

**Université de Sousse**  
**École Nationale d'Ingénieurs de**  
**Sousse**



***Rapport du projet module Base de données***

---

**Aéroport**

---

*Elaboré par :*

*Hela BEN HNIA*

*Nouha ESSID*

*Enseignant :*

*Mr Mohamed Nazih Omri*

***2ème année Informatique appliquée 1***  
***2022/2023***

# Sommaire

Introduction Générale.....	3
Chapitre 1.....	4
Modèle entité-association .....	4
1.1 Introduction.....	4
1.2 Modèle .....	4
1.3 Conclusion .....	5
Chapitre2 .....	6
Modèle Relationnel .....	6
2.1 Introduction.....	6
2.2 Modèle relationnel .....	6
2.3 Conclusion .....	7
Chapitre3 .....	8
Normalisation.....	8
3.1 Introduction.....	8
3.2 Les relations.....	8
3.3 Conclusion .....	13
Chapitre4 .....	14
Implémentation .....	14
4.1 Introduction.....	14
4.2 Outils utilisés .....	14
4.3 Implémentation de l'application .....	15
4.4 Conclusion .....	20
Conclusion Générale .....	21

# Introduction Générale

Une base de données est un ensemble organisé d'informations avec un objectif commun.

Peu importe le support utilisé pour rassembler et stocker les données (papier, fichiers, etc.), dès que des données sont rassemblées et stockées d'une manière organisée dans un but spécifique, on parle de base de données qui sera ainsi accessible à la demande pour plusieurs utilisateurs et des besoins divers.

# Chapitre 1

## Modèle entité-association

### 1.1 Introduction

Dans ce chapitre on va présenter le modèle entité-association de notre projet intitulé « Aéroport ».

### 1.2 Modèle

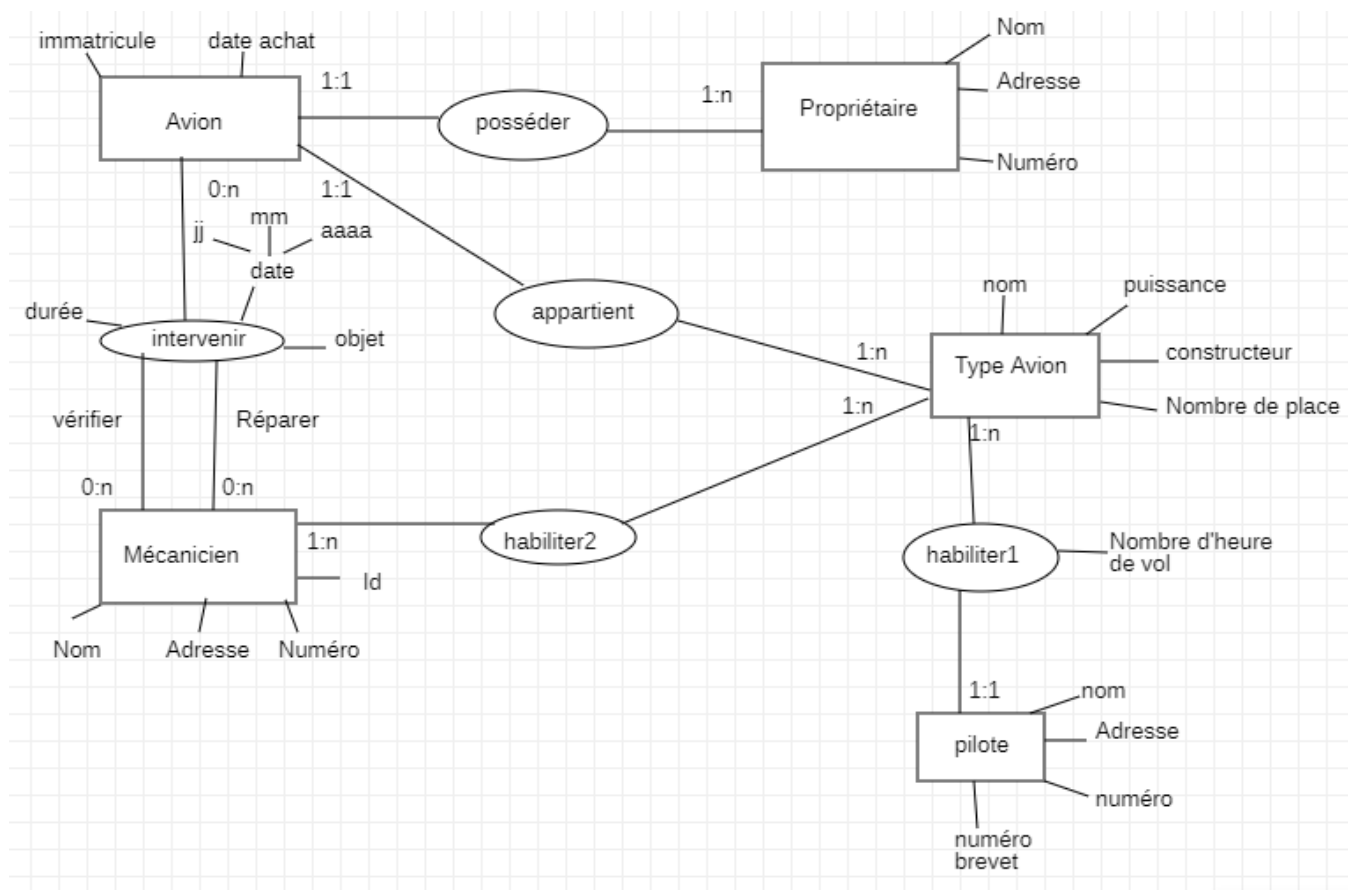


Figure1 : Modèle entité-association

- ✓ Posséder, intervenir, appartient, habiliter<sub>1</sub> et habiliter<sub>2</sub> sont des types d'association.
- ✓ Avion, pilote, mécanicien, type avion et propriétaire **des types d'entité.**

### 1.3 Conclusion

Dans ce chapitre, on a étudié les détails du projet et les présentés sous forme d'un modèle entité-association, pour mieux comprendre les principales tâches.

# Chapitre2

## Modèle Relationnel

### 2.1 Introduction

Le chapitre intitulé Modèle Relationnel a pour but de présenter le modèle entité- association précédent sous forme des relations entre les différentes entités.

### 2.2 Modèle relationnel

Pour avoir une transformation d'un modèle Entité-Association M.E/A en un modèle Relationnel MR il faut respecter 3 règles de transformation :

**Règle 1 :** Tout type d'Entité TE se transforme en une relation.

**Règle 2 :** En présence de cardinalité du type  $m : n$ ,  $m : n$  le type Association TA en question se transforme en une relation R.

**Règle 3 :** En présence de cardinalité du type  $0 : 1$  ou  $1 : 1$ , le type Association TA ne se transforme pas en une relation mais on ajoute tous les identifiants de type Entité coté cardinalité  $m : n$  au type Entité coté cardinalité type  $0 : 1$  ou  $1 : 1$ .

Après l'application de ces 3 règles on trouve les relations suivantes :

Propriétaire(IdProp, nom, adresse, numéro)

Avion (Immatricule, date-achat, #IdProp, #nomT)

#IdProp référence à l'identifiant du propriétaire.

#nomT référence au type.

Mécanicien (idM, nom, num-téléphone, adresse)

idM référence à l'identifiant du mécanicien.

Intervention(#immatricule, #IdR, #IdV, objet, date, durée)

#Immatricule référence Avion

#IdV réfère au mécanicien de vérification.

#IdR réfère au mécanicien de réparation.

Type (nomT, constructeur, puissance, nombre-place)

nomT référence le type d'avion.

Pilote (IdP, nom, adresse, numBrev, numTel)

IdP référence le pilote.

Habilitation1(#nomT, #IdP, nbr-heure-vol)

#IdP référence le pilote.

#nomT référence le type d'avion.

Habilitation2 (#IdM, #nomT)

#IdM référence le mécanicien.

#nomT référence le type d'avion.

## 2.3 Conclusion

Dans ce chapitre on a effectué la présentation du modèle relationnel de notre projet « Aéroport » dans lequel on a parlé des trois règles qui nous ramènent à effectuer une transformation du modèle Entité-Association en un modèle relationnel.

# Chapitre3

## Normalisation

### 3.1 Introduction

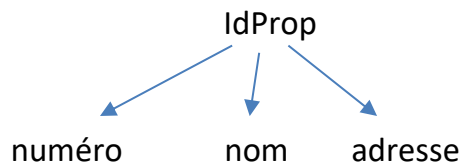
Dans ce chapitre on va normaliser notre base de données de notre projet.

### 3.2 Les relations

#### ➤ Normalisation

Propriétaire(IdProp, adresse, nom, numéro)

Graphe minimum :



Redondances :

- Pas de redondances

Identifiant de la relation :

- Id =( IdProp)

Forme normale :

**1FN :**

- La relation est en 1FN car tous les attributs de cette relation sont simples et monovalués en supposant que l'adresse est une chaîne de caractères et le numéro de téléphone est monovalué.

**2FN :**

- La relation est en 2FN car elle est en 1FN et chaque attribut qui ne fait pas partie de l'identifiant dépend de l'identifiant entier(IdProp).

**3FN :**

- Chaque attribut qui ne fait pas partie d'identifiant dépend directement de l'identifiant entier(IdProp).



**FNBC :**

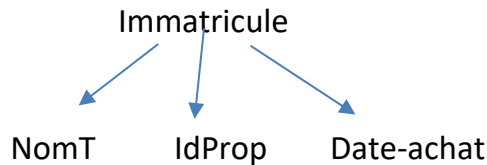
- La source de DF (IdProp) est un identifiant entier.

**4FN :**

- La relation est en 4FN car il n'y a pas de dépendances fonctionnels multivalués.

**Avion** (Immatricule, date-achat, #IdProp, #nomT)

**Graphe minimum :**



**Redondances :**

- Pas de redondances

**Identifiant de la relation :**

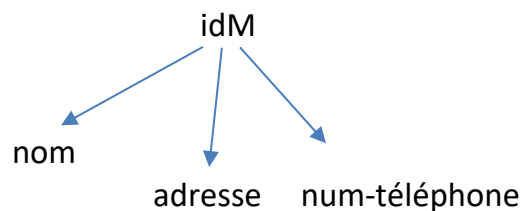
- Id= (Immatricule)

**Forme normale**

- ✚ **1FN** : La relation est en 1FN car tous les attributs de cette relation sont simples et monovalués .
- ✚ **2FN** : La relation est en 2FN car elle est en 1FN et chaque attribut qui ne fait pas partie de l'identifiant dépend de l'identifiant entier(Immatricule).
- ✚ **3FN** : Chaque attribut qui ne fait pas partie d'identifiant dépend directement de l'identifiant entier(Immatricule).
- ✚ **FNBC** : La source de DF (Immatricule) est un identifiant entier.
- ✚ **4FN** : La relation est en 4FN car il n'y a pas de dépendances fonctionnels multivalués.

**Mécanicien** (idM, nom, num-téléphone, adresse)

**Graphe minimum :**



### Redondances :

- Pas de redondances.

### Identifiant de la relation :

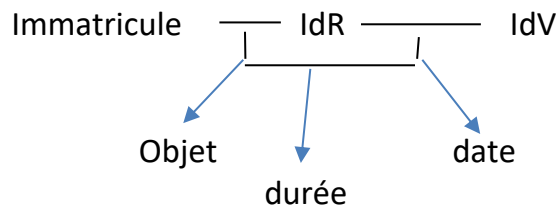
- $id = (idM)$

### Forme normale

- + **1FN** : La relation est en 1FN car tous les attributs de cette relation sont simples et monovalués en supposant que l'adresse est une chaîne de caractères et le numéro de téléphone est monovalué.
- + **2FN** : La relation est en 2FN car elle est en 1FN et chaque attribut qui ne fait pas partie de l'identifiant dépend de l'identifiant entier (IdM).
- + **3FN** : Chaque attribut qui ne fait pas partie d'identifiant dépend directement de l'identifiant entier (IdM).
- + **FNBC** : La source de DF (IdM) est un identifiant entier.
- + **4FN** : La relation est en 4FN car il n'y a pas de dépendances fonctionnelles multivalués.

**Intervention**(#immatricule, #IdR, #IdV, objet, date, durée)

### Graphe minimum :



### Redondances :

- Pas de redondances

### Identifiant de la relation :

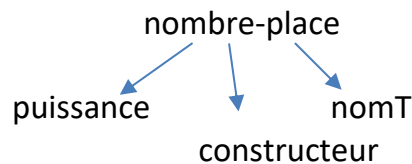
- $Id = (Immatricule, \#IdR, \#IdV)$

### Forme normale :

- + **1FN** : La relation est en 1FN car tous les attributs de cette relation sont simples et monovalués.
- + **2FN** : La relation est en 2FN car elle est en 1FN et chaque attribut qui ne fait pas partie de l'identifiant dépend de l'identifiant entier.
- + **3FN** : Chaque attribut qui ne fait pas partie d'identifiant dépend directement de l'identifiant entier.
- + **FNBC** : La source de DF est un identifiant entier.
- + **4FN** : La relation est en 4FN car il n'y a pas de dépendances fonctionnelles multivalués.

**Type** (nomT, constructeur, puissance, nombre-place)

**Graphe minimum**



**Redondances :**

- Pas de redondances

**Identifiant de la relation :**

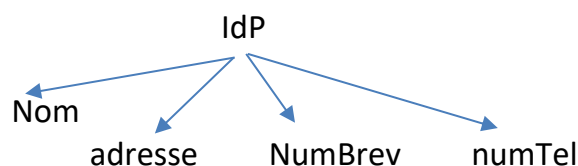
- Id=(nomT)

**Forme normale :**

- ✚ **1FN** : La relation est en 1FN car tous les attributs de cette relation sont simples et monovalués.
- ✚ **2FN** : La relation est en 2FN car elle est en 1FN et chaque attribut qui ne fait pas partie de l'identifiant dépend de l'identifiant entier.
- ✚ **3FN** : Chaque attribut qui ne fait pas partie d'identifiant dépend directement de l'identifiant entier.
- ✚ **FNBC** : La source de DF est un identifiant entier.
- ✚ **4FN** : La relation est en 4FN car il n'y a pas de dépendances fonctionnelles multivalués.

**Pilote** (IdP, nom, adresse, numBrev, numTel)

**Graphe minimum :**



**Redondances :**

- Pas de redondances

**Identifiant de la relation :**

- Id=(IdP)

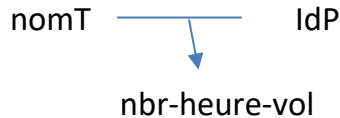
**Forme normale :**

- ✚ **1FN** : La relation est en 1FN car tous les attributs de cette relation sont simples et monovalués.
- ✚ **2FN** : La relation est en 2FN car elle est en 1FN et chaque attribut qui ne fait pas partie de l'identifiant dépend de l'identifiant entier.

- ✚ **3FN** : Chaque attribut qui ne fait pas partie d'identifiant dépend directement de l'identifiant entier.
- ✚ **FNBC** : La source de DF est un identifiant entier.
- ✚ **4FN** : La relation est en 4FN car il n'y a pas de dépendances fonctionnelles multivaluées.

**Habilitation1**(#nomT, #IdP, nbr-heure-vol)

**Graphe minimum :**



**Redondances :**

- Pas de redondances

**Identifiant de la relation :**

- Id=(#nomT, #IdP)

**Forme normale :**

- ✚ **1FN** : La relation est en 1FN car tous les attributs de cette relation sont simples et monovalués.
- ✚ **2FN** : La relation est en 2FN car elle est en 1FN et chaque attribut qui ne fait pas partie de l'identifiant dépend de l'identifiant entier.
- ✚ **3FN** : Chaque attribut qui ne fait pas partie d'identifiant dépend directement de l'identifiant entier.
- ✚ **FNBC** : La source de DF est un identifiant entier.
- ✚ **4FN** : La relation est en 4FN car il n'y a pas de dépendances fonctionnelles multivaluées.

**Habilitation2** (#IdM, #nomT)

**Graphe minimum :**



**Redondances :**

- Pas de redondances.

**Identifiant de la relation :**

- id =(#IdM, #nomT)

**Forme normale**

- ✚ **1FN** : La relation est en 1FN car tous les attributs de cette relation sont simples et monovalués en supposant que l'adresse est une chaîne de caractères et le numéro de téléphone est monovalué.
- ✚ **2FN** : La relation est en 2FN car elle est en 1FN et chaque attribut qui ne fait pas partie de l'identifiant dépend de l'identifiant entier.

- ✚ **3FN** : Chaque attribut qui ne fait pas partie d'identifiant dépend directement de l'identifiant entier.
- ✚ **FNBC** : La source de DF est un identifiant entier.
- ✚ **4FN** : La relation est en 4FN car il n'y a pas de dépendances fonctionnelles multivaluées.

### 3.3 Conclusion :

Dans ce chapitre, on a effectué la normalisation de notre projet « aéroport » afin d'éviter la redondance et d'assurer l'intégrité des données.

Ceci facilite également la maintenance de la base de données.

# Chapitre 4

## Implémentation

### 4.1 Introduction

Dans ce 4ème chapitre on va faire l'implémentation de notre projet en SQL et présenter l'outil qu'on a utilisé pendant cette étape.

### 4.2 Outils utilisés

Au cours de cette implémentation on a fait recours à MySQL.

MySQL Workbench est un logiciel de gestion et d'administration de bases de données MySQL. Via une interface graphique intuitive, il permet, entre autres, de créer, modifier ou supprimer des tables, des comptes utilisateurs, et d'effectuer toutes les opérations inhérentes à la gestion d'une base de données. Pour ce faire, il doit être connecté à un serveur MySQL.



Le logo de MySQL Workbench

## 4.3 Implémentation de l'application

Dans cette figure on présente les tables de notre base de données :

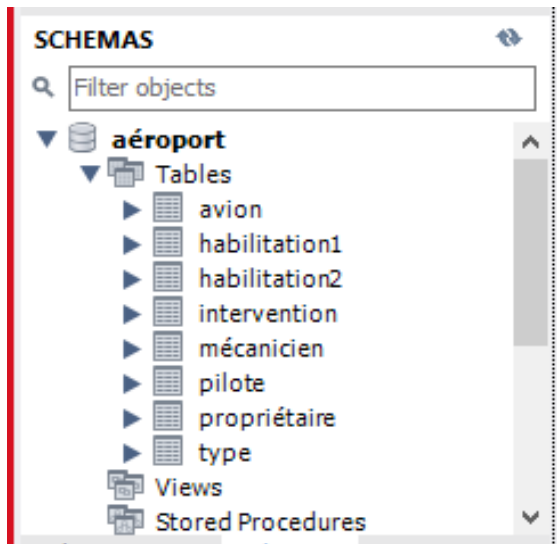


Figure des tables de la base de données

The screenshot shows the 'avion' table view. The left pane shows the 'aéroport' schema with the 'avion' table selected. The right pane shows the SQL query: `SELECT * FROM aéroport.avion;`. Below the query is the 'Result Grid' showing the data rows.

Immatricule	date-achat	idProp	nomT
45612	2017-06-24	2	hiscoo 70
47851	2010-07-07	5	ciso 1000
48797	2017-06-15	1	airBig
85746	2021-05-05	3	hiscoo 70
235648	2018-11-05	1	bumbastic
251463	2019-07-20	4	airBig
526341	2022-03-20	6	ciso 1000
NULL	NULL	NULL	NULL

Table: avion  
Columns: Immatricule (int PK), date-achat (date)

Figure : Le contenu de la table avion

Navigator: avion intervention intervention habilitation1 intervention mécanicien intervention propriétaire x

SCHEMAS

Filter objects

- habilitation1
  - Columns
  - Indexes
  - Foreign Keys
  - Triggers
- habilitation2
- intervention
- mécanicien
- pilote
- propriétaire
  - Columns
  - Indexes

Administration Schemas

Information

Table: **propriétaire**

Columns:

**idProp** int AI PK  
nom varchar(45)

1 • `SELECT * FROM aéroport.propriétaire;`

Limit to 1000 rows

Result Grid

	idProp	nom	adresse	numéro
1	1	Ali	Rue manouba	21345679
2	2	Alia	Rue el mourouj	2035689
3	3	Saleh	rue el nasr	24563156
4	4	Kamel	rue el amour	98564712
5	5	Radhia	rue el jamel	23564187
6	6	Radhwen	rue el amel	21345687
*	NULL	NULL	NULL	NULL

Figure : Le contenu de la table propriétaire

- Requête : Afficher les immatricules des avions dont le nom de propriétaire est « Ali » :

Navigator: avion avion x propriétaire

SCHEMAS

Filter objects

- aéroport
  - Tables
    - avion
    - habilitation1
    - habilitation2
    - intervention
    - mécanicien
    - pilote
    - propriétaire
    - type
  - Views
  - Stored Procedures

Administration Schemas

Information

1 • `SELECT Immatricule FROM aéroport.avion`

2 `join propriétaire On propriétaire.idProp= avion.idProp`

3 `WHERE propriétaire.nom = "Ali"`

Limit to 1000 rows

Result Grid

	Immatricule
1	48797
2	235648



**SCHEMAS**

Filter objects

- habilitation1
- habilitation2
- intervention
- mécanicien
- pilote
- propriétaire
- type
- Views
- Stored Procedures
- Functions
- sakila
- sql\_hr

Administration Schemas

Information

**Table: intervention**

**Columns:**

- immatricule int PK
- IdR int PK
- IdV int PK

1 • `SELECT * FROM aéroport.mécanicien;`

Result Grid

	idM	nom	num-téléphone	adresse
1	1	Mohamed	24567895	avenue 1
2	2	Ahmed	52416387	avenue sahloul
3	3	Hela	58274856	avenue zawya
4	4	Nouha	97852631	avenue falaise
5	5	Yosr	5641238	Rue Skanes
6	6	Nihel	96385245	Rue Soussa
*	NULL	NULL	NULL	NULL

mécanicien 1 x

Figure : Le contenu de la table mécanicien

**SCHEMAS**

Filter objects

- habilitation1
- habilitation2
- intervention
- mécanicien
- pilote
- propriétaire
- type
- Views
- Stored Procedures
- Functions
- sakila
- sql\_hr

Administration Schemas

Information

**Table: intervention**

**Columns:**

- immatricule int PK
- IdR int PK
- IdV int PK

1 • `SELECT * FROM aéroport.intervention;`

Result Grid

	immatricule	IdR	IdV	objet	datee	durée
1	48797	2	3	objet 4	2021-09-03	51 min
2	85746	5	6	objet 5	2022-12-09	47 min
3	251463	1	2	objet1	2022-06-16	20min
4	251463	2	5	objet 1	2019-09-07	30 min
5	526341	5	4	objet 2	2020-12-07	49 min
*	NULL	NULL	NULL	NULL	NULL	NULL

Figure : Le contenu de la table intervention

- Requête : Afficher l'immatricule et le nom du mécanicien dont le numéro de téléphone = « 52416387 »

The screenshot shows a database management interface. On the left, the 'SCHEMAS' pane lists various tables including 'habilitation1', 'intervention', 'mécanicien', and 'pilote'. The 'intervention' table is selected. The main pane displays a SQL query:

```
SELECT immatricule, nom FROM aeroport.intervention
JOIN mécanicien ON mécanicien.idM = intervention.IdR
WHERE aeroport.mécanicien.num_téléphone = "52416387"
```

Below the query, the 'Result Grid' shows the following data:

immatricule	nom
48797	Ahmed
251463	Ahmed

The screenshot shows the same database management interface, but now the 'pilote' table is selected. The main pane displays a SQL query:

```
SELECT * FROM aeroport.pilote;
```

Below the query, the 'Result Grid' shows the following data:

idP	nom	adresse	numBrev	numTel
1	Ragheb	tunis beb bhar	1200	25836974
2	Chiraz	Jendouba beb bhar	1201	96812974
3	Hayfa	Jerba beb bhar	1202	55552974
4	Mourad	kairouan beb jdid	1203	98765463
*	NULL	NULL	NULL	NULL

Figure : Le contenu de la table pilote

Navigator: SCHEMAS

Filter objects

- habilitation1
  - Columns
  - Indexes
  - Foreign Keys
  - Triggers
- habilitation2
- intervention
- mécanicien
- pilote
- propriétaire
- type

Views Administration Schemas

Information:

**Table: habilitation1**

**Columns:**

avion intervention intervention habilitation1 habilitation1 x

Limit to 1000 rows

1 • `SELECT * FROM aéroport.habilitation1;`

Result Grid

	nomT	idP	nbr-heure-vol
▶	airBig	1	10
	bibhos	1	20
	bibhos	3	15
	ciso 1000	2	30
	hiscoo 70	4	12
*	NULL	NULL	NULL

Edit: Export/Import

Figure : Le contenu de la table habilitation

- Requête : Afficher le nom et l'adresse des pilotes qui sont habilités sur l'avion de type « bibhos »

Navigator: SCHEMAS

Filter objects

- habilitation1
  - Columns
  - Indexes
  - Foreign Keys
  - Triggers
- habilitation2
- intervention
- mécanicien
- pilote
- propriétaire
- type

Views Administration Schemas

avion propriétaire intervention intervention habilitation1 x

Limit to 1000 rows

1 • `SELECT nom , adresse FROM aéroport.habilitation1`  
 2 `JOIN pilote ON pilote.idP = habilitation1.idP`  
 3 `WHERE aéroport.habilitation1.nomT = "bibhos"`

Result Grid

	nom	adresse
▶	Ragheb	tunis beb bhar
	Hayfa	Jerba beb bhar

Export: Wrap Cell Content: I A

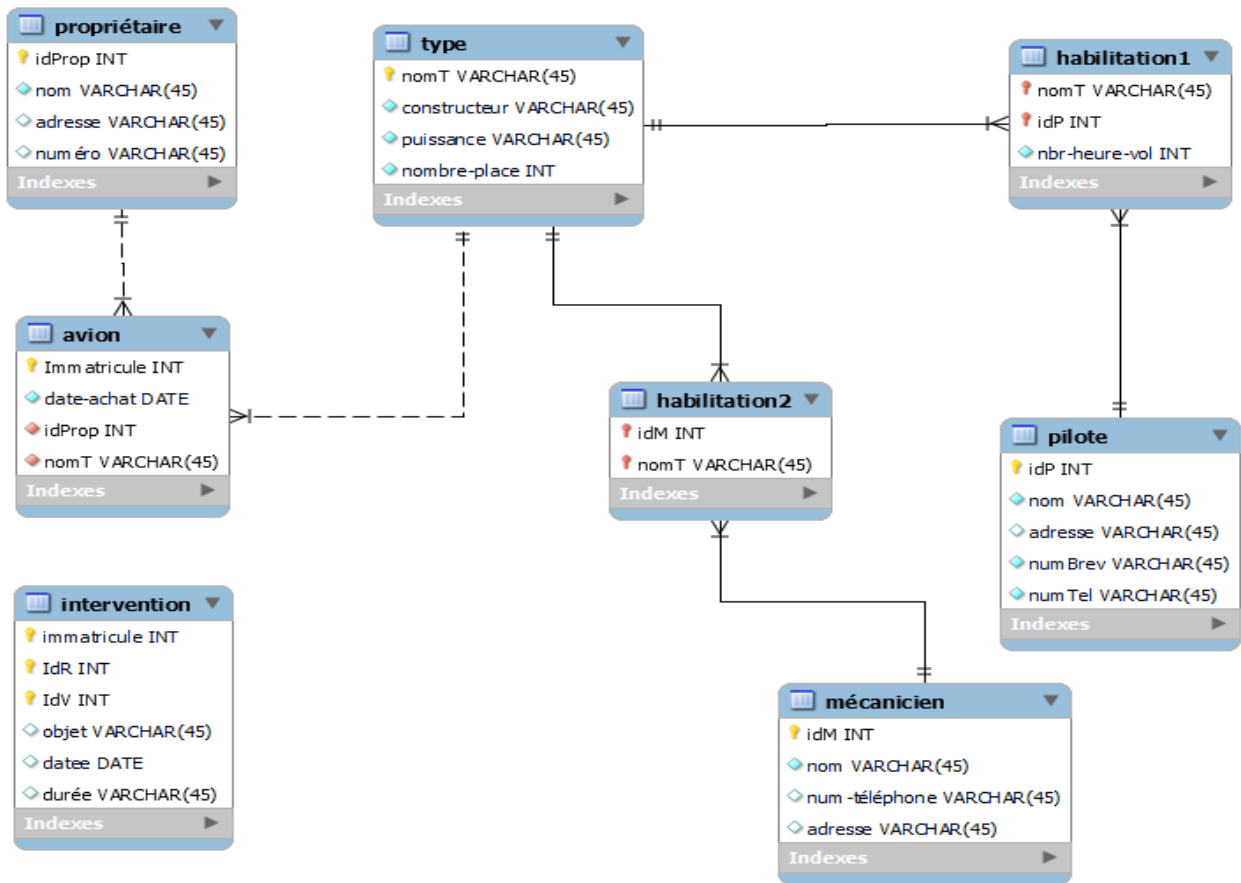


Figure : Vue Relationnelle de la base

## 4.4 Conclusion

Dans ce chapitre on a réalisé l'implémentation de la base de données de problème étudié dans les chapitres précédents en utilisant MySQL Workbench comme système de gestion de bases de données relationnelles.

## **Conclusion Générale**

Grâce à ce projet on a acquis de nombreuses compétences précieuses tel que l'implémentation de la base de données sur le serveur local et le SQL qui nous seront très utiles dans nos futurs projets.

On souhaite ainsi dans une seconde étape développer un site web ou application mobile et la connectée avec notre base de données.