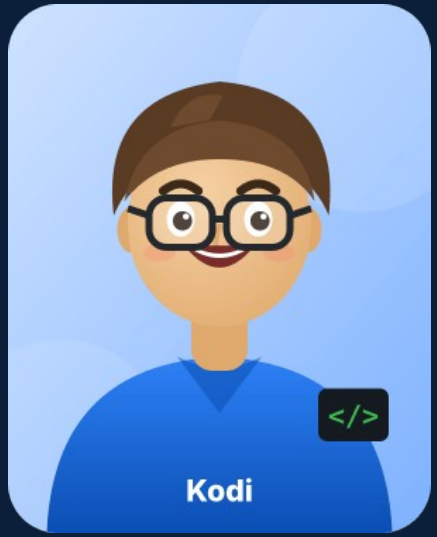
HEF-070 · MISSION 01 / 10

Le modèle relationnel

Des tables, des lignes, des liens entre elles : une idée vieille de 50 ans qui fait encore tourner le monde.



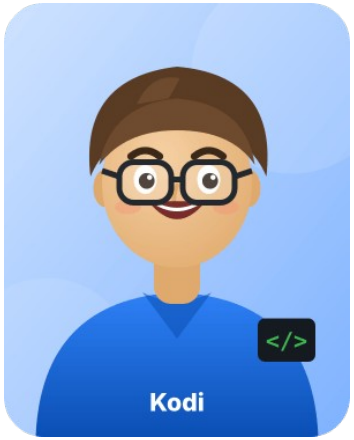


Badge : Archiviste des données

70 min · intermédiaire

L'histoire

HashBox stocke ses membres, ses cours et leurs inscriptions dans trois fichiers Excel qui s'envoient par e-mail. Résultat : Ada a deux fiches avec deux adresses différentes, Bob est inscrit à un cours qui n'existe plus, et personne ne sait quelle version est la bonne.



> *« Il te faut une base de données, dit Sam. Un endroit unique, avec des règles, que plusieurs personnes utilisent en même temps sans se marcher dessus. »*

Tu découvres le modèle relationnel et tu crées ta première base avec SQLite.

Ta mission

À la fin, je serai capable de…

1. définir table, ligne, colonne, clé primaire et clé étrangère
2. expliquer ce qu'est une relation entre tables (1-n, n-n)
3. créer une base SQLite et trois tables avec Python
4. insérer et lire des données
5. expliquer pourquoi une base vaut mieux que des fichiers séparés

Défi d'échauffement

Défi d'échauffement

Tu tiens la liste des élèves d'une école dans un tableur : nom, classe, professeur principal, adresse du professeur. Si le professeur change d'adresse, combien de lignes dois-tu modifier ? Que se passe-t-il si tu en oublies une ? Comment organiserais-tu les informations pour ne les écrire qu'**une seule fois** ?

Le concept

Qu'est-ce qu'une base de données ?

Une **base de données** est un ensemble organisé de données, géré par un **SGBD** (système de gestion de base de données) qui offre :

- le **stockage durable** ;
- la **recherche rapide** ;
- l'**intégrité** (des règles qui empêchent les données absurdes) ;
- la **concurrence** (plusieurs utilisateurs en même temps) ;
- la **fiabilité** (résiste aux pannes).

Le modèle relationnel

On range les données dans des **tables** (ou **relations**).

Le concept

Le modèle relationnel (suite)

id	nom	ville
1	Ada	Lyon
2	Bob	Lille

- une **colonne** (attribut) : un type d'information avec un **type** (entier, texte, date…) ;
- une **ligne** (enregistrement, tuple) : un élément décrit ;
- un **schéma** : la structure des tables (noms et types des colonnes).

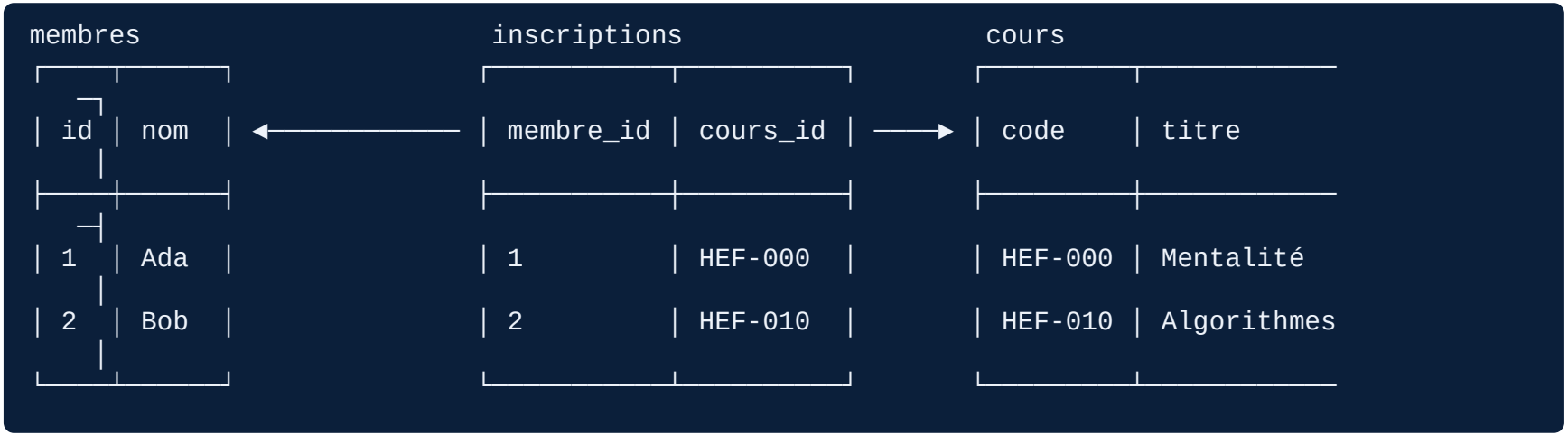
La clé primaire

Une **clé primaire** (*primary key*) identifie chaque ligne de façon **unique** et **stable**. Souvent un entier auto-incrémenté (`id`).

Le concept

La clé étrangère

Une **clé étrangère** (*foreign key*) est une colonne qui référence la clé primaire d'une autre table. C'est elle qui crée le **lien**.



Le concept

La clé étrangère (suite)

```
└┘
```

Les types de relations

Relation	Exemple	Réalisation
1 – n	un membre a plusieurs commentaires	clé étrangère dans la table du « n »
n – n	un membre suit plusieurs cours, un cours a plusieurs membres	**table de liaison** (inscriptions)
1 – 1	un membre a un profil	clé étrangère unique

SQL

SQL (*Structured Query Language*) est le langage standard pour décrire et interroger les données. Il est **déclaratif** : on dit *quoi* obtenir, pas *comment*.

Le concept

SQLite en Python

SQLite est une base complète intégrée dans un simple fichier, fournie avec Python (module `sqlite3`). Parfait pour apprendre : mêmes concepts que PostgreSQL ou MySQL.

```
import sqlite3

bd = sqlite3.connect(":memory:")            # base en mémoire (disparaît à la ferm
   eture)
bd.execute("CREATE TABLE membres (id INTEGER PRIMARY KEY, nom TEXT, ville TEXT)")
bd.execute("INSERT INTO membres (nom, ville) VALUES ('Ada', 'Lyon')")
bd.execute("INSERT INTO membres (nom, ville) VALUES ('Bob', 'Lille')")
for ligne in bd.execute("SELECT id, nom, ville FROM membres"):
    print(ligne)       # (1, 'Ada', 'Lyon') puis (2, 'Bob', 'Lille')
```

Le concept

SQLite en Python (suite)

🧠 ***Astuce** : une bonne table décrit **un seul type de chose**. Les liens entre les choses passent par des clés étrangères, pas par des copies.*

Exemple pas à pas

Problème : créer la base de HashBox (membres, cours, inscriptions) et répondre à « qui est inscrit à quoi ? ».

```
import sqlite3

bd = sqlite3.connect(":memory:")
bd.execute("PRAGMA foreign_keys = ON")      # SQLite : à activer pour vérifier les
    clés étrangères

bd.executescript('''
CREATE TABLE membres (
    id    INTEGER PRIMARY KEY,
    nom   TEXT NOT NULL,
    ville TEXT
);
CREATE TABLE cours (
    code  TEXT PRIMARY KEY,
    titre TEXT NOT NULL
```

Exemple pas à pas (suite)

```
);
CREATE TABLE inscriptions (
    membre_id INTEGER NOT NULL REFERENCES membres(id),
    cours_code TEXT NOT NULL REFERENCES cours(code),
    PRIMARY KEY (membre_id, cours_code)
);
''')

bd.executemany("INSERT INTO membres (nom, ville) VALUES (?, ?)", [("Ada", "Lyon"),
    ("Bob", "Lille"), ("Cléo", "Lyon")])
bd.executemany("INSERT INTO cours VALUES (?, ?)", [("HEF-000", "Mentalité"), ("HEF
   -010", "Algorithmes")])
bd.executemany("INSERT INTO inscriptions VALUES (?, ?)", [(1, "HEF-000"), (1, "HEF
   -010"), (2, "HEF-000")])

requete = '''
SELECT membres.nom, cours.titre
FROM inscriptions
JOIN membres ON membres.id = inscriptions.membre_id
```

Exemple pas à pas

Exemple pas à pas (suite)

```
JOIN cours   ON cours.code = inscriptions.cours_code
ORDER BY membres.nom, cours.titre
'''
for nom, titre in bd.execute(requete):
    print(nom, "→", titre)
# Ada → Algorithmes / Ada → Mentalité / Bob → Mentalité
```

Ce que la base garantit :

```
try:
    bd.execute("INSERT INTO inscriptions VALUES (99, 'HEF-000')")     # membre ine
   xistant
except sqlite3.IntegrityError as e:
    print("refusé :", e)
```

La base **refuse** une inscription pour un membre qui n'existe pas. Dans le tableur, personne ne l'aurait empêchée.

Exemple pas à pas (suite)

Pourquoi trois tables ? Un membre est écrit une seule fois, un cours une seule fois. Les inscriptions ne stockent que des **identifiants**. Changer la ville d'Ada se fait en **une seule ligne**.

À toi, avec aide

À toi, avec aide

Une bibliothèque a des livres, des lecteurs et des emprunts.

1. Dessine (en texte) les tables : colonnes, clés primaires, clés étrangères.
2. Crée-les avec SQLite.
3. Insère 3 livres, 2 lecteurs, 3 emprunts.
4. Écris la requête « qui a emprunté quoi ? ».

À toi, avec aide · réponse

Indices progressifs

Indice 1

Un emprunt relie un lecteur et un livre : table de liaison, avec une date.

Indice 2

La clé primaire d'un emprunt peut être un `id` propre (un lecteur peut emprunter plusieurs fois le même livre).

Indice 3

La requête a deux jointures, comme dans l'exemple.

À toi, avec aide · réponse

Réponse

```
import sqlite3

bd = sqlite3.connect(":memory:")
bd.execute("PRAGMA foreign_keys = ON")
bd.executescript('''
CREATE TABLE livres (id INTEGER PRIMARY KEY, titre TEXT NOT NULL, auteur TEXT);
CREATE TABLE lecteurs (id INTEGER PRIMARY KEY, nom TEXT NOT NULL);
CREATE TABLE emprunts (
    id INTEGER PRIMARY KEY,
    livre_id INTEGER NOT NULL REFERENCES livres(id),
    lecteur_id INTEGER NOT NULL REFERENCES lecteurs(id),
    date_emprunt TEXT NOT NULL
);
''')
bd.executemany("INSERT INTO livres (titre, auteur) VALUES (?, ?)", [("Dune", "Herb
   ert"), ("Fondation", "Asimov"), ("Ubik", "Dick")])
bd.executemany("INSERT INTO lecteurs (nom) VALUES (?)", [("Léa",), ("Omar",)])
bd.executemany("INSERT INTO emprunts (livre_id, lecteur_id, date_emprunt) VALUES (
   ?, ?, ?)",
```

À toi, avec aide · réponse

Réponse (suite)

```
              [(1, 1, "2025-01-05"), (2, 1, "2025-01-10"), (3, 2, "2025-01-12")])
for ligne in bd.execute('''SELECT lecteurs.nom, livres.titre, emprunts.date_emprun
   t
                           FROM emprunts JOIN lecteurs ON lecteurs.id = emprunts.l
   ecteur_id
                                         JOIN livres ON livres.id = emprunts.livre
   _id
                           ORDER BY emprunts.date_emprunt'''):
    print(ligne)
```

À toi, en solo

Concevoir le schéma d'une plateforme de cours (sans écrire encore de code) :

- les membres s'inscrivent à des cours ;
- chaque cours est composé de missions ordonnées ;
- chaque membre termine des missions et gagne des points (XP) ;
- chaque membre peut avoir un badge par mission.

Dessine les tables, les clés, les relations (1-n, n-n) et justifie chaque table de liaison. Puis crée le schéma avec SQLite et teste-le avec des données d'exemple : refuse une mission terminée pour un membre inexistant.

À toi, en solo · réponse

Pistes de correction

```
import sqlite3

bd = sqlite3.connect(":memory:")
bd.execute("PRAGMA foreign_keys = ON")
bd.executescript('''
CREATE TABLE membres (id INTEGER PRIMARY KEY, nom TEXT NOT NULL);
CREATE TABLE cours (code TEXT PRIMARY KEY, titre TEXT NOT NULL);
CREATE TABLE missions (
    id INTEGER PRIMARY KEY,
    cours_code TEXT NOT NULL REFERENCES cours(code),
    numero INTEGER NOT NULL,
    titre TEXT NOT NULL,
    badge TEXT,
    UNIQUE (cours_code, numero)
);
CREATE TABLE inscriptions (
    membre_id INTEGER NOT NULL REFERENCES membres(id),
    cours_code TEXT NOT NULL REFERENCES cours(code),
    PRIMARY KEY (membre_id, cours_code)
```

À toi, en solo · réponse

Pistes de correction (suite)

```
);
CREATE TABLE missions_terminees (
    membre_id INTEGER NOT NULL REFERENCES membres(id),
    mission_id INTEGER NOT NULL REFERENCES missions(id),
    xp INTEGER NOT NULL,
    terminee_le TEXT NOT NULL,
    PRIMARY KEY (membre_id, mission_id)
);
''')
bd.execute("INSERT INTO membres (nom) VALUES ('Ada')")
bd.execute("INSERT INTO cours VALUES ('HEF-000', 'Mentalité')")
bd.execute("INSERT INTO missions (cours_code, numero, titre, badge) VALUES ('HEF-0
   00', 1, 'Cadrer', 'Détective')")
bd.execute("INSERT INTO missions_terminees VALUES (1, 1, 100, '2025-02-01')")
try:
    bd.execute("INSERT INTO missions_terminees VALUES (42, 1, 100, '2025-02-01')")
except sqlite3.IntegrityError as e:
    print("refusé :", e)
```

À toi, en solo · réponse

Pistes de correction (suite)

Tables de liaison : `inscriptions` (membres n-n cours) et `missions_terminees` (membres n-n missions, avec des attributs propres : xp et date). Les badges sont un attribut de la mission (1-n : un badge par mission).

Pièges classiques

Piège	Pourquoi c'est un problème	Comment l'éviter
Tout dans une seule table	Duplications et incohérences	Une table = une sorte de chose
Oublier d'activer les clés étrangères (SQLite)	Aucune vérification des liens	`PRAGMA foreign_keys = ON`
Clé primaire instable	Un nom qui change casse les liens	Identifiant propre qui ne change jamais
Relation n-n sans table de liaison	Colonnes de listes impossibles à interroger	Table de liaison avec deux clés étrangères

Mini-boss

Mini-boss

Schéma d'une boutique en ligne. Besoins : clients, produits, commandes, lignes de commande (un produit peut apparaître dans plusieurs commandes avec une quantité), adresses de livraison (un client peut en avoir plusieurs).

1. Dessine le schéma avec clés primaires/étrangères et cardinalités.
2. Crée-le avec SQLite.
3. Insère au moins 3 clients, 5 produits, 4 commandes.
4. Écris les requêtes : (a) toutes les lignes de la commande 1 avec le nom du produit ; (b) le montant total de chaque commande (prix × quantité) ; (c) les clients qui n'ont jamais commandé.
5. Tente 2 insertions invalides et montre que la base les refuse.

Mini-boss · réponse

Pistes de correction

```
import sqlite3

bd = sqlite3.connect(":memory:")
bd.execute("PRAGMA foreign_keys = ON")
bd.executescript('''
CREATE TABLE clients (id INTEGER PRIMARY KEY, nom TEXT NOT NULL);
CREATE TABLE adresses (id INTEGER PRIMARY KEY, client_id INTEGER NOT NULL REFERENC
   ES clients(id), rue TEXT NOT NULL, ville TEXT NOT NULL);
CREATE TABLE produits (id INTEGER PRIMARY KEY, nom TEXT NOT NULL, prix_centimes IN
   TEGER NOT NULL);
CREATE TABLE commandes (id INTEGER PRIMARY KEY, client_id INTEGER NOT NULL REFEREN
   CES clients(id), adresse_id INTEGER NOT NULL REFERENCES adresses(id), date TEXT
    NOT NULL);
CREATE TABLE lignes (commande_id INTEGER NOT NULL REFERENCES commandes(id), produi
   t_id INTEGER NOT NULL REFERENCES produits(id), quantite INTEGER NOT NULL, PRIMA
   RY KEY (commande_id, produit_id));
''')
bd.executemany("INSERT INTO clients (nom) VALUES (?)", [("Ada",), ("Bob",), ("Cléo
   ",)])
```

Mini-boss · réponse

Pistes de correction (suite)

```
bd.executemany("INSERT INTO adresses (client_id, rue, ville) VALUES (?, ?, ?)", [(
   1, "1 rue A", "Lyon"), (2, "2 rue B", "Lille"), (3, "3 rue C", "Nice")])
bd.executemany("INSERT INTO produits (nom, prix_centimes) VALUES (?, ?)", [("Stylo
   ", 150), ("Cahier", 300), ("Sac", 2500), ("Règle", 200), ("Gomme", 100)])
bd.executemany("INSERT INTO commandes (client_id, adresse_id, date) VALUES (?, ?,
   ?)", [(1, 1, "2025-01-01"), (1, 1, "2025-01-03"), (2, 2, "2025-01-05"), (2, 2,
   "2025-01-06")])
bd.executemany("INSERT INTO lignes VALUES (?, ?, ?)", [(1, 1, 10), (1, 2, 2), (2,
   3, 1), (3, 4, 3), (4, 5, 5), (4, 1, 2)])

print(bd.execute("SELECT produits.nom, lignes.quantite FROM lignes JOIN produits O
   N produits.id = lignes.produit_id WHERE commande_id = 1").fetchall())
print(bd.execute("SELECT commande_id, SUM(prix_centimes * quantite) FROM lignes JO
   IN produits ON produits.id = produit_id GROUP BY commande_id").fetchall())
print(bd.execute("SELECT nom FROM clients WHERE id NOT IN (SELECT client_id FROM c
   ommandes)").fetchall())
for sql in ("INSERT INTO lignes VALUES (99, 1, 1)", "INSERT INTO produits (nom, pr
   ix_centimes) VALUES (NULL, 100)"):
    try:
```

Mini-boss · réponse

Pistes de correction (suite)

```
        bd.execute(sql)
    except sqlite3.IntegrityError as e:
        print("refusé :", e)
```

Remarque : le prix est stocké en centimes (entier) pour éviter les erreurs d'arrondi des flottants.

Quiz éclair

Quiz éclair

1. Une clé primaire :

A. identifie chaque ligne de façon unique
B. relie deux bases
C. chiffre les données

2. Une clé étrangère :

A. est toujours un texte
B. référence la clé primaire d'une autre table
C. est facultative et sans effet

3. Pour une relation n-n entre membres et cours, on utilise :

A. une colonne contenant une liste

Quiz éclair

Quiz éclair (suite)

B. un fichier texte
C. une table de liaison

4. 🔁 *Révision HEF-060, mission 09* : Un pare-feu à règles ordonnées applique :

A. la première règle qui correspond
B. la dernière règle
C. toutes les règles

5. 🔁 *Révision HEF-060, mission 08* : Une clé publique peut être :

A. gardée secrète
B. partagée avec tout le monde
C. utilisée uniquement pour déchiffrer

Quiz éclair · réponse

Réponses

1. **A**.
2. **B**.
3. **C**.
4. **A**.
5. **B**.

Je réfléchis

1. Quelles informations de ta vie sont dupliquées à plusieurs endroits (adresses, numéros) et poseraient problème en cas de changement ?
2. Qu'est-ce qui rend une base de données plus sûre qu'un tableur partagé ?

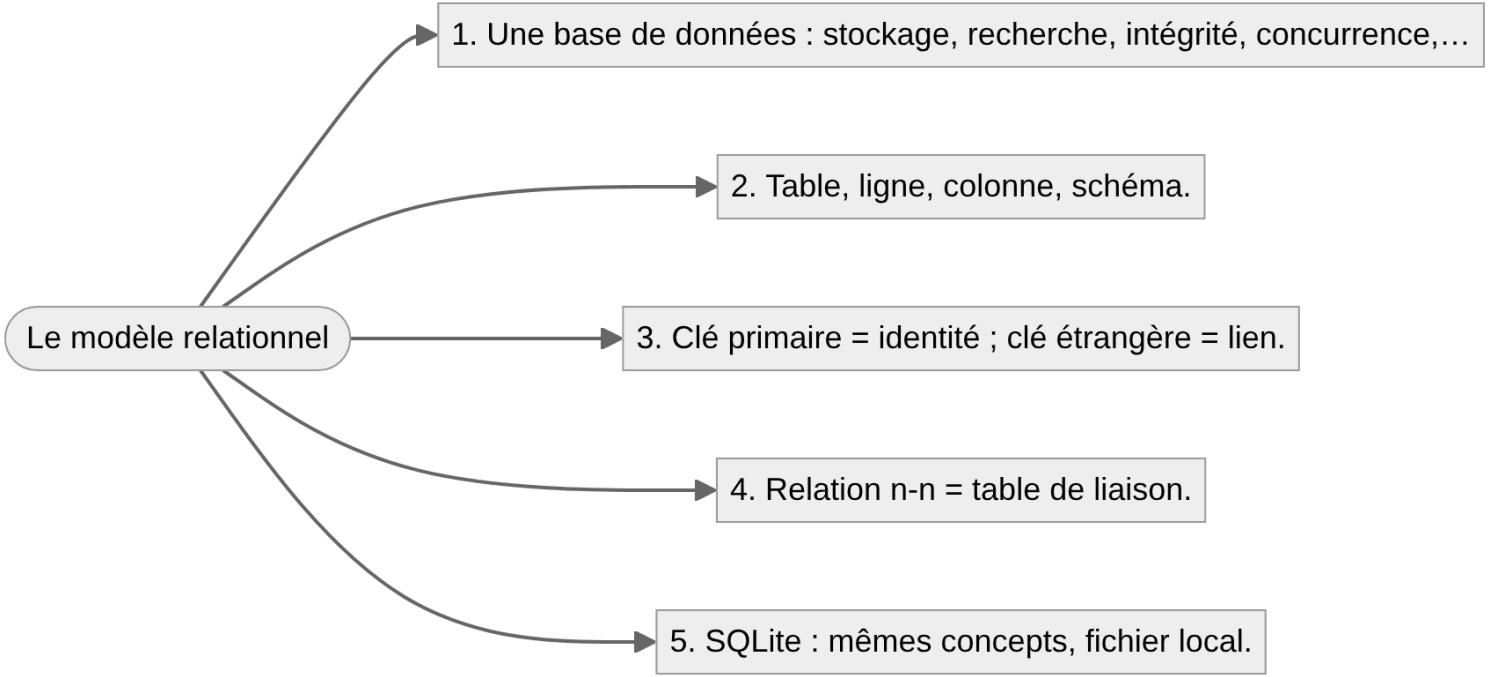
À retenir

À retenir

1. Une base de données : stockage, recherche, intégrité, concurrence, fiabilité.
2. Table, ligne, colonne, schéma.
3. Clé primaire = identité ; clé étrangère = lien.
4. Relation n-n = table de liaison.
5. SQLite : mêmes concepts, fichier local.

À retenir

Le schéma de la mission



MISSION ACCOMPLIE

+100 XP

Badge débloqué : Archiviste des données

Prochaine mission : Modélisation et normalisation

