

UNIVERSTE VIRTUELLE
DU BURKINA FASO

Parcours: Modélisation
Simulation et Calcul
Scientifique



université
virtuelle
Burkina ★ Faso

Rapport du Projet

Sujet 2 : Conception et Développement d'une Base de Données pour la Gestion des Opérations Commerciales et des Produits d'Assurance

Réalisé du 21 octobre 2024 au 5 décembre 2024

Les membres du groupe :

SEGDA Safiatou

ZONGO Celse Marius

Nom d'Enseignant en charge du projet :

Cheik Amed TRAORE

Année académique 2023-2024

Table des matières

Introduction.....	3
I. Présentation du projet.....	3
I.1 Problématique.....	3
I.2 Objectifs.....	3
Méthodologie.....	4
II. Phase de Réalisation.....	5
Planning de Réalisation (Diagramme de Gant).....	10
Bilan du Projet.....	10
Conclusion.....	10
Annexes.....	11

Introduction

Le secteur des assurances joue un rôle crucial dans l'économie moderne en fournissant des solutions pour gérer les risques et garantir la sécurité financière. Cependant, dans un environnement de plus en plus compétitif et dynamique, les compagnies d'assurance doivent constamment innover pour améliorer leurs processus internes et optimiser leurs opérations commerciales.

Au Burkina Faso, comme dans de nombreux pays en développement, les compagnies d'assurance rencontrent des défis majeurs dans la gestion des processus liés à la prospection des clients, la souscription des contrats, le suivi des paiements et le calcul des commissions des agents commerciaux. Ces processus sont souvent gérés de manière fragmentée, utilisant des systèmes obsolètes ou des méthodes manuelles, ce qui entraîne des inefficacités, des erreurs et une perte de temps.

Face à ces défis, une base de données relationnelle robuste et bien conçue peut fournir une solution efficace en centralisant les informations, en automatisant les tâches répétitives et en améliorant la prise de décision grâce à des analyses précises. Ce projet s'inscrit dans cette perspective et vise à concevoir un système de gestion optimisé pour une compagnie d'assurance.

I. Présentation du projet

Le projet repose sur l'utilisation des outils de modélisation UML et d'un système de gestion de bases de données relationnelles (Oracle Community). Il couvre l'ensemble des opérations commerciales, notamment la prospection, la gestion des souscriptions, et le suivi des performances des agents commerciaux.

I.1 Problématique

Les compagnies d'assurance opèrent dans un environnement où la gestion efficace des données est cruciale pour garantir l'efficacité des opérations et la satisfaction des clients. Actuellement, plusieurs défis majeurs se posent dans la gestion des processus commerciaux et des produits d'assurance.

I.2 Objectifs

Les objectifs principaux incluent :

- Modélisation UML :

Représenter les processus métier avec des diagrammes (cas d'utilisation, classes, séquences).

- Conception de la base de données :

Modèles conceptuel (MCD) et logique (MLD).

- Implémentation sous Oracle :

Création des tables, des vues, et des procédures stockées.

- Requêtes SQL avancées :

Interrogation des données et analyse des performances.

- Sécurité et intégrité :

Gestion des transactions et des accès.

- Optimisation et Reporting :

Analyse des performances et création de rapport automatisé.

II. Méthodologie

La méthodologie adoptée pour ce projet comprend les étapes suivantes :

- Modélisation UML

Création du diagramme de cas d'utilisation pour illustrer les interactions.

Élaboration du diagramme de classes pour définir les entités et leurs relations.

Conception du diagramme de séquence pour modéliser les scénarios de souscription.

- Conception de la Base de Données

Conception du Modèle Conceptuel de Données (MCD) à l'aide des diagrammes entité-association.

Traduction en Modèle Logique de Données (MLD) pour formaliser les tables, les relations, et les cardinalités.

Identification des clés primaires, des clés étrangères, et des contraintes d'intégrité.

- Implémentation

Création des tables, relations, vues, et index dans Oracle.

Écriture des procédures stockées pour automatiser des tâches (ex. calcul des commissions).

Tests unitaires pour valider les contraintes d'intégrité et les opérations CRUD.

- Écriture des Requêtes SQL Avancées

Rédiger des requêtes pour interroger les souscriptions, calculer les commissions, et générer des rapports.

Optimiser les requêtes avec des techniques comme l'indexation.

- Gestion des Transactions et Sécurité

Implémentation de mécanismes de transactions respectant les propriétés ACID.

Mise en place d'un contrôle d'accès basé sur les rôles des utilisateurs.

Configuration des politiques de sauvegarde et restauration.

- Tests et Validation

Test des fonctionnalités principales : souscription, paiements, calculs de commissions.

Simulation de scénarios réels pour valider les performances.

Résolution des anomalies identifiées.

III. Phase de Réalisation

La phase de réalisation est une phase qui consiste à concrétiser les spécifications définies dans les phases précédentes. Cette étape est divisée en plusieurs sous-phases, chacune répondant à un objectif spécifique. Voici une description détaillée de la phase de réalisation pour ce projet.

Modélisation UML

La modélisation UML (Unified Modeling Language) a pour objectif de définir la structure et le fonctionnement du système. Elle permet de clarifier les interactions entre les acteurs et le système, ainsi que les processus métiers à automatiser.

Diagramme de cas d'utilisation :

- Décrit les interactions entre les utilisateurs (agents commerciaux, clients, administrateurs) et le système.
- Principaux cas d'utilisation :
 1. Gestion des clients.
 2. Souscription d'un produit d'assurance.
 3. Calcul des commissions des agents commerciaux.
 4. Génération de rapports.

Diagramme de classes :

- Représente les principales entités du système et leurs relations, telles que :
 - Clients : identifiants, noms, adresses, etc.
 - Produits d'assurance : type, montant de la prime, durée, etc.
 - Souscriptions : date, statut, montant, etc.
 - Commissions : pourcentage, montant, date de paiement, etc.

Diagramme de séquence :

- Modélise les interactions dynamiques entre les acteurs et le système pour des scénarios spécifiques.
- Exemple : processus complet de souscription à un produit d'assurance, depuis la demande d'un client jusqu'à la génération des commissions.

Conception de la Base de Données

La conception de la base de données s'articule autour des modèles conceptuel et logique.

- Modèle conceptuel (MCD)

Le modèle conceptuel nous permet d'identifier les entités principales et leurs relations. Voici les principales entités : Client, Commercial, Produit d'assurance, Souscription, Paiement, Commission.

- Modèle logique (MLD)

Le modèle logique traduit le modèle conceptuel en structures de base de données relationnelle avec définition des tables, colonnes, clés primaires et clés étrangères.

Gestion des relations et contraintes d'intégrité

Cardinalités

- Clients (1) --- (N) Souscriptions
- Commerciaux (1) --- (N) Commissions
- Produits d'assurance (1) --- (N) Souscriptions
- Souscriptions (1) --- (1) Paiements

Contraintes d'intégrité

1. Clés primaires (PK) : Chaque table a une clé primaire qui identifie de manière unique chaque enregistrement.
2. Clés étrangères (FK) :
 - Dans la table Souscriptions, id_client fait référence à id_client dans Clients.
 - Dans la table Souscriptions, id_produit fait référence à id_produit dans Produits_Assurance.
 - Dans la table Paiements, id_souscription fait référence à id_souscription dans Souscriptions.
 - Dans la table Commissions, id_commercial fait référence à id_commercial dans Commerciaux et id_souscription fait référence à id_souscription dans Souscriptions.
3. Contraintes NOT NULL : Les attributs essentiels (comme les identifiants et les montants) ne peuvent pas être nuls.

4. Contraintes d'unicité : Par exemple, l'email des clients et des commerciaux peut avoir une contrainte d'unicité.

Implémentation dans Oracle

L'implémentation dans Oracle constitue une étape essentielle pour concrétiser le modèle UML en une base de données relationnelle robuste.

Les tâches réalisées dans cette phase incluent :

- Traduction du modèle UML : Les diagrammes de classes et les relations conçus dans UML ont été convertis en une structure relationnelle dans Oracle. Chaque entité UML (par exemple, Clients, Commerciaux, Produits d'assurance, Souscriptions) a été traduite en tables.
- Création des tables : Les tables ont été définies avec des types de données appropriés et des contraintes (clés primaires, clés étrangères, et contraintes d'intégrité) pour garantir la cohérence des données.
- Création de vues et procédures stockées :
 - Vues : Simplifient les requêtes en rassemblant les données complexes dans des ensembles lisibles (exemple : vue pour les souscriptions actives).
 - Procédures stockées : Automatisent les processus tels que le calcul des commissions et la génération de rapports.

Écriture de Requêtes SQL Avancées

L'analyse et l'exploitation des données sont facilitées par des requêtes SQL avancées développées pour répondre à des besoins spécifiques :

- Interrogation des souscriptions d'un client spécifique : Cette requête identifie les contrats d'assurance souscrits par un client donné, en indiquant les détails des produits, les montants, et les échéances.
- Calcul des commissions des commerciaux : Basée sur les primes souscrites par chaque agent commercial, une requête spécifique calcule le pourcentage de commission selon les règles définies dans les procédures stockées.
- Génération de rapports de performance : Une requête regroupe les ventes par commercial et produit pour fournir un rapport détaillé des performances individuelles ou d'équipe.
- Analyse des statistiques de vente : Requêtes permettant d'évaluer les tendances des ventes, comme le produit le plus vendu ou le segment de clientèle le plus actif.

Gestion des Transactions et de la Sécurité

La fiabilité et la sécurité des données sont fondamentales dans un système de gestion des opérations commerciales. Les mécanismes suivants ont été mis en place :

Gestion des transactions (ACID) :

- Atomicité : Chaque transaction (par exemple, l'ajout d'un paiement ou la mise à jour d'une souscription) est considérée comme une opération unique, réussissant ou échouant intégralement.
- Cohérence : Les données respectent les contraintes définies dans la base (exemple : aucun paiement n'est lié à un produit inexistant).
- Isolation : Les transactions concurrentes n'interfèrent pas les unes avec les autres, garantissant ainsi une intégrité parfaite.
- Durabilité : Une fois validée, une transaction est enregistrée de manière permanente, même en cas de panne.

Sécurité des données :

- Mise en place d'un système de contrôle d'accès basé sur les rôles :
 - Administrateurs : Accès complet à toutes les données et fonctions.
 - Commerciaux : Accès limité à leurs propres performances et données clients.
 - Enregistrement des actions sensibles dans un journal d'audit pour assurer la traçabilité.
-

Contrôle des Accès Utilisateurs

Le contrôle des accès est une mesure essentielle pour sécuriser les données et limiter les droits des utilisateurs selon leurs rôles. Ce système garantit que seules les personnes autorisées accèdent aux informations ou exécutent des actions spécifiques dans le système.

Rôles d'Utilisateurs et leurs Droits

- Administrateur :
 - Accès complet à toutes les fonctionnalités du système.
 - Gestion des utilisateurs (ajout, suppression, modification de droits).
 - Visualisation des rapports globaux (souscriptions, commissions, paiements, etc.).
- Commercial :
 - Accès limité à ses propres données de performance.
 - Consultation des souscriptions qu'il a réalisées.
 - Visualisation de ses commissions calculées.
- Agent de Souscription :
 - Gestion des données de souscription pour les clients.
 - Consultation des paiements et renouvellements.

- Pas d'accès aux statistiques globales ou au calcul des commissions.

Analyse et Optimisation des Requêtes SQL

Pour garantir une performance optimale des requêtes SQL et faciliter le reporting au sein d'une base de données pour la gestion des opérations d'une compagnie d'assurance.

Analyse et Optimisation des Performances des Requêtes SQL

- Création d'Index : Les index sont cruciaux pour améliorer les performances des recherches fréquentes. Voici comment créer des index sur les colonnes pertinentes.
- Statistiques pour l'Optimiseur : L'utilisation de statistiques permet à l'optimiseur de requêtes de choisir le meilleur plan d'exécution.

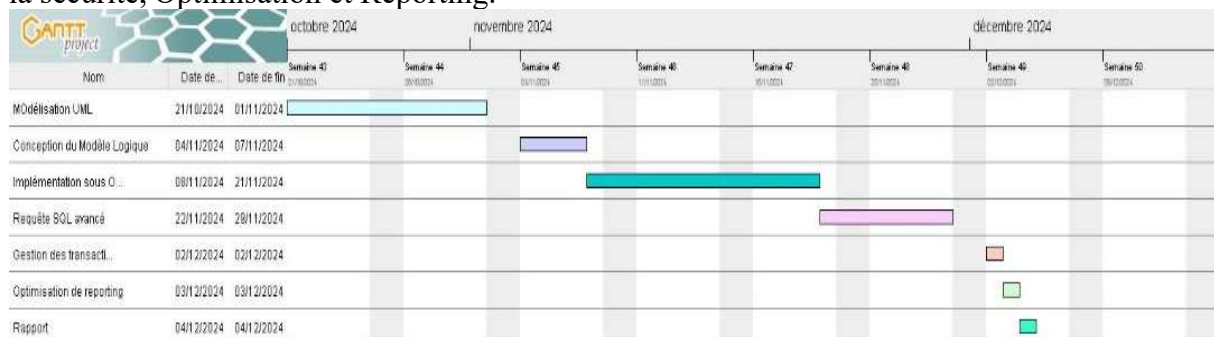
Création de Rapports Automatisés

- Procédure pour Générer un Rapport Mensuel : La procédure suivante permet de générer un rapport mensuel des souscriptions. Elle utilise un curseur référencé (Ref Cursor) pour retourner les résultats.
- Traitement du Rapport : Pour traiter le rapport généré par la procédure, vous pouvez l'exporter vers un fichier, l'afficher dans une interface utilisateur, ou l'envoyer par email, selon vos besoins.

La phase de réalisation du projet a permis de produire une base de données robuste, sécurisée, et performante, répondant aux besoins identifiés dans l'analyse. Les objectifs fixés, tels que la modélisation, l'implémentation, la gestion des accès, et la génération de rapports, ont été atteints avec succès. Les résultats des différentes étapes se représentera dans Annexe.

Planning du projet (diagramme Gant)

Le diagramme de Gant ci-dessous représente les différentes étapes du notre projet, comprenant la Modélisation UML, Conception de la Base de Données Relationnelle, Implémentation dans Oracle, Écriture de requêtes SQL avancées, Gestion des transactions et de la sécurité, Optimisation et Reporting.



Bilan du projet

Ce projet visait à concevoir et développer une solution complète de gestion des opérations pour une compagnie d'assurance, centrée sur une base de données relationnelle performante. Malgré quelques difficultés rencontrées notamment Optimisation des requêtes et la gestion des transaction simultanées, à travers une approche structurée, les objectifs de modélisation UML, de conception relationnelle, d'implémentation sous Oracle (ou PostgreSQL en option), et de sécurisation des données ont été atteints.

Conclusion

Le développement de cette base de données pour une compagnie d'assurance représente une avancée stratégique majeure dans la gestion des opérations commerciales et administratives. Grâce à une modélisation UML rigoureuse, une conception relationnelle robuste, et une implémentation optimisée sous Oracle ou PostgreSQL, le projet répond efficacement aux besoins d'organisation, d'automatisation, et de sécurité des données.

La solution développée apporte des bénéfices immédiats en termes de fiabilité, de performance, et de réduction des tâches manuelles, tout en offrant des outils d'analyse avancés pour soutenir la prise de décision stratégique. Les mécanismes de gestion des transactions garantissent l'intégrité des données, tandis que les contrôles d'accès protègent les informations sensibles, assurant ainsi la conformité aux standards de sécurité modernes.

En résumé, ce projet constitue une base solide pour les opérations de la compagnie d'assurance et peut évoluer vers une solution complète en intégrant des outils d'intelligence d'affaires ou d'autres systèmes connexes. Il s'agit d'un pas décisif vers la modernisation et l'efficacité organisationnelle, posant les jalons pour une croissance durable et compétitive.

Annexe

Modélisation UML

Diagramme de cas d'Utilisation En utilisant plaintext en ligne.

Décrit les interactions entre les utilisateurs (agents commerciaux, clients, administrateurs) et le système.

Résultat du code

@startuml

actor Client

actor Commercial

Client --> (Consulter les produits d'assurance)

Client --> (Souscrire à un produit d'assurance)

Client --> (Effectuer un paiement)

Commercial --> (Gérer les souscriptions)

Commercial --> (Calculer les commissions)

(Consulter les produits d'assurance) --> (Souscrire à un produit d'assurance)

(Souscrire à un produit d'assurance) --> (Effectuer un paiement)

@enduml

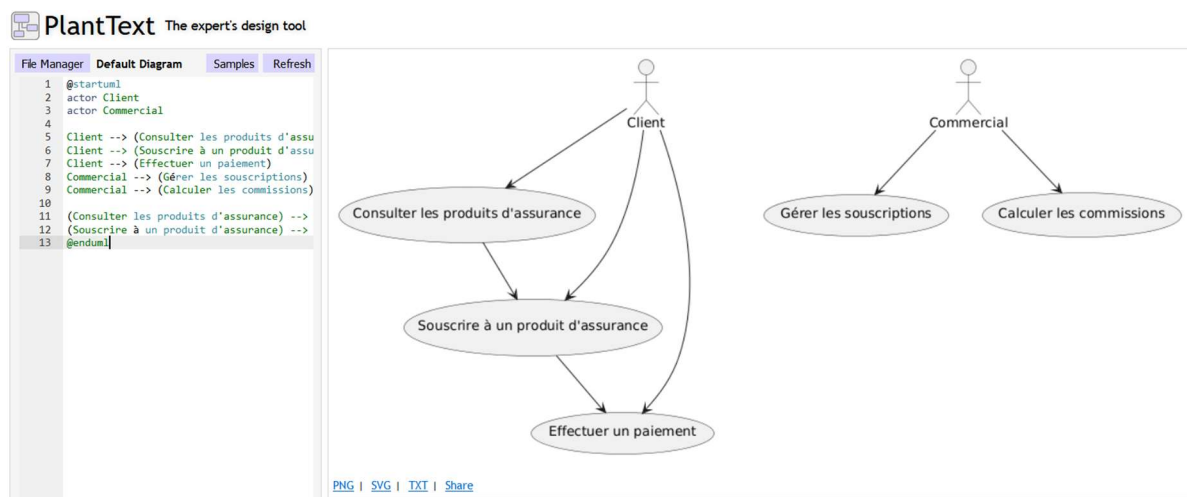


Diagramme de classe

Code :

@startuml

```
class Client {
    +id : int
    +nom : string
    +adresse : string
    +telephone : string
}
```

```
class Commercial {
    +id : int
    +nom : string
    +commission : float
}
```

```
}
```

```
class ProduitAssurance {
```

```
    +id : int
```

```
    +nom : string
```

```
    +type : string
```

```
    +prix : float
```

```
}
```

```
class Souscription {
```

```
    +id : int
```

```
    +dateSouscription : date
```

```
    +statut : string
```

```
}
```

```
class Paiement {
```

```
    +id : int
```

```
    +montant : float
```

```
    +date : date
```

```
}
```

```
class Commission {
```

```
    +id : int
```

```
    +montant : float
```

```
    +date : date
```

```
}
```

```
Client "1" -- "0..*" Souscription
```

```
Commercial "1" -- "0..*" Commission
```

```
ProduitAssurance "1" -- "0..*" Souscription
```

Souscription "1" -- "1" Paiement

@enduml

Resultat du code

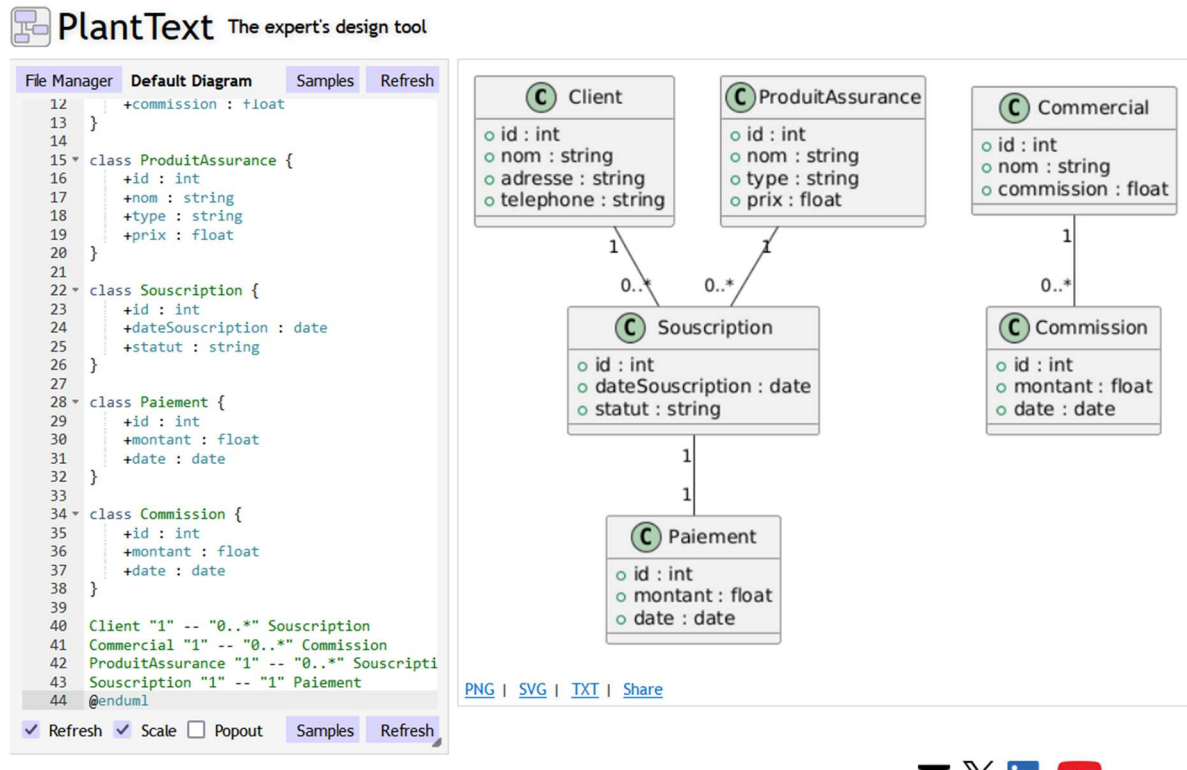


Diagramme de séquence

Code :

@startuml

actor Client

participant Système

participant ProduitAssurance

Client -> Système : Consulter les produits d'assurance

Système -> Client : Liste des produits

Client -> Système : Choisir un produit

Système -> ProduitAssurance : Créer une souscription

Système -> Client : Demander informations de paiement

Client -> Système : Fournir informations de paiement

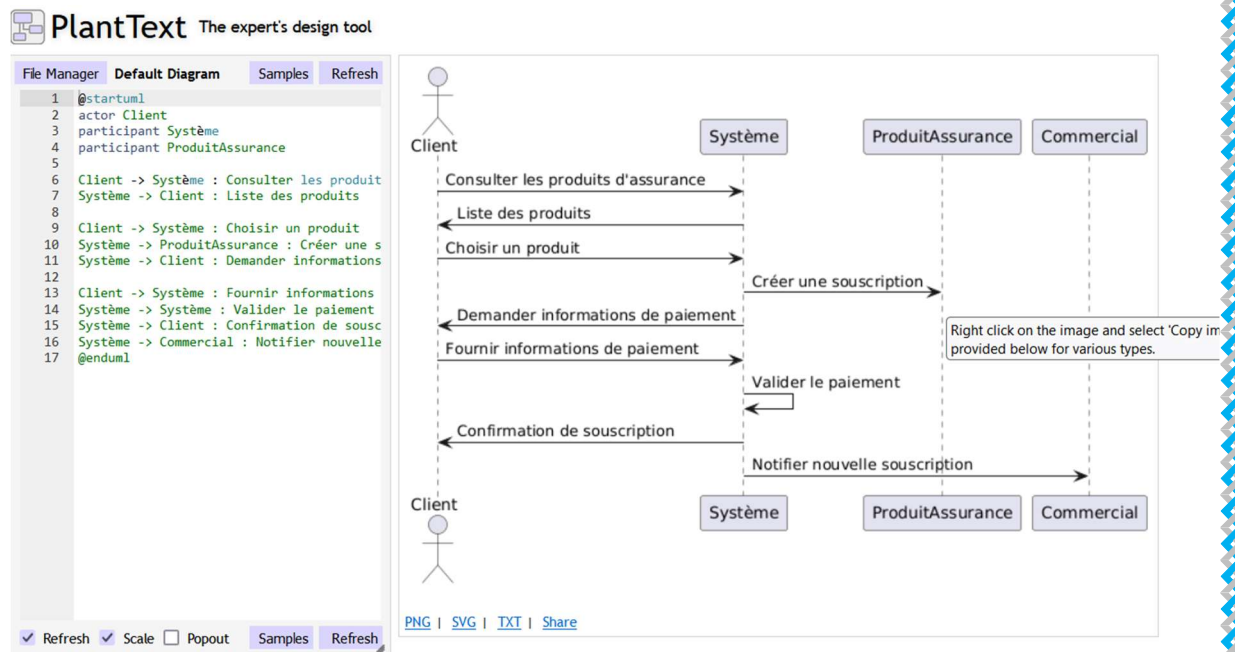
Système -> Système : Valider le paiement

Système -> Client : Confirmation de souscription

Système -> Commercial : Notifier nouvelle souscription

@enduml

Resultat de code :



Conception de la Base de Données Relationnelle

Modèle conceptuelle

[Client] --(1,N)--> [Souscription] --(1,1)--> [Paiement]

|

|

(1,N)

|

[Commercial] --(1,N)--> [Commission]

|

|

(1,N)

|
[Produit] --(1,N)--> [Souscription]

Modèle logique

Entités et Attributs

Voici les entités identifiées avec leurs attributs, types de données, et relations :

1. Client

- **idClient**: INT (Clé primaire, auto-incrément)
- **nom**: VARCHAR(100)
- **prenom**: VARCHAR(100)
- **email**: VARCHAR(150)
- **telephone**: VARCHAR(15)

2. Produit

- **idProduit**: INT (Clé primaire, auto-incrément)
- **nomProduit**: VARCHAR(100)
- **type**: VARCHAR(50)
- **prix**: DECIMAL(10, 2)

3. Souscription

- **idSouscription**: INT (Clé primaire, auto-incrément)
- **idClient**: INT (Clé étrangère vers Client)
- **idProduit**: INT (Clé étrangère vers Produit)
- **dateSouscription**: DATE
- **montant**: DECIMAL(10, 2)

4. Commercial

- **idCommercial**: INT (Clé primaire, auto-incrément)
- **nom**: VARCHAR(100)
- **prenom**: VARCHAR(100)
- **email**: VARCHAR(150)

5. Commission

- **idCommission**: INT (Clé primaire, auto-incrément)

- **idCommercial**: INT (Clé étrangère vers Commercial)
- **idSouscription**: INT (Clé étrangère vers Souscription)
- **montantCommission**: DECIMAL(10, 2)
- **dateCommission**: DATE

6. Paiement

- **idPaiement**: INT (Clé primaire, auto-incrément)
- **idSouscription**: INT (Clé étrangère vers Souscription)
- **montant**: DECIMAL(10, 2)
- **datePaiement**: DATE

Définition des tables :

CREATE TABLE Client (

idClient INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY, -- Clé primaire auto-incrémentée

nom VARCHAR(100) NOT NULL, -- Nom du client

prenom VARCHAR(100) NOT NULL, -- Prénom du client

email VARCHAR(150) UNIQUE NOT NULL, -- Email unique

telephone VARCHAR(15) -- Numéro de téléphone

);

CREATE TABLE Commercial (

idCommercial INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY, -- Clé primaire auto-incrémentée

nom VARCHAR(100) NOT NULL, -- Nom du commercial

prenom VARCHAR(100) NOT NULL, -- Prénom du commercial

email VARCHAR(150) UNIQUE NOT NULL -- Email unique

);

CREATE TABLE Produit (

idProduit INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY, -- Clé primaire auto-incrémentée

nomProduit VARCHAR(100) NOT NULL, -- Nom du produit

type VARCHAR(50) NOT NULL, -- Type d'assurance (ex: vie, santé,
etc.)

prix DECIMAL(10, 2) NOT NULL -- Prix du produit

);

CREATE TABLE Souscription (

idSouscription INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY, --
Clé primaire auto-incrémentée

idClient INT NOT NULL, -- Clé étrangère vers Client

idProduit INT NOT NULL, -- Clé étrangère vers Produit

dateSouscription DATE NOT NULL, -- Date de souscription

montant DECIMAL(10, 2) NOT NULL, -- Montant de la souscription

CONSTRAINT fk_client FOREIGN KEY (idClient) REFERENCES Client(idClient) ON
DELETE CASCADE, -- Relation avec Client

CONSTRAINT fk_produit FOREIGN KEY (idProduit) REFERENCES Produit(idProduit)
ON DELETE CASCADE -- Relation avec Produit

);

CREATE TABLE Souscription (

idSouscription INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY, --
Clé primaire auto-incrémentée

idClient INT NOT NULL, -- Clé étrangère vers Client

idProduit INT NOT NULL, -- Clé étrangère vers Produit

dateSouscription DATE NOT NULL, -- Date de souscription

montant DECIMAL(10, 2) NOT NULL, -- Montant de la souscription

CONSTRAINT fk_client FOREIGN KEY (idClient) REFERENCES Client(idClient) ON
DELETE CASCADE, -- Relation avec Client

CONSTRAINT fk_produit FOREIGN KEY (idProduit) REFERENCES Produit(idProduit)
ON DELETE CASCADE -- Relation avec Produit

);

CREATE TABLE Paiement (

idPaiement INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY, -- Clé
primaire auto-incrémentée

idSouscription INT NOT NULL, -- Clé étrangère vers Souscription

montant DECIMAL(10, 2) NOT NULL, -- Montant du paiement

datePaiement DATE NOT NULL, -- Date du paiement

CONSTRAINT fk_souscription FOREIGN KEY (idSouscription) REFERENCES
Souscription(idSouscription) ON DELETE CASCADE -- Relation avec Souscription

);

```
CREATE TABLE Commission (
```

```
    idCommission INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY, --  
    Clé primaire auto-incrémentée
```

```
    idCommercial INT NOT NULL, -- Clé étrangère vers Commercial
```

```
    idSouscription INT NOT NULL, -- Clé étrangère vers Souscription
```

```
    montantCommission DECIMAL(10, 2) NOT NULL, -- Montant de la  
    commission
```

```
    dateCommission DATE NOT NULL, -- Date de la commission
```

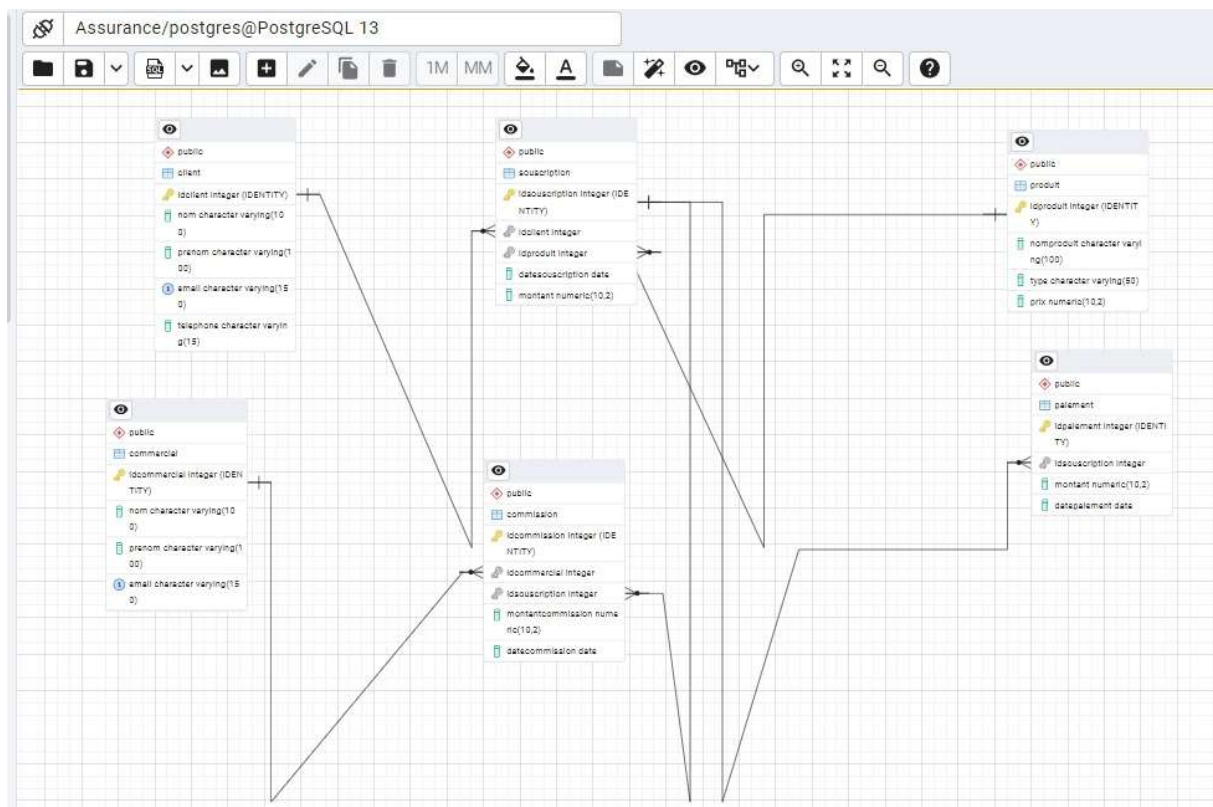
```
    CONSTRAINT fk_commercial FOREIGN KEY (idCommercial) REFERENCES  
    Commercial(idCommercial) ON DELETE CASCADE, -- Relation avec Commercial
```

```
    CONSTRAINT fk_souscription_commission FOREIGN KEY (idSouscription)  
    REFERENCES Souscription(idSouscription) ON DELETE CASCADE -- Relation avec  
    Souscription
```

);

Implémentation dans Oracle (PostgreSQL)

Traduction du modèle UML en une base de données relationnelle sous Oracle.



-- Création de la base de données

```
CREATE DATABASE AssuranceDB;
```

```
USE AssuranceDB;
```

```
CREATE TABLE Client (
```

```
    idClient INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
```

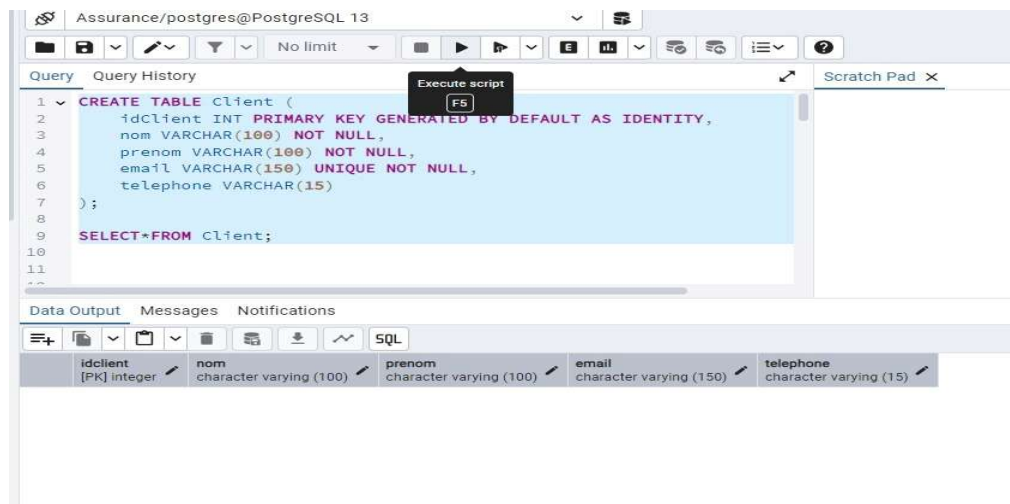
```
    nom VARCHAR(100) NOT NULL,
```

```
    prenom VARCHAR(100) NOT NULL,
```

```
    email VARCHAR(150) UNIQUE NOT NULL,
```

```
    telephone VARCHAR(15)
```

```
);
```



```
CREATE TABLE Commercial (
```

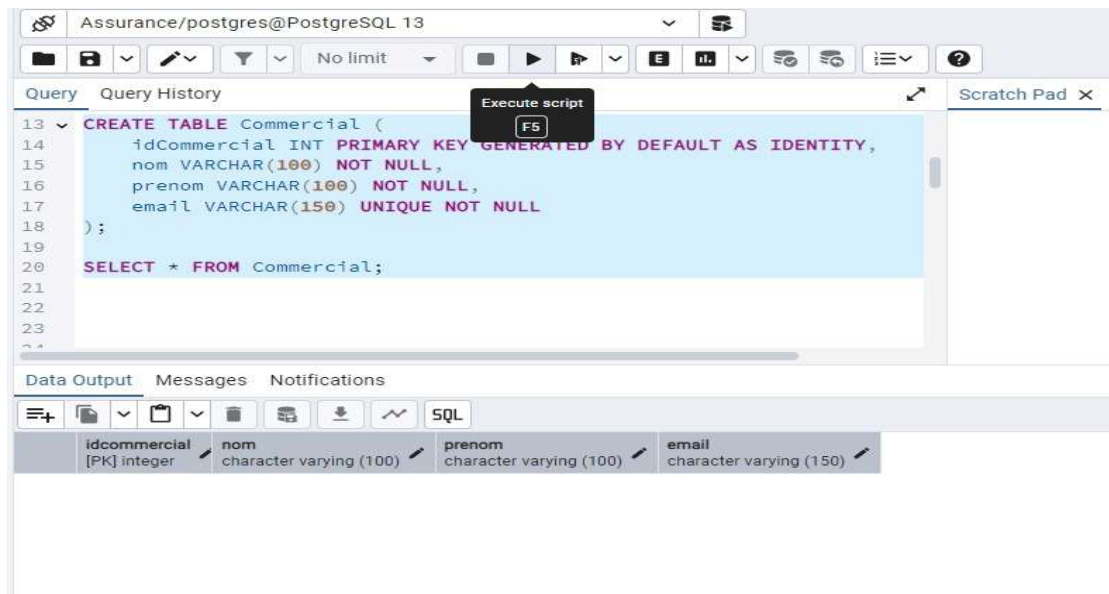
```
    idCommercial INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
```

```
    nom VARCHAR(100) NOT NULL,
```

```
    prenom VARCHAR(100) NOT NULL,
```

```
    email VARCHAR(150) UNIQUE NOT NULL
```

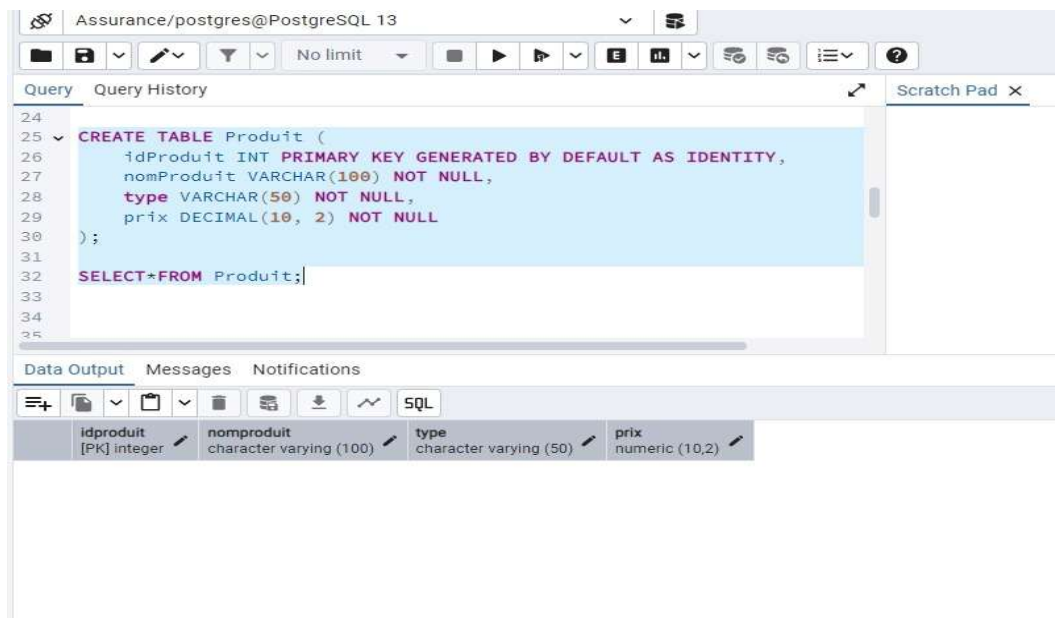
```
);
```



```

CREATE TABLE Produit (
    idProduit INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
    nomProduit VARCHAR(100) NOT NULL,
    type VARCHAR(50) NOT NULL,
    prix DECIMAL(10, 2) NOT NULL
);

```



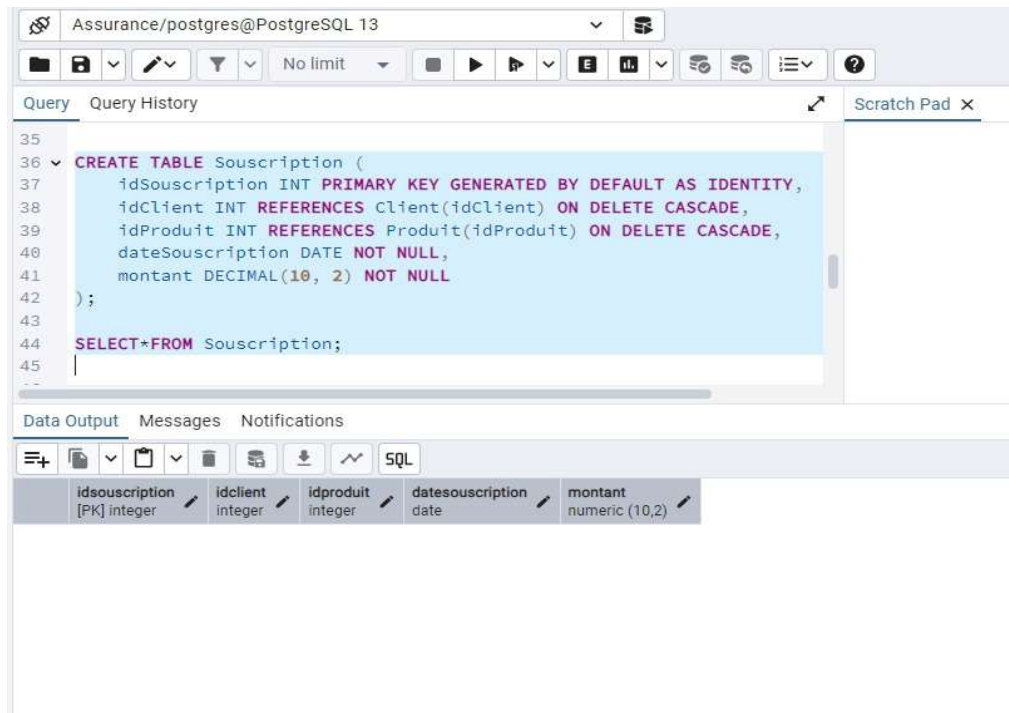
```

CREATE TABLE Souscription (
    idSouscription INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
    idClient INT REFERENCES Client(idClient) ON DELETE CASCADE,
    idProduit INT REFERENCES Produit(idProduit) ON DELETE CASCADE,

```



```
dateSouscription DATE NOT NULL,  
montant DECIMAL(10, 2) NOT NULL  
);
```



The screenshot shows a PostgreSQL query editor interface. The top bar indicates the connection is 'Assurance/postgres@PostgreSQL 13'. Below the toolbar, the 'Query' tab is active, displaying the following SQL code:

```
35  
36 CREATE TABLE Souscription (  
37     idSouscription INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,  
38     idClient INT REFERENCES Client(idClient) ON DELETE CASCADE,  
39     idProduit INT REFERENCES Produit(idProduit) ON DELETE CASCADE,  
40     dateSouscription DATE NOT NULL,  
41     montant DECIMAL(10, 2) NOT NULL  
42 );  
43  
44 SELECT * FROM Souscription;  
45
```

Below the query editor, the 'Data Output' tab is visible, showing the table structure for 'Souscription':

idsouscription	idclient	idproduit	datesouscription	montant
[PK] integer	integer	integer	date	numeric (10,2)

```
CREATE TABLE Paiement (  
    idPaiement INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,  
    idSouscription INT REFERENCES Souscription(idSouscription) ON DELETE  
CASCADE,  
    montant DECIMAL(10, 2) NOT NULL,  
    datePaiement DATE NOT NULL  
);
```

```

46
47 CREATE TABLE Paielement (
48     idPaielement INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
49     idSouscription INT REFERENCES Souscription(idSouscription) ON DELETE CASCADE,
50     montant DECIMAL(10, 2) NOT NULL,
51     datePaielement DATE NOT NULL
52 );
53
54 SELECT * FROM Paielement;
55
56

```

idpaielement	idsouscription	montant	datepaielement
[PK] integer	integer	numeric (10,2)	date

CREATE TABLE Commission (

idCommission INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,

idCommercial INT REFERENCES Commercial(idCommercial) ON DELETE CASCADE,

idSouscription INT REFERENCES Souscription(idSouscription) ON DELETE CASCADE,

montantCommission DECIMAL(10, 2) NOT NULL,

dateCommission DATE NOT NULL

);

```

59 CREATE TABLE Commission (
60     idCommission INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
61     idCommercial INT REFERENCES Commercial(idCommercial) ON DELETE CASCADE,
62     idSouscription INT REFERENCES Souscription(idSouscription) ON DELETE CASCADE,
63     montantCommission DECIMAL(10, 2) NOT NULL,
64     dateCommission DATE NOT NULL
65 );
66
67 SELECT * FROM Commission;
68
69

```

idcommission	idcommercial	idsouscription	montantcommission	datecommission
[PK] integer	integer	integer	numeric (10,2)	date

Vues et Procédures Stockées

- Vue

```
CREATE VIEW Vue_Souscriptions AS
```

```
SELECT
```

```
    s.idSouscription,
    c.nom AS nomClient,
    c.prenom AS prenomClient,
    p.nomProduit,
    s.dateSouscription,
    s.montant
```

```
FROM
```

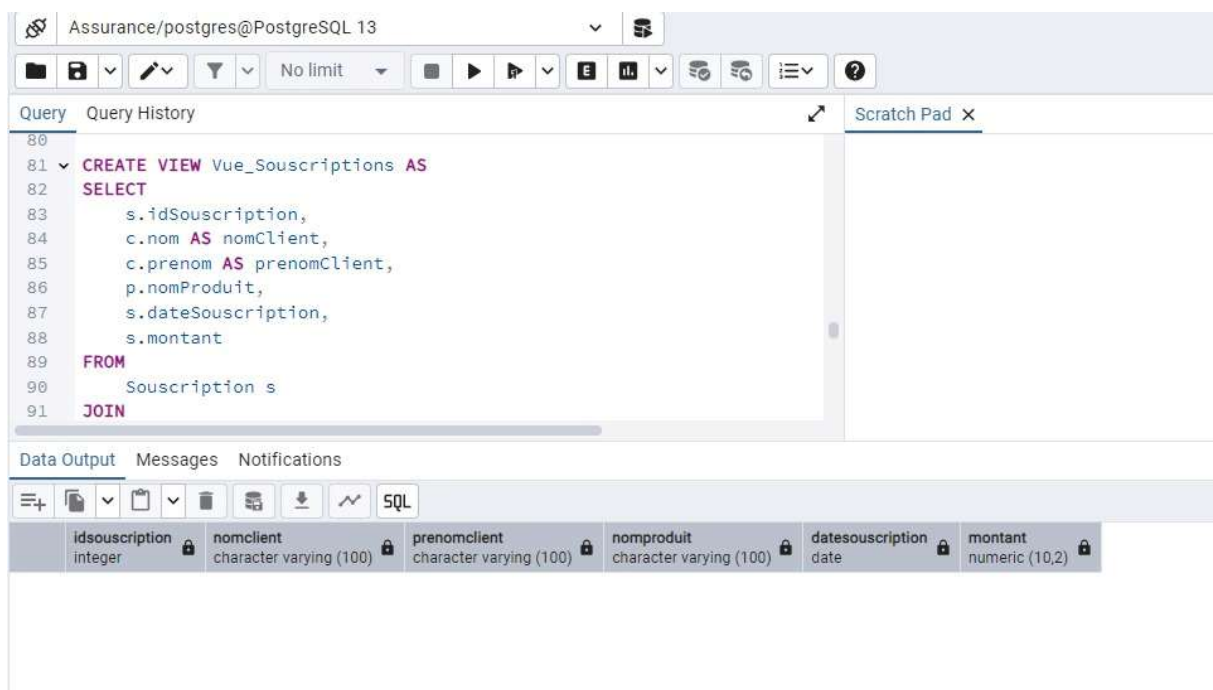
```
    Souscription s
```

```
JOIN
```

```
    Client c ON s.idClient = c.idClient
```

```
JOIN
```

```
    Produit p ON s.idProduit = p.idProduit;
```



Procédure Stocké

```
CREATE OR REPLACE PROCEDURE EnregistrerSouscription (
```

```
    p_idClient IN INT,
```

```
    p_idProduit IN INT,
```

```

    p_dateSouscription IN DATE,
    p_montant IN DECIMAL
) AS
    v_idSouscription INT;
BEGIN
    INSERT INTO Souscription (idClient, idProduit, dateSouscription, montant)
    VALUES (p_idClient, p_idProduit, p_dateSouscription, p_montant)
    RETURNING idSouscription INTO v_idSouscription;

    INSERT INTO Commission (idCommercial, idSouscription, montantCommission,
    dateCommission)
    VALUES (1, v_idSouscription, p_montant * 0.10, SYSDATE); -- Exemple de calcul de
    commission

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;

        RAISE_APPLICATION_ERROR(-20001, 'Erreur lors de l'enregistrement de la
    souscription: ' || SQLERRM);
END EnregistrerSouscription;
);

```

Écriture de Requêtes SQL Avancées

- **Interroger les Souscriptions d'un Client Spécifique**

```

SELECT
    s.idSouscription,
    p.nomProduit,
    s.dateSouscription,
    s.montant
FROM

```

Souscription s

JOIN

Client c ON s.idClient = c.idClient

JOIN

Produit p ON s.idProduit = p.idProduit

WHERE

c.idClient = :client_id; -- Remplacez :client_id par l'ID du client souhaité

- **Calculer les Commissions des Commerciaux**

SELECT

com.idCommercial,

com.nom,

com.prenom,

SUM(co.montantCommission) AS totalCommission

FROM

Commission co

JOIN

Commercial com ON co.idCommercial = com.idCommercial

GROUP BY

com.idCommercial, com.nom, com.prenom;

- **Générer des Rapports de Performance des Agents Commerciaux**

SELECT

p.idProduit,

p.nomProduit,

COUNT(s.idSouscription) AS nombreVentes,

SUM(s.montant) AS totalVentes

FROM

Produit p

LEFT JOIN

Souscription s ON p.idProduit = s.idProduit

GROUP BY

p.idProduit, p.nomProduit;

- **Analyser les Statistiques de Ventes de Produits d'Assurance**

SELECT

p.idProduit,

p.nomProduit,

COUNT(s.idSouscription) AS nombreVentes,

SUM(s.montant) AS totalVentes

FROM

Produit p

LEFT JOIN

Souscription s ON p.idProduit = s.idProduit

GROUP BY

p.idProduit, p.nomProduit;

Gestion des Transactions et de la Sécurité

Implémentation des mécanisme de gestion des transaction (ACID) pour garantir l'intégrité des opérations.

- Atomicité

CREATE OR REPLACE PROCEDURE EnregistrerSouscription (

p_idClient IN INT,

p_idProduit IN INT,

p_dateSouscription IN DATE,

p_montant IN DECIMAL

) AS

v_idSouscription INT;

BEGIN

-- Démarrer la transaction

BEGIN

-- Insérer la souscription

```

INSERT INTO Souscription (idClient, idProduit, dateSouscription, montant)
VALUES (p_idClient, p_idProduit, p_dateSouscription, p_montant)
RETURNING idSouscription INTO v_idSouscription;

-- Insérer la commission

INSERT INTO Commission (idCommercial, idSouscription, montantCommission,
dateCommission)
VALUES (1, v_idSouscription, p_montant * 0.10, SYSDATE);

-- Valider la transaction

COMMIT;

EXCEPTION

WHEN OTHERS THEN

    ROLLBACK; -- Annuler toutes les modifications en cas d'erreur

    RAISE_APPLICATION_ERROR(-20001, 'Erreur lors de l'enregistrement de la
souscription: ' || SQLERRM);

END;

END EnregistrerSouscription;

/

```

- Cohérence

```
ALTER TABLE Souscription
```

```
ADD CONSTRAINT fk_client FOREIGN KEY (idClient) REFERENCES Client(idClient);
```

- Isolation

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
```

```
BEGIN
```

```
-- Opérations de transaction ici
```

```
COMMIT;
```

```
END ;
```

- Durabilité

BEGIN

-- Opérations ici

COMMIT; -- Les modifications sont enregistrées de manière permanente

END ;

Gestion des Rôles

- Contrôle des Accès Utilisateurs

-- Création de rôles

CREATE ROLE administrateur;

CREATE ROLE commercial;

CREATE ROLE agent;

-- Attribution de privilèges

GRANT SELECT, INSERT, UPDATE, DELETE ON Client TO administrateur;

GRANT SELECT, INSERT ON Souscription TO commercial;

-- Attribution de rôles à des utilisateurs

GRANT administrateur TO user1;

GRANT commercial TO user2;

Optimisation de Reporting :

- Analyse et optimisation des performances des requêtes SQL

-- Index pour optimiser les recherches fréquentes

CREATE INDEX IDX_SOUSCRIPTION_DATE ON
Souscription(Date_Souscription);

CREATE INDEX IDX_SOUSCRIPTION_AGENT ON Souscription(ID_Agent);

CREATE INDEX IDX_CLIENT_NOM ON Client(Nom, Prenom);

-- Statistiques pour l'optimiseur

ANALYZE TABLE Souscription COMPUTE STATISTICS;

ANALYZE TABLE Client COMPUTE STATISTICS;

- Création de rapports automatisés pour suivre les souscriptions et les commissions

```
CREATE OR REPLACE PROCEDURE GenererRapportMensuel(
    p_mois IN NUMBER,
    p_annee IN NUMBER
) IS
    TYPE RefCursor IS REF CURSOR;
    v_rapport RefCursor;
BEGIN
    -- Rapport des ventes
    OPEN v_rapport FOR
        SELECT
            p.Nom AS Produit,
            COUNT(*) AS Nombre_Souscriptions,
            SUM(p.Prix_Base) AS CA_Total
        FROM Souscription s
        JOIN Produit p ON s.ID_Produit = p.ID_Produit
        WHERE EXTRACT(MONTH FROM s.Date_Souscription) = p_mois
        AND EXTRACT(YEAR FROM s.Date_Souscription) = p_annee
        GROUP BY p.Nom
        ORDER BY CA_Total DESC;

    -- Export ou traitement du rapport...
END;
```