

2025-02-23

Professeur : BELHOMME DIDIER

STAGE & ÉPREUVE INTÉGRÉE: ANALYSE DE LA BASE DE DONNEES

NOM DU Projet : EYEWEB

VANHEES Vincent

Table des matières

[1. Introduction 4](#_Toc192791082)

[2. Objectifs de la Base de Données 4](#_Toc192791083)

[3. Modélisation Relationnelle et Normalisation 4](#_Toc192791084)

[4. Performances et Optimisation 4](#_Toc192791085)

[4.1 Indexation 4](#_Toc192791086)

[4.2 Partitionnement des données 5](#_Toc192791087)

[4.2.1 Stockage et Optimisation 5](#_Toc192791088)

[5. Sécurité et Conformité 5](#_Toc192791089)

[5.1 Sécurité des Accès 5](#_Toc192791090)

[6. Thésaurus 5](#_Toc192791091)

[6.1 Entité : Token 5](#_Toc192791092)

[6.1.1 Description 5](#_Toc192791093)

[6.1.2 Relations et Contraintes 5](#_Toc192791094)

[6.1.3 Contraintes d’Intégrité 5](#_Toc192791095)

[6.1.4 Détails 6](#_Toc192791096)

[6.2 Entité : Address 6](#_Toc192791097)

[6.2.1 Description 6](#_Toc192791098)

[6.2.2 Relations et Contraintes 6](#_Toc192791099)

[6.2.3 Contraintes d’Intégrité 6](#_Toc192791100)

[6.2.4 Détails des Attributs 6](#_Toc192791101)

[6.3 Entité : Organizations 7](#_Toc192791102)

[6.3.1 Description 7](#_Toc192791103)

[6.3.2 Relations 7](#_Toc192791104)

[6.3.3 Contraintes d’Intégrité 7](#_Toc192791105)

[6.3.4 Détails des Attributs 7](#_Toc192791106)

[6.4 Entité : Patient 7](#_Toc192791107)

[6.4.1 Description 7](#_Toc192791108)

[6.4.2 Relations et Contraintes 7](#_Toc192791109)

[6.4.3 Détails des Attributs 8](#_Toc192791110)

[6.5 Entité : License 9](#_Toc192791111)

[6.5.1 Description 9](#_Toc192791112)

[6.5.2 Relations et contraintes 9](#_Toc192791113)

[6.5.3 Contraintes d’Intégrité 9](#_Toc192791114)

[6.5.4 Détails 9](#_Toc192791115)

[6.6 Entité : VerificationCode 9](#_Toc192791116)

[6.6.1 Description 9](#_Toc192791117)

[6.6.2 Relations et Contraintes 9](#_Toc192791118)

[6.6.3 Contraintes d’Intégrité 9](#_Toc192791119)

[6.6.4 Détails des Attributs 10](#_Toc192791120)

[6.7 Entité : Examen 10](#_Toc192791121)

[6.7.1 Description 10](#_Toc192791122)

[6.7.2 Relations et Contraintes 10](#_Toc192791123)

[6.7.3 Contraintes d’Intégrité 10](#_Toc192791124)

[6.7.4 Détails 11](#_Toc192791125)

[6.8 Entité : Calcul 12](#_Toc192791126)

[6.8.1 Description 12](#_Toc192791127)

[6.8.1 Relations et contraintes 12](#_Toc192791128)

[6.8.2 Contraintes d’Intégrité 12](#_Toc192791129)

[6.8.3 Détails des Attributs 12](#_Toc192791130)

[6.9 Entité : UserOrganization 13](#_Toc192791131)

[6.9.1 Description 13](#_Toc192791132)

[6.9.2 Relations et contraintes 13](#_Toc192791133)

[6.9.3 Contraintes d’Intégrité 13](#_Toc192791134)

[6.9.3.3 Détails des Attributs 13](#_Toc192791135)

[6.10 Entité : Link 13](#_Toc192791136)

[6.10.1 Description 13](#_Toc192791137)

[6.10.2 Relations et contraintes 14](#_Toc192791138)

[6.10.3 Contraintes d’Intégrité 14](#_Toc192791139)

[6.10.3.2 Détails des Attributs 14](#_Toc192791140)

[6.11 Entité : r\_user\_token 14](#_Toc192791141)

[6.11.1 Description 14](#_Toc192791142)

[6.11.2 Relations et contraintes 14](#_Toc192791143)

[6.11.3 Détails 14](#_Toc192791144)

[6.12 Entité : Identified 14](#_Toc192791145)

[6.12.1 Description 14](#_Toc192791146)

[6.12.2 Relations et contraintes 15](#_Toc192791147)

[6.12.3 Contraintes d’Intégrité 15](#_Toc192791148)

[6.12.4 Détails des Attributs 15](#_Toc192791149)

[6.13 Entité : User 15](#_Toc192791150)

[6.13.1 Description 15](#_Toc192791151)

[6.13.2 Relations 15](#_Toc192791152)

[6.13.3 Héritage et Contraintes d’Inclusion 16](#_Toc192791153)

[6.13.4 Contraintes d’Inclusion 16](#_Toc192791154)

[6.13.5 Trigger SQL : 16](#_Toc192791155)

[6.13.6 Détails des Attributs 16](#_Toc192791156)

[6.14 Entité : Workflow 17](#_Toc192791157)

[6.14.1 Description 17](#_Toc192791158)

[6.14.2 Relations et Contraintes 17](#_Toc192791159)

[6.14.3 Contraintes d’Intégrité 17](#_Toc192791160)

[6.14.4 Détails des Attributs 17](#_Toc192791161)

[6.15 Entité : Consultation 17](#_Toc192791162)

[6.15.1 Description 17](#_Toc192791163)

[6.15.2 Relations et contraintes 17](#_Toc192791164)

[6.15.3 Détails des Attributs 17](#_Toc192791165)

[6.16 Entité : Protocol 17](#_Toc192791166)

[6.16.1 Description 17](#_Toc192791167)

[6.16.2 Relations et contraintes 17](#_Toc192791168)

[6.16.3 Détails des Attributs 18](#_Toc192791169)

[6.17 Entité : Planning 18](#_Toc192791170)

[6.17.1 Description 18](#_Toc192791171)

[6.17.2 Relations et contraintes 18](#_Toc192791172)

[6.17.3 Détails des Attributs 18](#_Toc192791173)

[6.18 Entité : Role 18](#_Toc192791174)

[6.18.1 Description 18](#_Toc192791175)

[6.18.2 Relations et contraintes 18](#_Toc192791176)

[6.18.3 Détails des attributs 18](#_Toc192791177)

[6.19 Entité : Lens 19](#_Toc192791178)

[6.19.1 Description 19](#_Toc192791179)

[6.19.2 Relations et contraintes 19](#_Toc192791180)

[6.19.3 Détails 19](#_Toc192791181)

[6.20 Entité : Constant 19](#_Toc192791182)

[6.20.1 Description 19](#_Toc192791183)

[6.20.2 Relations et contraintes 19](#_Toc192791184)

[6.20.3 Détails des Attributs 20](#_Toc192791185)

[6.21 Entité : Diopter 20](#_Toc192791186)

[6.21.1 Description 20](#_Toc192791187)

[6.21.2 Relations et contraintes 20](#_Toc192791188)

[6.21.3 Détails des Attributs 20](#_Toc192791189)

[6.22 Entité : LensManufacturer 20](#_Toc192791190)

[6.22.1 Description 20](#_Toc192791191)

[6.22.2 Relations et contraintes 20](#_Toc192791192)

[6.22.3 Détails des Attributs 20](#_Toc192791193)

[7. Flux des données 21](#_Toc192791194)

[7.1 Inscription et gestion des utilisateurs 21](#_Toc192791195)

[7.1.1 Création d’un compte utilisateur 21](#_Toc192791196)

[7.2 Ajout d’un Patient et gestion des organisations 21](#_Toc192791197)

[7.2.1 Enregistrement d’un nouveau patient 21](#_Toc192791198)

[7.3 Enregistrement des Examens Biométriques 22](#_Toc192791199)

[7.3.1 Un patient réalise un examen ophtalmologique 22](#_Toc192791200)

[7.3.2 Flux des données et relations utilisées 22](#_Toc192791201)

[7.4 Calcul et Simulation des Implants Intraoculaires 22](#_Toc192791202)

[7.4.1 Génération d’un Calcul Intraoculaire 22](#_Toc192791203)

[7.5 Gestion des Lentilles Intraoculaires 23](#_Toc192791204)

[7.5.1 Sélection et commande d’une lentille 23](#_Toc192791205)

[7.6 Gestion des Accès et Sécurité 23](#_Toc192791206)

[7.6.1 Authentification et Permissions 23](#_Toc192791207)

[8. Évolutivité et Scalabilité 24](#_Toc192791208)

[8.1 Modularité et Extensibilité 24](#_Toc192791209)

[8.1.1 Ajout de nouvelles fonctionnalités métier : 24](#_Toc192791210)

[8.1.2 Évolutivité de l’examen biométrique : 24](#_Toc192791211)

[8.1.3 Gestion avancée des organisations : 24](#_Toc192791212)

[8.2 Intégration avec des outils externes 24](#_Toc192791213)

[8.2.1 API REST sécurisée : 24](#_Toc192791214)

[8.2.2 Exportation des données médicales : 24](#_Toc192791215)

[8.2.3 Connexion à des services de calcul avancés : 24](#_Toc192791216)

[9. Conclusion 25](#_Toc192791217)

[10. Annexes 26](#_Toc192791218)

[10.1 Modèle Conceptuel de Données (MCD) 26](#_Toc192791219)

[10.2 Modèle Logique de Données (MLD) 27](#_Toc192791220)

[10.3 Unified Modeling Language (UML) 28](#_Toc192791221)

# Introduction

Cette documentation fournit une analyse complète de la base de données en incluant :

* La structure relationnelle et les contraintes
* Les entités et leurs descriptions
* Les relations et contraintes d’intégrité
* Les flux de données et leur gestion
* Les bonnes pratiques et optimisations

Cette base de données a été conçue pour une application médicale permettant la gestion des patients, des examens biométriques, des calculs intraoculaires et des lentilles intraoculaires.

# Objectifs de la Base de Données

* Stocker et gérer les patients ainsi que leurs données biométriques et calculs intraoculaires.
* Assurer une traçabilité complète des examens médicaux et des décisions cliniques.
* Fournir un accès sécurisé et restreint aux différents utilisateurs du système (médecins, secrétaires, administrateurs).
* Garantir une bonne performance pour des requêtes complexes et volumineuses sur les données biométriques.

# Modélisation Relationnelle et Normalisation

Elle suit une modélisation relationnelle optimisée et normalisée (3NF) afin de garantir :

* Intégrité des données via l’utilisation de clés primaires et étrangères.
* Réduction de la redondance avec une organisation normalisée des relations.
* Optimisation des requêtes en minimisant les jointures inutiles et en utilisant des index. Catégorisation des tables

|  |  |
| --- | --- |
| Catégorie | Tables concernées |
| Gestion des utilisateurs | User, VerificationCode, Token, UserOrganization, Role |
| Gestion des patients | Patient, Address, Organizations |
| Gestion des examens | Examen, Consultation, Workflow, Protocol, Planning |
| Données biométriques | Calcul, Diopter, Constant, Lens, LensManufacturer |
| Gestion des relations | Link, r\_user\_token |

# Performances et Optimisation

## Indexation

* Index sur les clés primaires (id\_user, id\_patient, id\_examen, id\_calcul) pour accélérer la recherche.
* Index composite sur les jointures fréquentes (id\_patient, id\_examen).

## Partitionnement des données

* Partitionnement temporel sur Examen et Calcul basé sur les dates d’examen et de calcul.
* Partitionnement logique des patients par organisations.

### Stockage et Optimisation

* Utilisation du type JSONB pour certaines colonnes contenant des données flexibles.
* Stockage des valeurs numériques en DECIMAL(5,2) pour optimiser l’espace et la précision.

# Sécurité et Conformité

## Sécurité des Accès

* Authentification JWT : Utilisation de jetons pour l’authentification.
* Hachage des mots de passe avec BCrypt.
* Gestion des rôles et permissions via UserOrganization.
* Restrictions d’accès à certaines données sensibles (éviter l’accès direct aux NISS et aux données médicales).

# Thésaurus

## Entité : Token

### Description

La table Token gère les jetons d'authentification des utilisateurs via JWT (JSON Web Token).

Elle permet de gérer la session d'un utilisateur, la validation des accès et la révocation des tokens.

### Relations et Contraintes

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| 1-N avec User | Relation | Un utilisateur peut avoir plusieurs tokens actifs en même temps. |
| id\_user (clé étrangère) | Relation | Référence vers l’utilisateur détenteur du token. |

### Contraintes d’Intégrité

#### Un token doit être lié à un utilisateur (id\_user non nul)

* Contrainte SQL : ALTER TABLE Token ADD CONSTRAINT fk\_token\_user

FOREIGN KEY (id\_user) REFERENCES User(id\_user) ON DELETE CASCADE;

* Explication : Lorsqu'un utilisateur est supprimé, tous ses tokens sont supprimés automatiquement.

#### Un token ne peut pas être révoqué et valide en même temps

* Contrainte SQL : ALTER TABLE Token ADD CONSTRAINT chk\_token\_validity

CHECK (revoked = FALSE OR expired = FALSE);

* Explication : Un token est soit actif, soit expiré/révoqué, mais jamais les deux en même temps.

### Détails

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| id\_token | INT AUTO\_INCREMENT | Identifiant unique du token | 1 |
| token | VARCHAR(500) | Valeur du token | eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9 |
| tokenType | VARCHAR(50) | Type du token | BEARER |
| revoked | BOOLEAN | Indique si le token est révoqué | FALSE |
| expired | BOOLEAN | Indique si le token est expiré | FALSE |
| id\_user | INT | Référence vers un utilisateur | 2 |

## Entité : Address

### Description

La table Address stocke les informations d'adresse des utilisateurs et des patients.

Elle est référencée par Identified, permettant ainsi à chaque utilisateur et patient d’avoir une adresse unique.

### Relations et Contraintes

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| 1-N avec Identified | Relation | Une adresse peut être associée à plusieurs identités (User, Patient). |
| id\_address (clé étrangère) | Relation | Référencée par la table Identified. |

### Contraintes d’Intégrité

#### Une adresse doit être unique pour chaque entité Identified

* Contrainte SQL : ALTER TABLE Identified ADD CONSTRAINT fk\_identified\_address

FOREIGN KEY (id\_address) REFERENCES Address(id\_address) ON DELETE SET NULL;

* Explication : Si une adresse est supprimée, la référence dans Identified est mise à NULL.

#### Un code postal doit être numérique

* Contrainte SQL : ALTER TABLE Address ADD CONSTRAINT chk\_address\_zipcode

CHECK (zip\_code ~ '^[0-9]+$');

* Explication : Assure que le code postal ne contient que des chiffres.

### Détails des Attributs

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| id\_address | INT AUTO\_INCREMENT | Identifiant unique de l'adresse | 1 |
| street | VARCHAR(255) | Rue | rue Saint-Laurent |
| street\_number | VARCHAR(10) | Numéro | 42 |
| box\_number | VARCHAR(10) | Numéro de boîte | 2 |
| zip\_code | VARCHAR(10) | Code postal | 4000 |
| city | VARCHAR(100) | Ville | Liège |
| country | VARCHAR(100) | Pays | Belgique |

## Entité : Organizations

### Description

La table Organizations stocke les organisations médicales auxquelles peuvent appartenir les utilisateurs et les patients.

### Relations

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| 1-N avec Patient | Relation | Une organisation peut regrouper plusieurs patients. |
| 1-N avec UserOrganization | Relation | Une organisation peut avoir plusieurs utilisateurs affiliés. |
| id\_organizations (clé étrangère) | Relation | Référencé par Patient, License, UserOrganization. |

### Contraintes d’Intégrité

#### Un patient doit toujours être rattaché à une organisation (id\_organizations non nul)

* Contrainte SQL :ALTER TABLE Patient ADD CONSTRAINT fk\_patient\_organization

FOREIGN KEY (id\_organizations) REFERENCES Organizations(id\_organizations) ON DELETE CASCADE;

* Explication : Lorsqu’une organisation est supprimée, tous ses patients sont supprimés.

#### Une organisation ne peut pas avoir un nombre négatif de licences

* Contrainte SQL :ALTER TABLE Organizations ADD CONSTRAINT chk\_organization\_licenses

CHECK (nb\_licenses >= 0);

* Explication : Une organisation ne peut pas avoir un nombre négatif de licences actives.

### Détails des Attributs

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| id\_organizations | INT AUTO\_INCREMENT | Identifiant unique de l'organisation | 1 |
| name | VARCHAR(255) | Nom de l'organisation | C.H.R |
| is\_global | BOOLEAN | Indique si l'organisation est globale | TRUE |
| nb\_licenses | nb\_licenses | nb\_licenses | 10 |

## Entité : Patient

### Description

La table Patient stocke les informations médicales et administratives des patients, ainsi que leur affiliation à une organisation médicale.

Chaque patient est associé à une identité unique dans Identified, permettant une gestion cohérente des données personnelles et de contact.

Un patient peut être lié à plusieurs examens et calculs intraoculaires, ainsi qu'à une organisation médicale spécifique où il est suivi.

### Relations et Contraintes

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| 1-1 avec Identified | Héritage | Un patient hérite de Identified pour stocker ses informations personnelles et de contact. |
| 1-N avec Organizations | Relation | Un patient est affilié à une seule organisation médicale. |
| 1-N avec Examen | Relation | Un patient peut avoir plusieurs examens médicaux. |
| 1-N avec Calcul | Relation | Un patient peut avoir plusieurs calculs biométriques réalisés. |
| 1-N avec Consultation | Relation | Un patient peut être suivi dans plusieurs consultations médicales. |

#### Héritage et Contraintes d’Inclusion

* *Identified* est une entité parent utilisée pour stocker des informations communes aux entités *User* et *Patient*.
* *User* et *Patient* héritent de *Identified*, garantissant que chaque enregistrement est associé à une identité unique.

#### Contrainte d'inclusion

* id\_organizations IS NOT NULL pour assurer qu'un patient appartient toujours à une organisation.
* Patient doit appartenir à une organisation
* ALTER TABLE Patient ADD CONSTRAINT chk\_patient\_organization

CHECK (id\_organizations IS NOT NULL);

* User doit avoir une identité référencée
* ALTER TABLE User ADD CONSTRAINT chk\_user\_identified

CHECK (id\_identified IS NOT NULL);

#### Trigger SQL

CREATE TRIGGER prevent\_patient\_deletion

BEFORE DELETE ON Patient

FOR EACH ROW

WHEN EXISTS (SELECT 1 FROM Examen WHERE Examen.id\_patient = OLD.id\_patient)

OR EXISTS (SELECT 1 FROM Calcul WHERE Calcul.id\_patient = OLD.id\_patient)

BEGIN

RAISE EXCEPTION 'Impossible de supprimer un patient avec des examens ou calculs actifs';

END;

* Explication : Cette contrainte empêche la suppression accidentelle d'un patient qui possède encore des données médicales liées.

### Détails des Attributs

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| id\_patient | INT AUTO\_INCREMENT | Identifiant unique du patient | 1 |
| birthdate | DATE | Date de naissance du patient | 03-12-85 |
| niss | VARCHAR(100) | Numéro national du patient | 85-12-03.135-58 |
| gender | VARCHAR(50) | Genre du patient | M |
| job | VARCHAR(50) | Profession du patient | Ingénieur |
| hobbies | VARCHAR(50) | Loisirs du patient | Randonnée |
| Id\_identified | INT | Référence vers l’entité Identified | 2 |
| id\_organizations | INT | Référence vers l'organisation | 1 |

## Entité : License

### Description

La table License stocke les licences attribuées aux organisations.

Elle permet de suivre l’activation des licences, leur expiration et leur utilisateur responsable.

### Relations et contraintes

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| 1-N avec Organizations | Relation | Une organisation peut posséder plusieurs licences actives. |
| id\_organizations (clé étrangère) | Relation | Référence vers Organizations pour associer la licence à une organisation. |

### Contraintes d’Intégrité

#### Une licence doit être toujours associée à une organisation (id\_organizations non nul)

* Contrainte SQL : ALTER TABLE License ADD CONSTRAINT fk\_license\_organization

FOREIGN KEY (id\_organizations) REFERENCES Organizations(id\_organizations) ON DELETE CASCADE;

* Explication : Lorsqu’une organisation est supprimée, toutes ses licences sont supprimées.

#### La clé de licence doit être unique

* Contrainte SQL :ALTER TABLE License ADD CONSTRAINT unique\_license\_key UNIQUE (key);
* Explication : Une même clé de licence ne peut pas être attribuée à plusieurs organisations.

### Détails

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| id\_license | INT AUTO\_INCREMENT | Identifiant unique pour license | 1 |
| name | VARCHAR(200) | Nom de la licence | PREMIUM |
| key | VARCHAR(100) | Clé de la license. | L\_0001 |
| user\_by | VARCHAR(255) | Utilisateur ayant activé la license. | ADMIN |
| expiration\_date | VARCHAR(20) | Date d’expiration. | 2025-12-31 |
| id\_organizations | INT | Organisation associée. | 2 |

## Entité : VerificationCode

### Description

La table VerificationCode stocke les codes de validation utilisés pour l’authentification et la vérification des utilisateurs lors de leur inscription ou de la récupération de mot de passe.

### Relations et Contraintes

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| 1-1 avec User | Relation | Chaque utilisateur peut avoir un seul code de vérification actif. |
| id\_user (clé étrangère) | Relation | Référence vers User. |

### Contraintes d’Intégrité

#### Un utilisateur ne peut pas avoir plusieurs codes de vérification actifs

* Contrainte SQL : ALTER TABLE VerificationCode ADD CONSTRAINT unique\_verification\_code

UNIQUE (id\_user);

* Explication : Un utilisateur ne peut avoir qu’un seul code actif à la fois.

#### Un code ne peut pas être révoqué et valide en même temps

* Contrainte SQL : ALTER TABLE VerificationCode ADD CONSTRAINT chk\_verification\_validity

CHECK (revoked = FALSE OR expired = FALSE);

* Explication : Un code est soit valide, soit révoqué ou expiré, mais jamais les deux en même temps.

### Détails des Attributs

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| id\_verification\_code | INT AUTO\_INCREMENT | Identifiant unique du code | 1 |
| token | VARCHAR(100) | Code de validation | A1B2C3 |
| token\_type | VARCHAR(50) | Type du token | EMAIL\_CONFIRMATION |
| revoked | BOOLEAN | Indique si le code est révoqué | FALSE |
| expired | BOOLEAN | Indique si le code est expiré | FALSE |
| id\_user | INT | Référence vers un utilisateur | 3 |

## Entité : Examen

### Description

La table Examen stocke les examens biométriques réalisés sur les patients, comprenant des mesures ophtalmologiques essentielles.

### Relations et Contraintes

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| 1-N avec Patient | Relation | Un patient peut avoir plusieurs examens enregistrés. |
| id\_patient (clé étrangère) | Relation | Référence vers Patient pour identifier le patient examiné. |
| 1-N avec Calcul | Relation | Un examen peut être utilisé pour plusieurs calculs intraoculaires (id\_calcul). |

### Contraintes d’Intégrité

#### Chaque examen doit être associé à un patient (id\_patient non nul)

* Contrainte SQL : ALTER TABLE Examen ADD CONSTRAINT fk\_examen\_patient FOREIGN KEY (id\_patient) REFERENCES Patient(id\_patient) ON DELETE CASCADE;
* Explication : Lorsqu’un patient est supprimé, tous ses examens sont également supprimés.

#### Le type d’examen doit être valide (OCT, Topographie, Aberrométrie, etc.)

* Contrainte SQL :ALTER TABLE Examen ADD CONSTRAINT chk\_examen\_type CHECK (exam\_type IN ('OCT', 'Topographie', 'Aberrométrie', 'Pachymétrie', 'Biométrie'));
* Explication : Empêche les valeurs invalides dans exam\_type.

#### La qualité de l’examen doit être bien renseignée (Haute, Moyenne, Basse)

* Contrainte SQL : ALTER TABLE Examen ADD CONSTRAINT chk\_examen\_quality CHECK (exam\_quality IN ('Haute', 'Moyenne', 'Basse'));
* Explication : Garantit la standardisation des données de qualité.

### Détails

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attributs | Types | | Définition | Exemples |
| id\_examen | INT AUTO\_INCREMENT | Identifiant unique de l’examen | | 1 |
| eye\_side | VARCHAR(25) | Côté de l'œil concerné (OD = œil droit, OG = œil gauche) | | OD |
| exam\_type | VARCHAR(25) | Type d'examen effectué | | OCT |
| exam\_date | DATE | Date de l'examen | | 2025-01-05 |
| exam\_comment | VARCHAR(255) | Commentaire sur l'examen | | Bien déroulé |
| exam\_quality | VARCHAR(150) | Qualité de l'examen | | Haute |
| calcul\_date | DATE | Date du calcul effectué | | 2025-06-02 |
| eye\_status | VARCHAR(150) | Statut de l'œil examiné | | Stable |
| selected | VARCHAR(225) | Indique si cet examen est sélectionné | | Oui |
| acd | DECIMAL(5,2) | Profondeur de la chambre antérieure | | 3.45 |
| internal\_acd | DECIMAL(5,2) | Profondeur interne de la chambre antérieure | | 3.20 |
| pupil\_dia | DECIMAL(5,2) | Diamètre pupillaire | | 4.50 |
| pupil\_min | DECIMAL(5,2) | Diamètre pupillaire minimum | | 2.50 |
| pupil\_max | DECIMAL(5,2) | Diamètre pupillaire maximum | | 6.00 |
| wtw | DECIMAL(5,2) | Largeur cornéenne blanche à blanche | | 11.80 |
| cordk | DECIMAL(5,2) | Distance centrale cornéenne | | 0.50 |
| z40 | DECIMAL(5,2) | Aberration de 4e ordre Zernike | | 0.10 |
| hoa | DECIMAL(5,2) | Aberration optique de haut niveau | | 0.12 |
| kappa\_angle | DECIMAL(5,2) | Angle kappa | | 5.6 |
| n | DECIMAL(5,2) | Rayon de courbure K1 | | 42.75 |
| k1 | DECIMAL(5,2) | Rayon de courbure K1 | | 42.75 |
| k1\_mm | DECIMAL(5,2) | Rayon de courbure K1 en mm | | 7.80 |
| k1\_axis | DECIMAL(5,2) | Axe de l'astigmatisme K1 | | 90 |
| k2 | DECIMAL(5,2) | Rayon de courbure K2 | | 44.50 |
| k2\_mm | DECIMAL(5,2) | Rayon de courbure K2 en mm | | 7.60 |
| k2\_axis | DECIMAL(5,2) | Axe de l'astigmatisme K2 | | 180 |
| k\_astig | DECIMAL(5,2) | Astigmatisme cornéen | | 1.75 |
| k\_astig\_axis | DECIMAL(5,2) | Axe de l'astigmatisme | | 170 |
| k\_avg | DECIMAL(5,2) | Moyenne des rayons K | | 43.80 |
| k1\_cornea\_back | DECIMAL(5,2) | Rayon de courbure K1 de la face postérieure de la cornée | | 42.50 |
| k2\_cornea\_back | DECIMAL(5,2) | Rayon de courbure K2 de la face postérieure de la cornée | | 44.00 |
| k1\_axis\_cornea\_back | DECIMAL(5,2) | Axe K1 de la face postérieure de la cornée | | 85 |
| k2\_axis\_cornea\_back | DECIMAL(5,2) | Axe K2 de la face postérieure de la cornée | | 175 |
| cct | DECIMAL(5,2) | Épaisseur centrale de la cornée | | 550 |
| cct\_min | DECIMAL(5,2) | Épaisseur minimale centrale de la cornée | | 530 |
| sia\_cyl | DECIMAL(5,2) | Astigmatisme chirurgical induit | | 0.50 |
| sia\_axis | DECIMAL(5,2) | Axe de l'astigmatisme chirurgical induit | | 95 |
| snr | DECIMAL(5,2) | Rapport signal/bruit | | 25.00 |
| lens\_thickness | DECIMAL(5,2) | Épaisseur du cristallin | | 4.00 |
| asph\_qf | DECIMAL(5,2) | Asphericité frontale de la cornée | | 0.20 |
| apsh\_qb | DECIMAL(5,2) | Asphericité postérieure de la cornée | | 0.10 |
| al\_status | DECIMAL(5,2) | Statut de la longueur axiale | | 23.75 |
| al\_error | DECIMAL(5,2) | Erreur de mesure de la longueur axiale | | 0.02 |
| k\_pre\_refr\_avg | DECIMAL(5,2) | Moyenne de la réfraction préopératoire | | 43.25 |
| r\_pre\_refr\_avg | DECIMAL(5,2) | Rayon moyen de la réfraction préopératoire | | 7.80 |
| manifest\_refr\_date | DATE | Date de la réfraction manifeste | | 2025-06-02 |
| manifest\_refr\_sph | DECIMAL(5,2) | Sphère de la réfraction manifeste | | -2.50 |
| manifest\_refr\_cyl | DECIMAL(5,2) | Cylindre de la réfraction manifeste | | 0.75 |
| manifest\_refr\_axis | DECIMAL(5,2) | Axe de la réfraction manifeste | | 90 |
| manifest\_refr\_vd | DECIMAL(5,2) | Distance vertex en mm | | 12.00 |
| target\_refr\_sph | DECIMAL(5,2) | Sphère de la réfraction cible | | -2.00 |
| target\_refr\_cyl | DECIMAL(5,2) | Cylindre de la réfraction cible | | 0.50 |
| biometer | DECIMAL(5,2) | Longueur axiale mesurée | | 23.75 |
| importType | VARCHAR(25) | Type d'importation des données | | Manuel |
| id\_calcul | INT | Identifiant du calcul associé | | 123 |

## Entité : Calcul

### Description

La table Calcul stocke les résultats des calculs biométriques nécessaires aux interventions ophtalmologiques. Ils Ces calculs permettent de déterminer la réfraction cible et l’implant intraoculaire optimal pour le patient.

### Relations et contraintes

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| 1-N avec Patient | Relation | Un patient peut avoir plusieurs calculs biométriques. |
| 1-N avec Examen | Relation | Un calcul est basé sur un examen biométrique existant. |
| 1-N avec Constant et Diopter | Relation | Un calcul peut utiliser plusieurs constantes et dioptries. |
| 1-N avec Lens | Relation | Un calcul peut proposer plusieurs implants intraoculaires. |

### Contraintes d’Intégrité

#### Un calcul doit être toujours associé à un patient (id\_patient non nul)

* Contrainte SQL :ALTER TABLE Calcul ADD CONSTRAINT fk\_calcul\_patient FOREIGN KEY (id\_patient) REFERENCES Patient(id\_patient) ON DELETE CASCADE;

#### Un calcul doit être basé sur un examen existant (id\_examen non nul)

* Contrainte SQL : ALTER TABLE Calcul ADD CONSTRAINT fk\_calcul\_examen FOREIGN KEY (id\_examen) REFERENCES Examen(id\_examen);

#### Un calcul doit avoir une constante biométrique (id\_constant non nul)

* Contrainte SQL : ALTER TABLE Calcul ADD CONSTRAINT fk\_calcul\_constant FOREIGN KEY (id\_constant) REFERENCES Constant(id\_constant);

### Détails des Attributs

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| id\_calcul | INT AUTO\_INCREMENT | Identifiant unique du calcul | 1 |
| eye\_side | VARCHAR(25) | Côté de l'œil concerné | OD |
| target\_refraction | DECIMAL(5,2) | Réfraction cible | -2.50 |
| examen | INT | Référence vers un examen | 5 |
| constants | INT | Référence vers des constantes | 8 |
| lens | INT | Référence vers une lentille | 12 |
| diopters | INT | Référence vers une dioptrie | 15 |
| id\_constant | INT | Identifiant de la constante associée | 18 |
| id\_diopter | INT | Identifiant de la dioptrie associée | 3 |
| id\_patient | INT | Identifiant du patient associé | 6 |

## Entité : UserOrganization

### Description

La table UserOrganization établit une relation entre les utilisateurs et les organisations. Elle définit les rôles des utilisateurs au sein des différentes organisations.

### Relations et contraintes

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| N-N entre User et Organizations | Relation | Un utilisateur peut être affilié à plusieurs organisations et vice-versa. |
| 1-N avec Role | Relation | Chaque utilisateur dans une organisation a un rôle spécifique (Admin, Médecin, Secrétaire). |

### Contraintes d’Intégrité

#### Un utilisateur ne peut pas être lié à plusieurs organisations avec le même rôle

* Contrainte SQL : ALTER TABLE UserOrganization ADD CONSTRAINT unique\_user\_org\_role UNIQUE (id\_user, id\_organizations);

#### Un utilisateur doit avoir une organisation définie

* Contrainte SQL : ALTER TABLE UserOrganization ADD CONSTRAINT fk\_userOrganization\_organization FOREIGN KEY (id\_organizations) REFERENCES Organizations(id\_organizations) ON DELETE CASCADE;

### Détails des Attributs

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| id\_userOrganization | INT AUTO\_INCREMENT | Identifiant unique de l'association | 10 |
| is\_admin | BOOLEAN | Indique si l'utilisateur est admin | TRUE |
| id\_user | INT | Référence vers une organisation | 2 |
| id\_organizations | INT | Référence vers l'organisation concernée | 5 |

## Entité : Link

### Description

La table Link est une table de faits qui établit les relations entre différentes entités médicales.

### Relations et contraintes

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| 1-N avec Patient, Consultation, Workflow, Planning, Examen, Protocol | Relation | Permet de lier ces entités médicales entre elles. |

### Contraintes d’Intégrité

#### Une entité liée doit exister (id\_patient, id\_consultation, etc. non nuls)

* Contrainte SQL : ALTER TABLE Link ADD CONSTRAINT fk\_link\_patient FOREIGN KEY (id\_patient) REFERENCES Patient(id\_patient);

### Détails des Attributs

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| id\_link | INT AUTO\_INCREMENT | Identifiant unique du lien. | 10 |
| id\_consultation | INT | Référence vers une consultation. | 3 |
| id\_organization | INT | Référence vers l'organisation concernée. | 5 |
| id\_examen | INT | Référence vers l’ examen. | 7 |
| id\_planning | INT | Référence vers le planning. | 4 |
| id\_workflow | INT | Référence vers le workflow. | 2 |
| id\_patient | INT | Référence vers le patient concerné. | 9 |
| id\_protocol | INT | Référence vers le protocol concerné. | 5 |

## Entité : r\_user\_token

### Description

Table d’association entre User et Token pour gérer l’authentification.

### Relations et contraintes

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| N-N entre \_User et Token | Relation | Un utilisateur peut avoir plusieurs tokens, et un token peut être associé à plusieurs utilisateurs. |

### Détails

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| id\_token | INT AUTO\_INCREMENT | Identifiant unique du fabricant | 1 |
| id\_user | VARCHAR(100) | Nom du fabricant | Essilor |

## Entité : Identified

### Description

Stocke les informations d'identification des utilisateurs et des patients. Cette table est une table de base utilisée par les entités *User* et *Patient*.

### Relations et contraintes

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| 1-1 avec User et Patient | Héritage | Chaque utilisateur et patient a une identité unique. |
| 1-1 avec Address | Relation | Chaque identité est associée à une adresse unique. |

### Contraintes d’Intégrité

#### Un patient et un utilisateur doivent avoir une identité définie

* Contrainte SQL : ALTER TABLE Patient ADD CONSTRAINT fk\_patient\_identified FOREIGN KEY (id\_identified) REFERENCES Identified(id\_identified);

### Détails des Attributs

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| id\_identified | INT AUTO\_INCREMENT | Identifiant unique | 1 |
| firstname | VARCHAR(100) | Prénom | Vincent |
| lastname | VARCHAR(100) | Nom | Vanhees |
| mail | VARCHAR(255) | Adresse e-mail | vincent.vanhees@hotmail.com |
| phone | VARCHAR(20) | Numéro de téléphone | 0499887756 |
| id\_address | INT | Référence vers une adresse | 3 |

## Entité : User

### Description

La table User stocke les informations des utilisateurs inscrits dans le système.

Cela inclut les médecins, administrateurs, secrétaires médicales et tout autre personnel ayant un accès à la plateforme.

Cette table est fortement sécurisée car elle contient des données sensibles (mots de passe, statuts d’authentification).

L’authentification repose sur un système de tokens JWT et la validation d’email se fait via un code de vérification.

### Relations

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| 1-1 avec Identified | Héritage | Chaque User possède une identité unique stockée dans Identified. |
| 1-N avec Token | Relation | Un utilisateur peut avoir plusieurs sessions actives stockées dans Token. |
| 1-N avec VerificationCode | Relation | Un utilisateur peut générer plusieurs codes de vérification stockés dans VerificationCode. |
| N-N avec Organizations via UserOrganization | Relation | Un utilisateur peut appartenir à plusieurs organisations via la table pivot UserOrganization. |
| 1-N avec Role | Relation | Un utilisateur peut avoir plusieurs rôles stockés dans Role. |

### Héritage et Contraintes d’Inclusion

L’entité User hérite de Identified et permet donc de garantir que tout utilisateur possède une identité unique définie dans Identified.

### Contraintes d’Inclusion

L’identifiant id\_identified doit être unique pour chaque utilisateur afin d’assurer qu’un utilisateur est toujours rattaché à une identité

* Contrainte SQL : ALTER TABLE User ADD CONSTRAINT fk\_user\_identified

FOREIGN KEY (id\_identified) REFERENCES Identified(id\_identified) ON DELETE CASCADE;

* Explication : Si l’entrée associée dans Identified est supprimée, l’utilisateur est supprimé aussi (ON DELETE CASCADE) et assure donc une relation 1-1 stricte entre User et Identified.
* L’email de l’utilisateur doit être valide avant activation (valid\_email = TRUE)
* Contrainte SQL : ALTER TABLE User ADD CONSTRAINT chk\_valid\_email

CHECK (valid\_email = TRUE OR verified = FALSE);

* Explication : Un utilisateur ne peut pas être activé (verified = TRUE) si son email n’a pas été validé et ne peut donc pas être supprimé s’il possède des tokens actifs

### Trigger SQL :

CREATE TRIGGER prevent\_user\_deletion

BEFORE DELETE ON User

FOR EACH ROW

WHEN EXISTS (SELECT 1 FROM Token WHERE Token.id\_user = OLD.id\_user)

BEGIN

RAISE EXCEPTION 'Impossible de supprimer un utilisateur avec des tokens actifs';

END;

* Explication : Cela empêche la suppression accidentelle d’un utilisateur qui a encore des sessions ouvertes.

### Détails des Attributs

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| id\_user | INT AUTO\_INCREMENT | Identifiant unique de l'utilisateur | 1 |
| password | VARCHAR(255) | Mot de passe hashé | \*\*\*\*\*\* |
| verified | BOOLEAN | Indique si l'utilisateur est vérifié | TRUE |
| valid\_email | BOOLEAN | Indique si l'adresse e-mail est valide | TRUE |
| has\_read\_terms\_and\_conditions | BOOLEAN | Indique si l'utilisateur a lu les CGU | TRUE |
| token | VARCHAR(255) | Jeton d'authentification de l'utilisateur | eyJhbghuiozerb |
| user\_organization | VARCHAR(200) | Organisation de l'utilisateur | ADMIN |
| id\_token | INT | Référence vers un token | 4 |
| id\_verification\_code | INT | Référence vers un code de vérification | 5 |
| id\_identified | INT | Référence vers une identité associée | 2 |

## Entité : Workflow

### Description

La table Workflow représente les flux de travail médicaux liés aux examens, calculs et interventions chirurgicales.

### Relations et Contraintes

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| 1-N avec Link | Relation | Un workflow peut être associé à plusieurs entités médicales. |

### Contraintes d’Intégrité

#### Un workflow doit être unique et avoir un nom distinct.

* Contrainte SQL : ALTER TABLE Workflow ADD CONSTRAINT unique\_workflow\_name UNIQUE (nom);

### Détails des Attributs

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| d\_workflow | INT AUTO\_INCREMENT | Identifiant unique du workflow | 1 |
| nom | VARCHAR(200) | Nom du workflow | pré-chirurgie |

## Entité : Consultation

### Description

La table Consultation stocke les consultations médicales effectuées par les praticiens.

### Relations et contraintes

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| 1-N avec Link | Relation | Une consultation peut être liée à plusieurs workflows et examens. |

### Détails des Attributs

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| id\_consultation | INT AUTO\_INCREMENT | Identifiant unique de la consultation | 1 |

## Entité : Protocol

### Description

La table Protocol stocke les protocoles médicaux liés aux patients et aux interventions.

### Relations et contraintes

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| 1-N avec Link | Relation | Un protocole peut être associé à plusieurs examens et traitements. |

### Détails des Attributs

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| id\_protocol | INT AUTO\_INCREMENT | Identifiant unique du protocole | 1 |
| protocol\_datas | VARCHAR(255) | Données du protocole | "Préparation chirurgie" |

## Entité : Planning

### Description

La table Planning stocke les informations de planification médicale (rendez-vous, interventions, etc.).

### Relations et contraintes

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| 1-N avec Link | Relation | Un planning peut être associé à plusieurs consultations et workflows. |

### Détails des Attributs

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| id\_planning | INT AUTO\_INCREMENT | Identifiant unique du planning | 1 |
| planning\_datas | VARCHAR(255) | Données du planning | "RDV 10h - Chirurgie" |

## Entité : Role

### Description

La table Role définit les rôles des utilisateurs dans l’application (Admin, Médecin, Secrétaire, etc.).

### Relations et contraintes

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| 1-N avec User | Relation | Un utilisateur peut avoir plusieurs rôles. |

### Détails des attributs

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| id\_role | INT AUTO\_INCREMENT | Identifiant unique du rôle | 1 |
| name | VARCHAR(200) | Nom du rôle | ADMIN |
| id\_user | INT | Référence vers un utilisateur | 2 |

## Entité : Lens

### Description

La table Lens stocke les informations sur les lentilles intraoculaires utilisées en chirurgie.

### Relations et contraintes

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| 1-N avec LensManufacturer | Relation | Une lentille est fabriquée par un seul fabricant. |
| 1-N avec Calcul | Relation | Un calcul peut proposer plusieurs lentilles. |

### Détails

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| id\_lens | INT AUTO\_INCREMENT | Identifiant unique de la lentille | 1 |
| name | VARCHAR(50) | Nom de la lentille | FineVision |
| comment\_trade\_name | VARCHAR(50) | Nom commercial de la lentille | FineVision Toric |
| nominal | VARCHAR(50) | Nominal | 50 |
| haigis\_A0 | DECIMAL(5,2) | Constante Haigis A0 | 1.45 |
| haigis\_A1 | DECIMAL(5,2) | Constante Haigis A1 | 0.4 |
| haigis\_A2 | DECIMAL(5,2) | Constante Haigis A2 | 0.1 |
| hoffer\_QPACD | DECIMAL(5,2) | Constante Hoffer QPACD | 4.00 |
| holladay\_1SF | DECIMAL(5,2) | Constante Holladay 1 SF | 118.00 |
| srkta | DECIMAL(5,2) | Constante SRK/T | 118.00 |
| haigis\_A0\_optimized | DECIMAL(5,2) | Constante Haigis A0 optimisée | 1.48 |
| haigis\_A1\_optimized | DECIMAL(5,2) | Constante Haigis A1 optimisée | 0.41 |
| haigis\_A2\_optimized | DECIMAL(5,2) | Constante Haigis A2 optimisée | 0.11 |
| hoffer\_QPACD\_optimized | DECIMAL(5,2) | Constante Hoffer QPACD optimisée | 4.05 |
| holladay\_1SF\_optimized | DECIMAL(5,2) | Constante Holladay 1 SF optimisée | 118.50 |
| srkta\_optimized | DECIMAL(5,2) | Constante SRK/T optimisée | 118.50 |
| castrop\_C\_optimized | DECIMAL(5,2) | Constante Castrop C optimisée | 0.3 |
| castrop\_H\_optimized | DECIMAL(5,2) | Constante Castrop H optimisée | 0.2 |
| castro\_R\_optimized | DECIMAL(5,2) | Constante Castrop R optimisée | 0.1 |
| lens\_manufacturer | DECIMAL(5,2) | Référence vers le fabricant | 2 |
| calcul | VARCHAR(250) | Description du calcul appliqué | SRK/T |
| id\_calcul | INT | Référence vers le calcul associé | 3 |

## Entité : Constant

### Description

La table Constant stocke les constantes biométriques utilisées dans les calculs.

### Relations et contraintes

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| 1-N avec Calcul | Relation | Un calcul peut utiliser plusieurs constantes. |

### Détails des Attributs

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| id\_constant | INT AUTO\_INCREMENT | Identifiant unique de la constante | 1 |
| constant\_type | VARCHAR(50) | Type de constante utilisée dans les calculs | Haigis |
| value | VARCHAR(50) | Valeur de la constante | 1.55 |
| forumla | VARCHAR(50) | Formule associée | SRK/T |
| Calcul | VARCHAR(250) | Description du calcul appliqué | SRK/T Lens |

## Entité : Diopter

### Description

La table Diopter stocke les dioptries utilisées dans les calculs intraoculaires.

### Relations et contraintes

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| 1-N avec Calcul | Relation | Un calcul peut générer plusieurs dioptries. |

### Détails des Attributs

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| id\_diopter | INT AUTO\_INCREMENT | Identifiant unique de la dioptrie | 1 |
| iol\_power | DECIMAL(5,2) | Puissance de l'implant intraoculaire | 21.50 |
| value | DECIMAL(5,2) | Valeur de la dioptrie | 2.00 |
| formula | VARCHAR(50) | Formule de calcul de la dioptrie | Haigis |
| calcul | VARCHAR(250) | Description du calcul appliqué | SRK/T |

## Entité : LensManufacturer

### Description

La table LensManufacturer stocke les fabricants de lentilles intraoculaires.

### Relations et contraintes

|  |  |  |
| --- | --- | --- |
| Relation | Type | Explication |
| 1-N avec Lens | Relation | Un fabricant peut produire plusieurs lentilles. |

### Détails des Attributs

|  |  |  |  |
| --- | --- | --- | --- |
| Attributs | Types | Définition | Exemples |
| id\_lens\_manufacturer | INT AUTO\_INCREMENT | Identifiant unique du fabricant | 1 |
| name | VARCHAR(50) | Nom du fabricant | "Essilor" |
| Lenses | VARCHAR(50) | Lentilles fabriquées | "FineVision" |
| id\_lens | INT | Référence vers une lentille | 3 |

# Flux des données

## Inscription et gestion des utilisateurs

### Création d’un compte utilisateur

Un nouvel utilisateur (ex. médecin, administrateur, secrétaire) crée un compte et s'authentifie pour accéder au système.

#### Étapes du processus

* L’utilisateur remplit le formulaire d’inscription (nom, prénom, email, rôle, etc.).
* Le système génère une entrée dans la table \_User avec un identifiant unique (id\_user).
* Un code de vérification est généré et stocké dans VerificationCode, puis envoyé à l’email fourni.
* L’utilisateur reçoit l’email et saisit son code pour valider son inscription.

#### Le système vérifie le code :

* Si valide → Active le compte (User.verified = TRUE).
* Si invalide → Affiche un message d’erreur.

#### L’utilisateur se connecte en saisissant ses identifiants :

* Le système vérifie les informations (email et password).
* Un token JWT est généré (Token) et stocké en base.
* L’utilisateur peut maintenant naviguer dans l’application selon son rôle (UserOrganization).

#### Flux des données et relations utilisées

* User → Stocke l’utilisateur.
* VerificationCode → Assure la validation de l’inscription.
* Token → Authentifie l’utilisateur.
* UserOrganization → Définit le rôle et les permissions dans une organisation.

## Ajout d’un Patient et gestion des organisations

### Enregistrement d’un nouveau patient

Objectif : Ajouter un patient dans la base et l’associer à une organisation médicale.

#### Étapes du processus

* Un utilisateur (ex. médecin) accède à l’interface d’ajout de patient.
* Il saisit les informations du patient (nom, prénom, date de naissance, etc.).
* Le système crée une entrée dans Patient avec un identifiant unique (id\_patient).
* Le patient est affilié à une ou plusieurs organisations médicales (Organizations).
* L’association est enregistrée dans la table UserOrganization.
* Le patient peut maintenant avoir des examens biométriques et des calculs ophtalmologiques associés.

#### Flux des données et relations utilisées

* Patient → Stocke les données du patient.
* Organizations → Définit les centres médicaux où le patient est suivi.
* UserOrganization → Lie un patient à une organisation médicale.

## Enregistrement des Examens Biométriques

### Un patient réalise un examen ophtalmologique

Objectif : Stocker les mesures biométriques d’un patient après un examen médical.

#### Étapes du processus

* Un spécialiste réalise un examen biométrique avec des instruments ophtalmologiques.
* Il renseigne les valeurs biométriques dans le système :
* Courbure de la cornée
* Longueur axiale
* Profondeur de la chambre antérieure
* Épaisseur du cristallin
* etc.
* Le système crée une entrée dans Examen et la lie au patient via id\_patient.
* Une analyse des données est réalisée.
* L’examen peut être utilisé pour faire des calculs intraoculaires (Calcul).

### Flux des données et relations utilisées

* Examen → Stocke les valeurs mesurées.
* Patient → Permet d’identifier le patient concerné.
* Calcul → Utilise ces données pour simuler l’implantation d’une lentille.

## Calcul et Simulation des Implants Intraoculaires

### Génération d’un Calcul Intraoculaire

Objectif : Analyser les données biométriques pour déterminer la meilleure lentille pour le patient.

#### Étapes du processus

* Un médecin sélectionne un examen biométrique existant (Examen).
* Il lance un calcul intraoculaire (Calcul) basé sur :
* La réfraction cible
* Les valeurs biométriques
* Les formules mathématiques utilisées (Constant)
* Le système génère plusieurs simulations de correction :
* Différentes Diopter (puissances de correction) sont testées.
* Différentes Constant (paramètres optiques) sont appliquées.
* Une ou plusieurs lentilles intraoculaires (Lens) sont proposées.
* Le médecin choisit l’implant optimal, qui est alors associé au calcul.

#### Flux des données et relations utilisées

* Calcul → Contient les valeurs calculées.
* Diopter → Stocke les dioptries testées.
* Constant → Applique les constantes mathématiques aux formules.
* Lens → Propose différentes lentilles intraoculaires adaptées.

## Gestion des Lentilles Intraoculaires

### Sélection et commande d’une lentille

Objectif : Confirmer le choix de la lentille à implanter et gérer son fabricant.

#### Étapes du processus

* Une lentille intraoculaire est sélectionnée pour un patient.
* L’association est enregistrée dans LensManufacturer.
* Le fabricant de la lentille est identifié.
* La commande est validée et envoyée au fournisseur.

#### Flux des données et relations utilisées

* Lens → Identifie la lentille choisie.
* LensManufacturer → Fournisseur de la lentille.
* Calcul → Associe le calcul intraoculaire à la lentille sélectionnée.

## Gestion des Accès et Sécurité

### Authentification et Permissions

Objectif : Contrôler l’accès des utilisateurs selon leur rôle.

#### Étapes du processus

* Un utilisateur tente de se connecter avec son email et son mot de passe.
* Le système vérifie les informations d’identification (\_User).
* Un token JWT est généré (Token) et stocké pour gérer la session.
* Le rôle de l’utilisateur est récupéré (UserOrganization) :
* Si ADMIN → accès à tout le système.
* Si MEDECIN → accès aux patients et calculs.
* Si SECRETAIRE → accès limité aux dossiers patients.
* L’utilisateur navigue dans l’application selon ses droits.

#### Flux des données et relations utilisées

* User → Gère l’identité de l’utilisateur.
* Token → Authentifie la session.
* UserOrganization → Définit les permissions selon l’organisation.

# Évolutivité et Scalabilité

## Modularité et Extensibilité

### Ajout de nouvelles fonctionnalités métier :

Grâce à la structure relationnelle, l’extension du modèle est facilitée. De nouveaux types de calculs peuvent être intégrés simplement en ajoutant des entrées dans Constant et Diopter, sans impacter les structures existantes.

### Évolutivité de l’examen biométrique :

La table Examen peut être enrichie pour inclure d'autres mesures biométriques (par exemple, de nouveaux paramètres liés aux diagnostics ophtalmologiques).

Ajout de nouvelles méthodes d’analyse via la table Calcul en intégrant de nouvelles formules ou modèles de prédiction.

### Gestion avancée des organisations :

Possibilité d’ajouter facilement de nouveaux types d’organisations médicales dans Organizations.

Flexibilité pour gérer plusieurs établissements pour un même utilisateur grâce à UserOrganization.

* Interopérabilité avec des systèmes tiers :

La base de données permet la connexion à des APIs médicales externes pour importer automatiquement les données des appareils médicaux (OCT, topographes cornéens, etc.).

Support de FHIR/HL7 pour assurer une compatibilité avec d’autres systèmes hospitaliers.

## Intégration avec des outils externes

### API REST sécurisée :

Exposition des données via une API RESTful pour permettre aux médecins et administrateurs d'accéder aux informations en temps réel.

Authentification via JWT avec gestion des rôles utilisateurs (Role et UserOrganization).

### Exportation des données médicales :

Génération de rapports compatibles HL7/FHIR pour assurer la standardisation et l’interopérabilité avec d’autres applications médicales.

Fonctionnalité d’exportation des examens et calculs sous format PDF ou JSON pour intégration avec les dossiers médicaux électroniques (DME).

### Connexion à des services de calcul avancés :

Possibilité d’intégrer des algorithmes d’IA pour l’analyse prédictive des résultats biométriques.

Déploiement d’outils de Machine Learning pour optimiser le choix des implants intraoculaires (Lens).

# Conclusion

Cette nouvelle base de données a été conçue pour répondre aux exigences métier spécifiques du suivi médical ophtalmologique tout en garantissant une flexibilité et une évolutivité à long terme.

Les principales forces de cette architecture sont :

1. Une structuration efficace des données médicales :

La base garantit une traçabilité complète des patients, examens, calculs biométriques et implants intraoculaires. L’organisation des données assure un accès optimisé aux informations les plus critiques.

1. Une intégrité et cohérence renforcées :

Gestion robuste des relations entre les entités (via les clés étrangères et les contraintes d’intégrité).

Réduction des risques d’erreurs et garantie de la cohérence des données médicales.

1. Des performances optimales :

Utilisation d’index et partitionnement pour améliorer la rapidité des requêtes SQL.

Possibilité d’implémenter des vues matérialisées pour pré-calculer des données complexes et accélérer leur restitution.

1. Une sécurité avancée :

Chiffrement des mots de passe des utilisateurs.

Gestion des sessions sécurisées via JWT et système d’authentification avancé.

Contrôle des accès basé sur les rôles (Role) et les organisations (UserOrganization).

1. Interopérabilité et évolutivité :

Compatibilité HL7/FHIR pour échanger les données avec d’autres systèmes hospitaliers.

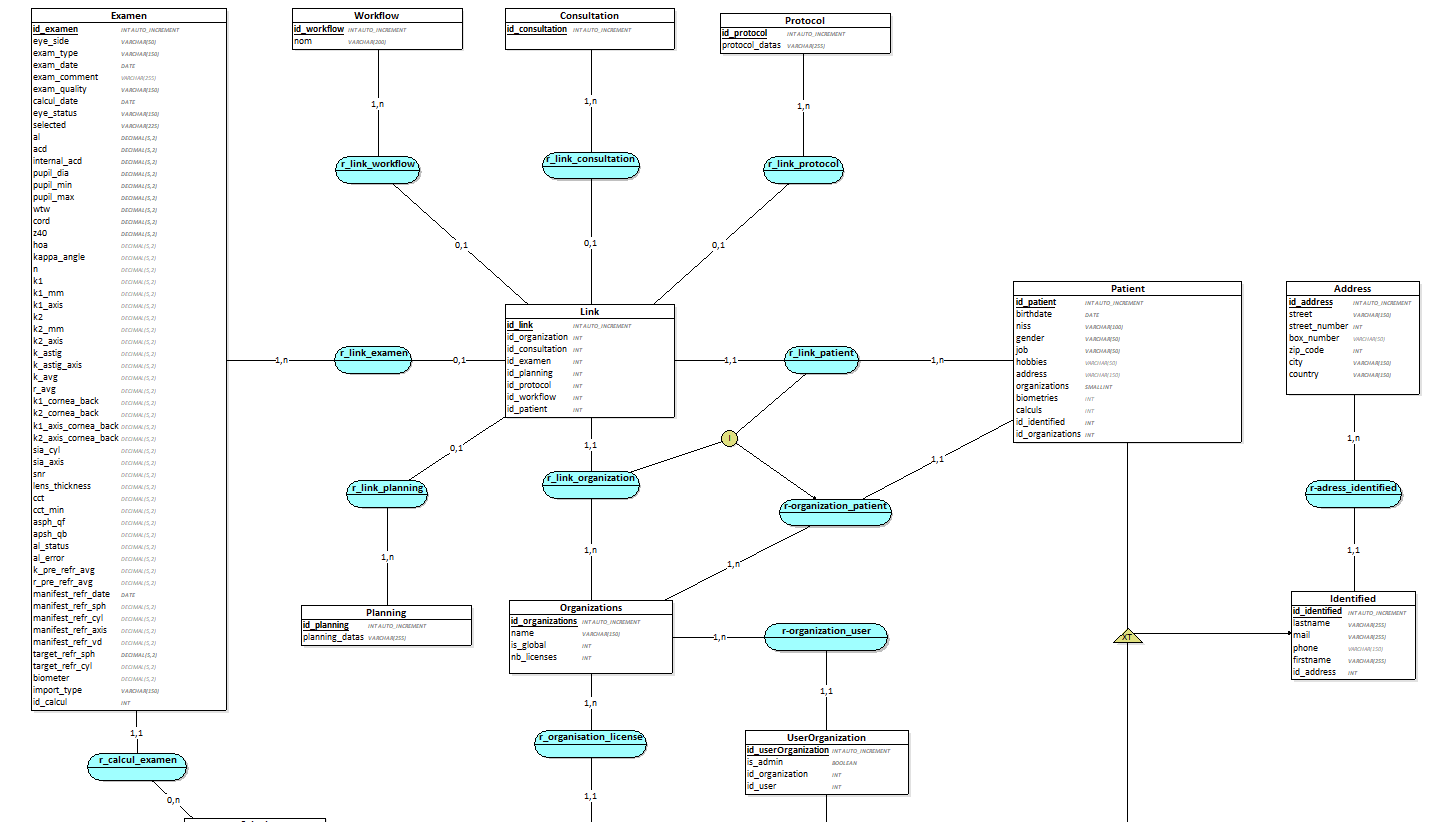
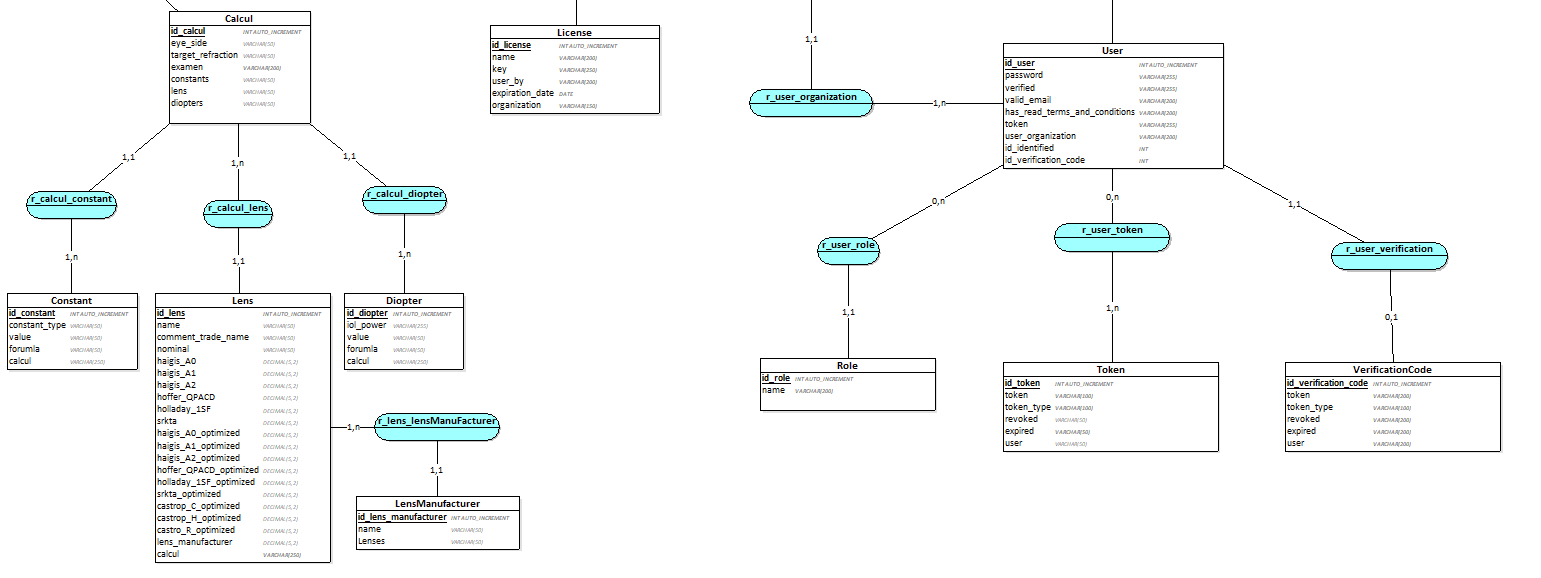
Connexion à des APIs médicales pour enrichir les données via l’importation des équipements médicaux.

Support des calculs avancés via des algorithmes de Machine Learning pour améliorer la précision des diagnostics et des prescriptions de lentilles intraoculaires.

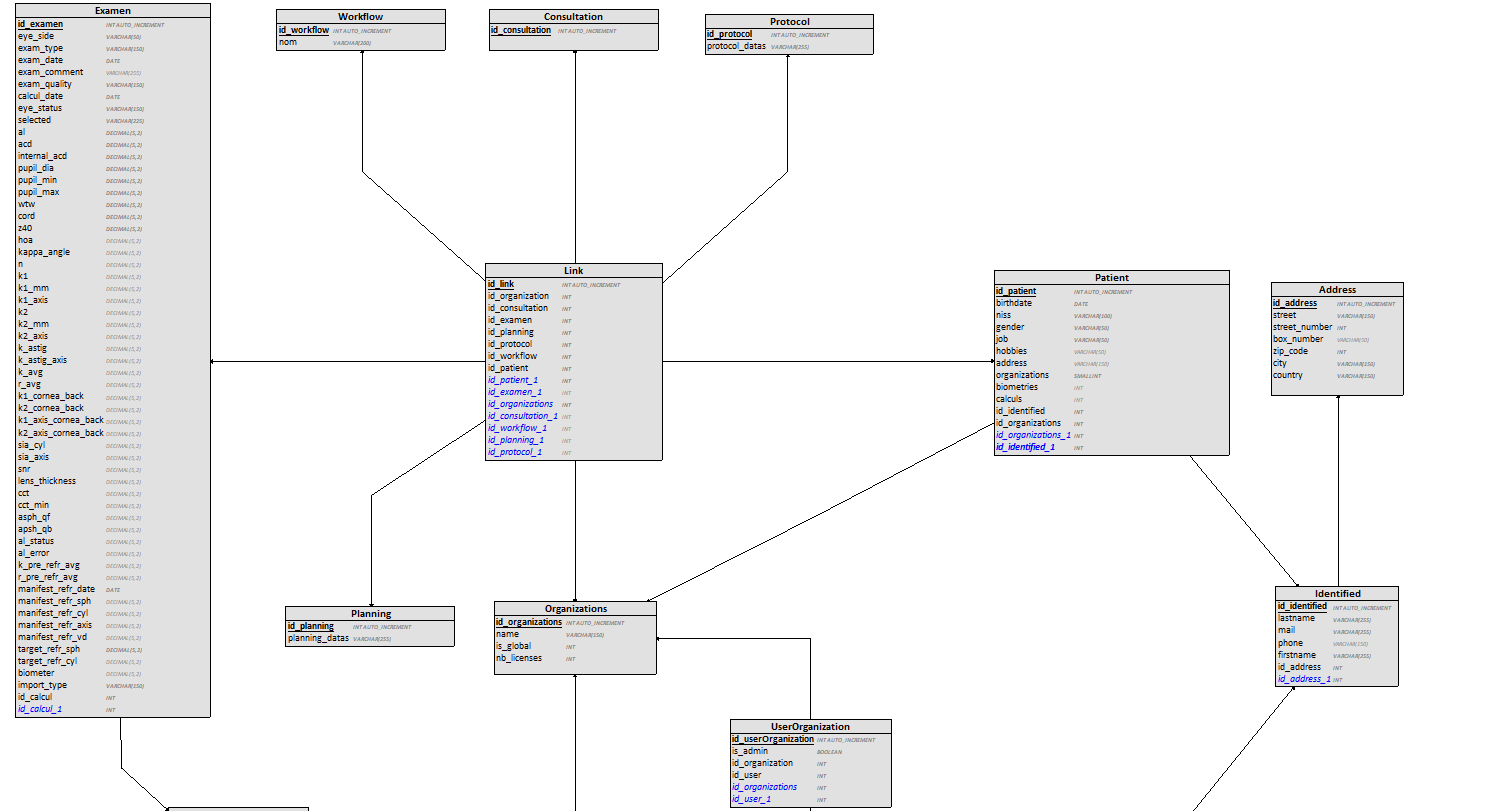
En conclusion, cette base de données pose les bases solides d’un système de gestion avancé pour les données médicales ophtalmologiques. Son architecture modulaire et sécurisée permet une intégration fluide avec d’autres solutions et garantit une expérience utilisateur optimale pour les médecins, secrétaires et administrateurs du système.

# Annexes

## Modèle Conceptuel de Données (MCD)



## Modèle Logique de Données (MLD)



## Unified Modeling Language (UML)

