Rapport E1

BASE DE DONNÉES

Migration automatique de deux bases de données, MySQL (Progestion) et PostgreSQL (ISIS), sur Azure Cloud dans un “InfoCentre” permettant de comparer les données de ISIS et Progestion

DAO Ngoc Mai | Développeuse Data IA | Novembre 2020

PRÉFACE

Ce projet de base de données est un projet complémentaire au projet de Machine Learning dans le but de couvrir toutes les compétences de la partie base de données nécessaires pour valider la formation Développeuse Data IA chez Simplon.

C'est un projet que j'ai fait chez Exakis pendant mon alternance. Ce projet est un vrai projet client. Donc toutes les données utilisées sur ce projet appartiennent au client, et par conséquent je n'ai pas le droit de les copier ou de les soumettre au jury. Je remercie d'avance le jury et les lecteurs de ce rapport pour leur compréhension.

1. Compréhension du besoin client

Mon client est une AIVS (Agence Immobilière à Vocation Sociale). C'est une structure à but social qui a les compétences d'une agence immobilière. Sa vocation répond à un double objectif :

- Favoriser l'accès et le maintien des personnes fragilisées dans un logement autonome tout en sécurisant le risque locatif du propriétaire
- Mobiliser des logements du parc privé pour loger les personnes en difficulté, en proposant des dispositifs adaptés aux propriétaires, notamment l'assistance à maîtrise d'ouvrage

Dans le cadre de la digitalisation de leur méthode de travail, le client a à sa disposition plusieurs applications. Deux d'entre elles sont extrêmement liées : ISIS et Progestion. Progestion est une application en-cours de déploiement chez mon client et ISIS est quant à elle utilisée depuis plusieurs années déjà. Cependant, ISIS et Progestion se basent sur des données communes, et bien évidemment ces données doivent être cohérentes entre les deux applications.

La demande client est donc la suivante : créer un rapport permettant de comparer les bases de données des deux applications et présenter les différences entre les deux.

Le client dispose donc de deux types de bases de données, une MySQL (Progestion) et une PostgreSQL (ISIS). Ces deux bases de données changent tous les jours car elles sont utilisées en production. Comme expliqué précédemment, mon client veut comparer les deux bases de données pour en voir les différences. Mais, avant de comparer ces deux bases de données, il souhaite les déplacer au même endroit, appelé InfoCentre, sur Azure Cloud et harmoniser les deux bases en PostgreSQL.

2. Traduction technique et choix technique du projet

Ce projet est un projet ETL (extraire, transformer, charger) qui est la procédure générale de copie de données d'une ou plusieurs sources dans un système de destination (figure 1).

Dans ce projet, il faut extraire deux bases de données, puis faire des transformations pour les charger dans une base de données unique sur Azure Cloud et enfin réaliser un rapport pour montrer le résultat des comparaisons entre les deux bases.

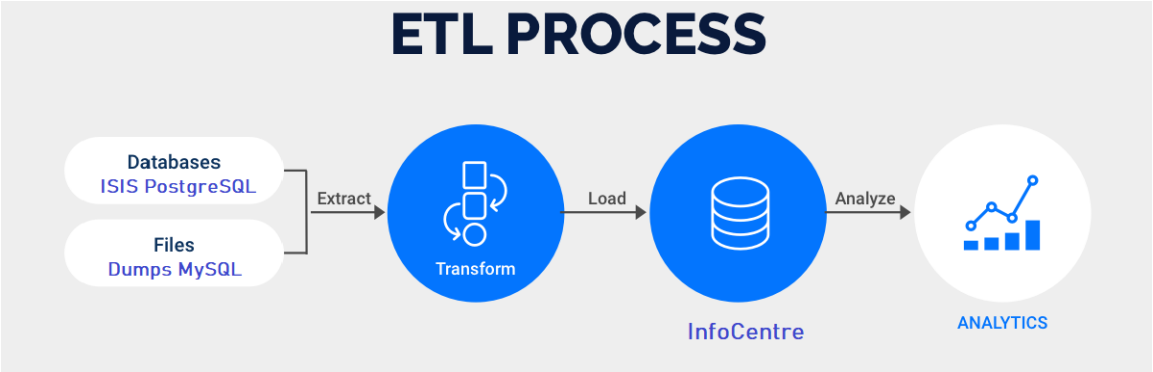

***Figure 1** : ETL processus*

Pour réaliser le processus ci-dessus, nous avons le choix entre deux techniques différentes : utiliser des outils d'ETL (SSIS, Data Factory…) ou utiliser des langues informatiques (Python, R, Java, C#…)

Exakis est un partenaire de Microsoft et ils sont spécialisés dans l'utilisation des outils Microsoft. Pour cette raison nous souhaitions utiliser l'outil d'ETL « Data Factory » de Microsoft pour réaliser ce projet. Cependant, après l'avoir testé, nous nous sommes rendu compte que les outils disponibles étaient extrêmement pratiques, mais qu'ils présentaient également certaines limites. Dans ce projet, les données du client ne sont pas des bases de données ouvertes, nous devons nous connecter via Azure Virtual Network (VNET) pour accéder à la base PostgreSQL et récupérer des dumps pour la base MySQL.

Du fait de ces deux limitations, nous ne pouvons pas utiliser Data Factory, j'ai donc décidé d'utiliser le langage Python pour développer un programme spécifique qui réalise tout ce processus d'ETL.

3. Planning prévisionnel

Le planning prévisionnel du projet est extrêmement important et essentiel, car il garantit le respect des délais prévus, il permet à toutes les personnes suivant le projet de suivre l'avancement des tâches facilement.

Pour ce projet, nous avions prévu 8 semaines du travail (40 jours) avec le planning prévisionnel suivant :

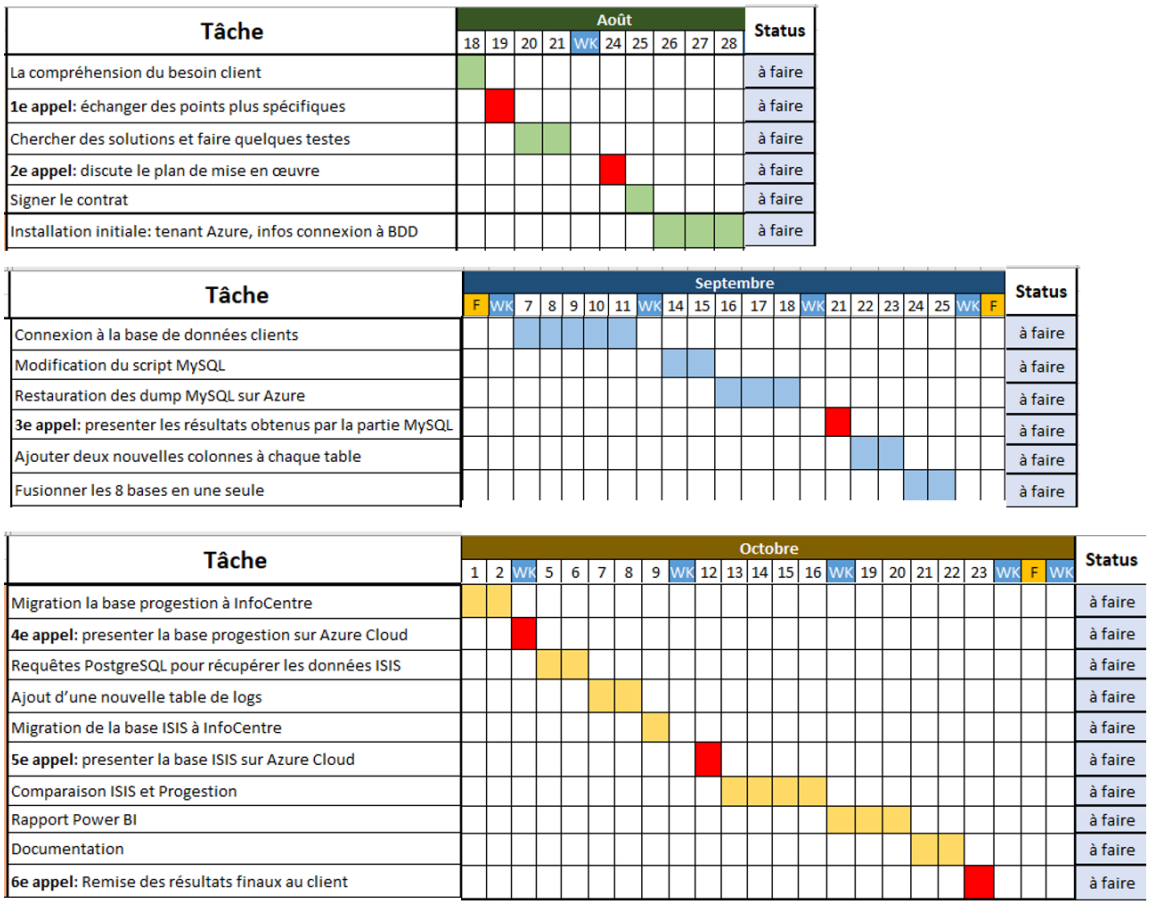

***Figure 2a** : Planning prévisionnel (version lisible)*

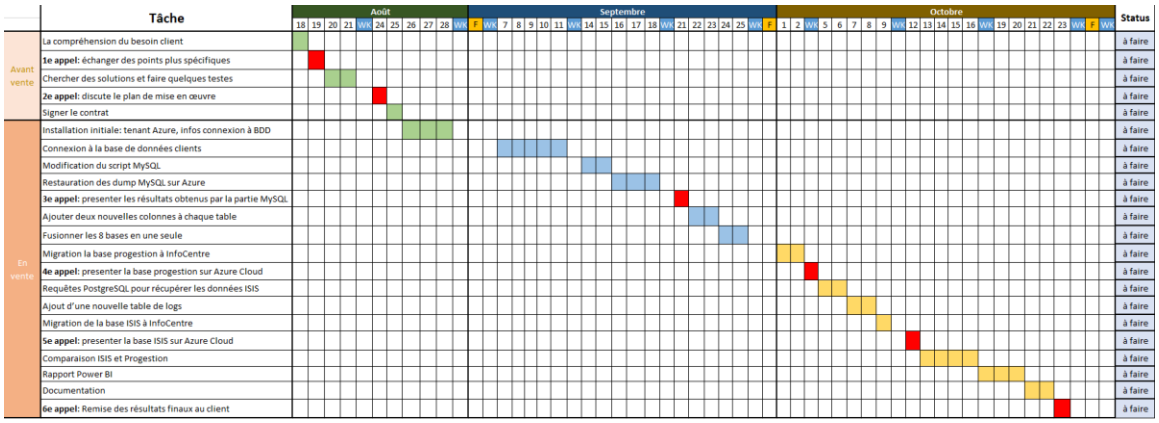

***Figure 2b** : Planning prévisionnel (version global)*

Comme nous pouvons le voir, le planning ci-dessus comporte 3 parties principales : la liste des tâches, la date de début et de fin de chaque tâche et l'état d'exécution de chaque tâche (à faire / en cours / en retard/ terminé).

Les carrés rouges indiquent les réunions entre nous (moi et les managers) et le client, afin de vérifier si le projet est dans les délais ou s'il y a des problèmes qui nécessitent une discussion plus approfondie.

Pendant le projet, il y a aussi des points à adapter, mais grâce à ce planning, nous avons pu garder la bonne trajectoire.

4. Réponse technique mise en œuvre dans le projet

4.1 Les bases de données

Le client possède deux bases de données différentes, correspondant chacune à une application :

- **Progestion** (MySQL) : Application de gestion locative
  - En pratique, les données de cette application sont séparées en 8 bases différentes, correspondant chacune à une région différente. Pour simplifier la comparaison, ces 8 bases devront fusionner en une seule

- Chaque nuit à 01h00, 8 sauvegardes de bases sont déposées sur Cloud Azure Storage avec 8 dumps MySQL
- Chaque dump a un poids de 60 Mo

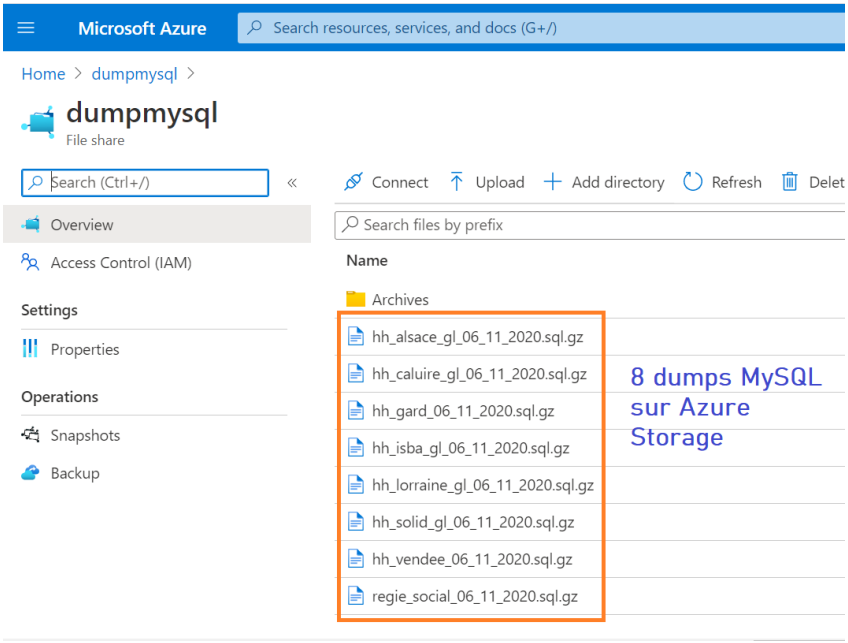

***Figure 3** : 8 dumps MySQL sont déposées sur Cloud Azure Storage tous les jours*

- **ISIS** (PostgreSQL) :
  - Application de suivi des familles, de l'accompagnement et du parc immobilier
  - Cette base de données est utilisée en production, chaque jour il y a de nombreuses personnes qui accèdent à cette base de données pour obtenir des informations, nos comparaisons pouvant être assez longues cela pourrait avoir un impact pour les autres utilisateurs. Pour cette raison, le client souhaite copier cette base de données vers un autre endroit où seul le rapport comparatif utilisera cette base

4.2 Schéma fonctionnel

Ce schéma fonctionnel analyse les besoins du client et résume leurs attentes. Les explications détaillées pour chaque étape viendront plus tard dans cette section.

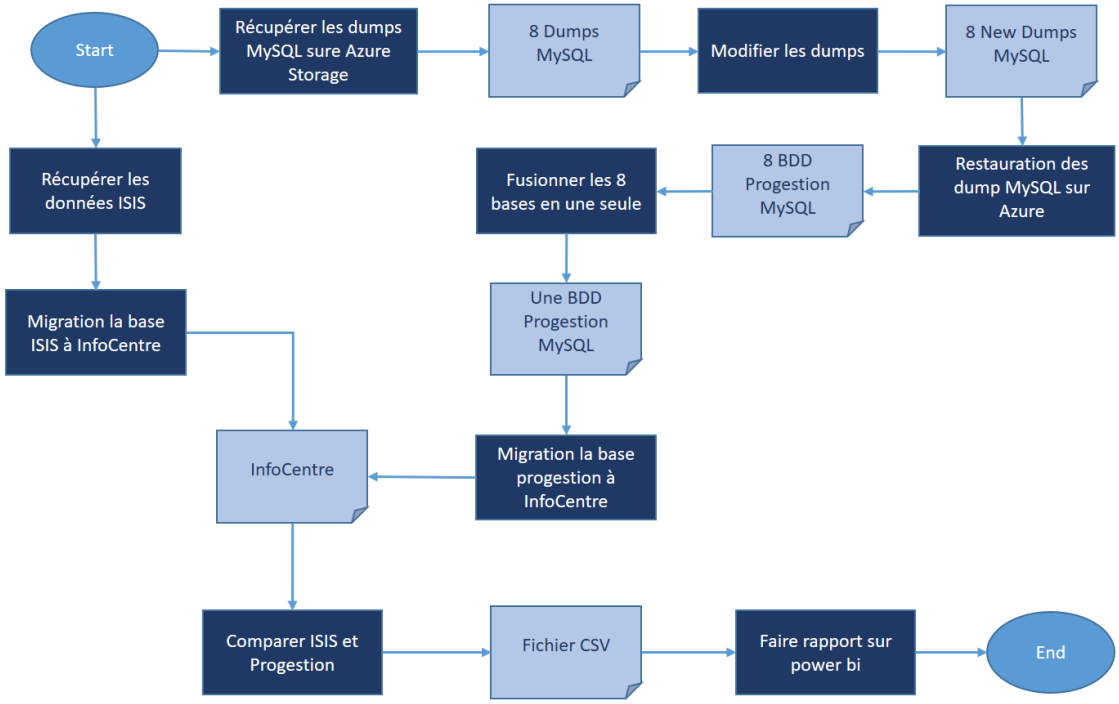

***Figure 4** : Schéma fonctionnel du projet*

Le schéma fonctionnel est une représentation simplifiée de la réalité permettant de décrire le fonctionnement du projet. Dans ce schéma fonctionnel, les formes en bleu foncé sont utilisées pour représenter les éléments fonctionnels, les formes en bleu clair sont utilisées pour représenter les sorties (résultats) de la fonction, les flèches permettent de relier ces éléments entre eux.

A travers ce schéma et ce projet, je voudrais présenter et détailler les deux parties principales de la base de données : le développement d'une base de données et l'exploitation d'une base de données.

4.2.1 Développement d'une base de données

Pour développer la base de données InfoCentre finale, j'ai dû passer par de nombreuses étapes différentes (figure 4) :

- Identifier et se connecter aux sources de données
- Collecter et sauvegarder ces données
- Préparer les données pour l'import en base de données
- Intégrer les données propres et préparées dans la base de données finale
- Optimiser la base de données et améliorer les performances
- Automatiser toutes les étapes ci-dessus pour mettre à jour la base de données chaque jour

Comme mentionné ci-dessus, mon client dispose de deux types de bases de données différents, je dois donc me connecter à ces deux différentes bases pour réaliser toutes les étapes ci-dessus. Je vais désormais expliquer plus en détail le processus pour chaque base.

❖ **Base 1 : Progestion (MySQL)**

➢ ***Restauration des dumps MySQL sur Azure Database for MySQL***

Pour les données de Progestion, je me connecte sur Azure Storage du client pour récupérer les dumps MySQL. J'utilise un script python et l'Azure Storage SDK (Software Development Kit) construit par Microsoft pour me connecter à Azure Cloud où 8 sauvegardes de base sont déposées par le client tous les matins à 1h00.

Après avoir récupéré les 8 dumps MySQL, je sais que la base du client est en MySQL version 5.5. Mais sur Azure MySQL, la version la plus ancienne est 5.6. Donc pour migrer cette base de données vers Azure Cloud, je dois modifier le script MySQL du client pour qu'il corresponde à la version la plus récente, et pour permettre leur migration vers Azure. Par exemple, je dois, entre autres, changer le « MySQL storage engine » de MyISAM à INNODB.

Une fois que j'ai les données propres et préparées pour l'import, je procède à leur intégration dans une base de données sur Azure. J'utilise une librairie Python « mysql.connector », pour établir une connexion (conn) à une base de données MySQL sur Azure via les informations de connexion : host, user, password, port, database. Puis j'utilise la fonction d'exécution « conn.execute () » pour intégrer des données propres et préparées dans la base de données.

- ***Fusionner les 8 bases en une seule***

Une fois l'exécution réussie, j'ai 8 bases de données différentes correspondant à 8 dumps MySQL. Cependant, en regardant le schéma fonctionnel du projet, je dois les combiner dans une seule base de données afin de faciliter les futures comparaisons. Pour ce faire, j'utilise l'opérateur UNION de MySQL. Cet opérateur est utilisé pour combiner les résultats de plusieurs « SELECT». Il supprime les lignes dupliquées (par défaut) entre les différents « SELECT», ce qui permet d'améliorer les performances de la base de données.

Mais avant tout, j'ajoute deux nouvelles colonnes (date_mis_a_jour, base_origine) à chaque table, ainsi lorsque les 8 bases seront fusionnées en une seule, on peut savoir pour chaque ligne quelle est la base d'origine. Cela nous permet également de savoir quand est-ce que la base a été mise à jour pour la dernière fois.

Cependant, j'ai rencontré un problème pour fusionner ces 8 bases. En effet, même si ces 8 bases viennent de la même application, il y a des différences de structures entre les différentes bases de données. Donc lors de l'UNION de toutes les bases en une seule, je dois garder seulement les colonnes communes à chaque base, car chaque « SELECT » dans l'opérateur UNION doit avoir le même nombre de champs et avec des types de données similaires. Si une colonne n'est pas présente dans toutes les bases (pour une table donnée) elle sera donc ignorée lors de la fusion.

Pour garder que les colonnes communes, je dois d'abord lister toutes les colonnes dans chaque table en utilisant la table de MySQL INFORMATION_SCHEMA COLUMNS. Une fois cela fait, j'utilise la fonction « fetchall() » pour stocker toutes les colonnes de chaque table. J'ai

ensuite créé un petit algorithme en Python pour obtenir les colonnes communes à chaque table entre les 8 bases de données.

J'utilise ensuite des requêtes classiques dans MySQL pour créer une base de données, créer des tables, intégrer les données dans une nouvelle base de données tel que : CREATE {DATABASE | SCHEMA}, CREATE TABLE, SELECT, UNION, INSERT INTO...

- ***Migration de la base « Progestion » à InfoCentre (Azure Database for PostgreSQL)***

Avant de déplacer la base « Progestion » vers Azure, je dois d'abord migrer la base de MySQL à PostgreSQL. La partie la plus compliquée et importante de cette étape est de changer les types de données MySQL en mappant correctement avec les types de données PostgreSQL (Figure 5), car MySQL et PostgreSQL ont différents types de données. Certains d'entre eux sont équivalents tandis que d'autres sont totalement différents.

MySQL	PostgreSQL
BINARY(n)	BYTEA
BIT	BOOLEAN
DATETIME	TIMESTAMP
DOUBLE	DOUBLE PRECISION
FLOAT	REAL
MEDIUMINT	INTEGER
TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB	BYTEA
TINYINT	SMALLINT
TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT	TEXT
VARBINARY(n), VARBINARY(max)	BYTEA
VARCHAR(max)	TEXT

***Figure 5** : Table de correspondance des types de données*

Après avoir obtenu des données propres et préparées, je me connecte à la base InfoCentre que j'ai créée sur Azure Cloud puis, j'intègre les données dans cette base en utilisant les fonctions « execute() » et « commit » de la classe « psycopg2 » de Python.

❖ **Base 2 : ISIS (PostgreSQL)**

Pour la base de données ISIS, je me connecte sur la base production via les informations connexions de leur base (host, user, password, port, database) en utilisant « Psycopg2 » qui est l'adaptateur de base de données PostgreSQL le plus populaire pour Python.

Pour des raisons de sécurité, je ne peux normalement pas me connecter directement à leur base de données via une adresse IP publique, je dois me connecter via le réseau virtuel Azure (VNet) qui est le bloc de construction fondamental pour les réseaux privés dans Azure. Le réseau virtuel permet à de nombreux types de ressources Azure, telles que les machines virtuelles (VM) Azure, de communiquer de manière sécurisée entre elles, avec Internet et avec les réseaux locaux. C'est pour cela que je dois me connecter à la base ISIS et exécuter les scripts Python via une VM.

Avec la base de données ISIS, je peux travailler directement sur la base de données, c'est-à-dire que je peux faire des requêtes directement, pas besoin d'exécuter via des dumps comme avec la base Progestion MySQL.

Après avoir utilisé des requêtes « SELECT », j'ai obtenu les données ISIS de production. J'ai continué avec des requêtes « CREATE » pour créer le deuxième schéma dans InfoCentre et j'ai ensuite créé les tables nécessaires pour ce schéma.

J'ajoute ensuite d'une nouvelle table de logs avec 3 champs (log_id, start_time, end_time) qui sera remplit à chaque migration. Avec cette table, le client peut savoir quand et en combien de temps les données ont été mises à jour.

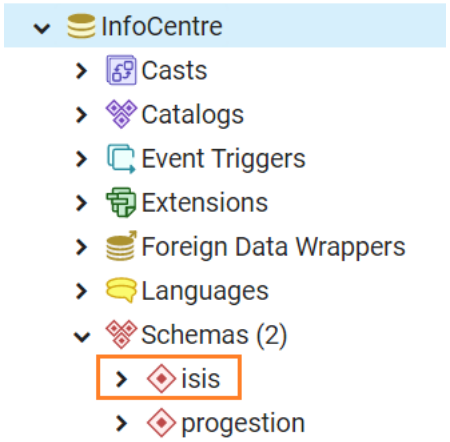

***Figure 6** : La base InfoCentre avec les deux schémas isis et progestion*

4.2.2 Automatisation

Cette automatisation a été faite avec le développement de scripts Python.

Tous les Scripts Python sont planifiés et stockés dans une VM windows 10 sur Azure Cloud via l'application « Task Scheduler ».

J'ai réalisé plusieurs scripts Python courts sur la partie de la base « Progestion » afin de permettre plus facilement leur automatisation. Pour la partie migration de la base ISIS, j'ai fait un seul script Python, car il est moins compliqué et il y a moins d'étapes à faire.

Nom de script Python	Description	start_time	end_time
migrate_mysql_separately.py	Restauration des dumps MySQL sur Azure Database for MySQL séparément dans des bases temporaires	1:15	2:15
union_mysql.py	Fusion des 8 bases en une seule	2:30	2:40
drop_schema_temps.py	Suppression des bases de données temporaires	2:45	2:55
migrate_mysql_to_postgresql.py	Migration de progestion à InfoCentre	3:00	5:00
moving_dumps_to_archives.py	Déplacement des dumps dans le dossier Archives	23:30	23:32

***Figure 7** : Description des scripts Python servant à migrer automatiquement la base de données Progestion*

Nom de script Python	Description	start_time	end_time
migrate_ISIS.py	Migration de la base ISIS à InfoCentre	23 :00	23 :58

***Figure 8** : Description du script Python servant à migrer automatiquement la base de données ISIS*

4.2.3 Exploitation d'une base de données

Une fois la base InfoCentre créée et remplie, je peux récupérer les données des deux bases et appliquer un traitement aux données sélectionnées pour comparer les deux bases, ISIS et Progestion. J'ai ensuite produit un rapport à l'aide de l'outil Power BI pour montrer les différences entre les deux bases.

Lorsque tout est fait, je publie mes résultats sur le service Power BI et je les partage avec les personnes concernées via leur adresse e-mail.

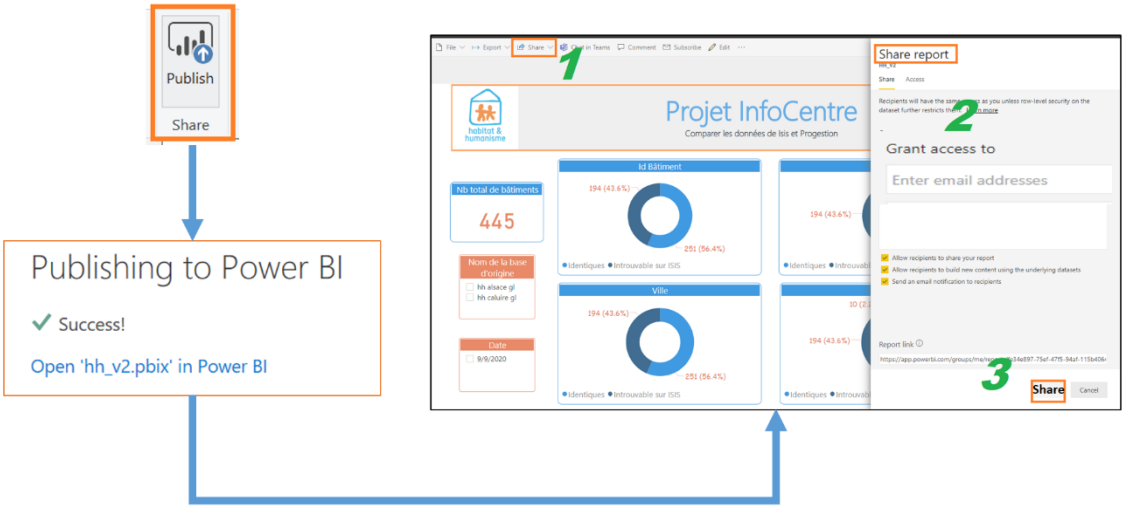

***Figure 9** : Processus pour partager les résultats avec d'autres personnes*

Dans la partie suivante, je vais entrer dans une explication détaillée d'une comparaison sur la cohérence des immeubles/bâtiments entre ISIS et PROGESTION.

Pour l'ensemble des bâtiments et par AIVS, je dois faire des requêtes pour récupérer les infos suivantes :

- Le nombre de bâtiments de Progestion qui portent un identifiant ISIS et qui correspondent bien à l'adresse principale indiquée dans ISIS pour cet immeuble (voir détails ci-dessous)
- Le nombre de bâtiments de Progestion qui portent un identifiant ISIS mais qui ne correspondent pas à l'adresse indiquée dans ISIS pour cet immeuble (voir détails ci-dessous)
- Le nombre de bâtiments Progestion ne portant pas d'identifiant ISIS (voir détails ci-dessous)

Les tables impactées sont les suivantes :

Dans Progestion :

- Table « batiment »
  - L'identifiant ISIS est stocké dans la colonne « ref_externe » et est construit de la manière suivante : «I_» concaténé avec la valeur de l'identifiant ISIS « idimmeuble »
  - Par exemple, un immeuble ISIS dont « idimmeuble » = 4322 sera identifié par le le code « I_4322 » dans la colonne « ref_externe » de Progestion
  - Champs relatifs à l'adresse : « batiment.adresse / batiment.cp / batiment.ville »

Dans ISIS :

- Table « immeuble » : clé primaire est « idimmeuble »
- Table « adresse_immeuble » :
  - La clé étrangère pointant sur la table immeuble est « idimmeuble »

    - On ne prend que l'adresse principale de l'immeuble avec comme condition de jointure : adresse_immeuble.blprincipale IS TRUE
- Table « adresse »
    - La clé étrangère pointant sur la table « adresse_immeuble » est idadresse
    - Le nom de la rue stocké dans le champ « adresse.lbrue »
- Table « admin_commune »
    - La clé étrangère pointant sur la table adresse : « idcommune »
    - Code postal stocké dans « admin_commune.cdpostal »
    - Ville stockée dans « admin_commune.lbcommune »

En fonction de ces champs, je dois vérifier que pour les bâtiments « Progestion » comportant un identifiant ISIS, les données de localisation sont cohérentes avec ISIS, à savoir que « adresse / code postal / ville du bâtiment Progestion » correspondent à « lbrue / cdpostal / lbcommune » du même immeuble dans ISIS.

Pour obtenir les informations décrites ci-dessus, j'ai fait des requêtes PostgresSQL (SELECT, JOIN…) sur les schémas « progestion » et « isis » dans la base InfoCentre, puis j'ai sauvegardé le résultat dans le fichier CSV.

id_batiment	ref_externe	batiment.adresse	batiment.cp	batiment.ville
50	I_1338	38 RUE VANDENBERGHE	59000	LILLE
57	I_4376	6 RUE DE LA TOISON DOR	59650	VILLENEUVE-DASCQ
61	I_1024	33 avenue Regnier	3310	NERIS-LES-BAINS
62	I_1023	2 rue Corneille	3300	CUSSET
63	I_1022	13 rue des Darcins	3300	CUSSET
64	I_1021	56 boulevard Du Sichon	3200	VICHY
65	I_1019	9 rue de la Gare	3200	VICHY
66	I_1020	4 place Lasteyras	3200	VICHY

***Figure 10** : Aperçu du fichier CSV avec résultat de requête pour le schéma « progestion »*

idimmeuble	adresse.lbrue	cdpostal	commune
4369	3 BIS avenue Pasteur	63400	CHAMALIERES
4376	6 rue de la Toison dOr	59650	VILLENEUVE-DASCQ
1217	5254 rue Papu	35000	RENNES
4400	impasse de la Campanella	13770	VENELLES
4429	15 avenue de la RÃ©publique	63540	ROMAGNAT
4430	8 rue Rembrandt	42100	SAINT-ETIENNE
4432	16 boulevard Lyautey	49000	ANGERS
4436	135 rue Didot	75014	PARIS
4441	58 rue Jean Paul Sartre	73000	CHAMBERY
4437	16 rue Chauvelot	75015	PARIS
4453	11 rue du Port	22000	SAINT-BRIEUC

***Figure 11** : Aperçu du fichier CSV avec résultat de requête pour le schéma « isis »*

Avec les figures 10 et 11, nous pouvons voir la liaison des deux résultats via la colonne « ref_externe ». J'utilise la librairie Pandas pour lire les deux fichiers CSV. Puis, je boucle sur chaque ligne des deux tables pour permettre leur comparaison. L'enregistrement du résultat est fait dans un fichier CSV et stocké dans un Blob Storage sur Azure Cloud.

id batiment pg	id_ref	id_immeuble isis	id immeuble?	adresse pg	adresse isis	adresse?	.....	region	date
150	Ne pas avoir	Introuvable sur ISIS	Introuvable sur ISIS	34 rue de la Tour	Introuvable sur ISIS	Introuvable sur ISIS	.....	hh alsace gl	9/9/2020
151	Ne pas avoir	Introuvable sur ISIS	Introuvable sur ISIS	16 Avenue du Rhin	Introuvable sur ISIS	Introuvable sur ISIS	.....	hh alsace gl	9/9/2020
153	Ne pas avoir	Introuvable sur ISIS	Introuvable sur ISIS	121A ROUTE DOBERHAUSBERGEN	Introuvable sur ISIS	Introuvable sur ISIS	.....	hh alsace gl	9/9/2020
50	I_1338	1338	Identiques	38 RUE VANDENBERGHE	38 rue vandenberghe	Identiques	.....	hh caluire gl	9/9/2020
57	I_4376	4376	Identiques	6 RUE DE LA TOISON DOR	6 rue de la toison dor	Identiques	.....	hh caluire gl	9/9/2020

***Figure 12** : Aperçu du fichier CSV de comparaison*

Ensuite, je me connecte au fichier CSV de comparaison (figure 12) présent sur Blob Azure qui servira d'interface pour communiquer avec Power BI afin de créer le rapport qui inclut 3 onglets : 1 global et deux vues détaillées.

5. Bilan du projet et améliorations

Concernant le bilan de ce projet, il y a deux résultats principaux : la base de données InfoCentre et le rapport montrant les différences entre ISIS et Progestion.

InfoCentre est accessible via l'outil « pgAdmin », qui est la plate-forme d'administration et de développement Open Source la plus populaire et la plus riche en fonctionnalités pour PostgreSQL. Le client peut donc accéder directement à la base de données si nécessaire.

Pour le rapport Power BI, les personnes avec qui je l'ai partagées peuvent y accéder, mais avec différents rôles. Ces rôles ont été paramétré au

moment du partage du rapport avec eux. Ces rôles sont : « owner » ou « read and reshare ».

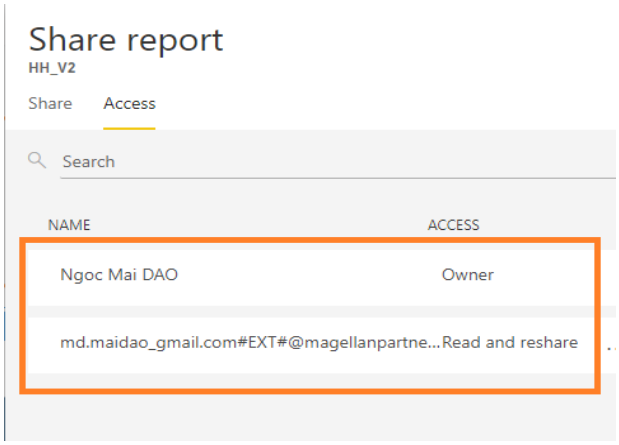

***Figure 13** : Différents rôles pouvant accéder au rapport sur Power BI*

Comme mentionné ci-dessus, le rapport se compose de 3 pages : Global et deux vues détaillées.

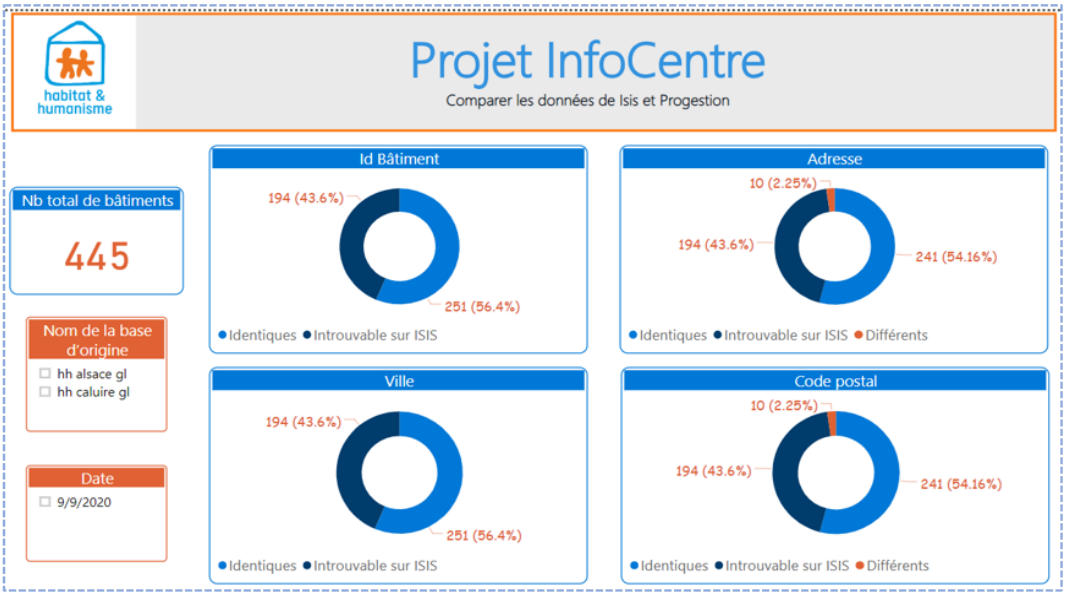

***Figure 14** : Première page du rapport (Global)*

Comme son nom l'indique, cette page est destinée à présenter les informations les plus élémentaires et générales sur la comparaison. Comme le montre la figure 14, nous pouvons voir les informations suivantes :

- Le nombre total de bâtiment dans la base de données de Progestion est 445
- 56.4% des Id bâtiments ont été trouvées dans ISIS
- 54.16% adresse sont identiques, 43.6% introuvable et 2.25% différentes
- 56.4% villes introuvables, les 43.6% restants coïncident tous avec ISIS
- 43.6 % des codes postaux sont sur ISIS mais sur ces 43.6 %, il y en a 2.25% qui sont différents. Le reste est introuvable

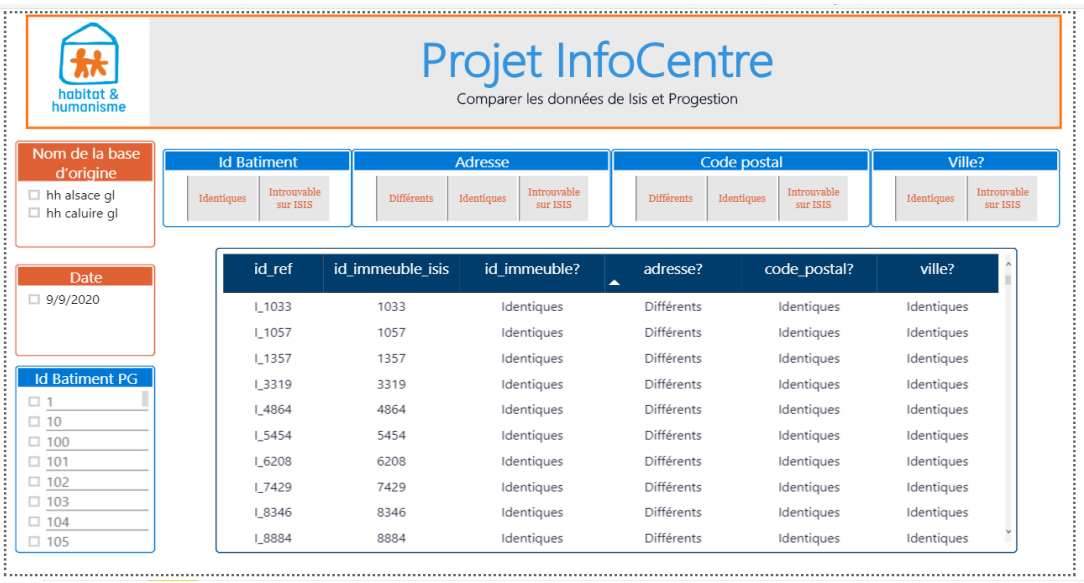

***Figure 15** : Deuxième page du rapport (Détail 1)*

La figure 15 montre des informations plus détaillées sur la comparaison. Pour chaque immeuble, on peut savoir si l'adresse, la ville ou le code postal correspondant à l'immeuble est identique ou différent ou introuvable sur ISIS.

Outre la page « détail 1 », j'ai également réalisé une page « détail 2 » qui est plus détaillée que la première. Avec cette page, nous pouvons voir pourquoi il y a des différences si une différence est trouvée.

Par exemple, dans la figure 16 ci-dessous, pour le bâtiment 170, nous voyons que l'adresse du bâtiment n'est pas identique entre Progestion et ISIS.

Mais si nous ne savons pas ce qu'il y a de différent, alors que dans la figure 17, nous pouvons voir la différence. L'adresse stockée dans ISIS est « 30 rue rabutin » et celle stockée dans Progestion est « 30 rue Rabutin Chantal ».

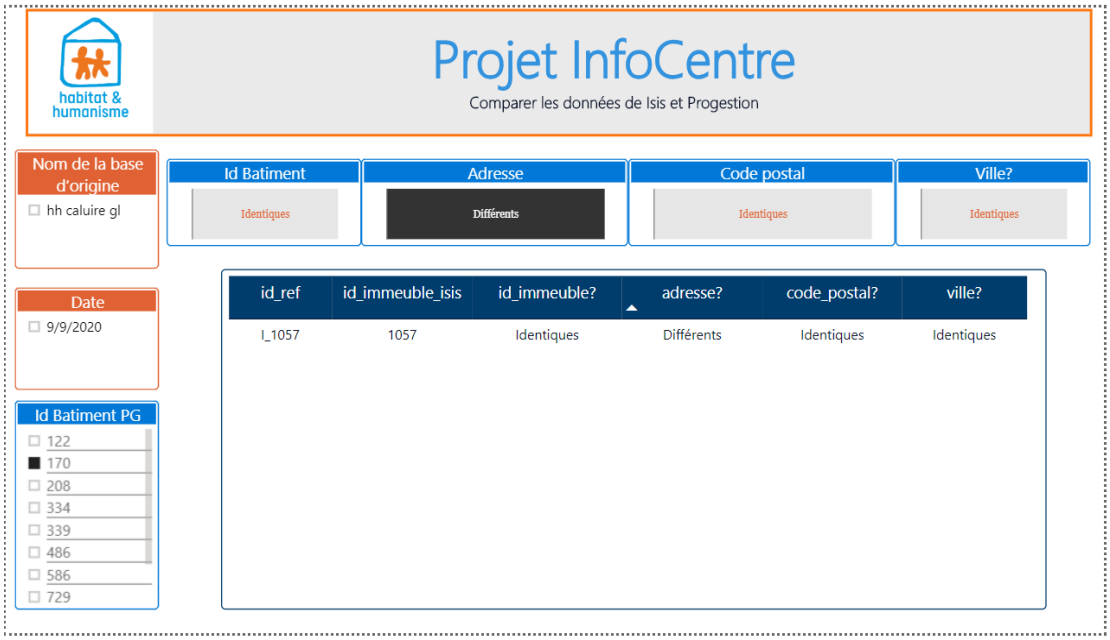

Figure 16 *: Deuxième page du rapport avec le bâtiment 170 qui a une différence sur l'adresse entre ISIS et Progestion*

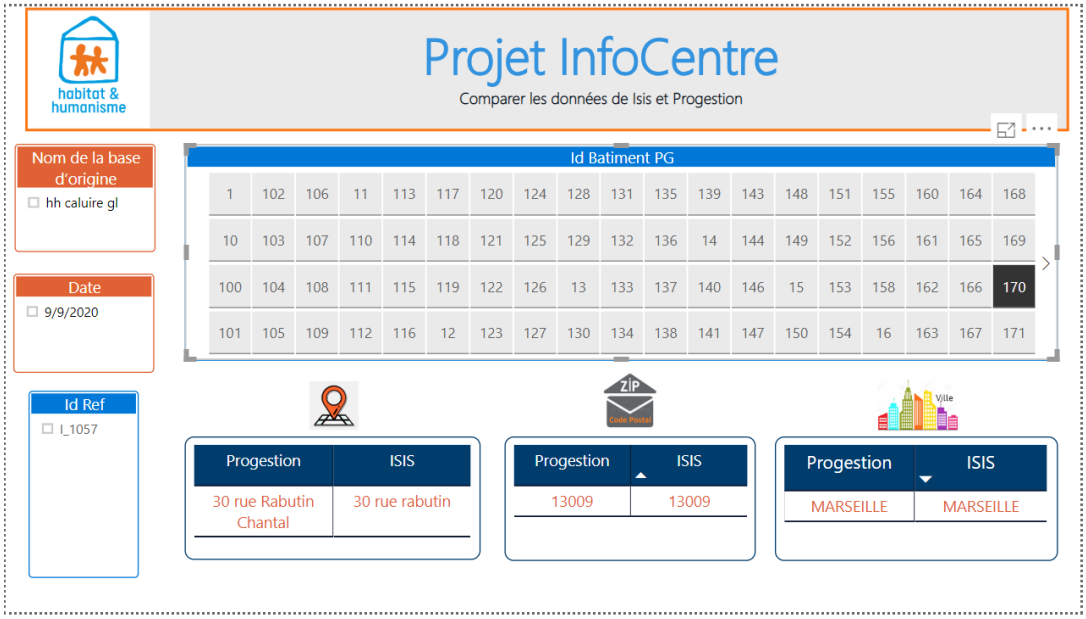

Figure 17 *: La troisième page du rapport (Détail 2)*

CONCLUSION

Nous avons de nombreuses manières et techniques différentes pour réaliser un projet de base de données en général ou un projet ETL en particulier.

Par exemple, pour ce projet, j'utilise deux fichiers csv temporaires qui vont me permettre de générer le fichier csv final de comparaison (qui sera donné en input à Power BI). Il est également possible de créer ce fichier de comparaison via SQL directement depuis les bases de données.

En général, les outils disponibles sont très utiles et pratiques, mais présentent des limites et ne peuvent pas être utilisés en toutes circonstances. Ce projet est l'un de ces cas. Par conséquent, en plus de savoir utiliser les outils, nous devons également connaître « le code » pour pouvoir résoudre des problèmes particuliers ou s'adapter enivrement spécifique.