Gendarmerie Nationale

Retour d'expérience

Reprise de données

Chef de projet : Chef d'Escadron César LIZUREY

Publication du RETEX : 12 mai 2019

Remerciements

Merci à tous ceux qui m'ont aidé dans le cadre de la reprise de données du Fichier des Personnes Recherchées 2 : sans eux le projet aurait été un échec. Je ne peux citer tout le monde mais je pense bien évidemment à l'équipe de développement et à Thierry BRÉNUCHOT, un adjoint... exceptionnel ! Je pense aussi à mes camarades de la police impliqués dans la reprise : Nicolas et bien évidemment Sandrine, sans qui le projet n'aurait pas eu la même saveur.

Merci à Jérémy KESPITE pour l'aide qu'il m'a apporté dans le cadre de la rédaction du script de reprise, en particulier pour avoir été mon instructeur PERL.

Merci à Éric POMMEREAU pour la relecture de cet article. Fondateur de la Communauté Informatique au sein du Ministère de l'Intérieur (CIMI), véritable passionné et professionnel, il fait partie des rencontres que j'ai eu de la chance de faire à l'occasion de mon affectation au Service des Technologies et des Systèmes d'Information de la Sécurité Intérieure (ST(SI)^2).

Merci à mon père pour son sa relecture ô combien avisée !

COMMUNAUTE INFORMATIQUE
AU SEIN DU MINISTERE DE L'INTERIEUR

Et enfin merci à la Gendarmerie Nationale pour m'avoir donné l'opportunité de gérer un projet de si grande ampleur.

Résumé

Le développement des applications impliquant des changements de données importants peuvent entraîner la gestion d'une reprise de données, celle-ci pouvant devenir un véritable sous-projet. La réussite de cette reprise de données est indispensable à l'efficacité du fonctionnement de toute nouvelle application car elle doit garantir la continuité du service.

Le présent document a ainsi comme objectif d'en présenter les différentes étapes en précisant la temporalité dans laquelle elles s'inscrivent.

Il faudra d'abord s'interroger sur l'opportunité de l'externalisation de la reprise, ce choix étant lourd de conséquence, notamment en terme de protection des données. Une fois cettedécision prise, il s'agira de localiser les données que l'on souhaite reprendre, car ces données source essentielles peuvent être dispersées dans des endroits divers et variés.

Viendra ensuite le transfert et le stockage de ces données, qui devront faire l'objet de choix judicieux afin de sécuriser le trajet de la donnée de bout en bout, dans la mesure où ce trajet sera parcouru de nombreuses fois.

La donnée sera après cela transformée, cette étape faisant l'objet de spécifications précises et d'un développement agile, c'est-à-dire itératif et incrémental, car c'est le cœur du sujet : comment altérer la donnée source sur le fond et sur la forme de manière à ce qu'elle puisse être intégrée sans difficulté dans le système cible ?

Enfin, c'est la "bascule" qui se jouera : l'ensemble des étapes précédemment évoquées seront jouées en conditions réelles, et chaque acteur devra respecter la partition qui lui aura été donnée, toutes les parties prenantes ayant répété à plusieurs reprises les actions qu'ils devront mener le jour J.

Préambule

J'écris cet article suite à mon affectation en tant que chef de projet FPR2, qui m'a valu de passer un peu plus de trois années en poste au ST(SI)^2. Ce projet consistait en la création du nouveau Fichier des Personnes Recherchées à partir des deux FPR déjà existants : le Fichier des Personnes Recherchées de la Gendarmerie Nationale (FPR GN) et celui de la Police Nationale (FPR PN). Ceux-ci arrivaient à un état d'obsolescences technique et fonctionnelle tel qu'il devenait indispensable de bâtir un nouveau système unique pour les deux forces.

Bien évidemment, bâtir un unique système implique la transmission des données des systèmes précédents, et c'est en cela qu'il fallut s'atteler à une reprise de données. Afin d'illustrer mon propos lorsque j'aborderai certaines notions, je donnerai ainsi des exemples concrets tirés de mon expérience, ce qui permettra au lecteur de contextualiser les préconisations sous un angle pratique. Ces exemples apparaîtront sous la forme d'encadrés :

Dans le FPR2
Exemple tiré de mon expérience dans le cadre de ce projet.

Une reprise de données peut être un sujet important dans un projet, voire un sous-projet à part entière entrant sur le chemin critique du projet principal. La littérature en ligne peut être assez anxiogène en présentant la tâche comme longue et difficile et incite souvent à confier les clés des données à des prestataires externes. Avec de la méthode, il est pourtant facile de maîtriser les risques et de réaliser la reprise dans des délais plus que raisonnables.

La reprise de données n'est qu'une étape parmi d'autres : la mise en production, le choix des développeurs, l'arbitrage budgétaire ou encore la conception de l'architecture, par exemple, sont autant d'étapes qui requerront une attention particulière. Un projet est un course automobile et chaque virage doit être négocié de la meilleure des manières. La reprise des données n'est qu'un de ces virages.

La reprise n'est pas non plus régie par des règles de temporalité : plusieurs solutions existent, mais je suggère de réaliser la tâche en parallère pour des raisons que j'évoquerai plus loin. Appliquer un calendrier de reprise avant de comprendre le projet est en tout cas très risqué.

De manière analogue, il ne faut pas tester une reprise avec uniquement une portion des données, mais bel et bien avec l'ensemble de celles-ci, sous peine de faire face à des déconvenues le jour J. Trouver le moyen de disposer de l'ensemble des données sources est d'ailleurs la première étape de la reprise, celle-ci devant faire l'objet des tests les plus complets dès que possible : il faut multiplier les tests jusqu'à ce que déclencher une reprise de données devienne ludique, faute de quoi le jour de la véritable reprise ne sera pas forcément une partie de plaisir.

La reprise de données peut tout à fait être réalisée sans un outil d'extraction, transformation et chargement (ETL) : si le modèle de données est très proche, que vous n'avez pas d'inquiétude quant à perdre certaines données, que de travailler en boîte noire ne vous incommode pas, qu'il n'y a pas trop de règles métier à appliquer, pas trop de données à traiter et que vous avez déjà un ETL à votre disposition, c'est effectivement une option à étudier. Dans les autres cas, je pense qu'il vaut mieux se dispenser d'ETL.

Il va de soi qu'il peut exister autant de stratégies de reprise de données que de chefs de projet, et que cet article n'a pas la prétention de définir l'état de l'art, il s'agit simplement de donner aux futurs chefs de projet les clés d'une reprise de données qui a fonctionné. À chaque étape, le chef de projet a des choix technologiques et fonctionnels à faire : ceux-ci doivent être cohérents et appropriés au contexte et il ne saurait y avoir de méthode universelle.

Table des matières

Table des figures

1

Introduction

1.1 Qu'est-ce qu'une reprise de données ?

Vous êtes concerné par une reprise de données si vous avez à faire évoluer une application qui utilise des données et que vous souhaitez soit :

— effectuer un changement majeur dans votre application (construction d'une structure de données radicalement différente de la précédente, fusion de deux applications, etc.) : certains changements dans une application sont mineurs, c'est-à-dire que les développeurs peuvent altérer la structure des données en même temps qu'ils font évoluer l'application (par exemple concaténer deux champs "nom de famille" et "prénom" en un seul champ "nom" ou passer une date d'un format "JJ/MM/AA" à "JJ/MM/AAAA HH :MM"), il faut donc estimer correctement l'impact des changements effectués pour ne pas créer un chantier de reprise qui n'a pas lieu d'être ;
— effectuer un changement complet d'application qui viendra remplacer l'ancienne, par exemple dans le cadre de la mise en place du Règlement Général sur la Protection des Données (RGPD), la majorité des outils utilisés par les professionnels ou les particuliers n'étant pas adaptée à la nouvelle réglementation.

Dans le FPR2

J'étais dans le cas d'un changement complet d'application : même s'il s'agissait de regrouper le FPR GN et le FPR PN, la nouvelle application impliquait des changements techniques (structure des données complètement revue) ou fonctionnels (pas le même processus d'inscription ou de consultation) considérables.

Vous ne vous trouvez pas confronté à un problème de reprise de données si vous développez une application qui viendra consommer les données d'une application déjà existante : vous devez plutôt vous orienter vers le développement d'Application Programming Interface (API) au sein de la première application afin que la nouvelle puisse consommer ses données : il n'y a ainsi pas de duplication de données et le contrat d'interface entre les deux applications permet à la nouvelle de respecter plus facilement le RGPD.

L'esprit d'une reprise est d'automatiser le maximum de tâches :

— techniquement par le biais de scripts à exécuter ;
— humainement en s'entraînant à répéter les actions qui devront être réalisées le jour de la bascule.

que ce soit techniquement mais également par une répétition humaine conférant aux différents acteurs les réflexes qui devront être mis en œuvre le jour J.

1.2 Quelles sont les étapes ?

Une reprise des données est un chantier conséquent qui implique de la méthode ; je propose ainsi de découper les travaux de la manière suivante, les différentes parties étant détaillées par la suite :

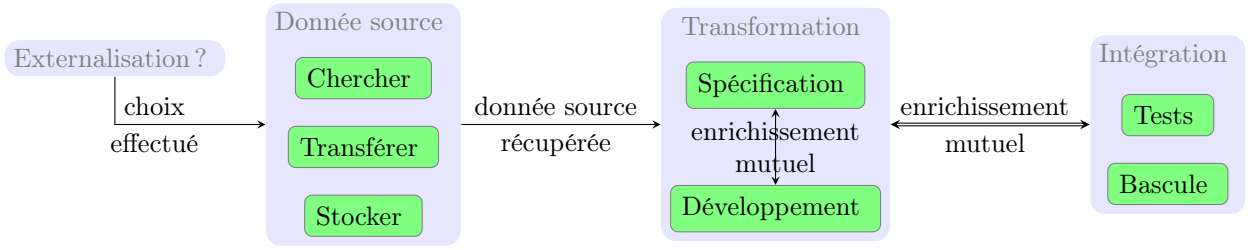

FIGURE 1.1 – Différentes étapes de la reprise de données

1.3 Quand commencer ?

Les données évoluent forcément avec le temps, que ce soit sur le fond (taille des données) ou sur la forme (modèle de données), il faut donc bien étudier à partir de quel moment la reprise doit être lancée.

Il ne faut pas commencer trop tôt les travaux de reprise : au début du développement de la nouvelle application, la structure des données qu'utilisera celle-ci n'est pas suffisamment mature, car elle changera au gré des différentes recettes fonctionnelles. Il apparaît alors difficile de faire correspondre les anciennes données avec des nouvelles dont la structure n'est pas finalisée. Cette évolution et donc consolidation de la structure des nouvelles données (architecture ou modèle de données) est la courbe "quantité de changements de la donnée" dans le schéma ci-dessous.

De manière analogue, il ne faut pas se consacrer à la reprise de données consécutivement au développement de l'application : cela induit le risque de commettre plusieurs erreurs au cours de la conception de nouvelle application (la reprise de données pouvant en effet être très bénéfique à la rédaction des spécifications, comme il sera expliqué plus loin). En étudiant les anciennes données, cela peut permettre d'éviter certains mauvais choix vis-à-vis de la nouvelle application. Il ne faut donc pas commencer trop tard, la reprise étant effectivement d'une étape à mener de concert avec le développement de la nouvelle application.

Je préconise donc une parallélisation de la reprise de données avec le développement de l'application.

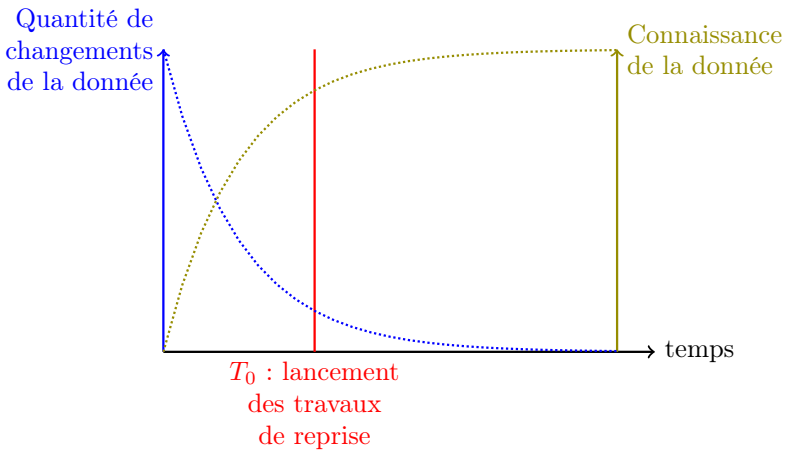

FIGURE 1.2 – Moment optimal de lancement des travaux de reprise selon l'évolution des données

2

Opportunité d'externalisation [1]

Externaliser le chantier de la reprise de données offre deux avantages :

— consacrer l'équipe interne à des tâches de développement de l'application en elle-même, et laisser le chantier des données qui seront utilisées dans l'application à un prestataire : c'est un choix qui peut se comprendre lorsque la contrainte de temps se fait sentir, mais attention, cet argument ne vaut que si les données à reprendre ne sont pas sensibles et structurellement simples. Il s'agira ici de transférer un effort intellectuel simple et cadré dans le temps à un prestataire ;

— se décharger de la responsabilité de la reprise de données : il peut être plus confortable pour certains chefs de projets d'utiliser un prestataire pour se soulager d'une tâche de cette ampleur, rendant de ce fait délicat l'explication aux fonctionnels d'un éventuel retard d'une tâche qui est de surcroît sur le chemin critique. Hormis s'il n'y a pas d'échéance précise, opter pour une externalisation est donc risqué, et ne pas maîtriser la reprise augmente la probabilité de l'échec du projet : c'est au chef de projet de dessiner le chemin menant des anciennes données aux nouvelles qu'il veut voir suivre par l'équipe.

Au-delà de ces raisons basiques, il convient pour tout chef de projet de mesurer les risques ddu choix de l'externalisation. Voici une liste non exhaustive de ceux que j'ai pu analyser, sachant que pour chacun d'eux le couple "probabilité d'occurrence"/"gravité" dépend de chaque projet, et c'est à l'aune de cette estimation que vous devrez faire votre choix :

2.1 Retard et augmentation du coût

Le prestataire connaît forcément moins bien les données que les personnes impliquées dans le projet : il y aura donc nécessairement une courbe d'apprentissage, ce qui se traduira par une prestation de cadrage afin de délimiter le périmètre de la prestation et/ou des retards et/ou une prestation en régie plutôt qu'au forfait, les prestataires externes étant traditionnellement frileux à s'engager sur des éléments non cadrés. Au résultat : les glissements de calendrier peuvent conduire le projet à prendre du retard et coûter plus cher.

Dissocier développement et reprise de données revient à perdre de vue que l'un et l'autre se construisent ensemble : lorsque l'on décortique la donnée source, il est possible de découvrir certaines logiques métier passées sous les écrans radar jusqu'alors, en particulier dans des projets où l'application initiale n'a pas été décortiquée depuis longtemps : ainsi, travailler sur la reprise de données apporte un éclairage supplémentaire lors de la conception de la nouvelle application. Peut-être pourrons-nous tirer profit des tempêtes de cerveaux de nos prédécesseurs, ou avons-nous repéré une erreur du passé qu'il s'agit de ne pas commettre à nouveau ? De même, participer au développement de la nouvelle application permet de comprendre certaines données sources dont la fonction pouvait susciter quelques questions auparavant, et qu'on aurait eu tort de ne pas reprendre en supposant erronément leur inutilité. Ainsi, dresser une barrière hermétique entre reprise et développement engendre une latence de ces deux facettes alors que la parallélisation de celles-ci favorise leur enrichissement mutuel.

Qui dit prestataire externe en charge de la reprise de données dit exposition d'éléments externes à des partenaires qui peuvent être frileux lorsqu'il s'agit de partager leurs données ou connaissances métier. Plus le projet est d'envergure, plus bâtir une équipe solide et de confiance est difficile : le risque est donc d'engendrer des freins au dialogue, telle ou telle partie prenante voyant d'un mauvais œil le fait de partager ses secrets avec des non initiés.

1. Plus communément appelée "analyse make or buy"

2.2 Perte de données sensibles

Si les données que vous souhaitez récupérer sont critiques, les confier à un prestataire externe peut être risqué. À l'heure de l'espionnage industriel généralisé, une fuite de données est une menace à prendre au sérieux : en plus d'un contrat de confidentialité, il faudra faire travailler le prestataire sur des machines non reliées à internet (fuites par boîtes courriels, Dropbox, Google Drive, bouts de code sur Stackoverflow, etc.) sur lesquels il ne pourra brancher aucun périphérique non homologué (USB, Bluetooth, écran, etc.). Même si le prestataire est le plus honnête du monde, le maillon le plus faible de la chaîne de la SSI est l'homme, qui peut parfois commettre des erreurs lourdes de conséquences. Se faire voler son ordinateur ou pirater des données sauvegardées en ligne n'est pas de la fiction.

Pour éviter le risque de fuite de données, certains pourraient avoir l'idée de fournir au prestataire des données dites "anonymisées". Il s'agit de transformer les données d'identité (nom, prénom, date et lieu de naissance) afin que le prestataire ne puisse pas disposer d'éléments réels. Dans la mesure où il n'aura pas accès à l'intégralité des données réelles, il ne pourra donc pas s'engager sur la réussite exhaustive de la reprise de données, une grande partie des erreurs rencontrée lors d'une reprise de données concerne des erreurs de format : si le format initial est altéré, difficile de se prononcer sur les chances de réussite de la véritable reprise [2]. Se pose de plus la question du reste des données, car si les données d'identité sont sensibles, les autres le sont toutes autant : répartition géographique des acheteurs de l'entreprise ou flotte des véhicules, nombre de fiches S et motifs dans le cas du FPR, etc. Ce n'est pas parce qu'une donnée à été anonymisée qu'elle n'est plus sensible : elle peut l'être à bien des égards pour qui saura la traiter.

Pour ce qui concerne les connaissances métier, il est évident qu'il est plus facile de reprendre une donnée dont on connaît le contexte métier. La documentation du système source peut être soit pauvre, soit en inadéquation avec ce qui est effectivement mis en place. Il faut donc comprendre la donnée à l'aune du métier qui la concerne. Il en va de même pour le système cible, dans des proportions théoriquement moindres dans la mesure où le nouveau système est censé être mieux documenté et la donnée mieux structurée (modèle de données revu, par exemple). Lorsqu'on externalise la reprise de données, le prestataire externe a de facto accès à la logique métier liée aux données, et peut ensuite s'en inspirer pour aider des concurrents, sans qu'il soit fait de lien explicite. Il s'agit d'un risque de fuite de capital immatériel, bien trop souvent sous-estimé [3].

Dans le FPR2

Comme vous l'avez sans doute deviné avec les propos tenus ci-dessus, je n'ai pas externalisé ce chantier :

- en tant que chef de projet, je me devais de comprendre l'intégralité des données qui existaient dans les systèmes existant afin d'éclairer les fonctionnels et de cadrer au mieux leurs besoins. Comprendre les données permet d'améliorer la compréhension du métier ;
- en tant que gendarme, je ne pouvais pas laisser à un prestataire externe la main sur des données relatives aux personnes recherchées, en particulier dans un contexte où les fantasmes allaient bon train concernant notamment les "Fichés S", où bon nombre d'experts autoproclamés se succédaient sur les plateaux télévisés. Poser des questions à la Direction générale de la Sécurité intérieure (DGSI) sur des fiches S particulières en présence d'un civil n'aurait pas été de bon goût ;
- j'ai personnellement été étonné de voir qu'un prestataire externe avait proposé le concours d'un développeur senior dont l'enquête administrative [a] a permis de découvrir qu'il était très défavorablement connu des services de police. Sur la base des éléments dont disposait ce prestataire, celui-ci aurait pu découvrir lui-même la partie émergée de l'iceberg. Il faut donc redoubler de vigilance quant à l'intégrité des prestataires externes lorsque ceux-ci ont accès à des données sensibles pour votre organisation.

a. Tous les civils désireux de travailler au sein de la Gendarmerie doivent passer au crible de l'enquête administrative. Cette enquête permet de s'assurer que le candidat est compatible avec un emploi au sein d'une institution en charge de la sécurité du pays.

Nous allons nous focaliser dans cet article sur une reprise effectuée en interne. Cependant, si vous décidez en tant que chef de projet d'externaliser cette tâche, vous pourrez vous baser sur ce document afin de comparer les travaux du prestataire avec les exemples qui y sont décrits.

2. Les premières erreurs qui pourront survenir proviendront de l'encodage des données sources : si on altère l'encodage, on se crée un angle mort. Viennent ensuite les erreurs de caractères spéciaux et de typage (format de date, entier/chaîne de caractères, etc.).

3. Lire à ce sujet : Jean-Sébastien LANTZ, *La Valo - Valorisation stratégique et financière*, Maxima, 2004

3

Données source

3.1 Localiser

La première chose à déterminer est la localisation et la forme de la donnée à reprendre. Cela peut paraître évident, mais les données peuvent parfois être disséminées à plusieurs endroits, sous des formats différents. Il existe plusieurs pistes pour trouver les données :

3.1.1 Dans les spécifications

Tout d'abord, dans les spécifications des applications utilisant les données. Les spécifications peuvent indiquer quelle base de données attaquer et comment, quels fichiers sont utilisés (des données peuvent se trouver dans des fichiers de configuration pour les applications les moins bien conçues). Certaines applications n'ont pas été développées en respectant des spécifications consignées dans un seul document, ou les spécifications ont été perdues au fil du temps. Tous les courriels ou feuilles imprimées et archivées peuvent être utiles. La reprise de données commence donc dans certains cas par de l'archéologie.

Les spécifications de la nouvelle application peuvent être elles aussi d'un grand intérêt, car l'utilité de certaines données que l'on souhaite reprendre peut être comprise à la lumière de ce qui est souhaité dans la nouvelle application. En effet, l'application dont on souhaite reprendre des données peut avoir connu par sédimentation des évolutions non documentées et implémentées dans l'urgence, sans que la nomenclature des termes utilisés soit de nature à permettre une compréhension rapide de ce qui a été réalisé. Ainsi, passer en revue les fonctionnalités de la nouvelle application peut lever le mystère de données sibyllines.

Dans le FPR2

Il y avait quelques indicateurs booléens dans la structure des anciennes données, sans qu'il soit possible de comprendre d'un coup d'oeil ce à quoi elles correspondaient. C'est en comparant aux attendus de la nouvelle application qu'il eût été possible de comprendre ces booléens et d'en arbitrer la reprise.

3.1.2 Dans le code

Le deuxième emplacement où trouver la donnée est dans le code. Bien évidemment, à moins d'une application très mal développée qui stockerait des données en dur dans le code (il en existe...), il s'agit ici de comprendre grâce aux lignes de code utilisant les données sources quelle est la structure des données auxquelles l'application accède : on sait dans quelles bases de données chercher ou ce qui est attendu comme donnée [1] (par exemple une chaîne de caractères correspondant à un nom).

3.1.3 Dans les bases de données

Enfin, les données peuvent être localisées de manière triviale dans les bases de données. Pour les applications récentes, les Système de Gestion de Base de Données (SGBD) modernes permettent de s'y retrouver facilement, mais chercher dans des bases de type General Comprehensive Operating System (GCOS) ou pire nécessitera plusieurs heures d'ingénierie inverse...

1. C'est là qu'on reconnaît a posteriori les bons développeurs qui auront su correctement organiser et documenter leur code.

Dans le FPR2

Les deux anciens FPR reposaient sur une technologie plus que vieillissante, avec heureusement des hommes clés dans chaque institution (GN & PN) qui étaient capables de m'éclairer lorsque, une fois confronté à ce qui m'apparaissait comme des bizarreries, j'avais des interrogations sur la structure ou l'utilité de telle ou telle donnée. C'est ainsi que je m'étais aperçu que la donnée censée apparaître dans les deux systèmes avait été altérée d'une manière différente selon le FPR de provenance [a]. J'ai donc pu faire un choix de stratégie de reprise, en ciblant précisément d'où et comment j'allais prendre la donnée source. Ce choix n'a pas été sans conséquence : en effet, si je choisis de récupérer la donnée source à partir d'un seul endroit, il faudra tout de même que je vérifie a posteriori qu'à la fois les données du FPR GN et du FPR PN figurent bien dans le FPR2. Les échanges entre deux systèmes structurés différemment comportaient des erreurs qu'il fallait au fil de l'eau récupérer à la main. Dans une reprise de données, il faut limiter au strict minimum cette correction par un opérateur humain.

a. Le FPR GN et PN étant des applications différentes, elles consommaient les données de manières différentes. Ainsi, une fiche provenant du FPR PN était transformée pour venir dans le FPR GN, et vice versa. Il y avait donc une manière plus simple et plus rapide de récupérer la donnée source.

3.2 Transférer les données

3.2.1 Déterminer le chemin parcouru par la donnée

Après avoir compris où se trouvait la donnée source, la tâche à réaliser est de déterminer la meilleure manière de récupérer la donnée source, en partant pour cela des responsables techniques de cette donnée. En effet, c'est d'eux que tout proviendra, et une fois ces responsables déterminés, il faut obtenir via l'administrateur des données l'autorisation formelle de récupérer les données de production. Cette autorisation peut aller de soi, en particulier si la même équipe est en charge du Maintien en Condition Opérationnelle (MCO) de l'ancienne application et du développement de la nouvelle, mais la transmission fera parfois parcourir à la donnée un chemin plus ou moins long, si les données sources sont exploitées par un autre exploitant que les données cibles, cela peut se compliquer.

Dans le FPR2

Sécurité oblige, le transfert de la donnée source était un véritable parcours du combattant : pour certaines données qui provenaient d'une institution, mon contact technique m'a communiqué le nom d'un interlocuteur qui était chargé de transmettre les données à un autre interlocuteur, après validation d'un interlocuteur intermédiaire, qui les mettait en forme, qui envoyait ensuite ces données mises en forme à un autre interlocuteur, qui était chargé de les communiquer à mon exploitant, qui à son tour me les déposait sur le serveur de mon choix, toutes ces opérations devant respecter des règles SSI très strictes. Et il ne s'agissait que d'une partie des données !

Il est très important de connaître l'intégralité de la chaîne de transmission et le rôle de chacun, car lors de la bascule, il faudra s'assurer que chacun soit à son poste.

3.2.2 Forme du transfert

Pour la forme du transfert, il faut bien sûr sécuriser au maximum tous les flux : rien par courriel et uniquement des protocoles documentés et validés par le RSSI [2]. Il faut maximiser les échanges téléphoniques pour avertir des transferts et ainsi éviter qu'un courriel intercepté soit l'occasion pour un pirate de tenter de se placer sur le chemin du transfert.

Dans le FPR2

Je récupérais une certaine partie des données en me déplaçant physiquement dans le bureau d'un interlocuteur d'une autre institution, qui me confiait un disque dur crypté, avec un mot de passe qu'il me communiquait en même temps, et que nous étions donc les seuls à connaître. À chaque fois qu'il est possible, il ne faut pas hésiter à utiliser la transmission physique entre deux personnes, qui est un processus parfois plus rapide que le transfert de fichier sur un réseau.

2. Je ne vais pas détailler ici toutes les bonnes pratiques en termes de chiffrement et de protocoles à utiliser dans le cadre de transferts de fichier.

Outre le fait qu'il faille sécuriser le transfert, il faut également connaître le contenu de la donnée envoyée par l'expéditeur, en convenant avec lui d'une liste d'éléments à comptabiliser : nombre de fichiers envoyés, nombres d'éléments de tel type dans tel fichier, etc. Il faut couvrir le maximum de points de contrôles pour s'assurer de l'intégrité de la donnée en bout de chaîne.

Dans le FPR2

J'ai par exemple demandé à avoir le nombre de fiches de chaque type ainsi que le nombre d'identités, en distinguant les hommes et les femmes entre autres. Ce n'est bien sûr pas exhaustif, mais voilà le genre de questions à poser à l'expéditeur.

3.2.3 Mise en forme de la donnée

La "mise en forme" de la donnée au cours des étapes du transfert est une étape à ne pas négliger : une donnée source reçue brute induira en bout de ligne des efforts d'ingénierie inverse dont on peut aisément se dispenser si les maillons de la chaîne effectuent certaines opérations basiques. Il s'agit donc ici de rentrer un peu dans la technique pour éviter de se consommer inutilement.

Si la donnée provient de SGBD modernes, cette étape est bien évidemment plus aisée, mais si vous devez manipuler de la donnée provenant de systèmes propriétaires ou anciens, il ne faut rien laisser au hasard. L'objectif est d'obtenir dès que possible, idéalement avant que vous receviez la donnée de vos interlocuteurs pour ne pas avoir à faire les transformations vous-même, des fichiers permettant une intégration facile en base de données. Chaque étape du transfert de la donnée source doit être l'occasion de formater les données en autant de fichiers séparant les données par ponctuation, dit de type Comma Separated Value (CSV) que nécessaires pour être transposés en une ou plusieurs tables.

Je préconise ainsi de travailler avec des CSV formatés de la sorte :

— encodage unique, de préférence Universal Character Set Transformation Format - 8 bits (UTF-8) : les erreurs d'encodage lors du traitement des données est parfois un calvaire, et s'en affranchir au plus tôt garantit une sérénité ;
— séparateur unique " ;", en prêtant attention au traitement des données qui comprennent ce caractère, afin que le script de transformation de la donnée (cf. plus loin) interprète correctement la donnée ;
— en-têtes intelligibles bien définis : parfois, la donnée source est non décrite ou comprend des descriptions équivoques, avec des enchaînements de caractères ne permettant pas de saisir la fonction de tel ou tel champ, en particulier dans les vieux systèmes contraints par la taille dans leur définitions de champ ;
— noms formatés avec un horodatage permettant d'automatiser la reprise et de rejouer des jeux de données.

Dans le FPR2

Par exemple, le CSV des fiches S du FPR PN était nommé "FPRPN_FICHE_S_250315_143000" pour des données transmises le 25/03/2015 à 14h30. Ainsi, lorsque le script de transformation des données (décrit plus tard) traitera les fiches S, il récupérera le fichier qui commence par "FPRPN_FICHE_S" et logguera l'horodatage du fichier utilisé. Il est ainsi possible de passer le même script sur des fichiers datés différemment, ce qui est particulièrement utile pour rattraper de nouvelles exceptions depuis la dernière exécution script.

Vous aurez en effet besoin d'avoir plusieurs versions d'un CSV, chaque version correspondant à un état de la donnée source pris à un instant donné, afin d'observer l'évolution de la donnée : a-t-on affaire à un sous-ensemble de données dont la taille croît ? Si oui, à quelle vitesse et à quel volume de données peut-on s'attendre le jour de la reprise réelle ? Dans le cas contraire, s'il s'agit d'une table qui décroît, est-ce que la donnée qu'elle concerne est toujours pertinente ? Les fonctionnels peuvent-il arbitrer pour l'abandon de cette donnée dans le système final ?

Je ne listerai pas les différentes mises en forme utilisées pour le FPR2, les exemples cités ci-dessus donnant l'esprit de cette étape, mon propos a simplement pour objet d'insister sur le "nettoyage" des données afin de faciliter l'automatisation du script de reprise : ce transfert de données source étant rejoué plusieurs fois, il convient de ne pas redécouvrir le mécanisme à chaque transfert.

3.3 Stocker les données

Une attention toute particulière devra être portée au sujet de l'utilisation des données reçues. Elles devront être stockées dans un endroit spécifique, auquel seules un nombre déterminé de personnes de confiance

peuvent avoir accès, cet accès devant être tracé, car il s'agit de ne pas créer une application fantôme permettant discrètement d'accéder à des données pour lesquelles les règles d'accès sont strictes.

Il faut garder à l'esprit qu'à un endroit seront stockées les différentes versions de données source qui seront utilisées pour les tests de reprise : toute donnée inutile devra être supprimée dans les plus brefs délais.

4

Transformation de la donnée

Venons-en à la partie qui sera après appelée le "script de reprise". Le script de reprise est le cœur de la bascule : il s'agit de transformer les données reçues en données intégrables dans le nouveau système. Cette transformation requiert des connaissances techniques et fonctionnelles : une maîtrise de l'algorithmie, des technologies utilisées et du métier de la donnée traitée.

4.1 Spécifications

Avant de se lancer dans le développement du script, il convient bien évidemment de rédiger des spécifications. Il faut au minimum disposer de 4 documents, appelés Spécifications Fonctionnelles Détaillées (SFD) :

- *SFD de la donnée source :* c'est le document qu'il convient d'avoir à la fin de la partie "donnée source" évoquée plus haut ;
- *SFD de la donnée de destination :* il s'agit d'un document à rédiger lors de la conception de la nouvelle application ; sans lui, inutile d'entamer une reprise car maîtriser la donnée de destination est un prérequis ;
- *SFD de la transformation de la donnée source en donnée de destination :* c'est un document qui évoluera au fil des tests, jusqu'à aboutir à une version finale sur laquelle se basera la bascule. Les paragraphes suivant expliquent comment remplir cette SFD, en particulier "Transformation de la donnée" ;
- *SFD de la bascule :* déroulé complet de la bascule précisant le rôle de chacun et la manière dont les données transitent et sont transformées. C'est le document clé pour le jour J et comme pour la SFD de transformation, ce document connaîtra plusieurs versions au fil des tests.

4.2 Développement

L'objectif du script est de produire des données intégrables dans la nouvelle application. Globalement, hormis quelques exceptions, il s'agira de générer des fichiers qui seront injectés dans le SGBD moderne que vous aurez choisi. Il convient donc de choisir un langage qui sait habilement manipuler des fichiers pour en produire d'autres. À ce titre, après en avoir comparé plusieurs, je ne peux que conseiller le langage PERL [1], qui me semble le langage le plus approprié pour réaliser cette tâche. En effet, sa rapidité d'exécution, sa simplicité de codage et le fait qu'il ait été conçu justement pour traiter de l'information de type textuel faisaient de lui le candidat idéal.

En plus de produire des fichiers exploitables, le script génère un journal qui permet de corriger les erreurs rencontrées.

Dans le FPR2

Mon équipe de développement étant déjà bien occupée avec le développement du cœur métier applicatif, leur confier le script de reprise aurait décalé la livraison de certaines fonctionnalités, ce que je ne pouvais me permettre. De plus, ils auraient mis plus de temps à le développer le script de reprise, n'ayant d'une part pas la vision de l'ancien métier (les leur enseigner les aurait perturbé dans l'appropriation du nouveau métier), ni les connaissances en le langage que j'avais choisi, PERL.

J'ai donc décidé de me charger du codage du script, le seul obstacle dans mon cas étant que je ne connaissais pas le PERL, mais la courbe d'apprentissage est fort heureusement compatible avec des délais contraints.

1. https://fr.wikipedia.org/wiki/Perl_(langage)

4.2.1 Analyse de la source

La première opération à effectuer est de s'assurer que les valeurs correspondent bien aux attendues, que ce soit sur le fond ou la forme :

- *référentiels :* les données issues d'un référentiel doivent bel et bien figurer dedans ;
- *type de données :* un entier doit être un entier (pas d'espace entre les chiffres, pas de décimale, etc.), une chaîne de caractères d'une certaine longueur également, etc. ;
- *textes libres :* pas d'injection de code possible, pas de caractères interdits (penser au transcodage), etc. ;
- *dates :* les anciens systèmes ont souvent tendance à manipuler de la donnée de type JJMMAAAA et non pas de l'horodatage ;
- *encodage :* si on s'attend à l'UTF-8, il ne faut pas avoir de l'ISO-8859-1, par exemple ;
- *colonnes :* vérifier que les bonnes colonnes soient remplies et qu'il n'y ait pas de décalage.

Lors du passage du script de reprise, c'est lui qui effectuera cette vérification, mais il convient d'affiner le travail en amont, en utilisant un tableur pour exploiter le CSV. Un comparatif de plusieurs outils m'a fait adopter Microsoft Excel dont les filtres et les fonctions "INDEX" et "EQUIV" [2] permettent de repérer rapidement les anomalies.

Dans le FPR2

J'ai bien évidemment essayé LibreOffice, afin de maximiser le nombre d'outils en source libre utilisés dans le cadre du projet, mais compte tenu de la taille des fichiers à exploiter, le temps de traitement et le nombre d'erreurs m'ont fait adopter son concurrent de la firme de Redmond.

Il faudra donc documenter la SFD de transformation afin qu'elle précise que le script analyse les données sources de manière à vérifier qu'elles correspondent aux attendus, c'est-à-dire à la SFD de la donnée source. C'est à cette étape qu'il faut utiliser de manière intensive les expressions régulières.

4.2.2 Transformation de la donnée

Principe

Le script produit des CSV à intégrer dans le SGBD de destination, ainsi qu'un journal, un "log". Ce log, qui contient par définition beaucoup d'informations, de préférence explicites, permet d'affiner les SFD et le script en lui même au fur et à mesure de ses exécutions : le premier passage génère un grand nombre de déchets, le suivant moins dans la mesure où des corrections ont été apportées, et ainsi de suite. Dans l'idéal, il faut arriver à zéro erreur le jour J (toutes les données étant reprises), mais on peut déterminer un plafond d'erreurs traitables humainement.

Afin de ne pas se perdre, il faut découper le script en modules, l'ensemble des données sources pouvant ne pas être traitées d'un seul bloc, afin d'isoler chaque CSV généré et analyser l'exactitude des éléments au fur et à mesure.

Dans le FPR2

De manière assez grossière, le FPR2 peut être découpé entre identités, corps commun des fiches et corps spécifiques des fiches. J'ai donc découpé mon script en respectant ce principe, pour arriver à une cinquantaine de scripts de taille plus petite, et m'attarder sur chacun à tour de rôle. On revient aux fondamentaux : diviser pour régner.

Le script est rejoué maintes et maintes fois, initialement avec les mêmes données de production, puis avec des données plus récentes. Attention, les anciens systèmes pouvant être plus tolérant dans l'ajout de nouvelles données au regard des règles métier implémentées dans la nouvelle application, il faut bien tester l'ensemble des données à chaque version du script ; une nouvelle version du script peut avoir négligé une particularité qui ne figure plus dans la dernière version des données sources mais qui est susceptible de s'y retrouver ultérieurement [3].

Garder des données de production datant de dates différentes peut donc être utile. Une autre bonne pratique peut être mise en œuvre : lorsque la donnée source contient des éléments rendant la transformation soit très compliquée soit impossible en raison des nouvelles règles métier, il est plus facile de "nettoyer" la donnée source en effectuant des modifications à partir de l'interface d'administration de l'application initiale.

2. Débarassez-vous de "RECHERCHEV" et "RECHERCHEH" : https://www.excel-pratique.com/fr/fonctions/index_equiv.php.

3. Parfois, certaines lignes de code paraîtront obsolètes alors qu'elles servent bien à gérer une exception particulière. La documentation du code est primordiale...

Dans le FPR2

Un problème de date engendrait des erreurs d'intégration dans le nouveau FPR. Les fonctionnels ont donc changé les dates incriminées afin qu'elles répondent aux critères de la nouvelle application.

Le script doit être documenté afin de pouvoir être relu et/ou repris par différents développeurs.

Détermination des règles

Une étape qui se situe à mi-chemin entre le fonctionnel et le technique, c'est d'ailleurs à ce stade que la SFD de transformation évoluera, car au fur et à mesure des tests du script nous nous apercevons des erreurs rencontrées, et il faudra préciser la manière dont elles seront traitées.

L'algorithme convertit des fichiers contenant les données source en des fichiers contenant les données cible en respectant des règles de transformations qui suivent notamment des tables de correspondance.

Pour chaque transformation, il faut effectuer plusieurs passes. Chaque étape est plus ou moins liée aux autres, mais la reprise est l'occasion de répéter de nombreuses vérifications pour ne rien laisser au hasard :

1. *table de correspondance*[4] *descendante brute :* on essaye ici simplement de faire correspondre une donnée source à une donnée cible, en terme de colonne. Il peut arriver que des "carrés" de la donnée source ne rentre pas dans les "ronds" de la donnée cible, c'est-à-dire, par exemple, qu'une "ville" décrite dans la donnée source ne soit pas comprise comme une "ville" dans le système cible[5]. Le produit de cette étape est :
   - *un fichier de transformation :* il indique de quelle donnée source on part pour arriver à quelle donnée cible. Un fichier basique comprend seulement deux colonnes, mais il peut en avoir plusieurs (par exemple : "libellé ville en texte libre" + "département" = "code INSEE") ;
   - *une liste de données source non exploitées :* toutes les données source qui n'auront pas trouvé de correspondance dans les données cible. Par exemple, pour les villes, "MARSEILE" n'est pas trouvé, mais un humain analysant ce fichier comprendra rapidement qu'il s'agit de "MARSEILLE". Attention, il ne s'agit pas que de fautes d'orthographe mais il peut y avoir des données source qu'on ne sait pas exploiter au premier abord ;
   - *une liste de données de destination non pourvues :* très utile pour s'apercevoir si les règles de transformation sont exhaustives. Par exemple, si on remarque qu'aucun individu n'a les yeux marron, il y a des questions à se poser en termes de conversion ;
   - *une liste de tâches à faire pour terminer la complétion*[6] ;
2. *table de correspondance montante brute :* on cherche ici à effectuer l'opération inverse. Pour chaque colonne de la donnée de destination, on va chercher ce qui correspondrait dans la donnée source. Le produit de cette opération vient enrichir le précédent, on ne crée rien de nouveau, il s'agit d'ingénierie inverse ;
3. *affinage de la table de correspondance des valeurs :* à partir des données non exploitées, il s'agit de déterminer si elles peuvent l'être afin de remplir les données de destination non pourvues. En effet, si des données source existent, il y a de grandes chances qu'elles aient leur utilité. Il faut donc ici les analyser pour trouver des règles de type "colonne A et colonne B donne telle donnée de destination selon telle règle" ;

   Dans le FPR2

   Il y avait une règle métier qui imposait de donner un cadre juridique dans certaines fiches du FPR2, l'ancien FPR ne mentionnant pas cette information mais disposant d'un nombre d'années de décision judiciaire et d'une date de création. C'est entre autres sur la base de ces éléments qu'il a été possible de construire la correspondance.

4. *"référentialiser" des données libres :* sans doute la partie qui peut prendre le plus de temps. Il s'agit ici de collecter toutes les valeurs prises par un champ, et de les faire correspondre à un référentiel de destination.

4. Parfois appelé "mapping" par ceux qui ne connaîtraient pas le terme de notre belle langue française.
5. Exemple tiré du FPR car les villes étaient stockées en texte libre dans le FPR PN et obéissaient au référentiel Institut National de la Statistique et des Études Économiques (INSEE) dans FPR2.
6. Rappelons-nous que nous rédigeons ici une SFD, document qui sert d'échange avec les fonctionnels. Il faut leur donner des consignes afin d'aider dans la transformation des données. Dans le cas des villes non trouvées, comme "MARSEILE", il peut s'agir de faire correspondre chaque libellé à un code INSEE.

Dans le FPR2

Par exemple, les villes étaient en texte libre, et il fallait les faire correspondre avec des codes INSEE. Il faut donc lister toutes les valeurs prises, déterminer lesquelles peuvent être récupérées facilement (si je vois “BREST”, je vais chercher directement le bon code), et déterminer la correspondance du reliquat. Les difficultés étaient :
— villes mal orthographiées (ex. : “SARCELLE”) ;
— villes orthographiées différemment (ex. : “SAINT-MARTIN”, “ST MARTIN”, “ST. MARTIN”) ;
— villes inexistantes (ex. : “SENEGAL”) ;
— villes en doublon (ex. : “SAINT DENIS” dans le 93 et dans les DOM-COM) ;
— précision dans la ville (ex. : “PARIS” et ses arrondissements) ;

5. *affiner les référentiels :* il se peut que les référentiels utilisés par la source ou la destination ne correspondent pas. Il faut donc déterminer avec les fonctionnels lorsqu'une donnée de référentiel source ne correspond pas à un référentiel de destination si :
— on lui attribue une correspondance ;

Dans le FPR2

On a par exemple fait correspondre un “frisé” du FPR PN avec un “crépu” du FPR2

— on modifie le référentiel de destination en ajoutant la donnée en question : il s'agissait d'un oubli dans la conception de la nouvelle application ;
— on modifie le référentiel source, dans les cas où on s'aperçoit qu'on en a déjà besoin dans l'application existante. Attention dans ce cas à l'altération des données déjà présentes en base ;
— on abandonne cette donnée, quitte à la mettre dans un champ libre. Il est parfois intéressant de rajouter une colonne “précision” dans le CSV ;

Dans le FPR2

J'ai utilisé à plusieurs reprises la colonne “précision”, qui apparaissait dans la nouvelle application de manière totalement transparente pour les utilisateurs.

6. *déterminer les données non reprenables :* certaines données sources ne peuvent effectivement pas être intégrées dans le système cible. Il faut donc :
   (a) demander aux responsables fonctionnels du système cible si on peut abandonner cette donnée. En effet, certaines données peuvent être obsolètes et la reprise de données l'occasion de procéder à un nettoyage ;
   (b) si ce n'est pas possible, proposer :
   — de reprendre dans un champ libre de type “précisions” ;
   — d'enrichir un référentiel de destination si cela résout le problème ;
   — de prévoir un champ structuré supplémentaire dans le système cible ;
7. *déterminer les données manquantes :* il se peut que certaines données de destination restent non pourvues. Si dans le système cible la donnée n'est pas obligatoire, cela n'est pas grave, mais si elle est impliquée dans des contraintes de base, de calcul ou d'Interface Homme Machine (IHM), cela peut poser problème. Plusieurs solutions à proposer aux responsables fonctionnels :
— modifier la donnée source afin de pouvoir l'intégrer dans le script de reprise. Bien évidemment, cela demane aux responsables fonctionnels de modifier beaucoup de données sources, et il ne faut pas que cette pratique se généralise trop. Néanmoins, pour des modifications gérables à taille humaine, il est possible d'effectuer certaines opérations sur la donnée source pour qu'elle devienne traitable.

Dans le FPR2

Pour les interdits de stade, par exemple, il a été possible d'ajouter “STADE :” suivi du nom du stade concerné. Cela permettait de ne pas toucher à l'application originale et de remplir automatiquement un champ structuré dans l'application cible ;

— rendre le système cible moins contraignant afin que les données en question ne deviennent plus obligatoires. L'inconvénient est qu'il s'agit de revenir sur certaines spécifications, ce qui peut être agaçant pour les développeurs (un peu moins pour les fonctionnels qui ne se rendent pas forcément compte des conséquences des modifications de règles métier), mais qui peut très bien être intégré dans une méthode agile ;
— quoiqu'il arrive, une donnée existant dans l'ancien système est consultable en l'état, elle doit donc pouvoir l'être sans modification dans le système cible. Il faut donc s'accorder avec les responsables fonctionnels sur ce point, en permettant au système cible de traiter une donnée source qui ne respecte pas toutes les contraintes du système cible. Si la donnée ainsi reprise fait l'objet de modifications au sein du système cible, il faudra en revanche que ces contraintes soient respectées.

Dans le FPR2
Par exemple, il manquait pour certaines fiches liées aux permis de conduire le type d'arrêté qui les concernait, cette donnée étant obligatoire dans le FPR2. Les fiches ont donc été reprises sans cette donnée, et si un opérateur souhaite modifier dans l'IHM dédiée du FPR2 une fiche impactée, il devra faire en sorte qu'elle obéisse aux contraintes du nouveau système.

8. *déterminer les exceptions :* étape très importante dans le cadre de la rédaction des SFD, il faut impérativement rattraper le maximum d'exceptions. Si une donnée n'est pas reprise, il faut savoir pourquoi. On doit tracer une exception afin d'analyser si le problème vient du script ou de la donnée source, pour ensuite apporter les corrections nécessaires au script ou à la donnée source ;
9. *déterminer les informations à logguer :* il ne faut pas négliger cette étape. Durant l'exécution du script, il faut savoir ce qui se déroule correctement, ce qui doit faire l'objet d'avertissement et ce qui doit faire l'objet d'erreurs. Il ne faut pas effectuer une action sans envisager le succès ou l'échec de celle-ci ;
10. *déterminer les informations d'intégration :* il ne suffit pas de convertir simplement les données, il faut savoir quand et comment une donnée a été transformée, afin que ces données reprises puissent être distinguées des données produites par le nouveau système. Le script doit prévoir cette information.

Dans le FPR2
J'ai mis dans l'historique des fiches la notion de reprise, en distinguant l'inscription dans le système source et la reprise dans le système cible.

Les SFD ainsi rédigées doivent pouvoir être comprises par un personnel qui n'a pas participé à leur rédaction. Dans la mesure où ce document peut être lu par plusieurs types de personnes (techniciens et fonctionnels), il convient de le rendre le plus claire possible. De ce fait, des codes couleurs particuliers peuvent être utilisés pour désigner des données sources, des données de destination, des fichiers de travail (tables de correspondance, par exemple). Les termes doivent également être intelligibles pour des techniciens mais également pour des fonctionnels.

Description du script

Architecture des dossiers Le script comprend plusieurs fichiers répartis dans plusieurs dossiers :
— `/` : dossier racine, il comprend tous les scripts PERL, ainsi que scripts de "nettoyage", permettant de supprimer ce qu'il faut dans les dossiers pour rejouer une reprise sans s'y perdre. Par exemple `nettoyer_donnees.sh` peut servir pour supprimer les CSV produits ;
— `correspondance` : dans ce dossier seront placés par le script les JavaScript Object Notation (JSON) servant à réaliser les correspondances. Le script utilisera en mémoire ces fichiers ;
— `donnees` : la donnée source figure dans ce dossier ;
— `donnees_saines` : la donnée source transformée en début de script pour pallier les erreurs d'encodage, par exemple ;
— `erreurs` : le script produira des erreurs de toute sorte. Il pourra par exemple faire figurer `erreurs_encodage.log` ou `erreurs_referentiel.log`. De même, il convient d'horodater les fichiers d'erreur, pour savoir quelle version du script les a générés et quelle version de la donnée source. Ainsi, on aura plutôt `erreurs_referentiel_123456789_v1.2_v2.7.log` ;
— `sortie` : ensemble des CSV données de destination qui sont produits par le script. Ce sont ces CSV qui seront intégrés dans la nouvelle application.

— `source_correspondance` : ensemble des CSV qui serviront pour effectuer les correspondances. Ces fichiers sont manipulables par les fonctionnels, qui peuvent apporter des précision. Cela permet de se dégager d'une analyse fonctionnelle que les fonctionnels sont les plus à même de réaliser. Ce sont les fichiers qui seront traités par le script et transformés en JSON dans le dossier `correspondance` afin de pouvoir être utilisés en mémoire.

Dans le FPR2

Le référentiel des couleurs d'yeux n'était pas le même dans les anciens FPR et dans le nouveau, qui est bien évidemment plus précis. Il a fallu réaliser une correspondance entre les anciennes valeurs et les nouvelles, cela a donné l'occasion d'écrire un CSV administré par les fonctionnels.

De même, les différents passages du script donnent l'occasion de repérer, par exemple, des villes n'appartenant pas au référentiel des communes car il s'agissait de texte libre et des coquilles se glissaient. Il pouvait par exemple y avoir "MARSEILE", qui devait trouver sa correspondance dans le nouveau référentiel. Les fonctionnels étaient destinataires des erreurs de référentiel et enrichissaient un CSV.

Ces CSV avaient donc leur place dans le dossier `source_correspondance`.

Fonctions utiles Le script commence par quelques "`use`", que je conseille mais qui ne sont ni obligatoires, ni exhaustifs :

— `use strict` et `use warnings` serviront à voir les erreurs de manière explicite ;
— `use Data::Dumper` : utile lors de la phase de debug ;
— `use POSIX qw(strftime)` et `use Time::HiRes` : pour pouvoir utiliser l'horodatage en "timestamp" ;
— `use Data::UUID::LibUUID` : les identifiants uniques de type Universally Unique IDentifier (UUID) sont souvent utilisés pour garantir l'unicité d'une donnée, et se retrouvent dans beaucoup de tables de bases de données.
— `use JSON` : incontournable pour traiter le format JSON.
— `use Switch` : suivant la version de PERL que vous utilisez, il faudra utiliser cet import ;
— `use Path::Tiny qw(path)` : pour pouvoir lire les fichiers ;
— `use Search::Tools::UTF8` et `use Encode` : pour repérer les erreurs d'encodage et y remédier ;
— `use Term::ProgressBar` : pour afficher des barres de progression dans la console quand le script est en train de s'exécuter. Ce n'est pas une option graphique superflue, elle constitue un bon indicateur de l'avancée des travaux.

Je conseille ensuite un `require "$path/Util.pm";` qui permet de mutualiser toutes les fonctions communes dans un seul fichier.

Également, je conseille dans ces fonctions commune de créer une fonction `chronomètre` qui permet de vérifier le temps écoulé entre chaque étape ainsi que celui écoulé depuis le début. Alliée à la barre de progression, il est donc possible de comparer le délai d'exécution du script sur des machines différentes ou selon des versions de scripts ou de données source différentes. On peut donc privilégier telle ou telle machine pour la reprise et éviter des régressions de performance entre les versions.

Vérification de la donnée source Le script commencera par vérifier les données source, en écrivant les informations souhaitées dans le journal ou la sortie console selon l'utilisation qu'on veut en faire :

— la présence des fichiers attendus (par exemple *"Fichier «FICHE_S» manquant"*) : certains fichiers sont obligatoires pour une reprise, d'autres peuvent être ignorés pour tester une reprise partielle ;
— la date des données doit apparaître (par exemple *"Fiches S extraites à la date du XXX"*) : il faut connaître la date d'extraction des données source pour ne pas commettre d'erreurs lors de l'analyse. C'est typiquement une information qui apparaît dans la console ;
— l'intégrité des données (par exemple *"en-tête «date d'inscription» manquant dans le fichier «FICHE_S»"* ou *"donnée manquante ligne XXX"*) : certaines lignes peuvent être ignorées mais il convient alors de prêter attention aux erreurs d'intégration si on souhaite réaliser un test en ignorant les lignes incriminées ;
— l'encodage (par exemple *"encodage inconnu ligne XXX du fichier FICHE_S"*) : les erreurs d'encodage peuvent être repérées et le script peut produire des fichiers sains en transformant certains caractères. C'est l'objectif du dossier `donnees_saines` décrit plus haut.

On voit qu'il est question de parcourir les fichiers ligne par ligne, ce qui induit une complexité au moins linéaire. C'est le prix à payer pour garantir une reprise de qualité : avoir une complexité inférieure, logarithmique ou racinaire par exemple, suppose que vous ne passiez pas par l'intégralité des lignes, le choix des lignes ignorées doit alors être effectué avec précaution.

Une fois cette vérification terminée, la transformation des données en tant que telle peut avoir lieu.

Transformation de la donnée source Il s'agit tout simplement de traduire en code ce qui aura été indiqué dans la SFD.

Ce que je conseille est de coder dans un premier temps de manière assez brute, pour ensuite chercher les éventuelles factorisations de code. Cela permet de pouvoir tester rapidement le script et de ne pas faire d'erreur en pensant optimiser en avance un algorithme qu'en réalité on complique. En outre, il vaut mieux avoir une compréhension de ce qu'il se passe pour toutes les données pour ensuite mutualiser ce qui peut l'être.

Dans le FPR2

Bien évidemment que tous les types de fiches avaient beaucoup de points commun. J'ai d'abord codé ce qu'il se passait pour chaque type de fiche, pour ensuite voir les fonctions que l'on pouvait créer pour effectuer les tâches répétées parmi tous les types.

Statistiques Il ne faut pas oublier ce point : vous avez déjà vérifié normalement que la donnée source que vous avez reçue correspond bien à ce qu'il y avait dans l'application, selon des métriques particulières. Il convient alors de vérifier que ce que le script a produit s'est fait sans dégradation de la donnée, ou du moins sans dégradation non maîtrisée, car vous pouvez tout à fait avoir fait le choix d'altérer la donnée de manière volontaire pendant l'exécution du script.

Le script devra donc comptabiliser les entités que vous aurez déterminé significatives pour témoigner de l'intégrité de la donnée produite.

5

Intégration de la donnée

C'est l'étape la plus simple.

5.1 Méthode d'intégration

Une fois que la donnée est transformée, nous avons obtenu des fichiers à intégrer dans le système cible. Nous avons précédemment vu qu'il était question de produire des fichiers CSV : ce sont ces fichiers qui seront injectés dans le SGBD de votre choix. Il se peut que vous utilisiez des bases NoSQL, auquel cas vous n'avez pas généré de CSV mais des fichiers JSON qui ne seront pas compliqués à intégrer. Si vous avez fait le choix du NoSQL c'est parce que vous maîtrisiez les intégrations de nouvelles données.

Dans le cas contraire, vous manipulez un SGBD plus traditionnel, certainement PostGreSQL ou Oracle, voire MariaDB ou MySQL. Je ne saurais que vous conseiller de migrer vers PostGreSQL, technologie libre ayant déjà fait ses preuves. C'est en tout cas la technologie sur laquelle se base mes conseils d'intégration de données ; nul doute que le procédé que je décris est transposable dans la technologie de votre choix.

L'outil qui permet selon moi d'intégrer les données le plus rapidement et de la manière la plus transparente est PGLoader [1].

Pour l'intégration de la donnée, le mieux est de disposer tout d'abord d'une base de données vide. Cela permet de régler facilement les problèmes d'identifiant d'autoincrément, d'index, de trigger et de contraintes. Vous pouvez intégrer les données dans une base en contenant déjà d'autres, mais ça rajoute de la complexité à l'opération et il faudra gérer ces notions.

5.2 Autoincrement

L'autoincrement est un compteur automatique dans une base de données qui s'incrémente à chaque nouvelle entrée. Il y a donc deux possibilités : soit laisser vide la colonne concernée par cet autoincrement lors de la transformation des données de manière à ce qu'elle soit remplie lors de l'intégration des données dans la nouvelle base, soit remplir cette colonne à l'aide du script de reprise. Je conseille fortement cette seconde solution, car on garde la maîtrise de la donnée et on peut appliquer les règles particulières qu'on souhaite (numéros réservés par exemple).

C'est donc à cet étape que vous devez prendre en compte les autoincréments pour actualiser le script.

5.3 Erreurs, contraintes et trigger

La plupart des erreurs que vous rencontrerez seront des erreurs de données vides alors que le système s'attend à quelque chose ("`NOT NULL`"). Il faudra donc enrichir la SFD pour prévoir les différentes causes de donnée vide et modifier le script en conséquence.

Pour ce qui est des contraintes, je conseille de toutes les lever avant l'intégration, pour les remettre par la suite. Les messages d'erreur générés devront alors être étudiés pour en chercher la cause et modifier le script afin de les faire disparaître. Il s'agit d'une opération simple mais qui nécessite une bonne connaissance non seulement du script, mais également de la base de données du système cible.

Les triggers doivent bien être placés avant l'injection. Là encore, selon la complexité des triggers que vous aurez créés, il est possible que vous rencontriez des erreurs, en particulier quand il s'agit de créer des entités devant être liées à des données que vous aurez créées avec le script. Là encore, pas de difficulté, il suffit d'être méthodique pour associer les bons éléments entre eux. Ce que je conseille pour traiter cette partie est de procéder

1. Formidable outil de Dimitri Fontaine disponible ici : https://github.com/dimitri/pgloader

par échantillons, ce qui permet de comprendre la logique d'intégration et de déclenchement des triggers, pour construire l'algorithme qui sera ensuite testé avec la totalité des données.

5.4 Statistiques

Encore une fois, il faudra vérifier que la donnée intégrée a pu l'être sans altération non désirée. Il s'agit ici de mettre en place des métriques côté base de données pour vérifier que l'ensemble des éléments souhaités y soit.

Il ne s'agit pas de la dernière vérification, puisque la vérification finale s'effectue dans l'application, comme nous le verrons.

5.5 Tests

Il y a deux choses à tester : que l'intégration des données, dans leur forme, satisfasse aux souhaits des fonctionnels, et que techniquement tout se déroule sans accroc. Dans l'idéal, afin de ne pas avoir à demander plusieurs fois les mêmes tests aux fonctionnels, il faut s'assurer de la technique en amont, ou être en mesure de repasser les tests fonctionnels après une modification technique.

Par exemple : les fonctionnels ont réalisé une batterie de tests, et ensuite des modifications ont été réalisées sur le script afin de rajouter des informations techniques (autoincrement, par exemple), le MDD a été modifié ou le serveur cible d'intégration des données a subi quelques changements. Les fonctionnels n'ont pas à être impactés par ces changements, il faut donc s'assurer que les tests fonctionnels qu'ils ont réalisés soient toujours valables avec les nouvelles contraintes techniques. Il faut donc s'approprier les cahiers de test.

5.5.1 Tests fonctionnels

Afin de garantir une totale impartialité et n'induire aucun biais dans le traitement de la donnée, il vaut mieux confier aux fonctionnels la rédaction de leurs cahiers de tests et une certaine indépendance vis-à-vis de l'exécution de ceux-ci (en leur fixant tout de même des échéances !).

En revanche, il faut documenter les tests, notamment par le biais d'un document versionné qui sert de base d'échange avec les fonctionnels, et offrir à ceux-ci une plate-forme de test qui correspond à la dernière version de l'application.

Document de test

Ce document est composé de plusieurs parties :

- *déroulé des tests :* décrit la manière dont les tests doivent être réalisés, de l'envoi de la donnée source à son intégration sur la plate-forme de test ;
- *description de la donnée envoyée :* pour connaître la base de travail ;
- *modification du script par rapport à la reprise précédente :* ce journal des modifications permet aux fonctionnels de comprendre ce qu'ils peuvent avoir à tester et ont ainsi moins l'impression de travailler en aveugle ;
- *déroulé du script :*
  - cette partie contient les informations du journal produites par le script (durée totale et durées par étape, nombre d'éléments comptés dans le cadre de la statistique, informations techniques d'où a été exécuté le script, etc.) ;
  - également, il faut préciser les fichiers produits par script : fichiers CSV ayant vocation à être intégrés notamment ;
  - les erreurs sont importantes elles aussi. Typées (“encodage”, “ville”, “date”, etc.), elles permettront de comprendre ce qui n'a pas fonctionné pour apporter des solutions techniques ou fonctionnelles.
- *intégration des données :* sensiblement comme pour le “déroulé du script”, sauf qu'il s'agit là de s'intéresser à la partie où les données produites par le scripts sont intégrées en base ;
- *remarques aux responsables fonctionnels :* il convient d'expliquer tout ce qui a été écrit avant et de proposer des solutions ;
- *tâches pour le chef de projet :* dans la mesure où nous donnons des consignes aux fonctionnels, il faut en retour leur indiquer ce qui est envisagé pour la prochaine version du script.

Plate-forme de test

La plate-forme de test doit tout d'abord héberger la dernière version de l'application. Tester une reprise de données sur une version d'application obsolète ne garantit pas que les résultats obtenus soient toujours valables.

Ensuite, elle doit contenir les données de manière temporaire, sécurisée et tracée :

— *temporaire :* les données ne doivent être présentes que lors des phases de test déterminées à l'avance. Une fois la période de test terminée, les données doivent être détruites ;

— *sécurisée :* les testeurs doivent être nommément désignés et disposer de comptes personnels. Les personnes non habilitées ne doivent pas avoir accès aux données ;

— *tracée :* il s'agit de données de production, même si elles sont utilisées dans le cadre d'un test de reprise, il convient de garder les traces d'accès afin d'éviter que l'on se serve de la plate-forme de test comme un moyen détourné d'utiliser l'application sans être tracé. Après avoir vérifié que les traces correspondent bien aux cahiers de test, il est possible de supprimer les traces pour ne garder que les cahiers de test en archive.

5.5.2 Tests techniques

Valider fonctionnellement que la reprise fonctionne est une chose, s'assurer que le processus de bascule complet se déroule sans incident technique en est une autre.

Il peut s'avérer compliqué de réaliser des tests de bout en bout sur le plan technique, surtout lorsque la chaîne de transmission de la donnée implique un grand nombre d'acteurs. Il est alors bon de scinder ces tests, qui pourront donc avoir lieu en parallèle. Il y a deux choses à valider : le fait que les opérations puissent se dérouler, et le fait que les opérations puissent se dérouler avec un grand volume de données. En effet, il n'est pas forcément utile de réaliser dès le début des tests techniques avec l'intégralité des données, le tout est de s'assurer que les informations transitent correctement, ce qui représente parfois un temps de travail conséquent, en particulier lorsque de vieilles technologies sont en jeu.

Les tests techniques serviront également à choisir le mode de transfert de la donnée lors des étapes pour lesquelles vous avez le choix.

Prenons l'exemple du script : vous pouvez le faire tourner sur le serveur sur lequel le SGBD est hébergé, le faire tourner sur un ordinateur plus puissant pour transférer les données sur clé USB, ou bien d'autres solutions.

Pour la transmission des données, des protocoles de sécurité peuvent engendrer une certaine latence, bien qu'il s'agisse d'informatique. Dès lors, récupérer les données sur un disque dur crypté et le faire transiter par une personne physique qui effectuera le trajet en moto permet de gagner en rapidité.

Il faut envisager toutes les solutions, ce qui implique avoir recours à une imagination parfois débordante !

6

Bascule

La bascule est l'opération qui consiste à passer de l'ancien système au nouveau. Même si c'est par définition une des dernières étapes d'un projet, il convient de s'y prendre longtemps à l'avance car elle mobilisera un grand nombre d'acteurs et il est nécessaire de préparer au mieux la partition qui sera jouée par chacun le jour J.

Comme pour chaque étape, il faut documenter les travaux réalisés et effectuer des tests.

6.1 Document de la bascule

Je me suis basé sur un tableur qui contenait les colonnes suivantes :

— *numero de l'opération :* permet de se référer rapidement à un numéro d'opération le jour J ;
— *heure de début de l'opération :* afin que chaque acteur puisse préparer son emploi du temps, bien évidemment avec une marge d'erreur plus grande au fur et à mesure qu'on avance dans les opérations ;
— *temps écoulé :* pour que le chef de projet puisse estimer au final la durée totale de la bascule, dont le cœur se déroulera en quelques minutes ou quelques heures mais qui peut s'étaler, opérations annexes comprises, sur plusieurs jours ;

> **Dans le FPR2**
> Je me suis consacré à la reprise des données en elles-mêmes, puis des photographies associées à certaines fiches. J'ai pu le faire lors d'une phase après la nuit de la reprise, au calme.

— *temps estimé :* pour étudier les opérations qui méritent qu'on consacre du temps à leur accélération ;
— *intitulé :* permet en dehors de la reprise de se référer à une opération en particulier (exemple : "nettoyage du Serveur de Stockage en Réseau (NAS)") ;
— *description :* permet de savoir ce qui est réalisé précisément durant cette opération ;
— *acteurs :* les personnes doivent être nommément désignées et pas seulement un service. D'ailleurs, un annuaire peut être utilement mis dans un onglet du tableur ;
— *risques :* que peut-il mal se passer durant l'opération ? Quelle est la gravité et quelle est la probabilité d'occurrence ?
— *actions :* que doit-il être fait afin de réduire le risque ?
— *utilisation de l'ancien système :* dans quelles conditions utilise-t-on l'ancien système ? En effet, il est possible qu'une interruption de service ne soit pas envisageable, auquel cas comment organiser au mieux la transition entre deux systèmes ? Il s'agit la plupart du temps d'un savant mélange d'autorisations de lecture et d'écriture ;

> **Dans le FPR2**
> Il a fallu par exemple lors d'une phase consistant en l'extraction des données interdire l'alimentation des anciens fichiers pour n'en permettre que la consultation, puis autoriser l'alimentation en gardant en mémoire ce qui est nouvellement inscrit (d'où l'intérêt d'une reprise de nuit, les inscriptions étant moins nombreuses), pour ensuite interdire à nouveau l'alimentation mais également la consultation des anciens fichiers afin de se consacrer exclusivement au nouveau, devenu opérationnel, après y avoir intégré les fiches qui avaient été inscrites entre temps.
> Ce serait d'ailleurs mon conseil : ne pas faire de reprise de données "vivantes". Il vaut mieux prendre une photographie des données, les reprendre, puis rendre les données reprises vivantes. Cela évite bien des problèmes mais nécessite une discipline et une rapidité dans l'exécution de la bascule.

— *utilisation du nouveau système :* idem supra.

Ces opérations étant séquencées en plusieurs phases, chaque phase étant un regroupement de lignes. Il y a des phases préparatoires, des phases “cœur de reprise” qui se déroulent de préférence la nuit, puis des phases “correctrices” ou “annexes” qui se déroulent forcément avec moins de pression.

6.2 Tests

6.2.1 Théorique

Il faut tout d'abord s'assurer que le document évoqué précédemment ait bien été lu et accepté par l'ensemble des parties prenantes. L'idéal, selon la taille de l'équipe, est de faire une bascule en mode “bac à sable” dans une salle, assis autour d'une table, pourquoi pas en associant en visioconférence suivant les disponibilités de chacun. Comme une équipe de tournage qui relit le script d'un film qu'elle doit tourner, l'ensemble des acteurs lit son texte successivement et apporte des remarques.

Cet exercice doit inclure également le maximum de systèmes tiers dont dépendent l'application : il faut en effet qu'ils sachent comment évoluera celle-ci, toutes les relations (en particulier les interconnexions) devant faire l'objet d'un contrat d'interface précis.

Cet exercice théorique est très révélateur et permet en une demi-journée de lever des lièvres auxquels il faudra s'atteler avant de réaliser un premier test en conditions réelles.

6.2.2 Pratique

Mon premier conseil est de réaliser des tests de bascule sur la plate-forme de production. A priori, vu qu'il s'agit d'une nouvelle application, cela ne pose pas de problème.

La bascule doit être jouée plusieurs fois, pas forcément d'une seule traite tout le temps mais au moins une fois.

Au cours des tests de bascule, il faut prévoir un retour arrière si tout ne se déroule pas comme prévu. Chaque opération décrite dans le document peut échouer, il est intéressant de prévoir les règles de retour en arrière pour chacune d'entre elles. Le chef de projet doit être entièrement consacré à la bascule, et être capable d'arbitrer et de juger une situation, par rapport notamment au nombre d'erreurs rencontrées et surtout à l'effort qu'il faut consentir pour y remédier. Il est possible de tolérer un certain nombre d'erreurs dont on sait qu'elles seront rattrapables une fois la bascule terminée.

Lors de la bascule, il y a une dernière vérification à effectuer : je conseille dans une application de prévoir une interface d'administration avec au moins les métriques des données. Il s'agit de disposer des statistiques évoquées à plusieurs reprises dans ce document. Ce sera la vérification finale, qui doit normalement correspondre aux données source, modulo les altérations qui auront été acceptées ou déterminées.

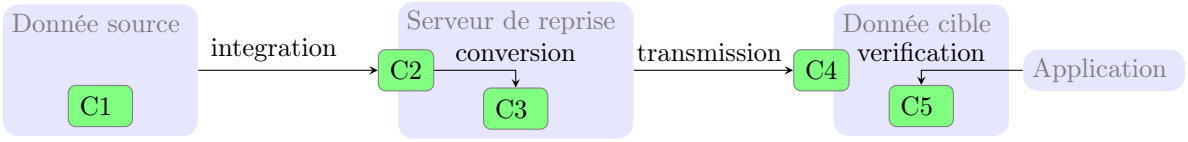

FIGURE 6.1 – Différentes vérifications lors de la reprise

Les différentes vérifications sont donc, au moins :

1. avant l'envoi de la donnée source, à l'endroit où elles sont stockées ;
2. après le transfert, sur le serveur de réception. Cette opération peut être réalisée par le script ;
3. après la exécution du script et donc transformation de la donnée, toujours sur le serveur. Là encore le script peut être utile pour accomplir cette tâche ;
4. après intégration, comptabilisation en base ;
5. dernière vérification en utilisant directement l'application.

6.3 Le jour J

Le “jour J” sera très probablement la “nuit N”, afin de limiter à la fois la rupture de service mais également l'évolution des données.

Il s'agira simplement de dérouler le scénario qui aura fait l'objet de plusieurs jours, semaines ou mois de préparation. Une grande lucidité est requise, le sommeil ne devra donc pas manquer, tout comme une bonne

hydratation et une bonne alimentation en particulier lors des étapes les plus délicates qui pourront s'avérer éprouvantes. Une bonne sieste de plusieurs heures avant la nuit peut être salvatrice.

Le jour J commence par un appel des troupes. Une conférence téléphonique est très pratique car elle permet d'avoir les mains libres et à quiconque d'intervenir pour signaler un incident ou une étape terminée.

Il ne faut pas mettre tous ses œufs dans le même panier : toute l'équipe ne devra pas être sollicitée, et il ne faudra pas forcément solliciter les meilleurs dans tous les domaines. Des choix devront être réalisés, car, que la bascule se déroule bien ou non, il faudra qu'une équipe fraîche et dispose puisse prendre le relais.

Il faudra en revanche avoir plusieurs œufs dans plusieurs paniers. En effet, certains acteurs de la bascule n'auront qu'un rôle mineur à jouer et il est possible que ce projet très important pour vous n'ait pas la même priorité dans leur esprit et que des oublis soient commis. De ce fait, leur adjoindre un membre de votre équipe ou les associer à un autre membre plus impliqué permet de s'assurer de la présence effective de tout le monde.

Enfin, la principale différence avec tous les travaux réalisés précédemment est que cette fois-ci l'utilisateur est directement concerné. L'objectif est de rendre les opérations aussi transparentes que possible pour celui-ci. Dans l'idéal, une organisation méthodique des bascules d'URL suffit, les utilisateurs gardant le même lien pour accéder à l'application, mais il faudra certainement assortir cela à différents messages :

— sur les anciens clients lorsqu'il s'agit de client lourd ;
— sur les intranets ;
— par messagerie ;
— par tout support au sein de l'organisation (reportage écrit dans le journal interne).

L'utilisateur devra être informé des changements et des nouveautés, et ne devra plus avoir accès à des applications ou données obsolètes, le plus fastidieux étant de se charger des clients lourds, pour lesquels qu'il n'est parfois pas prévu un télédéploiement, auquel cas l'envoi d'un message d'erreur à ces clients lourds en particulier peut indiquer à l'utilisateur de migrer vers la nouvelle application, bien que ce soit le rôle de l'équipe projet de s'assurer de l'éradication de ces anciens clients.

6.4 La bascule, et après ?

Ça y est, les nouvelles données sont intégrées dans l'application. La reprise est derrière nous, mais tout n'est pas terminé pour autant.

En effet, il faut prêter une attention particulière au traitement dans l'application des données intégrées. Les règles métier de l'ancienne et de la nouvelle application ne sont pas forcément les mêmes, elles le sont même rarement. Lorsque la nouvelle est moins contraignante, il n'y a pas de problème particulier, mais lorsque c'est l'inverse, la donnée reprise peut en devenir un. Je conseille de prévoir la consultation en lecture de la donnée reprise, mais dès lors qu'il s'agira d'effectuer des modifications, d'être intransigeant sur l'application des nouvelles règles métier. Un message peut être affiché lorsqu'une modification de donnée reprise est sollicitée :

> *"Attention, il s'agit d'une donnée reprise d'un ancien système. Pour qu'elle puisse être intégrée, il est possible qu'il faille renseigner des éléments qui n'étaient pas renseignés auparavant."*

Ainsi, la donnée reprise sera petit à petit nettoyée, au fil de la vie de la nouvelle application. Une métrique peut être incluse dans l'interface d'administration, donnant l'évolution de données reprises restant dans la base de données. Si les fonctionnels ont le temps, ils peuvent même se consacrer à un travail de toilettage de la donnée reprise pour que 100% des données dans l'application respectent les nouvelles règles métier, pourquoi pas en se basant sur une analyse des données ne respectant pas les règles métier. Techniquement, c'est réalisable lors de la conception du script de reprise, mais il est plus facile de vérifier les règles métier par un nouveau script au sein de l'application, les règles métier étant par définition déjà incluses dans le code.

On peut donc avoir dans l'interface d'administration un camembert indiquant la proportion de données reprise et un histogramme indiquant quelles règles métier ne sont pas respectées :

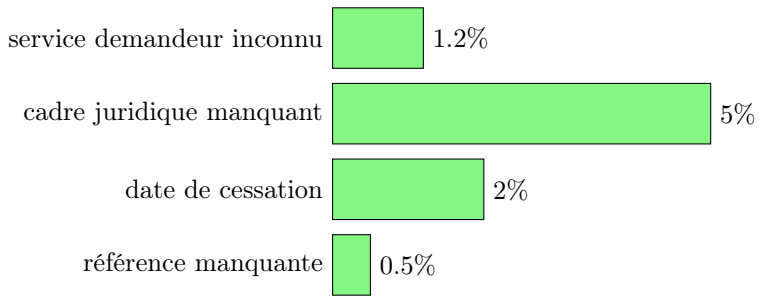

FIGURE 6.2 – Conflits de règles métier

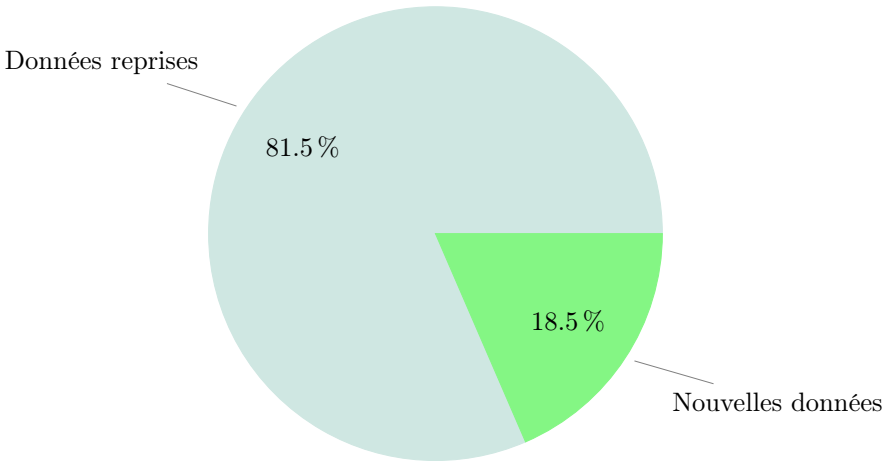

FIGURE 6.3 – Taux de propreté de la donnée

Un sujet annexe qui mériterait à lui seul un article est celui des systèmes tiers : certains pourront cohabiter sans difficulté avec la nouvelle application, d'autres devront rester durant un certain temps interconnectés avec l'ancienne, principalement par manque de main d'œuvre pour faire évoluer leur côté de l'interface. Il s'agira alors de rétrocéder la nouvelle donnée dans l'ancien système, en l'altérant forcément lors de la conversion, celle-ci pouvant être documentée grâce au travail réalisé lors de la conception de la reprise de données : il s'agit globalement de parcourir le chemin inverse.

7

Conclusion

La reprise de données est terminée, la nouvelle application fonctionne correctement et le chef de projet a le sentiment du devoir accompli.

Ce document vous a présenté les bases d'une reprise qui s'est très bien déroulée, et il n'a pour cela fallu que deux ingrédients : de la méthode et du dialogue entre les différentes parties prenantes, le matériau de base étant bien évidemment une implication du chef de projet qui doit maîtriser son sujet, en l'occurrence ici plus précisément la nouvelle application et les données qu'elle traite. Comme vous avez pu le remarquer au travers du petit nombre de précisions techniques, il n'est pas nécessaire d'être un fin technicien pour réussir (même si cela aide fortement), du moment que vous vous entourez de personnes clés sur les bonnes tâches.

Les différents éléments de ce document pourront donc être intégrés dans une stratégie de reprise prenant en compte les caractéristiques de votre projet.

Acronymes

API Application Programming Interface. 11, *Glossaire :* API

CIMI Communauté Informatique au sein du Ministère de l'Intérieur. 3
CSV Comma Separated Value. 17, 20, 22–24, 27, 28

DGSI Direction générale de la Sécurité intérieure. 14

ETL Extract-Transform-Load. 7

FPR GN Fichier des Personnes Recherchées de la Gendarmerie Nationale. 7
FPR PN Fichier des Personnes Recherchées de la Police Nationale. 7
FPR2 nouveau Fichier des Personnes Recherchées. 3

GCOS7 General Comprehensive Operating System. 15

IHM Interface Homme Machine. 22, 23
INSEE Institut National de la Statistique et des Études Économiques. 21, 22

JSON JavaScript Object Notation. 23, 24, 27

MCO Maintien en Condition Opérationnelle. 16

NAS Network Attached Storage, en français Serveur de Stockage en Réseau. 31

RGPD Règlement Général sur la Protection des Données. 11

SFD Spécification Fonctionnelle Détaillée. 19–21, 23, 25, 27
SGBD Système de Gestion de Base de Données. 15, 17, 19, 20, 27, 29
ST(SI)^2 Service des Technologies et des Systèmes d'Information de la Sécurité Intérieure. 3, 7

UTF-8 Universal Character Set Transformation Format - 8 bits. 17, 20
UUID Universally Unique IDentifier. 24

Glossaire

API une API est un moyen normé d'accéder à des données d'une application. Par exemple, la sécurité sociale peut offrir une API *"délivrance de la liste des numéros de sécurité sociale sur présentation d'un triplet nom/prénom/date de naissance"*, cette API étant ouverte à des clients qui auront signé un contrat d'interface précisant les modalités de l'interconnexion. 11

Client dans ce document, un "client" s'agit réellement de la partie de l'application offerte aux utilisateurs pour qu'ils envoient des requêtes au serveur, dans une architecture "client/serveur" classique. Il faut faire la distinction avec les références aux fonctionnels. 33

Fonctionnels, ou responsables fonctionnels Dans une gestion de projet, il est davantage fait référence au "client". Ici, pour ne pas confondre le "client" applicatif (lourd ou léger), il est fait mention des "fonctionnels", qui sont des parties prenantes au projet ayant la même fonction qu'un "client" traditionnel: contribuent à définir le besoin, participent à la recette, etc. 13, 14, 17, 21–24, 28, 33

Méthode agile méthode de développement qui permet, en relation continue avec les fonctionnels, de développer l'application en partant d'un embryon qui est alimenté en nouvelles fonctionnalités testées au fur et à mesure. On parle de développement itératif (étape par étape, chacune des étapes étant une "itération") et incrémental (on fait évoluer l'application petit à petit). 23

Règles métier les règles métier sont l'ensemble des contraintes sur la gestion des données dans l'application demandées par les fonctionnels afin que l'application fournisse le service souhaité. Par exemple: *"si je crée une personne mineure, il ne faut pas qu'elle puisse avoir le permis de conduire"*. 7, 20, 23, 33