Document technique

Requêtez une base de données avec SQL

Contenu du document

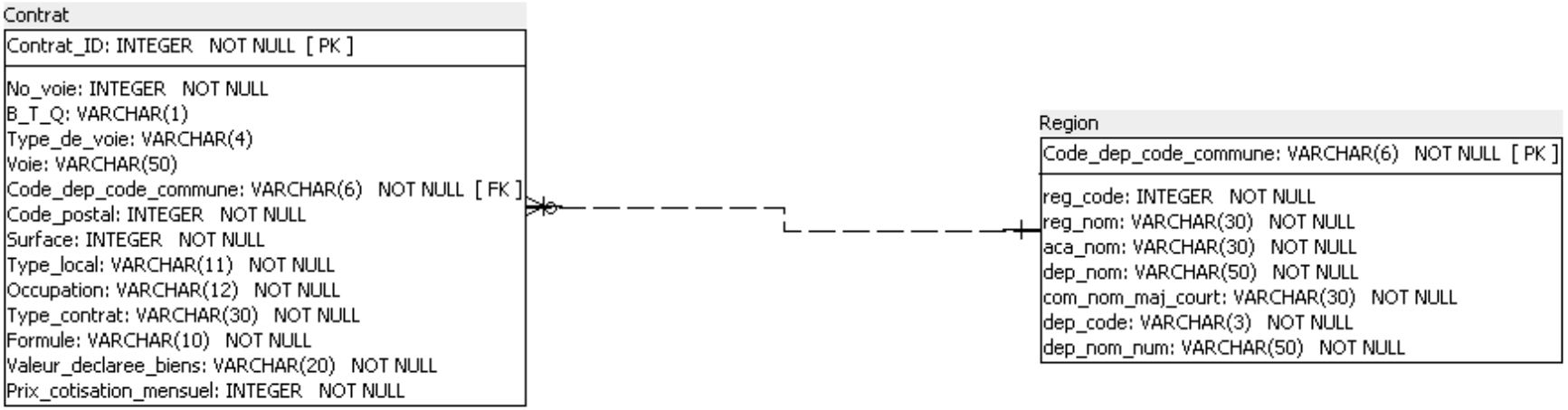
1) Capture d'écran du dictionnaire de données

	Nom des colonnes	Type de données	Taille	Clé	Description
CONTRAT.CSV	Contrat_ID	INT	6	Clé primaire	Id unique pour les contrats
	No_voie	INT	4		Numéro dans la 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	4		Type de voie pour l'adresse du logement assuré: rue, av (Avenue), rte (Route), ...
	Voie	VARCHAR	50		Libellé de la voie pour l'adresse du logement assuré
	Code_dep_code_commune	VARCHAR	6	Clé secondaire	Concaténation du code département et code commune pour avoir une clé unique
	Code_postal	INT	5		Code postal pour l'adresse du logement assuré
	Surface	INT	4		Surface du logement
	Type_local	VARCHAR	11		Type du logement (appartement ou maison)
	Occupation	VARCHAR	12		Locataire ou propriaitaire
	Type_contrat	VARCHAR	30		Type de contrat (résidence principale, secondaire ou mise en location)
	Formule	VARCHAR	10		Formule de compte (Integral ou classique)
	Valeur_declaree_biens	VARCHAR	20		Valeur déclarée du bien (quatre fourchettes de prix)
	Prix_cotisation_mensuel	INT	3		Montant de la cotisation mensuelle
REGION.CSV	Code_dep_code_commune	VARCHAR	6	Clé primaire	Concaténation du code département et code commune pour avoir une clé unique
	reg_code	INT	2		Code de la région
	reg_nom	VARCHAR	30		Nom de la région
	aca_nom	VARCHAR	30		Nom de l'académie
	dep_nom	VARCHAR	50		Nom du département
	com_nom_maj_court	VARCHAR	30		Nom de la commune
	dep_code	VARCHAR	3		Code du département
	dep_nom_num	VARCHAR	50		Département et son numéro

2) Lexique des termes utilisés dans le dictionnaire de données

1. **INT** : Un type de données qui représente un nombre entier. Il est utilisé pour stocker des valeurs numériques sans décimales, comme 1, 100, ou -25.
2. **VARCHAR** : Un type de données qui stocke du texte de longueur variable. On l'utilise pour des chaînes de caractères comme des noms ou des descriptions (par exemple, VARCHAR(50) permet de stocker jusqu'à 50 caractères).
3. **Clé primaire** : Un identifiant unique pour chaque ligne d'une table. Elle garantit que chaque entrée est unique et permet de retrouver rapidement les données. Par exemple, un numéro de client unique.
4. **Clé secondaire (ou clé étrangère)** : Un lien entre deux tables, utilisé pour relier les données d'une table à une autre. Elle référence une clé primaire d'une autre table pour créer une relation entre les deux tables.

3) Schéma relationnel de la base de données



NULL : Une valeur qui représente "aucune donnée" ou "inconnue". Si une colonne accepte NULL, cela signifie qu'elle peut rester vide pour certaines lignes.

NOT NULL : Une contrainte qui empêche une colonne d'être vide. Si une colonne est définie comme NOT NULL, elle doit toujours contenir une valeur pour chaque ligne.

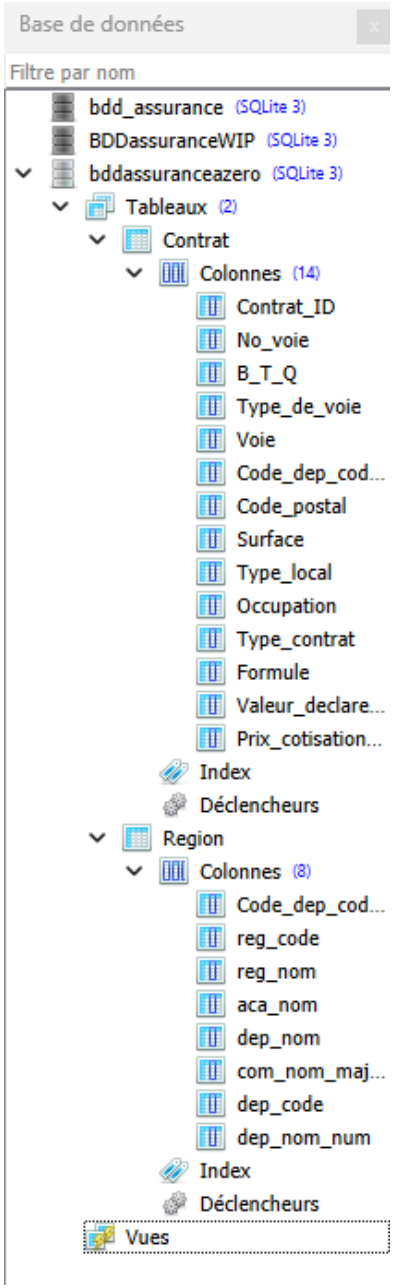
4) Code SQL générant les tables

```
CREATE TABLE Region (
        Code_dep_code_commune VARCHAR(6) NOT NULL,
        reg_code INTEGER NOT NULL,
        reg_nom VARCHAR(30) NOT NULL,
        aca_nom VARCHAR(30) NOT NULL,
        dep_nom VARCHAR(50) NOT NULL,
        com_nom_maj_court VARCHAR(30) NOT NULL,
        dep_code VARCHAR(3) NOT NULL,
        dep_nom_num VARCHAR(50) NOT NULL,
        CONSTRAINT Region_pk PRIMARY KEY (Code_dep_code_commune)
);


CREATE TABLE Contrat (
        Contrat_ID INTEGER NOT NULL,
        No_voie INTEGER NOT NULL,
        B_T_Q VARCHAR(1),
        Type_de_voie VARCHAR(4),
        Voie VARCHAR(50),
        Code_dep_code_commune VARCHAR(6) NOT NULL,
        Code_postal INTEGER NOT NULL,
        Surface INTEGER NOT NULL,
        Type_local VARCHAR(11) NOT NULL,
        Occupation VARCHAR(12) NOT NULL,
        Type_contrat VARCHAR(30) NOT NULL,
        Formule VARCHAR(10) NOT NULL,
        Valeur_declaree_biens VARCHAR(20) NOT NULL,
        Prix_cotisation_mensuel INTEGER NOT NULL,
        CONSTRAINT Contrat_pk PRIMARY KEY (Contrat_ID)
);


ALTER TABLE Contrat ADD CONSTRAINT Region_Contrat_fk
FOREIGN KEY (Code_dep_code_commune)
REFERENCES Region (Code_dep_code_commune)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;
```

5) Capture d’écran de la base de données chargées



Données importées dans le tableau 'Contrat' avec succès. Nombre de lignes importées : 30335

Données importées dans le tableau 'Region' avec succès. Nombre de lignes importées : 38916

6) Lexique des termes utilisés dans les requête SQL

1. **SELECT** : Utilisé pour choisir les colonnes à afficher dans les résultats d'une requête. Par exemple, SELECT nom récupère les valeurs de la colonne "nom".
2. **FROM** : Spécifie la table de laquelle on extrait les données. Par exemple, FROM clients indique que les données proviennent de la table "clients".
3. **WHERE** : Filtre les résultats pour ne récupérer que les lignes qui respectent une condition donnée. Par exemple, WHERE âge > 18 sélectionne uniquement les lignes où l'âge est supérieur à 18.
4. **JOIN** : Combine des données de plusieurs tables en fonction d'une condition commune. Par exemple, JOIN commandes ON clients.id = commandes.client_id relie la table "clients" à "commandes".
5. **AND** : Combine plusieurs conditions dans une requête. Par exemple, WHERE âge > 18 AND pays = 'France' filtre les lignes où l'âge est supérieur à 18 et le pays est la France.
6. **DISTINCT** : Supprime les doublons dans les résultats d'une requête. Par exemple, SELECT DISTINCT ville retourne chaque ville unique dans la colonne "ville".
7. **ORDER BY** : Trie les résultats par ordre croissant ou décroissant selon une ou plusieurs colonnes. Par exemple, ORDER BY nom ASC trie les résultats par "nom" en ordre croissant.
8. **LIMIT** : Limite le nombre de lignes dans les résultats. Par exemple, LIMIT 10 récupère seulement les 10 premières lignes.
9. **AVG** : Calcule la moyenne d'une colonne numérique. Par exemple, AVG(salaire) donne la moyenne des salaires dans les données.
10. **GROUP BY** : Regroupe les lignes ayant des valeurs identiques dans une ou plusieurs colonnes pour appliquer des fonctions d'agrégation (comme COUNT, AVG). Par exemple, GROUP BY département regroupe les résultats par département.

11. **HAVING** : Utilisé pour filtrer les groupes créés par GROUP BY selon une condition. Par exemple, HAVING COUNT(*) > 5 ne garde que les groupes contenant plus de 5 lignes.
12. **COUNT** : Compte le nombre de lignes dans les résultats. Par exemple, COUNT(*) donne le nombre total de lignes sélectionnées.