



Stripe : Conception d'une architecture de données unifiée

Unification des systèmes OLTP, OLAP et NoSQL

Kévin Chatelain

Architecte en Intelligence Artificielle

Bloc 2 – « Concevoir et déployer
des architecture de données (pour l'IA) »

Jedha 2023/2024





Gestion des données intégrées

Présentation du cadre de l'entreprise

Contexte

Stripe repose sur la gestion de données massives pour offrir des services de paiement fluides et fiables.

Cependant, l'expansion mondiale de Stripe expose l'entreprise à des défis en matière de **scalabilité**, de **performance** et de **conformité réglementaire**.

Défis

- ✓ Concevoir une architecture unifiée et performante.
- ✓ Garantir l'intégrité des transactions et des analyses.
- ✓ Exploiter les données non structurées efficacement.
- ✓ Intégrer des pipelines fiables et synchronisés.
- ✓ Assurer la conformité avec le GDPR et PCI-DSS.

Problématique

Comment optimiser la gestion et l'intégration des données ?



Section 1: Gestion des données intégrées

Présentation des architectures



Gestion des données intégrées

Présentation des architectures

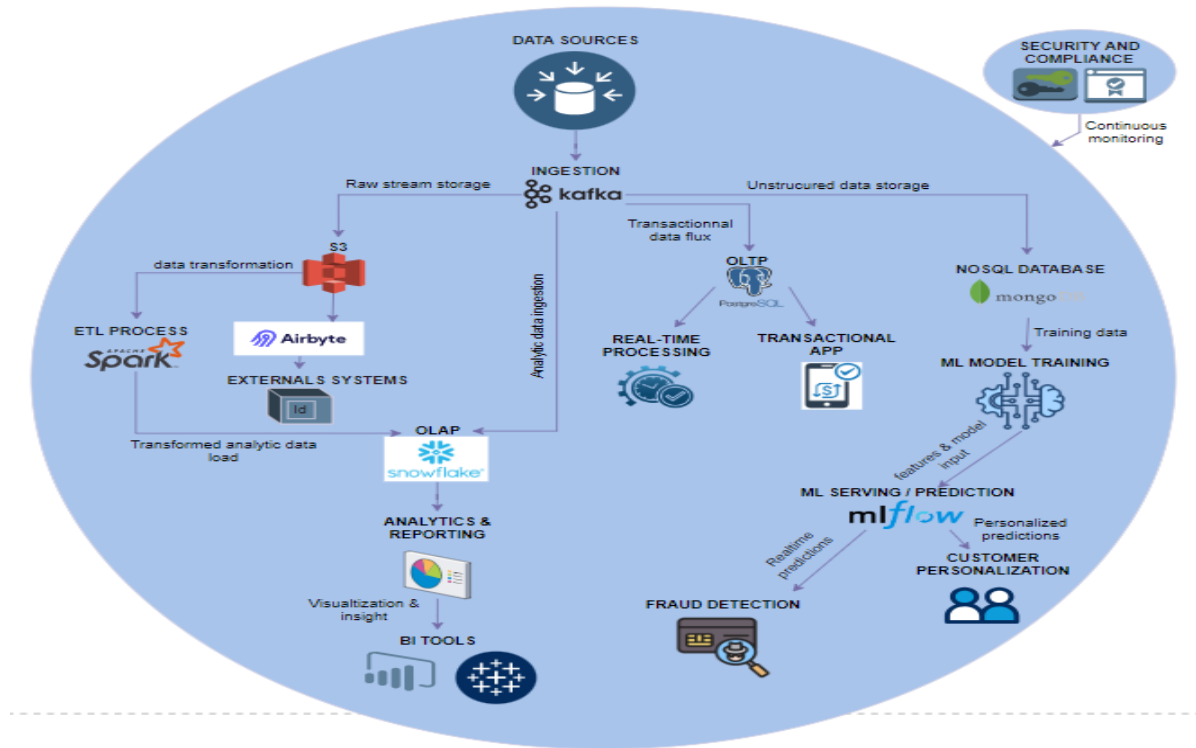
- Point 1 : Architecture de la gestion des données
 - .
- Point 2 : Architecture de l'OLAP
 - .
- Point 3 : Architecture de l'OLTP
 - .
- Point 4 : Architecture du NoSQL



stripe

Architecture de la gestion des données

Efficacité et conformité du flux d'information





Plan de sécurité et de conformité

Mesures pour une gestion sécurisée des données

1

☐ **Chiffrement des données**

Protection des données en transit et au repos via des protocoles de sécurité avancés (TLS, AES-256).

2

☐ **Contrôle d'accès et authentification**

Gestion des accès avec un contrôle basé sur les rôles (RBAC) et une authentification multi-facteurs (MFA).

3

☐ **Audit et traçabilité des accès**

Suivi des activités avec des journaux d'audit et alertes en cas d'accès suspect.

4

☐ **Sécurité des systèmes et services**

Protection des bases de données et des endpoints avec des configurations sécurisées et des pare-feu (WAF).

5

☐ **Surveillance en temps réel et alertes**

Mise en place de systèmes SIEM pour le suivi et la détection des menaces en continu.

6

☐ **Gestion des données sensibles et conformité**

Anonymisation et pseudonymisation des données pour se conformer aux réglementations (GDPR, PCI-DSS).

7

☐ **Plan de réponse aux incidents de sécurité**

Préparation et formations pour une réponse rapide et efficace aux incidents de sécurité.

8

☐ **Tests de sécurité et audits**

Réalisation de tests de pénétration réguliers et audits de conformité.



Architecture de la gestion des données

Avantages stratégique

- **Centralisation des données** : Regroupement des données de multiples sources pour un accès et une analyse facilités.
- **Traitement en temps réel et différé** : Kafka pour l'ingestion en temps réel, S3 pour le stockage brut, offrant flexibilité et rapidité.
- **Analyses avancées** : OLAP (Snowflake) pour des insights détaillés et une exploration multi-dimensionnelle.
- **Machine Learning intégré** : MongoDB et ML Model Training pour des prédictions et une personnalisation enrichies.



Architecture de la gestion des données

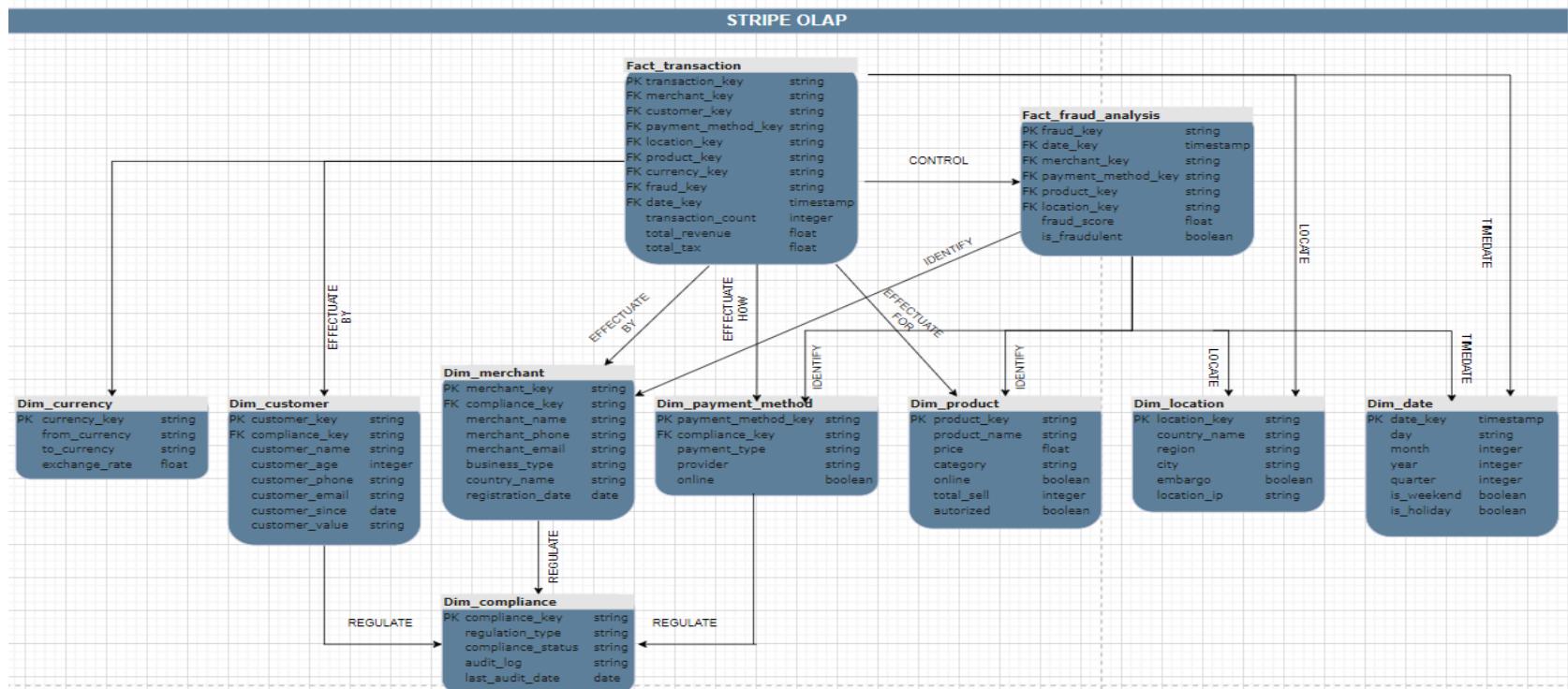
Avantages stratégique (suite)

- **Détection de fraude et personnalisation client** : Analyse en temps réel pour la fraude et des recommandations personnalisées.
- **Sécurité et conformité** : Surveillance continue pour respecter les réglementations à chaque étape.
- **Flexibilité et évolutivité** : Infrastructure scalable (Spark, Kafka, Snowflake) pour s'adapter aux besoins croissants de Stripe.



Architecture de l'OLAP

Analyses performantes et évolutives





stripe

Architecture de l'OLAP

Atouts clés

- **Analyse multi-dimensionnelle efficace** : Structure en tables de faits et dimensions pour des requêtes rapides et des insights détaillés.
- **Prévention de la fraude** : Table de faits dédiée pour détecter rapidement les comportements suspects.
- **Conformité intégrée** : Suivi des réglementations et audits facilités grâce à la dimension conformité.
- **Évolutivité** : Architecture flexible permettant d'ajouter de nouvelles dimensions pour s'adapter aux besoins changeants.



Architecture de l'OLAP

Exemple d'une requête

The screenshot shows a database management interface. On the left, a sidebar displays a tree view of the database structure under 'OLAP_STRIP_PROJECT'. The main area shows a SQL query being executed. The query is as follows:

```
1 SELECT
2   p.product_name,
3   p.category,
4   p.total_sell
5 FROM OLAP_STRIP_PROJECT.PUBLIC.Dim_product p
6 WHERE p.total_sell > 100;
```

Below the query, the 'Results' tab is active, displaying a table with 3 rows and 3 columns: PRODUCT_NAME, CATEGORY, and TOTAL_SELL.

	PRODUCT_NAME	CATEGORY	TOTAL_SELL
1	Laptop	Electronics	150
2	T-Shirt	Clothing	500
3	Apple	Groceries	10000

On the right side of the results table, there is a 'Query Details' section showing the following information:

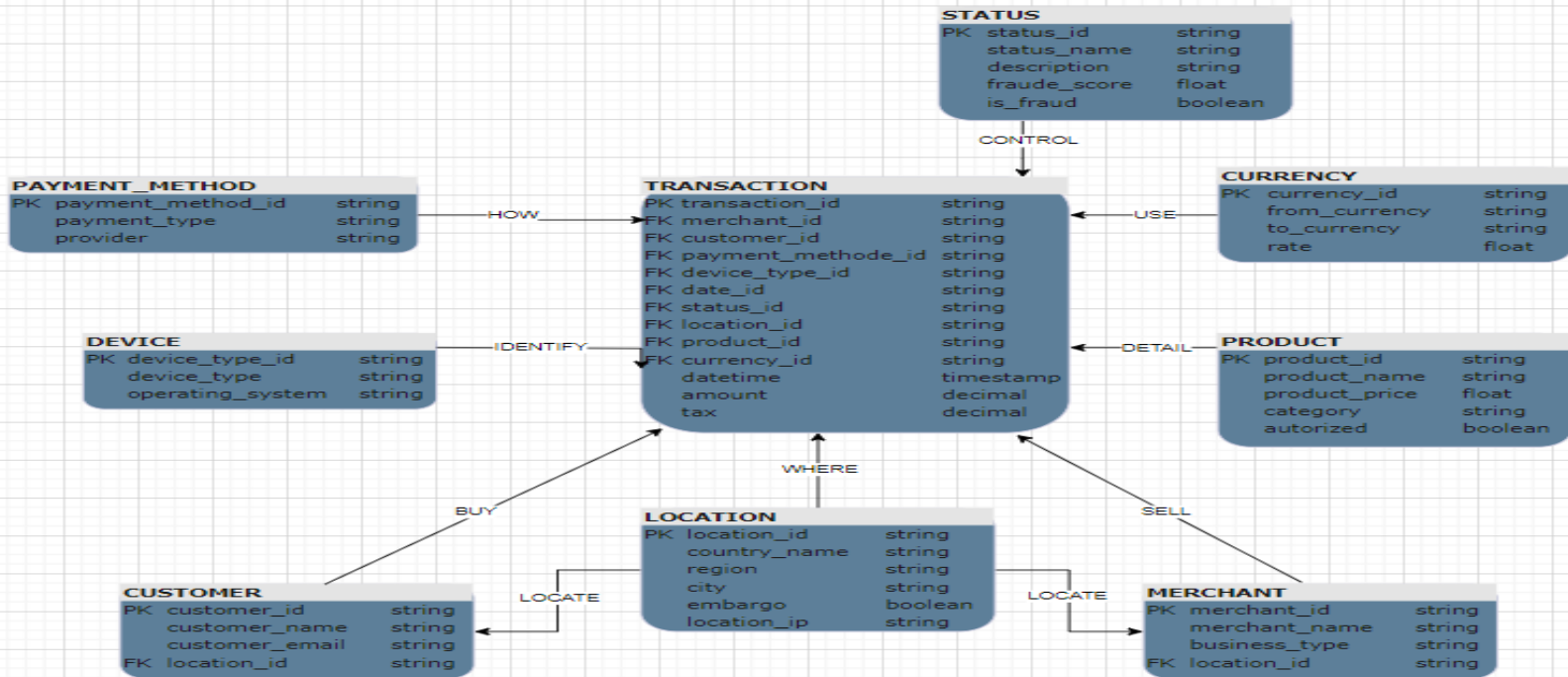
- Query duration: 124ms
- Rows: 3
- Query ID: 01b86581-0001-401b-0...



Architecture de l'OLTP

Intégrité transactionnelle

STRIPE OLTP





stripe

Architecture de l'OLTP

Exemple d'une requête

The screenshot shows a database management interface with a sidebar on the left containing a tree view of database objects. The main area displays a SQL query and its results. The query is a SELECT statement filtering transactions by amount. The results are shown in a table with 5 columns: TRANSACTION_ID, MERCHANT_ID, CUSTOMER_ID, AMOUNT, and TAX. Two rows of data are visible. Below the table, a 'Query Details' section shows a query duration of 174ms and 2 rows returned.

2024-11-15 11:18pm

Databases Worksheets

Search objects

- OLAP_STRIP_PROJECT
- OLTP_STRIP_PROJECT
 - INFORMATION_SCHEMA
 - PUBLIC
 - Tables
 - CURRENCY
 - CUSTOMER
 - DEVICE
 - LOCATION
 - MERCHANT
 - PAYMENT_METHOD
 - PRODUCT
 - STATUS
 - TRANSACTION
- SNOWFLAKE
- SNOWFLAKE_SAMPLE_DATA

OLTP_STRIP_PROJECT.PUBLIC Settings

```
1 SELECT transaction_id, merchant_id, customer_id, amount, tax
2 FROM OLTP_STRIP_PROJECT.PUBLIC.Transaction
3 WHERE amount > 100;
4
```

Results Chart

	TRANSACTION_ID	MERCHANT_ID	CUSTOMER_ID	AMOUNT	TAX
1	TXN001	MERCHANT001	CUSTOMER001	120.50	10.00
2	TXN003	MERCHANT003	CUSTOMER003	150.00	12.50

Query Details

Query duration 174ms

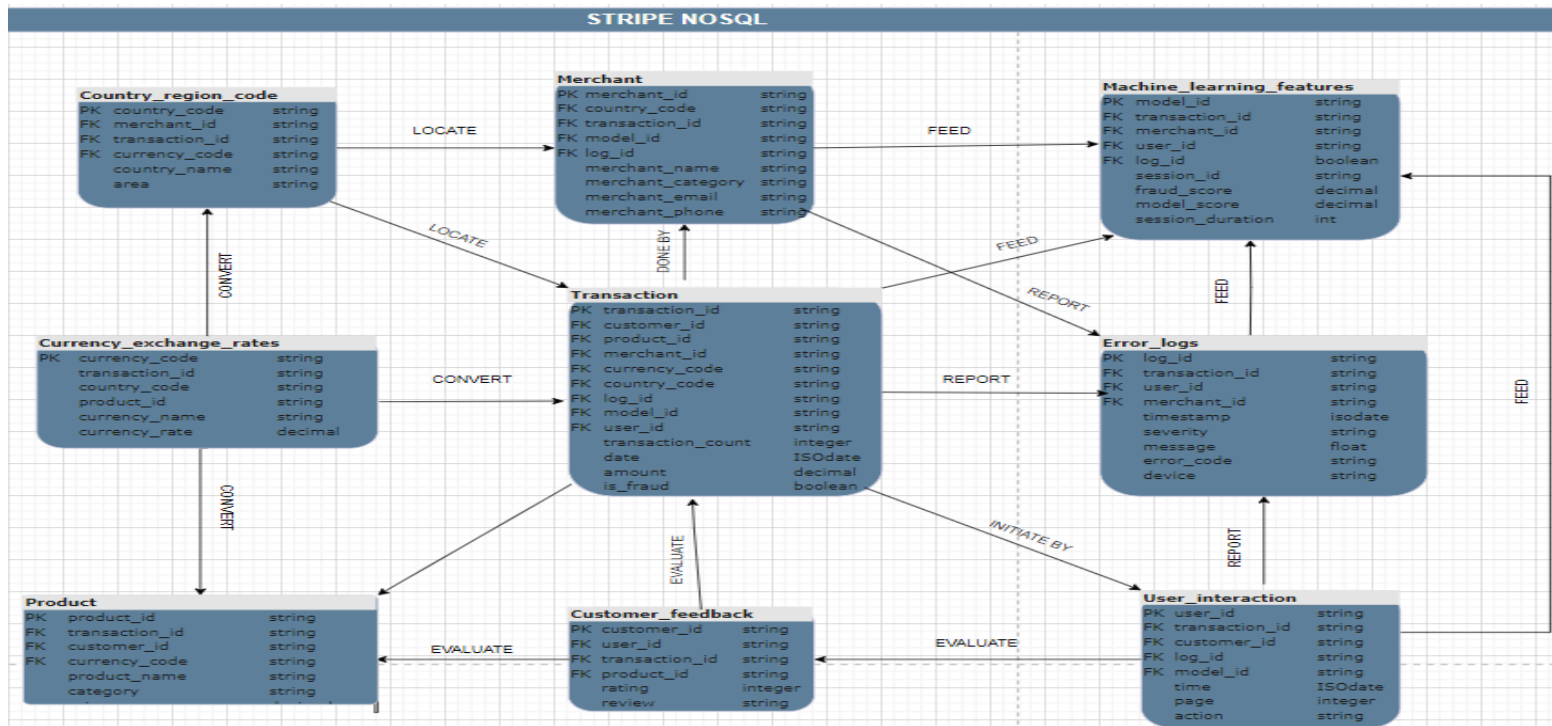
Rows 2



stripe

Architecture noSQL

Gestion flexible des données





stripe

Architecture de NoSQL

Exemple d'une requête

stripe_nosql

Transactions

country_region_codes

currency_exchange_ra...

customer_feedback

error_logs

machine_learning_feat...

merchants

products

user_interactions

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 2.28KB TOTAL DOCUMENTS: 4 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns 0 Aggregation Search Indexes

INSERT DOCUMENT

Filter {
 "amount": { "\$gt": 500 },
 "status": "fraudulent"
}

Reset Apply Options

QUERY RESULTS: 1-1 OF 1

```
_id: ObjectId('67111fb3ab30536f0dfd73e2')
transaction_id: "TX1005"
customer_id: "CUST005"
merchant_id: "MERCH003"
date: Object
amount: 890.5
currency: Object
fraud_score: 0.75
status: "fraudulent"
location: Object
```



Section 2: Machine learning & sécurité

Présentation des stratégies adoptées



stripe

Gestion des données intégrées

Présentation des architectures

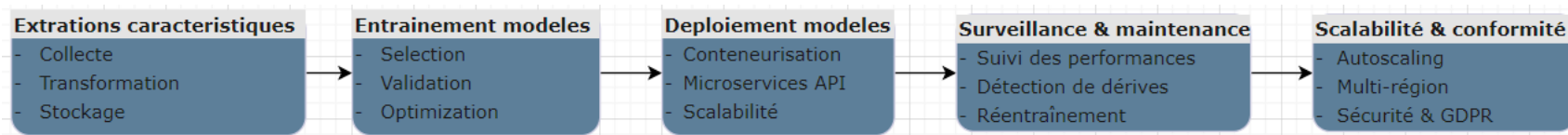
- Point 1 : Stratégie d'intégration Machine Learning
- Point 2 : Plan de sécurité et de conformité



stripe

Stratégie d'intégration Machine Learning

De l'extraction au monitoring des modeles



- **Réactivité accrue** : Feature Store centralisé pour un accès rapide aux données.
- **Déploiement optimisé** : Conteneurisation et microservices pour flexibilité et scalabilité.
- **Conformité intégrée** : Respect des normes de sécurité (GDPR, PCI-DSS) à chaque étape.
- **Précision maintenue** : Surveillance continue et réentraînement automatisé pour gérer les dérives.



Merci de votre Attention!

Dans l'attente de vos questions.

