C\_306

Gestion d’un aéroport

Christopher Ristic

Ristic Christopher – FID1

ETML / 1004 LAUSANNEVennes 12 1004 Lausanne

Table des matières

[1 Introduction 4](#_Toc197931338)

[1.1 Objectif 4](#_Toc197931339)

[1.2 Description du fonctionnement de l’aéroport 4](#_Toc197931340)

[2 Travail à réaliser 6](#_Toc197931341)

[2.1 Réaliser la modélisation avec la méthode MERISE 6](#_Toc197931342)

[2.2 Création des tables de la base de données 6](#_Toc197931343)

[2.3 Charger des données 6](#_Toc197931344)

[2.4 Mettre en place une stratégie de sauvegardes et de restauration 7](#_Toc197931345)

[2.5 Requêtes SQL 8](#_Toc197931346)

[2.6 Index 9](#_Toc197931347)

[2.7 Utilisateurs et rôles 9](#_Toc197931348)

[2.8 Transaction 11](#_Toc197931349)

[2.9 Utilisation de l’IA 11](#_Toc197931350)

[3 La modélisation avec la méthode MERISE 12](#_Toc197931351)

[4 Création des tables de la base de données 13](#_Toc197931352)

[4.1 Commandes 13](#_Toc197931353)

[5 Charger des données 14](#_Toc197931354)

[5.1 Table aeroport 14](#_Toc197931355)

[5.2 Table passager 15](#_Toc197931356)

[5.3 Table compagnie 15](#_Toc197931357)

[5.4 Table type avion 15](#_Toc197931358)

[5.5 Table avion 15](#_Toc197931359)

[5.6 Table programme vol 15](#_Toc197931360)

[5.7 Table vol 16](#_Toc197931361)

[5.8 Table reservation 16](#_Toc197931362)

[6 Mettre en place une stratégie de sauvegardes et de restauration 16](#_Toc197931363)

[6.1 Sauvegarde complète 16](#_Toc197931364)

[6.2 Restauration 16](#_Toc197931365)

[7 Requêtes SQL 16](#_Toc197931366)

[7.1 Requête 1 16](#_Toc197931367)

[7.2 Requête 2 16](#_Toc197931368)

[7.3 Requête 3 17](#_Toc197931369)

[7.4 Requête 4 17](#_Toc197931370)

[7.5 Requête 5 17](#_Toc197931371)

[7.6 Requête 6 17](#_Toc197931372)

[7.7 Requête 7 17](#_Toc197931373)

[7.8 Requête 8 17](#_Toc197931374)

[7.9 Requête 9 17](#_Toc197931375)

[7.10 Requête 10 18](#_Toc197931376)

[8 Index 18](#_Toc197931377)

[8.1 Requête 1 18](#_Toc197931378)

[8.1.1 Analyse 18](#_Toc197931379)

[8.1.2 Index 18](#_Toc197931380)

[8.2 Requête 2 18](#_Toc197931381)

[8.2.1 Analyse 18](#_Toc197931382)

[8.2.2 Index 18](#_Toc197931383)

[9 Utilisateurs et rôles 19](#_Toc197931384)

[9.1 Privilège pour chaque rôle 19](#_Toc197931385)

[9.1.1 Administrateur 19](#_Toc197931386)

[9.1.2 Gestionnaire des Vols 19](#_Toc197931387)

[9.1.3 Agent de Réservation 19](#_Toc197931388)

[9.1.4 Agent de Comptoir 19](#_Toc197931389)

[9.1.5 Analyste 19](#_Toc197931390)

[9.2 Utilisateur et association de rôle 19](#_Toc197931391)

[9.2.1 Administrateur 19](#_Toc197931392)

[9.2.2 Gestionnaire des Vols 20](#_Toc197931393)

[9.2.3 Agent de Réservation 20](#_Toc197931394)

[9.2.4 Agent de Comptoir 20](#_Toc197931395)

[9.2.5 Analyste 20](#_Toc197931396)

[10 Transaction 20](#_Toc197931397)

[10.1 Cas n°1 20](#_Toc197931398)

[10.2 Cas n°2 20](#_Toc197931399)

[11 Conclusion 21](#_Toc197931400)

[11.1 Conclusion personnelle 21](#_Toc197931401)

[Table des illustrations 22](#_Toc197931402)

[Bibliographie 23](#_Toc197931403)

# 

# Introduction

Un aéroport est un espace complexe où interagissent plusieurs entités : passagers, compagnies aériennes, avions, vols et infrastructures. Pour assurer un fonctionnement efficace, chaque processus repose sur un système de gestion centralisé utilisant une base de données relationnelle.

## Objectif

Modéliser une base de données d’un aéroport, réaliser des requêtes SQL, etc

## ****Description du fonctionnement de l’aéroport****

L’aéroport international **SkyConnect** repose sur une infrastructure complexe qui permet d’assurer un suivi rigoureux des vols et des passagers. Pour garantir une gestion fluide des opérations, un système de gestion de bases de données a été conçu. Ce système repose sur plusieurs concepts clés permettant d’optimiser l’exploitation des ressources aéroportuaires et d’assurer la coordination entre les différents acteurs du transport aérien.

Le premier élément fondamental est la gestion des aéroports. Chaque aéroport du monde est identifié par deux codes :

* le code **IATA**, composé de trois lettres et utilisé par les passagers et les compagnies aériennes pour identifier facilement un aéroport (comme CDG pour Paris Charles de Gaulle). Ce code n’est pas unique.
* le code **ICAO**, composé de quatre lettres et utilisé principalement par les systèmes de navigation aérienne et les contrôleurs du trafic aérien (comme LFPG pour CDG). Ce code est unique.

Ces codes permettent d’éviter toute confusion et d’uniformiser la communication à l’échelle internationale. Chaque aéroport enregistré dans le système possède un nom officiel, qui est utilisé dans les rapports et les documents administratifs.

Les compagnies aériennes sont les acteurs principaux du transport aérien. Elles possèdent un code **IATA** à deux lettres, leur permettant d’être identifiées rapidement dans les billets d’avion et les systèmes de réservation (par exemple, AF pour Air France ou LH pour Lufthansa). Chaque compagnie est enregistrée sous son nom officiel et est associée à un aéroport principal, qui est sa base d’opérations. Cette information est essentielle car elle permet de savoir où une compagnie entretient ses avions et où se situent ses centres de maintenance.

Les compagnies aériennes exploitent une flotte d’avions, qui sont des appareils spécifiques possédant des caractéristiques bien définies. Chaque avion appartient à une compagnie aérienne et est définit par un type d’avion. Par exemple, un Airbus A320 et un Boeing 747 sont deux types d’avions distincts. Le type d’avion définit un modèle générique et possède une description détaillée. La capacité d’un avion peut être différente entre deux mêmes modèles d’avion. Lorsqu’un avion est ajouté au système, il est associé à un type d’avion et à une compagnie, ce qui permet de suivre précisément quel appareil appartient à quelle entreprise et de planifier son utilisation.

Un des éléments essentiels de la gestion du trafic aérien est la planification des vols. Il est important de bien comprendre la distinction entre un programme de vol et un vol réel.

Un programme de vol est une planification régulière qui définit quels vols doivent être opérés à quelles dates et à quels horaires. Par exemple, le vol AF001 entre Paris et New York peut être programmé tous les lundis, mercredis et vendredis à 10h00.

Un programme de vol n’a pas d’avion attribué à l’avance. Il s'agit d'un plan, qui ne dépend pas d'un avion précis.

Un programme de vol est une règle générale qui définit :

* Une route aérienne (départ & arrivée).
* Une fréquence (jours de la semaine).
* Des horaires prévus.
* Une compagnie exploitante.

Exemple :

Air France programme un vol entre Paris et New York.

AF001 : CDG → JFK

Départ prévu : 10h00

Opéré tous les : Lundi, Mercredi, Vendredi

Aucun avion n’est attribué à l’avance.

Ce programme existe avant même qu’un avion soit affecté à un vol précis.

Chaque vol réel suit un programme, mais il est lié à :

* Un jour précis.
* Un avion spécifique.
* Un horaire réel de départ et d’arrivée.

Exemple :

Le vol AF001 du 12 juin 2025 a été opéré.

Avion : Airbus A380 immatriculé F-HPJA

Départ réel : 10h05

Arrivée réelle : 13h15

Compagnie exploitante : Air France

Ce vol est une application concrète du programme de vol.

Chaque vol est opéré par une compagnie aérienne spécifique et un avion précis. Il est également rattaché à un aéroport de départ et un aéroport d’arrivée, qui définissent son itinéraire et qui peuvent différer du programme de vol. Les horaires de départ et d’arrivée prévus sont renseignés, ce qui permet aux passagers et aux équipes techniques d’organiser leurs opérations. La gestion de ces informations est cruciale pour éviter les conflits de vols et garantir que les avions sont disponibles aux bons moments. Mais en cas de problème, l’heure de départ et/ou l’heure d’arrivée réelles peuvent bien sûr différer des heures planifiées dans le programme de vol.

Les passagers constituent une autre entité essentielle du système. Chaque passager est enregistré avec un numéro de passeport unique, qui permet d’identifier la personne de manière certaine. Il possède également un nom et un prénom, qui sont utilisés dans les billets d’avion et les systèmes de contrôle à l’aéroport. Le suivi des passagers est important non seulement pour la gestion des réservations, mais aussi pour des raisons de sécurité et de conformité aux réglementations aériennes.

Le système de réservations permet aux passagers de réserver un siège spécifique sur un vol donné. Une réservation est associée à un passager et à un vol précis, garantissant que chaque voyageur dispose d’un siège attribué. Chaque réservation comprend également un prix, qui reflète le coût du billet pour ce vol particulier. Grâce à cette base de données, il est possible de suivre qui a réservé quel vol, d’optimiser le remplissage des avions et de générer des rapports sur les tendances de voyage.

L’interconnexion entre ces différentes entités assure une gestion fluide et efficace du trafic aérien. En utilisant ce modèle, l’aéroport **SkyConnect** peut suivre chaque vol en temps réel, attribuer correctement les avions aux compagnies, gérer les horaires des vols, assurer la coordination entre les passagers et les compagnies aériennes, et garantir une expérience de voyage optimale. Ce système permet également d’optimiser la gestion des ressources aéroportuaires en s’assurant que les avions sont bien utilisés, que les compagnies respectent leurs créneaux horaires et que les passagers disposent d’un suivi clair de leurs réservations.

# Travail à réaliser

Chaque étape ci-dessous doit être détaillée dans votre rapport.

## Réaliser la modélisation avec la méthode MERISE

A l’aide du texte ci-dessus, vous devez commencer par modéliser l’aéroport en utilisant la méthode **MERISE**. Pour cela, vous devez réaliser le **MCD** et le **MLD** avec l’outil **Looping**.

**Vous devez impérativement faire valider le MCD et le MLD à votre enseignant avant de passer aux étapes suivantes.**

## Création des tables de la base de données

Lorsque vos **MCD** et **MLD** seront validés, vous pourrez obtenir l’ensemble des requêtes SQL de type **CREATE TABLE** pour **MySQL** grâce à **Looping** et les exécuter dans votre base de données **db\_aeroport**.

## Charger des données

Nous devons maintenant importer des données dans nos différentes tables. Pour cela, votre enseignant vous donnera en temps et en heure un ensemble de fichiers avec l’extension **tsv** que vous devrez chargerà l’aide de la commande **LOAD DATA**.

Il est important que toutes les données soient chargées dans votre base de données.

Exemple de résultat de la commande LOAD DATA pour la table t\_reservation.

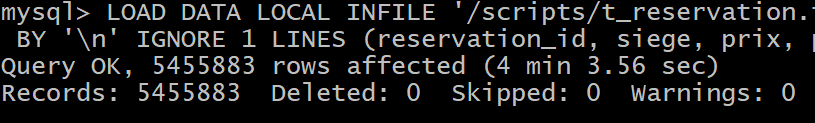


Figure 1: Résultat du LOAD DATA pour la table t\_reservation

A noter qu’il y a 0 **Skipped** et 0 **Warnings.**

Les commandes permettant de chargées doivent être présentées dans votre rapport.

## Mettre en place une stratégie de sauvegardes et de restauration

Imaginons que notre base de données **db\_aeroport** soit réellement en production. Il serait crucial de mettre en place une stratégie de sauvegardes robustes pour éviter la perte de données en cas de panne, d’erreur humaine ou de cyberattaque.

Nous partons du principe que sur cette base de données, il y a de nombreux INSERT et UPDATE qui sont réalisés régulièrement mais **jamais** de DELETE.

**Travail à réaliser :**

A l’aide de la commande **mysqldump**, vous devez mettre en place :

* Un backup complet une fois par semaine : Par exemple, chaque dimanche.
* Un backup différentiel chaque jour de la semaine à minuit.

A vous d’imaginer une solution pour mettre en place cette stratégie de sauvegardes.

**Aide :** Pour cela, vous avez le droit de modifier les tables existantes.

Une fois la stratégie de sauvegardes mis en place, n’oubliez pas de tester et d’expliquer la manière de restaurer les données.

Voici le scénario que l’on souhaite tester.

1. J’effectue mon backup complet
2. Un INSERT et un UPDATE sont effectués sur la db.
3. J’effectue le backup différentiel

A l’aide de mes backups, j’ai la capacité de restaurer la DB.

Vous devez bien sûr expliquer cela en détail dans votre rapport.

## Requêtes SQL

Maintenant, nous allons réaliser un ensemble de requêtes SQL sur notre base de données aéroport pour mettre en pratique tous les concepts étudiés dans le module 106.

Requête n°1:

Afficher les 2 premiers avions de la compagnie **compagnie\_id = 60**, triés par capacité décroissante.

Vous devez afficher toutes les colonnes.

Requête n°2:

Afficher les 3 premiers vols les plus récents (heure de départ la plus tardive).

Vous devez afficher toutes les colonnes.

Requête n°3:

Afficher le nombre total de vols opérés par chaque compagnie aérienne (compagnie\_fk).

Le résultat doit être ordonné par le nombre de vols de manière décroissante.

Aide : Vous ne devez afficher que 2 colonnes dans le SELECT.

Requête n°4:

Afficher la liste des passagers, leur numéro de passeport, ainsi que l'aéroport de départ du vol réservé.

1ère colonne : nom du passager

2ème colonne : prénom du passager

3ème colonne : numéro de passeport du passager

4ème colonne : nom de l’aéroport de départ

Requête n°5:

Affiche le prix moyen des réservations pour chaque vol (vol\_fk).

Le résultat doit être ordonné par le prix moyen du plus petit au plus grand.

Requête n°6:

Afficher tous les noms d’aéroports, ainsi que le nom des compagnies basées dans chaque aéroport (y compris ceux sans compagnie).

Requête n°7:

Afficher toutes les compagnies aériennes, avec le nombre d’avions qu’elles possèdent, même celles qui n’en ont pas. Le résultat doit être ordonné par le nombre d’avions du plus grand au plus petit.  
  
Requête n°8:

Afficher uniquement les compagnies (compagnie\_fk) ayant au moins 1500 vols.

Le résultat doit être ordonné par le nombre de vols du plus grand au plus petit et limité au 5 premier enregistrements.

Requête n°9:

Afficher le nombre de vols opérés par chaque compagnie (nom) depuis chaque aéroport de départ (nom).

Ordonner le résultat par le nom (alphabétiquement) et par le nombre de vols du plus grand au plus petit.

1ère colonne : nom de la compagnie

2ème colonne : nom de l’aéroport

3ème colonne : nombre de vols

Requête n°10:

Afficher la capacité totale des avions pour chaque compagnie (nom) et type d’avion (nom).

Ordonner le résultat par la capacité totale des avions du plus grand au plus petit.

Aide : 3 colonnes dans le SELECT.

Toutes ces requêtes doivent être présentées dans le rapport.

## Index

Soit les 2 requêtes suivantes :

Requête n°1:

SELECT V.vol\_id, V.heure\_depart, V.heure\_arrivee, C.nom AS compagnie

FROM t\_vol AS V

INNER JOIN t\_compagnie AS C ON V.compagnie\_fk = C.compagnie\_id

WHERE V.compagnie\_fk = 3 AND V.heure\_depart > '12:00:00'

ORDER BY V.heure\_depart DESC;

Requête n°2 :

SELECT A.nom AS aeroport, COUNT(V.vol\_id) AS nombre\_vols

FROM t\_vol AS V

INNER JOIN t\_aeroport AS A ON V.aeroport\_depart\_fk = A.aeroport\_id

WHERE V.heure\_depart > '10:00:00'

GROUP BY A.nom

ORDER BY nombre\_vols DESC;

**Travail à réaliser**:

Vous devez créer des index pour accélérer l’exécution de ces requêtes.

Vous devez expliquer vos choix et donner les requêtes SQL permettant la création de ces index.

## Utilisateurs et rôles

Pour garantir une gestion efficace et sécurisée de la base de données de l’aéroport, tous les utilisateurs n’ont pas accès aux mêmes informations ni aux mêmes fonctionnalités. Chaque employé joue un rôle bien défini et doit uniquement pouvoir effectuer les actions nécessaires à sa fonction. L’objectif est d’éviter les erreurs, les modifications involontaires et de protéger les données sensibles.

Dans cette base de données, plusieurs types d’utilisateurs interagissent au quotidien. Voici un aperçu des différents rôles et de leurs responsabilités.

**L'Administrateur :**

L’administrateur est le super-utilisateur du système. Il est le seul à avoir un contrôle total sur l’ensemble de la base de données. C’est lui qui crée les nouveaux comptes, définit les rôles des utilisateurs et peut modifier n’importe quelle information. Il est responsable de la sécurité et de l’intégrité des données.

Exemples d’actions qu’il peut effectuer :

* Ajouter ou supprimer un employé dans la base.
* Modifier les droits d’un utilisateur.
* Supprimer des données obsolètes ou incorrectes.

**Le Gestionnaire des Vols :**

Le gestionnaire des vols est en charge de la planification et de la modification des vols. Il doit pouvoir ajouter de nouveaux vols à l’horaire, modifier les horaires en cas de retard ou d’annulation, et attribuer un avion à un vol.

Exemples d’actions qu’il peut effectuer :

* Ajouter un vol dans la base de données.
* Modifier l’heure de départ ou d’arrivée d’un vol.
* Planifier un nouvel vol.

**L’Agent de Réservation :**

L’agent de réservation est en contact direct avec les passagers. Il doit pouvoir enregistrer de nouveaux passagers, créer des réservations et modifier celles existantes.

Exemples d’actions qu’il peut effectuer :

* Ajouter un passager dans la base de données.
* Réserver un siège pour un passager sur un vol.
* Modifier une réservation (changer un siège, modifier un nom).

**L’Agent de Comptoir :**

L’agent de comptoir est en charge de l’enregistrement des passagers et de la validation des billets. Son rôle est crucial au moment de l’embarquement.

Exemples d’actions qu’il peut effectuer :

* Vérifier une réservation et confirmer l’enregistrement d’un passager.
* Modifier le siège attribué à un passager.

Ce qu'il ne peut PAS faire :

* Il ne peut pas modifier les vols ou créer de nouvelles réservations.
* Il ne peut pas supprimer une réservation.

**L’Analyste**

L’analyste travaille sur les statistiques et la performance de l’aéroport. Il n’a aucun droit d’écriture sur la base de données, mais il peut consulter toutes les informations.

Exemples d’actions qu’il peut effectuer :

* Lire les statistiques de fréquentation des vols.
* Analyser les tendances des réservations.
* Générer des rapports de performance.

Ce qu'il ne peut PAS faire :

* Il ne peut pas modifier ou supprimer des données.
* Il ne peut pas créer de nouvelles réservations ou ajouter des vols.

**Travail à réaliser**

Dans cette section, vous devez créer les rôles avec les bons privilèges.

Vous devez également créer au moins un utilisateur par rôle.

## Transaction

**Cas n°1**: Annulation d’un vol

Lorsqu’un vol est annulé, nous allons supprimer l’enregistrement correspondant à ce vol. Mais il est également important de supprimer toutes les réservations liées à ce vol et ainsi ne pas laisser de réservations orphelines.

Nous avons donc besoin de gérer l’annulation d’un vol avec une transaction.

Vous devez proposer une manière de réaliser cela à l’aide de requête SQL.

**Cas n°2**:

Soit la transaction suivante :

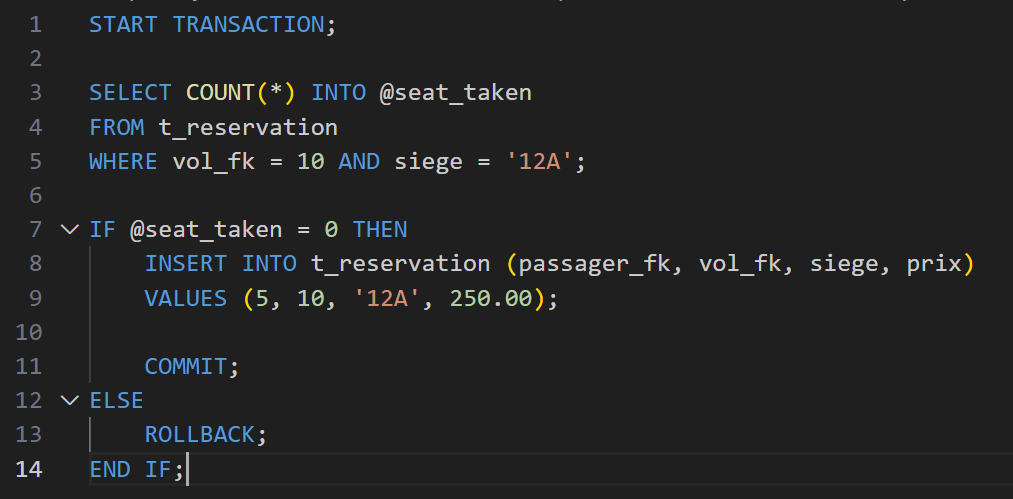


Figure 2: Transaction à expliquer

Vous devez expliquer ce que fait cette transaction.

## Utilisation de l’IA

Le recours à des outils en ligne d’intelligence artificielle (ex. : Chat GPT) doit être mentionné dans le rapport de manière détaillée et explicite et ne peut servir que d’inspiration à la réalisation ou d’aide à l’apprentissage. En cas d’abus, l’évaluation du projet en tiendra compte.

# La modélisation avec la méthode MERISE

Pour le MCD, la particularité a été de déterminer la position du jour de la semaine. Comme on le voit sur la Figure 4: MCD, les jours de la semaine sont devenu une table, ce qui veut dire qu’un programme ne peut pas être à nouveau le même jour. L’association "prévoir" devient une table qui contient un id du programme et un id du jour de la semaine.

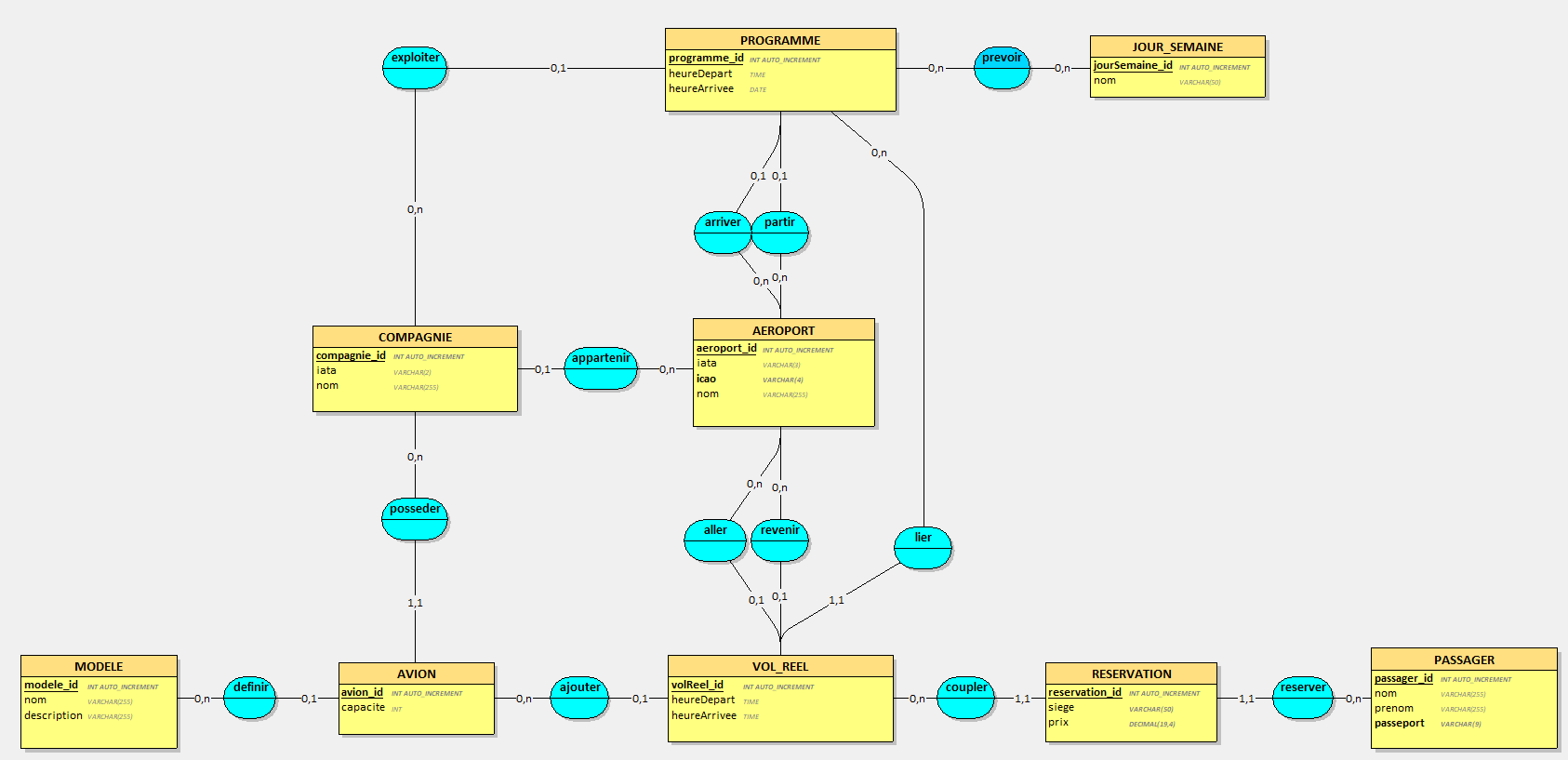


Figure 3: MCD

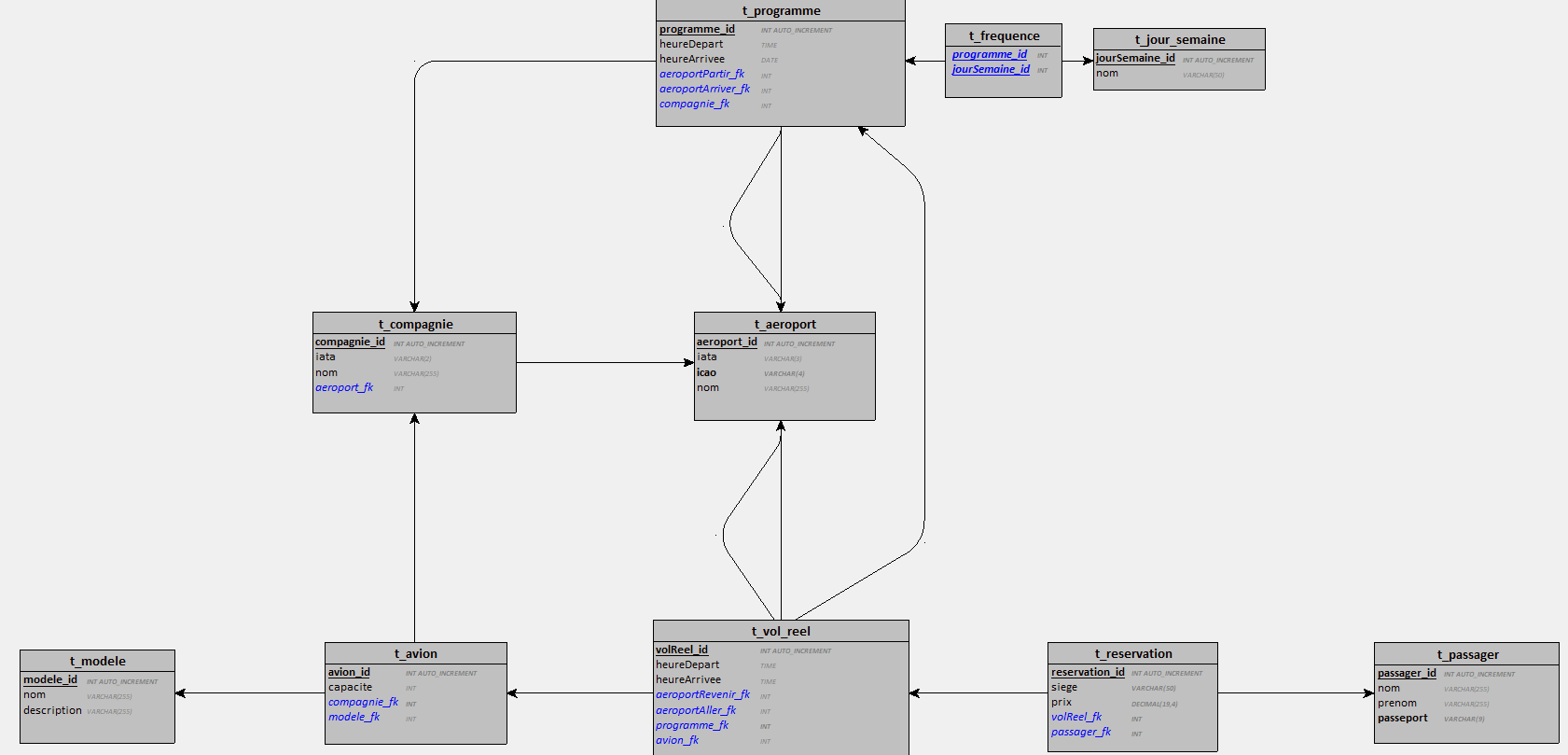


Figure 4: MLD

# Création des tables de la base de données

## Commandes

Pour se connecter à la base de données dans docker :

Création de la base de données :

Pour créer les tables, il faut aller dans PHPmyAdmin, sélectionner la base de donnée puis aller dans la fenêtre SQL, et copier-coller le contenu du fichier CREATE\_TABLE\_7.sql

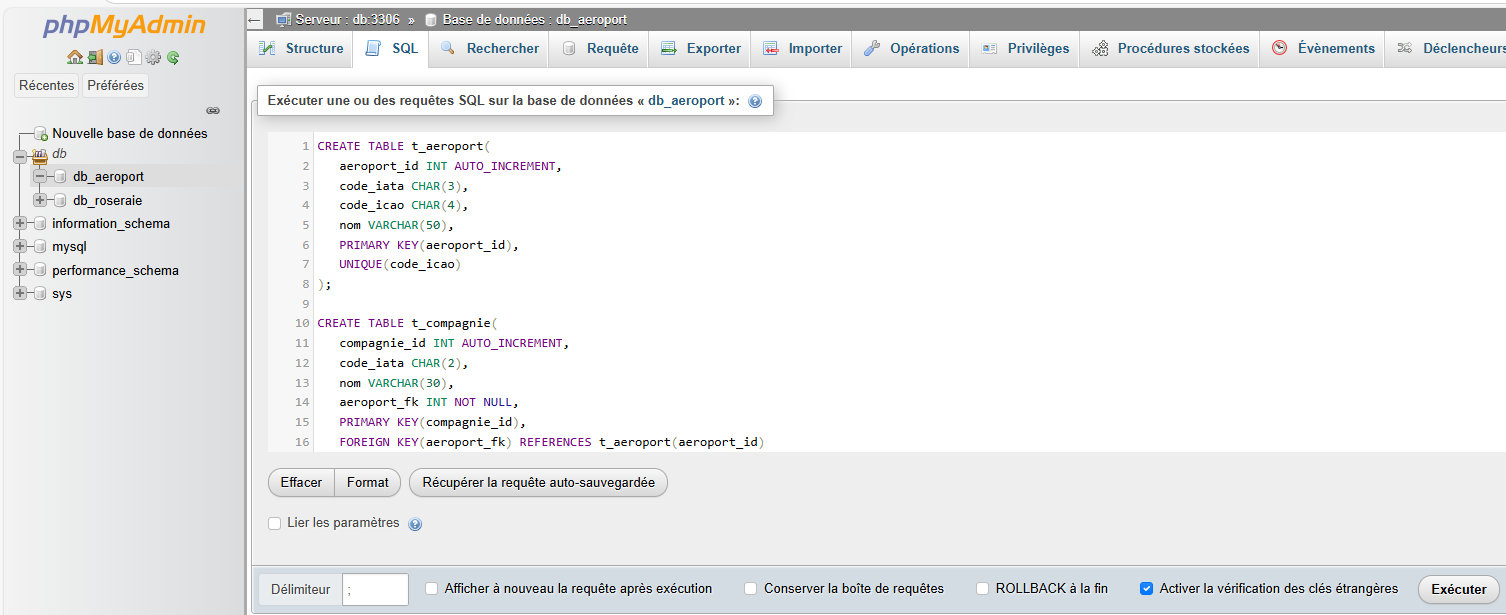


Figure 5: Création des tables

Après exécution, les tables ont bien été créé avec aucune erreur.

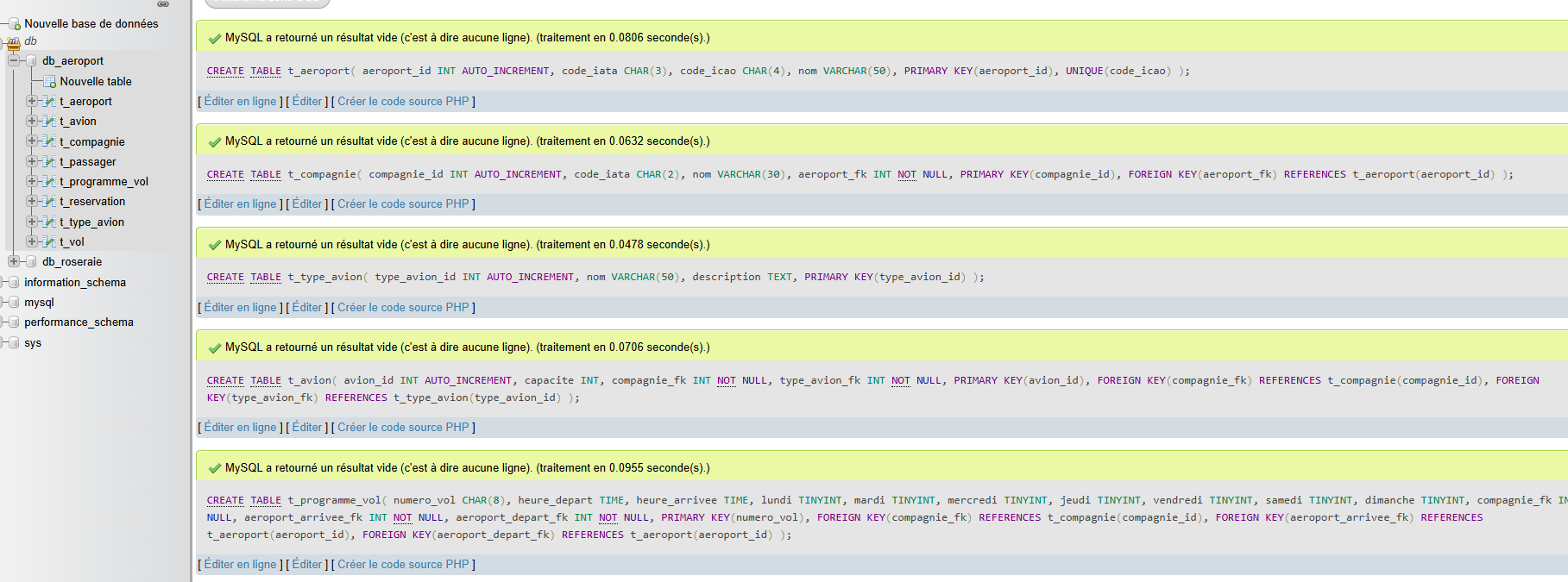


Figure 6: Résultat création tables

# Charger des données

Avant de commencer à charger les données, on regarde les dépendances qui détermineront par quelle table il faut commencer à importer les données. De gauche à droite on a :

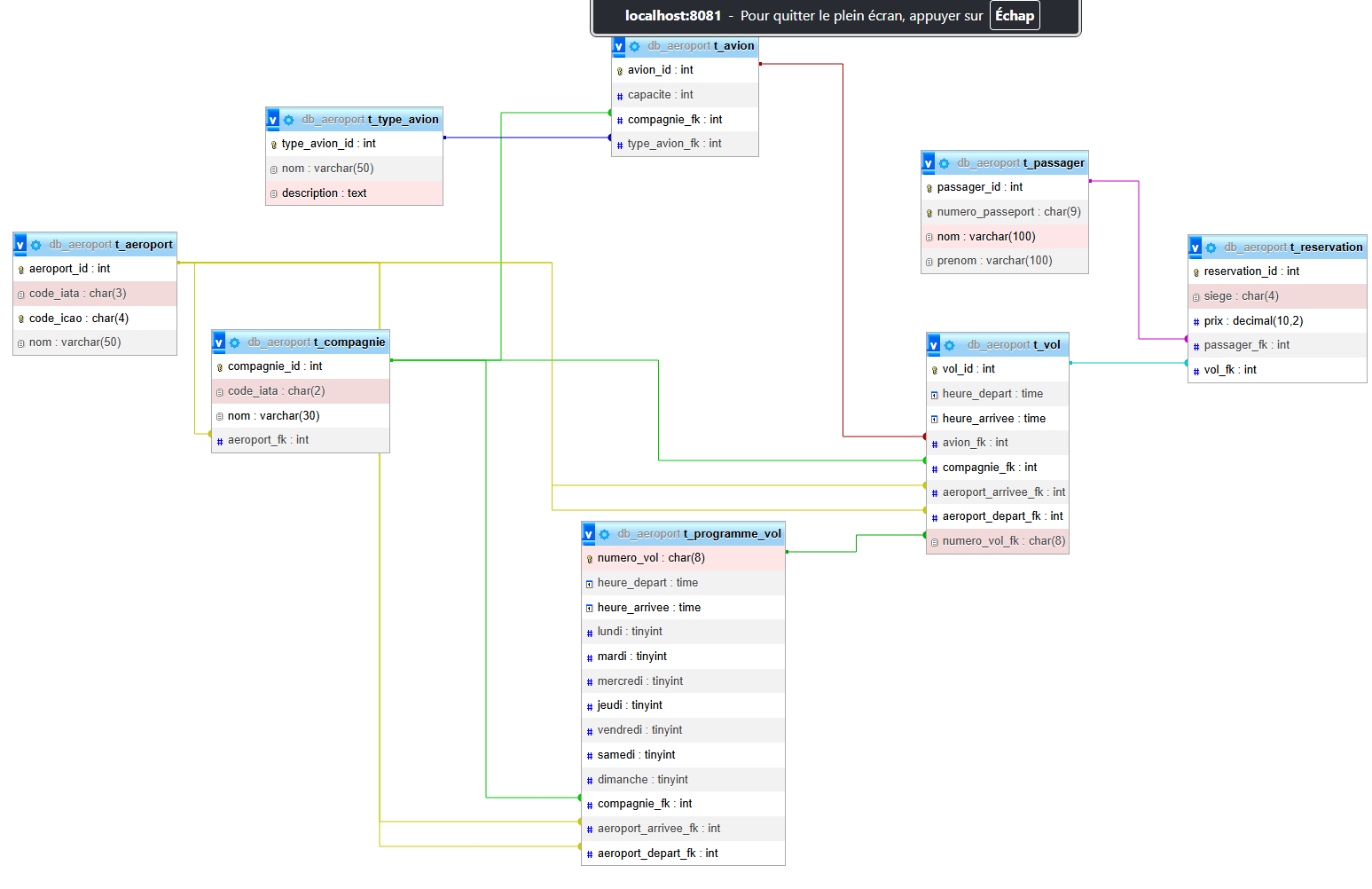


Figure 7: Schéma aeroport

Pour chaque table il faut adapter la commande pour importer les données en vérifiant le contenu des fichiers qui contiennent les données.

## Table aeroport

Après avoir essayé la commande :

**LOAD DATA LOCAL INFILE 't\_aeroport.tsv'  
INTO TABLE t\_aeroport   
IGNORE 1 LINES;**

Une erreur est survenue “loading local data is disabled this must be enabled”, il faut changer la valeur de local\_infile tel que :

**SET GLOBAL local\_infile = 1;**

Maintenant il y a une autre erreur de permission, il faut exécuter ceci :

**docker exec -it db bash**

**mysql --local-infile=1 -u root -p**

Pour pouvoir accèder au fichier tsv, il doit être placé dans le dossier monté de docker : /scripts

Pour prendre en charge les valeurs NULL dans la base de données lors du LOAD DATA, on doit intégrer la commande suivante :

**NULLIF(colonne, 'NULL');**

La commande finale sera :

**LOAD DATA LOCAL INFILE 't\_aeroport.tsv' INTO TABLE t\_aeroport IGNORE 1 LINES**

**(@col1, @col2, @col3, @col4)**

**SET aeroport\_id = NULLIF(@col1, 'NULL'),**

**code\_iata = NULLIF(@col2, 'NULL'),**

**code\_icao = NULLIF(@col3, 'NULL'),**

**nom = NULLIF(@col4, 'NULL');**

## Table passager

LOAD DATA LOCAL INFILE 't\_passager.tsv' INTO TABLE t\_passager IGNORE 1 LINES

## Table compagnie

LOAD DATA LOCAL INFILE 't\_compagnie.tsv' INTO TABLE t\_compagnie IGNORE 1 LINES

## Table type avion

LOAD DATA LOCAL INFILE 't\_type\_avion.tsv' INTO TABLE t\_type\_avion IGNORE 1 LINES

## Table avion

LOAD DATA LOCAL INFILE 't\_avion.tsv' INTO TABLE t\_avion IGNORE 1 LINES

(@col1, @col2, @col3, @col4)

SET avion\_id = NULLIF(@col1, 'NULL'),

capacite = NULLIF(@col2, 'NULL'),

compagnie\_fk = NULLIF(@col3, 'NULL'),

type\_avion\_fk = NULLIF(@col4, 'NULL') ;

## Table programme vol

LOAD DATA LOCAL INFILE 't\_programme\_vol.tsv' INTO TABLE t\_programme\_vol IGNORE 1 LINES

## Table vol

LOAD DATA LOCAL INFILE 't\_vol.tsv' INTO TABLE t\_vol IGNORE 1 LINES;

## Table reservation

LOAD DATA LOCAL INFILE 't\_reservation.tsv' INTO TABLE t\_reservation IGNORE 1 LINES;

# Mettre en place une stratégie de sauvegardes et de restauration

## Sauvegarde complète

Une solution possible serait de suivre la règle suivante :

* 1 sauvegarde externe
* Sur 2 supports différents
* 3 copies des données

Et de faire les sauvegardes la nuit, pour éviter de saturer le réseau.

Ensuite la commande pour faire la sauvegarde complète :

* mysqldump -u [utilisateur] -p [nom\_db] > [nomBackup].sql

## Restauration

Pour restaurer les données dans la base de données, il faut rajouter la ligne suivante à la première ligne du fichier backup :

* USE [Nom\_de\_la\_db];

Puis la commande pour restaurer :

• mysql -u [utilisateur] -p < [nomBackup].sql

# Requêtes SQL

## Requête 1

SELECT \*

FROM `t\_avion`

JOIN t\_compagnie as c

ON compagnie\_fk = c.compagnie\_id

WHERE c.compagnie\_id = 60

ORDER BY capacite DESC

LIMIT 2;

## Requête 2

SELECT \*

FROM t\_vol

ORDER BY heure\_depart DESC

LIMIT 3;

## Requête 3

SELECT COUNT(numero\_vol) AS nombre\_de\_vols, nom

FROM t\_programme\_vol

JOIN t\_compagnie

ON t\_compagnie.compagnie\_id = t\_programme\_vol.compagnie\_fk

GROUP BY nom

ORDER BY nombre\_de\_vols DESC;

## Requête 4

SELECT t\_passager.nom, prenom, numero\_passeport, t\_aeroport.nom AS Departure\_Airport

FROM t\_passager

JOIN t\_reservation

ON t\_passager.passager\_id = t\_reservation.passager\_fk

JOIN t\_vol

ON t\_vol.vol\_id = t\_reservation.vol\_fk

JOIN t\_aeroport

ON t\_vol.aeroport\_depart\_fk = t\_aeroport.aeroport\_id;

## Requête 5

SELECT AVG(prix), vol\_fk

From t\_reservation

GROUP BY vol\_fk

ORDER BY AVG(prix) ASC

## Requête 6

SELECT t\_aeroport.nom, t\_compagnie.nom

FROM t\_aeroport

LEFT JOIN t\_compagnie

ON t\_aeroport.aeroport\_id = t\_compagnie.aeroport\_fk ;

## Requête 7

SELECT t\_compagnie.nom, COUNT(a.avion\_id) as Nb\_avion

FROM `t\_compagnie`

LEFT JOIN t\_avion AS a

ON a.compagnie\_fk = t\_compagnie.compagnie\_id

GROUP BY t\_compagnie.nom;

## Requête 8

SELECT t\_vol.compagnie\_fk, COUNT(vol\_id) AS Nb\_vol

FROM t\_vol

GROUP BY t\_vol.compagnie\_fk

HAVING Nb\_vol >= 1500

ORDER BY Nb\_vol DESC

LIMIT 5;

## Requête 9

SELECT t\_compagnie.nom, a.nom, COUNT(v.vol\_id) as Nb\_vol

FROM `t\_compagnie`

JOIN t\_aeroport AS a

ON a.aeroport\_id = aeroport\_fk

JOIN t\_vol AS v

ON v.compagnie\_fk = compagnie\_id

GROUP BY t\_compagnie.nom, a.nom

ORDER BY t\_compagnie.nom ASC, Nb\_vol DESC;

## Requête 10

SELECT DISTINCT t\_compagnie.nom, t\_type\_avion.nom, capacite

FROM t\_avion

JOIN t\_compagnie

On t\_compagnie.compagnie\_id = t\_avion.compagnie\_fk

JOIN t\_type\_avion

On t\_avion. type\_avion\_fk = t\_type\_avion.type\_avion\_id

ORDER by capacite DESC;

# Index

## Requête 1

SELECT V.vol\_id, V.heure\_depart, V.heure\_arrivee, C.nom AS compagnie

FROM t\_vol AS V

INNER JOIN t\_compagnie AS C ON V.compagnie\_fk = C.compagnie\_id

WHERE V.compagnie\_fk = 3 AND V.heure\_depart > '12:00:00'

ORDER BY V.heure\_depart DESC;

La requête prend 0.0049 secondes pour s’exécuter.

### Analyse

La requête comporte un filtrage avec le mot clef WHERE, il faudra créer un index pour accélérer celui-ci.

### Index

CREATE INDEX idx\_vol\_compagnie\_fk\_heure\_depart

ON t\_vol(compagnie\_fk, heure\_depart);

Après avoir créé l’index, le temps d’exécution de la même requête diminue à 0.0005 secondes.

## Requête 2

SELECT A.nom AS aeroport, COUNT(V.vol\_id) AS nombre\_vols

FROM t\_vol AS V

INNER JOIN t\_aeroport AS A ON V.aeroport\_depart\_fk = A.aeroport\_id

WHERE V.heure\_depart > '10:00:00'

GROUP BY A.nom

ORDER BY nombre\_vols DESC;

La requête prend 0.2319 secondes pour s’exécuter.

### Analyse

La requête comporte un filtrage avec le mot clef WHERE, il faudra créer un index pour accélérer celui-ci.

### Index

CREATE INDEX idx\_vol\_aeroport\_fk\_heure\_depart

ON t\_vol(aeroport\_depart\_fk, heure\_depart);

Après avoir créé l’index, le temps d’exécution de la même requête diminue à 0.0995 secondes.

# Utilisateurs et rôles

## Privilège pour chaque rôle

### Administrateur

**A tous les privilèges.**

CREATE ROLE 'admin';

GRANT ALL PRIVILEGES ON \*.\* TO 'admin' WITH GRANT OPTION;

### Gestionnaire des Vols

Peut **ajouter, modifier, supprimer des vols**, mais pas gérer les utilisateurs. Il a accès uniquement aux tables relatives aux vols.

CREATE ROLE 'gestionnaire\_vols';

GRANT INSERT, UPDATE, DELETE ON `t\_vol` TO 'gestionnaire\_vols';

GRANT SELECT ON `t\_vol` TO 'gestionnaire\_vols';

### Agent de Réservation

**Peut ajouter et modifier des réservations**, mais ne peut pas supprimer ou gérer les vols. Il a accès aux tables des passagers et des réservations.

CREATE ROLE 'agent\_reservation';

GRANT INSERT, UPDATE ON `t\_passager`, `t\_reservation` TO 'agent\_reservation';

GRANT SELECT ON `t\_reservation` TO 'agent\_reservation';

### Agent de Comptoir

Peut uniquement **confirmer les enregistrements** et **modifier les sièges**. Il n'a pas de droits pour modifier ou supprimer des réservations ou vols. Il a uniquement accès aux informations liées à l'enregistrement des passagers.

CREATE ROLE 'agent\_comptoir';

GRANT SELECT, UPDATE ON `t\_reservation`, `t\_passager` TO 'agent\_comptoir';

### Analyste

N'a **aucun droit d'écriture**, mais peut consulter toutes les données, en particulier pour générer des rapports de performance. Il a uniquement des privilèges SELECT sur toutes les tables nécessaires.

CREATE ROLE 'analyste';

GRANT SELECT ON `t\_vol`, `t\_reservation`, `t\_passager`, `t\_aeroport` TO 'analyste';

## Utilisateur et association de rôle

### Administrateur

CREATE USER 'admin\_user'@'localhost' IDENTIFIED BY 'password\_admin';

GRANT 'admin' TO 'admin\_user'@'localhost';

### Gestionnaire des Vols

CREATE USER 'gestionnaire\_user'@'localhost' IDENTIFIED BY 'password\_gestionnaire';

GRANT 'gestionnaire\_vols' TO 'gestionnaire\_user'@'localhost';

### Agent de Réservation

CREATE USER 'reservation\_user'@'localhost' IDENTIFIED BY 'password\_reservation';

GRANT 'agent\_reservation' TO 'reservation\_user'@'localhost';

### Agent de Comptoir

CREATE USER 'comptoir\_user'@'localhost' IDENTIFIED BY 'password\_comptoir';

GRANT 'agent\_comptoir' TO 'comptoir\_user'@'localhost';

### Analyste

CREATE USER 'analyste\_user'@'localhost' IDENTIFIED BY 'password\_analyste';

GRANT 'analyste' TO 'analyste\_user'@'localhost';

# Transaction

## Cas n°1

START TRANSACTION;

Point de retour initial

SAVEPOINT avant\_suppression\_reservations;

Suppression des réservations liées au vol

DELETE FROM t\_reservation

WHERE vol\_fk = 10;

Point de retour après suppression des réservations

SAVEPOINT apres\_suppression\_reservations;

Suppression du vol lui-même

DELETE FROM t\_vol

WHERE vol\_id = 10;

COMMIT;

## Cas n°2

La transaction commence à partir de START TRANSACTION, et se termine lors d’un COMMIT ou ROLLBACK. Dans la transaction, s’il n’y a aucun siège qui est pris, alors on fait un INSERT et un COMMIT. Sinon on fait un ROLLBACK et la transaction est annulée.

# Conclusion

Ce projet a permis de concevoir et de mettre en œuvre une base de données complète pour la gestion d’un aéroport, en appliquant des méthodes de modélisation, de manipulation de données. Il illustre l’importance d’une structure bien pensée pour assurer la cohérence, l’efficacité et la fiabilité du système.

## Conclusion personnelle

Ce projet a été une expérience enrichissante qui m’a permis de mettre en pratique l’ensemble des compétences acquises dans le module, de la modélisation des données avec MERISE jusqu’à l’exécution de requêtes complexes en SQL. J’ai mieux compris l’importance d’une bonne structuration des données pour garantir un fonctionnement fluide d’un système aussi critique qu’un aéroport. La mise en place des rôles utilisateurs, des sauvegardes, et l’optimisation des requêtes m’ont également sensibilisé aux problématiques de performance en environnement réel.

# Table des illustrations

[Figure 1: Résultat du LOAD DATA pour la table t\_reservation 7](file:///C:\Users\pu36ylg\Documents\Git\Aeroport\Christopher_Ristic_Rapport.docx#_Toc197931106)

[Figure 2: Transaction à expliquer 11](file:///C:\Users\pu36ylg\Documents\Git\Aeroport\Christopher_Ristic_Rapport.docx#_Toc197931107)

[Figure 3: MCD 12](file:///C:\Users\pu36ylg\Documents\Git\Aeroport\Christopher_Ristic_Rapport.docx#_Toc197931108)

[Figure 4: MLD 12](file:///C:\Users\pu36ylg\Documents\Git\Aeroport\Christopher_Ristic_Rapport.docx#_Toc197931109)

[Figure 5: Création des tables 13](file:///C:\Users\pu36ylg\Documents\Git\Aeroport\Christopher_Ristic_Rapport.docx#_Toc197931110)

[Figure 6: Résultat création tables 13](file:///C:\Users\pu36ylg\Documents\Git\Aeroport\Christopher_Ristic_Rapport.docx#_Toc197931111)

[Figure 7: Schéma aeroport 14](file:///C:\Users\pu36ylg\Documents\Git\Aeroport\Christopher_Ristic_Rapport.docx#_Toc197931112)

# Bibliographie

*How to LOAD DATA setting default value NULL if column field is empty*. (s.d.). Récupéré sur https://stackoverflow.com/questions/58412117/how-to-load-data-setting-default-value-null-if-column-field-is-empty.