A Proposal for Stripe's Data Architecture

Stripe is a financial services company which provides payment processing solutions through APIs that web developers can use in their websites or mobile applications. The company processes billions of transactions annually, supporting millions of merchants worldwide.

The sensitive nature of the data managed by Stripe, along with the need of efficient

Data Pipeline

Les données recueillies sont transférées depuis l'API de paiement utilisée par les sites marchands. On en distingue deux types:

- A chaque transaction bancaire effectuée, un relevé est envoyé
- Des données de télémétrie sont transférées

Online Transaction Processing Model

Our proposed OLTP database architecture is presented in figure 1. The core of the database is a registry of all financial transactions occurring within Stripe scope. Additional tables contain information about merchants and customers. 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 do not originate from the same source as transactions, and are not produced at the same time.

Online Analytical Processing Model

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

NoSQL 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 adaptée à chacune.

- 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.

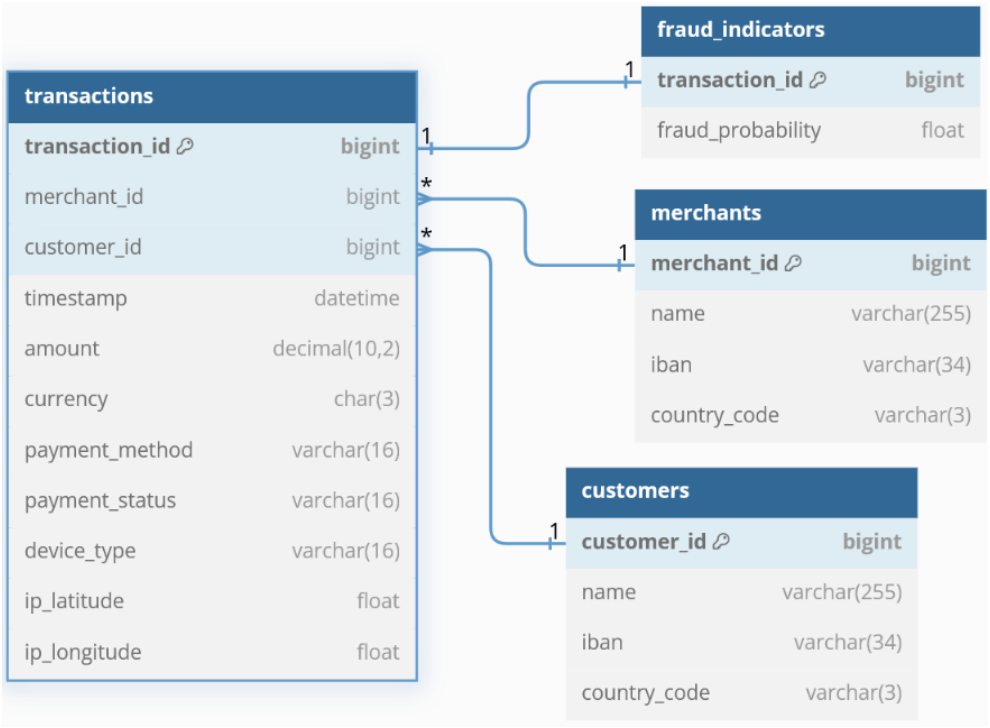

Figure 1: Proposed OLTP database structure.

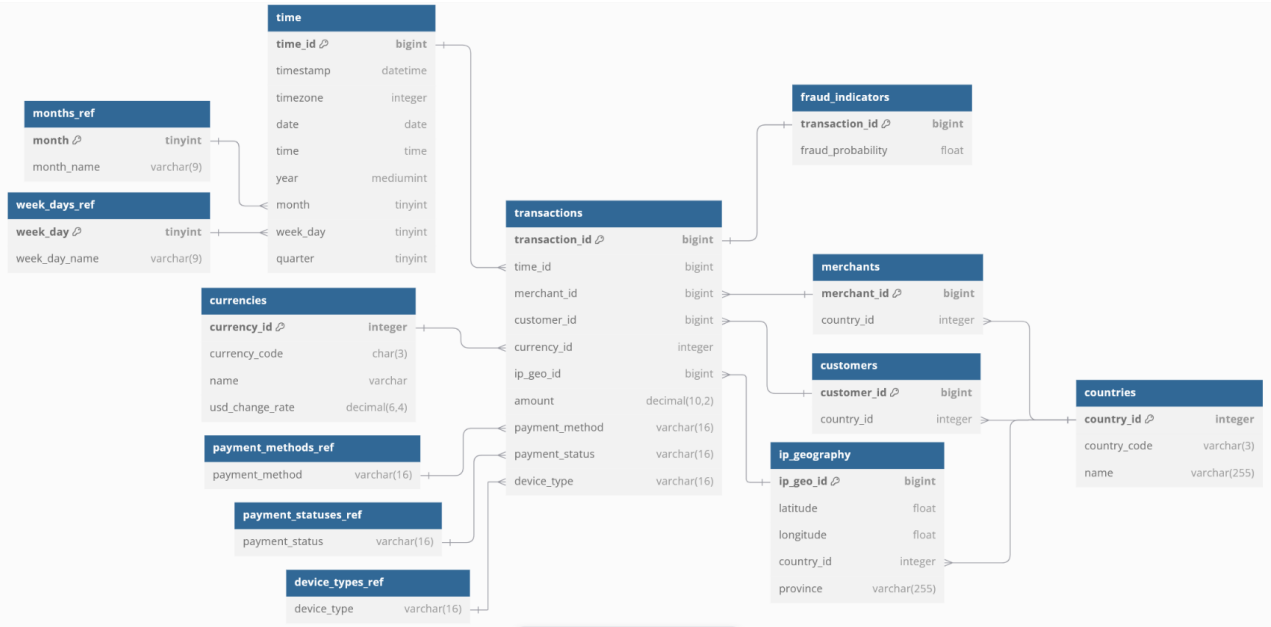

Figure 2: Proposed OLAP database structure.

Table 1: 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`
`time`	`datetime`	UTC transaction timestamp	`2023-11-18 17:43:02.4`
`amount`	`decimal(10,2)`	Transaction amount (in currency unit)	`43.15`
`currency_code`	`char(3)`	Currency code (ISO 4217)	`'GBP'`
`payment_method`	`varchar(16)`	Payment method	`'credit_card'`
`device_type`	`varchar(16)`	Device used for payment	`'mobile'`
`status`	`varchar(16)`	Transaction status	`'sucess'`
`ip_latitude`	`float`	IP-based geolocation latitude	`49.6833300`
`ip_longitude`	`float`	IP-based geolocation longitude	`10.5333300`
		`merchants` table	
`merchant_id`	`bigint`	Unique merchant id	`12345`
`name`	`varchar`	Merchant name	`'Amazon UK'`
`iban`	`varchar`	Merchant IBAN	`'GB82WEST12345678765432'`
`country_code`	`char(2)`	Merchant registration country code	`'GB'`
		`customers` table	
`customer_id`	`bigint`	Unique customer id	`234567`
`name`	`varchar`	Customer name	`'John Doe'`
`iban`	`varchar`	Customer IBAN	`'GB82WEST12345678765432'`
`country_code`	`char(2)`	Customer country code	`'GB'`
		`fraud_indicators` table	
`transaction_id`	`bigint`	Transaction id	`123456789`
`fraud_probability`	`float`	Fraud probability	`0.12`

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.

Table 2: Data dictionary for the main tables in `transactions` OLAP schema.

Field Name	Type	Description	Example
		`transactions` table	
`transaction_id`	`bigint`	Unique transaction id	`123456789`
`time_id`	`bigint`	Transaction time id	`123456789`
`merchant_id`	`bigint`	Merchant id	`12345`
`customer_id`	`bigint`	Customer id	`234567`
`currency_code`	`char(3)`	Currency code (ISO 4217)	`'GBP'`
`ip_geo_id`	`bigint`	IP geolocalization id	`1234567`
`payment_method_id`	`integer`	Payment method id	`1`
`payment_status_id`	`integer`	Payment status id	`2`
`device_type_id`	`integer`	Device id	`3`
`amount`	`decimal(10,2)`	Transaction amount (in currency unit)	`43.15`
		`time` table	
`time_id`	`bigint`	Unique transaction time id	`123456789`
`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`
		`merchants` table	
`merchant_id`	`bigint`	Unique merchant id	`12345`
`country_code`	`char(2)`	Merchant registration country code	`'GB'`
		`customers` table	
`customer_id`	`bigint`	Unique customer id	`234567`
`country_code`	`char(2)`	Customer country code	`'GB'`
		`fraud_indicators` table	
`transaction_id`	`bigint`	Transaction id	`123456789`
`fraud_probability`	`float`	Fraud probability	`0.12`
		`ip_geography` table	
`ip_geo_id`	`bigint`	Geolocation id	`123456789`
`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`	`varchar(255)`	Province name	`'Darmstadt'`

Table 3: Data dictionary for the reference tables in `transactions` OLAP schema.

Field Name	Type	Description	Example
		`currencies` table	
`currency_code`	`char(3)`	Currency code (ISO 4217)	`'GBP'`
`currency_name`	`varchar(255)`	Currency name	`'Pound sterling'`
`usd_change_rate`	`decimal(6,4)`	currency/USD change rate	`1.2479`
		`countries` table	
`country_code`	`char(3)`	Country code (ISO 3166-1 alpha-2)	`'GB'`
`country_name`	`varchar(255)`	Country name	`'United Kingdom'`
		`payment_methods` table	
`payment_method_id`	`integer`	Unique payment method id	`1`
`payment_method`	`varchar(16)`	Payment method	`'credit_card'`
		`payment_statuses` table	
`payment_status_id`	`integer`	Unique payment status id	`1`
`payment_status`	`varchar(16)`	Payment status	`'sucessful'`
		`device_types` table	
`device_type_id`	`integer`	Unique device type id	`1`
`device_type`	`varchar(16)`	Device type	`'mobile'`
		`months` table	
`month`	`tinyint`	Month number	`11`
`payment_status`	`varchar(9)`	Month name	`'november'`
		`week_days` table	
`week_day`	`tinyint`	Week day number	`6`
`week_day_name`	`varchar(9)`	Day name	`'saturday'`