UNIVERSITÉ DE SHERBROOKE

Faculté des Sciences

Département d'Informatique

IFT723 – Sujets Approfondis en Base de Données

RAPPORT DU PROJET DE SESSION :

Gestion des Paiements des Droits de Scolarité

Enseignant :

Luc Lavoie

Auteurs :

Othmane El Biyaali, elbo1901@usherbrooke.ca

Maxime Sourceaux, soum3004@usherbrooke.ca

Alseny Toumany Soumah, soua2604@usherbrooke.ca

Mohamed Boubacar Boureima, boum3688@usherbrooke.ca

Table des Matières

Avant-projet

Ce projet de Gestion des Étudiants et des Paiements des Droits de Scolarité est né principalement de la demande du Professeur **LUC LAVOIE** et **nous (membres du groupe)** dans le but de travailler sur un projet de session. Notre vision est de mettre en place un système de gestion avancé qui améliore le processus de collecte des paiements des étudiants, l'efficacité administrative, et de garantir la sécurité des données des étudiants.

Pour atteindre ces objectifs, nous avons identifié un ensemble de visions clés, chacune visant à répondre à des besoins et exigences spécifiques.

Introduction

Le premier jalon de ce projet marque le point de départ de notre parcours vers la création d'une Base de Données et de son API. Il s'agit d'une étape cruciale où nous adaptons, mettons en œuvre et, le cas échéant, migrons notre base de données initiale. Cette phase initiale requiert une compréhension approfondie des exigences de notre projet.

Le deuxième jalon a pour objectif principal de garantir que notre base de données puisse répondre de manière optimale à la 6eme forme normale (6FN). Pour ce faire, nous avons analysé et modélisé attentivement nos nouvelles exigences, effectué les adaptations nécessaires au schéma de données, et mettre en œuvre ces modifications du modèle de la 5FN à la 6FN pour la conception de la BDT.

Le troisième jalon de notre projet consiste en la modélisation, la conception et la mise à l'essai de l'API de la BDT. Ce cycle de travail bâti sur l'élaboration d'une base de données bi temporalisée (BDT), en mettant à profit les compétences confirmées et développées au cours de l'activité IFT723. Ce jalon vise à concrétiser notre vision d'une base de données temporalisée, suivie d'une API pour répondre aux exigences spécifiques de notre projet. La mise à l'essai de l'API garantira sa fonctionnalité, sa stabilité et son adaptabilité dans des scénarios d'utilisation variés.

De plus, ces jalons impliquent la rédaction de deux documents essentiels : la spécification des exigences du modèle (SEM) et la spécification de conception de la base de données (SCBD). La SEM documentera de manière détaillée les besoins du modèle de données, tandis que la SCBD décrira la structure conceptuelle de la base de données, y compris les schémas, les contraintes, et les détails techniques.

La réussite de ces jalons repose non seulement sur notre capacité à mettre en œuvre la modélisation et conception de la BDT, mais également sur notre capacité à collaborer efficacement en tant qu'équipe. La cohérence et la qualité de notre travail dépendent de notre communication et de notre coordination. Nous avions travaillé ensemble pour atteindre nos objectifs dans les délais impartis.

En fin de compte, ces jalons posent les bases solides sur lesquelles nous allons construire notre BDT. Il est essentiel de l'aborder avec rigueur, précision et une vision claire de l'objectif final du projet.

Étude d'Opportunités

La mise en place d'un système de gestion avancé pour améliorer le processus de collecte des paiements des étudiants pourrait revêtir un intérêt significatif pour diverses raisons. Tout d'abord, l'optimisation de ce processus pourrait entraîner une réduction des erreurs de paiement, une augmentation de l'efficacité administrative et une amélioration de la satisfaction des étudiants en facilitant et en accélérant les transactions financières. En outre, un tel système pourrait offrir une plus grande transparence dans le suivi des paiements, permettant une gestion plus précise et efficiente des finances de l'établissement éducatif. En termes de nécessité, si l'actuel système présente des lacunes en termes de sécurité des données étudiantes ou s'il est obsolète, la mise en place d'un système avancé pourrait être vitale pour garantir la confidentialité et la protection des informations sensibles des étudiants. En somme, l'introduction d'un système de gestion avancé pourrait offrir des avantages tangibles tels qu'une meilleure efficacité et des économies de temps, ainsi que des avantages intangibles tels qu'une image institutionnelle renforcée et une meilleure satisfaction des étudiants, justifiant ainsi l'intérêt voire la nécessité de lancer ce projet.

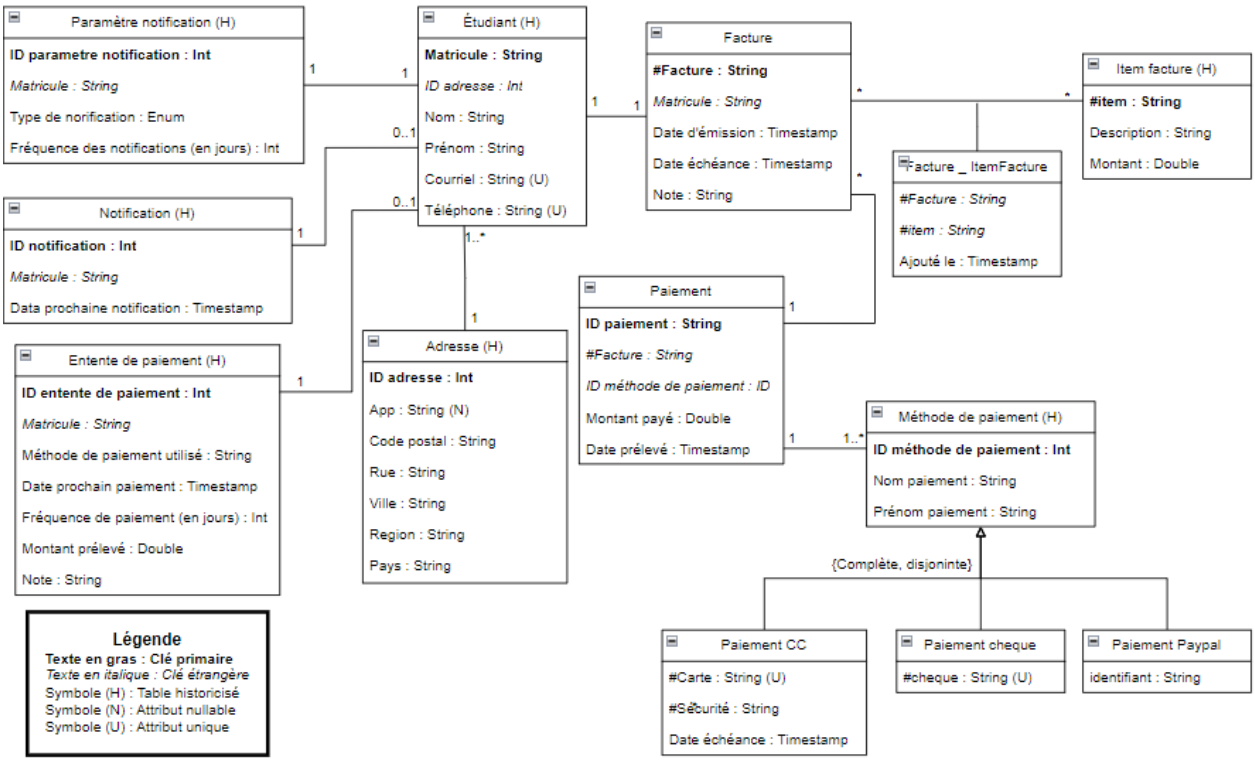
Étude de Faisabilité

L'analyse de d'opportunité de ce projet de gestion de paiement pour les étudiants à l'université implique plusieurs aspects clés. Au **niveau technique**, il est essentiel d'évaluer la disponibilité des compétences requises pour développer une plateforme robuste, sécurisée et adaptable, notamment des experts en technologies de paiement en ligne et en sécurité informatique. Du **point de vue financier**, une estimation minutieuse des coûts de développement, d'implémentation, de maintenance et de formation du personnel est nécessaire pour garantir la viabilité budgétaire du projet sur le long terme. Sur le **plan opérationnel**, l'intégration sans heurts de ce nouveau système dans les processus existants est cruciale pour éviter les perturbations dans les opérations universitaires ; cela implique une analyse détaillée des flux de travail actuels et une planification attentive de la transition. Quant aux **risques potentiels**, la résistance au changement parmi les étudiants ou le personnel nécessite une stratégie de communication et de formation solide pour promouvoir les avantages du système. En parallèle, des tests exhaustifs et des plans de sauvegarde doivent être mis en place pour anticiper et atténuer les problèmes techniques tels que l'accessibilité, la compatibilité et les performances lors du déploiement de la plateforme. Une analyse rigoureuse de ces éléments est cruciale pour assurer le succès de cette initiative et minimiser les risques potentiels.

Jalon 1 : Conception de la BD et de son API

Conception de la base de données en 5FN :

La conception de la base de données (BD) et de son interface programmatique (API) constitue une étape vitale dans le processus de développement d'une application. En amont de cette étape, il est impératif d'identifier les entités clés du système, de définir les attributs associés à chaque entité, et d'établir les relations entre elles en utilisant des clés étrangères pour assurer la cohérence des données. La normalisation de la base de données et l'application de contraintes d'intégrité contribuent à optimiser la structure en garantissant une gestion efficace et performante des données. Simultanément, la conception de l'API nécessite une réflexion approfondie sur les opérations à exécuter, les points de terminaison correspondants, le format des données échangées, et les mécanismes de gestion des erreurs. En fournissant une documentation exhaustive, cette étape assure une compréhension claire et une utilisation fluide de la base de données et de son API tout au long du cycle de vie de l'application.



1. Identification des Entités :

Pour le projet de gestion de paie des étudiants universitaires, les entités principales sont "Etudiant", "Facture", "Adresse", "Notification", "ParametreNotification", "EntentePaiement", "ItemFacture", "Paiement", "MethodePaiement", "PaiementCheque" et "PaiementPaypal".

2. Relations entre les Entités :

- Etudiant - Facture :

Relation "Un-à-Plusieurs" : Un étudiant peut avoir plusieurs factures, mais chaque facture est associée à un seul étudiant.

- Etudiant - Adresse :

Relation "Un-à-Un" : Chaque étudiant a une adresse associée.

- Etudiant - Notification :

Relation "Un-à-Plusieurs" : Un étudiant peut recevoir plusieurs notifications, mais chaque notification est associée à un seul étudiant.

- Etudiant - ParametreNotification :

Relation "Un-à-Un" : Chaque étudiant peut avoir un ensemble de paramètres de notification.

- Etudiant - EntentePaiement :

Relation "Un-à-Plusieurs" : Un étudiant peut avoir plusieurs ententes de paiement, mais chaque entente de paiement est associée à un seul étudiant.

- Facture - ItemFacture :

Relation "Un-à-Plusieurs" : Une facture peut avoir plusieurs items, mais chaque item de facture est associé à une seule facture.

- Facture - Paiement :

Relation "Un-à-Plusieurs" : Une facture peut être associée à plusieurs paiements, mais chaque paiement est lié à une seule facture.

- Paiement - MethodePaiement :

Relation "Un-à-Un" avec héritage : Un paiement peut être associé à une méthode de paiement spécifique, telle que "PaiementCheque" ou "PaiementPaypal". Ceci peut être implémenté à l'aide d'une structure d'héritage où "MethodePaiement" est la classe parente et les autres sont les classes filles.

3. Normalisation de la BD :

Dans le cadre de notre projet de gestion de la paie des étudiants universitaires, nous avons choisi de normaliser nos tables en 5FN (Cinquième Forme Normale) afin d'atteindre une structure de base de données optimale, garantissant l'efficacité, la cohérence des données, et la flexibilité nécessaire pour répondre aux besoins spécifiques du système.

- Dépendances de Jointure :

Nous avons minutieusement analysé les dépendances de jointure au sein de nos tables. Cette étape a impliqué l'identification de situations où des informations pouvaient être dérivées de la combinaison de colonnes provenant de différentes tables.

- Décomposition des Tables :

Lorsque des dépendances de jointure ont été identifiées, nous avons procédé à la décomposition appropriée de nos tables. Cela signifie que nous avons créé des relations indépendantes pour éliminer ces dépendances, permettant une gestion plus claire et précise des données.

Raisons du Choix de la 5FN :

- Complexité des Relations :

En raison de la complexité des relations entre les différentes entités dans notre projet, la 5FN s'est avérée être une option pertinente. Elle nous a permis de gérer de manière plus granulaire les dépendances subtiles entre les différentes composantes du système, évitant ainsi la redondance d'informations.

- Flexibilité et Évolutivité :

La normalisation en 5FN offre une grande flexibilité, ce qui est crucial dans un environnement où les besoins peuvent évoluer avec le temps. Cette approche nous permet d'ajuster plus facilement notre schéma de base de données sans compromettre l'intégrité des données.

- Performance :

Bien que la performance ne soit pas toujours le critère principal dans toutes les situations, la conception en 5FN peut contribuer à une meilleure optimisation des requêtes, en particulier lorsque des opérations de jointure complexes sont nécessaires.

4. Définition des domaines :

En écrivant le code du scripts SQL de notre base de données, on a défini les domaines comme type pour les attributs de nos entités, ces domaines présente des avantages significatifs en termes de consistance des données, réutilisation du code et facilité de maintenance. En définissant des règles spécifiques aux types de données, les domaines garantissent la cohérence des informations tout en facilitant la compréhension du schéma grâce à une centralisation des contraintes. Cette approche permet une évolutivité aisée du modèle de données, simplifiant la gestion des politiques de données et assurant l'intégrité référentielle.

Conception de l'Interface Programmatique (API) :

L'API de notre base de données inclut des procédures stockées pour la génération, la modification, l'insertion et la suppression d'enregistrements pour ces entités, ainsi que des fonctions pour récupérer des informations spécifiques. L'utilisation de procédures stockées et de fonctions offre une encapsulation des opérations liées aux entités, favorisant la modularité et la gestion simplifiée.

Le système ÉMIR (Évaluation, Modification, Insertion, Retrait) et le système CRUD (Create, Retrieve, Update, Delete) sont deux approches distinctes pour dénommer des opérations dans le contexte d'une base de données. Les règles de dénomination dans le système ÉMIR suivent des conventions spécifiques, distinguant les opérations par des suffixes tels que "_eva" pour l'évaluation, "_mod" pour la modification, "_ins" pour l'insertion, et "_ret" pour le retrait. De plus, des suffixes supplémentaires comme "_gen" (général), "_sst" (sans statut), et "_exs" (exigences strictes) sont utilisés pour préciser les caractéristiques des opérations.

Jalon 2 : Modélisation et conception de la BDT

Modélisation de la base de données temporalisé :

Table : Étudiant

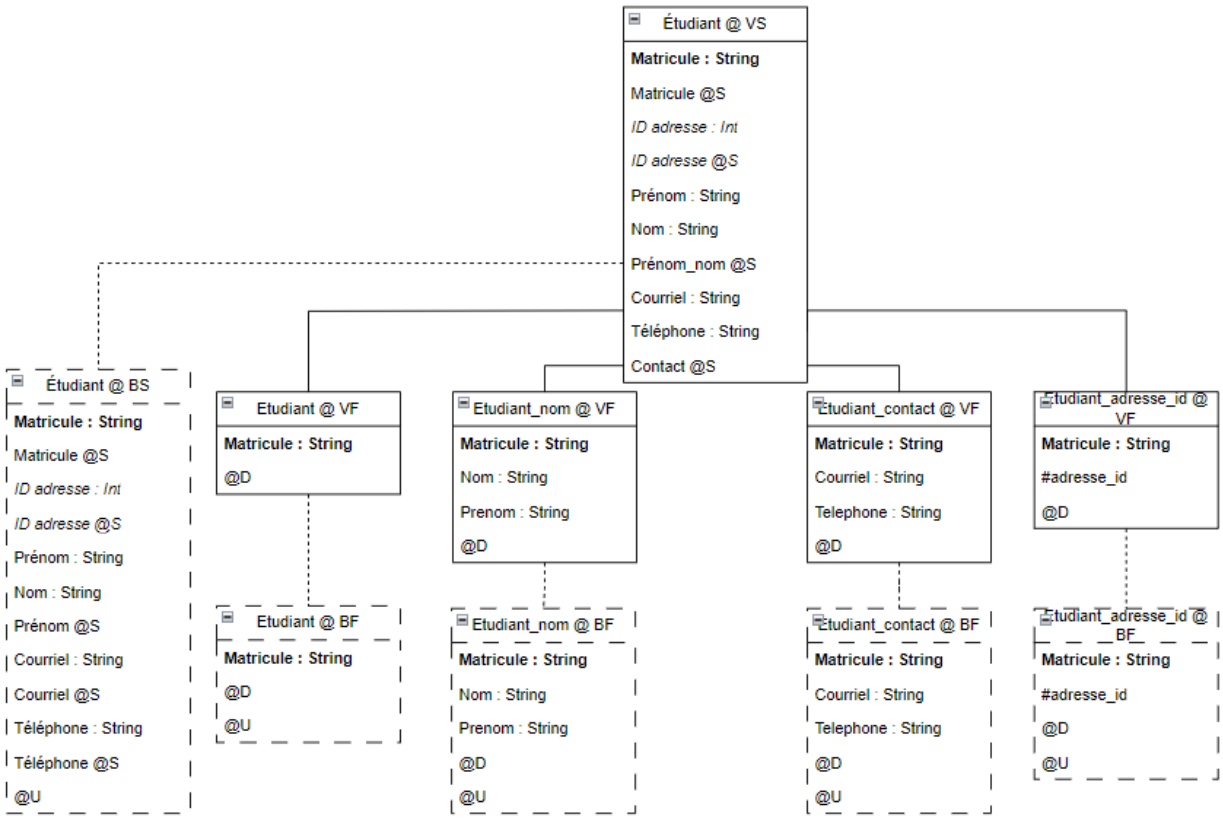


Table : Adresse

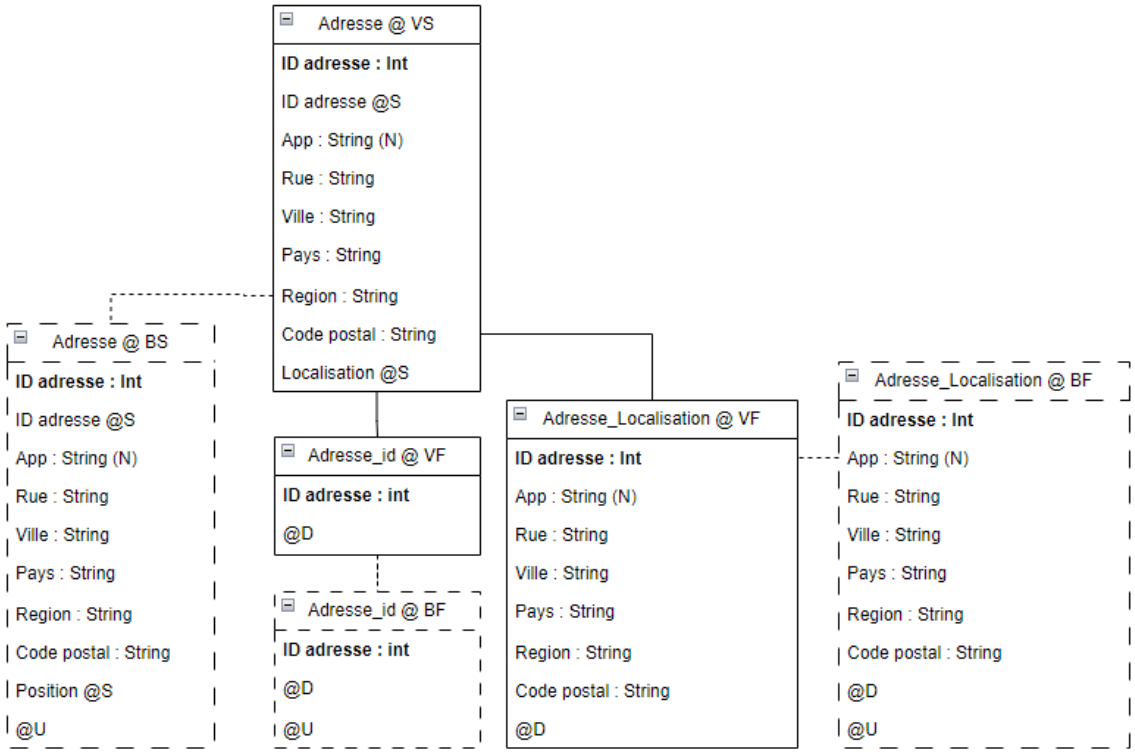


Table : Paramètre notification

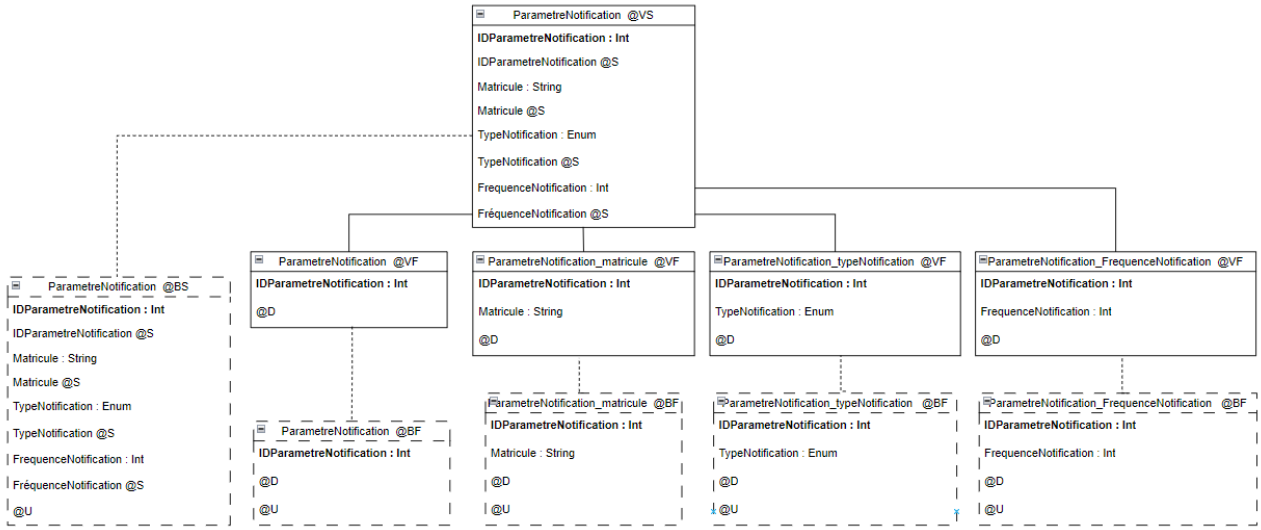


Table : Notification

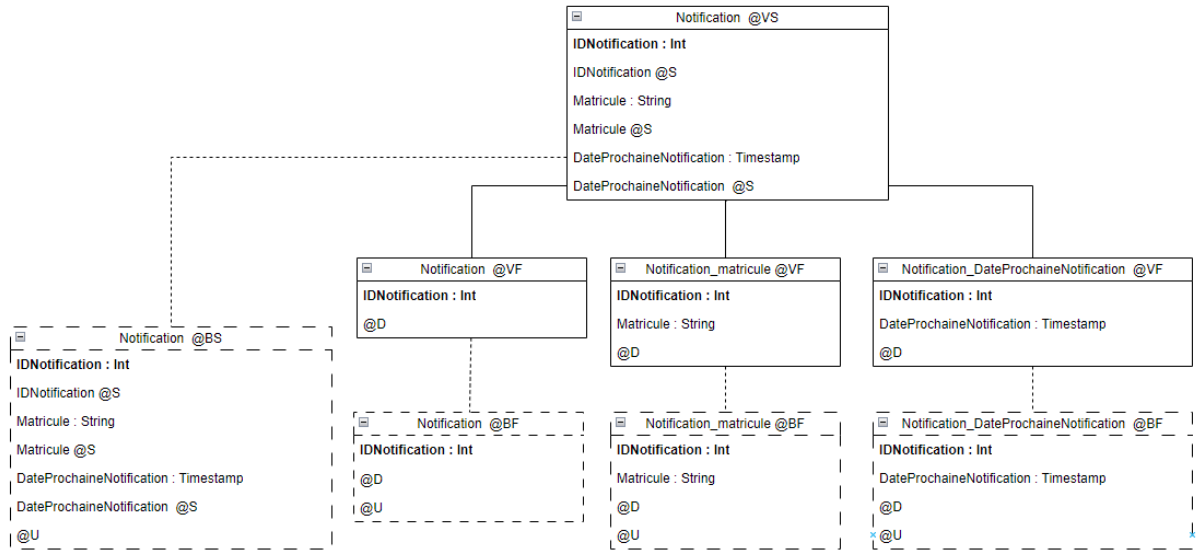


Table : Entente de paiement

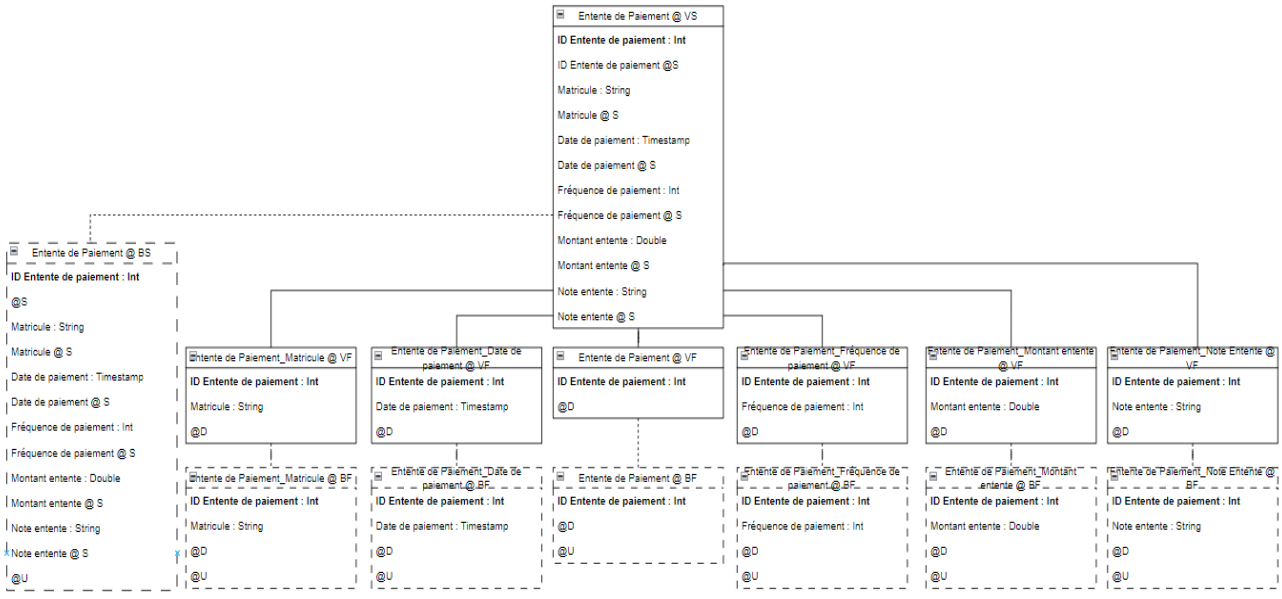


Table : Item Facture

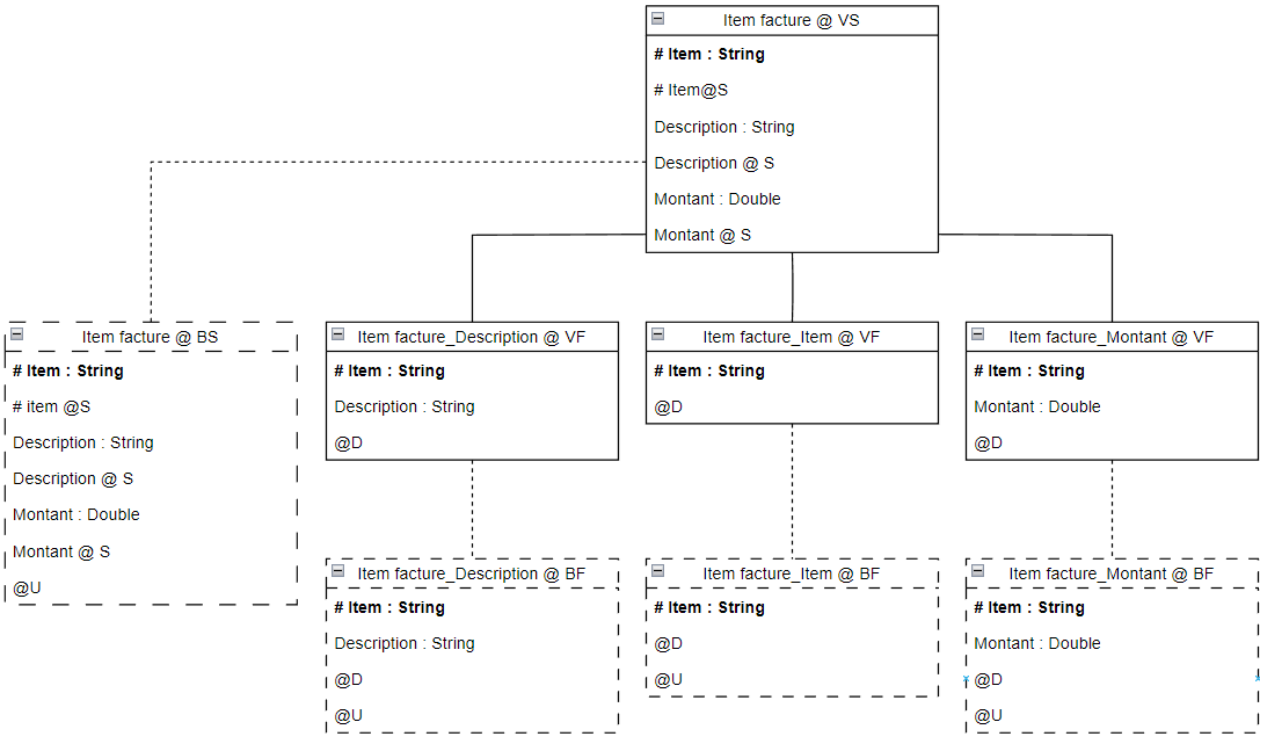


Table : Méthode de paiement CC

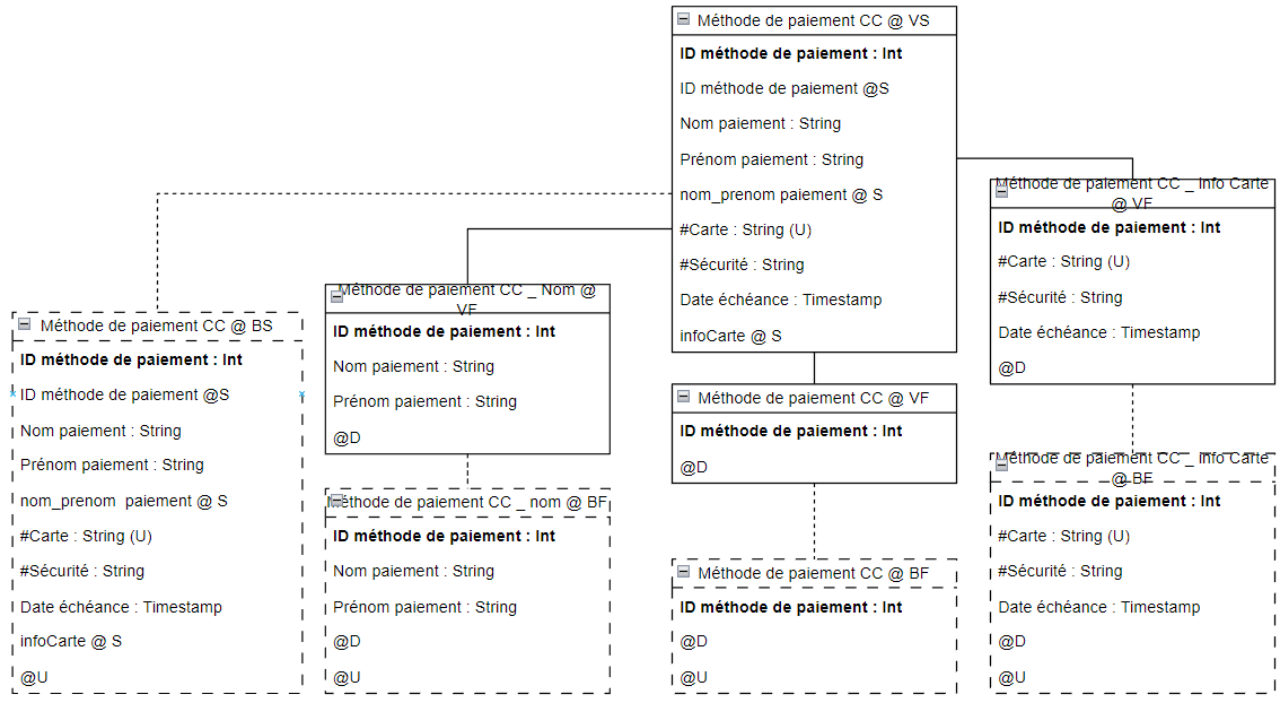


Table : Méthode de paiement cheque

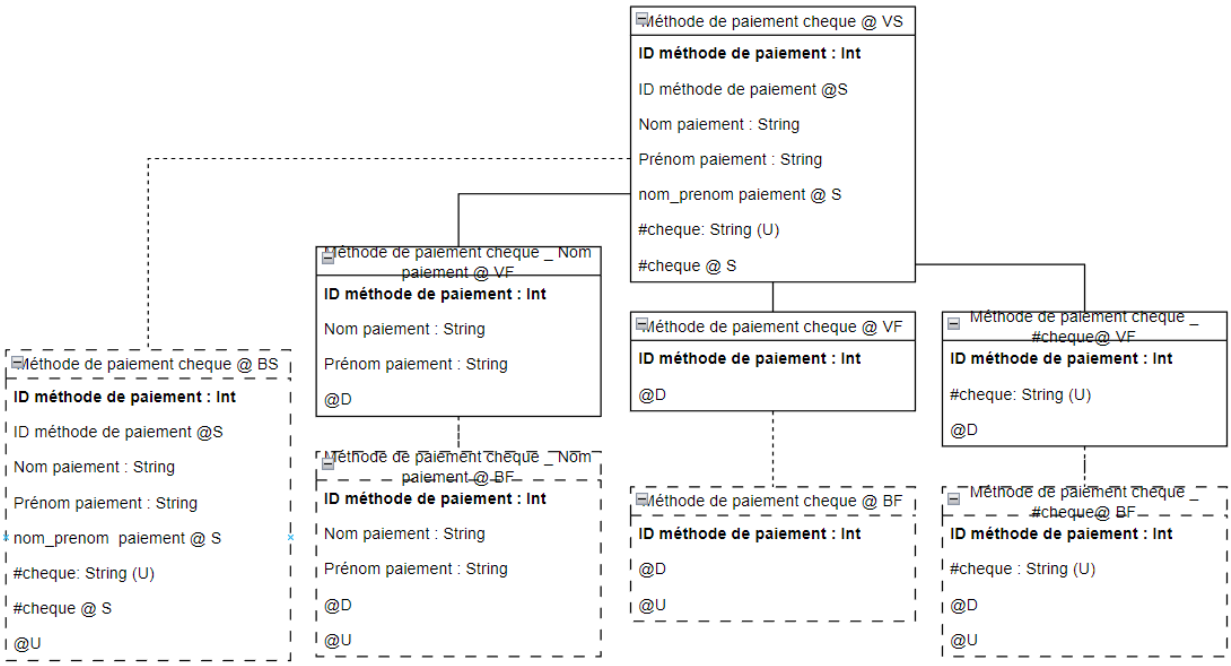
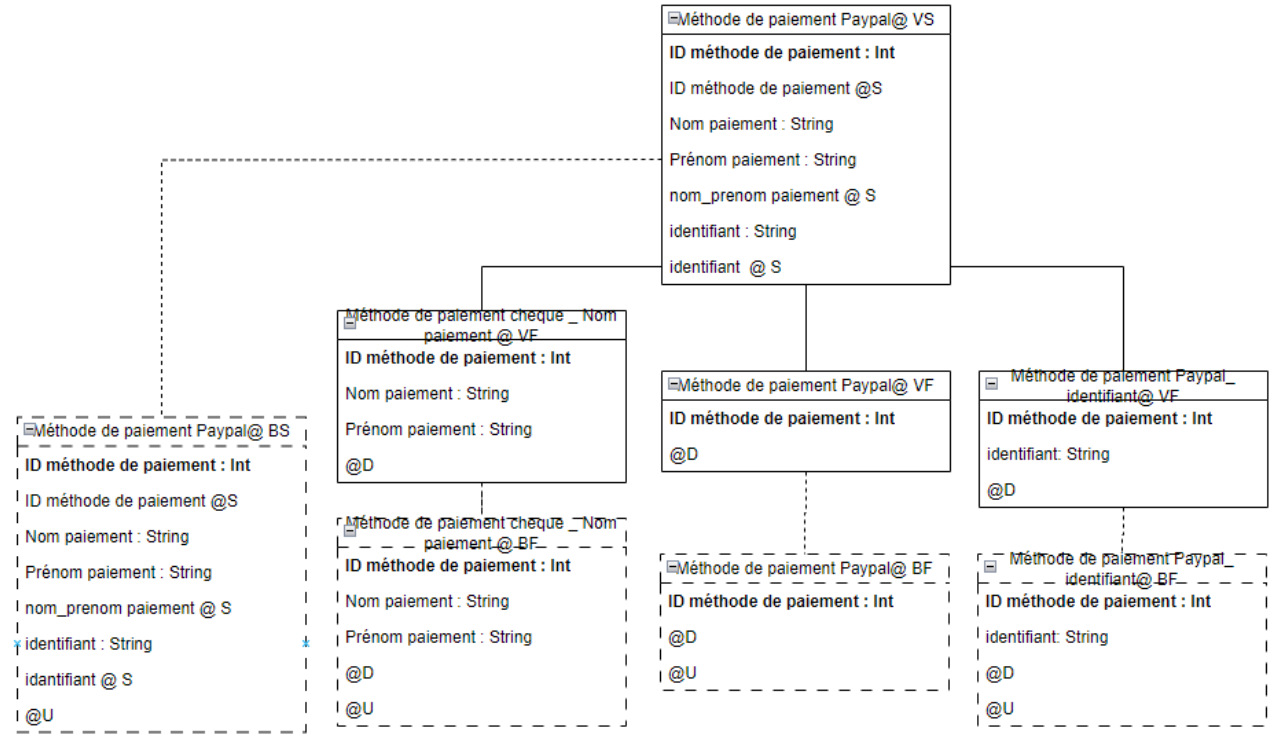


Table : Méthode de paiement Paypal



Jalon 3 : Modélisation, conception et mise à l'essai de l'API de la BDT

Tests des tables

Table Étudiant

Insertion de données dans la table courante

L'insertion dans la table courante se fait à l'aide de la fonction « etudiant_courante_ajout_at » pour spécifier le début de la date de validité, ou avec la fonction « etudiant_courante_ajout_now » pour faire débuter la date de validité au moment présent.

Ces fonctions ont été testé à l'aide des commandes suivantes :

```
--Fonction d'ajout dans la table courante--
CALL etudiant_courante_ajout_at('1000111000', 'max', 'S', 'maxS@test.ca', '0001112222', 1, '2016-06-22 19:10:25');
CALL etudiant_courante_ajout_now('1000111001', 'max2', 'S', 'max2S@test.ca', '3334445555', 2);
```

Les résultats de ces fonctions sont l'ajout dans la table courante, tel que montré dans l'image suivante.

	matricule [PK] character (10)	matricule_since timestamp without time zone	prenom text	nom text	nom_prenom_since timestamp without time zone	courriel text	telephone character varying (10)	contact_since timestamp without time zone	adresseid integer	adresseid_since timestamp without time zone
1	1000111000	2016-06-22 19:10:25	max	S	2016-06-22 19:10:25	maxS@te...	0001112222	2016-06-22 19:10:25	1	2016-06-22 19:10:25
2	1000111001	2023-12-10 16:48:19	max2	S	2023-12-10 16:48:19	max2S@t...	3334445555	2023-12-10 16:48:19	2	2023-12-10 16:48:19

Puisqu'un ajout a été réalisé dans la table courante, deux lignes ont été ajouté à la table de transaction, tel quel montré à l'image suivante.

	matricule character (10)	matricule_since timestamp without time zone	prenom text	nom text	nom_prenom_since timestamp without time zone	courriel text	telephone character varying (10)	contact_since timestamp without time zone	adresseid integer	adresseid_since timestamp without time zone	etudiant_transaction "IFT723".tsrange_sec
1	1000111000	2016-06-22 19:10:25	max	S	2016-06-22 19:10:25	maxS@te...	0001112222	2016-06-22 19:10:25	1	2016-06-22 19:10:25	["2023-12-10 16:48:19",)
2	1000111001	2023-12-10 16:48:19	max2	S	2023-12-10 16:48:19	max2S@t...	3334445555	2023-12-10 16:48:19	2	2023-12-10 16:48:19	["2023-12-10 16:48:19",)

Modification des données dans la table courante

La modification de la table courante se fait attribut par attribut. Chaque attribut peut être modifié à une date spécifique, ou à l'instant présent. L'attribut « Matricule » ne peut être modifié puisqu'il fait partie de la clé primaire de la table. Ces fonctions sont présentées dans le tableau suivant.

Attribut	Date spécifique	Instant présent
Nom & prenom	etudiant_courante_modifier_nom_prenom_at	etudiant_courante_modifier_nom_prenom_now
Courriel & Téléphone	etudiant_courante_modifier_contact_at	etudiant_courante_modifier_contact_now
AdresseID	etudiant_courante_modifier_adresseID_at	etudiant_courante_modifier_adresseID_now

Ces fonctions ont été testé à l'aide des commandes suivantes :

```
--Fonction modification de la table courante
CALL etudiant_courante_modifier_nom_prenom_at('1000111000', 'max_2', 'S_2', '2017-06-22 19:10:25');
CALL etudiant_courante_modifier_nom_prenom_now('1000111001', 'max2_2', 'S_2');
CALL etudiant_courante_modifier_contact_at('1000111000', 'maxS_2@test.ca', '6667778888', '2017-06-22 19:10:25');
CALL etudiant_courante_modifier_contact_now('1000111001', 'max2_2S@test.ca', '1234567890');
CALL etudiant_courante_modifier_adresseID_at('1000111000', 3, '2017-06-22 19:10:25');
CALL etudiant_courante_modifier_adresseID_now('1000111001', 4)
```

Note : Il est important d'exécuter ces fonctions les unes après les autres, et non d'un seul coup comme le suggère la photo ci-dessus. Dans le cas contraire, les intervalles des temps de validité et de transaction ne seront pas valides.

Les résultats de ces fonctions sont la modification des attributs dans la table courante, tel que montré dans l'image suivante.

	matricule [PK] character (10)	matricule_since timestamp without time zone	prenom text	nom text	nom_prenom_since timestamp without time zone	courriel text	telephone character varying (10)	contact_since timestamp without time zone	adresseid integer	adresseid_since timestamp without time zone
1	1000111000	2016-06-22 19:10:25	max_2	S_2	2017-06-22 19:10:25	maxS_2@test...	6667778888	2017-06-22 19:10:25	3	2017-06-22 19:10:25
2	1000111001	2023-12-10 16:48:19	max2_2	S_2	2023-12-10 16:49:46	max2_2S@te...	1234567890	2023-12-10 16:49:57	4	2023-12-10 16:50:07

Puisque des modifications ont été réalisé dans la table courante, la table de transaction a enregistré ces changements.

	matricule character (10)	matricule_since timestamp without time zone	prenom text	nom text	nom_prenom_since timestamp without time zone	courriel text	telephone character varying (10)	contact_since timestamp without time zone	adresseid integer	adresseid_since timestamp without time zone	etudiant_transaction "IFT723".tsrange_sec
1	1000111000	2016-06-22 19:10:25	max	S	2016-06-22 19:10:25	maxS@test.ca	0001112222	2016-06-22 19:10:25	1	2016-06-22 19:10:25	["2023-12-10 16:48:19","2023-12-10 16:49:30")
2	1000111001	2023-12-10 16:48:19	max2	S	2023-12-10 16:48:19	max2S@test...	3334445555	2023-12-10 16:48:19	2	2023-12-10 16:48:19	["2023-12-10 16:48:19","2023-12-10 16:49:46")
3	1000111000	2016-06-22 19:10:25	max_2	S_2	2017-06-22 19:10:25	maxS@test.ca	0001112222	2016-06-22 19:10:25	1	2016-06-22 19:10:25	["2023-12-10 16:49:30","2023-12-10 16:49:52")
4	1000111001	2023-12-10 16:48:19	max2_2	S_2	2023-12-10 16:49:46	max2S@test...	3334445555	2023-12-10 16:48:19	2	2023-12-10 16:48:19	["2023-12-10 16:49:46","2023-12-10 16:49:57")
5	1000111000	2016-06-22 19:10:25	max_2	S_2	2017-06-22 19:10:25	maxS_2@test...	6667778888	2017-06-22 19:10:25	1	2016-06-22 19:10:25	["2023-12-10 16:49:52","2023-12-10 16:50:01")
6	1000111000	2016-06-22 19:10:25	max_2	S_2	2017-06-22 19:10:25	maxS_2@test...	6667778888	2017-06-22 19:10:25	3	2017-06-22 19:10:25	["2023-12-10 16:50:01",)
7	1000111001	2023-12-10 16:48:19	max2_2	S_2	2023-12-10 16:49:46	max2_2S@te...	1234567890	2023-12-10 16:49:57	2	2023-12-10 16:48:19	["2023-12-10 16:49:57","2023-12-10 16:50:07")
8	1000111001	2023-12-10 16:48:19	max2_2	S_2	2023-12-10 16:49:46	max2_2S@te...	1234567890	2023-12-10 16:49:57	4	2023-12-10 16:50:07	["2023-12-10 16:50:07",)

Les tables de validité ont aussi capté ces changements, avec leurs tables de transaction.

Attributs	Table de validité	Table de transaction
Nom & prenom	matricule character (10) | prenom text | nom text | nom_prenom_validite "IFT723".tsrange_sec 1 | 1000111000 | max | S | ["2016-06-22 19:10:25","2017-06-22 19:10:25") 2 | 1000111001 | max2 | S | ["2023-12-10 16:48:19","2023-12-10 16:49:46")	matricule character (10) | prenom text | nom text | nom_prenom_validite "IFT723".tsrange_sec | nom_prenom_transaction "IFT723".tsrange_sec 1 | 1000111000 | max | S | ["2016-06-22 19:10:25","2017-06-22 19:10:25") | ["2023-12-10 16:49:30",) 2 | 1000111001 | max2 | S | ["2023-12-10 16:48:19","2023-12-10 16:49:46") | ["2023-12-10 16:49:46",)
Courriel & Téléphone	matricule character (10) | courriel text | telephone character varying (10) | contact_validite "IFT723".tsrange_sec 1 | 1000111000 | maxS@te... | 0001112222 | ["2016-06-22 19:10:25","2017-06-22 19:10:25") 2 | 1000111001 | max2S@t... | 3334445555 | ["2023-12-10 16:48:19","2023-12-10 16:49:57")	matricule character (10) | courriel text | telephone character varying (10) | contact_validite "IFT723".tsrange_sec | contact_transaction "IFT723".tsrange_sec 1 | 1000111000 | maxS@te... | 0001112222 | ["2016-06-22 19:10:25","2017-06-22 19:10:25") | ["2023-12-10 16:49:52",) 2 | 1000111001 | max2S@t... | 3334445555 | ["2023-12-10 16:48:19","2023-12-10 16:49:57") | ["2023-12-10 16:49:57",)
AdresseID	matricule character (10) | adresseid integer | adresseid_validite "IFT723".tsrange_sec 1 | 1000111000 | 1 | ["2016-06-22 19:10:25","2017-06-22 19:10:25") 2 | 1000111001 | 2 | ["2023-12-10 16:48:19","2023-12-10 16:50:07")	matricule character (10) | adresseid integer | adresseid_validite "IFT723".tsrange_sec | adresseid_transaction "IFT723".tsrange_sec 1 | 1000111000 | 1 | ["2016-06-22 19:10:25","2017-06-22 19:10:25") | ["2023-12-10 16:50:01",) 2 | 1000111001 | 2 | ["2023-12-10 16:48:19","2023-12-10 16:50:07") | ["2023-12-10 16:50:07",)

Effacement des données dans la table courante

L'effacement de données dans la table courante se fait à l'aide de la fonction « etudiant_courante_retrait_at » pour spécifier le début de la date de validité, ou avec la fonction « etudiant_courante_retrait_now» pour faire débuter la date de validité au moment présent.

Ces fonctions ont été testé à l'aide des commandes suivantes :

```
--Fonction effacement dans la table courante--
CALL etudiant_courante_retrait_at('1000111000', '2018-06-22 19:10:25');
CALL etudiant_courante_retrait_now('1000111001');
```

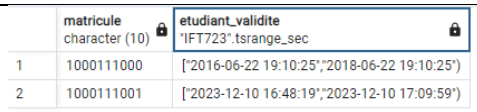
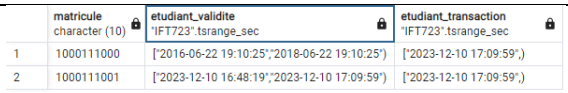
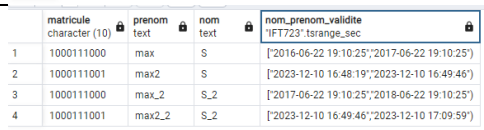
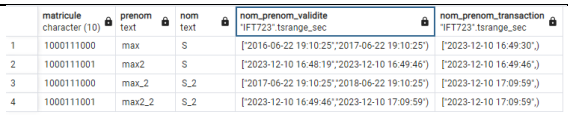
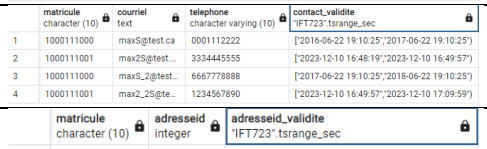
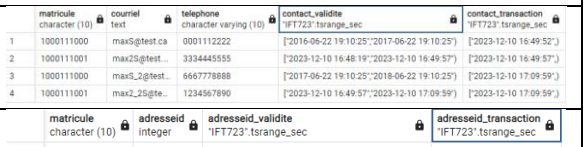
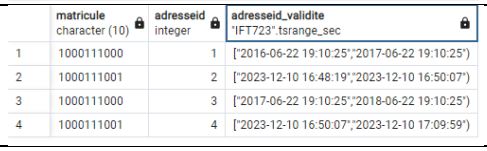
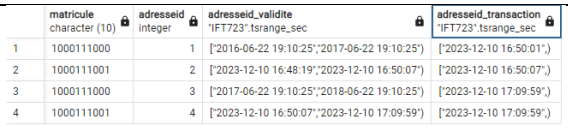
Les résultats de ces fonctions sont l'effacement des données dans la table courante, tel que montré dans l'image suivante.

	matricule [PK] character (10)	matricule_since timestamp without time zone	prenom text	nom text	nom_prenom_since timestamp without time zone	courriel text	telephone character varying (10)	contact_since timestamp without time zone	adresseid integer	adresseid_since timestamp without time zone

Puisque des modifications ont été réalisé dans la table courante, la table de transaction a enregistré ces changements.

	matricule character (10)	matricule_since timestamp without time zone	prenom text	nom text	nom_prenom_since timestamp without time zone	courriel text	telephone character varying (10)	contact_since timestamp without time zone	adresseid integer	adresseid_since timestamp without time zone	etudiant_transaction "IFT723".tsrange_sec
1	1000111000	2016-06-22 19:10:25	max	S	2016-06-22 19:10:25	maxS@test.ca	0001112222	2016-06-22 19:10:25	1	2016-06-22 19:10:25	["2023-12-10 16:48:19","2023-12-10 16:49:30")
2	1000111001	2023-12-10 16:48:19	max2	S	2023-12-10 16:48:19	max2S@test.ca	3334445555	2023-12-10 16:48:19	2	2023-12-10 16:48:19	["2023-12-10 16:48:19","2023-12-10 16:49:46")
3	1000111000	2016-06-22 19:10:25	max_2	S_2	2017-06-22 19:10:25	maxS@test.ca	0001112222	2016-06-22 19:10:25	1	2016-06-22 19:10:25	["2023-12-10 16:49:30","2023-12-10 16:49:52")
4	1000111001	2023-12-10 16:48:19	max2_2	S_2	2023-12-10 16:49:46	max2S@test.ca	3334445555	2023-12-10 16:48:19	2	2023-12-10 16:48:19	["2023-12-10 16:49:46","2023-12-10 16:49:57")
5	1000111000	2016-06-22 19:10:25	max_2	S_2	2017-06-22 19:10:25	maxS_2@test.ca	6667778888	2017-06-22 19:10:25	1	2016-06-22 19:10:25	["2023-12-10 16:49:52","2023-12-10 16:50:01")
6	1000111001	2023-12-10 16:48:19	max2_2	S_2	2023-12-10 16:49:46	max2_2S@test....	1234567890	2023-12-10 16:49:57	2	2023-12-10 16:48:19	["2023-12-10 16:49:57","2023-12-10 16:50:07")
7	1000111000	2016-06-22 19:10:25	max_2	S_2	2017-06-22 19:10:25	maxS_2@test.ca	6667778888	2017-06-22 19:10:25	3	2017-06-22 19:10:25	["2023-12-10 16:50:01","2023-12-10 17:09:59")
8	1000111001	2023-12-10 16:48:19	max2_2	S_2	2023-12-10 16:49:46	max2_2S@test....	1234567890	2023-12-10 16:49:57	4	2023-12-10 16:50:07	["2023-12-10 16:50:07","2023-12-10 17:09:59")

Les tables de validité ont aussi capté ces changements, avec leurs tables de transaction.

Attributs	Table de validité	Table de transaction
Matricule		
Nom & prenom		
Courriel & Téléphone		
AdresseID		

	matricule character (10)	etudiant_validite "IFT723".tsrange_sec
1	1000111000	["2016-06-22 19:10:25","2018-06-22 19:10:25")
2	1000111001	["2023-12-10 16:48:19","2023-12-10 17:09:59")

	matricule character (10)	etudiant_validite "IFT723".tsrange_sec	etudiant_transaction "IFT723".tsrange_sec
1	1000111000	["2016-06-22 19:10:25","2018-06-22 19:10:25")	["2023-12-10 17:09:59",)
2	1000111001	["2023-12-10 16:48:19","2023-12-10 17:09:59")	["2023-12-10 17:09:59",)

	matricule character (10)	prenom text	nom text	nom_prenom_validite "IFT723".tsrange_sec
1	1000111000	max	S	["2016-06-22 19:10:25","2017-06-22 19:10:25")
2	1000111001	max2	S	["2023-12-10 16:48:19","2023-12-10 16:49:46")
3	1000111000	max_2	S_2	["2017-06-22 19:10:25","2018-06-22 19:10:25")
4	1000111001	max2_2	S_2	["2023-12-10 16:49:46","2023-12-10 16:49:57")

	matricule character (10)	prenom text	nom text	nom_prenom_validite "IFT723".tsrange_sec	nom_prenom_transaction "IFT723".tsrange_sec
1	1000111000	max	S	["2016-06-22 19:10:25","2017-06-22 19:10:25")	["2023-12-10 16:49:30",)
2	1000111001	max2	S	["2023-12-10 16:48:19","2023-12-10 16:49:46")	["2023-12-10 16:49:46",)
3	1000111000	max_2	S_2	["2017-06-22 19:10:25","2018-06-22 19:10:25")	["2023-12-10 17:09:59",)
4	1000111001	max2_2	S_2	["2023-12-10 16:49:46","2023-12-10 17:09:59")	["2023-12-10 17:09:59",)

	matricule character (10)	courriel text	telephone character varying (10)	contact_validite "IFT723".tsrange_sec
1	1000111000	maxS@test.ca	0001112222	["2016-06-22 19:10:25","2017-06-22 19:10:25")
2	1000111001	max2S@test...	3334445555	["2023-12-10 16:48:19","2023-12-10 16:49:57")
3	1000111000	maxS_2@test...	6667778888	["2017-06-22 19:10:25","2018-06-22 19:10:25")
4	1000111001	max2_2S@te...	1234567890	["2023-12-10 16:49:57","2023-12-10 17:09:59")

	matricule character (10)	courriel text	telephone character varying (10)	contact_validite "IFT723".tsrange_sec	contact_transaction "IFT723".tsrange_sec
1	1000111000	maxS@test.ca	0001112222	["2016-06-22 19:10:25","2017-06-22 19:10:25")	["2023-12-10 16:49:52",)
2	1000111001	max2S@test...	3334445555	["2023-12-10 16:48:19","2023-12-10 16:49:57")	["2023-12-10 16:49:57",)
3	1000111000	maxS_2@test...	6667778888	["2017-06-22 19:10:25","2018-06-22 19:10:25")	["2023-12-10 17:09:59",)
4	1000111001	max2_2S@te...	1234567890	["2023-12-10 16:49:57","2023-12-10 17:09:59")	["2023-12-10 17:09:59",)

	matricule character (10)	adresseid integer	adresseid_validite "IFT723".tsrange_sec
1	1000111000	1	["2016-06-22 19:10:25","2017-06-22 19:10:25")
2	1000111001	2	["2023-12-10 16:48:19","2023-12-10 16:50:07")
3	1000111000	3	["2017-06-22 19:10:25","2018-06-22 19:10:25")
4	1000111001	4	["2023-12-10 16:50:07","2023-12-10 17:09:59")

	matricule character (10)	adresseid integer	adresseid_validite "IFT723".tsrange_sec	adresseid_transaction "IFT723".tsrange_sec
1	1000111000	1	["2016-06-22 19:10:25","2017-06-22 19:10:25")	["2023-12-10 16:50:01",)
2	1000111001	2	["2023-12-10 16:48:19","2023-12-10 16:50:07")	["2023-12-10 16:50:07",)
3	1000111000	3	["2017-06-22 19:10:25","2018-06-22 19:10:25")	["2023-12-10 17:09:59",)
4	1000111001	4	["2023-12-10 16:50:07","2023-12-10 17:09:59")	["2023-12-10 17:09:59",)

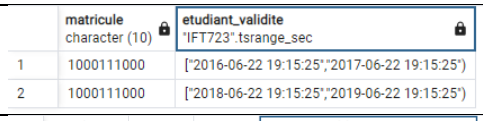
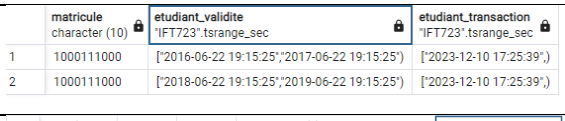
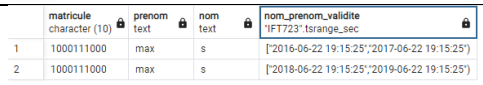
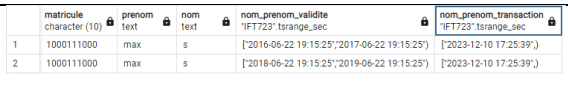
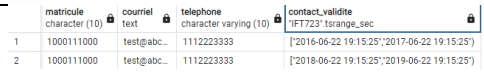
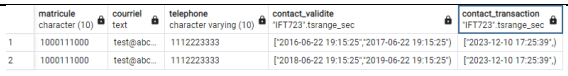
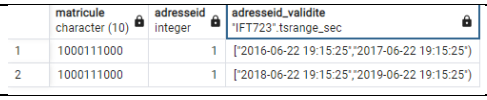
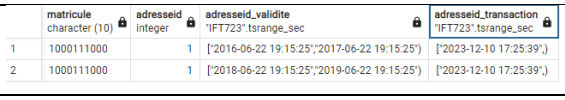
Insertion de données dans les tables de validité

L'insertion de données dans les tables de validité se fait à l'aide de la commande « etudiant_validite_ajout ».

La fonction a été testé à l'aide de la commandes suivantes :

```
--Fonction pour l'ajout des tables de validités
CALL etudiant_validite_ajout('1000111000', 'max', 's', 'test@abc.ca', '1112223333', 1, '2016-06-22 19:15:25','2017-06-22 19:15:25');
CALL etudiant_validite_ajout('1000111000', 'max', 's', 'test@abc.ca', '1112223333', 1, '2018-06-22 19:15:25','2019-06-22 19:15:25');
```

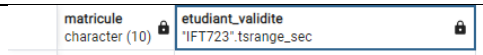
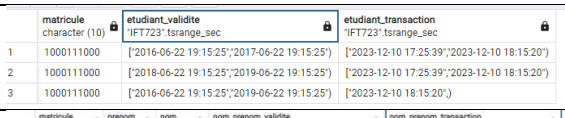
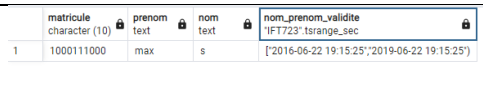
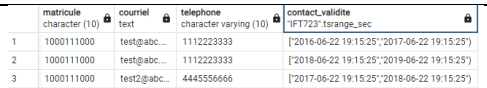
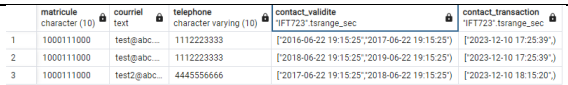
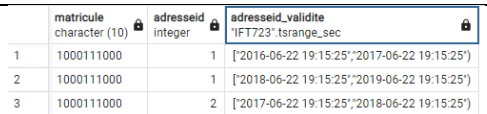
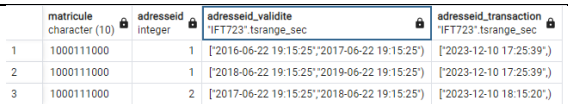
Le résultat de ces fonctions sont l'ajout de données dans les table de validité, tel que montré dans les images suivantes, les tables de transactions ont aussi enregistrés ces changements

Attributs	Table de validité	Table de transaction
Matricule		
Nom & prenom		
Courriel & Téléphone		
AdresseID		

Dans le cas d'une insertion de données adjacentes à celle déjà présente dans la table de validité, elles seront combinées. Ce cas est démontré en rajoutant la ligne suivante, qui contient des données similaires pour le nom et le prénom, et des données différentes pour l'adresse, le courriel et le téléphone pour la période définie entre les deux lignes déjà présente dans la base de données.

```
--Fonction pour forcer la résolution de données adjacente des tables de validités
CALL etudiant_validite_ajout('1000111000', 'max', 's', 'test2@abc.ca', '4445556666', 2, '2017-06-22 19:15:25','2018-06-22 19:15:25');
```

On devrait donc avoir une seule ligne dans les tables de validité du matricule et du nom et prénom, et trois lignes dans les tables du courriel et téléphone et adresseID.

Attributs	Table de validité	Table de transaction
Matricule		
Nom & prenom		
Courriel & Téléphone		
AdresseID		

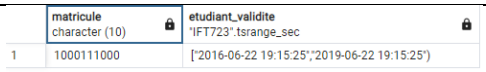
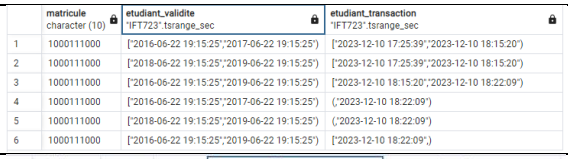
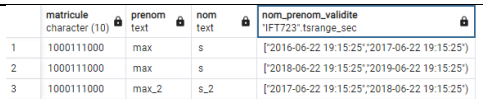
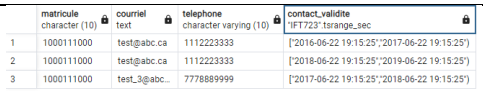
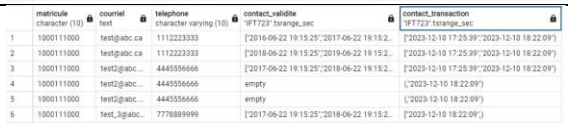
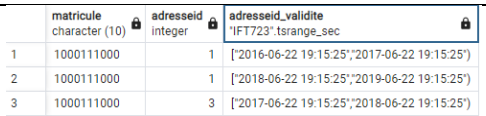
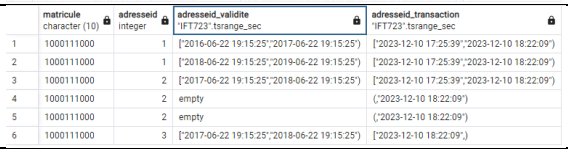
Modification des données dans les tables de validité

La modification des données dans les tables de validité se fait à l'aide de la commande « etudiant_validite_modification ». Elle ne fait qu'effacer et insérer les données sur la période fourni en paramètre.

La fonction a été testé à l'aide de la commandes suivantes :

```
--Fonction pour modifier des tables de validités
CALL etudiant_validite_modification('1000111000', 'max_2', 's_2', 'test_3@abc.ca', '7778889999', 3, '2017-06-22 19:15:25','2018-06-22 19:15:25')
```

Le résultat de la modification est représenté dans les images suivantes :

Attributs	Table de validité	Table de transaction
Matricule		
Nom & prenom		
Courriel & Téléphone		
AdresseID		

Effacement des données dans les tables de validité

L'effacement de données dans les tables de validité se fait à l'aide de la commande « etudiant_validite_effacer ».

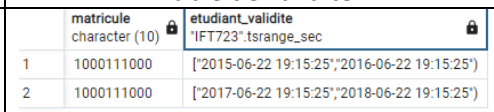
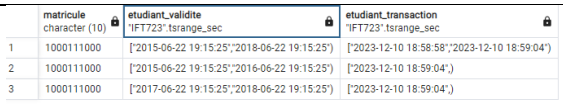
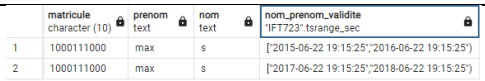
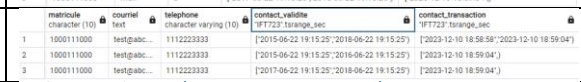
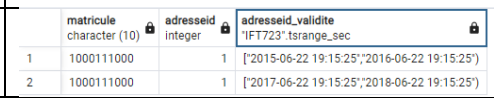
Lors de l'effacement, 4 situations peuvent survenir, tel que montré dans le schéma suivant. Chacune de ces situations seront testés.

```
--Si le nouveau range est contenu dans un autre element
--Ancien   : ==========
--Effacer  :    ====
--Résultat : ===    ===

--Si des elements sont contenus dans le range
--Ancien 1 :     ====
--Ancien 2 :         =====
--Effacer  :    ===========
--Résultat :

--S'il y a chevauchement & ne s'étend pas sur la droite
--Ancien   : =======
--Effacer  :      =====
--Résultat : =====

--S'il y a chevauchement & ne s'étend pas sur la gauche
--Ancien   :    =======
--Effacer  : =====
--Résultat :      =====
```

La première situation peut être testé avec les commandes suivantes

```
--Fonction pour l'effacement des tables de validités
    --Si le nouveau range est contenu dans un autre element
    --Ancien   : ==========
    --Effacer  :    ====
    --Résultat : ===    ===
CALL etudiant_validite_ajout('1000111000', 'max', 's', 'test@abc.ca', '1112223333', 1, '2015-06-22 19:15:25','2018-06-22 19:15:25');
CALL etudiant_validite_effacer('1000111000', '2016-06-22 19:15:25', '2017-06-22 19:15:25');
```

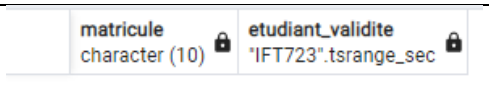
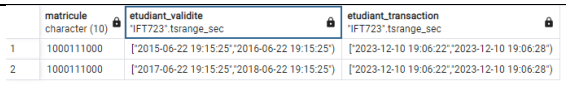
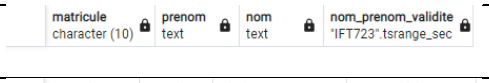
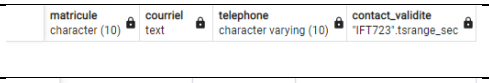
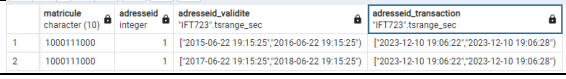
Le résultat de l'effacement est représenté dans les images suivantes :

Attributs	Table de validité	Table de transaction
Matricule		
Nom & prenom		
Courriel & Téléphone		
AdresseID		

La seconde situation peut être testé avec les commandes suivantes

```
    --Si des elements sont contenus dans le range
    --Ancien 1 :     ====
    --Ancien 2 :         =====
    --Effacer  :    ===========
    --Résultat :
CALL etudiant_validite_ajout('1000111000', 'max', 's', 'test@abc.ca', '1112223333', 1, '2015-06-22 19:15:25','2016-06-22 19:15:25');
CALL etudiant_validite_ajout('1000111000', 'max', 's', 'test@abc.ca', '1112223333', 1, '2017-06-22 19:15:25','2018-06-22 19:15:25');
CALL etudiant_validite_effacer('1000111000', '2014-06-22 19:15:25', '2019-06-22 19:15:25');
```

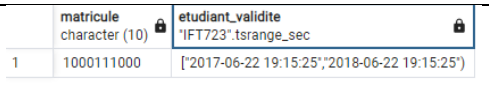
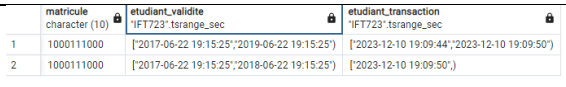
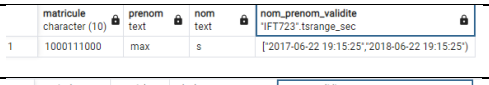
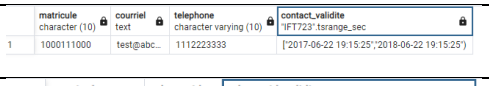
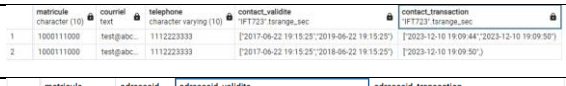
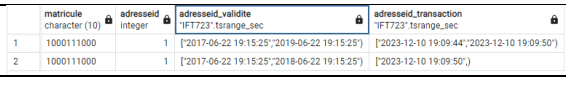
Le résultat de l'effacement est représenté dans les images suivantes :

Attributs	Table de validité	Table de transaction
Matricule		
Nom & prenom		
Courriel & Téléphone		
AdresseID		

La troisième situation peut être testé avec les commandes suivantes

```
    --S'il y a chevauchement & ne s'étend pas sur la droite
    --Ancien   : =======
    --Effacer  :      =====
    --Résultat : =====
CALL etudiant_validite_ajout('1000111000', 'max', 's', 'test@abc.ca', '1112223333', 1, '2017-06-22 19:15:25','2019-06-22 19:15:25');
CALL etudiant_validite_effacer('1000111000', '2018-06-22 19:15:25', '2020-06-22 19:15:25');
```

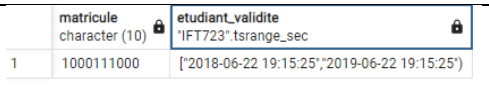
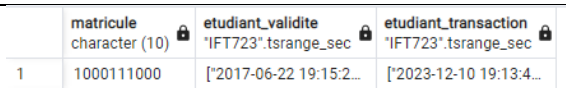
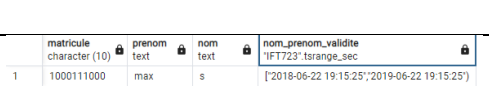
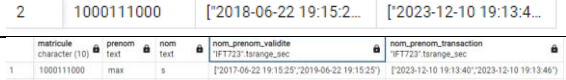
Le résultat de l'effacement est représenté dans les images suivantes :

Attributs	Table de validité	Table de transaction
Matricule		
Nom & prenom		
Courriel & Téléphone		
AdresseID		

La quatrième situation peut être testé avec les commandes suivantes

```
    --S'il y a chevauchement & ne s'étend pas sur la gauche
    --Ancien   :    =======
    --Effacer  : =====
    --Résultat :      =====
CALL etudiant_validite_ajout('1000111000', 'max', 's', 'test@abc.ca', '1112223333', 1, '2017-06-22 19:15:25','2019-06-22 19:15:25');
CALL etudiant_validite_effacer('1000111000', '2016-06-22 19:15:25', '2018-06-22 19:15:25');
```

Le résultat de l'effacement est représenté dans les images suivantes :

Attributs	Table de validité	Table de transaction
Matricule		
Nom & prenom		

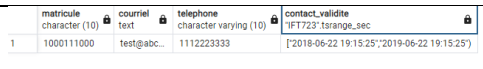
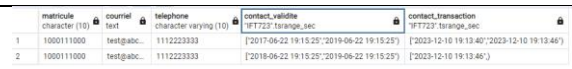
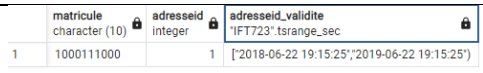
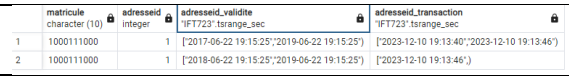
Courriel & Téléphone		
AdresseID		

Table Adresse

Insertion de données dans la table courante

L'insertion dans la table courante se fait à l'aide de la fonction « Adresse_Courante_ajout_at» pour spécifier le début de la date de validité, ou avec la fonction « Adresse_Courante_ajout_now » pour faire débuter la date de validité au moment présent.

Ces fonctions ont été testé à l'aide des commandes suivantes :

```
--Fonction d'ajout dans la table courante
CALL Adresse_Courante_ajout_at(1, '1', 'testRue', 'testVille', 'Qc',  'A1A 1A1', 'CA', '2017-06-22 19:10:25');
CALL Adresse_Courante_ajout_now(2, '2', 'testRue2', 'testVille2', 'Qc2',  'A2A 2A2', 'CA');
```

Les résultats de ces fonctions sont l'ajout dans la table courante, tel que montré dans l'image suivante.

	adresseid [PK] integer	adresseid_since timestamp without time zone	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_since timestamp without time zone
1	1	2017-06-22 19:10:25	1	testRue	testVille	Qc	A1A 1A1	CA	2017-06-22 19:10:25
2	2	2023-12-11 00:19:56	2	testRue2	testVille2	Qc2	A2A 2A2	CA	2023-12-11 00:19:56

Puisqu'un ajout a été réalisé dans la table courante, deux lignes ont été ajouté à la table de transaction, tel quel montré à l'image suivante.

	adresseid integer	adresseid_since timestamp without time zone	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_since timestamp without time zone	adresse_transaction "IFT723".tsrange_sec
1	1	2017-06-22 19:10:25	1	testRue	testVille	Qc	A1A 1A1	CA	2017-06-22 19:10:25	["2023-12-11 00:19:56",)
2	2	2023-12-11 00:19:56	2	testRue2	testVille2	Qc2	A2A 2A2	CA	2023-12-11 00:19:56	["2023-12-11 00:19:56",)

Modification des données dans la table courante

La modification de la table courante se fait à l'aide de la commande
« Adresse_Courante_modifier_Localisation_at » pour spécifier le début de la date de validité, ou avec la fonction « Adresse_Courante_modifier_Localisation_now» pour faire débuter la date de validité au moment présent.

Ces fonctions ont été testé à l'aide des commandes suivantes :

```
--Fonction de modification de la table courante
CALL Adresse_Courante_modifier_Localisation_at(1, '1', 'testRue_2', 'testVille_2', 'Qc_2',  'A1A 1A1', 'CA', '2019-06-22 19:10:25');
CALL Adresse_Courante_modifier_Localisation_now(2, '2', 'testRue2_2', 'testVille2_@', 'Qc2_2',  'A2A 2A2', 'CA');
```

Note : Il est important d'exécuter ces fonctions les unes après les autres, et non d'un seul coup comme le suggère la photo ci-dessus. Dans le cas contraire, les intervalles des temps de validité et de transaction ne seront pas valides.

Les résultats de ces fonctions sont la modification des attributs dans la table courante, tel que montré dans l'image suivante.

	adresseid [PK] integer	adresseid_since timestamp without time zone	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_since timestamp without time zone
1	1	2017-06-22 19:10:25	1	testRue_2	testVille_2	Qc_2	A1A 1A1	CA	2019-06-22 19:10:25
2	2	2023-12-11 00:19:56	2	testRue2_2	testVille2...	Qc2_2	A2A 2A2	CA	2023-12-11 00:30:47

Puisque des modifications ont été réalisé dans la table courante, la table de transaction a enregistré ces changements.

	adresseid integer	adresseid_since timestamp without time zone	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_since timestamp without time zone	adresse_transaction "IFT723".tsrange_sec
1	1	2017-06-22 19:10:25	1	testRue	testVille	Qc	A1A 1A1	CA	2017-06-22 19:10:25	["2023-12-11 00:19:56","2023-12-11 00:30:40")
2	1	2017-06-22 19:10:25	1	testRue_2	testVille_2	Qc_2	A1A 1A1	CA	2019-06-22 19:10:25	["2023-12-11 00:30:40",)
3	2	2023-12-11 00:19:56	2	testRue2	testVille2	Qc2	A2A 2A2	CA	2023-12-11 00:19:56	["2023-12-11 00:19:56","2023-12-11 00:30:47")
4	2	2023-12-11 00:19:56	2	testRue2_2	testVille2...	Qc2_2	A2A 2A2	CA	2023-12-11 00:30:47	["2023-12-11 00:30:47",)

La table de validité de localisation a aussi capté les changements, avec sa table de transaction.

	adresseid integer	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_validite "IFT723".tsrange_sec
1	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2017-06-22 19:10:25","2019-06-22 19:10:25")
2	2	2	testRue2	testVille2	Qc2	A2A 2A2	CA	["2023-12-11 00:19:56","2023-12-11 00:30:47")

	adresseid integer	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_validite "IFT723".tsrange_sec	localisation_transaction "IFT723".tsrange_sec
1	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2017-06-22 19:10:25","2019-06-22 19:10:25")	["2023-12-11 00:30:40",)
2	2	2	testRue2	testVille2	Qc2	A2A 2A2	CA	["2023-12-11 00:19:56","2023-12-11 00:30:47")	["2023-12-11 00:30:47",)

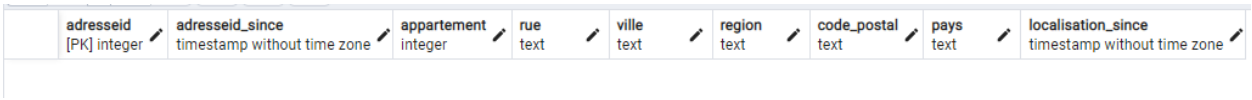
Effacement des données dans la table courante

L'effacement de données dans la table courante se fait à l'aide de la fonction « Adresse_Courante_retrait_at » pour spécifier le début de la date de validité, ou avec la fonction « Adresse_Courante_retrait_now» pour faire débuter la date de validité au moment présent.

Ces fonctions ont été testé à l'aide des commandes suivantes :

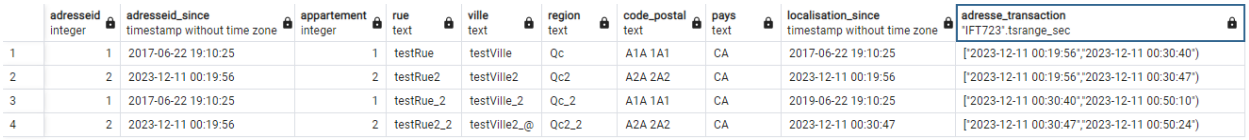
```
--Fonction d'effacement de la table courante
CALL Adresse_Courante_retrait_at(1, '2020-06-22 19:10:25');
CALL Adresse_Courante_retrait_now(2);
```

Les résultats de ces fonctions sont l'effacement des données dans la table courante, tel que montré dans l'image suivante.



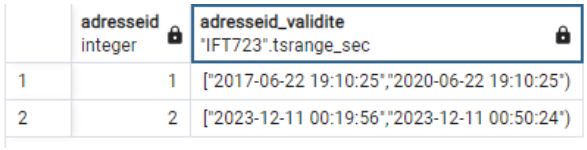
	adresseid [PK] integer	adresseid_since timestamp without time zone	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_since timestamp without time zone

Puisque des modifications ont été réalisé dans la table courante, la table de transaction a enregistré ces changements.

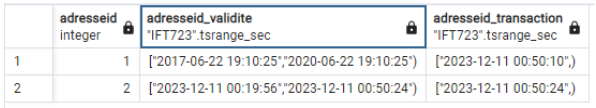


	adresseid integer	adresseid_since timestamp without time zone	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_since timestamp without time zone	adresse_transaction "IFT723".tsrange_sec
1	1	2017-06-22 19:10:25	1	testRue	testVille	Qc	A1A 1A1	CA	2017-06-22 19:10:25	["2023-12-11 00:19:56","2023-12-11 00:30:40")
2	2	2023-12-11 00:19:56	2	testRue2	testVille2	Qc2	A2A 2A2	CA	2023-12-11 00:19:56	["2023-12-11 00:19:56","2023-12-11 00:30:47")
3	1	2017-06-22 19:10:25	1	testRue_2	testVille_2	Qc_2	A1A 1A1	CA	2019-06-22 19:10:25	["2023-12-11 00:30:40","2023-12-11 00:50:10")
4	2	2023-12-11 00:19:56	2	testRue2_2	testVille2_@	Qc2_2	A2A 2A2	CA	2023-12-11 00:30:47	["2023-12-11 00:30:47","2023-12-11 00:50:24")

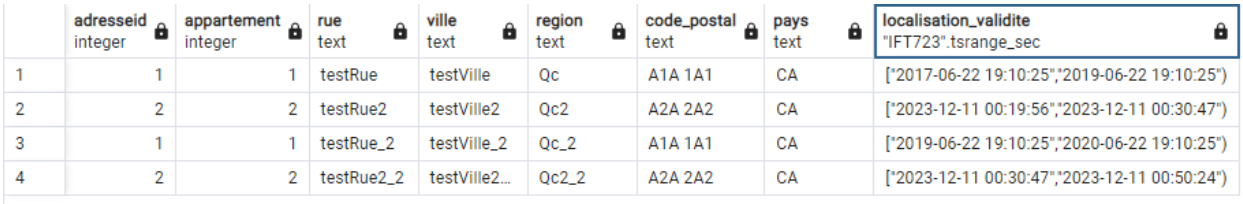
Les tables de validité d'adresseID et de localisation ont aussi capté les changements, avec leurs tables de transaction.



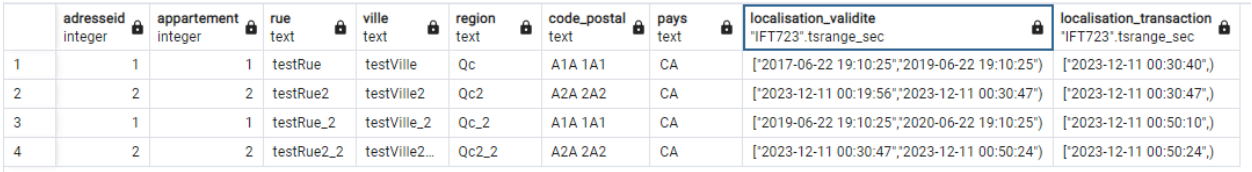
	adresseid integer	adresseid_validite "IFT723".tsrange_sec
1	1	["2017-06-22 19:10:25","2020-06-22 19:10:25")
2	2	["2023-12-11 00:19:56","2023-12-11 00:50:24")



	adresseid integer	adresseid_validite "IFT723".tsrange_sec	adresseid_transaction "IFT723".tsrange_sec
1	1	["2017-06-22 19:10:25","2020-06-22 19:10:25")	["2023-12-11 00:50:10",)
2	2	["2023-12-11 00:19:56","2023-12-11 00:50:24")	["2023-12-11 00:50:24",)



	adresseid integer	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_validite "IFT723".tsrange_sec
1	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2017-06-22 19:10:25","2019-06-22 19:10:25")
2	2	2	testRue2	testVille2	Qc2	A2A 2A2	CA	["2023-12-11 00:19:56","2023-12-11 00:30:47")
3	1	1	testRue_2	testVille_2	Qc_2	A1A 1A1	CA	["2019-06-22 19:10:25","2020-06-22 19:10:25")
4	2	2	testRue2_2	testVille2...	Qc2_2	A2A 2A2	CA	["2023-12-11 00:30:47","2023-12-11 00:50:24")



	adresseid integer	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_validite "IFT723".tsrange_sec	localisation_transaction "IFT723".tsrange_sec
1	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2017-06-22 19:10:25","2019-06-22 19:10:25")	["2023-12-11 00:30:40",)
2	2	2	testRue2	testVille2	Qc2	A2A 2A2	CA	["2023-12-11 00:19:56","2023-12-11 00:30:47")	["2023-12-11 00:30:47",)
3	1	1	testRue_2	testVille_2	Qc_2	A1A 1A1	CA	["2019-06-22 19:10:25","2020-06-22 19:10:25")	["2023-12-11 00:50:10",)
4	2	2	testRue2_2	testVille2...	Qc2_2	A2A 2A2	CA	["2023-12-11 00:30:47","2023-12-11 00:50:24")	["2023-12-11 00:50:24",)

Insertion de données dans les tables de validité

L'insertion de données dans les tables de validité se fait à l'aide de la commande « adresse_validite_ajout ».

La fonction a été testé à l'aide de la commandes suivantes :

```
--Fonction pour l'ajout dans les tables de validités
CALL adresse_validite_ajout(1, '1', 'testRue', 'testVille', 'Qc',  'A1A 1A1', 'CA', '2016-06-22 19:10:25', '2017-06-22 19:10:25');
CALL adresse_validite_ajout(1, '1', 'testRue', 'testVille', 'Qc',  'A1A 1A1', 'CA', '2018-06-22 19:10:25', '2019-06-22 19:10:25');
```

Le résultat de ces fonctions est l'ajout de données dans les table de validité, tel que montré dans les images suivantes, les tables de transactions ont aussi enregistrés ces changements

	adresseid integer	adresseid_validite "IFT723".tsrange_sec
1	1	["2016-06-22 19:10:25","2017-06-22 19:10:25")
2	1	["2018-06-22 19:10:25","2019-06-22 19:10:25")

	adresseid integer	adresseid_validite "IFT723".tsrange_sec	adresseid_transaction "IFT723".tsrange_sec
1	1	["2016-06-22 19:10:25","2017-06-22 19:10:25")	["2023-12-11 01:46:48",)
2	1	["2018-06-22 19:10:25","2019-06-22 19:10:25")	["2023-12-11 01:47:25",)

	adresseid integer	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_validite "IFT723".tsrange_sec
1	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2016-06-22 19:10:25","2017-06-22 19:10:25")
2	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2018-06-22 19:10:25","2019-06-22 19:10:25")

	adresseid integer	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_validite "IFT723".tsrange_sec	localisation_transaction "IFT723".tsrange_sec
1	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2016-06-22 19:10:25","2017-06-22 19:10:25")	["2023-12-11 01:46:48",)
2	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2018-06-22 19:10:25","2019-06-22 19:10:25")	["2023-12-11 01:47:25",)

Dans le cas d'une insertion de données adjacentes à celle déjà présente dans la table de validité, elles seront combinées. Ce cas est démontré en rajoutant la ligne suivante :

```
--Fonction pour forcer la résolution de données adjacentes des tables de validités
CALL adresse_validite_ajout(1, '1', 'testRue', 'testVille', 'Qc',  'A1A 1A1', 'CA', '2017-06-22 19:10:25', '2018-06-22 19:10:25');
```

On devrait donc avoir une seule ligne dans les tables de validité.

	adresseid integer	adresseid_validite "IFT723".tsrange_sec
1	1	["2016-06-22 19:10:25","2019-06-22 19:10:25")

	adresseid integer	adresseid_validite "IFT723".tsrange_sec	adresseid_transaction "IFT723".tsrange_sec
1	1	["2016-06-22 19:10:25","2017-06-22 19:10:25")	["2023-12-11 01:46:48","2023-12-11 01:50:26")
2	1	["2018-06-22 19:10:25","2019-06-22 19:10:25")	["2023-12-11 01:46:48","2023-12-11 01:50:26")
3	1	["2016-06-22 19:10:25","2019-06-22 19:10:25")	["2023-12-11 01:50:26",)

	adresseid integer	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_validite "IFT723".tsrange_sec
1	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2016-06-22 19:10:25","2019-06-22 19:10:25")

	adresseid integer	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_validite "IFT723".tsrange_sec	localisation_transaction "IFT723".tsrange_sec
1	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2016-06-22 19:10:25","2017-06-22 19:10:2...	["2023-12-11 01:46:48","2023-12-11 01:50:26")
2	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2018-06-22 19:10:25","2019-06-22 19:10:2...	["2023-12-11 01:46:48","2023-12-11 01:50:26")
3	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2016-06-22 19:10:25","2019-06-22 19:10:2...	["2023-12-11 01:50:26",)

Modification des données dans les tables de validité

La modification des données dans les tables de validité se fait à l'aide de la commande « adresse_validite_modification ». Elle ne fait qu'effacer et insérer les données sur la période fourni en paramètre.

La fonction a été testé à l'aide de la commandes suivantes :

```
--Fonction pour modifier les tables de validités
CALL adresse_validite_modification(1, '1', 'testRue_2', 'testVille_2', 'Qc_2',  'A1A 1A1', 'CA', '2017-06-22 19:10:25', '2018-06-22 19:10:25');
```

Le résultat de la modification est représenté dans les images suivantes :

	adresseid integer	adresseid_validite "IFT723".tsrange_sec
1	1	["2016-06-22 19:10:25","2019-06-22 19:10:25")

	adresseid integer	adresseid_validite "IFT723".tsrange_sec	adresseid_transaction "IFT723".tsrange_sec
1	1	["2016-06-22 19:10:2...	["2023-12-11 01:59:00","2023-12-11 01:59:08")
2	1	["2018-06-22 19:10:2...	["2023-12-11 01:59:00","2023-12-11 01:59:08")
3	1	["2016-06-22 19:10:2...	["2023-12-11 01:59:08","2023-12-11 01:59:14")
4	1	["2016-06-22 19:10:2...	(,"2023-12-11 01:59:14")
5	1	["2018-06-22 19:10:2...	(,"2023-12-11 01:59:14")
6	1	["2016-06-22 19:10:2...	["2023-12-11 01:59:14",)

	adresseid integer	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_validite "IFT723".tsrange_sec
1	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2016-06-22 19:10:25","2017-06-22 19:10:25")
2	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2018-06-22 19:10:25","2019-06-22 19:10:25")
3	1	1	testRue_2	testVille_2	Qc_2	A1A 1A1	CA	["2017-06-22 19:10:25","2018-06-22 19:10:25")

	adresseid integer	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_validite "IFT723".tsrange_sec	localisation_transaction "IFT723".tsrange_sec
1	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2016-06-22 19:10:...	["2023-12-11 01:59:00","2023-12-11 01:59:08")
2	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2018-06-22 19:10:...	["2023-12-11 01:59:00","2023-12-11 01:59:08")
3	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2016-06-22 19:10:...	["2023-12-11 01:59:08","2023-12-11 01:59:14")
4	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2016-06-22 19:10:...	["2023-12-11 01:59:14",)
5	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2018-06-22 19:10:...	["2023-12-11 01:59:14",)
6	1	1	testRue_2	testVille_2	Qc_2	A1A 1A1	CA	["2017-06-22 19:10:...	["2023-12-11 01:59:14",)

Effacement des données dans les tables de validité

L'effacement de données dans les tables de validité se fait à l'aide de la commande « adresse_validite_effacer ».

Lors de l'effacement, 4 situations peuvent survenir, tel que montré dans le schéma suivant. Chacune de ces situations seront testés.

```
--Si le nouveau range est contenu dans un autre element
--Ancien   : ==========
--Effacer  :    ====
--Résultat : ===    ===

--Si des elements sont contenus dans le range
--Ancien 1 :     ====
--Ancien 2 :         =====
--Effacer  :    ===========
--Résultat :

--S'il y a chevauchement & ne s'étend pas sur la droite
--Ancien   : =======
--Effacer  :      =====
--Résultat : =====

--S'il y a chevauchement & ne s'étend pas sur la gauche
--Ancien   :    =======
--Effacer  : =====
--Résultat :      =====
```

La première situation peut être testé avec les commandes suivantes

```
--Fonction pour l'effacement des tables de validités
    --Si le nouveau range est contenu dans un autre element
    --Ancien   : ==========
    --Effacer  :    ====
    --Résultat : ===    ===
CALL adresse_validite_ajout(1, '1', 'testRue', 'testVille', 'Qc',  'A1A 1A1', 'CA', '2015-06-22 19:10:25', '2018-06-22 19:10:25');
CALL adresse_validite_effacer(1, '2016-06-22 19:15:25', '2017-06-22 19:15:25');
```

Le résultat de l'effacement est représenté dans les images suivantes :

	adresseid integer	adresseid_validite "IFT723".tsrange_sec
1	1	["2015-06-22 19:10:25","2016-06-22 19:15:25")
2	1	["2017-06-22 19:15:25","2018-06-22 19:10:25")

	adresseid integer	adresseid_validite "IFT723".tsrange_sec	adresseid_transaction "IFT723".tsrange_sec
1	1	["2015-06-22 19:10:2...	["2023-12-11 02:05:58","2023-12-11 02:06:29")
2	1	["2015-06-22 19:10:2...	["2023-12-11 02:06:29",)
3	1	["2017-06-22 19:15:2...	["2023-12-11 02:06:29",)

	adresseid integer	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_validite "IFT723".tsrange_sec
1	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2015-06-22 19:10:25","2016-06-22 19:15:25")
2	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2017-06-22 19:15:25","2018-06-22 19:10:25")

	adresseid integer	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_validite "IFT723".tsrange_sec	localisation_transaction "IFT723".tsrange_sec
1	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2015-06-22 19:10:...	["2023-12-11 02:05:58","2023-12-11 02:06:29")
2	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2015-06-22 19:10:...	["2023-12-11 02:06:29",)
3	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2017-06-22 19:15:...	["2023-12-11 02:06:29",)

La seconde situation peut être testé avec les commandes suivantes

```
    --Si des elements sont contenus dans le range
    --Ancien 1 :     ====
    --Ancien 2 :         =====
    --Effacer  :    ===========
    --Résultat :
CALL adresse_validite_ajout(1, '1', 'testRue', 'testVille', 'Qc',  'A1A 1A1', 'CA', '2015-06-22 19:10:25', '2016-06-22 19:10:25');
CALL adresse_validite_ajout(1, '1', 'testRue', 'testVille', 'Qc',  'A1A 1A1', 'CA', '2017-06-22 19:10:25', '2018-06-22 19:10:25');
CALL adresse_validite_effacer(1, '2014-06-22 19:15:25', '2019-06-22 19:15:25');
```

Le résultat de l'effacement est représenté dans les images suivantes :

	adresseid integer	adresseid_validite "IFT723".tsrange_sec

	adresseid integer	adresseid_validite "IFT723".tsrange_sec	adresseid_transaction "IFT723".tsrange_sec
1	1	["2015-06-22 19:10:25","2016-06-22 19:10:25")	["2023-12-11 02:09:38","2023-12-11 02:09:44")
2	1	["2017-06-22 19:10:25","2018-06-22 19:10:25")	["2023-12-11 02:09:38","2023-12-11 02:09:44")

	adresseid integer	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_validite "IFT723".tsrange_sec

	adresseid integer	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_validite "IFT723".tsrange_sec	localisation_transaction "IFT723".tsrange_sec
1	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2015-06-22 19:10:25","2016-06-22 19:10:25")	["2023-12-11 02:09:38","2023-12-11 02:09:44")
2	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2017-06-22 19:10:25","2018-06-22 19:10:25")	["2023-12-11 02:09:38","2023-12-11 02:09:44")

La troisième situation peut être testé avec les commandes suivantes

```
    --S'il y a chevauchement & ne s'étend pas sur la droite
    --Ancien   : =======
    --Effacer  :      =====
    --Résultat : =====
CALL adresse_validite_ajout(1, '1', 'testRue', 'testVille', 'Qc',  'A1A 1A1', 'CA', '2017-06-22 19:10:25', '2019-06-22 19:10:25');
CALL adresse_validite_effacer(1, '2018-06-22 19:15:25', '2020-06-22 19:15:25');
```

Le résultat de l'effacement est représenté dans les images suivantes :

	adresseid integer	adresseid_validite "IFT723".tsrange_sec
1	1	["2017-06-22 19:10:25","2018-06-22 19:15:25")

	adresseid integer	adresseid_validite "IFT723".tsrange_sec	adresseid_transaction "IFT723".tsrange_sec
1	1	["2017-06-22 19:10:25","2019-06-22 19:10:25")	["2023-12-11 02:11:35","2023-12-11 02:12:20")
2	1	["2017-06-22 19:10:25","2018-06-22 19:15:25")	["2023-12-11 02:12:20",)

	adresseid integer	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_validite "IFT723".tsrange_sec
1	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2017-06-22 19:10:25","2018-06-22 19:15:25")

	adresseid integer	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_validite "IFT723".tsrange_sec	localisation_transaction "IFT723".tsrange_sec
1	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2017-06-22 19:10:25","2019-06-22 19:10:25")	["2023-12-11 02:11:35","2023-12-11 02:12:20")
2	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2017-06-22 19:10:25","2018-06-22 19:15:25")	["2023-12-11 02:12:20",)

La quatrième situation peut être testé avec les commandes suivantes

```
    --S'il y a chevauchement & ne s'étend pas sur la gauche
    --Ancien   :    =======
    --Effacer  : =====
    --Résultat :      =====
CALL adresse_validite_ajout(1, '1', 'testRue', 'testVille', 'Qc',  'A1A 1A1', 'CA', '2017-06-22 19:10:25', '2019-06-22 19:10:25');
CALL adresse_validite_effacer(1, '2016-06-22 19:15:25', '2018-06-22 19:15:25');
```

Le résultat de l'effacement est représenté dans les images suivantes :

	adresseid integer	adresseid_validite "IFT723".tsrange_sec
1	1	["2018-06-22 19:15:25","2019-06-22 19:10:25")

	adresseid integer	adresseid_validite "IFT723".tsrange_sec	adresseid_transaction "IFT723".tsrange_sec
1	1	["2017-06-22 19:10:25","2019-06-22 19:10:25")	["2023-12-11 02:14:45","2023-12-11 02:14:51")
2	1	["2018-06-22 19:15:25","2019-06-22 19:10:25")	["2023-12-11 02:14:51",)

	adresseid integer	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_validite "IFT723".tsrange_sec
1	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2018-06-22 19:15:25","2019-06-22 19:10:25")

	adresseid integer	appartement integer	rue text	ville text	region text	code_postal text	pays text	localisation_validite "IFT723".tsrange_sec	localisation_transaction "IFT723".tsrange_sec
1	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2017-06-22 19:10:25","2019-06-22 19:10:25")	["2023-12-11 02:14:45","2023-12-11 02:14:51")
2	1	1	testRue	testVille	Qc	A1A 1A1	CA	["2018-06-22 19:15:25","2019-06-22 19:10:25")	["2023-12-11 02:14:51",)

Conception de l'API :

- **Modélisation de l'API :**

L'API (Interface de Programmation Applicative) d'une base de données est un ensemble de méthodes et de procédures accessibles par les développeurs pour interagir avec la base de données. Elle comprend des procédures stockées qui exécutent des opérations telles que la création, la modification, l'insertion et la suppression d'enregistrements pour différentes entités dans la base de données. Ces procédures stockées offrent une encapsulation des fonctionnalités spécifiques à chaque entité, ce qui signifie qu'elles regroupent les opérations associées à ces entités, favorisant ainsi la modularité et la gestion simplifiée du code. Par exemple, elles permettent de définir des fonctions spécifiques pour récupérer des informations ciblées à partir de la base de données. En rassemblant ces opérations liées à une entité spécifique au sein de procédures stockées et de fonctions, l'API facilite l'organisation du code, rend les opérations de la base de données plus accessibles et simplifie le processus de développement en fournissant des interfaces claires et spécifiques pour interagir avec la structure et les données de la base de données.

Annexe : 'IFT723_api.sql'

- **Note technique :**

Une note technique recensant l'ensemble des scripts et présentant leur utilisation typique dans le cadre des tests sert à fournir une documentation détaillée sur les différents scripts utilisés pour effectuer des tests. Cette note technique offre une vue d'ensemble exhaustive de tous les scripts, en décrivant précisément leur fonctionnement, leur objectif et leur utilisation spécifique dans le processus de test. Elle permet aux membres de l'équipe technique ou aux testeurs de comprendre et d'utiliser efficacement ces scripts pour

mener à bien les tests, en explicitant les scénarios, les conditions et les données pour lesquels ces scripts sont conçus.

Annexe : ‘Note_technique.md’

Rapport hebdomadaire

Du 02/11/2023 au 09/11/2023

Au courant de cette semaine, une étude de projet a été réalisé. Cette étude comprend la vision du projet, les exigences du projet et les besoins du projet, qui comprends les besoins en tant que tel, une étude d’opportunité et une étude de faisabilité. Toutes ces sections sont comprises dans la section d’avant-projet du présent document.

Du 09/11/2023 au 16/11/2023

Lors de cette semaine, la conception de la base de données a été réalisé. Pour ce faire, toutes les tables composant la base de données ont été modélisé en cinquième forme normale.

Du 16/11/2023 au 23/11/2023

Lors de cette semaine, la conception de la base de données temporalisé a été réalisé. Pour ce faire, les tables qui doivent être historicisé ont été transformé en sixième forme normale.

Du 23/11/2023 au 30/11/2023

Lors de cette semaine, la programmation de la base de données a été commencé. Du code pour les tables « Étudiant », « Adresse », « ItemFacture » et « Paiement » a été écrit.

Du 30/11/2023 au 07/12/2023

Lors de cette semaine, la programmation des tables « Étudiant » et « Adresse » a été complété. Ces deux tables comprennent toutes les fonctionnalités attendues de tables historicisées.

Les différents livrables ont été débuté : la spécification des exigences du modèle (SEM), la spécification de conception de la base de données (SCBD) et la note technique.

Après le 07/12/2023

Tous les livrables ont été rédigés : Le rapport de projet, la spécification des exigences du modèle (SEM), la spécification de conception de la base de données (SCBD), la note technique et le document d’information de l’équipe.

Bilan de fin de projet

Acquis de formation

Plusieurs notions ont été acquises lors de ce projet, principalement sur la bi-temporalisation des données dans une base de données.

Avant de temporaliser une table, elle doit être transformé pour être conforme à la cinquième forme normale, puis chaque attribut, ou groupe d'attribut, doit être isolé selon la sixième forme normale. Ces formes normales étaient de nouveaux concepts qui ont dû être appris, puis appliqué dans un contexte pratique.

Le concept de table courante et de tables de validité était aussi nouveau, ainsi que les liens qui existent entre ces deux catégories de tables.

Les tables de logs sont un concept très connu, mais nous n'avions pas eu l'opportunité de les implémenter ou de créer les automatismes (trigger) attachés à ces tables.

Finalement, l'un des acquis principaux que nous avons tous réalisé au courant de ce projet concernes les contraintes qui s'appliquent aux données de toutes les tables temporalisées, c'est-à-dire la non-redondance, la non-circonlocution, la non-contradiction et la compacité des données.

Perspective de perfectionnement

Plusieurs améliorations pourraient être apporté à la base de données réalisé et livré dans le cadre de ce projet. Une liste non-exhaustive sera présenté dans cette section.

- L'entièreté du système proposé dans le schéma de la cinquième forme normale pourrait être implémenté, ce qui pourrait permettre de valider la solution proposée à notre problème.
- Lors d'opération sur les tables de validité, il faudrait étendre les opérations sur la table courante afin d'assurer que les contraintes sur les données sont respectées. Par exemple, si une valeur ajoutée dans la table de validité est adjacente temporellement à une donnée présente dans la table courante, il faudrait modifier la table courante à la place d'insérer dans la table de validité. De même lors de l'effacement d'une période dans la table de validité.
- Des mesures de sécurité devrait être implémenté pour sécuriser les données. Pour ce faire, une encryption des données devrait être réalisé.
- Un API a été réalisé pour la base de données, mais pas d'interface graphique pour ce dernier. Cet ajout faciliterait l'utilisation de la base de données.
- Aucun test de performance n'a été réalisé lorsque beaucoup de données sont présente dans la base de données. Des tests avec quelques milliers/millions de données permettrait de valider que le design utilisé fonctionne correctement.

Partie 4 : Méthodologie et Processus

Le projet de création d'une base de données relationnelle pour la gestion des paiements des factures universitaires s'est appuyé sur une méthodologie rigoureuse visant à concevoir et à mettre en œuvre un système efficient, sécurisé et adapté aux besoins spécifiques de l'établissement.

Description du processus global : L'initiative a débuté par une analyse approfondie des besoins et des exigences de l'université en matière de gestion des paiements des factures. Cette phase préliminaire a permis d'identifier les principaux objectifs et de définir clairement les critères à remplir pour le succès du projet.

Étapes du projet : La méthodologie adoptée s'est articulée autour de plusieurs étapes distinctes. La première étape a consisté à modéliser les données et à concevoir un schéma relationnel répondant aux normes de la cinquième forme normale (5FN) pour assurer une gestion optimale des informations relatives aux paiements des factures.

Outils et technologies : Pour la modélisation de la base de données, nous avons utilisé des outils de conception et de normalisation spécifiques, tout en se basant sur PostgreSQL comme système de gestion de base de données.

Gestion des ressources : Nous avons géré efficacement les ressources en dédiant une équipe de développeurs et en planifiant les étapes du projet en fonction des contraintes de temps et de budget.

Problèmes rencontrés et solutions : Pendant le processus, nous avons fait face à des défis liés à la complexité de la modélisation des données. Ces défis ont été résolus grâce à une collaboration étroite entre les membres de l'équipe, une recherche approfondie et une itération attentive du schéma de base de données.

Rétrospective et leçons apprises : La phase de rétrospective a été cruciale pour identifier les aspects positifs et les axes d'amélioration. Les leçons apprises ont été intégrées dans nos processus pour des projets futurs, notamment en mettant l'accent sur une modélisation encore plus détaillée dès le début du projet.

Normalisation : Atteinte de la 5ème Forme Normale (5FN) pour le schéma de base de données afin d'assurer l'efficacité et la cohérence des données tout en réduisant les redondances.

Historisation : Évolution vers la 6ème Forme Normale (6FN) en mettant en place un système de log pour enregistrer un historique complet des modifications de données.

Élaboration d'API : Création d'une interface d'application pour interagir avec le schéma de base de données.

Utilisation : Utilisation de fonctions, procédures et déclencheurs (triggers) écrits en script SQL PostgreSQL pour manipuler et gérer les données.

Langage : Écriture de scripts SQL sous forme de PostgreSQL.

Jeu de données : Format CSV pour faciliter l'importation et l'utilisation dans PostgreSQL.

Documentation : Chaque script SQL minutieusement documenté dans un fichier.md pour assurer la traçabilité des opérations.

Meetings : Tenues via la plateforme MS Teams pour discuter des progrès, des défis et des décisions importantes.

Plateforme d'installation PostgreSQL : Utilisation de Docker pour simplifier la configuration et la gestion de l'environnement de base de données.

Méthodologie de travail : Nous avons opté pour une approche itérative, combinant des éléments des méthodologies Agile. Cela nous a permis de planifier le projet de manière progressive tout en répondant aux besoins évolutifs de l'université.

Conclusion

L'achèvement de ces jalons marque une étape cruciale dans notre projet de Base de Données et d'API, intégrant les solutions recommandées. La phase initiale a posé des bases solides en adaptant et mettant en œuvre la base de données. Le deuxième jalon a optimisé la base de données de la 5ème forme normale à la 6ème forme normale, intégrant nos solutions préconisées. Le troisième jalon a modelé, conçu et testé l'API en tirant parti des compétences et connaissances acquises du cours IFT723, incluant nos solutions pour une base de données temporalisée.

La rédaction des documents SEM et SCBD a pris en compte ces solutions pour répondre efficacement aux besoins du projet. Notre succès dépend de notre collaboration en tant qu'équipe, et nous avons atteint nos objectifs dans les délais impartis, intégrant les solutions comme des éléments clés.

En conclusion, ces jalons posent des bases solides pour notre BDT, en mettant en œuvre des solutions qui garantissent une base de données et une API optimales, répondant ainsi de manière efficace aux besoins spécifiques de notre projet.