“Requêter une base de données avec SQL”

Ce projet a pour objectif de créer et gérer une base de données pour analyser le marché des assurances habitation en France avec SQL.

Objectifs

- Comprendre les types de données utilisées.
- Créer un schéma relationnel pour structurer les données.
- Création d'une base de données.
- Charger les données dans un SGBD et vérifier leur intégrité.
- Réaliser des analyses avec des requêtes SQL.

1.1 Exploration des données

Dictionnaire des données: *"CONTRAT.CSV"*

Données sur les contrats d'assurance habitation

Nom des colonnes	Type de données	Taille	Clé	Description
Contrat_ID	VARCHAR	8	PK	ID unique pour les contrats. Les données sont seulement des chiffres avec une taille de 6.
No_voie	VARCHAR	5		Numéro de voie pour l'adresse du logement assuré.
B_T_Q	VARCHAR	1		Indicateur éventuel de répétition pour l'adresse du logement assuré sur un caractère
Type_de_voie	VARCHAR	5		Abbréviation du type de voie pour l'adresse du logement assuré: RUE, AV (Avenue), QUAI, ecc
Voie	VARCHAR	40		Libellé de la voie pour l'adresse du logement assuré
Code_dep_code_commune	VARCHAR	6	PFK	Concaténation du code départemental officiel avec le code commune pour avoir une clé unique
Code_postal	CHAR	5		Code postal pour l'adresse du logement assuré

Nom des colonnes	Type de données	Taille	Clé	Description
Surface	MEDIUMINT UNSIGNED	5		La surface du logement assuré en mètres carrés
Type_local	VARCHAR	20		Le type du bien assuré (appartement, maison.. Ecc)
Occupation	VARCHAR	35		La personne qui occupe le bien (locataire, propriétaire... etc.)
Type_contrat	VARCHAR	50		Le type de contrat que l'assurance a stipulé avec le client.
Formule	VARCHAR	15		La formule choisie par le client avec le contrat stipulé
Valeur_declaree_biens	VARCHAR	12		La valeur déclarée pour les biens.
Prix_cotisation_mensuel	MEDIUMINT UNSIGNED	5		Le prix de la cotisation mensuelle pour le client

1.2 Exploration des données

Dictionnaire des données: *"REGION.CSV"*

Référentiel des régions françaises extrait de *data.gouv.fr*

Nom des colonnes	Type de données	Taille	Clé	Description
Code_dep_code_commune	VARCHAR	6	PK	Concaténation du code départemental officiel avec le code commune pour avoir une clé unique.
reg_code	VARCHAR	2		L'identifiant régional officiel sans préfixe.
reg_nom	VARCHAR	26		Le nom officiel de la région.
aca_nom	VARCHAR	30		Le centre académique de la zone geographique.
dep_nom	VARCHAR	43		Le nom officiel du département.
com_nom_maj_court	VARCHAR	45		Le nom de la ville.
dep_code	VARCHAR	3		Le code départamental officiel
dep_nom_num	*VARCHAR*	*28*		*Le nom du département avec le code départamental* (redondance: c'est la concatenation de "dep_nom" + "dep_code")

Chaque type de donnée a été associé à une contrainte de taille spécifique
afin d'optimiser l'efficacité de la base de données et d'accélérer les requêtes et manipulations.

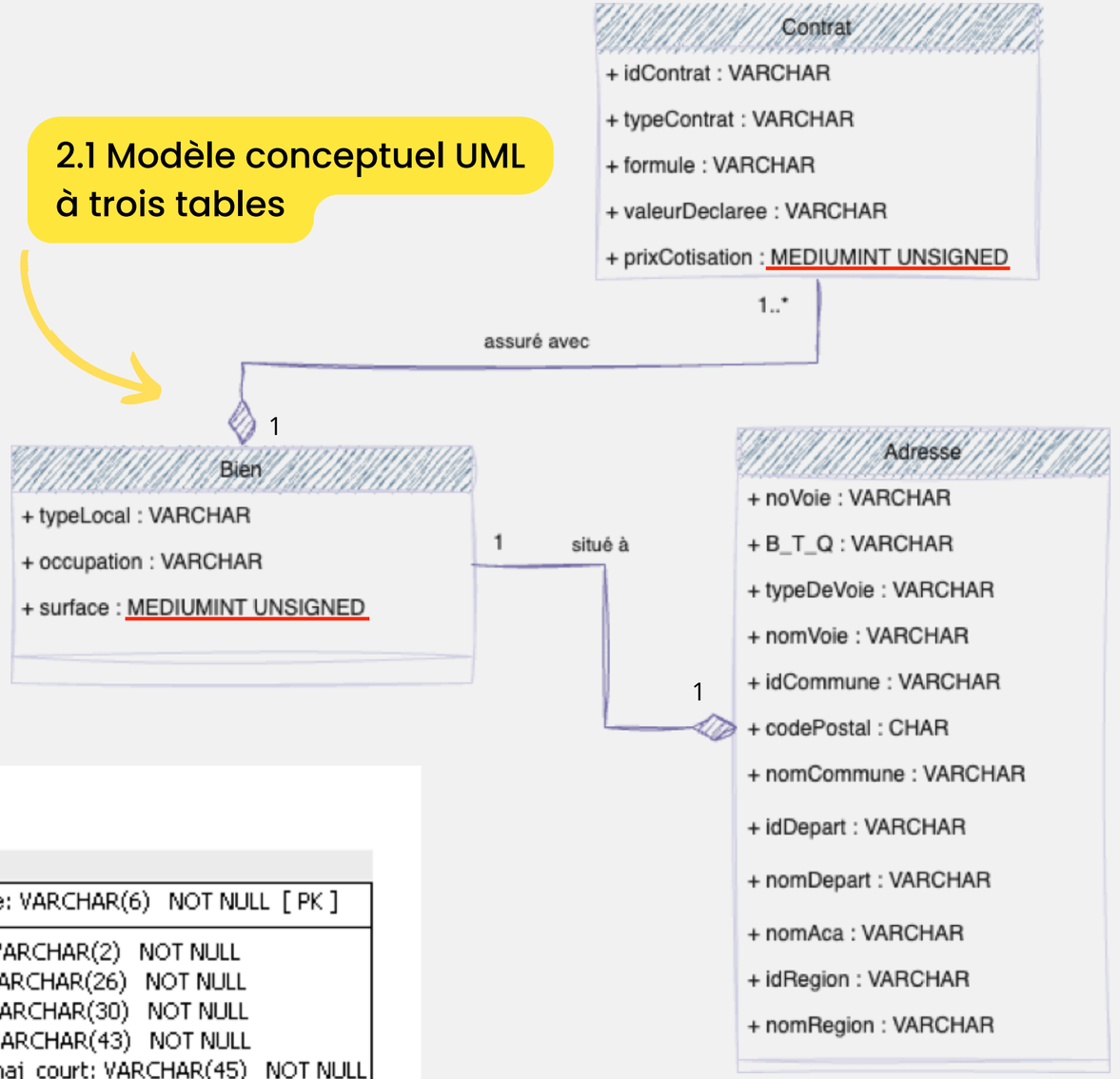
Cette démarche s'appuie sur une recherche approfondie, prenant en compte
toutes les possibilités d'insertion et une analyse minutieuse des données existantes.

2. Modèle conceptuel UML et modèle relationnel

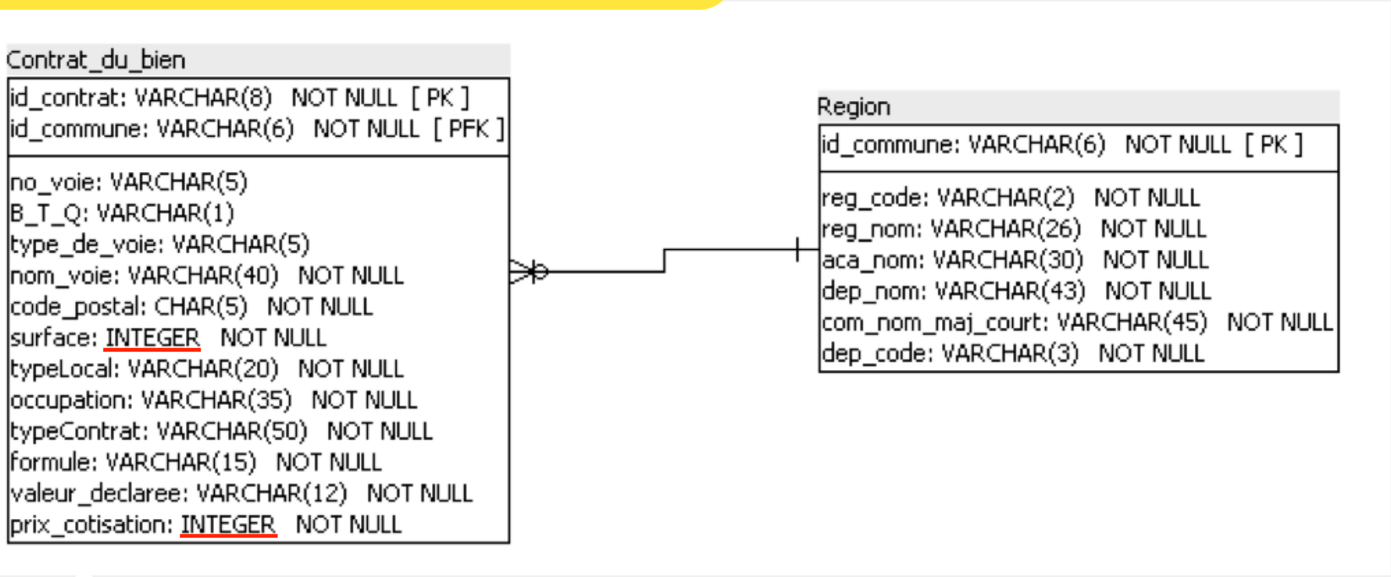
Une analyse initiale basée sur un modèle conceptuel UML a conduit à envisager une structure à trois tables pour mieux représenter certaines relations et détails des données.

Cependant, la structure finale a été ajustée à deux tables, conformément aux exigences du projet, afin de respecter les consignes et éviter des modifications importantes des fichiers de données fournies.

2.1 Modèle conceptuel UML à trois tables



2.2 Schéma relationnel normalisé 3NF à deux tables



3. Code SQL pour créer les tables

Le schéma relationnel diffère légèrement en raison des limites de *SQL Power Architect*, qui n'accepte pas *MEDIUMINT UNSIGNED* ni certaines contraintes, ajustées manuellement dans le code SQL final.

```sql
CREATE TABLE Region (
                id_commune VARCHAR(6) NOT NULL UNIQUE,
                reg_code CHAR(2) NOT NULL,
                reg_nom VARCHAR(26) NOT NULL,
                aca_nom VARCHAR(30) NOT NULL,
                dep_nom VARCHAR(43) NOT NULL,
                com_nom_maj_court VARCHAR(45) NOT NULL,
                dep_code VARCHAR(3) NOT NULL,
                PRIMARY KEY (id_commune)
);

```

```sql
CREATE TABLE Contrat_du_bien (
                id_contrat VARCHAR(8) NOT NULL UNIQUE,
                id_commune VARCHAR(6) NOT NULL,
                no_voie VARCHAR(5),
                B_T_Q VARCHAR(1),
                type_de_voie VARCHAR(5),
                nom_voie VARCHAR(40) NOT NULL,
                code_postal CHAR(5) NOT NULL,
                surface MEDIUMINT UNSIGNED NOT NULL,
                typeLocal VARCHAR(20) NOT NULL,
                occupation VARCHAR(35) NOT NULL,
                typeContrat VARCHAR(50) NOT NULL,
                formule VARCHAR(15) NOT NULL,
                valeur_declaree VARCHAR(12) NOT NULL,
                prix_cotisation MEDIUMINT UNSIGNED NOT NULL,
                PRIMARY KEY (id_contrat, id_commune)
);

ALTER TABLE Contrat_du_bien ADD CONSTRAINT region_contrat_du_bien_fk
FOREIGN KEY (id_commune)
REFERENCES Region (id_commune)
ON DELETE CASCADE
ON UPDATE CASCADE;
```

4. Chargement de la base de données

4.1 Système choisi

MySQL et *MySQL WORKBENCH*

4.2 Correction des incohérences

Un problème a été détecté dans les données initiales :
la clé primaire étrangère (*code_dep_code_commune*) ne correspondait pas à la clé primaire de la table "*REGION*" dans trois cas spécifiques (*97434, 97460, 97470*).

À la place du code commune, le code postal avait été inséré par erreur. Cette incohérence empêchait l'insertion des données.

Pour résoudre ce problème, les valeurs ont été corrigées manuellement.

4.3 Chargement des données

Les fichiers CSV ont été chargés dans les tables "*contrat_du_bien*" (**30 335 lignes**) et "*region*" (**38 916 lignes**),

```
Query OK, 30335 rows affected (1.62 sec)
Records: 30335  Deleted: 0  Skipped: 0  Warnings: 0
```

```
Query OK, 38916 rows affected (1.28 sec)
Records: 38916  Deleted: 0  Skipped: 0  Warnings: 0
```

puis leur insertion complète a été vérifiée, avec le code SQL suivant:

```sql
SELECT count(DISTINCT id_contrat)
AS count_data FROM contrat_du_bien
UNION
SELECT count(DISTINCT id_commune)
FROM region;
```

count_data
30335
38916

5. Requêtes SQL

Avec MySQL WORKBENCH

1 Comprendre la requête

Quelle est la surface moyenne des contrats à Paris ?

Lister les numéros de contrats avec leur surface pour la commune de Caen.

Quels sont les 5 contrats qui ont les surfaces les plus élevées ?

2 Traitement du code SQL

```sql
SELECT round(avg(c.surface)) AS Surface_Moyenne_m2
FROM contrat_du_bien c
JOIN region r ON c.id_commune = r.id_commune
WHERE r.com_nom_maj_court LIKE '%Paris %'
AND r.dep_nom = 'Paris';
```

```sql
SELECT c.id_contrat, c.surface AS surface_m2,
r.com_nom_maj_court
FROM contrat_du_bien c
JOIN region r ON c.id_commune = r.id_commune
WHERE lower(r.com_nom_maj_court) = 'caen';
```

```sql
SELECT id_contrat, surface AS surface_m2
FROM contrat_du_bien
ORDER BY surface DESC
LIMIT 5;
```

3 Visualisation du résultat sur MySQL

Surface_Moyenne_m2
52

id_contrat	surface_m2	com_nom_maj_court
103791	35	CAEN
103792	99	CAEN
103793	40	CAEN
103794	20	CAEN

id_contrat	surface_m2
104211	815
105463	742
130878	595
100822	570
109872	559

5. Requêtes SQL

Autres Exemples

```sql
SELECT round(avg(prix_cotisation))
AS prix_moyen_par_mois
FROM contrat_du_bien;
```

Le prix moyen des cotisations mensuels est de 19 euros

```sql
SELECT valeur_declaree,
count(valeur_declaree)
FROM contrat_du_bien
GROUP BY valeur_declaree
ORDER BY no_contrats DESC;
```

La majorité des biens est déclarée pour une valeur inférieure à 25 000 euros

Seulement 104 biens ont déclaré une valeur supérieure à 100 000 euros

```sql
SELECT DISTINCT r.reg_code,
r.reg_nom AS Region,
count(c.id_contrat) AS no_contrats
FROM contrat_du_bien c
JOIN region r ON c.id_commune = r.id_commune
GROUP BY r.reg_code, r.reg_nom
ORDER BY no_contrats;
```

L'Île de France est la région avec le plus de contrats.

La région avec le moins de contrats est la Réunion

```sql
SELECT r.dep_code, r.dep_nom,
round(avg(c.prix_cotisation))
AS Prix_moyen
FROM contrat_du_bien c
JOIN region r
ON c.id_commune = r.id_commune
GROUP BY r.dep_code, r.dep_nom
ORDER BY Prix_moyen DESC LIMIT 10;
```

La cotisation moyenne la plus chère est à Paris: 36 euros

```sql
SELECT r.reg_nom, c.formule,
count(c.formule) AS no_contrats
FROM contrat_du_bien c
JOIN region r ON c.id_commune = r.id_commune
WHERE lower(r.reg_nom) LIKE '%pays%loire%'
AND lower(c.formule) = 'integral'
GROUP BY r.reg_nom, formule;
```

Les contrats avec formule intégrale des Pays de la Loire sont 589

```sql
SELECT c.id_contrat, c.typeContrat,
c.formule, c.typeLocal, r.dep_code
FROM contrat_du_bien c
JOIN region r ON c.id_commune = r.id_commune
WHERE lower(c.typeLocal) LIKE '%maison%'
AND r.dep_code = '71'
ORDER BY c.id_contrat;
```

Dans le département 71, il y a 4 contrats pour les maisons, répartis équitablement entre les formules classique et intégrale.

6. Contraintes du projet

- colonne dep_nom_num redondante
- Modèle conceptuel des données qui conduit sur un modèle à trois tables, à la place de deux
- 3 données corrompues sur la clé étrangère empêchent l'insertion des données
- SQL Power Architect limite l'insertion de certains types de données
- Requête 2 égale à la rêquete 8

7. Solutions proposées

- colonne dep_nom_num pas prise en compte
- Retour au modèle original de deux tables du projet guidé, pour respecter les consignes
- 3 données corrigées à la main en recherchant le bon code_dep_code_commune
- Modifications souhaitées fait à la main directement dans le code SQL
- La requête 2 a été changée pour la différencier de la 8