Mémoire de fin d'études

Migration de bases de données

Julien Ivars

Master informatique et mobilité
UHA 4.0.5 - Promotion 2024 / 2025

Alternance réalisée chez **UNIT SOLUTIONS AG**

Tuteur professionnel
M. Cédric Martin

Tuteur pédagogique
M. Mounir ELBAZ

Remerciements

J'aimerais remercier monsieur le directeur d'Unit Solutions M. Thierry MOEBEL pour m'avoir donné l'opportunité de rejoindre l'entreprise et d'effectuer ma première année de master en alternance. Je le remercie également d'avoir pris en compte mes intérêts en me confiant un projet captivant, correspondant parfaitement aux attentes de mon année. De plus, je suis reconnaissant qu'il ait prolongé mon contrat pour l'année prochaine, me permettant ainsi de me lancer dans le monde du travail et de poursuivre mon évolution au sein de l'entreprise.

Je souhaite exprimer ma gratitude envers M. Cédric MARTIN, mon tuteur en entreprise, pour son accompagnement tout au long de l'année sur le projet. Sa transmission de connaissances techniques et ses explications sur l'architecture et le fonctionnement du projet ont été d'une grande aide pour moi.

Je remercie chaleureusement tous mes collègues chez Unit Solutions pour leur partage de connaissances, leur bonne humeur et leur soutien.

Je tiens à exprimer ma reconnaissance envers toute l'équipe pédagogique de l'UHA 4.0, notamment M. Mounir ELBAZ, M. Pierre-Alain MULLER, M. Florent BOURGEOIS, M. Daniel DA FONSECA, M. Pierre SCHULLER et Mme. Audrey, ainsi que les étudiants de l'UHA 4.0. Leur soutien, leur partage de connaissances, leur accompagnement et leurs conseils au long de l'année m'ont permis de mener à bien mon projet professionnel.

Enfin, je souhaite exprimer ma gratitude envers les relecteurs de ce rapport pour leurs précieux conseils, qui m'ont permis de mener à bien l'écriture de ce rapport.

Sommaire

Introduction

Après avoir réalisé mon parcours de license professionnelle « Développeur nformatique » au sein de l'UHA et obtenu mon diplôme, j'ai souhaité approfondir mes connaissance rejoignant le cursus master proposé par l'UHA 4.0 qui fait suite à la license.

Mon parcours de master à été réalisé au sein de l'entreprise Unit Solutions basée à Allschwil en Suisse, qui s'était déjà proposée de me suivre dans mon cursus universitaires pour les deux années précédentes. Mes contributions principales se sont orientées sur le projet InfSuite et l'environnement l'entourant. L'application pour laquelle j'ai pu apporter ma contribution à comme objectif premier de gérer le suivi et la maintenance d'état d'ouvrages d'art.

Dans ce mémoire je vous présenterais les détails du projet InfSuite et de ma ccontribution au projet. J'ai principalement eu pour objectif de planifier et de réaliser une migration de base de données. En effet, la base de données étant un point clef de l'application, une maintenance de cette dernière est nécessaire pour assurer une certaine pérennité de l'application. Cette étape de migraton s'inscrit dans un projet de maintenir les technologies de l'application à jour et de permettre de palier à d'autres problèmes.

Dans ce document je commencerais par présenter ce qui m'a amené à rejoindre le cursus master et les compétence acquises durant ma formation, j'aborderais par la suite les enjeux, une analyse et le plan d'action de la migrationet, puis j'expliquerais la réalisation et les problèmes rencontrer et enfin je pourrais conclure ce document.

1. Contexte

Dans cette première section, je vais remettre en contexte le concept de formation, détailler mon parcours académique et présenter l'entreprise qui m'accueille en alternance.

1.1. Présentation de la formation

L'UHA 4.0 propose des formats de formation légèrement différents des cursus traditionnels pour favoriser une immersion professionnelle tout en poursuivant des études universitaires. La formation inclut un stage obligatoire d'une durée minimale de 6 mois en complément de la période de formation. Cette immersion professionnelle fournit aux étudiants les outils essentiels pour intégrer le monde du travail. La première année introduit le développement et fournit les outils fondamentaux pour comprendre la logique de la programmation. La deuxième année approfondit ces connaissances en abordant la programmation orientée objet avec des langages tels que Java ou C#. La troisième année de Licence ajoute une méthodologie approfondie de gestion de projet en mode agile.

Après ces trois années, les étudiants peuvent, sur admission, rejoindre le parcours master de l'UHA 4.0. Ce parcours aborde les fondements de la recherche et des thématiques variées telles que la cybersécurité et l'algorithmie. Le master, également en alternance, impose une période en entreprise plus soutenue, passant de 6 à 9 mois, tout en réalisant les projets donnés par les encadrants de l'UHA 4.0.

1.2. Mon parcours académique

Après l'obtention de mon baccalauréat scientifique, j'ai intégré l'UHA 4.0 pour y suivre trois années de formation. Ces années m'ont permis de réaliser deux stages de 6 mois dans deux entreprises et une troisième année en alternance chez Unit Solutions AG. Après l'obtention de ma Licence, j'ai décidé de poursuivre en master à l'UHA 4.0 pour compléter mes acquis et acquérir de nouvelles compétences. Cette formation en alternance privilégie le temps en entreprise, passant de 6 à 9 mois.

Durant ma première année en DU 4.0.4, j'ai eu l'opportunité d'aborder des sujet complexes d'algorithmie, d'intelligence artificielle, de fouille de données et quelques notions de sécurité informatique. Pour mettre en pratique nos acquis lors des topos nous réalisons, en groupe de 3 élèves, un fil-rouge regroupant les connaissances acquises dans un sujet concret. Durant cette première année, nous avons pu mettre en place un système de suivi et d'alerte de l'état d'une plante par analyse d'./images, relevé de données environnementales, algorithmes de prédiction, …

1.3. L'entreprise Unit solutions

Unit Solutions est une entreprise Suisse, située à Allschwil dans le canton de Bâle-Campagne. Elle a été fondée en 1986 initialement sous le nom de « CAD Rechenzentrum AG », nom qui a été changé en 2015 pour « Unit Solutions AG ». L'entreprise est spécialisée dans la conception d'applications modernes web et mobiles, dans les systèmes de localisation (Système d'information géographique SIG, plans, modèles 3D, photos, GPS) et dans l'exploitation de données et la génération de rapports. Unit Solutions comptabilise une vingtaine d'employés décomposés en plusieurs équipes, avec pour chacune d'entre elles, un responsable (Cf. annexe 1). L'équipe du support est directement en relation avec les clients, ils sont chargés du bon fonctionnement des outils utilisés en interne à l'entreprise et des serveurs utilisés pour la publication des différentes solutions développées. Ensuite chaque équipe a la responsabilité du suivi et du développement d'une ou plusieurs applications. J'ai été intégré dans l'équipe de développement Mobility-team (Cf. la figure 1, ci- après). Figure 1 : organigramme de l'équipe de développement Mobility-team L'entreprise concentre principalement son activité sur la réalisation des projets InfKuba, Kuba, Mobilité douce et Observo :

- InfKuba et Kuba sont une suite de logiciels permettant le suivi et la maintenance d'infrastructures tels qu'une route, un pont, une barrière anti-avalanche ou encore un filet anti-éboulement.
- Observo est une application mobile permettant de relever des observations sur des infrastructures au travers d'une carte. Initialement autonome, elle tend à collaborer avec la suite InfKuba. En effet, nous pouvons désormais importer et exporter des données entre les applications. L'utilisation d'Observo est plutôt destinée à une utilisation en déplacement en se rendant directement auprès de l'infrastructure ciblée.
- Mobilité douce est une application web, facilitant la coordination à travers les cantons de Suisse pour l'entretien de chemins, la planification d'itinéraires ainsi que la signalisation pour les modes de mobilité douce comme la randonnée, le vélo et le roller.

2. Etat de l'art

2.1. Présentation d'InfSuite

Début **2016**, les différents cantons Suisses étaient sur le client lourd Kuba, l'application mère d'InfSuite. Ce logiciel était payé par l'OFROU pour une meilleure gestion des infrastructures routières en Suisse.

En **2019**, elle lance un appel d'offres dans le but de réduire les coûts liés aux applications qu'elle utilise. L'application Kuba étant une apllication hautement complète et complexe, le budget n'était alors pas forcément adapté à tous les cas d'usages des différents cantons, certains yant des besoins différents en termes de gestion.

Unit Solutions décide alors de se lancer dans la conception d'une nouvelle solution nommée InfSuite, qui reprnd le principe de l'application mère Kuba. La grande différence sur cette nouvelles application , est le découpage plus fin des différentes fonctionnalités. elle permet de proposer aux différents clients de ne leur orctroyer l'accès à certaines fonctionnalités que s'ils en ont réellement besoin et permer ainsi de faire coller le budget au besoin.

Les principaux objetifs de l'application étaient donc:
- De n'implémenter que les fonctionnalités que les cantons seraient suceptibles d'utiliser.
- Vendre l'application à l'OFROU apuuyé par la nouvelle philosophie de l'application qui réduit les coûts de l'application.
- Livrer une version utilisable au bout d'une année et demi maxium pour permettre aux cantons de tester l'application et de s'adapter au nouveau système.

Finalement adopté par la suite par la majorité des cantons suisses, l'application InfSuite est une application géographique basée sur des technologies web récentes. Le but de l'application est de permettre de faire l'état et le suivi d'ouvrages d'art dans le temps. Lorsqu'un organisme résponable de la préservation des ouvrages d'art vient faire des constatations à une certaine date d'un ouvrage, il enregistres toutes les informations et données relatives dans l'application.
Lorsqu'une expertise ulterieure est réalisée sur l'ouvrage, de nouvelles observation seront ajoutées et avec ces nouvelles données l'application fournira un résumé de l'évolution de cet ouvrage, si il a un comportement particulier, s'il faut planifier une intervention de conservation, etc.

Toutees ces données sont accessibles depuis n'importe quel terminal et depuis n'importe ou tant que le client possède une connexion internet et un compte ayant les droits requis pour accéder aux données.
La suite logicielle InfSuite comprend plusieurs types d'outils, tous basés sur le même fonctionnement, mais avec des domaines d'application différents pour permettre

d'effectuer des constatations plus spécifiques en fonction du domaine ciblé. Par exemple, sur l'outil InfAqua, il est possible de faire l'observation de cours d'eau (lit de rivières, berges…), InfRail des voies ferrées, InfVias de routes… Pour conserver la compréhensibilité de ce document, InfKuba sera l'outil de référence pour le fonctionnement technique de l'application.

L'application présente de manière simplifiées les données relatives aux différentes constatations sur une carte. Elle représente sous forme de pastilles colotées la note moyenne des ouvrages par zones géographiques lorsque plusieurs ouvrages se trouvent très proches les uns des autres comme le montre la

En fonction du niveau de zoom, les différents ouvrages se séparent les uns des autres et permettent de voir la dernière note calculée pour l'ouvrage sous forme d'une petite épingle qui représente l'objet d'infrastructure. La couleur varie du vert au rouge indiquant respectivement que l'ouvrage est en bon état ou qu'il faut intervenir le plus rapidement possible. En ouvrant les détails de l'objet on peut retrouver un onglet regroupant les informations relatives à l'ouvrage d'art. On y retrouve des données basiques comme son nom, le canton propriétaire de l'ouvrage, ses dimensions, des commentaires, sa position géographique, des photos, etc.

D'autres onglets permettent de retrouver des informations plus précises telles que les observations de l'ouvrage, des graphiques sur l'évolution de l'ouvrage dans le temps, visualiser l'ouvrage en 3D (si fourni par le client) …

Chaque ouvrage appartient à un canton, mais il peut être prêté à d'autres cantons, comme dans le cas où un pont serait partagé entre deux cantons par exemple. Un autre exemple d'ouvrage partagé est lorsque la gestion est déléguée à une ville plutôt qu'au canton. À ce moment, dans l'application, l'ouvrage appartient au canton et est « prêté » à la ville qui en est chargé. Chaque donnée générée à partir de l'application, est rattachée au client qui en est à l'origine, ce qui permet d'avoir un historique en cas de problème. Pour permettre de rendre l'application plus légère, l'application est basée sur une architecture trois tiers.

Le premier tiers de l'application correspond à la base de données. Cette base de données est une base PostgreSql. C'est ici que toutes les données sont stockées ainsi que les relations entre chaque donnée existante. Je détaillerais les particularités techniques de cette base de données qui peuvent expliquer la pertinence de sont utilisation dans ce contexte plus loins dans ce rapport.

Le second tiers de l'application correspond à l'API. Une API web, est une interface de programmation qui permet à des applications informatiques de communiquer entre elles via Internet. Elle définit les règles et les formats de données pour faciliter l'échange d'informations entre les différentes applications.

Cette API est le serveur back-end qui permet de faire la relation entre le monde extérieur et les données brutes stockées en base de données. Le serveur est développé en C# avec

l'ORM Entity Framework. Cette interface permet notamment de mettre en forme les données pour qu'elles correspondent aux besoins du troisième et dernier tiers.

Le dernier tiers correspond à la partie visible de l'application. Appelé frontend, il met à disposition de manière visuelle et simplifiée les données. Ce dernier tiers permet également à l'utilisateur de créer, modifier ou effacer des données. Il est développé en TypeScript avec le framework Angular pour permettre de créer une application dynamique et réactive.

Le dernier service complémentaire est un serveur GIS. Ce serveur permet de fournir des informations cartographiques comme des adresses. Il permet également de délivrer les fonds de cartes. Ils peuvent être hébergés et entretenus par Unit Solutions, ou alors par d'autres entreprises comme le fond de carte Open Street Map appartenant à la fondation du même nom.

Le schéma ci-dessous représente les trois tiers de l'application communiquant ensemble et le serveur GIS permettant de fournir des informations complémentaires. L'application propose à l'utilisateur de personnaliser toute l'application. Cela comprend par exemple quel fond de carte afficher, quelles données afficher, personnaliser l'affichage de ces données tel que le type d'affichage des épingles… Ces configurations sont propres à l'utilisateur et sont sauvegardées en base de données, en parallèle de toutes les autres données de l'application.

2.2. PostgreSQL, un système open source

2.2.1. Présentation de PostgreSQL

PostgreSQL est un système de gestion de base de données relationnelle (SGBDR[1]). Le projet est initié en 1986 par Michael Stonebraker et Andrew Yu à l'Université de Californie à Berkley.

L'une des force majeur de ce système est d'être OpenSource, ce qui signifie qu'il est développé et maintenu par la communauté en plus des développements apportés par la société mère PostgreSQL.

PostgreSQL tient sa réputation de sa fiabilité, sa robustesse et sa richesse fonctionnelle que je détaillerais juste après.

2.2.2. Les principes de base

Comme dit précédement, PostgreSQL est un SGBDR[1]. Il utilise le language SQL[2] pour chercher ou manipuler les données stockées. Le système met à disposition une serie de fonctions pour permettre ces interactions, à savoir:

[1]Système de gestion de base de données relationnelle
[2]Structured Query Language

- Les transactions: un ensemble d'une ou de plusieures opérations regroupées en une seule opération atomique.
- Les vues: table virtuelle qui sélectionne et affiche des données à partir d'une ou plusieurs tables réelles.
- Les contraintes d'inrégrité: règles qui garantissent la validité et la cohérence des données dans une base de données.
- Les procédures stockées: programme écrit en SQL qui est stocké dans une base de données et peut être exécuté à la demande.
- Les triggers: procédure stockée qui est automatiquement exécutée en réponse à un événement spécifique sur une table.
- Les fonctions utilisateurs: procédure stockée qui renvoie une valeur et peut être utilisée dans une requête SQL comme une fonction intégrée.

PostgreSQL à également l'avantage d'être multiplateforme. Il peut ainsi fonctionner sur des environnements variés avec des systèmes d'exploitations différents, comme par exemple Windows, Linux, Mac,… L'une des forces de ce système de gestion de base de données réside dans sa capacité à gérer des volumes importants de données allant jusqu'à plusieurs Terraoctets. Cette gestion passe par différents points clefs, à savoir:

- L'indexation
- Le partitionnement
- La gestion du cache
- Des notions de concurrence et d'isolation
- De la réplication et du sharding[3].

2.2.3. Les avantages de PostgreSQL

PostgreSQL est un SGBDR[1] très populaire pour plusieurs raisons:

- Il est open source, ce qui signifie qu'il est gratuit et que son code source est disponible pour tous. Cela permet à la communauté de développeurs de contribuer à son amélioration et de créer des extensions pour ajouter des fonctionnalités supplémentaires.
- Il est très fiable et robuste, ce qui en fait un choix idéal pour les applications critiques et les environnements de production.
- Comme vu précédement il est très performant, grâce à son moteur de stockage et son optimiseur de requêtes. Il est capable de gérer de gros volumes de données et de supporter des charges de travail élevées.
- Le système au complet est très flexible, grâce à son architecture modulaire et à son support des extensions. Il peut s'adapter à de nombreux types d'applications et de besoins, notement pour des applications géographiques avec des besoins plus complets.
- De pars sa nature open source, il est compatible avec de nombreux langages de programmation, tels que Python, Java, C++, Ruby, PHP, etc.

Il est également important de noter que PostgreSQL tient sa popularité, au delà de ses performances et fonctionnalités déjà complètes, de par sa capacité à gérer des types de

[3] Alié à la réplication il permet de répartir la charge sur plusieures instances d'un même serveur.

données bien plus complexes. Il propose la gestion de modèles de données complexes tel que des données géographiques et des données attributaires, mais permet surtout de gérer les relations entre ces données.
Cette gestions de données complexe permet une ouverture sur d'autre système, notamment QGIS, un système d'informations géographiques, et ainsi d'étendre les fonctionnalités proposées par ce système.
En type de fichiers volumineux, on peut par exemple citer les fichiers MAJICS, RPG, référetiels vecteurs, …

2.2.4. Les inconvénients de PostgreSQL

PostgreSQL présente également quelques inconvénients qu'il faut prendre en compte:
- Il peut être plus complexe à installer et à configurer que d'autres SGBDR[1], tels que MySQL ou SQLite.
- Il peut nécessiter plus de ressources matérielles (mémoire, CPU, espace disque) que d'autres SGBDR[1] pour fonctionner de manière optimale.
- Il peut être moins performant que d'autres SGBDR[1] pour certaines tâches spécifiques, telles que les requêtes de type OLAP (Online Analytical Processing).

« PostgreSQL 12 - Guide de l'administrateur » de Guillaume Lelarge et Stéphane Schildknecht, éditions Eyrolles, 2020.

« PostgreSQL - Maîtrisez les fondamentaux du SGBD open source » de Régis Montoya, éditions ENI, 2019.

« PostgreSQL - Le guide complet de l'administrateur et du développeur » de Joshua D. Drake et Peter Eisentraut, éditions Pearson, 2018.

2.2.5. Conclusion

PostgreSQL est un SGBDR[1] open source très populaire, grâce à sa fiabilité, sa robustesse, sa richesse fonctionnelle et sa flexibilité. Il est utilisé dans de nombreux domaines, tels que la finance, la santé, l'éducation, le gouvernement, etc. Il est également compatible avec de nombreux langages de programmation et de nombreux systèmes d'exploitation. Cependant, il peut être plus complexe à installer et à configurer que d'autres SGBDR[1] et nécessiter plus de ressources matérielles. Malgré ces inconvénients, PostgreSQL reste un choix idéal pour de nombreuses applications critiques et environnements complexes.

2.3. Problématique

2.4. Existants

Effectuer une migration de base de données n'est pas une tâche anodine. Cela demande du travail en amont, il faut analyser les différents scénarios possibles, estimer un budget pour

une telle tâche, réaliser de potentiels développements, s'assurer de la fiabilités avant de mettre tout ça en pratique et enfin la réalisation de l'étape cruciale sur les environnements sensibles. Pour faire une migration de base de données il existes de nombreuses solutions, certaines plus couteuse, d'autre plus fiables, encore d'autres des plus spécialisées, … Il faut donc dans un premier temps trouvers différents outils pour comparer les avantages et leur faiblesses.

2.4.1. Les outils

On peut dans un premier temps penser à effectuer cette migration grâce à des outils spécialisés qui se chargent de faire la migration automatiquement et de s'assurer de la pérennité entre les données de l'ancienne base et les données insérée dans la nouvelle.
En prenant pour exemple l'outil Oracle Data Dump fournit par l'entreprise Oracle, ce logiciel permet de sauvegarder et restaurer des données et des métadonnées pour les bases Oracle.
Il est rapide et fiable, on peut ainsi lui fournir un fichier dump de la base source et l'outil va se charger, grâce à ce fichier, d'intégrer les données dans la base cible.
Microsoft propose sa solution alternative SSMA pour importer les types de bases Access, DB2, MySQL, Oracle, SAP ASE vers leurs différents SGBDR[1] propriétaires (à savoir les suites SQL Server).
Sur le même principe les outils MySQL workbench, AWS Database Migration Service (DMS) et PgAdmin permettent de réaliser le même type de migration vers leurs systèmes propriétaires à savoir respectivement MySQL, AWS et PostgreSQL.

Les principaux désavantages de ce genre de solutions sont:
- L'import des données reste assez stricte et ne permet pas de flexibilité, si les données ne sont pas compatiblé, il faut passer du temps à travailler le modèle de données pour essayer de palier aux incompatibilités.
- Ils peuvent également rendre les migrations onnéreuses avec certaines solutions qui coutent jusqu'à plusieures centaines d'euros pour les entreprises.
- Les logiciels proposés peuvent parfois être complexes et demander un certain temps d'adaptation avant de réellement pouvoir effectuer la tâche de migration.

2.4.2. Les types de migrations

Il est possible, comme vu précédement, d'effectuer une migration **à partir d'outils automatiques**. L'avantage est de déléguer la majorité de la complexité à une application qui va se charger d'effectuer la tâche cruciale et de réduire les risques d'erreurs pour de grosses applications. On retrouve en général des outils plus poussés pour offrir une plus grande précision et une meilleurs cohérence dans la migration des données. Il faut principalement noter, pour ce genre d'outils, qu'ils sont en général à déstination d'un certain type de base précis et qu'ils peuvent donc manquer de compatibilié. Ils peuvent également ne pas prendre en charge tous les types de bases de données ou tous les scénarios de migration, ce qui peut limiter leur utilité dans certains cas.
Enfin le cout réel de ces outils peut être élevé autant par leur prix réel fixé par l'éditeur,

mais aussi puisque la majorité du temps ils necessitent une expertise pour être pris en main avant de pouvoir être réellement utilisé.

Une alternative à prendre en compte est la migration **manuelle**. Cette solution est interessante pour les petites bases de données sans trop de complexité. Le but est d'exporter un fichier dump de la base source et de l'importer dans la base cible si possible, ou d'exporter les données sous un autre format pour l'importer simplement. Les couts de cette techniques sont faibles voir nuls et permet une flexibilité lors de la migration. Cependant le temps de migration peut se révéler très long, apporte un gros facteur de risque d'erreurs et devient complexe voir inaproprié dans le cadre de bases avec de gros volumes.

Une autre technique consiste elle à effectuer la migration **via des scripts**. Le but est de développer un processus qui va récupérer les données de la base source, effectuer si besoin des opérations sur les différentes données pour les copier dans la base cible pas la suite.
Quasiement tous les languages de programmation permettent de créer des scripts pour effectuer ce genre de migration, il suffit qu'un driver pour les bases utilisées soit disponible pour le language souhaités.
L'avantage de ce type de migration réside dans la flexibilité totale pour personnaliser le processus de migration. Il est également possibe de migrer des bases de données plus grandes et plus complexes avec des incompatibilités entre elles, puisqu'il est possible d'agir sur les données avant de les transférer vers la nouvelle base, ce qui confère un contrôle total sur la migration.
Cependant il faut noter que les couts de la migration peuvent également devenir élevés puisqu'il faut développer un script, donc payer un développeur. Il faut également prendre en compte que le temps de migration peut être long puisqu'il faut développer la solutions en amont et qu'il faut en général s'assurer de la qualité de code avant de l'utiliser sur les environnements sensibles.
Même en essayant de prévenir les différents cas d'erreurs possibles, il se peut que certaines arrivent à se glisser, dans ce cas, le risque d'erreur humaines n'est pas négligeable.

Enfin, une solution peut se porter sur **la migration en temps réelle**. La migration en temps réel est une technique qui permet de répliquer les données de la base source vers la base sible en temps réel, sans interropre les opérations sur la base source. Elle est souvent utilisée pour minimiser le temps d'arrêt lors de la migration de bases de données critiques (sceteur de la santé, du commerce, de la sécurité,…).
Ce processus de migration en temps réel implique les même couts qu'une migration par script ou par outil automatiques puisqu'elle va se reposer sur l'un de ces deux pilier. Elle va cependant permettre de minimiser l'impact sur le service actif puisqu'elle ne requiert généralement pas de mettre le service à l'arret.
Il faut cependant noter que lors de la migration, l'impact des erreurs pouvant se glisser durant le processus de migratiuon, deviendrait rapidement beaucoup plus important puisque les données sont consommées à l'instant ou elles sont migrées.
La migration ene temps réel n'est donc réellement interessante que dans le cas ou, la

relation entre une application qui ne peut pas avoir de temps d'inactivité avec les données de la bases, est critique.

2.5. Conclusion

Il existe de nombreux outils pour effectuer des migrations de données à partir de bases de données. Cependant il faut prendre en compte différents critères, comme la complexité de la migration, le budget aloué, les système source et cibles, mais aussi la philosophie à appliquer lors de cette étape. Par exemple dans un cas le temps de aloué à la migration peut être ignoré si ce qui compte est d'effectuer la migration sans arrêter le service, dans un autre cas le temps d'arrêt est moins important que la fiabilité de la migration.
Il faut donc réfléchir en amont à la manière d'effectuer cette étape mais surtout à la pertinence de cette tâche pour éviter des couts supplémentaires qui pourraient se révéler inutiles.
Toutes ces questions permettent de cibler besoin et donc le type de migration souhaité pour, par la suite, transférer ses données entre deux système.

3. Réalisation

3.1. Introduction

3.2. Analyse préliminaire

3.2.1. Pertinence et philosophie

3.2.2. Tests de performance

3.2.3.

Conclusion

Conclusion

Glossaire

- **Dump** : Un fichier « dump » est un fichier de sauvegarde qui contient une copie de toutes les données et métadonnées d'une base de données à un instant T.
- **OFROU** : L'Office Fédérale des Routes est l'autorité suisse compétente pour l'infrastructure routière
- **Sample** : Sample.

Liste des abréviations

- **OFROU** : Office Fédérale des Routes
- **sample** : sample

Bibliographie et webographie

Table des matières

Annexes

Résumé

Abstract

Mots-clés

-
-