généralisé (anonyme)
postal_code_prefix: "750" -- Seulement 3 premiers chiffres
birth_year: 1990 -- Seulement l'année
ip_range: "185.123.0.0/16" -- Subnet seulement

```

```

-- Service séparé (Token Vault) - accès ultra-restreint
CREATE TABLE token_mapping (
    customer_token UUID PRIMARY KEY,
    customer_id_oltp UUID, -- Chiffré at rest
    created_at TIMESTAMP,
    accessed_count INTEGER
);

-- OLAP contient SEULEMENT le token
CREATE TABLE dim_customer (
    customer_key INTEGER PRIMARY KEY,
    customer_token UUID, -- ✅ Pas de PII
    age_group VARCHAR(10),

```

```

country_code VARCHAR(3),
segment VARCHAR(50)
-- ✗ PAS d'email, nom, adresse, etc.
);

```

Règles strictes :

- Token Vault = base séparée, accès DPO uniquement
- Logs exhaustifs de chaque accès
- Justification obligatoire (ticket GDPR)
- Retention: suppression token après X mois

Correction de OLAP

```

CREATE TABLE dim_customer (
    customer_key INTEGER PRIMARY KEY,

    -- Identifiant pseudonymisé (hash one-way)
    customer_hash VARCHAR(64) NOT NULL,

    -- Données agrégées/généralisées
    age_group VARCHAR(10),          -- "25-34" au lieu de
32
    country_code VARCHAR(3),        -- "FR" (OK, pas PII
seul)
    region VARCHAR(50),            -- "Ile-de-France"
    postal_code_prefix VARCHAR(3), -- "750" au lieu de
"75001"

    -- Comportement (anonyme)
    customer_segment VARCHAR(50),  -- "High_Value",
"Regular"
    lifetime_transactions INTEGER,

```

```

lifetime_value DECIMAL(18,2),
first_transaction_year INTEGER, -- 2023 (pas date
complète)

-- Métadonnées non-identifiantes
preferred_language VARCHAR(10),
currency VARCHAR(3),

-- ✗ SUPPRIMÉ (PII)
-- email VARCHAR(255),           ← INTERDIT
-- phone_number VARCHAR(20),     ← INTERDIT
-- full_name VARCHAR(255),       ← INTERDIT
-- birth_date DATE,              ← INTERDIT
-- ip_address INET,              ← INTERDIT

-- SCD Type 2
effective_from TIMESTAMP,
effective_to TIMESTAMP,
is_current BOOLEAN
);

```

Table FACT_TRANSACTION

```

CREATE TABLE fact_transaction (
    transaction_key BIGINT PRIMARY KEY,

    -- FKs vers dimensions anonymisées
    date_key INTEGER,
    customer_key INTEGER, -- → dim_customer (anonymisé)
    merchant_key INTEGER,

    -- ✓ Données transactionnelles (OK)
    amount DECIMAL(18,2),

```

```

currency_code VARCHAR(3),
status VARCHAR(20),

-- ✓ Géographie agrégée
country_code VARCHAR(3),
region VARCHAR(50),

-- ✗ SUPPRIMÉ (PII ou identifiable)
-- ip_address INET,                ← INTERDIT
-- device_fingerprint VARCHAR,    ← INTERDIT (unique
identifier)
-- email VARCHAR,                ← INTERDIT

-- ✓ Métriques agrégées OK
fraud_score DECIMAL(5,4),
processing_time_ms INTEGER
);

```

Airflow DAG avec Anonymisation

```

# tasks/anonymize_customer_data.py

def anonymize_pii(data):
    """Anonymisation irréversible avant chargement OLAP"""

    return {
        # Hash one-way (avec salt rotatif)
        'customer_hash': hashlib.sha256(
            (data['email'] + SALT +
             str(data['created_at'].year)).encode()
        ).hexdigest(),
    }

```

```

        # Extraction domaine seulement
        'email_domain': data['email'].split('@')[1] if
data['email'] else None,

        # Généralisation âge
        'age_group': get_age_group(data['age']),

        # Géolocalisation agrégée
        'country_code': data['country'],
        'postal_code_prefix': data['postal_code'][:3] if
data['postal_code'] else None,

        # Temporalité agrégée
        'first_transaction_year':
data['first_transaction_date'].year,

        # ❌ PII SUPPRIMÉES (pas chargées dans OLAP)
        # 'email': SUPPRIMÉ
        # 'phone': SUPPRIMÉ
        # 'full_address': SUPPRIMÉ
    }

# DAG Airflow
with DAG('daily_olap_load_gdpr_compliant') as dag:

    extract = PostgresOperator(
        task_id='extract_oltp',
        sql='SELECT * FROM customer WHERE updated_at >= {{
ds }}'
    )

    anonymize = PythonOperator(
        task_id='anonymize_pii',

```

```
        python_callable=anonymize_pii
    )

    load_snowflake = SnowflakeOperator(
        task_id='load_anonymized_to_olap',
        sql='COPY INTO dim_customer_staging FROM
@s3_anonymized_stage'
    )

    extract >> anonymize >> load_snowflake
```

...