

Modèle Conceptuel de Données (MCD) et Modèle Logique de Données (MLD) - TaskFlow

1. Modèle Conceptuel de Données (MCD)

Entités principales

USER

- id
- email
- roles
- password
- first_name
- last_name
- created_at
- reset_token
- reset_token_expires_at
- is_verified

PROJECT

- id
- title
- description
- created_at
- updated_at

TASK

- id
- title
- description
- status
- priority
- due_date
- created_at
- updated_at

- completed_at

COLLABORATION_REQUEST

- id
- status
- message
- response
- created_at
- updated_at
- responded_at

NOTIFICATION

- id
- type
- title
- message
- status
- created_at
- read_at
- action_url
- data

Associations

1. OWNS (USER → PROJECT)

- Cardinalité : 1,N - 1,1
- Un utilisateur peut posséder plusieurs projets
- Un projet appartient à un seul utilisateur (propriétaire)

2. COLLABORATES (USER ↔ PROJECT)

- Cardinalité : 0,N - 0,N
- Un utilisateur peut collaborer sur plusieurs projets
- Un projet peut avoir plusieurs collaborateurs

3. CONTAINS (PROJECT → TASK)

- Cardinalité : 1,1 - 0,N
- Un projet contient zéro ou plusieurs tâches

- Une tâche appartient à un seul projet

4. **IS_ASSIGNED** (USER → TASK)

- Cardinalité : 0,1 - 0,N
- Un utilisateur peut être assigné à plusieurs tâches
- Une tâche peut être assignée à un seul utilisateur (ou aucun)

5. **SENDS** (USER → COLLABORATION_REQUEST)

- Cardinalité : 1,1 - 0,N
- Un utilisateur peut envoyer plusieurs demandes
- Une demande est envoyée par un seul utilisateur

6. **RECEIVES** (USER → COLLABORATION_REQUEST)

- Cardinalité : 1,1 - 0,N
- Un utilisateur peut recevoir plusieurs demandes
- Une demande est reçue par un seul utilisateur

7. **CONCERNS** (PROJECT → COLLABORATION_REQUEST)

- Cardinalité : 1,1 - 0,N
- Un projet peut faire l'objet de plusieurs demandes
- Une demande concerne un seul projet

8. **NOTIFIES** (USER → NOTIFICATION)

- Cardinalité : 1,1 - 0,N (destinataire)
- Un utilisateur peut recevoir plusieurs notifications
- Une notification est destinée à un seul utilisateur

9. **TRIGGERS** (USER → NOTIFICATION)

- Cardinalité : 0,1 - 0,N (expéditeur)
- Un utilisateur peut déclencher plusieurs notifications
- Une notification peut être déclenchée par un utilisateur (ou le système)

10. **RELATES_TO_PROJECT** (PROJECT → NOTIFICATION)

- Cardinalité : 0,1 - 0,N
- Un projet peut générer plusieurs notifications
- Une notification peut concerner un projet

11. **RELATES_TO_TASK** (TASK → NOTIFICATION)

- Cardinalité : 0,1 - 0,N
- Une tâche peut générer plusieurs notifications
- Une notification peut concerner une tâche

2. Modèle Logique de Données (MLD)

Tables et relations

Table USER

```
sql

USER (
  id INTEGER PRIMARY KEY AUTO_INCREMENT,
  email VARCHAR(180) UNIQUE NOT NULL,
  roles JSON NOT NULL,
  password VARCHAR(255) NOT NULL,
  first_name VARCHAR(100) NOT NULL,
  last_name VARCHAR(100) NOT NULL,
  created_at DATETIME NOT NULL,
  reset_token VARCHAR(255),
  reset_token_expires_at DATETIME,
  is_verified BOOLEAN NOT NULL DEFAULT FALSE
)
```

Table PROJECT

```
sql

PROJECT (
  id INTEGER PRIMARY KEY AUTO_INCREMENT,
  owner_id INTEGER NOT NULL,
  title VARCHAR(255) NOT NULL,
  description LONGTEXT,
  created_at DATETIME NOT NULL,
  updated_at DATETIME,
  FOREIGN KEY (owner_id) REFERENCES USER(id)
)
```

Table PROJECT_USER (Table de liaison pour la collaboration)

```
sql
```

```
PROJECT_USER (  
  project_id INTEGER NOT NULL,  
  user_id INTEGER NOT NULL,  
  PRIMARY KEY (project_id, user_id),  
  FOREIGN KEY (project_id) REFERENCES PROJECT(id) ON DELETE CASCADE,  
  FOREIGN KEY (user_id) REFERENCES USER(id) ON DELETE CASCADE  
)
```

Table TASK

```
sql  
  
TASK (  
  id INTEGER PRIMARY KEY AUTO_INCREMENT,  
  project_id INTEGER NOT NULL,  
  assignee_id INTEGER,  
  title VARCHAR(255) NOT NULL,  
  description LONGTEXT,  
  status VARCHAR(20) NOT NULL,  
  priority VARCHAR(20) NOT NULL,  
  due_date DATETIME,  
  created_at DATETIME NOT NULL,  
  updated_at DATETIME,  
  completed_at DATETIME,  
  FOREIGN KEY (project_id) REFERENCES PROJECT(id),  
  FOREIGN KEY (assignee_id) REFERENCES USER(id) ON DELETE SET NULL  
)
```

Table COLLABORATION_REQUEST

```
sql
```

```
COLLABORATION_REQUEST (
  id INTEGER PRIMARY KEY AUTO_INCREMENT,
  project_id INTEGER NOT NULL,
  sender_id INTEGER NOT NULL,
  invited_user_id INTEGER NOT NULL,
  status VARCHAR(20) NOT NULL,
  message LONGTEXT,
  response LONGTEXT,
  created_at DATETIME NOT NULL,
  updated_at DATETIME,
  responded_at DATETIME,
  FOREIGN KEY (project_id) REFERENCES PROJECT(id) ON DELETE CASCADE,
  FOREIGN KEY (sender_id) REFERENCES USER(id),
  FOREIGN KEY (invited_user_id) REFERENCES USER(id),
  UNIQUE KEY unique_project_invitation (project_id, invited_user_id)
)
```

Table NOTIFICATION

```
sql

NOTIFICATION (
  id INTEGER PRIMARY KEY AUTO_INCREMENT,
  recipient_id INTEGER NOT NULL,
  sender_id INTEGER,
  project_id INTEGER,
  task_id INTEGER,
  type VARCHAR(50) NOT NULL,
  title VARCHAR(255) NOT NULL,
  message LONGTEXT NOT NULL,
  status VARCHAR(20) NOT NULL,
  created_at DATETIME NOT NULL,
  read_at DATETIME,
  action_url VARCHAR(255),
  data JSON,
  FOREIGN KEY (recipient_id) REFERENCES USER(id),
  FOREIGN KEY (sender_id) REFERENCES USER(id),
  FOREIGN KEY (project_id) REFERENCES PROJECT(id),
  FOREIGN KEY (task_id) REFERENCES TASK(id)
)
```

Contraintes d'intégrité

Contraintes de domaine

- USER.email : Format email valide, unique

- `USER.is_verified` : Boolean (0 ou 1)
- `TASK.status` : Valeurs autorisées ('todo', 'in_progress', 'completed')
- `TASK.priority` : Valeurs autorisées ('low', 'medium', 'high')
- `COLLABORATION_REQUEST.status` : Valeurs autorisées ('pending', 'accepted', 'refused')
- `NOTIFICATION.status` : Valeurs autorisées ('unread', 'read')

Contraintes de référence

- Toutes les clés étrangères doivent référencer des enregistrements existants
- `PROJECT.owner_id` → `USER.id` (NOT NULL)
- `TASK.project_id` → `PROJECT.id` (NOT NULL)
- `TASK.assignee_id` → `USER.id` (NULL autorisé - tâche non assignée)
- Suppression en cascade pour les relations projet-collaborateur

Contraintes métier

- Un utilisateur ne peut pas s'inviter lui-même (`COLLABORATION_REQUEST.sender_id ≠ invited_user_id`)
- Une seule demande de collaboration par projet et utilisateur invité
- Les dates de mise à jour doivent être postérieures aux dates de création
- `TASK.completed_at` n'est renseigné que si `status = 'completed'`

Index recommandés

```
sql
-- Index pour les requêtes fréquentes
CREATE INDEX idx_project_owner ON PROJECT(owner_id);
CREATE INDEX idx_task_project ON TASK(project_id);
CREATE INDEX idx_task_assignee ON TASK(assignee_id);
CREATE INDEX idx_task_status ON TASK(status);
CREATE INDEX idx_task_due_date ON TASK(due_date);
CREATE INDEX idx_notification_recipient ON NOTIFICATION(recipient_id);
CREATE INDEX idx_notification_status ON NOTIFICATION(status);
CREATE INDEX idx_collaboration_status ON COLLABORATION_REQUEST(status);
```

Règles de gestion identifiées

1. **Propriété des projets** : Seul le propriétaire d'un projet peut le modifier ou le supprimer
2. **Gestion des tâches** : Seul le propriétaire du projet peut créer/modifier/supprimer des tâches
3. **Assignment** : Les tâches ne peuvent être assignées qu'aux collaborateurs du projet
4. **Collaboration** : Un utilisateur ne peut collaborer que sur invitation acceptée

5. **Notifications** : Générées automatiquement pour les événements importants

6. **Unicité des invitations** : Une seule invitation active par projet et utilisateur