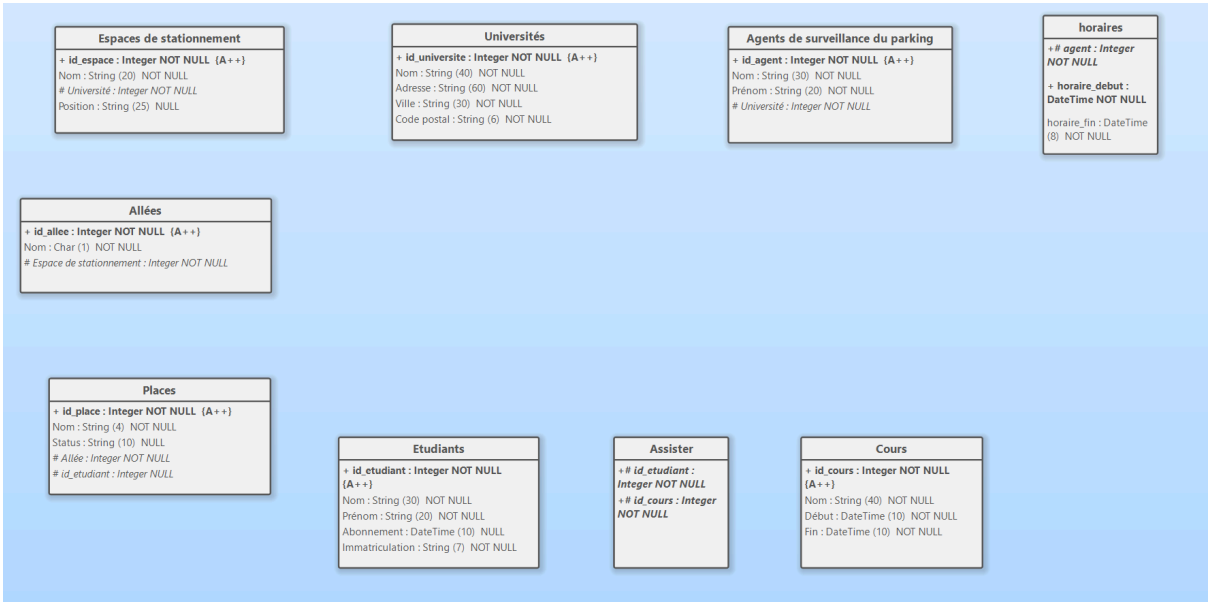
CHAN WITZEL Marco
LECERF Martin
HATTON Lilian
POCHART Clément

TP 2 - GESTIONNAIRE DE STATIONNEMENT

1) Normalisation de votre modèle logique des données

En reprenant notre modèle de langage des données, nous avons revu notre modèle en fonction des formes normales à appliquer et avons réalisé de légers correctifs afin d'atteindre la Forme Normale de Boyce-Codd suivante.



2) Transformation de votre MLD en MPD

Universités(id_universite, Nom, Adresse, Ville, Code postal)

Espaces de stationnement(id_espace, #id_universite, Nom, Position)

Allées(id_allee, #id_espace, Nom)

Places(id_place, #id_allee, #id_etudiant, Nom, Status,)

Agents de surveillance du parking(id_agent, #id_universite, Nom, Prénom, Horaires)

horaires(#agent, heure_debut, heure_fin)

Étudiants(id_etudiant, Nom, Prénom, Abonnement, Immatriculation)

Cours(id_cours, Nom, Début, Fin)

Assister(#id_etudiant, #id_cours)

CHAN WITZEL Marco
LECERF Martin
HATTON Lilian
POCHART Clément

3) **Mise à jour du dictionnaire de données (DD)**

Universités

Colonne	Type	PK	FK	Contraintes	Description	Règles de gestion
id_universite	INT	YES	NO	NOT_NULL	Identifiant unique de l'université	ID doit être unique
Nom	VARCHAR(40)	NO	NO	NOT_NULL	Dénomination de l'université	Doit contenir uniquement des caractères alphabétiques
adresse	VARCHAR(60)	NO	NO	NOT_NULL	Adresse de l'université	Doit respecter un format d'adresse valide (ex : 1 Avenue rue).
Ville	VARCHAR(30)	NO	NO	NOT_NULL	Ville de l'université	Doit contenir uniquement des caractères alphabétiques
Code postal	VARCHAR(6)	NO	NO	NOT_NULL	Code postal de l'université	Doit contenir uniquement des caractères alphanumérique

Espace de stationnement

Colonne	Type	PK	FK	Contraintes	Description	Règles de gestion
id_espace	INT	YES	NO	NOT_NULL	Identifiant unique de l'université	ID doit être unique
universite	INT	NO	YES (Universités.id_universite)	NOT_NULL, FOREIGN KEY	Identifiant de l'université dont le stationnement dépend, référence la table Universités.	Doit correspondre à un id_universite existant.
Nom	VARCHAR(20)	NO	NO	NOT_NULL	Nom de l'espace de stationnement	Doit contenir uniquement des caractères

CHAN WITZEL Marco
LECERF Martin
HATTON Lilian
POCHART Clément

						alphabétiques
Position	VARCHAR(25)	NO	NO	NULL	Où est géographiquement situé l'espace de stationnement	Peut être null

Allées

Colonne	Type	PK	FK	Contraintes	Description	Règles de gestion
id_allee	INT	YES	NO	NOT_NULL	Identifiant unique de l'allée	ID doit être unique
espace_stationnement	INT	NO	YES (Espace de stationnement.id_stationnement)	NOT_NULL, FOREIGN KEY	Identifiant de l'espace de stationnement dont dépend l'allée, référence la table Espace de stationnement.	Doit correspondre à un id_espace existant.
Nom	CHAR(1)	NO	NO	NOT_NULL	Nom de l'allée	Doit contenir uniquement un caractères alphabétiques ("A")

Places

Colonne	Type	PK	FK	Contraintes	Description	Règles de gestion
id_place	INT	YES	NO	NOT_NULL	Identifiant unique de la place	ID doit être unique
Allée	INT	NO	YES (Allées.id_allee)	NOT_NULL, FOREIGN KEY	Identifiant de l'allée dont dépend la place, référence la table Allées	Doit correspondre à un id_allée existant.
id_etudiant	INT	NO	YES (Etudiants.id_etudiant)	NULL, FOREIGN KEY	Identifiant de l'étudiant ayant réserver la place (si réserver cf status), référence la table Etudiants	Doit correspondre à un id_étudiants existant. Null si et seulement si Status null
Nom	SmallInteger(3)	NO	NO	NOT_NULL	Nom de la place	Doit contenir uniquement des

CHAN WITZEL Marco
LECERF Martin
HATTON Lilian
POCHART Clément

						caractères alphabétiques
Status	VARCHAR(10)	NO	NO	NULL	Status de la place	Doit contenir uniquement des caractères alphabétiques

Agents de surveillance du parking

Colonne	Type	PK	FK	Contraintes	Description	Règles de gestion
id_agent	INT	YES	NO	NOT_NULL	Identifiant unique de l'agent	ID doit être unique
id_universite	INT	NO	YES (Universités.id_universite)	NOT_NULL, FOREIGN KEY	Identifiant de l'université dont dépend l'agent, référence la table Universités	Doit correspondre à un id_universite existant.
Nom	VARCHAR(30)	NO	NO	NOT_NULL	Nom de l'agent	Doit contenir uniquement des caractères alphabétiques
Prénom	VARCHAR(20)	NO	NO	NOT_NULL	Prénom de l'agent	Doit contenir uniquement des caractères alphabétiques

Horaires

Colonne	Type	PK	FK	Contraintes	Description	Règles de gestion
id_agent	INT	YES	YES	NOT_NULL, FOREIGN KEY	Identifiant unique de l'agent	ID doit être unique
horaire_debut	DATETIME	YES	NO	NOT_NULL	Date et heure de commencement de l'agent	Doit être du type DATETIME
horaire_fin	DATETIME	NO	NO	NOT_NULL	Date et heure de fin de journée de l'agent	Doit être du type DATETIME

8TRD151 - Introduction aux bases de données

CHAN WITZEL Marco
LECERF Martin
HATTON Lilian
POCHART Clément

Etudiants

Colonne	Type	PK	FK	Contraintes	Description	Règles de gestion
id_etudiant	INT	YES	NO	NOT_NULL	Identifiant unique de l'étudiant	ID doit être unique
Nom	VARCHAR(30)	NO	NO	NOT_NULL	Nom de l'étudiant	Doit contenir uniquement des caractères alphabétiques
Prénom	VARCHAR(20)	NO	NO	NOT_NULL	Prénom de l'étudiant	Doit contenir uniquement des caractères alphabétiques
Abonnements	DATETIME	NO	NO	NULL	Temps avant la fin de l'abonnement (0 si non abonné)	Doit être du type DATETIME
Immatriculation	VARCHAR(7)	NO	NO	NOT_NULL	Immatriculations du vehicule de l'etudiant	Doit respecter un format d'immatriculation valide selon les normes locales.

Cours

Colonne	Type	PK	FK	Contraintes	Description	Règles de gestion
id_cours	INT	YES	NO	NOT_NULL	Identifiant unique du cours	ID doit être unique
Nom	VARCHAR(40)	NO	NO	NOT_NULL	Nom du cours	Doit contenir uniquement des caractères alphanumériques
Début	DATETIME	NO	NO	NOT_NULL	date et heure du début du cours	Doit être l'heure du début du cours et du type DATETIME
Fin	DATETIME	NO	NO	NOT_NULL	date et heure du fin du cours	Doit être l'heure de fin du cours et du type DATETIME

CHAN WITZEL Marco
LECERF Martin
HATTON Lilian
POCHART Clément

Assister

4) **Création des scripts SQL de création des tables**

```
CREATE TABLE IF NOT EXISTS Universites (
   id_universite     INT NOT NULL AUTO_INCREMENT,
   Nom               VARCHAR(40) NOT NULL,
   Adresse    VARCHAR(60) NOT NULL,
   Ville             VARCHAR(30) NOT NULL,
   Code_postal    VARCHAR(6) NOT NULL,
   PRIMARY KEY (id_universite)
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS Agents_de_surveillance_du_parking(
   id_agent    INT NOT NULL AUTO_INCREMENT,
   Nom               VARCHAR(30) NOT NULL,
   Prenom    VARCHAR(20) NOT NULL,
   universite    INT NOT NULL,
   PRIMARY KEY (id_agent),
   FOREIGN KEY (universite) REFERENCES Universites(id_universite)
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS horaires(
   agent             INT NOT NULL,
   horaire_debut    DATETIME NOT NULL,
   horaire_fin    DATETIME NOT NULL,
   PRIMARY KEY (agent, horaire_debut),
   FOREIGN KEY (agent) REFERENCES Agents_de_surveillance_du_parking(id_agent)
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS Espaces_de_stationnement(
   id_espace    INT NOT NULL AUTO_INCREMENT,
   Nom               VARCHAR(20) NOT NULL,
   universite    INT NOT NULL,
   PRIMARY KEY (id_espace),
   FOREIGN KEY (universite) REFERENCES Universites(id_universite)
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS Allees(
   id_allee    INT NOT NULL AUTO_INCREMENT,
   Nom               VARCHAR(20) NOT NULL,
```

CHAN WITZEL Marco
LECERF Martin
HATTON Lilian
POCHART Clément

```
    Espaces_de_stationnement    INT NOT NULL,
    PRIMARY KEY (id_allee),
    FOREIGN KEY (Espaces_de_stationnement) REFERENCES
Espaces_de_stationnement(id_espace)
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS Etudiants(
    id_etudiant    INT NOT NULL AUTO_INCREMENT,
    Nom            VARCHAR(30) NOT NULL,
    Prenom    VARCHAR(20) NOT NULL,
Abonnement    DATETIME,
Immatriculation    VARCHAR(7) NOT NULL,
    PRIMARY KEY (id_etudiant)
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS Places(
    id_place    INT NOT NULL AUTO_INCREMENT,
    Nom            VARCHAR(4) NOT NULL,
    Status         VARCHAR(10) NOT NULL,
Allee    INT NOT NULL,
id_etudiant INT NOT NULL,
FOREIGN KEY (id_etudiant) REFERENCES Etudiants(id_etudiant),
    PRIMARY KEY (id_place),
    FOREIGN KEY (Allee) REFERENCES Allees(id_allee)
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS Cours(
    id_cours    INT NOT NULL AUTO_INCREMENT,
    Nom            VARCHAR(40) NOT NULL,
Debut              DATETIME,
Fin       DATETIME,
    PRIMARY KEY (id_cours)
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS Assister(
    id_etudiant    INT NOT NULL,
    id_cours    INT NOT NULL,
    PRIMARY KEY (id_etudiant, id_cours),
FOREIGN KEY (id_etudiant) REFERENCES Etudiants(id_etudiant),
FOREIGN KEY (id_cours) REFERENCES Cours(id_cours)
) ENGINE=InnoDB;
```

5) **Documentation et commentaires**

8TRD151 - Introduction aux bases de données

CHAN WITZEL Marco
LECERF Martin
HATTON Lilian
POCHART Clément

Pour la question 1, quelques modifications ont été nécessaires pour notamment normaliser la table des agents de surveillance, cela à nécessité l'ajout d'une autre table pour stocker les horaires des agents. Une erreur s'est glissée dans la table des cours avec une liste d'étudiants qui aurait dû être supprimée car le lien était déjà fait avec la table Assister. Il à aussi fallu modifier le type des clefs primaires vers int pour permettre l'auto incrémentation. Autrement le document était déjà normalisé.

Pour la question 2 il "suffit" donc de prendre les relations, les clés primaire et étrangère, les attributs de chaque occurrence et de récrire sous forme Table(CléPrimaire, #CléÉtrangère, AutreColonne) avec ici Table la relation

Après avoir fait cela, pour la question 3 nous devons reporter les informations du modèle physique de données de la question 2 dans le tableau, les contraintes étant déjà écrites dans le modèle logique de données de la question et les descriptions étant déjà faites lors du dictionnaire de données du TP1.
Il faut donc identifier les règles de gestion en reprenant l'analyse de la donnée faites lors du TP1, les besoins du "client" et les modifications apportées depuis.

Pour la question 4, il s'agit de traduire le MLD en sql, chaque requête créée individuellement les tables en faisant attention à l'ordre si il existe des clefs étrangères la table pointée doit être créé avant.