Documentation des fonctions `insert` et `update`

1. Fonction `insert`

La fonction `insert` permet d'insérer des données dans une table MySQL, avec support de : - Chiffrement de champs spécifiques - Gestion des relations enfants récursives (arborescence) - Retour des données insérées filtrées par `returnFields` - Gestion des clés primaires et clés uniques

1.1 Paramètres du JSON d'entrée

Champ	Type	Obligatoire	Description
`table`	string	Oui	Nom de la table cible.
`rows`	array[object]	Oui	Lignes à insérer. Chaque objet peut contenir des champs et un champ `children`.
`primaryKey`	array[string]	Non	Colonnes de la clé primaire pour récupérer la ligne après insertion.
`uniqueFields`	array[string]	Non	Colonnes à vérifier pour éviter les doublons avant insertion.
`encryptFields`	array[string]	Non	Colonnes à chiffrer avant insertion.
`returnFields`	array[string]	Non	Colonnes à retourner dans le résultat après insertion.
`decryptReturn`	boolean	Non	Déchiffre les champs spécifiés dans `encryptFields`. Par défaut `true`.

1.2 Structure pour enfants (`children`)

Chaque ligne peut contenir un champ `children` :

```
"children": [
  {
    "table": "child_table",
    "foreignKey": "parent_id",
    "rows": [ ... ],
    "primaryKey": ["id"],
    "encryptFields": [ ... ],
    "returnFields": [ ... ]
  }
]
```

- `foreignKey` relie l'enfant au parent. - Support récursif pour plusieurs niveaux d'enfants.

1.3 Exemple JSON

```
{
  "table": "users",
  "rows": [
    {
      "username": "john_doe",
      "email": "john@example.com",
      "password": "123456",
      "children": [
        {
          "table": "posts",
          "foreignKey": "user_id",
          "rows": [
            {"title": "Post 1", "content": "Contenu 1"}
          ],
          "primaryKey": ["id"],
          "returnFields": ["id", "title"]
        }
      ]
    }
  ],
  "primaryKey": ["id"],
  "encryptFields": ["password"],
  "returnFields": ["id", "username"]
}
```

---

2. Fonction `update`

La fonction `update` permet de modifier des données existantes dans une table MySQL.

2.1 Paramètres du JSON d'entrée

Champ	Type	Obligatoire	Description
`table`	string	Oui	Table à mettre à jour.
`data`	object	Oui	Données à modifier (champ: nouvelle valeur).
`where`	object	Oui	Conditions pour trouver les lignes à modifier. Supporte opérateurs (`=`, `>`, `<`, `IN`, `LIKE`).
`uniqueFields`	array[string]	Non	Colonnes à vérifier pour éviter les doublons.
`encryptFields`	array[string]	Non	Colonnes à chiffrer avant la mise à jour.
`returnFields`	array[string]	Non	Colonnes à retourner après la modification.

Champ	Type	Obligatoire	Description
`decryptReturn`	boolean	Non	Déchiffre les champs chiffrés spécifiés. Par défaut `true`.

2.2 Exemple JSON

```
{
  "table": "users",
  "data": {"username": "john_updated", "email": "new_email@example.com"},
  "where": {"id": 1},
  "encryptFields": ["email"],
  "returnFields": ["id", "username", "email"],
  "decryptReturn": true
}
```

2.3 Points importants

1. `table` et `where` sont obligatoires.
2. Support de la cryptographie via `encryptFields` et `decryptReturn`.
3. `returnFields` permet de limiter les colonnes retournées.
4. Vérification des `uniqueFields` empêche les doublons.
5. La fonction retourne un JSON :

```
{
  "success": true,
  "data": [ ... lignes modifiées ... ]
}
```