OBJECTIFS

Objectif Global

Concevoir et créer une base de données SQL

Objectifs pédagogiques

- Savoir concevoir une base de donnée SQL
- Savoir mettre en oeuvre des scripts de création de base de données SQL
- Savoir mettre en oeuvre des scripts de remplissage de base de données SQL

Démarche pédagogique

Vous allez, en suivant par étape la méthode Merise, concevoir une base de données à partir d'un cahier des charges. Vous allez ensuite créer les scripts de création et d'insertion de données de test.

Compétences

- Créer un mémo
- Concevoir une base de données SQL
- Ecrire des scripts sql de création et d'insertion

MODALITÉS

Durée

1,5J

ITÉRATION 1

Conception

Modalités

- Travail individuel en autonomie

Livrables

Le schéma de la base de données créé à partir du cahier des charges

Objectifs

Concevoir un schéma de base de données SQL à partir d'un cahier des charges

Partie 1 : Analyse du cahier des charges

Cahier des charges : Mairie de NumCity

La mairie de NumCity a exprimé le besoin de mettre en place une solution efficace pour recenser plusieurs types de lieux utiles aux citoyens. Il s'agit notamment des points de parking pour vélos et trottinettes, ainsi que des points de recharge électrique. En parallèle, la mairie souhaite permettre aux utilisateurs de participer activement à ce recensement.

Chaque lieu devra être géolocalisé avec précision grâce à des coordonnées GPS, comprenant une latitude et une longitude. Pour mieux organiser ces lieux, ils seront regroupés dans des zones clairement définies, lesquelles seront modélisées par des polygones composés de plusieurs points GPS.

En ce qui concerne les points de recharge électrique, ils devront être spécifiés avec différents modes de charge disponibles, tels que les charges rapides et lentes, afin de répondre aux besoins variés des utilisateurs. Certains de ces lieux devront également inclure des équipements supplémentaires comme un chargeur pour les pneus, renforçant ainsi leur utilité pour les citoyens.

Les citoyens intéressés par ce projet devront pouvoir s'inscrire au système avec des informations minimales telles que leur nom, leur adresse e-mail et un mot de passe sécurisé. Une fois inscrits, ils auront la possibilité de proposer de nouveaux lieux à inclure dans la base de données ou de demander des modifications concernant des lieux existants.

Enfin, la base de données devra être conçue de manière à rester évolutive. Cela signifie qu'elle devra être suffisamment flexible pour permettre l'ajout futur de nouvelles fonctionnalités ou catégories de lieux sans nécessiter une refonte complète de son architecture.

Voici les points à considérer :

- Chaque lieu doit être géolocalisé avec des coordonnées GPS précises (latitude et longitude).
- Les lieux doivent être regroupés dans des zones définies par un polygone constitué de plusieurs points GPS.
- Les utilisateurs doivent s'inscrire avec un minimum d'informations : nom, adresse e-mail et mot de passe.
- Les utilisateurs doivent pouvoir proposer de nouveaux lieux ou demander des modifications sur des lieux existants.

- La base de données doit être conçue pour être facilement extensible afin de s'adapter à de futurs besoins.

Travail à réaliser

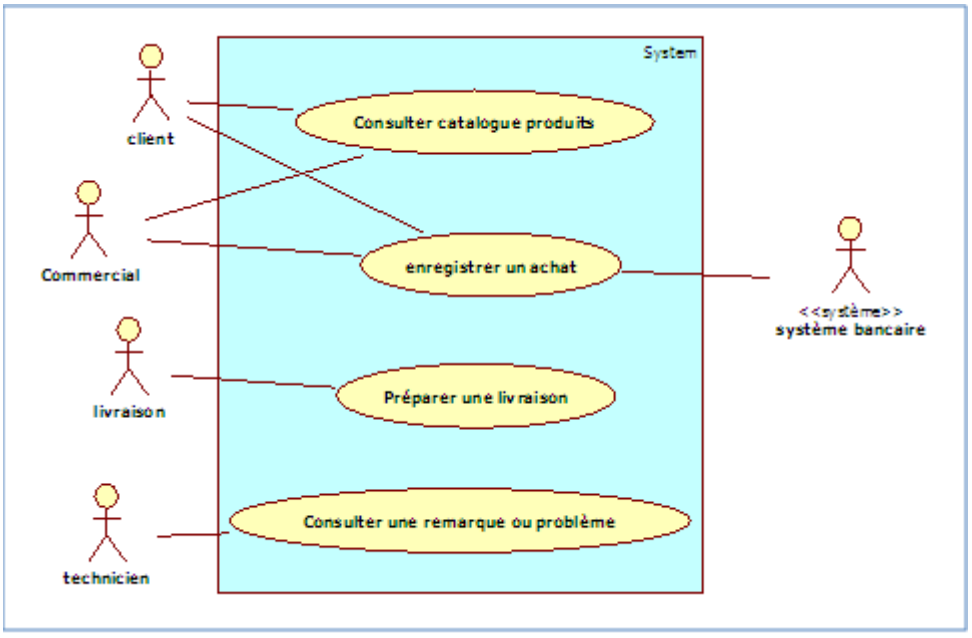
Analysez ce cahier des charges et identifiez les informations principales .

Partie 2 : Conception de la base de données avec la méthode Merise

La méthode Merise est une méthode de modélisation et de conception des systèmes d'information. Elle est structurée autour de trois niveaux principaux : le modèle conceptuel, le modèle logique et le modèle physique. Le modèle conceptuel permet de représenter les données de manière indépendante de toute technologie, en mettant l'accent sur les entités et les relations entre elles. Le modèle logique traduit ces concepts en structures compatibles avec le système de gestion de bases de données choisi, tandis que le modèle physique détaille les aspects techniques d'implémentation.

Étape 0 : Analyse du cahier des charges

Afin de s'assurer d'une bonne compréhension du cahier des charges, il est intéressant de créer un diagramme de Use Case UML. Ce diagramme permet de lister les rôles et les cas d'usage de l'application dans son ensemble. Voici un exemple de diagramme :



Le diagramme de use case se compose :

- un rectangle représentant le système
- des ovales dans le rectangles pour chaque cas d'usage (un cas d'usage commence par un verbe d'action)
- des "acteurs" qui interagissent avec les cas d'usage

Après avoir identifié les cas d'usages, pour approfondir la compréhension, il est recommandé de décrire chaque cas d'usage sous la forme de scénario. Par exemple, le cas d'usage "se connecter" peut se représenter sous la forme suivante :

1. l'utilisateur clique sur se connecter
2. le système affiche un formulaire de connexion
3. l'utilisateur saisit son login et son mot de passe
4. l'utilisateur clique sur se connecter
5. le système vérifie le login et le mot de passe
6. le système informe du succès

Le travail sur les scénarios vous aidera à identifier les jeux de données de test.

Quelques ressources :

- https://fr.wikipedia.org/wiki/Diagramme_de_cas_d%27utilisation
- https://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-cas-utilisation

Étape 1 : Création d'un dictionnaire de données

Un dictionnaire de données est un tableau qui détaille les informations nécessaires pour chaque entité de la base de données. Son objectif principal est de définir clairement les attributs, leur type, leur rôle, et les contraintes qui leur sont appliquées. Ce tableau contient généralement les colonnes suivantes :

1. **Code mnémonique** : Identifiant unique.
2. **Nom de l'attribut** : Désigne précisément le champ à inclure dans la base de données.
3. **Type de données** : Spécifie le type SQL adapté, tel que VARCHAR, INTEGER, ou FLOAT.
4. **Contraintes** : Liste les règles spécifiques pour cet attribut, comme NOT NULL, UNIQUE.
5. **Description** : Fournit une explication succincte de ce que représente l'attribut et son utilité.

Prenez le temps de lister vos idées et d'imaginer comment elles pourront être utilisées dans des scénarios pratiques.

Pour approfondir vos connaissances sur la méthode Merise et la création de dictionnaire, vous pouvez consulter les ressources suivantes :

- https://ineumann.developpez.com/tutoriels/merise/initiation-merise/

Étape 2 : Modèle conceptuel de données (MCD)

Le MCD, ou Modèle Conceptuel de Données, est une représentation graphique qui montre les relations entre vos entités et clarifie leur structure. Une **entité** représente un objet ou un concept important pour le système (par exemple, un utilisateur, un lieu, ou une zone), tandis qu'une **relation** décrit le lien entre ces entités (par exemple, un utilisateur propose un lieu). Chaque relation précise la nature de l'interaction et ses cardinalités, qui indiquent combien d'instances d'une entité sont associées à une instance de l'autre entité.

Pour approfondir vos connaissances sur les MCD et leur conception, vous pouvez consulter les ressources suivantes :

- https://ineumann.developpez.com/tutoriels/merise/initiation-merise/

Pour faciliter la création du MCD et les étapes suivantes, vous pouvez utiliser le logiciel Looping. Ce logiciel fonctionne sous Windows mais peut également fonctionner sur Linux avec Wine (moins stable)

Voici les questions que vous devez vous poser pour créer un MCD :

1. **Relations principales :**
   - Comment les utilisateurs interagissent-ils avec les lieux ?
   - Un utilisateur peut-il être lié à plusieurs propositions ?
   - Un lieu peut-il être associé à plusieurs zones ?

2. **Cardinalités :**
   - Un utilisateur peut-il proposer un seul lieu ou plusieurs ?

- Une zone peut-elle inclure un seul lieu ou plusieurs ?

Utilisez un outil comme DB-Main, Lucidchart ou tout autre logiciel de diagramme pour créer votre MCD. Ces outils permettent de visualiser vos entités et leurs relations de manière intuitive. DB-Main est un logiciel gratuit bien adapté aux débutants, tandis que Lucidchart propose une interface moderne et collaborative. D'autres alternatives incluent Visual Paradigm, qui offre une version éducative gratuite, ou des outils en ligne comme draw.io. Vous pouvez consulter des tutoriels spécifiques sur ces outils sur leurs sites officiels ou des plateformes comme YouTube. Une fois que vous avez produit une version initiale, demandez-vous si les relations sont logiques et complètes.

Étape 3 : Modèle logique de données (MLD)

Transformez votre MCD en MLD en adoptant une vue tabulaire. Cette transition se fait en plusieurs étapes :

1. **Identifiez les entités et relations dans votre MCD** : Chaque entité devient une table dans le MLD. Les relations sont traduites en tables si elles impliquent une relation de type N:N, ou en clés étrangères si elles sont de type 1:N ou 1:1.
2. **Ajoutez les attributs aux tables** : Transférez les attributs identifiés dans le MCD aux tables correspondantes dans le MLD, tout en veillant à définir les types de données appropriés pour chaque attribut.
3. **Définissez les clés primaires** : Chaque table doit avoir une clé primaire unique qui identifie ses enregistrements. Utilisez souvent un identifiant automatique (SERIAL en PostgreSQL).
4. **Ajoutez les clés étrangères** : Pour représenter les relations, insérez des clés étrangères dans les tables concernées. Cela établit un lien logique entre les tables.
5. **Spécifiez les contraintes** : Ajoutez des contraintes comme NOT NULL, UNIQUE, ou des règles spécifiques selon les besoins identifiés.

Pour des explications détaillées et des exemples, vous pouvez consulter des ressources comme :

- https://ineumann.developpez.com/tutoriels/merise/initiation-merise/

Points à considérer :

- Définissez les tables associées à chaque entité et relation.

- Assurez-vous que chaque table ait une clé primaire bien définie.
- Identifiez les clés étrangères et les types de relations (1:N, N:N).
- Choisissez des types de données PostgreSQL adaptés (par exemple, SERIAL pour les identifiants automatiques, VARCHAR pour les textes, et des types géographiques pour les coordonnées GPS).

Réfléchissez à la meilleure façon de structurer vos tables afin de faciliter les requêtes futures. Une fois cette structure définie, générez un schéma visuel de la base de données pour mieux comprendre les relations entre les tables et vérifier leur cohérence. Vous pouvez utiliser des outils comme draw.io, Lucidchart ou DB-Main pour créer ce schéma.

Étape 4 : Normalisation

La normalisation est un processus méthodique utilisé pour structurer les données dans une base de manière logique et efficace. Elle a pour objectif principal de minimiser les redondances, d'éviter les anomalies lors des opérations de mise à jour, d'insertion ou de suppression, et de garantir la cohérence des données. Le processus de normalisation passe par plusieurs étapes appelées formes normales :

Pour approfondir votre compréhension de la normalisation, vous pouvez explorer ces ressources :

- https://sgbd.developpez.com/tutoriels/cours-complet-bdd-sql/?page=normalisation

1. **Première forme normale (1NF) :**

- Toutes les valeurs doivent être atomiques.

Un exemple typique d'une base de données qui ne respecte pas la première forme normale (1NF) est une table qui contient des colonnes avec des valeurs multiples ou non atomiques. Par exemple :

ClientID	Nom	Commandes
1	Dupont	Commande1,Commande2
2	Martin	Commande3

Dans cette structure, la colonne Commandes contient plusieurs valeurs séparées par une virgule, ce qui enfreint la règle selon laquelle toutes les valeurs doivent être atomiques. Pour respecter la 1NF, cette table peut être corrigée en séparant les commandes dans une nouvelle table et en établissant une relation :

Table Clients :

ClientID	Nom
1	Dupont
2	Martin

Table Commandes :

CommandeID	ClientID
Commande1	1
Commande2	1
Commande3	2

Cela permet de s'assurer que chaque valeur dans chaque colonne est atomique et respecte la 1NF.

2. **Deuxième forme normale (2NF) :**

- Éliminez les dépendances partielles entre colonnes et clés primaires.

Un exemple d'une base de données qui ne respecte pas la deuxième forme normale (2NF) est une table où des colonnes dépendent partiellement de la clé primaire. Par exemple :

CommandeID	ClientNom	ClientAdresse	DateCommande
1	Dupont	Paris	2024-01-01
2	Dupont	Paris	2024-01-02

Ici, ClientNom et ClientAdresse dépendent de CommandeID mais ne concernent pas directement l'information spécifique à une commande. Cela cause une redondance, car les informations sur le client sont répétées pour chaque commande.

Pour respecter la 2NF, vous pouvez diviser cette table en deux :

Table Clients :

ClientID	ClientNom	ClientAdresse
1	Dupont	Paris

Table Commandes :

CommandeID	ClientID	DateCommande
1	1	2024-01-01
2	1	2024-01-02

Cela supprime la dépendance partielle, en liant les commandes aux clients via une clé étrangère ClientID. Ainsi, chaque information est bien isolée selon sa fonction.

3. **Troisième forme normale (3NF) :**

- Assurez-vous que tous les attributs ne dépendent que de la clé primaire.

Une base de données qui ne respecte pas la troisième forme normale (3NF) contient des attributs qui dépendent de manière transitive de la clé primaire. Par exemple :

CommandeID	ClientID	ClientNom	ClientAdresse	DateCommande
1	1	Dupont	Paris	2024-01-01
2	1	Dupont	Paris	2024-01-02

Dans cette table, ClientNom et ClientAdresse dépendent de ClientID et non directement de la clé primaire CommandeID. Cela entraîne une redondance inutile et complique la mise à jour des données.

Pour respecter la 3NF, vous pouvez séparer les informations sur le client dans une nouvelle table :

Table Clients :

ClientID	ClientNom	ClientAdresse
1	Dupont	Paris

Table Commandes :

CommandeID	ClientID	DateCommande
1	1	2024-01-01
2	1	2024-01-02

Ainsi, chaque table contient uniquement des attributs qui dépendent directement de sa clé primaire, éliminant toute dépendance transitive.

Prenez un moment pour revérifier votre modèle. **Identifiez des pistes d'amélioration.** Vous pourriez, par exemple, vérifier la présence de redondances dans les tables, optimiser les relations en ajustant les clés étrangères, ou valider les types de données pour qu'ils correspondent aux besoins métiers. Ces actions permettront d'assurer une meilleure efficacité et cohérence de votre base de données.

ITÉRATION 2

Scripts

Modalités

- Travail individuel en autonomie

Livrables

Les scripts de création et d'insertion sont fonctionnels et conformes à la conception

Objectifs

Créer une base de données en utilisant le langage SQL.

Partie 3 : Mise en œuvre technique

Script SQL pour la création de la base

Réalisez un script SQL pour créer votre base de données en respectant votre MLD. Voici un exemple :

```
CREATE DATBASE numcity;

 CREATE TABLE utilisateurs (
        id SERIAL PRIMARY KEY,
        nom VARCHAR(100) NOT NULL,
        email VARCHAR(255) UNIQUE NOT NULL,
        mot_de_passe VARCHAR(255) NOT NULL,
   date_inscription TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Votre mission :

- Créez les tables pour chaque entité et chaque relation.
- Ajoutez des contraintes comme les clés étrangères, l'unicité, et les valeurs par défaut.

Test et optimisation

1. **Insérer des données de test :** Commencez par insérer des données dans vos tables pour vérifier que votre base de données fonctionne correctement. Par exemple :

```
INSERT INTO utilisateurs (nom, email, mot_de_passe) VALUES ('Jean Dupont',
'jean.dupont@example.com', 'motdepasse1');
 INSERT INTO utilisateurs (nom, email, mot_de_passe) VALUES ('Alice Martin',
'alice.martin@example.com', 'motdepasse2');
```

2. **Effectuer des requêtes simples :** Testez des requêtes simples comme la récupération de données pour vous assurer que les informations sont correctement stockées et accessibles. Exemple :

```
SELECT * FROM utilisateurs;
```

3. **Tester les relations entre les tables :** Vérifiez que les relations fonctionnent comme prévu en insérant des données liées et en effectuant des jointures. Par exemple :

```
INSERT INTO commandes (client_id, date_commande) VALUES (1, '2024-01-01');
SELECT utilisateurs.nom, commandes.date_commande FROM utilisateurs
JOIN commandes ON utilisateurs.id = commandes.client_id;
```

4. **Valider les contraintes :** Essayez d'insérer des données invalides pour vérifier que les contraintes définies dans vos tables sont bien respectées (par exemple, unicité d'un e-mail ou absence de valeur NULL).

5. **Optimiser les performances (optionnel) :** Ajoutez des index sur les colonnes les plus recherchées pour améliorer les performances de vos requêtes. Exemple :

```
CREATE INDEX idx_email ON utilisateurs(email);
```

6. **Analyser et ajuster :** Utilisez les outils d'analyse fournis par PostgreSQL comme EXPLAIN ou ANALYZE pour vérifier le coût des requêtes et identifier des améliorations possibles.

En suivant ces étapes, vous vous assurez que votre base de données est robuste et prête pour une utilisation dans un environnement réel. **Testez votre base :**

- Insérez des données, effectuez des mises à jour et des suppressions.
- Réalisez des requêtes complexes pour valider les relations.

2. **Optimisez les performances (optionnel) :** L'optimisation des performances est essentielle pour garantir une exécution rapide des requêtes et une gestion efficace des ressources système, surtout lorsque le volume de données augmente. Par exemple, la création d'index sur des colonnes souvent recherchées peut accélérer les opérations, tandis que l'analyse des requêtes à l'aide d'outils comme EXPLAIN ou ANALYZE permet de détecter les goulots d'étranglement. Pour approfondir, explorez des ressources telles que :

- https://neon.tech/postgresql/postgresql-indexes/postgresql-create-index
- PGTune - Generate optimized configurations for PostgreSQL.
  - Créez des index sur les colonnes souvent recherchées (par exemple, les e-mails des utilisateurs ou les coordonnées GPS).
  - Analysez le coût des requêtes et ajustez votre structure en conséquence.

Partie 4 : Projet final

Pour clôturer ce cours, vous allez livrer un projet complet qui inclut :

1. Un dictionnaire de données bien documenté.
2. Les modèles MCD et MLD avec leurs justifications.
3. Un script SQL pour la création de la base et l'insertion de données de test.
4. Un rapport final expliquant vos choix, vos décisions techniques, et les améliorations possibles.

Conclusion

Ce cours vous offre une approche rigoureuse pour concevoir et implémenter une base de données relationnelle. Chaque étape est une occasion de renforcer vos compétences, alors prenez le temps d'explorer, d'expérimenter et d'apprendre. Vous aurez ainsi toutes les bases pour réaliser des projets complexes dans le futur.