diginamic FORMATION
« Votre passeport pour l'emploi numérique »

Requêtage Many-to-Many avec SQLModel

Une relation **many-to-many** permet à :

- un héros d'appartenir à **plusieurs équipes**,
- une équipe de contenir **plusieurs héros**.

Contrairement aux relations One-to-Many, cela nécessite une **table de liaison intermédiaire**, également appelée *link table*.

Sommaire

- Requêtage Many-to-Many avec SQLModel
  - Sommaire
  - Définir une relation many-to-many
    - Déclaration des modèles
    - 🔍 Détails importants :
  - Création avec plusieurs liens
  - Ajout d'un lien Many-to-Many
  - Suppression d'un lien Many-to-Many

Définir une relation many-to-many

Déclaration des modèles

> Table de liaison : `HeroTeamLink`

```
class HeroTeamLink(SQLModel, table=True):
    team_id: int | None = Field(default=None, foreign_key="team.id",
primary_key=True)
    hero_id: int | None = Field(default=None, foreign_key="hero.id",
primary_key=True)
```

> Table de données : `Team`

```
class Team(SQLModel, table=True):
    id: int | None = Field(default=None, primary_key=True)
    name: str = Field(index=True)
    headquarters: str

    heroes: list["Hero"] = Relationship(back_populates="teams",
link_model=HeroTeamLink)
```

> Table de données : `Hero`

```
class Hero(SQLModel, table=True):
    id: int | None = Field(default=None, primary_key=True)
    name: str = Field(index=True)
    secret_name: str
    age: int | None = Field(default=None, index=True)

    teams: list[Team] = Relationship(back_populates="heroes",
link_model=HeroTeamLink)
```

🔍 Détails importants :

- `HeroTeamLink` contient **deux clés étrangères**, définies comme **clé primaire combinée**.
- Les champs `heroes` et `teams` permettent une navigation **dans les deux sens**.
- Le paramètre `link_model=HeroTeamLink` indique le modèle utilisé pour faire le lien.

Création avec plusieurs liens

```
team_preventers = Team(name="Preventers", headquarters="Sharp Tower")
team_z_force = Team(name="Z-Force", headquarters="Sister Margaret's Bar")

hero_deadpond = Hero(name="Deadpond", secret_name="Dive Wilson", teams=
[team_z_force, team_preventers])
hero_rusty_man = Hero(name="Rusty-Man", secret_name="Tommy Sharp", age=48, teams=
[team_preventers])
hero_spider_boy = Hero(name="Spider-Boy", secret_name="Pedro Parqueador", teams=
[team_preventers])

session.add(hero_deadpond)
session.add(hero_rusty_man)
session.add(hero_spider_boy)
session.commit()
```

> 💡 Ici, chaque héros peut être ajouté à plusieurs équipes **en une seule instruction**.

Ajout d'un lien Many-to-Many

```
hero_spider_boy = session.exec(select(Hero).where(Hero.name == "Spider-
Boy")).one()
team_z_force = session.exec(select(Team).where(Team.name == "Z-Force")).one()

team_z_force.heroes.append(hero_spider_boy)
session.add(team_z_force)
session.commit()
```

> ⚠ Ce n'est **pas une mise à jour d'objet**, mais bien un **ajout de lien** dans la table de liaison.

Suppression d'un lien Many-to-Many

```
hero_spider_boy.teams.remove(team_z_force)
session.add(team_z_force)
session.commit()
```

> Contrairement au One-to-Many, on **ne supprime pas l'objet**. On **retire la relation** entre les deux entités. On peut aussi utiliser `session.delete(team_z_force)` pour supprimer l'équipe, mais cela entraînerait la suppression de tous les liens associés dans `HeroTeamLink`.