**UNIVERSITÉ du Québec à Trois-Rivières**

*Bases de données I*  
  
PAR  
  
Ismaïl Biskri et Ramla Ghali

**Rapport du Projet « gestion d’une agence de voyage »**

Auteurs :

Abdelkrim Salhi

Josue Ilunga Mulangu

Malek Khalifa

Sidi Moctar Issoufi Maiga

Avril 2025

Table des matières

**Introduction4**

Évaluation et Analyse des besoins 4

Modélisation conceptuelle et logique 8

Création des tables et insertion des données sur Oracle 13

Présentation des requêtes et des déclencheurs16

Conclusion 28

**Introduction**

Dans un contexte de transformation numérique croissante, les entreprises doivent s'adapter pour améliorer leur efficacité et offrir de meilleurs services à leurs clients. C'est dans cette optique que l'agence de voyage ÉCO VOYAGE, récemment acquise par M. Larue et un groupe d'investisseurs de Trois-Rivières, souhaite moderniser ses opérations en informatisant la gestion des clients, des réservations et des accompagnateurs.

L'analyse effectuée par M. Larue a révélé que l'absence d'un système informatisé entraîne des inefficacités dans la gestion des réservations et des paiements, limitant ainsi la capacité de l'agence à offrir un service rapide et fiable à ses clients. Afin de remédier à cette situation, un projet de développement d'une base de données relationnelle a été initié par Georges, son neveu, qui n'a malheureusement pas pu le mener à terme.

Notre mission est donc de poursuivre le travail entrepris en concevant et en mettant en place un système de gestion performant à l'aide d'Oracle. Ce système va permettre une gestion optimisée des clients, des voyages, des réservations et des paiements, tout en assurant la cohérence et l'intégrité des données. De plus, il va être évolutif afin de répondre aux besoins futurs de l'agence, notamment l'installation de terminaux d'information et la mise en place d'une plateforme en ligne pour la réservation des voyages.

Ce rapport détaille l'analyse et la conception du modèle Entité-Relation, la transformation des données en tables relationnelles et l'implantation du système à l'aide d'Oracle. Il inclut également une série de requêtes SQL et de triggers destinés à automatiser certaines opérations et garantir la fiabilité du système. Enfin, un guide utilisateur sera fourni afin de faciliter la prise en main de l'application par les futurs utilisateurs.

**1. Évaluation et Analyse des besoins**

**1.1 Contexte et Objectifs**

L’agence Éco Voyage cherche à optimiser la gestion de ses réservations et de son programme de fidélité, le Club Sélect. Actuellement, la gestion des réservations et des clients repose sur des processus manuels et disparates, entraînant des inefficacités et des risques d’erreurs. L’objectif de cette étude est d’identifier les besoins fonctionnels et les exigences en matière de données

afin de concevoir un système efficace et structuré.

**1.2 Identification des Besoins**

Les besoins identifiés se concentrent sur plusieurs axes :

* **Gestion des réservations** : enregistrement des voyages, suivi des paiements, annulations et modifications.
* **Gestion des clients** : création et mise à jour des profils clients, suivi des préférences.
* **Gestion du programme CLUB SELECT** : attribution des rabais sur les voyages, gestion des statuts des membres.
* **Gestion des offres et promotions** : application des réductions selon le programme de fidélité, la date de réservation et le mode de paiement.
* **Gestion des annulations de réservation**: gérer les frais liés à l’annulation d’une réservation.

**1.3 Répertoire des éléments de données**

| **Nom de l’élément** | **Description** | **Domaine de valeurs** |
| --- | --- | --- |
| ID Réservation | Identifiant unique d’une réservation | Alphanumérique |
| ID Client | Identifiant unique du client | Numérique |
| noAgent | Identifiant unique de l’agent | Alphanumérique |
| noMembre | Identifiant unique du membre (Ex: ETU‑10) | Numérique |
| idAccompagnateur | Identifiant unique de l’accompagnateur | Alphanumérique |
| nomAgence | Identifiant unique de l’agence | Domaine\_Nom\_firme |
| idForfait | Identifiant unique du forfait | Alphanumérique |
| noCircuit | Identifiant unique du circuit | Alphanumérique |
| idForfait | Identifiant unique du forfait | Alphanumérique |
| idFrais | Identifiant unique des frais | Alphanumérique |
| refPaiement | Identifiant unique du paiement | Alphanumérique |
| idAnnulation | Identifiant unique de l’annulation | Alphanumérique |
| idReduction | Identifiant unique de la reduction | Alphanumérique |
| adresseAgence | Adresse postale de l’agence | Domaine\_Adresse |
| emailClient | Adresse courrielle du client | Domaine\_Email |
| emailAgence | Adresse courrielle de l’agence | Domaine\_Email |
| telephoneClient | Numéro de téléphone client | Domaine\_NumTel |
| telephoneAccom | Numéro de téléphone de l’accompagnateur | Domaine\_NumTel |
| telephone | Numéro de téléphone de l’agence | Domaine\_NumTel |
| fax | Numéro de fax de l’agence | Domaine\_NumTel |
| dateReservation | La date de réservation du voyage | Domaine\_Date |
| dateAnnulation | La date d’annulation du voyage | Domaine\_Date |
| date\_depart | La date du départ du voyage | Domaine\_Date |
| date\_retour | La date de retour du voyage | Domaine\_Date |
| nomClient | Le nom complet du client | Domaine\_chaineCaractère |
| typeMembre | Selon l’âge du client le champs prend une valeur qui décrit la tranche d’âge | Domaine\_chaineCaractère |
| nomAccom | Le nom complet de l’accompagnateur | Domaine\_chaineCaractère |
| nomDestination | Le nom de la destination du voyage | Domaine\_chaineCaractère |
| typeForfait | Nom du forfait | Domaine\_chaineCaractère |
| nomAgent | Le nom complet de l’agent | Domaine\_chaineCaractère |
| typeFrais | Description des frais ajoutés | Domaine\_chaineCaractère |
| typePaiement | Description du paiement | Domaine\_chaineCaractère |
| statutReservation | Etat de réservation | Domaine\_chaineCaractère |
| pourcentageReduction | Precision de la valeur de reduction | Domaine\_chaineCaractère |
| statutReservation | Etat de réservation | Domaine\_chaineCaractère |
| typeAnnulation | Source d’annulation ou cause | Domaine\_chaineCaractère |
| Prix | Le prix de base du circuit | Domaine\_Montant |
| prixMembre | Montant à payer pour une adhésion | Domaine\_Montant |
| valeurFrais | Precision de la valeur des frais | Domaine\_chaineCaractère |
| Acompte | Le montant payé à la réservation | Domaine\_Montant |
| MontantReduction | Le montant calculé de chaque réduction | Domaine\_Montant |

**1.4 Répertoire des domaines des données**

| **Nom du domaine** | **Définition** |
| --- | --- |
| Alphanumérique | Chaine de caractères unique attribuée à un élément (client, réservation…) pour garantir son unicité composée de lettres et de chiffres |
| Numérique | Information exprimée uniquement par des nombres |
| Domaine\_Nom\_firme | Chaîne de caractères alphabétiques (max: 100) Ex: HDC France |
| Domaine\_Adresse | Chaîne de caractères (max: 250) syntaxe: <numéro> <rue>, <ville>, <pays> Ex: 7D rue du ballon, Noisy-le-grand, France |
| Domaine\_Email | Chaine de caractère qui doit obligatoirement contenir « @ » |
| Domaine\_NumTel | Information exprimée uniquement par des nombres |
| Domaine\_Date | Chaine de caractère sous la forme JJ/MM/AAAA |
| Domaine\_chaineCaractère | Chaine de caractères max (250) |
| Domaine\_Montant | Des données numérique avec virgule flottante |

**2. Modélisation conceptuelle et logique :**

**2.1 Modèle Entité Relation :**

Ci-dessous on vous présente le modèle Entité Relation du système :

Une image contenant texte, diagramme, Plan, schématique

Le contenu généré par l’IA peut être incorrect.

**2.1 Explication du modèle :**

* Description des entités principales
* **Client**

Attributs : idClient\*, nomClient, emailClient, telephoneClient, âge

Représente un client de l’agence de voyage. L’identifiant idClient est la clé primaire.

* **Membre**

Attributs : noMembre\*, typeMembre,prixMembre

Chaque client peut être un membre (CLUB SELECT). Un client a au plus un profil de membre.

* **Réservation**

Attributs : idReservation\*, dateReservation, statutReservation, acompte

Permet de suivre les réservations faites par les clients. C’est une entité centrale du modèle.

* **Paiement**

Attributs : refPaiement\*, typePaiement

Permet de gérer les différents types de paiements effectués.

* **Réduction**

Attributs : idReduction\*, montantReduction

Représente une réduction qui peut être appliquée à une réservation.

* **Annulation**

Attributs : idAnnulation\*, typeAnnulation, dateAnnulation

Permet de gérer l’annulation d’une réservation.

* **Circuit**

Attributs : noCircuit\*, date\_depart, date\_retour, prix

Définit un circuit touristique (voyage, visite, etc.)

* **Destination**

Attributs : idDestination\*, nomDestination

Un circuit contient une ou plusieurs destinations.

* **Forfait**

Attributs : idForfait\*, typeForfait

Définit des offres particulières associées à des circuits.

* **Frais**

Attributs : idFrais\*, pourcentageFrais, typeFrais

Permet de gérer des frais supplémentaires éventuels.

* **Accompagnateur**

Attributs : idAccompagnateur\*, nomAccom, telephoneAccom

Désigne la personne en charge de l’accompagnement d’un circuit.

* **Agent**

Attributs : noAgent\*, nomAgent

L’agent effectue des opérations de réservation au sein de l’agence.

* **Agence**

Attributs : nomAgence, adresseAgence, emailAgence, telephone, fax

L’entité agence représente les bureaux physiques de l’entreprise.

* Relations et cardinalités
* **Est (entre client et membre)**

Cardinalité : [0,1] client — [1,1] membre

Un client peut être un membre, mais chaque membre doit être un client.

* **Faire (entre client et réservation)**

Cardinalité : [1,N] client — [1,1] réservation

Un client peut faire plusieurs réservations, mais une réservation est faite par un seul client.

* **Appliquer (entre réduction et réservation)**

Cardinalité : [0, N] réduction — [0, N] réservation

Une réservation peut bénéficier de plus d’une réduction (Réservez tôt, CLUB SELECT …), et une réduction peut être utilisée dans plusieurs réservations.

* **Réaliser (entre réservation et annulation)**

Cardinalité : [1,1] annulation — [0,1] réservation

Une réservation peut être annulée (ou non), chaque annulation correspond à une seule réservation.

* **Régler (entre paiement et réservation)**

Cardinalité : [1,1] paiement — [1, N] réservation

Une réservation peut être réglée via plusieurs paiements. Chaque paiement est lié à une réservation unique.

* **Concerner (entre réservation et circuit)**

Cardinalité : [0, N] circuit — [1,1] réservation

Une réservation concerne un seul circuit, et un circuit peut être concerné par plusieurs réservations.

* **Ajouter (entre frais et réservation)**

Cardinalité : [0, N] frais — [0, N] réservation

Permet de modéliser les frais supplémentaires qui peuvent être ajoutés à une réservation, le même type de frais peut se rajouter à plusieurs réservations et une réservation peut subir plusieurs frais (changement de circuit avant moins de 45jours, frais d’annulation du voyage)

* **Effectuer (entre agent et réservation)**

Cardinalité : [0, N] agent — [1,1] réservation

Un agent peut effectuer plusieurs réservations, et une réservation peut être faite avec l’intervention d’un et un seul agent.

* **Gérer (entre accompagnateur et réservation)**

Cardinalité : [0, N] accompagnateur — [1,1] réservation

Un accompagnateur gère plusieurs réservations, mais chaque réservation a un seul accompagnateur.

* **Contenir (entre circuit et destination)**

Cardinalité : [1, N] circuit — [1,1] destination

Un circuit contient une ou plusieurs destinations. Chaque destination est associée à un ou plusieurs circuits.

* **Appartenir (entre circuit et forfait)**

Cardinalité : [1, N] forfait — [1,1] circuit

Un circuit appartient à un type de forfait, chaque forfait peut regrouper plusieurs circuits.

* **Travailler (entre agent et agence)**

Cardinalité : [1,1] agent — [1,N] agence

Chaque agent travaille dans une seule agence.

* Justification des choix de modélisation

Le modèle distingue bien les acteurs (clients, membres, agents, accompagnateurs) des entités métier (réservation, circuit, forfait, etc.).

Le client est au centre du processus de réservation, ce qui reflète une logique métier réaliste dans une application touristique.

La relation membre permet de gérer des fonctionnalités avancées (réductions, privilèges).

Les relations avec cardinalité [0, N] permettent une souplesse pour intégrer des frais, des paiements multiples, ou l’absence d'annulation.

L'ajout des entités frais, réduction, et forfait permet une personnalisation fine des offres touristiques.

**3. Création des tables et insertion des données sur Oracle :**

**3.1 Création des tables :**

En appliquant les règles de conversion du modèle Entité relation vers des tables on obtient 16 tables tels que :

**Client** (idClient, nomClient, emailClient, telephoneClient, age)

**Agence** (nomAgence, adresseAgence, emailAgence, telephone, fax)

**Accompagnateur** (idAccompagnateur, nomAccom, telephoneAccom, nomAgence)

**Membre** (noMembre, type, idClient)

**Agent** (noAgent, nomAgent)

**Destination** (idDestination, nomDestination)

**Forfait** (idForfait, type)

**Circuit** (noCircuit, date\_depart, date\_retour, prix, idForfait)

**DestinationCircuit** (idDestination, noCircuit)

**Reduction** (idReduction, pourcentage)

**Frais** (idFrais, pourcentageFrais, type)

**Annulation** (idAnnulation, typeAnnulation, dateAnnulation, idReservation)

**Reservation** (idReservation, dateReservation, montant, statut, idClient, idAccompagnateur, noAgent, noCircuit,acompte)

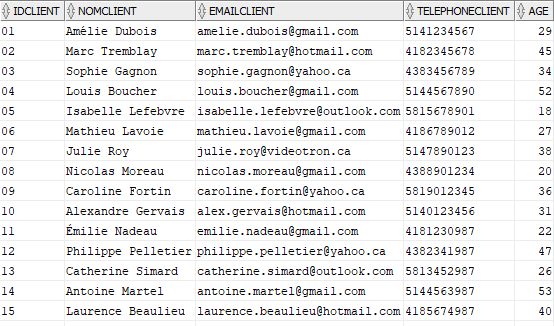
**Paiement** (refPaiement, typePaiement,somme,datePaiement, idReservation)

**ReservationReduction** (idReservation, idReduction)

**ReservationFrais** (idReservation, idFrais)

**3.2 Implémentation sur Oracle et insertion des données :**

Après exécution du code que vous trouverez sur le fichier (CodeSource.sql) en pièce jointe on arrive à stocker toutes les informations nécessaires pour le fonctionnement de l’agence de voyage et ci-dessous vous trouverez quelques exemples des tables remplies :



* Extrait des données dans la table Client –

Une image contenant texte, Police, capture d’écran, nombre

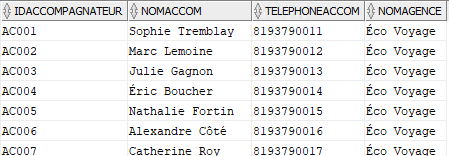
Le contenu généré par l’IA peut être incorrect.

* Extrait des données dans la table Membre –

Une image contenant texte, Police, capture d’écran, nombre

Le contenu généré par l’IA peut être incorrect.

* Extrait des données dans la table Agent –



* Extrait des données dans la table Accompagnateur –

Une image contenant texte, capture d’écran, nombre, Police

Le contenu généré par l’IA peut être incorrect.

* Extrait des données dans la table Circuit –

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

* Extrait des données dans la table Frais –

Une image contenant texte, capture d’écran, nombre, Police

Le contenu généré par l’IA peut être incorrect.

* Extrait des données dans la table Réservation –

Une image contenant texte, Police, capture d’écran, nombre

Le contenu généré par l’IA peut être incorrect.

* Extrait des données dans la table Reduction –

**4. Présentation des requêtes et des déclencheurs :**

**4.1 les déclencheurs :**

1. **Déclencheur pour générer le numéro du membre et type et prix d’adhésion en fonction de l'âge du client :**

CREATE OR REPLACE TRIGGER trg\_set\_nomembre

BEFORE INSERT ON membre

FOR EACH ROW

DECLARE

v\_age NUMBER;

BEGIN

-- Récupérer l'âge depuis la table client

SELECT age INTO v\_age FROM client WHERE idClient = :NEW.idClient;

-- Générer le nomembre et le type en fonction de l'âge

IF v\_age < 21 THEN

:NEW.nomembre := 'ETU-' || :NEW.idClient;

:NEW.type := 'Étudiant';

:NEW.prix := 150; -- Prix d'adhésion pour étudiant

ELSIF v\_age BETWEEN 21 AND 60 THEN

:NEW.nomembre := 'ADU-' || :NEW.idClient;

:NEW.type := 'Adulte';

:NEW.prix := 250; -- Prix d'adhésion pour adulte

ELSE

:NEW.nomembre := 'AOR-' || :NEW.idClient;

:NEW.type := 'Âge d''or';

:NEW.prix := 125; -- Prix d'adhésion pour personne agée

END IF;

END;

/

1. **Déclencheur pour générer le montant d'acompte avant chaque insertion dans la table réservation :**

CREATE OR REPLACE TRIGGER trg\_set\_acompte\_reservation

BEFORE INSERT ON reservation

FOR EACH ROW

DECLARE

v\_prix\_circuit circuit.prix%TYPE;

v\_acompte\_calcule NUMBER;

BEGIN

-- Récupérer le prix du circuit associé

SELECT prix

INTO v\_prix\_circuit

FROM circuit

WHERE noCircuit = :NEW.noCircuit;

-- Calculer l'acompte comme le plus élevé entre 500$ et 10% du prix du circuit

v\_acompte\_calcule := GREATEST(500, v\_prix\_circuit \* 0.10);

-- Assigner la valeur calculée à la colonne acompte

:NEW.acompte := v\_acompte\_calcule;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- Si aucun circuit n’est trouvé, mettre l’acompte à 500$ par défaut

:NEW.acompte := 500;

END;

/

1. **Déclencheur permettant d'insérer une ligne dans la table reservationFrais (Frais de après chaque réservation effectuée à moins de 21 jours du départ :**

CREATE OR REPLACE TRIGGER trg\_frais\_reservation\_tardive

AFTER INSERT ON reservation

FOR EACH ROW

DECLARE

v\_date\_depart circuit.date\_depart%TYPE;

v\_jours\_restants NUMBER;

BEGIN

-- Récupération de la date de départ avec le bon nom d'attribut

SELECT date\_depart

INTO v\_date\_depart

FROM circuit

WHERE noCircuit = :NEW.noCircuit;

-- Calcul des jours restants avant le départ

v\_jours\_restants := v\_date\_depart - :NEW.dateReservation;

-- Application des frais uniquement si réservation < 21 jours avant départ

IF v\_jours\_restants < 21 AND v\_jours\_restants >= 0 THEN

BEGIN

-- Insertion des frais de réservation tardive (F6)

INSERT INTO reservationFrais (idReservation, idFrais)

VALUES (:NEW.idReservation, 'F6');

DBMS\_OUTPUT.PUT\_LINE('[Succès] Frais tardifs appliqués à la réservation ' ||

:NEW.idReservation || ' (' || v\_jours\_restants || ' jours avant départ)');

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

NULL; -- Ignorer si les frais sont déjà appliqués

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('[Erreur] Impossible d\''appliquer les frais: ' || SQLERRM);

END;

END IF;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('[Erreur] Dans le déclencheur: ' || SQLERRM);

END;

/

1. **Déclencheur permettant d'insérer une ligne dans la table reservationreduction (Reduction après chaque réservation éligible à la réduction « réservez tôt » :**

CREATE OR REPLACE TRIGGER trg\_apply\_reduction\_reztot

AFTER INSERT ON reservation

FOR EACH ROW

DECLARE

v\_date\_depart circuit.date\_depart%TYPE;

v\_jours\_avant\_depart NUMBER;

BEGIN

-- Récupérer la date de départ du circuit

SELECT date\_depart INTO v\_date\_depart

FROM circuit

WHERE noCircuit = :NEW.noCircuit;

-- Calculer le nombre de jours avant le départ

v\_jours\_avant\_depart := v\_date\_depart - :NEW.dateReservation;

-- Appliquer la réduction si plus de 90 jours

IF v\_jours\_avant\_depart > 90 THEN

INSERT INTO reservationReduction (idReservation, idReduction, montantReduction)

VALUES (:NEW.idReservation, 'REZTOT', 100);

END IF;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Erreur trg\_apply\_reduction\_reztot: ' || SQLERRM);

END;

/

1. **Déclencheur permettant d'insérer une ligne dans la table reservationreduction (Reduction après chaque réservation éligible à la réduction « Club Select » :**

CREATE OR REPLACE TRIGGER trg\_apply\_reduction\_clubselect

AFTER INSERT ON reservation

FOR EACH ROW

DECLARE

v\_est\_membre NUMBER := 0;

v\_prix\_circuit NUMBER := 0;

v\_montant\_reduction NUMBER := 0;

BEGIN

-- Vérifier si le client est membre

SELECT COUNT(\*) INTO v\_est\_membre

FROM membre

WHERE idClient = :NEW.idClient;

IF v\_est\_membre > 0 THEN

-- Récupérer le prix du circuit

SELECT prix INTO v\_prix\_circuit

FROM circuit

WHERE noCircuit = :NEW.noCircuit;

-- Calcul de la réduction de 5%

v\_montant\_reduction := v\_prix\_circuit \* 0.05;

-- Insertion de la réduction dans la table

INSERT INTO reservationReduction (idReservation, idReduction, montantReduction)

VALUES (:NEW.idReservation, 'CLUBSELECT', v\_montant\_reduction);

END IF;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Erreur trg\_apply\_reduction\_clubselect: ' || SQLERRM);

END;

/

1. **Déclencheur permettant d'insérer une ligne dans la table reservationreduction (Reduction après chaque réservation éligible à la réduction « Clients Fidèles » :**

CREATE OR REPLACE TRIGGER trg\_apply\_reduction\_fidelite

AFTER INSERT ON reservation

FOR EACH ROW

DECLARE

v\_nb\_voyages\_passes NUMBER := 0;

BEGIN

SELECT COUNT(\*) INTO v\_nb\_voyages\_passes

FROM reservation r

JOIN circuit c ON r.noCircuit = c.noCircuit

WHERE r.idClient = :NEW.idClient

AND r.idReservation != :NEW.idReservation

AND c.date\_depart >= ADD\_MONTHS(SYSDATE, -60) -- 5 dernières années

AND r.statut = 'validé';

IF v\_nb\_voyages\_passes > 0 THEN

INSERT INTO reservationReduction (idReservation, idReduction, montantReduction)

VALUES (:NEW.idReservation, 'FIDELITE', 50);

END IF;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Erreur trg\_apply\_reduction\_fidelite: ' || SQLERRM);

END;

/

1. **Déclencheur permettant d'insérer une ligne dans la table reservationreduction (Reduction après chaque réservation éligible à la réduction « Paiement Comptant ou chèque » :**

CREATE OR REPLACE TRIGGER trg\_reduction\_paiement\_comptant

AFTER INSERT ON paiement

FOR EACH ROW

WHEN (NEW.typePaiement IN ('Espèce', 'Chèque'))

DECLARE

v\_prix\_circuit NUMBER := 0;

v\_montant\_reduction NUMBER := 0;

BEGIN

-- Récupérer le prix du circuit associé à la réservation

SELECT c.prix INTO v\_prix\_circuit

FROM reservation r

JOIN circuit c ON r.noCircuit = c.noCircuit

WHERE r.idReservation = :NEW.idReservation;

-- Calcul de la réduction de 2%

v\_montant\_reduction := v\_prix\_circuit \* 0.02;

-- MERGE évite les doublons

MERGE INTO reservationReduction rr

USING (

SELECT :NEW.idReservation AS idRes, 'CHEQUE' AS idRed, v\_montant\_reduction AS montantRed FROM dual

) src

ON (rr.idReservation = src.idRes AND rr.idReduction = src.idRed)

WHEN NOT MATCHED THEN

INSERT (idReservation, idReduction, montantReduction)

VALUES (src.idRes, src.idRed, src.montantRed);

EXCEPTION

WHEN OTHERS THEN

NULL; -- Ignorer les erreurs silencieusement

END;

/

1. **Déclencheur pour insérer les frais de changement de circuit dans reservationFrais:**

CREATE OR REPLACE TRIGGER trg\_apply\_frais\_changement\_circuit

AFTER UPDATE OF noCircuit ON reservation

FOR EACH ROW

WHEN (OLD.noCircuit IS NOT NULL AND NEW.noCircuit IS NOT NULL AND OLD.noCircuit != NEW.noCircuit)

DECLARE

    v\_frais\_changement NUMBER := 100;

BEGIN

    -- Insérer les frais de changement dans reservationFrais

    INSERT INTO reservationFrais (idReservation, idFrais)

    VALUES (:NEW.idReservation, 'F7');

END;

/

1. **Déclencheur permet d'insérer une ligne dans la table reservationFrais après chaque annulation de réservation :**

CREATE OR REPLACE TRIGGER trg\_insert\_reservation\_frais\_annulation

AFTER INSERT ON annulation

FOR EACH ROW

DECLARE

v\_date\_depart circuit.DATE\_DEPART%TYPE;

v\_jours\_avant\_depart NUMBER;

v\_id\_frais frais.idFrais%TYPE;

BEGIN

IF :NEW.typeAnnulation != 'decision Agence' THEN

-- Récupérer la date de départ du circuit associé à la réservation

SELECT c.date\_Depart

INTO v\_date\_depart

FROM circuit c

JOIN reservation r ON r.noCircuit = c.noCircuit

WHERE r.idReservation = :NEW.idReservation;

-- Calculer le nombre de jours entre l'annulation et le départ

v\_jours\_avant\_depart := v\_date\_depart - :NEW.dateAnnulation;

-- Déterminer le type de frais d'annulation en fonction du délai

IF v\_jours\_avant\_depart >= 45 THEN

v\_id\_frais := 'F1'; -- Acompte

ELSIF v\_jours\_avant\_depart BETWEEN 31 AND 44 THEN

v\_id\_frais := 'F2'; -- 40%

ELSIF v\_jours\_avant\_depart BETWEEN 21 AND 30 THEN

v\_id\_frais := 'F3'; -- 60%

ELSIF v\_jours\_avant\_depart BETWEEN 1 AND 20 THEN

v\_id\_frais := 'F4'; -- 100%

ELSE

v\_id\_frais := 'F5'; -- 100%

END IF;

-- Insérer les frais d'annulation dans reservationFrais

INSERT INTO reservationFrais (idReservation, idFrais)

VALUES (:NEW.idReservation, v\_id\_frais);

-- Mettre à jour le statut de la réservation à 'Annulée'

UPDATE reservation

SET statut = 'Annulée'

WHERE idReservation = :NEW.idReservation;

ELSE

-- Cas spécifique : annulation sur décision de l'agence

INSERT INTO reservationFrais (idReservation, idFrais)

VALUES (:NEW.idReservation, 'F8');

-- Mettre à jour le statut de la réservation à 'Annulée'

UPDATE reservation

SET statut = 'Annulée'

WHERE idReservation = :NEW.idReservation;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Erreur: Données non trouvées pour la réservation ' || :NEW.idReservation);

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Erreur dans trg\_insert\_reservation\_frais\_annulation: ' || SQLERRM);

END;

/

**4.1 les requêtes :**

1. **Requête qui affiche la liste des annulations avec les détails des frais (Le : clients n'ayant pas payé leur solde à la date convenue) :**

SELECT

a.idreservation AS reservationAnnulée,

a.dateannulation,

r.acompte + SUM(p.somme) - COALESCE(SUM(rr.montantReduction), 0) AS FraisEncaissés,

c.nomClient

FROM annulation a

JOIN reservation r ON a.idreservation = r.idreservation

JOIN paiement p ON p.idreservation = r.idreservation

JOIN client c ON c.idclient = r.idclient

LEFT JOIN reservationReduction rr ON rr.idReservation = r.idreservation

WHERE a.typeannulation = 'retard paiement'

GROUP BY a.idreservation, a.dateannulation, r.acompte, c.nomClient;

1. **Requête qui affiche la liste des gens auxquels nous devrons envoyer une lettre d'avertissement 2 semaines avant la date à laquelle leur solde deviendra dû :**

SELECT

r.idreservation,

c.idclient,

c.nomclient,

ci.date\_depart,

ci.prix AS prixInitial,

-- Prix après réduction

ci.prix - COALESCE((

SELECT SUM(rr2.montantReduction)

FROM reservationReduction rr2

WHERE rr2.idReservation = r.idreservation

), 0) AS prixApresReduction,

-- Reste à payer après paiements, acompte et réductions

ci.prix

- COALESCE((

SELECT SUM(p2.somme)

FROM paiement p2

WHERE p2.idreservation = r.idreservation

), 0)

- r.acompte

- COALESCE((

SELECT SUM(rr2.montantReduction)

FROM reservationReduction rr2

WHERE rr2.idReservation = r.idreservation

), 0) AS resteApayer

FROM reservation r

JOIN client c ON c.idclient = r.idclient

JOIN circuit ci ON ci.nocircuit = r.nocircuit

WHERE ci.date\_depart - SYSDATE < 59 -- 14 jours plus 45 jours avant le départ ( date limite de paiement )

AND (

ci.prix

- COALESCE((

SELECT SUM(p2.somme)

FROM paiement p2

WHERE p2.idreservation = r.idreservation

), 0)

- r.acompte

- COALESCE((

SELECT SUM(rr2.montantReduction)

FROM reservationReduction rr2

WHERE rr2.idReservation = r.idreservation

), 0)

) > 0

ORDER BY ci.date\_depart;

1. **Requête qui affiche la liste des circuits avec les informations pertinentes sur chaque circuit :**

SELECT c.nocircuit,

c.date\_depart,

c.date\_retour,

c.prix,

LISTAGG(d.nomdestination, ', ') WITHIN GROUP (ORDER BY d.nomdestination) AS destinations,

f.type AS typeForfait

FROM circuit c

JOIN destinationCircuit dc ON c.nocircuit = dc.nocircuit

JOIN destination d ON dc.iddestination = d.iddestination

JOIN forfait f ON c.idforfait = f.idforfait

WHERE c.date\_depart > SYSDATE

GROUP BY c.nocircuit, c.date\_depart, c.date\_retour, c.prix, f.type;

1. **Requête qui affiche la liste des clients qui ont des frais de changement de circuit :**

SELECT

f.idfrais,

rf.idreservation,

r.idClient ,

c.nomclient,

r.nocircuit as circuitReservé

FROM

frais f

JOIN reservationfrais rf ON rf.idfrais = f.idfrais

JOIN reservation r ON r.idreservation = rf.idreservation

JOIN client c ON c.idclient = r.idclient

WHERE f.idfrais = 'F7' ;

1. **Requête qui affiche la liste des clients avec le circuit réservé et les destinations associées :**

SELECT

c.idclient,

c.nomclient,

r.idreservation,

r.nocircuit,

LISTAGG (d.nomdestination, ', ') WITHIN GROUP (ORDER BY d.nomdestination) AS destinations

FROM client c

JOIN reservation r ON r.idclient = c.idclient

JOIN circuit ci ON r.nocircuit = ci.nocircuit

JOIN destinationCircuit dc ON ci.nocircuit = dc.nocircuit

JOIN destination d ON dc.iddestination = d.iddestination

GROUP BY c.idclient, c.nomclient, r.idreservation, r.nocircuit;

1. **Requête qui affiche la liste des réservations annulées avec les détails des frais d'annulation :**

SELECT

    r.idreservation,

    r.nocircuit,

    c.nomclient,

    a.dateannulation,

    ci.date\_depart,

    -- Total payé (paiements + acompte, sans retrait des réductions)

    (SELECT COALESCE(SUM(somme), 0)

     FROM paiement

     WHERE idReservation = r.idreservation) + r.acompte AS totalPaye,

    f.type AS typeAnnulation,

    -- Frais d'annulation (calculés uniquement sur le total payé)

    CASE

        WHEN f.idfrais = 'F1' THEN r.acompte

        WHEN REGEXP\_LIKE(f.pourcentagefrais, '^\d+%$') THEN

            (

                (SELECT COALESCE(SUM(somme), 0)

                 FROM paiement

                 WHERE idReservation = r.idreservation) + r.acompte

            ) \* TO\_NUMBER(REGEXP\_SUBSTR(f.pourcentagefrais, '\d+')) / 100

        ELSE 0

    END AS fraisAnnulation,

    -- Montant remboursé (total payé - frais)

    CASE

        WHEN f.idfrais = 'F1' THEN

            (SELECT COALESCE(SUM(somme), 0)

             FROM paiement

             WHERE idReservation = r.idreservation)

        WHEN REGEXP\_LIKE(f.pourcentagefrais, '^\d+%$') THEN

            (

                (SELECT COALESCE(SUM(somme), 0)

                 FROM paiement

                 WHERE idReservation = r.idreservation) + r.acompte

            ) - (

                (

                    (SELECT COALESCE(SUM(somme), 0)

                     FROM paiement

                     WHERE idReservation = r.idreservation) + r.acompte

                ) \* TO\_NUMBER(REGEXP\_SUBSTR(f.pourcentagefrais, '\d+')) / 100

            )

        ELSE

            (SELECT COALESCE(SUM(somme), 0)

             FROM paiement

             WHERE idReservation = r.idreservation) + r.acompte

    END AS montantRembourse

FROM reservation r

JOIN annulation a ON a.idreservation = r.idreservation

JOIN client c ON c.idclient = r.idclient

JOIN circuit ci ON ci.nocircuit = r.nocircuit

JOIN reservationfrais rf ON rf.idreservation = r.idreservation

JOIN frais f ON f.idfrais = rf.idfrais

WHERE r.statut = 'Annulée'

GROUP BY r.idreservation, r.nocircuit, c.nomclient, a.dateannulation, ci.date\_depart,

         f.idfrais, f.type, f.pourcentagefrais, r.acompte;

1. **Requête qui affiche la liste des clients avec le montant total payé et le solde restant à payer pour chaque réservation :**

SELECT

    r.idreservation,

    c.idclient,

    c.nomclient,

    -- Montant des réductions (sous-requête)

    (SELECT COALESCE(SUM(montantReduction), 0)

     FROM reservationReduction

     WHERE idReservation = r.idreservation) AS totalReduction,

    -- Total payé (paiements + acompte, sans retrait des réductions)

    (SELECT COALESCE(SUM(somme), 0)

     FROM paiement

     WHERE idReservation = r.idreservation) + r.acompte AS totalPaye,

    -- Solde restant = (prix - réductions) - total payé

    (ci.prix

        - (SELECT COALESCE(SUM(montantReduction), 0)

           FROM reservationReduction

           WHERE idReservation = r.idreservation)

        - (

            (SELECT COALESCE(SUM(somme), 0)

             FROM paiement

             WHERE idReservation = r.idreservation) + r.acompte

        )

    ) AS soldeRestant

FROM client c

JOIN reservation r ON r.idclient = c.idclient

JOIN circuit ci ON r.nocircuit = ci.nocircuit

WHERE r.statut = 'validé'

-- filtre les réservations où le solde est encore dû

AND (

    (ci.prix

        - (SELECT COALESCE(SUM(montantReduction), 0)

           FROM reservationReduction

           WHERE idReservation = r.idreservation)

        - (

            (SELECT COALESCE(SUM(somme), 0)

             FROM paiement

             WHERE idReservation = r.idreservation) + r.acompte

        )

    ) > 0

)

ORDER BY totalPaye desc;

1. **Requête qui affiche la liste des clients avec le montant total payé et le solde restant à payer pour chaque réservation, y compris les réservations annulées :**

SELECT

r.idreservation,

r.statut,

c.idclient,

c.nomclient,

SUM(p.somme) + r.acompte + COALESCE(SUM(rr.montantReduction), 0) AS totalPaye,

ci.prix - (SUM(p.somme) + r.acompte + COALESCE(SUM(rr.montantReduction), 0)) AS soldeRestant

FROM client c

JOIN reservation r ON r.idclient = c.idclient

JOIN circuit ci ON r.nocircuit = ci.nocircuit

JOIN paiement p ON p.idreservation = r.idreservation

LEFT JOIN reservationReduction rr ON rr.idReservation = r.idreservation

WHERE r.statut IN ('validé', 'Annulée')

GROUP BY r.statut, r.idreservation, c.idclient, c.nomclient, r.acompte, ci.prix

HAVING ci.prix - (SUM(p.somme) + r.acompte + COALESCE(SUM(rr.montantReduction), 0)) > 0

ORDER BY c.nomclient;

1. **Requête qui affiche la liste des circuits avec le nombre de réservations et le montant total encaissé pour chaque circuit :**

SELECT

c.nocircuit,

COUNT(DISTINCT r.idreservation) AS nombreReservations,

-- Paiements + acomptes sans duplication

(

(SELECT COALESCE(SUM(p2.somme), 0)

FROM paiement p2

JOIN reservation r2 ON p2.idreservation = r2.idreservation

WHERE r2.nocircuit = c.nocircuit

AND r2.statut = 'validé')

+

(SELECT COALESCE(SUM(r3.acompte), 0)

FROM reservation r3

WHERE r3.nocircuit = c.nocircuit

AND r3.statut = 'validé')

) AS montantTotalEncaisse

FROM circuit c

JOIN reservation r ON r.nocircuit = c.nocircuit

WHERE r.statut = 'validé'

GROUP BY c.nocircuit

ORDER BY c.nocircuit;

1. **Requête qui affiche la liste des accompagnateurs avec les réservations associées :**

SELECT

a.idaccompagnateur,

a.nomaccom,

r.idreservation,

r.nocircuit

FROM accompagnateur a

JOIN reservation r ON r.idaccompagnateur = a.idaccompagnateur

WHERE r.statut = 'validé'

ORDER BY a.nomaccom, r.idreservation;

1. **Requête qui affiche la liste des réductions appliquées à chaque réservation :**

SELECT

    r.idreservation,

    c.nomclient,

    rr.idreduction,

    rr.nomreduction,

    red.montantreduction

FROM reservation r

JOIN client c ON c.idclient = r.idclient

JOIN reservationreduction rr ON rr.idreservation = r.idreservation

Join reduction red ON red.idreduction = rr.idreduction

WHERE r.statut = 'validé'

ORDER BY r.idreservation, rr.idreduction;

**N.B : Veuillez consulter 6 vues enregistrés sur notre session Oracle en utilisant quelques requêtes présentées au-dessus.**

**Conclusion**

Ce projet de conception et d'implémentation d'une base de données pour l'agence de voyage Éco Voyage a permis de répondre aux besoins identifiés en modernisant la gestion des réservations, des clients, des paiements et des promotions. À travers une analyse approfondie des besoins, nous avons élaboré un modèle conceptuel robuste (MCD) et l'avons transformé en un modèle logique (MLD) implémenté sous Oracle. Les 16 tables créées couvrent tous les aspects métiers, depuis la gestion des circuits et destinations jusqu'aux réductions et frais spécifiques.

Les déclencheurs automatisent des processus critiques tels que :

* La génération des numéros de membres selon l'âge des clients.
* Le calcul des acomptes et des réductions (Club Select, réservation anticipée, etc.).
* L'application des frais d'annulation ou de modification, garantissant ainsi la conformité aux politiques de l'agence.

Les requêtes SQL fournissent des outils opérationnels pour :

* Suivre les paiements en attente et les relances clients.
* Analyser les annulations et leurs impacts financiers.
* Optimiser l'allocation des accompagnateurs et la planification des circuits.

Ce système offre une solution intégrée, évolutive et fiable, facilitant la transition vers une gestion numérique pour Éco Voyage. Il prépare également le terrain pour des extensions futures, telles qu'une plateforme en ligne ou des terminaux d'information. Enfin, la documentation fournie (modèles, scripts, guide utilisateur) assure une maintenance et une prise en main aisées par les équipes de l'agence.

Ce travail illustre l'importance d'une modélisation rigoureuse et d'une automatisation intelligente pour transformer les défis opérationnels en opportunités d'amélioration continue.