Stripe Data Architecture

Stripe is a leading global financial technology company, founded in 2010, that powers online payment processing for millions of businesses across over 120 countries. With billions of transactions processed annually and clients ranging from startups to Fortune 500 companies, Stripe operates at massive scale and complexity. As its operations have grown, Stripe's data architecture has become a strategic priority, requiring the integration in a single system of a large variety of data. This proposal outlines a comprehensive data infrastructure designed to ensure performance, consistency, and compliance while enabling advanced use cases such as fraud detection, customer insights, and predictive analytics.

Architecture Overview

The data integration architecture is presented in figure 1. It follows a hybrid model combining real-time streaming and batch processing, supporting low-latency data sync for operational use cases (e.g., fraud detection) and high-throughput batch processing for analytics.

- Data streams originating from Stripe API (e.g. bank transaction information, telemetry) are pipelined to the relevant database systems using kafka streams.
- A reference database holds slowly changing reference data such as country and currency codes, merchant information of currency exchange rates. Any change in this data is reflected to the systems that depend on it through change data capture (CDC).
- Data is archived periodically in a data lake, with batch processing managed by Apache Airflow.
- The loading of less time-sensitive data such as audit logs or customer feedback is handled by Airflow batches.

We present in table 1 a list of possible providers for the various systems of our architecture.

Table 1: Proposed providers for the various systems of the architecture.

System	Provider
OLTP/OLAP	Snowflake, Redshift
NoSQL	MongoDB, DynamoDB
Data Lake	Amazon S3, Azure Data Lake

Reference Database

Slowly changing or static reference data (e.g. country reference, merchant information, currency change rates) is stored in a reference database that serves as a single source of truth. The OLTP, OLAP and NoSQL systems incorporate a copy of the relevant reference tables for faster access. Central updates are propagated through change data capture (CDC). The data is subject to the security and compliance policy described below (e.g. encryption of merchant information). This approach has the advantage of a finer-grained monitoring and control of data access and modification. We present in table 2 a data dictionary for some tables in the reference database.

Online Transaction Processing (OLTP) Data Model

Our proposed OLTP database architecture is presented in figure 2. The core of the database is a registry of all financial transactions occurring within Stripe scope. Tables containing information about

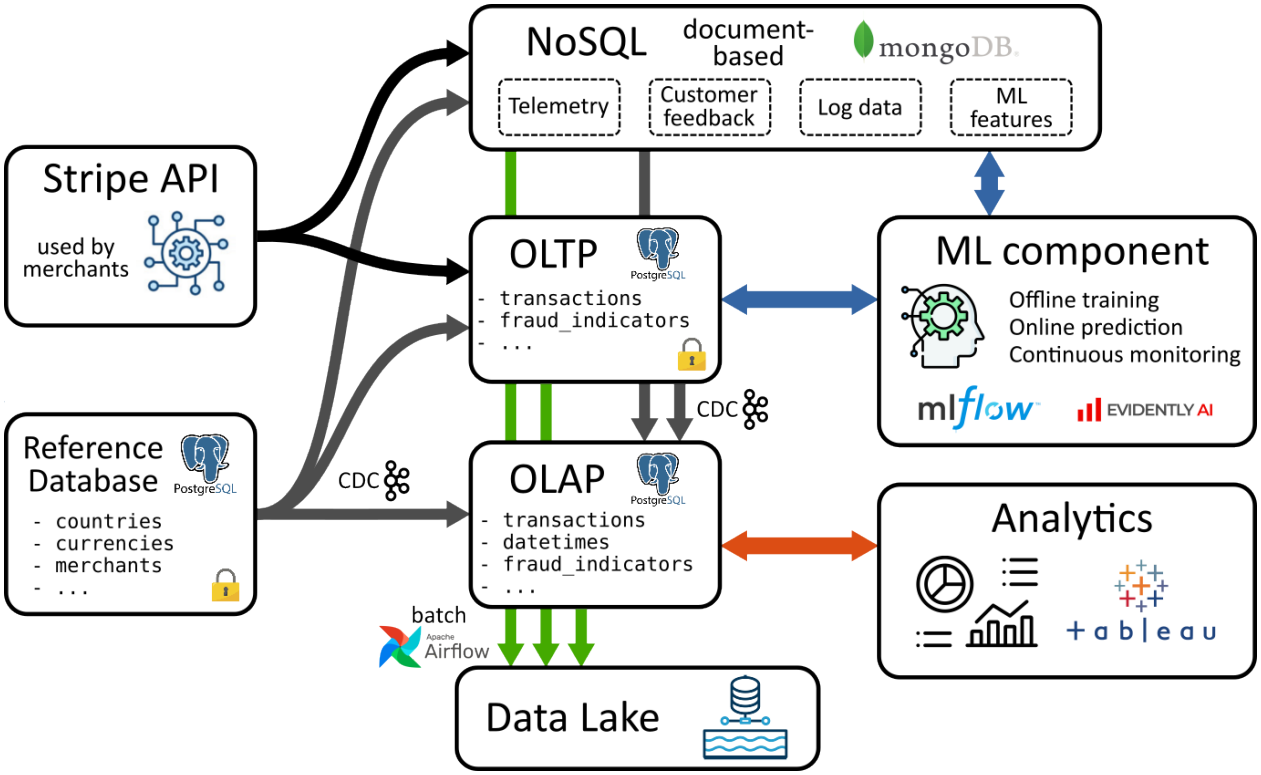

Figure 1: Overview of the proposed data architecture. The description is given in the text. The lock symbols in the OLTP and Reference databases indicate that the security of these systems is critical.

merchants and customers are cached from the reference database. Fraud indicators are stored in a dedicated table in a one-to-one correspondence with the main transactions table. The rationale behind this choice is that fraud indicators originate from a different pipeline (e.g. the AI pipeline).

Online Analytical Processing (OLAP) Data Model

Pour l'analyse et la préparation de features pour la détection de fraude, on utilisera une base

NoSQL Data Model

L'entreprise doit aussi gérer des données semi-structurées telles que les fichiers de log, les reçus de transactions ou encore la télémétrie effectuée sur la plateforme de paiement. Les contraintes pour chaque catégorie de données sont diverses et on adoptera une solution basée sur une document-based NoSQL database.

- Les informations de session ayant principalement des contraintes de disponibilité associée à des requêtes simples, on s'orientera vers une database de type key-value.
- Les fichiers de log sont structurés comme une succession d'évènements. Ceux-ci doivent être stockés de façon à pouvoir alimenter en temps réel des algorithmes de détection d'anomalies, afin d'assurer une réponse rapide en cas d'incident. Il est aussi nécessaire qu'ils soient lisibles par un humain pour une analyse approfondie. On s'orientera donc naturellement vers une document-based NoSQL database pour ce cas d'usage.
- Les fichiers non sensibles peuvent être stockés dans un espace approprié (Amazon S3) et indexés par une document-based database qui contiendra aussi les métadonnées.
- La télémétrie dans une column database
- Les features pour le machine learning peuvent être stockés dans une graph database.

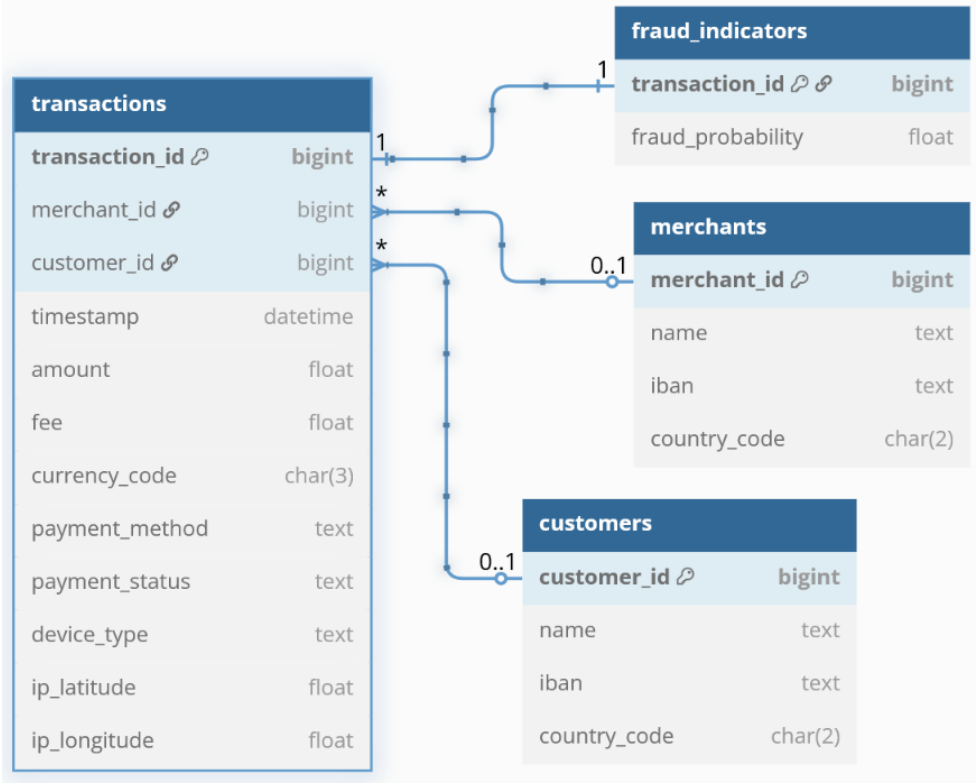

Figure 2: Proposed OLTP database structure.

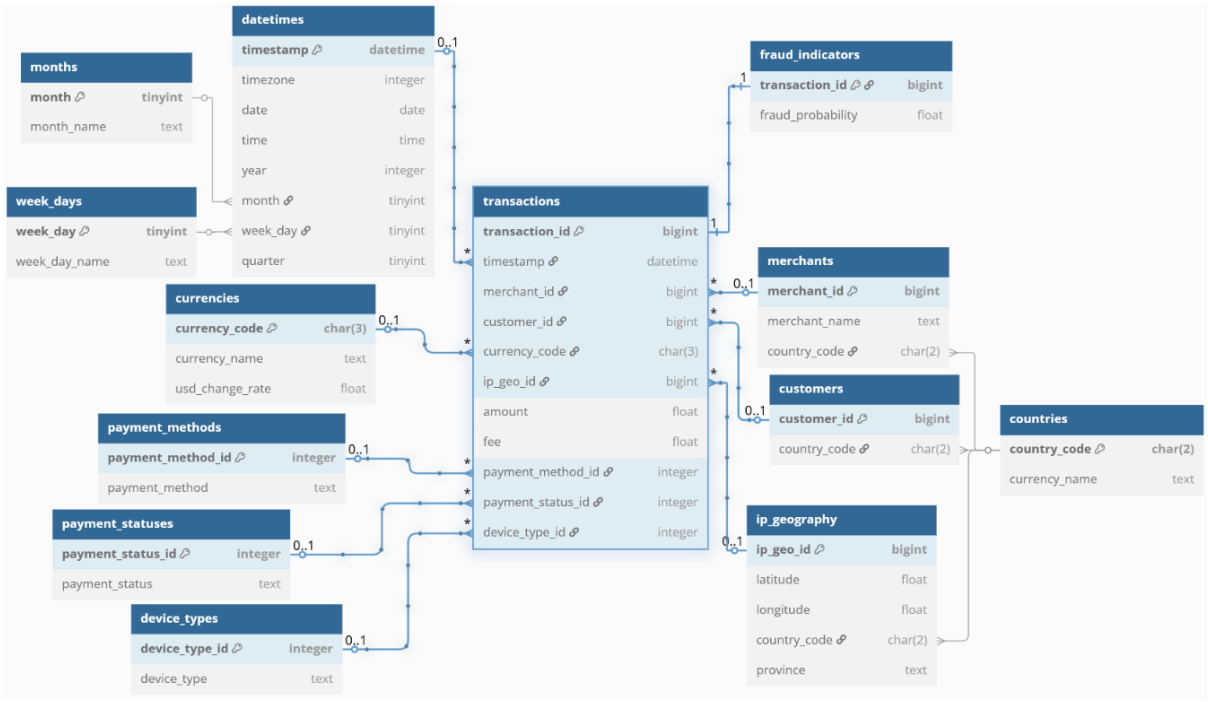

Figure 3: Proposed OLAP database structure.

Table 2: Data dictionary for the `transactions` OLTP schema.

Field Name	Type	Description	Example
		`countries` table	
`country_code`	`char(3)`	Country code (ISO 3166-1 2-letter)	`'GB'`
`country_name`	`text`	Country name	`'United Kingdom'`
		`currencies` table	
`currency_code`	`char(3)`	Currency code (ISO 4217)	`'EUR'`
`currency_name`	`text`	Currency name	`'Euro'`
`usd_change_rate`	`char(2)`	Currency to USD change rate	`1.08`
		`merchants` table	
`merchant_id`	`bigint`	Unique merchant id	`12345`
`name`	`text`	Merchant name	`'Amazon UK'`
`iban`	`text`	Merchant IBAN	`'GB82WEST12345678765432'`
`country_code`	`char(2)`	Merchant registration country code	`'GB'`
		`customers` table	
`customer_id`	`bigint`	Unique customer id	`234567`
`name`	`text`	Customer name	`'John Doe'`
`iban`	`text`	Customer IBAN	`'GB82WEST12345678765432'`
`country_code`	`char(2)`	Customer country code	`'GB'`

Security and Compliance

The company stores sensitive user data such as banking information. Il est important de sécuriser ces donnees pour deux raisons. D'une part, pour etre en accord avec la legislation locale (eg GDPR). D'autre part, la fuite de ces données impacterait la confiance accordée à l'entreprise par ses clients, avec en conséquence une baisse potentielle des revenus. Afin de limiter la surface d'attaque possible, les données sensibles sont confinées dans la base OLTP, et chiffrées à l'interieur de celle-ci.

Il est nécessaire de reporter en partie de ces données dans la base OLAP, notamment pour l'étude des délits financiers (fraude, blanchiment, etc). Afin de limiter les risques, la base OLAP ne continent que des éléments anonymisés de ces données, telles que la localisation ou le nom de la banque. Afin de maintenir la performance du système, ces données ne sont pas chiffrées et on se limitera à en sécuriser l'accès et le transfert (définition de roles pour limiter l'accès, etc). De la même façon, aucune donnée sensible n'est stockée directement dans la base NoSQL.

On distinguera les fichiers selon leurs contraintes en termes de sécurité. Les fichiers sensibles tels que les reçus bancaires seront stockés dans un datalake dédié avec un accès restreint. Ils seront indexés exclusivement dans la base OLTP (non indiqué sur le schéma). Les autres fichiers seront de la même façon enregistrés dans un datalake et indexés dans la base NoSQL dédiée.

Des backups chiffrés sont mis à jour à intervalles réguliers afin d'assurer le rétablissement du service en cas d'incident majeur.

Enfin, un système de log est mis en place afin d'enregistrer toutes les connexions aux serveurs, les accès aux données, les requêtes effectuées, etc. Un service additionnel peut être mis en place afin de détecter les anomalies en temps réel.

Machine learning integration

Table 3: Data dictionary for the `transactions` OLTP schema.

Field Name	Type	Description	Example
		`transactions` table	
`transaction_id`	`bigint`	Unique transaction id	`123456789`
`merchant_id`	`bigint`	Merchant id	`12345`
`customer_id`	`bigint`	Customer id	`234567`
`timestamp`	`datetime`	UTC transaction timestamp	`2023-11-18 17:43:02.4`
`amount`	`float`	Transaction amount (in currency unit)	`43.15`
`fee`	`float`	Transaction fee (in currency unit)	`0.53`
`currency_code`	`char(3)`	Currency code (ISO 4217)	`'GBP'`
`payment_method`	`text`	Payment method	`'credit_card'`
`payment_status`	`text`	Transaction status	`'sucess'`
`device_type`	`text`	Device used for payment	`'mobile'`
`ip_latitude`	`float`	IP-based geolocation latitude	`49.6833300`
`ip_longitude`	`float`	IP-based geolocation longitude	`10.5333300`
		`fraud_indicators` table	
`transaction_id`	`bigint`	Transaction id	`123456789`
`fraud_probability`	`float`	Fraud probability	`0.12`

Table 4: Data dictionary for the main tables in `transactions` OLAP schema.

Field Name	Type	Description	Example
		`transactions` table	
`transaction_id`	`bigint`	Unique transaction id	`123456789`
`merchant_id`	`bigint`	Merchant id	`12345`
`customer_id`	`bigint`	Customer id	`234567`
`timestamp`	`datetime`	UTC transaction timestamp	`2023-11-18 17:43:02.4`
`amount`	`float`	Transaction amount (in currency unit)	`43.15`
`fee`	`float`	Transaction fee (in currency unit)	`0.53`
`currency_code`	`char(3)`	Currency code (ISO 4217)	`'GBP'`
`payment_method_id`	`integer`	Payment method id	`1`
`payment_status_id`	`integer`	Payment status id	`2`
`device_type_id`	`integer`	Device id	`3`
`ip_geo_id`	`bigint`	IP geolocation id	`123456`
		`datetimes` table	
`timestamp`	`datetime`	UTC transaction timestamp	`2023-11-18 17:43:02.4`
`timezone`	`integer`	Timezone offset in minutes	`-120` for UTC-02:00
`date`	`date`	Transaction date	`2023-11-18`
`time`	`time`	UTC transaction time	`17:43:02.4`
`year`	`mediumint`	Transaction year	`2023`
`month`	`tinyint`	Transaction month	`11`
`week_day`	`tinyint`	Transaction week day (`0` is sunday)	`6` (saturday)
`quarter`	`tinyint`	Transaction quarter	`4`
		`fraud_indicators` table	
`transaction_id`	`bigint`	Transaction id	`123456789`
`fraud_probability`	`float`	Fraud probability	`0.12`
		`ip_geography` table	
`ip_geo_id`	`bigint`	IP geolocation id	`123456`
`latitude`	`float`	IP-based geolocation latitude	`49.6833300`
`longitude`	`float`	IP-based geolocation longitude	`10.5333300`
`country_code`	`char(2)`	country code (ISO 3166-1 alpha-2)	`'DE'`
`province`	`text`	Province name	`'Darmstadt'`