Chapitre 3
L'intégration de données

1. Introduction

Transporter ou échanger des données nécessite de mettre en place des services ainsi que, bien souvent, une infrastructure spécifique. Cette dernière doit répondre à des besoins particuliers qui dépassent souvent celui de simples interconnexions. Le démarrage d'un projet d'intégration de données doit donc au préalable faire l'objet d'une étude approfondie afin de bien comprendre toutes les modalités de connectivité mais aussi de transformations qui seront nécessaires. En effet, selon les besoins en matière de consommation (ou de diffusion) des données et des contraintes posées par les systèmes en amont et en aval, il faudra choisir le bon pattern d'intégration, voire en combiner plusieurs. Souvent, il sera aussi nécessaire de manipuler et transformer les données source afin de les adapter à la source données ciblée. Le choix de la solution est bien sûr un point fondamental, voire la clé de voûte de tout projet d'intégration de données.

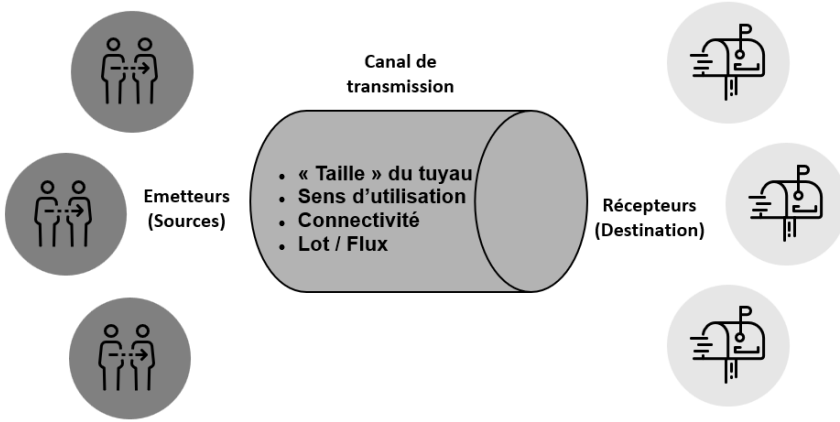
2. Propriétés d'une solution d'intégration de données

Dès lors que l'on aborde la notion d'intégration de données, on peut imaginer facilement des données empruntant un canal allant d'émetteurs vers des récepteurs. Ce canal de transmission permet en quelque sorte de faire transiter les données des sources (émetteurs) vers leurs destinations (récepteurs).

Comme tout composant, ce canal a ses propres propriétés qui devront être ajustées en fonction du besoin à adresser. Il est en effet évident qu'un besoin d'acquisition de données en mode temps réel n'aura pas les mêmes propriétés qu'un batch d'alimentation de « Data Warehouse ».

Voici donc les grandes propriétés à prendre en considération :

– les caractéristiques propres du canal de transmission des données (débit, nombre d'émetteurs, nombre de récepteurs, etc.) ;
– son architecture/sa structure ;
– son ou ses sens d'utilisation (liaison) ;
– le type de connexion nécessaire (synchrone ou asynchrone) ;
– son mode d'utilisation (lot ou flux).



Caractéristiques du canal de transmission

À noter qu'il faut aussi prendre en compte les contraintes propres à l'environnement. Les émetteurs et récepteurs de données ayant aussi leurs propres caractéristiques – et donc leurs propres contraintes – à prendre en compte dès le début du projet d'intégration de données.

On pourrait encore ajouter des notions de niveau de services comme le temps de réponse, le besoin en scalabilité, la tolérance de panne ou la gestion de reprise, mais focalisons notre attention sur les grandes caractéristiques fondamentales dans le choix de la solution à mettre en œuvre.

Aujourd'hui, on entend souvent parler de pipeline de données (ou Data Pipeline que nous aborderons plus loin dans l'ouvrage). L'image du pipeline est plutôt bien trouvée et s'apparente à première vue à la notion de canal. Néanmoins, si la notion de Data Pipeline englobe bel et bien la notion de canal de transmission des données, elle peut aussi inclure un certain nombre de transformations nécessaires pour rendre la donnée correctement utilisable. Il faut donc voir le Data Pipeline comme un moyen d'interconnexion de données riche permettant d'ingérer, de traiter, de préparer, de transformer et de gérer la qualité de données de manière contrôlée et gouvernée.

Commençons par aborder les fondations d'une solution d'intégration de données : sa « tuyauterie » en quelque sorte.

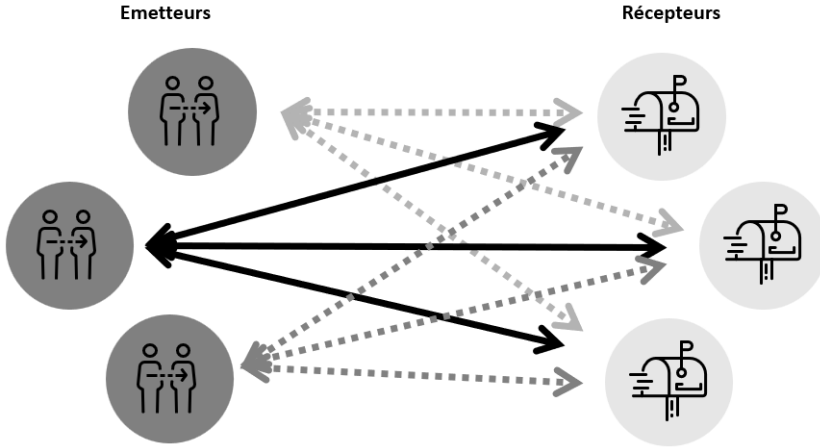
2.1 Architecture pour l'intégration de données

On distingue au moins deux grandes typologies d'architecture d'intégration de données. On les distingue principalement par la manière dont les échanges sont permis entre les émetteurs (ou sources) et les récepteurs (ou consommateurs).

L'architecture point à point (ou peer to peer)

Dans ce type d'architecture d'échanges, chaque émetteur peut directement se connecter à chaque récepteur pour échanger des données. Chaque échange (ou lien émetteur-récepteur) nécessite par ailleurs une nouvelle connexion et une gestion propre de la sécurité des échanges.

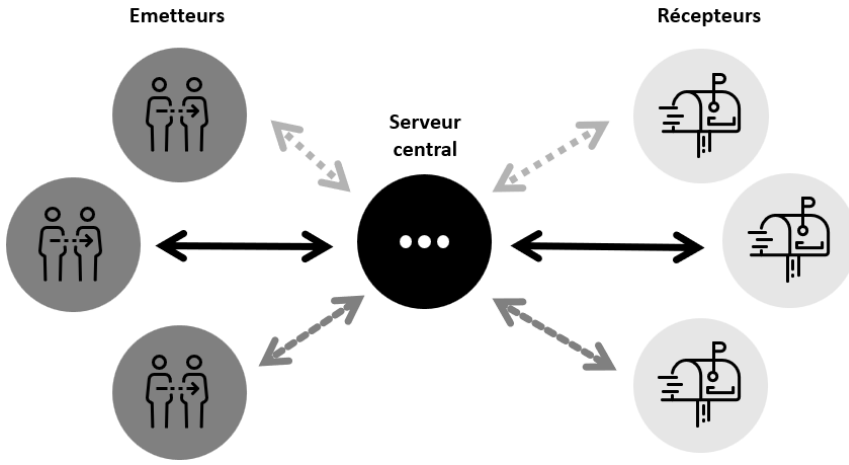
Si ce mode ne nécessite pas de solution technique particulière au premier abord (car chaque lien émetteur-récepteur gère sa propre connectivité), il est très utilisé car l'ajout de nouveaux éléments (émetteur ou récepteur) est simple et rapide. Ce type d'architecture est donc très simple à mettre en place mais peut rapidement devenir totalement ingérable s'il y a trop d'émetteurs-récepteurs. Dans le langage courant, on parle alors de « Spaghetti-ware ».



Architecture point à point

L'architecture centralisée

Ce type d'architecture nécessite un élément central, un serveur (on parle souvent de concentrateur ou Hub) qui sert de concentrateur des données provenant des émetteurs mais aussi de distributeur pour les récepteurs. On comprend immédiatement l'intérêt en matière de scalabilité et de gouvernance de ce type d'approche. Ces qualités intrinsèques ont un prix : celui d'une implémentation beaucoup plus complexe qui nécessite une solution adéquate (hébergée par le serveur central : le concentrateur). Fort heureusement, bon nombre de solutions clé en main proposent aujourd'hui ce type d'approche.



Architecture centralisée

Ce type d'architecture est tellement flexible qu'elle permet d'aller plus loin que de simples échanges de données. Bien souvent ces concentrateurs proposent, en sus de la consolidation et de la rationalisation de données, d'autres services tels que :

– l'ajout d'un système de contrôle de qualité des données ;
– une traçabilité de bout en bout ;
– une analyse complète des données ;
– des services d'alertes automatiques ;
– des possibilités de cache et de persistance permettant d'optimiser les échanges.

Voyons rapidement quelques caractéristiques et différences de ces deux grands types d'infrastructures d'échanges dans ce tableau récapitulatif :

Architecture point à point	Architecture centralisée
Implémentation rapide	Implémentation plus complexe car nécessitant une infrastructure dédiée
Connectivité directe	Connectivité émetteur/récepteur – serveur central. Le serveur doit donc proposer ses propres connecteurs.
Maintenance sur chaque nœud (émetteur ou récepteur)	Maintenance centralisée (serveur)
Sécurité sur chaque point	Sécurité centralisée
Auditabilité sur chaque connexion	Auditabilité centralisée
Gouvernance complexe de l'ensemble	Gouvernance centralisée
Scalabilité faible (idéal quand peu de nœuds)	Scalabilité forte (beaucoup de nœuds/types d'échanges)
Chaque nouvel échange (E-R) demande un nouveau développement de liaison point à point	Une fois les émetteurs/récepteurs déclarés au niveau du serveur il est beaucoup plus facile de rajouter des liaisons
Optimisations complexes et par connexion (par contre sans danger pour les autres nœuds)	Optimisation via cache, partage de données, etc.
Pas de point faible	Le point faible devient le serveur

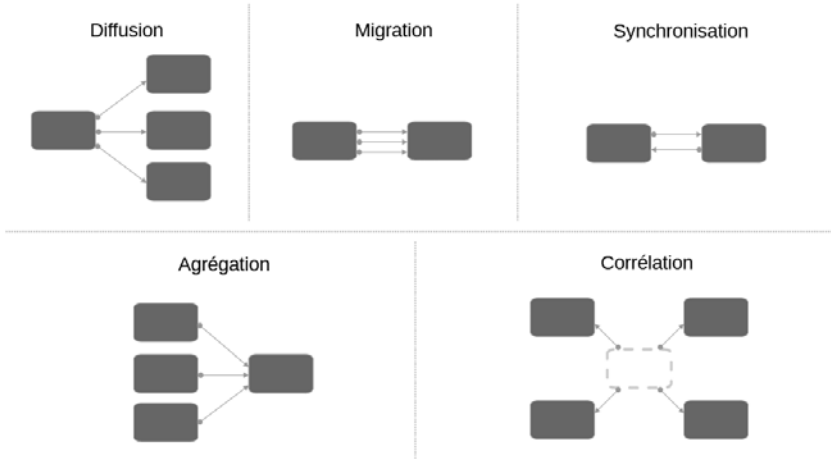
Remarque

Cette liste n'est pas exhaustive et dépend aussi de l'environnement technique et du (ou des) besoin(s).

2.2 Les grands modèles d'intégration de données

À vrai dire, il existe beaucoup de manières d'intégrer de la donnée vers un ou plusieurs systèmes. Néanmoins, on peut classer ces façons d'intégrer les données en plusieurs grands modèles (on parle aussi souvent de patterns d'intégration) :

– la diffusion de données ;
– la migration de données ;
– la synchronisation ;
– l'agrégation ;
– la corrélation.



Les modèles ou patterns d'intégration de données

Bien souvent et pour un besoin particulier, il est nécessaire de combiner plusieurs patterns entre eux afin, par exemple, de garantir la cohérence des informations ou tout simplement leur accessibilité. À titre d'exemple un MDM (*Master Data Management* ou référentiel de données) fonctionnant en Hub combinera a minima l'agrégation et la synchronisation. Dans certains cas de figure, il pourra même proposer la diffusion et la corrélation en sus.

2.2.1 La diffusion de données

La diffusion consiste à envoyer des données (à partir d'un système source) vers une autre source de données, et ce en temps réel ou en mode flux (Data Streaming). Les données envoyées sont donc en quantité faible mais peuvent être envoyées de manière très régulière. Il faut noter ici que la communication est unidirectionnelle et qu'en aucun cas le récepteur ne peut communiquer avec l'émetteur.

2.2.2 La migration de données

Quand on aborde la notion de migration de données, on pense immédiatement à un changement de système (de type bases de données ou ERP), mais il faut voir cela de manière plus générique. La migration de données consiste en effet à déplacer de manière effective des données d'une ou plusieurs sources de données vers une ou plusieurs autres sources de données. La notion de déplacement est très importante car elle implique une duplication des données à chaque point (source et cible). L'une des caractéristiques de la migration est le volume de données envoyées : en général on parle d'un volume important de données, et il faut donc prévoir un certain temps de transit de ces données (on parle alors souvent de fenêtre de batch).

À noter que la migration, par nature, s'applique sur un environnement cible vierge, c'est-à-dire exempt de données (a contrario de la synchronisation). On parle aussi de bascule d'un environnement vers un autre à l'inverse d'une cohabitation.

2.2.3 La synchronisation

La synchronisation est par nature bidirectionnelle (à ne pas confondre avec la diffusion de données qui se contente de publier une information d'un émetteur vers un récepteur et qui est, elle, unidirectionnelle). Dans le cas d'une synchronisation, les choses se compliquent car le résultat doit être la copie conforme de l'émetteur et du récepteur. Comment alors gérer des conflits ? Que faire quand une information existe des deux côtés mais qu'elle n'est pas identique ? Que faire si une information est supprimée ?

La synchronisation pose intrinsèquement de nombreuses questions qu'il faut définir et anticiper au préalable.

Voici quelques exemples de problèmes que l'on rencontre généralement :

– des conflits ou incohérences entre les données (provenant de la même source de données ou de sources de données différentes) ;
– des retards de synchronisation (problèmes de latence) ;
– des problèmes de dépendances de données.

Remarque

Une solution de type Data Hub par exemple est parfaitement adaptée pour adresser ces problèmes.

En général les outils de synchronisation disposent d'un moteur de règles qui permet de définir les modes de synchronisation et surtout les règles de gestion de conflits qui peuvent arriver lors de l'exécution.

2.2.4 L'agrégation

L'agrégation de données est une méthode permettant d'extraire et de traiter des données issues de plusieurs sources de données. Ce modèle consolide (ou agrège) les informations au sein d'une seule source de données (cible). Cette approche offre une alternative à la gestion laborieuse de multiples migrations quotidiennes, nécessitant une maintenance manuelle pour garantir l'exactitude, la synchronisation et l'actualisation des données.

Avec un modèle d'agrégation, il est possible de conserver la flexibilité d'interroger plusieurs systèmes à la demande tout en fusionnant les ensembles de données au moment opportun. Cela permet, par exemple de créer des rapports facilement. Cette approche simplifie aussi le processus en éliminant le besoin de migrations fréquentes et en offrant une gestion plus dynamique des données à travers différentes sources.

2.2.5 La corrélation

La corrélation ressemble à s'y méprendre à la synchronisation, à ceci près qu'elle permet d'associer à la volée des données de plusieurs sources sans les copier. La synchronisation bidirectionnelle quant à elle copie (ou réplique) les données pour effectuer le travail ensuite.