# Stripe Business case

#### **Aperçu**

**Entreprise :** Stripe  
**Secteur :** Technologie financière (FinTech)  
**Fondée :** 2010  
**Siège social :** San Francisco, Californie  
**Produit principal :** Plateforme de traitement des paiements pour les entreprises

**Contexte :** Stripe est une société leader dans le domaine des technologies financières qui propose une gamme complète de produits pour gérer les paiements et les opérations financières des entreprises de toutes tailles. L'entreprise traite des milliards de transactions chaque année, au service de millions de commerçants dans le monde entier. À mesure que Stripe continue de croître, elle est confrontée à une complexité croissante dans la gestion de ses données sur diverses plateformes, notamment les systèmes transactionnels (OLTP), les systèmes analytiques (OLAP) et les bases de données non relationnelles (NoSQL).

Stripe souhaite créer une infrastructure de données de pointe qui intègre de manière transparente ces différents types de systèmes de données pour soutenir à la fois l'excellence opérationnelle et les analyses avancées. L'objectif est de créer une architecture de données unifiée et évolutive capable de gérer des volumes importants de données transactionnelles, de prendre en charge des requêtes analytiques complexes et de gérer des données non structurées pour des cas d'utilisation avancés tels que la détection de fraude, l'analyse du comportement des clients et les recommandations en temps réel.

#### **Scénario d'entreprise**

À mesure que Stripe se développe à l’échelle mondiale, l’entreprise est confrontée à plusieurs défis :

1. **Charges de travail transactionnelles complexes :**les systèmes OLTP de Stripe gèrent des millions de transactions chaque jour, avec des attentes élevées en termes de disponibilité, de cohérence et de rapidité. Ces transactions incluent les paiements, les remboursements, les rétrofacturations et la gestion des abonnements. L'entreprise doit s'assurer que ces systèmes sont hautement disponibles, évolutifs et capables de prendre en charge de nouvelles fonctionnalités telles que la détection des fraudes en temps réel.
2. **Exigences en matière d'analyse avancée (OLAP) :**Stripe doit fournir des analyses fiables à ses clients et à ses parties prenantes internes. Cela comprend l'analyse des revenus, la segmentation des clients, les rapports de conformité et le suivi des performances de divers produits. L'entreprise a besoin d'un système OLAP capable de traiter efficacement des requêtes complexes, de prendre en charge les analyses ad hoc et de fournir des informations en temps quasi réel.
3. **Gestion des données non structurées et semi-structurées :**Stripe collecte de grandes quantités de données non structurées et semi-structurées, telles que des journaux, des commentaires clients et des données d'interaction à partir de son site Web et de ses applications mobiles. Ces données sont essentielles pour créer des modèles d'apprentissage automatique pour la détection des fraudes, la personnalisation des clients et l'analyse prédictive. L'entreprise a besoin d'une base de données NoSQL capable de stocker et de traiter efficacement ces données, de les intégrer aux données relationnelles et de fournir des capacités d'interrogation flexibles.
4. **Intégration des données entre les systèmes :**l'intégration des données entre les systèmes OLTP, OLAP et NoSQL constitue un défi de taille. Stripe a besoin d'une architecture de données unifiée qui permette un flux de données transparent entre ces systèmes, garantissant que les données sont cohérentes, à jour et accessibles pour divers cas d'utilisation.
5. **Conformité et sécurité :**Étant donné la nature sensible des données financières, Stripe doit s'assurer que son architecture de données est conforme aux réglementations telles que le RGPD, la norme PCI-DSS et la CCPA. L'entreprise doit également mettre en œuvre des mesures de sécurité robustes pour protéger les données au repos et en transit, gérer les contrôles d'accès et surveiller l'utilisation des données.

#### **Exigences commerciales**

1. **Intégrité transactionnelle (OLTP) :**les systèmes OLTP de Stripe doivent gérer des volumes élevés de transactions avec une latence minimale. Le système doit prendre en charge les propriétés ACID pour garantir la cohérence et l'intégrité des données dans les environnements distribués. Il doit également prendre en charge des fonctionnalités telles que les transactions multi-documents, la synchronisation des données en temps réel et la reprise après sinistre.
2. **Analyse évolutive et efficace (OLAP) :**le système OLAP doit prendre en charge des requêtes complexes sur de grands ensembles de données. Il doit être capable d'agréger des données sur plusieurs dimensions, de prendre en charge l'analyse de séries chronologiques et de gérer des jointures et des sous-requêtes à grande échelle. Le système doit fournir des réponses à faible latence pour les requêtes métier critiques et prendre en charge les analyses historiques et en temps quasi réel.
3. **Gestion flexible des données (NoSQL) :**Stripe a besoin d'un système NoSQL capable de gérer divers types de données, notamment les données JSON, XML et binaires. Le système doit prendre en charge une conception de schéma flexible, permettre l'interrogation efficace de données imbriquées et non structurées et s'intégrer à d'autres systèmes de données pour fournir une vue globale des données.
4. **Intégration et cohérence des données :**l'architecture des données doit garantir un flux de données cohérent entre les systèmes OLTP, OLAP et NoSQL. Stripe doit mettre en œuvre des pipelines de données capables de gérer le traitement par lots, la diffusion de données en continu et la synchronisation en temps réel. L'architecture doit prendre en charge les processus ETL/ELT et permettre la transformation et l'enrichissement des données.
5. **Sécurité et conformité :**l'architecture des données doit inclure des mesures de sécurité robustes, telles que le chiffrement, les contrôles d'accès et la journalisation des audits. Stripe doit s'assurer que son stockage et son traitement des données répondent aux exigences réglementaires dans différentes juridictions. Le système doit prendre en charge les rapports de conformité automatisés et la surveillance en temps réel des failles de sécurité.

#### **Sources de données**

1. **Données transactionnelles (OLTP) :**capture toutes les transactions financières traitées par Stripe. Les champs incluent :

* ID de transaction
* ID du commerçant
* ID client
* Montant de la transaction
* Devise
* Mode de paiement
* Date et heure de la transaction
* Localisation (géolocalisation basée sur IP)
* Type d'appareil (mobile, ordinateur de bureau, etc.)
* Statut (réussi, échoué, remboursé)
* Indicateurs de fraude (par exemple, scores d'anomalie)

1. **Données analytiques (OLAP) :**données agrégées et historiques utilisées pour la création de rapports et l'analyse. Les champs incluent :

* Indicateurs de revenus (quotidiens, hebdomadaires, mensuels)
* Données de segmentation de la clientèle
* Indicateurs de performance du produit
* Données d'analyse de la fraude
* Journaux de conformité et d'audit

1. **Données non structurées et semi-structurées (NoSQL) :**données collectées à partir de diverses sources, telles que les journaux, les interactions avec les clients et les données générées par des machines. Les champs incluent :

* Données de journal (journaux d'erreurs, journaux d'accès)
* Données d’interaction utilisateur (données de parcours de clics, données de session)
* Fonctionnalités d'apprentissage automatique (données d'entrée pour les modèles)
* Commentaires des clients (avis, réponses aux sondages)

1. **Données de référence :**données statiques ou à évolution lente utilisées dans tous les systèmes :

* Codes de pays et de région
* Taux de change des devises
* Informations sur le commerçant
* Catalogues de produits

#### **Exigences techniques**

1. **Conception de modèles de données (OLTP, OLAP, NoSQL) :**vous devez concevoir des modèles de données pour chaque type de système de données :

* **OLTP :** concentrez-vous sur les schémas normalisés, l’intégrité transactionnelle et l’optimisation des performances.
* **OLAP :** concevez des schémas en étoile ou en flocon pour des requêtes et des agrégations efficaces. Incluez des tables pré-agrégées pour les requêtes courantes.
* **NoSQL :** concevez des schémas flexibles qui prennent en charge les données non structurées et permettent une interrogation et une récupération de données efficaces.

1. **Architecture du pipeline de données :**développez un pipeline de données qui intègre les données des systèmes OLTP, OLAP et NoSQL. Le pipeline doit prendre en charge le traitement des données par lots et en temps réel, la transformation et l'enrichissement des données. Envisagez d'utiliser des outils comme Apache Kafka pour le streaming de données en temps réel et Apache Airflow pour orchestrer les processus ETL.
2. **Évolutivité et optimisation des performances :**assurez-vous que l'architecture des données peut évoluer horizontalement pour gérer des volumes de données croissants. Mettez en œuvre des stratégies d'indexation, de partitionnement et de mise en cache pour optimiser les performances sur tous les systèmes. Envisagez l'utilisation de bases de données distribuées et de partitionnement des données pour une haute disponibilité.
3. **Cohérence et synchronisation des données :**implémentez des mécanismes pour garantir la cohérence des données entre les systèmes OLTP, OLAP et NoSQL. Envisagez d'éventuels modèles de cohérence, le cas échéant, et implémentez des stratégies de résolution des conflits pour les données distribuées. Utilisez les techniques CDC (Change Data Capture) pour maintenir la synchronisation des données en temps réel.
4. **Sécurité et conformité :**Concevez un cadre de sécurité qui inclut le chiffrement des données, le contrôle d'accès basé sur les rôles et la journalisation des audits. Assurez-vous que l'architecture est conforme aux réglementations en vigueur (par exemple, RGPD, PCI-DSS). Mettez en œuvre des contrôles de conformité automatisés et des fonctionnalités de reporting au sein des systèmes de données.
5. **Analyse avancée et apprentissage automatique :**intégrez des modèles d'apprentissage automatique au système NoSQL pour détecter les fraudes en temps réel et personnaliser les clients. Assurez-vous que le pipeline de données prend en charge l'extraction de fonctionnalités et le déploiement de modèles. Fournissez des mécanismes de surveillance des performances des modèles et de mise à jour des modèles selon les besoins.

#### **Tâche**

En tant que membre de l'équipe d'ingénierie des données de Stripe, votre tâche consiste à concevoir une architecture de données complète qui intègre les systèmes OLTP, OLAP et NoSQL. Votre conception doit répondre aux besoins de l'entreprise en matière d'intégrité transactionnelle, d'analyse avancée, de gestion flexible des données et de conformité réglementaire.

1. **Modèle de données OLTP :**

* Concevez un schéma normalisé qui prend en charge le traitement transactionnel à volume élevé.
* Assurez-vous que le modèle prend en charge les propriétés ACID et fournit des mécanismes de réplication et de basculement des données en temps réel.

1. **Modèle de données OLAP :**

* Concevez un schéma en étoile ou en flocon de neige qui prend en charge les requêtes et les agrégations complexes.
* Proposer une stratégie pour gérer les jointures à grande échelle, les sous-requêtes et l’analyse des séries chronologiques.
* Incluez un plan pour les pré-agrégations, les vues matérialisées ou les tables récapitulatives pour optimiser les performances des requêtes.

1. **Modèle de données NoSQL :**

* Concevez un schéma capable de gérer des données non structurées et semi-structurées.
* Proposer des stratégies pour gérer les relations entre les documents, y compris l’intégration, le référencement et l’indexation.
* Assurez-vous que le modèle prend en charge l’intégration avec les systèmes OLTP et OLAP.

1. **Architecture d'intégration des données :**

* Développer un pipeline de données qui garantit un flux de données cohérent et en temps réel entre les systèmes OLTP, OLAP et NoSQL.
* Inclut à la fois les composants de traitement par lots et de streaming.
* Proposer des outils et des technologies pour orchestrer et gérer le pipeline de données.

1. **Sécurité et conformité :**

* Concevez un cadre de sécurité qui inclut le cryptage des données, le contrôle d’accès et la journalisation d’audit.
* Proposer des stratégies pour assurer la conformité aux réglementations telles que le RGPD et le PCI-DSS.
* Élaborer un plan de reporting et de surveillance automatisés de la conformité.

1. **Intégration de l'apprentissage automatique :**

* Concevoir un processus d’intégration de modèles d’apprentissage automatique au sein du système NoSQL.
* Proposer des stratégies de détection de fraude en temps réel, de personnalisation des clients et d’analyse prédictive.
* Inclure des mécanismes de surveillance et de mise à jour des modèles d’apprentissage automatique.

## Étude du modèle relationnelle

### Intégrité transactionnelle

les systèmes OLTP de Stripe doivent gérer des volumes élevés de transactions avec une latence minimale. Le système doit prendre en charge les propriétés ACID pour garantir la cohérence et l'intégrité des données dans les environnements distribués. Il doit également prendre en charge des fonctionnalités telles que les transactions multi-documents, la synchronisation des données en temps réel et la reprise après sinistre.

#### Latence minimale

Voici quelques règles pour minimiser les latences :

* Utiliser les bonnes indexations
* Ajouter des indexes sur les colonnes fréquemment utilisées dans les filtres (WHERE), JOIN, ORDER BY.
* Utiliser des indexes composés pour les requêtes multi-colonnes.
* Éviter les indexes inutiles qui ralentissent les INSERT et UPDATE.
* Utiliser EXPLAIN ANALYZE (PostgreSQL) pour identifier les requêtes lentes.
* Limiter le volume de données récupérées.
* Mettre en place du cache, par exemple avec REDIS.
* Utiliser des bases de données près des utilisateurs.
* Partitionner la table Transaction pour optimiser les insertions et éviter le goulot d’étranglement (par exemple par date ou par région).

#### Cohérence et l'intégrité des données dans les environnements distribués

Monitorer et Détecter les Anomalies.

Détection d’anomalies via des logs distribués (Kafka, Fluentd).  
Alertes et monitoring avec Prometheus, Grafana.

#### Transactions multi-documents

Pour mettre en place des transactions multi documents tout en gardant des propriétés ACID dans une base de données transactionnelle comme PostgreSQL, il faut utiliser les commandes BEGIN, COMMIT et ROLLBACK permettant l’annulation de toutes les modifications en cas d’erreur.

#### Surveillance et monitoring

ACID

* Atomicité (A) : Une transaction est entièrement validée ou annulée.
* Cohérence (C) : Les données passent d’un état valide à un autre.
* Isolation (I) : Les transactions concurrentes ne se perturbent pas.
* Durabilité (D) : Les modifications validées sont persistantes, même après une panne.

## Plan de sécurité et de conformité

### Sécurité et résilience

Pour éviter les pannes il est possible d’utiliser un environnement distribué comme par exemple Kubernetes où certains Pods pourront prendre la main en cas de défaillances d’autres Pods.

Il peut être intéressant de placer les serveurs dans plusieurs régions du monde permettant des accès plus rapides mais aussi de prendre la main en cas de panne de l’un d’entre eux.

Il est nécessaire de chiffrer les données sensibles, d’utiliser des systèmes d’authentification fortes (OAuth2), d’utiliser des outils de logs et de monitoring et de former les collaborateurs à la cybersécurité pour minimiser au maximum le piratage.

### Mise en conformité

Voici quelques règles à mettre en place :

* Collecte minimale : Ne stocker que les données strictement nécessaires.
* Anonymisation / Pseudonymisation : Réduire les risques en remplaçant les données personnelles par des identifiants.

Droits des utilisateurs : Implémenter des APIs pour :

* Accéder aux données (Droit d’accès).
* Supprimer les données (Droit à l’oubli).
* Modifier les données (Droit de rectification).

Logs de Transactions Financières (PCI-DSS)

* Tentatives de paiements frauduleux.
* Transactions rejetées pour suspicion de fraude.
* Anomalies de latence dans le traitement des paiements.

Logs de Protection des Données Personnelles (RGPD)

* Actions sur les données utilisateur (consultation, modification, suppression).
* Exercice des droits des utilisateurs (exportation, suppression des données).
* Transferts de données hors de l'UE.

## IA

Collecte des Données Transactionnelles

Extraction des transactions depuis Stripe via l'API de Stripe ou d'autres connecteurs NoSQL (par exemple, MongoDB, Elasticsearch).

Identification des caractéristiques pertinentes à extraire pour la détection de fraude : montant, pays, adresse IP, géolocalisation, méthode de paiement, et comportements des utilisateurs.

1.2. Prétraitement des Données

Les données brutes doivent être traitées avant de pouvoir être utilisées pour l'entraînement du modèle :

Nettoyage des données : gestion des valeurs manquantes, traitement des erreurs d'encodage.

Extraction de caractéristiques : par exemple, calculer des agrégats des transactions passées, l'historique des paiements d'un utilisateur, et l'évolution de la fréquence des transactions.

Création de nouvelles fonctionnalités basées sur le temps (par exemple, la fréquence des transactions dans une période donnée) ou des comportements (par exemple, un utilisateur effectuant une transaction à partir d'un pays inhabituel).

Conversion des données non structurées en format utilisable, par exemple en analysant les logs de transactions ou en analysant les comportements atypiques via anomalie detection.

2. Entraînement du Modèle de Détection de Fraude

2.1. Choix du Modèle

Pour la détection de fraude, des modèles de classification binaire sont utilisés pour prédire si une transaction est frauduleuse ou légitime. Les algorithmes classiques incluent :

Forêts aléatoires (Random Forest)

Machines à vecteurs de support (SVM)

Réseaux neuronaux (Deep Learning)

Modèles d'anomalie : Isolation Forests ou Autoencoders.

2.2. Formation du Modèle

Utilisation de bibliothèques comme scikit-learn, TensorFlow, ou XGBoost pour entraîner le modèle de détection de fraude sur un historique de données transactionnelles. Ce processus peut inclure l'utilisation de features extraites des données.

2.3. Validation du Modèle

Validation croisée : Tester le modèle sur un sous-ensemble des données pour vérifier sa capacité à prédire correctement les transactions frauduleuses et à éviter les faux positifs.

Mesures de performance : Utilisation des courbes ROC, précision, rappel, F1-score pour évaluer le modèle.

3. Déploiement et Intégration du Modèle dans un Système NoSQL

Une fois le modèle validé, il est prêt à être déployé dans l'infrastructure de production pour effectuer des prédictions en temps réel.

3.1. Création d'une API de Prédiction

Déployer une API RESTful pour appeler le modèle ML afin de faire des prédictions en temps réel :

FastAPI ou Flask peuvent être utilisés pour déployer cette API.

Le modèle prédit si une transaction est frauduleuse ou non et retourne cette prédiction au système.

3.2. Interaction avec Stripe et NoSQL

Récupération des transactions : L'API de prédiction peut récupérer les informations des transactions en temps réel à partir de la base NoSQL ou directement depuis Stripe.

Exécution de la prédiction : Le modèle prend les données transactionnelles et retourne une prédiction (fraude/non fraude).

Stockage des résultats : La prédiction, ainsi que les détails de la transaction, sont enregistrés dans un système NoSQL comme MongoDB pour une analyse ultérieure. Les données peuvent être utilisées pour des alertes ou des analyses de fraude.

3.3. Gestion de la Latence et des Délais

Les transactions financières nécessitent des réponses très rapides. Utilisez des méthodes d'optimisation :

Modèles légers : Choisissez des modèles plus rapides à exécuter (par exemple, Random Forest) si des prédictions doivent être effectuées en quelques millisecondes.

Traitement par lots : Pour les analyses plus profondes ou historiques, les données peuvent être traitées en arrière-plan et les résultats intégrés dans les rapports ou les alertes.

4. Surveillance en Temps Réel et Réajustement du Modèle

Une fois le modèle déployé, il est important de surveiller ses performances pour assurer son efficacité continue et pour ajuster le modèle en fonction des nouvelles données et tendances.

4.1. Surveillance de la Prédiction en Temps Réel

Utilisation d'outils comme Prometheus et Grafana pour surveiller la latence des prédictions, le taux de fausses alertes, et la réactivité du système.

Alertes automatisées en cas de transaction suspecte (par exemple, seuils définis pour la probabilité de fraude).

4.2. Feedback Loop et Réentraînement

Feedback : Si une transaction frauduleuse est identifiée par le modèle mais n'est pas en réalité frauduleuse, un mécanisme de feedback humain ou automatisé permet de corriger les erreurs.

Réentraînement périodique : Le modèle peut être réentraîné en utilisant de nouvelles transactions et des annotations de fraude, afin d'améliorer la précision avec le temps.

## Pipeline de Données pour Stripe

### 1. Ingestion de Données

Sources de données :

OLTP : Transactions financières (paiements, remboursements, etc.).

NoSQL : Journaux, interactions utilisateurs, commentaires clients.

OLAP : Indicateurs de revenus, données de segmentation client, etc.

Outils :

Apache Kafka : Pour le streaming en temps réel des transactions et des interactions utilisateurs.

Apache NiFi : Pour l'ingestion et le routage des données non structurées.

## 2. Stockage des Données

OLTP :

PostgreSQL/MySQL : Pour les transactions en temps réel avec des propriétés ACID.

OLAP :

Snowflake : Pour le stockage et l'analyse des données historiques, offrant des capacités d'évolutivité et de requêtes rapides.

NoSQL :

MongoDB/Cassandra : Pour les données non structurées et semi-structurées.

### 3. Traitement des Données

Traitement par lots :

Apache Spark : Pour le traitement par lots des données historiques et l'agrégation des indicateurs.

Traitement en temps réel :

Apache Kafka Streams : Pour le traitement en temps réel des données en streaming.

### 4. Orchestration

Apache Airflow : Pour orchestrer les workflows de données, gérer les dépendances et planifier les tâches ETL/ELT.

### 5. Transformation et Enrichissement

Apache Spark : Pour la transformation des données, l'enrichissement et l'agrégation.

Change Data Capture (CDC) : Pour maintenir la synchronisation des données en temps réel entre les systèmes.

### 6. Stockage des Données Transformées

Data Lake :

Amazon S3/Google Cloud Storage : Pour le stockage des données brutes et transformées.

### 7. Analyse et Reporting

OLAP :

Tableau/Power BI : Pour la visualisation et l'analyse des données.

Machine Learning :

TensorFlow/PyTorch : Pour l'intégration de modèles de machine learning dans le pipeline.

### 8. Surveillance et Alertes

ELK Stack (Elasticsearch, Logstash, Kibana) : Pour la journalisation et la surveillance des données.

Prometheus/Grafana : Pour la surveillance des performances et la génération d'alertes.

### 9. Sécurité et Conformité

Cryptage :

TLS/AES : Pour le cryptage des données en transit et au repos.

Contrôle d’accès :

AWS IAM : Pour la gestion des accès et des rôles.

Journalisation d’audit :

Splunk : Pour la journalisation et l'analyse des logs d'audit.

Conformité :

RGPD, PCI-DSS, CCPA : Mise en œuvre de contrôles de conformité automatisés et rapports.

### 10. Intégration de l'Apprentissage Automatique

Modèles de Machine Learning :

TensorFlow/PyTorch : Pour la détection des fraudes en temps réel et la personnalisation des clients.

Surveillance et Mise à Jour des Modèles :

MLflow : Pour la gestion du cycle de vie des modèles de machine learning.