Documentation complète de la fonction `getDataWithFiltersAndPagination`

1. Présentation

La fonction `getDataWithFiltersAndPagination` permet de récupérer des données d'une base MySQL avec : - Filtrage via conditions (`where`) - Pagination - Tri (`orderBy`) - Gestion des relations imbriquées - Gestion de champs chiffrés (`encryptFields` / `decryptReturn`) - Sélection des colonnes à retourner (`returnFields`) - Arborescence des résultats (`tree`)

La fonction retourne un tableau JSON structuré :

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

2. Options principales

Champ	Type	Obligatoire	Description
`table`	string	Oui	Nom de la table principale.
`fields`	array[string]	Non	Colonnes à récupérer. Par défaut `['*']`.
`where`	array	Non	Conditions de filtre. Peut inclure `$or` ou `$and`, opérateurs (`=` , `>` , `<` , `IN` , `BETWEEN` , `LIKE`).
`orderBy`	array	Non	Tri des résultats. Exemple `[{'column':'created_at','direction':'DESC'}]`.
`pagination`	array	Non	Pagination `{page:1, limit:10}`.
`tree`	array	Non	Configuration d'arborescence des résultats `{idField:'id', parentField:'parent_id'}`.
`encryptFields`	array[string]	Non	Colonnes chiffrées à déchiffrer après récupération.
`returnFields`	array[string]	Non	Colonnes à retourner après récupération.
`decryptReturn`	boolean	Non	Déchiffre les champs chiffrés. Par défaut `true`.

3. Relations imbriquées

Chaque relation est un objet avec : | Champ | Type | Obligatoire | Description | |----------------|----------------|------------|------------| | `foreignKey` | string | Oui | Clé étrangère dans la table principale. | | `targetKey` | string | Non | Clé correspondante dans la table enfant (par défaut `id`). | | `fields` | array[string] | Non | Colonnes à récupérer dans la relation. | | `where` | array |

Non | Filtrage pour la relation. | | `orderBy` | array | Non | Tri pour la relation. | | `pagination` | array | Non | Pagination pour la relation. | | `tree` | array | Non | Arborescence spécifique pour la relation. | | `relations` | array | Non | Relations imbriquées (relation enfant de la relation). | | `returnFields` | array[string] | Non | Colonnes à retourner pour la relation. | | `encryptFields` | array[string] | Non | Colonnes à déchiffrer pour la relation. | | `decryptReturn` | boolean | Non | Déchiffrement conditionnel pour la relation.

4. Exemple JSON complet

```
{
  "table": "users",
  "fields": ["id", "username", "email", "created_at"],
  "where": {"$or": [{"status":"active"},{"role":"admin"}]},
  "orderBy": [{"column":"created_at","direction":"DESC"}],
  "pagination": {"page":1,"limit":10},
  "tree": {"idField":"id","parentField":"parent_id"},
  "encryptFields": ["email"],
  "returnFields": ["id", "username", "email"],
  "relations": {
    "posts": {
      "foreignKey": "id",
      "targetKey": "user_id",
      "fields": ["id", "title", "created_at"],
      "orderBy": [{"column":"created_at","direction":"DESC"}],
      "pagination": {"page":1,"limit":5},
      "encryptFields": ["title"],
      "returnFields": ["id", "title"],
      "relations": {
        "comments": {
          "foreignKey": "id",
          "targetKey": "post_id",
          "fields": ["id", "content", "created_at"],
          "returnFields": ["id", "content"]
        }
      }
    }
  }
}
```

5. Points importants

1. `table` et `relations[].foreignKey` sont obligatoires.
2. `fields`, `where`, `orderBy`, `pagination`, `tree`, `encryptFields`, `returnFields` et `decryptReturn` sont optionnels.
3. Supporte les relations imbriquées à plusieurs niveaux.
4. Applique le décryptage et le filtrage des colonnes à chaque niveau.
5. Permet l'arborescence des résultats avec `tree`.
6. Combine pagination, filtrage, tri et cryptographie.

Cette documentation peut être exportée au format PDF en utilisant n'importe quel convertisseur HTML/ Markdown → PDF.