EC3 / Bases de données - SQL
Devoir Maison
Fabien Pageot

Contexte

Le **Festival de Jazz de la Villette** est un événement musical qui se déroule chaque année à Paris. Pendant une dizaine de jours, plusieurs scènes accueillent des concerts regroupant différents artistes venus du monde entier. Le festival attire des milliers de spectateurs (appelés festivaliers) qui achètent leurs billets à l'avance ou sur place.
L'organisation repose également sur de nombreux membres du personnel (bénévoles, techniciens, agents de sécurité, etc.), assignés à différents concerts pour assurer la logistique et la sécurité.

L'objectif est de concevoir une base de données relationnelle permettant de gérer :

- les artistes programmés,
- les concerts (date, heure, scène, artistes associés),
- les festivaliers et leurs billets,
- le personnel affecté aux concerts.

Règles de gestion métier

Un **festivalier** peut assister à plusieurs **concerts**, et chaque **concert** peut accueillir plusieurs **festivaliers**.

Un **concert** se déroule obligatoirement sur une seule **scène**, et chaque **scène** peut accueillir plusieurs **concerts**.

Un **artiste** peut jouer dans plusieurs **concerts**, et chaque **concert** peut accueillir plusieurs **artistes**.

Un **membre du personnel** peut travailler sur plusieurs **concerts**, et chaque **concert** mobilise plusieurs **personnels**.

Chaque **concert** possède une **date** et une **heure de début** qui doivent être uniques pour ce **concert**.

Un **festivalier** est identifié par son **nom**, **prénom** et **adresse e-mail**.

Un **artiste** est identifié par son **nom**, son **style musical** et son **pays**.

Un **membre du personnel** est identifié par son **nom**, **prénom** et **fonction**.

Modèle Conceptuel de données

Entités

Artistes : chaque musicien ou groupe programmé au festival.
→ Attributs : nom, style musical, pays d'origine.

Scènes : lieux physiques où se déroulent les concerts.
→ Attributs : nom, capacité d'accueil.

Concerts : chaque événement programmé (un ou plusieurs artistes sur une scène à un horaire donné).
→ Attributs : date, heure de début.

Festivaliers : spectateurs assistant au festival.
→ Attributs : nom, prénom, adresse e-mail, numéro de pass.

Personnels : personnes travaillant pour l'organisation du festival (techniciens, bénévoles, agents de sécurité).
→ Attributs : nom, fonction.

Associations

Programmer (Concerts ↔ Scènes) : un concert est programmé sur une scène.

Jouer (Artistes ↔ Concerts) : un artiste joue lors d'un concert.
→ Attributs : ordre de passage, durée prévue.

Assister à (Festivaliers ↔ Concerts) : un festivalier assiste à un concert.
→ Attributs : type de billet, date d'achat.

Travailler sur (Personnel ↔ Concerts) : un membre du personnel travaille sur un concert.
→ Attributs : rôle, horaire de prise de poste.

Cardinalités

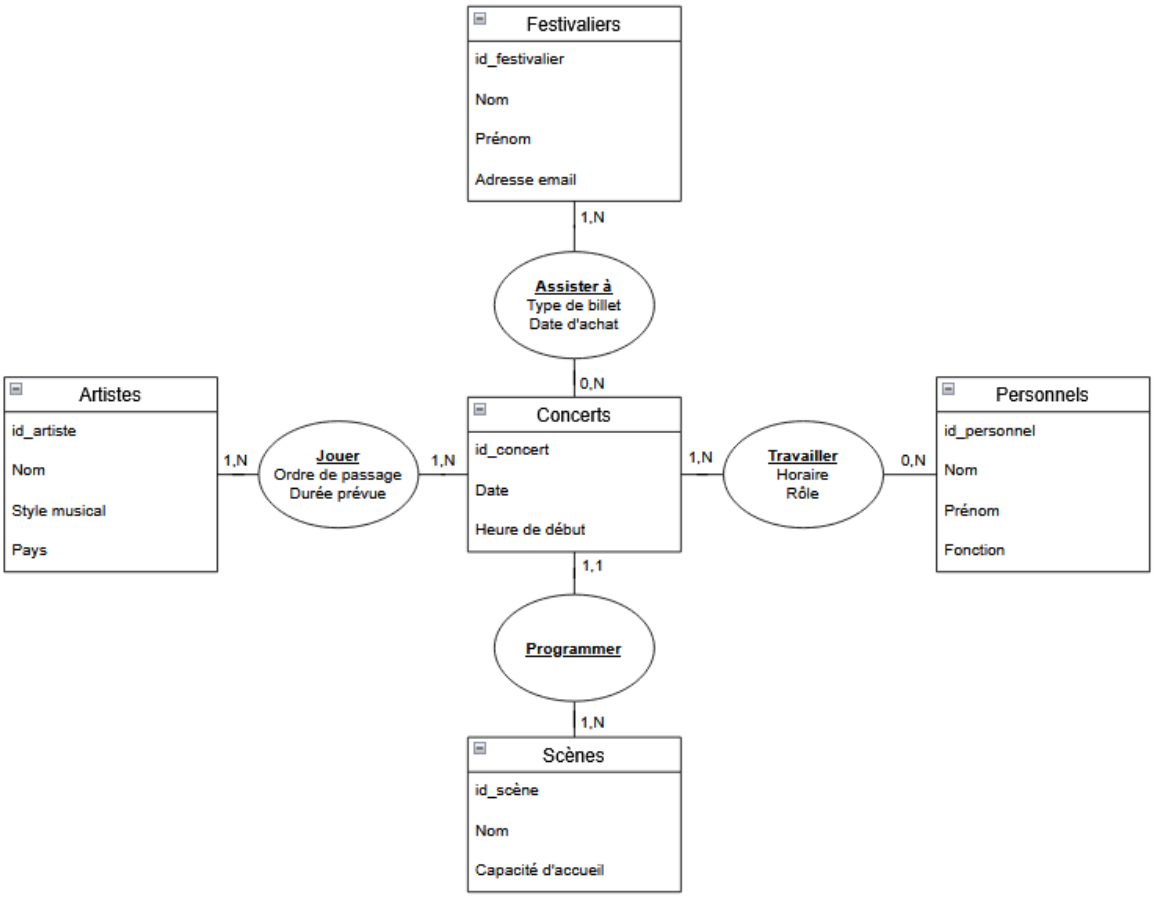
Programmer : Un concert est programmé sur une et une seule scène (cardinalité (1,1) côté Concerts) et une scène peut accueillir un ou plusieurs concerts (cardinalité (0,N) côté Scènes).

Jouer : Un artiste peut jouer dans un ou plusieurs concerts (1,N) et un concert accueille un ou plusieurs artistes (1,N)

Assister à : Un festivalier peut assister à un ou plusieurs concerts (1,N) et un concert peut accueillir zéro, un ou plusieurs festivaliers (0,N).

Travailler sur : Un membre du personnel peut travailler sur zéro, un ou plusieurs concerts (0,N) et un concert mobilise au moins une personne, mais souvent plusieurs (1,N).

Schéma MCD



Modèle logique de données

Conversion des associations

Programmer (Concerts ↔ Scènes)

- Règle métier : *Un concert est toujours associé à une seule scène, mais une scène peut accueillir plusieurs concerts.*
- Conséquence : la cardinalité (1,1) côté Concerts et (0,N) côté Scènes signifie que la clé primaire de **Scènes** peut être insérée comme clé étrangère dans **Concerts**.
- Transformation : ajout de la clé étrangère `id_scene` dans `concerts`.

Jouer (Artistes ↔ Concerts)

- Règle métier : *Un artiste peut jouer dans plusieurs concerts, et un concert accueille plusieurs artistes.*
- Conséquence : relation N,N.
- Transformation : création d'une table `jouer(#id_artiste, #id_concert, ordre_passage, duree_prevue)` avec clé primaire composée.

Assister à (Festivaliers ↔ Concerts)

- Règle métier : *Un festivalier peut assister à plusieurs concerts, et un concert peut accueillir plusieurs festivaliers.*
- Conséquence : relation N,N.
- Transformation : création d'une table `assister(#id_festivalier, #id_concert, type_billet, date_achat)` avec clé primaire composée.

Travailler sur (Personnel ↔ Concerts)

- Règle métier : *Un membre du personnel peut travailler sur plusieurs concerts, et chaque concert mobilise plusieurs personnels.*
- Conséquence : relation N,N.
- Transformation : création d'une table `travailler(#id_concert, #id_personnel, role, horaire)` avec clé primaire composée.

MLD textuel

artistes(id_artiste, nom, style_musical, pays)

scenes(id_scene, nom, capacite_accueil)

concerts(id_concert, date_concert, heure_debut, #id_scene)

festivaliers(id_festivalier, nom, prenom, adresse_email)

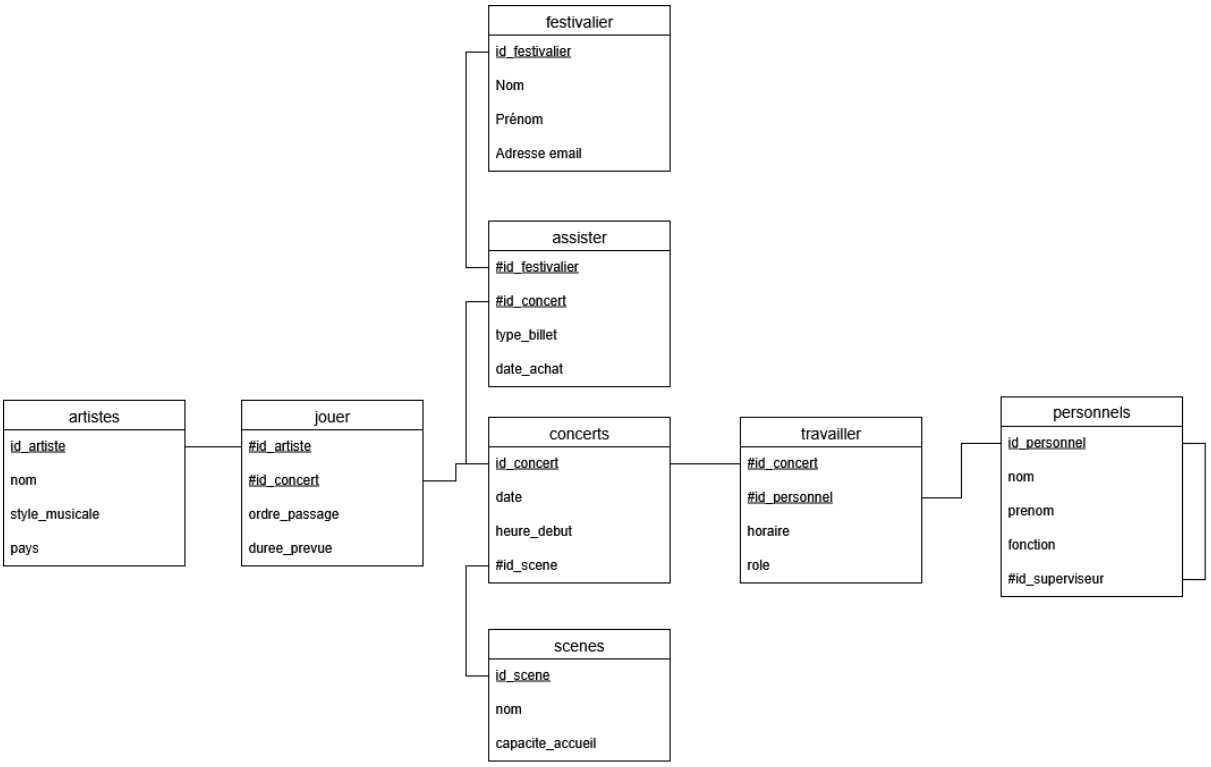
personnels(id_personnel, nom, prenom, fonction, #id_superviseur)

jouer(#id_artiste, #id_concert, ordre_passage, duree_prevue)

assister(#id_festivalier, #id_concert, type_billet, date_achat)

travailler(#id_concert, #id_personnel, horaire, role)

Schéma MLD


festivalier
id_festivalier
Nom
Prénom
Adresse email
assister
#id_festivalier
#id_concert
type_billet
date_achat
artistes
id_artiste
nom
style_musicale
pays
jouer
#id_artiste
#id_concert
ordre_passage
duree_prevue
concerts
id_concert
date
heure_debut
#id_scene
travailler
#id_concert
#id_personnel
horaire
role
personnels
id_personnel
nom
prenom
fonction
#id_superviseur
scenes
id_scene
nom
capacite_accueil

Modèle Physique de Données

Tables finales, choix des types et contraintes

Table Artistes

- `id_artiste` : **SERIAL** – **PRIMARY KEY** – identifiant unique généré automatiquement.
- `nom` : **VARCHAR(100)** – **NOT NULL** – chaque artiste doit avoir un nom, longueur suffisante pour des noms composés.
- `style_musical` : **VARCHAR(50)** – facultatif – permet de stocker le genre musical principal, valeur courte.
- `pays` : **VARCHAR(50)** – facultatif – indique le pays d'origine de l'artiste.

Table Scènes

- `id_scene` : **SERIAL** – **PRIMARY KEY** – identifiant unique généré automatiquement.
- `nom` : **VARCHAR(100)** – **NOT NULL** – chaque scène doit avoir un nom unique, longueur suffisante.
- `capacite_accueil` : **INTEGER** – **CHECK (capacite_accueil > 0)** – garantit que la capacité est un nombre positif.

Table Concerts

- `id_concert` : **SERIAL** – **PRIMARY KEY** – identifiant unique généré automatiquement.
- `date_concert` : **DATE** – **NOT NULL** – un concert doit avoir une date.
- `heure_debut` : **TIME** – **NOT NULL** – un concert doit avoir une heure de début précise.
- `id_scene` : **INT** – **NOT NULL, FOREIGN KEY** vers `scenes(id_scene)` – chaque concert est associé à une scène existante.

Table Festivaliers

- `id_festivalier` : **SERIAL** – **PRIMARY KEY** – identifiant unique généré automatiquement.
- `nom` : **VARCHAR(100)** – **NOT NULL** – le nom du festivalier doit être connu.
- `prenom` : **VARCHAR(100)** – **NOT NULL** – le prénom du festivalier doit être connu.
- `adresse_email` : **VARCHAR(150)** – **NOT NULL, UNIQUE** – identifie de manière fiable chaque festivalier (pas de doublons).

Table Personnels

- `id_personnel` : **SERIAL** – **PRIMARY KEY** – identifiant unique généré automatiquement.

- `nom` : **VARCHAR(100)** – **NOT NULL** – obligatoire pour identifier un membre du personnel.
- `prenom` : **VARCHAR(100)** – **NOT NULL** – obligatoire pour identifier un membre du personnel.
- `fonction` : **VARCHAR(50)** – facultatif – rôle général du membre (sécurité, technique, etc.).
- `id_superviseur` : **INT** – **FOREIGN KEY** vers `personnels(id_personnel)` – permet de représenter une hiérarchie interne (relation réflexive).

Table Participations (anciennement *Jouer*)

- `id_artiste` : **INT** – **FOREIGN KEY** vers `artistes(id_artiste)` – identifie l'artiste.
- `id_concert` : **INT** – **FOREIGN KEY** vers `concerts(id_concert)` – identifie le concert.
- `ordre_passage` : **INT** – facultatif – indique l'ordre de passage de l'artiste dans le concert.
- `duree_prevue` : **INTERVAL** – facultatif – durée prévue de la performance.
- **Clé primaire composée** : (`id_artiste`, `id_concert`) – garantit qu'un artiste ne peut être inscrit qu'une fois par concert.

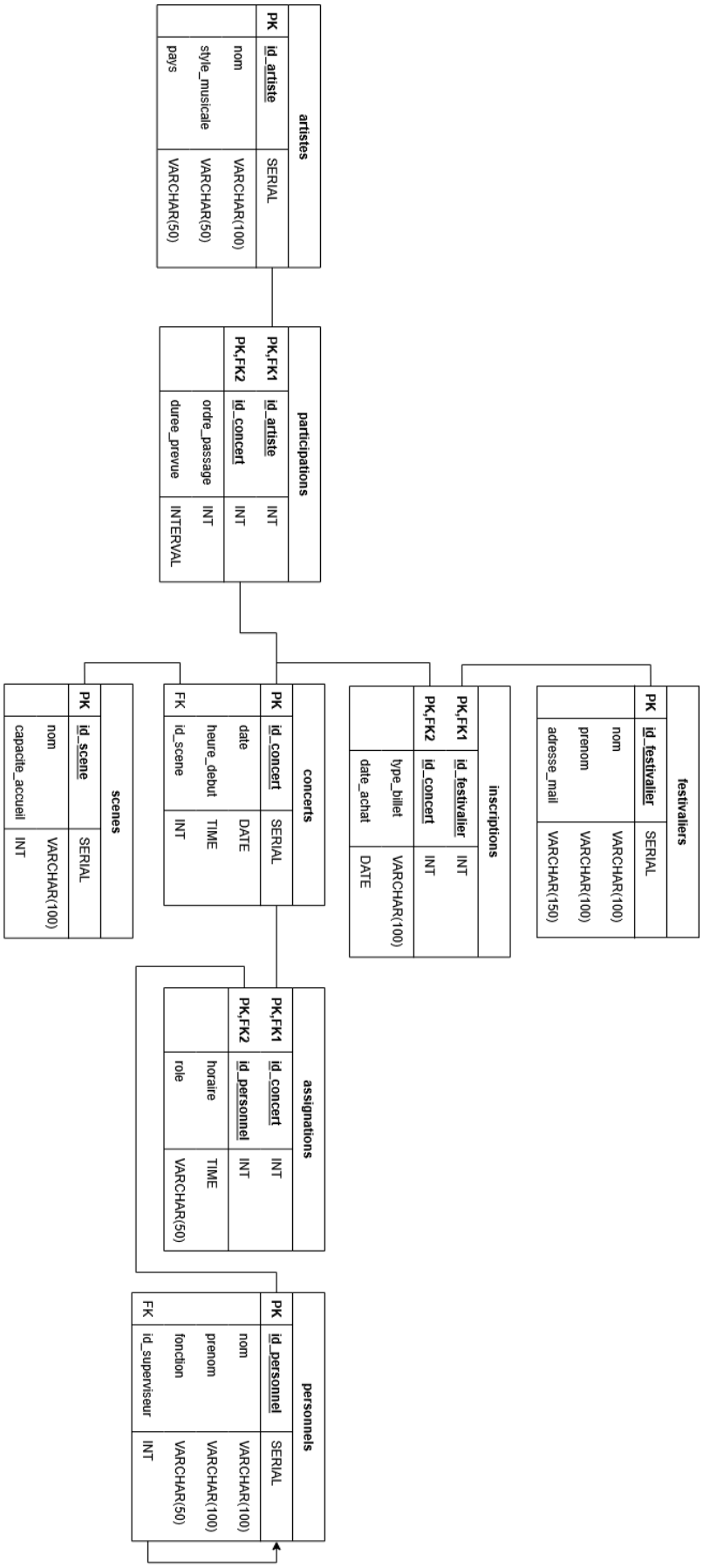
Table Inscriptions (anciennement *Assister*)

- `id_festivalier` : **INT** – **FOREIGN KEY** vers `festivaliers(id_festivalier)` – identifie le spectateur.
- `id_concert` : **INT** – **FOREIGN KEY** vers `concerts(id_concert)` – identifie le concert.
- `type_billet` : **VARCHAR(50)** – facultatif – type de billet (ex. journée, soirée, VIP).
- `date_achat` : **DATE** – facultatif – date d'achat du billet.
- **Clé primaire composée** : (`id_festivalier`, `id_concert`) – empêche les doublons (un festivalier ne peut s'inscrire qu'une fois à un concert donné).

Table Assignations (anciennement *Travailler*)

- `id_concert` : **INT** – **FOREIGN KEY** vers `concerts(id_concert)` – identifie le concert.
- `id_personnel` : **INT** – **FOREIGN KEY** vers `personnels(id_personnel)` – identifie le membre du personnel.
- `horaire` : **TIME** – facultatif – heure précise de la prise de poste.
- `role` : **VARCHAR(50)** – facultatif – rôle précis pour ce concert (sécurité, accueil, technique).
- **Clé primaire composée** : (`id_concert`, `id_personnel`) – garantit qu'un membre du personnel n'a qu'une assignation par concert.

Schéma MPD


artistes
PK
id_artiste
SERIAL
nom
VARCHAR(100)
style_musicale
VARCHAR(50)
pays
VARCHAR(50)
participations
PK,FK1
id_artiste
INT
PK,FK2
id_concert
INT
ordre_passage
INT
duree_prevue
INTERVAL
festivaliers
PK
id_festivalier
SERIAL
nom
VARCHAR(100)
prenom
VARCHAR(100)
adresse_mail
VARCHAR(150)
inscriptions
PK,FK1
id_festivalier
INT
PK,FK2
id_concert
INT
type_billet
VARCHAR(100)
date_achat
DATE
concerts
PK
id_concert
SERIAL
date
DATE
heure_debut
TIME
FK
id_scene
INT
scenes
PK
id_scene
SERIAL
nom
VARCHAR(100)
capacite_accueil
INT
assignations
PK,FK1
id_concert
INT
PK,FK2
id_personnel
INT
horaire
TIME
role
VARCHAR(50)
personnels
PK
id_personnel
SERIAL
nom
VARCHAR(100)
prenom
VARCHAR(100)
fonction
VARCHAR(50)
FK
id_superviseur
INT