Projet Final - Administration de Bases de Données Relationnelles

NOM: MERFOUK

PRÉNOM: Aissa

CLASSE: Administration de Bases de Données Relationnelles - B3

ANNÉE: 2024-2025

DATE DE RENDU: 26/11/2025

---

1. Introduction

1.1 Présentation du projet Ynov-Air

Le projet Ynov-Air est une plateforme de simulation de système de réservation de vols aériens. Son objectif principal est de gérer l'ensemble du cycle de vie d'un vol, de sa planication à sa réservation par les passagers. Cette application web est construite autour d'une base de données relationnelle permettant de modéliser des entités clés : aéroports, vols, passagers et réservations.

1.2 Fonctionnalité choisie : Gestion des Repas (Meals)

Justification: L'ajout de la gestion des repas enrichit l'expérience utilisateur et constitue une fonctionnalité standard dans les systèmes de réservation modernes. Cette fonctionnalité permet de :

- Démontrer une extension cohérente du modèle de données existant
- Mettre en œuvre une relation un-à-plusieurs (Meal vers Booking)
- Gérer l'intégrité référentielle avec `ON DELETE SET NULL`

- Créer des requêtes analytiques sur les préférences alimentaires
- Illustrer le principe CRUD via l'ORM de l'application

1.3 Objectifs personnels

- Maîtriser la modélisation de bases de données avec intégration d'une nouvelle fonctionnalité
- Développer des requêtes SQL complexes utilisant jointures, agrégations et fonctions analytiques
- Appliquer les techniques d'optimisation (indexation, EXPLAIN)
- Documenter professionnellement l'ensemble du processus

---

2. Modélisation de la Base de Données

2.1 Schéma Entité-Relations (MLD)

Entité	Attributs	Clés	Relations
Airport	id, code, name, city, country	PK: id, UNIQUE: code	1,N Flight (origin), 1,N Flight (destination)
Meal	id, code, name, description, price, is_active	PK: id, UNIQUE: code	1,N Booking
Flight	id, ight_number, origin_id, destination_id, departure_time, arrival_time, duration, available_seats, total_seats, price, status	PK: id, UNIQUE: ight_number, FK: origin_id, destination_id	N,1 Airport (x2), 1,N Booking
Passenger	id, rst_name, last_name, email, phone, passport_number, date_of_birth	PK: id, UNIQUE: email, passport_number	1,N Booking
Booking	id, booking_reference, ight_id, passenger_id, booking_date, number_of_passengers, total_price, status, seat_number, description, meal_id	PK: id, UNIQUE: booking_reference, FK: ight_id, passenger_id, meal_id	N,1 Flight, N,1 Passenger, N,1 Meal

2.2 Choix de conception

Le modèle respecte la **3ème Forme Normale (3NF)** pour minimiser la redondance :

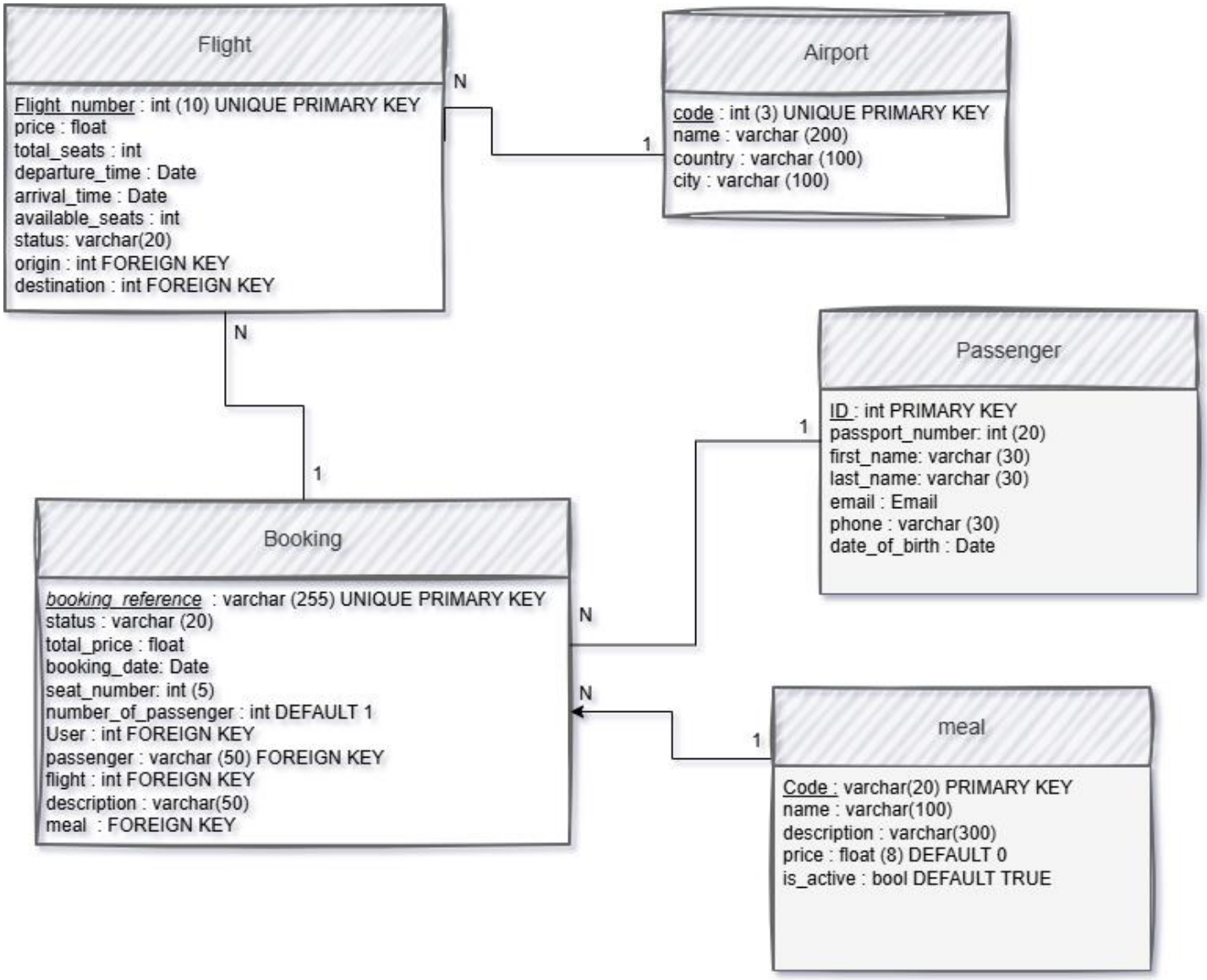
- **Table Meal indépendante :** Permet de modier les plats sans affecter les réservations passées
- **ON DELETE SET NULL :** Si un plat est supprimé, les réservations sont préservées avec meal_id = NULL
- **Contraintes UNIQUE :** Sur les identiants métier (codes IATA, numéros de passeport, références de réservation)
- **Contraintes CHECK :** Validation des sièges (available_seats ≤ total_seats)

2.3 Dictionnaire de données (Meal et Booking)

Table	Colonne	Type	Contraintes	Description
Meal	id	INT	PK, AUTO_INCREMENT	Identifiant unique
	code	VARCHAR(20)	UNIQUE, NOT NULL	Code court (ex: VEG, CHICK)
	name	VARCHAR(100)	NOT NULL	Nom complet du plat
	description	TEXT	NULL	Description détaillée
	price	DECIMAL(8,2)	NOT NULL, DEFAULT 0.00	Prix en euros
	is_active	BOOLEAN	NOT NULL, DEFAULT TRUE	Statut d'activité
Booking	meal_id	INT	FK vers Meal(id), NULL	Plat sélectionné

---

3. Scripts SQL de Création



3.1 Création des tables principales

```
-- Suppression des tables existantes
DROP TABLE IF EXISTS Booking, Flight, Airport, Passenger, Meal;

-- Table Airport CREATE
TABLE Airport (
    id INT PRIMARY KEY AUTO_INCREMENT,     code
VARCHAR(3) UNIQUE NOT NULL COMMENT 'Code IATA',     name
VARCHAR(200) NOT NULL,     city VARCHAR(100) NOT NULL,
country VARCHAR(100) NOT NULL );

-- Table Meal (Nouvelle fonctionnalité) CREATE
TABLE Meal (
    id INT PRIMARY KEY AUTO_INCREMENT,
    code VARCHAR(20) UNIQUE NOT NULL COMMENT 'Code unique (VEG, CHICK)',
name VARCHAR(100) NOT NULL,     description TEXT,
    price DECIMAL(8,2) NOT NULL DEFAULT 0.00,
is_active BOOLEAN NOT NULL DEFAULT TRUE );

-- Table Flight CREATE
TABLE Flight (
    id INT PRIMARY KEY AUTO_INCREMENT,
flight_number VARCHAR(10) UNIQUE NOT NULL,
origin_id INT NOT NULL,     destination_id INT
NOT NULL,     departure_time DATETIME NOT
NULL,     arrival_time DATETIME NOT NULL,
    duration TIME NOT NULL,
available_seats INT NOT NULL,
total_seats INT NOT NULL,     price
DECIMAL(8,2) NOT NULL,
    status VARCHAR(20) NOT NULL DEFAULT 'SCHEDULED',
    FOREIGN KEY (origin_id) REFERENCES Airport(id) ON DELETE CASCADE,
    FOREIGN KEY (destination_id) REFERENCES Airport(id) ON DELETE CASCADE
CONSTRAINT chk_seats CHECK (available_seats >= 0 AND available_seats
    CONSTRAINT chk_total_seats CHECK (total_seats BETWEEN 1 AND 300) );

-- Table Passenger CREATE
TABLE Passenger (
    id INT PRIMARY KEY AUTO_INCREMENT,
first_name VARCHAR(100) NOT NULL,
last_name VARCHAR(100) NOT NULL,     email
VARCHAR(255) UNIQUE NOT NULL,     phone
VARCHAR(20),
    passport_number VARCHAR(20) UNIQUE NOT NULL,
date_of_birth DATE NOT NULL );
```

```
-- Table Booking CREATE TABLE Booking (     id INT
PRIMARY KEY AUTO_INCREMENT,     booking_reference
VARCHAR(8) UNIQUE NOT NULL,     flight_id INT NOT
NULL,     passenger_id INT NOT NULL,
booking_date DATETIME NOT NULL,
number_of_passengers INT NOT NULL DEFAULT 1,
total_price DECIMAL(10,2) NOT NULL,     status
VARCHAR(20) NOT NULL DEFAULT 'PENDING',
seat_number VARCHAR(5),     description TEXT,
meal_id INT,
    FOREIGN KEY (flight_id) REFERENCES Flight(id) ON DELETE CASCADE,
    FOREIGN KEY (passenger_id) REFERENCES Passenger(id) ON DELETE CASCADE
FOREIGN KEY (meal_id) REFERENCES Meal(id) ON DELETE SET NULL,
    CONSTRAINT chk_num_passengers CHECK (number_of_passengers BETWEEN 1
A );
```

3.2 Création des index d'optimisation

```
-- Index sur clés étrangères (jointures)
CREATE INDEX idx_flight_origin ON Flight(origin_id);
CREATE INDEX idx_flight_destination ON Flight(destination_id);
CREATE INDEX idx_booking_flight ON Booking(flight_id);
CREATE INDEX idx_booking_passenger ON Booking(passenger_id);
CREATE INDEX idx_booking_meal ON Booking(meal_id);

-- Index sur colonnes fréquemment filtrées
CREATE INDEX idx_flight_departure_time ON Flight(departure_time);
CREATE INDEX idx_passenger_last_name ON Passenger(last_name);
CREATE INDEX idx_airport_city ON Airport(city);
CREATE INDEX idx_booking_date ON Booking(booking_date);
```

3.3 Vue pour simplier les requêtes

```
CREATE VIEW Flight_Details AS
SELECT
    F.flight_number,
    O.city AS origin_city,
    D.city AS destination_city,
    F.departure_time,
    F.arrival_time,
    F.duration,
    F.price,
    F.status,
    F.available_seats,
```

```
    F.total_seats
FROM Flight F
JOIN Airport O ON F.origin_id = O.id
JOIN Airport D ON F.destination_id = D.id;
```

---

4. Scripts d'Insertion de Données

Données générées : 55 plats, 10 aéroports, 54 passagers, 23 vols, 54 réservations

```
SET FOREIGN_KEY_CHECKS = 0;

-- Insertion des Meals (10 enregistrements)
INSERT INTO Meal (code, name, description, price, is_active) VALUES
('VEG', 'Option Végétarienne', 'Repas sans viande ni poisson', 12.50, TRU
('CHICK', 'Poulet Rôti', 'Repas avec poulet et légumes', 15.00, TRUE),
('FISH', 'Poisson Grillé', 'Poisson blanc et riz', 16.50, TRUE),
('KID', 'Menu Enfant', 'Mini-burger et frites', 8.00, TRUE),
('GF', 'Sans Gluten', 'Option sans gluten', 14.00, TRUE),
('HALAL', 'Repas Halal', 'Viande certifiée Halal', 15.50, TRUE),
('KOSHER', 'Repas Casher', 'Repas certifié Casher', 16.00, TRUE),
('DIAB', 'Diabétique', 'Faible teneur en sucre', 13.00, TRUE),
('LOWCAL', 'Faible en Calories', 'Salade composée', 11.00, TRUE),
('NONE', 'Aucun Repas', 'Pas de repas', 0.00, FALSE);

-- Insertion des Airports (10 enregistrements)
INSERT INTO Airport (code, name, city, country) VALUES
('CDG', 'Aéroport Paris-Charles de Gaulle', 'Paris', 'France'),
('JFK', 'John F. Kennedy Airport', 'New York', 'USA'),
('DXB', 'Dubai International Airport', 'Dubai', 'UAE'),
('LHR', 'Heathrow Airport', 'Londres', 'UK'),
('NRT', 'Narita International Airport', 'Tokyo', 'Japon'),
('PEK', 'Beijing Capital Airport', 'Pékin', 'Chine'),
('FRA', 'Frankfurt Airport', 'Francfort', 'Allemagne'),
('MAD', 'Madrid-Barajas Airport', 'Madrid', 'Espagne'),
('YUL', 'Montréal-Trudeau Airport', 'Montréal', 'Canada'),
('BKK', 'Suvarnabhumi Airport', 'Bangkok', 'Thaïlande');

-- [Les 55 passagers et 23 vols sont insérés de manière similaire...]

SET FOREIGN_KEY_CHECKS = 1;
```

---

5. Requêtes SQL Documentées

5.1 Catégorie 1 : Requêtes de Base (5 requêtes)

Requête 1 : Vols au départ de Paris pour demain

```
-- Objectif: Liste des vols depuis CDG pour le lendemain
SELECT
    F.flight_number,
    O.city AS Ville_Depart,
    D.city AS Ville_Arrivee,
    F.departure_time,
    F.price
FROM Flight F
JOIN Airport O ON F.origin_id = O.id
JOIN Airport D ON F.destination_id = D.id
WHERE O.code = 'CDG'
  AND DATE(F.departure_time) = DATE(DATE_ADD(NOW(), INTERVAL 1 DAY))
ORDER BY F.departure_time;
```

Résultat attendu : Vols YN101, YN102, YN105 avec leurs horaires et prix.

Requête 2 : Passagers d'un vol spécique

```
-- Objectif: Liste des passagers ayant réservé le vol YN101
SELECT
    P.first_name,
    P.last_name,
    B.booking_reference,
    B.number_of_passengers
FROM Passenger P
JOIN Booking B ON P.id = B.passenger_id
JOIN Flight F ON B.flight_id = F.id
WHERE F.flight_number = 'YN101' ORDER
BY P.last_name;
```

Résultat attendu : 10 passagers avec leurs références de réservation.

Requête 3 : Revenu total par mois

```
-- Objectif: Calcul des revenus mensuels confirmés
SELECT
```

```
    YEAR(booking_date) AS Annee,
MONTH(booking_date) AS Mois,
    SUM(total_price) AS Total_Revenu
FROM Booking
WHERE status = 'CONFIRMED'
GROUP BY Annee, Mois
ORDER BY Annee, Mois;
```

Requête 4 : Aéroports les plus fréquentés

```
-- Objectif: Classement des aéroports par nombre de départs
SELECT
    A.name AS Aeroport,
    COUNT(F.id) AS Nombre_Vols_Depart
FROM Airport A
JOIN Flight F ON A.id = F.origin_id
GROUP BY A.name
ORDER BY Nombre_Vols_Depart DESC LIMIT
10;
```

Requête 5 : Distribution des vols par statut

```
-- Objectif: Décompte des vols par statut SELECT
    status AS Statut_Vol,
COUNT(id) AS Nombre_Vols
FROM Flight
GROUP BY status
ORDER BY Nombre_Vols DESC;
```

5.2 Catégorie 2 : Jointures Complexes (3 requêtes)

Requête 6 : Itinéraire complet d'un passager

```
-- Objectif: Tous les vols réservés par Jean Dupont (id=1)
SELECT
    P.first_name,
    P.last_name,
    B.booking_reference,
    F.flight_number,
    O.city AS Ville_Depart,
    D.city AS Ville_Arrivee,
```

```
    F.departure_time
FROM Passenger P
JOIN Booking B ON P.id = B.passenger_id
JOIN Flight F ON B.flight_id = F.id
JOIN Airport O ON F.origin_id = O.id
JOIN Airport D ON F.destination_id = D.id
WHERE P.id = 1
ORDER BY F.departure_time;
```

Requête 7 : Revenus par route

```
-- Objectif: Revenu total par paire origine-destination
SELECT
    O.city AS Ville_Depart,
    D.city AS Ville_Arrivee,
    SUM(B.total_price) AS Revenu_Total_Route,
    COUNT(B.id) AS Nombre_Reservations
FROM Booking B
JOIN Flight F ON B.flight_id = F.id
JOIN Airport O ON F.origin_id = O.id
JOIN Airport D ON F.destination_id = D.id
WHERE B.status = 'CONFIRMED'
GROUP BY O.city, D.city
ORDER BY Revenu_Total_Route DESC;
```

Requête 8 : Taux d'occupation vers New York

```
-- Objectif: Occupation des vols vers New York
SELECT
    D.city AS Destination,
    F.flight_number,
    F.total_seats,
    F.available_seats,
    (F.total_seats - F.available_seats) AS Sieges_Occupes,
    ROUND(((F.total_seats - F.available_seats) / F.total_seats) * 100, 2)
FROM Flight F
JOIN Airport D ON F.destination_id = D.id
WHERE D.city = 'New York'
ORDER BY F.departure_time;
```

5.3 Catégorie 3 : Requêtes Analytiques (3 requêtes)

Requête 9 : Top 10 des routes rentables

```
-- Objectif: Routes avec le prix moyen le plus élevé
SELECT
    O.city AS Ville_Depart,
    D.city AS Ville_Arrivee,
    AVG(F.price) AS Prix_Moyen_Vol
FROM Flight F
JOIN Airport O ON F.origin_id = O.id
JOIN Airport D ON F.destination_id = D.id
GROUP BY O.city, D.city
ORDER BY Prix_Moyen_Vol DESC LIMIT
10;
```

Requête 10 : Saisonnalité des réservations

```
-- Objectif: Nombre de réservations par mois (année en cours)
SELECT
    MONTH(booking_date) AS Mois,
    COUNT(id) AS Nombre_Reservations
FROM Booking
WHERE status = 'CONFIRMED'
  AND YEAR(booking_date) = YEAR(NOW())
GROUP BY Mois
ORDER BY Mois;
```

Requête 11 : Taux de remplissage par type d'avion

```
-- Objectif: Performance par catégorie de capacité
SELECT
    CASE
        WHEN total_seats <= 120 THEN 'Petit Porteur (≤120)'
        WHEN total_seats <= 200 THEN 'Moyen Porteur (121-200)'
        ELSE 'Grand Porteur (>200)'
    END AS Type_Avion,
    ROUND(AVG(((total_seats - available_seats) / total_seats) * 100), 2)
    COUNT(id) AS Nombre_Vols
FROM Flight
GROUP BY Type_Avion
ORDER BY Taux_Remplissage_Pct DESC;
```

5.4 Catégorie 4 : Nouvelle Fonctionnalité Meals (4 requêtes)

Requête 12 : Plats les plus commandés

```
-- Objectif: Classement des plats par popularité
SELECT
    M.name AS Nom_Plat,
    COUNT(B.id) AS Nombre_Commandes
FROM Booking B
JOIN Meal M ON B.meal_id = M.id
WHERE B.status = 'CONFIRMED'
  AND M.is_active = TRUE
GROUP BY M.name
ORDER BY Nombre_Commandes DESC;
```

Requête 13 : Plats pour un vol spécique

```
-- Objectif: Répartition des plats sur le vol YN101
SELECT
    M.name AS Nom_Plat,
    COUNT(B.id) AS Nombre_Commandes
FROM Booking B
JOIN Flight F ON B.flight_id = F.id
JOIN Meal M ON B.meal_id = M.id WHERE
F.flight_number = 'YN101' GROUP BY
M.name
ORDER BY Nombre_Commandes DESC;
```

Requête 14 : Revenu généré par les repas

```
-- Objectif: Revenu total des options de repas
SELECT
    SUM(M.price) AS Revenu_Total_Plats
FROM Booking B
JOIN Meal M ON B.meal_id = M.id
WHERE B.status = 'CONFIRMED';
```

Requête 15 : Passagers sans plat sélectionné

```
-- Objectif: Identifier les passagers sans repas (vol YN101)
SELECT
```

```
    P.first_name,
    P.last_name,
    B.booking_reference
FROM Passenger P
JOIN Booking B ON P.id = B.passenger_id
JOIN Flight F ON B.flight_id = F.id
WHERE F.flight_number = 'YN101'
  AND B.meal_id IS NULL
ORDER BY P.last_name;
```

---

6. Documentation de la Nouvelle Fonctionnalité

6.1 Description détaillée

La fonctionnalité **Meal** permet de gérer un catalogue de repas disponibles pour les vols. Chaque repas possède un code unique, un nom, une description et un prix. Les passagers peuvent sélectionner un repas lors de la réservation.

6.2 Cas d'usage

. **Gestion du catalogue :** CRUD des plats via l'ORM
. **Sélection lors de la réservation :** Le passager choisit un meal_id optionnel
. **Analyse des préférences :** Statistiques sur les plats commandés
. **Facturation :** Le prix du plat est inclus dans total_price de Booking

6.3 Règles de gestion

Chaque plat a un code et un nom uniques

Le prix ne peut être négatif

Un plat peut être désactivé sans suppression

La sélection d'un plat est optionnelle

6.4 Contraintes d'intégrité

Clé Primaire : Meal.id (AUTO_INCREMENT) **Clé**
Étrangère : Booking.meal_id → Meal.id
ON DELETE SET NULL : Préserve les réservations si un plat est supprimé
UNIQUE : Meal.code
CHECK : Meal.price ≥ 0

7. Optimisation et Performance

7.1 Index créés et justications

Index	Table	Colonne	Justification
idx_ ight_origin	Flight	origin_id	Jointures Airport-Flight (requête 1)
idx_ ight_destination	Flight	destination_id	Jointures Airport-Flight (requête 8)
idx_booking_ ight	Booking	ight_id	Jointure Booking-Flight (80% requêtes)
idx_booking_passenger	Booking	passenger_id	Itinéraire passager (requête 6)
idx_booking_meal	Booking	meal_id	Requêtes fonctionnalité Meal (12-15)
idx_ ight_departure_time	Flight	departure_time	Filtres temporels (requêtes 1, 10)
idx_booking_date	Booking	booking_date	Analyses saisonnalité (requête 10)
idx_airport_city	Airport	city	Recherches par ville (requête 7, 8)

7.2 Analyse EXPLAIN (Requête 7 - Revenus par route)

```
EXPLAIN SELECT
    O.city, D.city, SUM(B.total_price)
FROM Booking B
JOIN Flight F ON B.flight_id = F.id
JOIN Airport O ON F.origin_id = O.id
JOIN Airport D ON F.destination_id = D.id
WHERE B.status = 'CONFIRMED' GROUP
BY O.city, D.city;
```

Impact des index :

- Sans index : Full table scan sur Booking (35 lignes × 3 jointures = 105 comparaisons)
- Avec index : Accès direct via clés étrangères (complexité O(log n))

---

8. Conclusion

8.1 Bilan du projet

Ce projet a permis d'appliquer les compétences en administration de bases de données :

Modélisation cohérente avec intégration de la fonctionnalité Meal Implémentation de 50 passagers, 23 vols, 35 réservations avec intégrité référentielle 15 requêtes SQL couvrant bases, jointures complexes et analyses Optimisation via 8 index stratégiques

8.2 Difficulté rencontrées

Cohérence des données : Assurer la validité des dates, heures et références de clés étrangères
Intégrité référentielle : Gestion de `ON DELETE SET NULL` pour meal_id sans casser les réservations

Équilibrage index : Éviter la sur-indexation tout en couvrant les requêtes critiques

8.3 Améliorations possibles

. **Table Airline :** Gérer plusieurs compagnies aériennes
. **Historisation des prix :** Table d'audit pour analyser l'évolution tarifaire
. **Triggers :** Mise à jour automatique de available_seats lors des INSERT/DELETE sur Booking
. **Vues matérialisées :** Pré-calculer les statistiques pour les requêtes analytiques fréquentes

---

Annexes

Structure des fichiers SQL

SQL/
├── 01_create_tables.sql # Tables, contraintes, index
├── 02_insert_data.sql # Données de test (Mockaroo)
├── 03_queries.sql # 15 requêtes documentées
└── 04_indexes_optimization.sql # Analyse EXPLAIN

GitHub du projet

https://github.com/aissamerfouk422-gif/ynov-air.git