

Modèle relationnel Vs Objet

Table des matières

I. Contexte	3
II. S'initier sur les aspects essentiels de ORDBMS	3
A. Object Relational Database	3
B. Exemple d'un ORD	3
C. ORDBMS	6
III. Comprendre la différence avec le modèle relationnel et quand utiliser chaque modèle	7
A. Éléments de ORD	7
B. Différence avec le modèle relationnel.....	8
IV. Essentiel	10
V. Auto-évaluation	11
A. Exercice	11
B. Test.....	11
Solutions des exercices	12

I. Contexte

Durée : 1 h

Environnement de travail : Draw.io / Visual-Paradigm

Prérequis : Connaître les bases de données relationnelles et les bases de SQL

Contexte

Les bases de données relationnelles sont largement utilisées pour stocker et gérer des données, mais elles peuvent présenter des limites lorsqu'il s'agit de gérer des données complexes ou non scalaires. C'est dans ce contexte qu'est née la notion de bases de données objet-relationnelles (ORD). Les ORD ont été conçues pour combiner les fonctionnalités des bases de données relationnelles et des bases de données orientées objet, afin de permettre une manipulation plus flexible des données complexes. Ces systèmes offrent la possibilité de représenter les données sous forme d'objets, d'utiliser des types de données définis par l'utilisateur et de bénéficier d'autres fonctionnalités avancées de la programmation orientée objet. Nous allons voir dans ce cours comment les ORD offrent une solution adaptée pour répondre aux besoins de gestion de données complexes dans différents domaines d'application, tout en tirant parti des avantages du modèle relationnel existant.

II. S'initier sur les aspects essentiels de ORDBMS

A. Object Relational Database

Définition

L'Object Relational Database (ORD), autrement connue sous le nom de base de données objet-relationnelle, est un type de base de données qui fusionne les capacités des bases de données relationnelles et des bases de données orientées objet. Elle offre un moyen plus flexible de manipuler des données, en particulier des données complexes ou non scalaires.

Dans une ORD, vous pouvez avoir des objets complexes constitués de plusieurs parties. Par exemple, vous pourriez avoir un objet « *voiture* » qui a des attributs tels que la marque, le modèle, l'année, la couleur, et même un autre objet « *propriétaire* ». De plus, les ORD permettent la définition de types de données définis par l'utilisateur, qui peuvent être utilisés pour créer des structures de données plus complexes.

B. Exemple d'un ORD

Les acronymes

Plus vous avancez dans votre apprentissage, plus vous rencontrez d'acronymes, certains en anglais, d'autres en français. Voici un tableau synthétisant les acronymes présents dans ce cours :

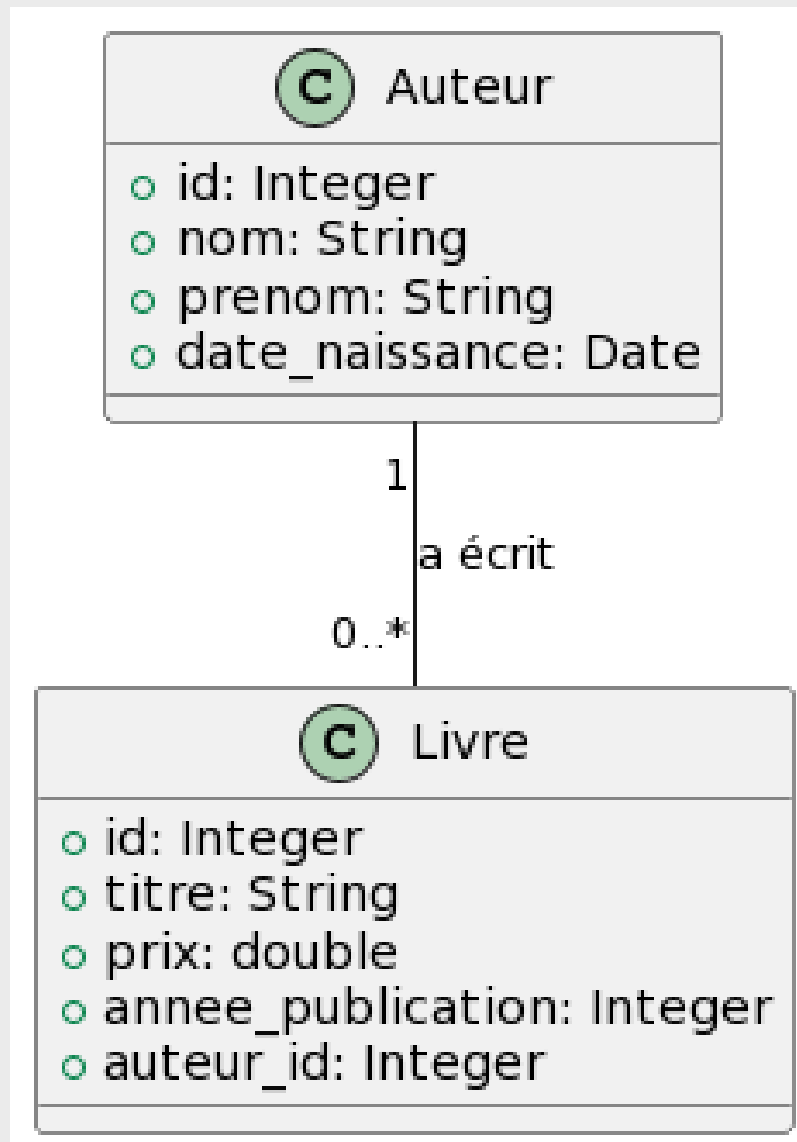
Acronyme (anglais)	Acronyme (français)	Explication
DB	BD	Base de données (Database) C'est un ensemble organisé et structuré d'informations stockées numériquement. Elles peuvent être manipulées et accédées de plusieurs manières.
ORD	BDO-R	Base de données Objet-Relationnelle (Object-Relational Database) Type de base de données qui combine les éléments des bases de données relationnelles et orientées objet.

Acronyme (anglais)	Acronyme (français)	Explication
DBMS	SGBD	Système de Gestion de Base de Données (Database Management System) Il s'agit du logiciel qui gère l'interaction entre les utilisateurs finaux, l'application et la base de données.
RDBMS	SGBDR	Système de Gestion de Base de Données Relationnelle (Relational Database Management System) Une base de données organisée autour de la notion de tables de données, reliées entre elles par des relations.
ORDBMS	SGBDOR	Système de Gestion de Base de Données Objet-Relationnel (Object-Relational Database Management System) Une base de données qui allie les caractéristiques des bases de données relationnelles et des bases de données orientées objet.
OODBMS	SGBDOO	Système de Gestion de Base de Données Orientée Objet (Object-Oriented Database Management System) Une base de données qui utilise la modélisation orientée objet pour la représentation des données.

Exemple Gestion d'une bibliothèque

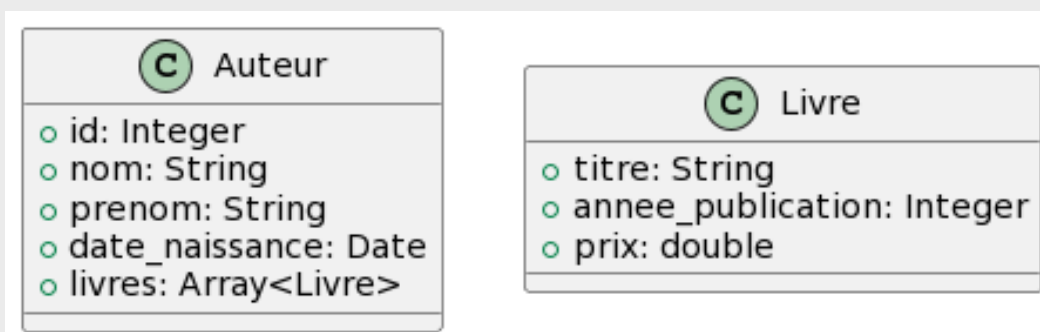
Prenons l'exemple d'un système de gestion de bibliothèque pour mieux comprendre le fonctionnement d'une base de données objet-relationnelle (ORD). Dans ce système, nous avons des entités telles que Livre et l'auteur. Dans une base de données relationnelle, nous aurions une table pour chaque entité, avec chaque ligne de la table représentant une instance de cette entité.

Par exemple, nous pourrions avoir une table « Livre » avec des colonnes pour le titre, le prix, etc., et une table « Auteur » avec des colonnes pour le nom, l'adresse, etc. Pour associer un livre à un auteur, nous aurions besoin d'une clé étrangère dans la table « Livre », indiquant l'id de l'auteur. Nous aurons donc une relation entre les deux.



DiagrammeRelationnel

Dans une ORD, cependant, nous pouvons avoir un objet *Auteur* qui comprend non seulement des attributs simples tels que le nom et l'adresse, mais aussi des attributs complexes tels qu'une liste de *Livres*. Chaque *Livre* pourrait alors être un objet avec ses propres attributs, comme le titre et le prix. On peut donc à l'avance créer un type « *Livre* », et définir comme attribut de ma table *auteur*, un tableau de ce type « *Livre* ». De cette façon, les données sont plus étroitement couplées et l'objet *Auteur* encapsule plus d'informations relatives à l'auteur et à ses œuvres.



DiagrammeObjet

C. ORDBMS

Définition

Un système de gestion de base de données objet-relationnelle (ORDBMS) ou Système de Gestion de Base de Données Relationnel-Objet (SGBDRO) est un logiciel qui implémente une base de données objet-relationnelle. Il offre un langage de requête similaire au SQL utilisé dans les bases de données relationnelles, mais avec des fonctionnalités supplémentaires pour supporter les concepts orientés objet.

Par exemple, un ORDBMS pourrait permettre d'écrire des requêtes qui exploitent l'héritage, un concept clé de la programmation orientée objet. Cela signifie que vous pouvez avoir une classe « *Publication* » avec des sous-classes « *Livre* » et « *Magazine* », et écrire une requête qui renvoie toutes les publications, qu'elles soient des livres ou des magazines.

Un autre avantage des ORDBMS est leur capacité à gérer des types de données définis par l'utilisateur. Par exemple, si vous développez une application pour une compagnie d'assurance, vous pourriez définir un type de données « *Police d'assurance* » avec des attributs tels que le nom du titulaire de la police, le montant de la couverture, la date d'expiration, etc.

Exemple

Voici quelques exemples de systèmes de gestion de bases de données objet-relationnelles (ORDBMS) largement utilisés :

- PostgreSQL : il s'agit de l'un des systèmes de bases de données open source les plus avancés. PostgreSQL offre un support robuste pour les fonctionnalités objet-relationnelles, notamment le support pour les types définis par l'utilisateur, les héritages de table, et les fonctions stockées.
- Oracle Database : Oracle offre des fonctionnalités objet-relationnelles dans sa base de données depuis la version 8i. Les utilisateurs peuvent créer des types d'objets, des tables d'objets et peuvent effectuer des requêtes qui retournent des objets entiers ou des attributs d'objets.
- IBM DB2 : DB2 offre également des fonctionnalités objet-relationnelles, y compris le support pour les types de données définis par l'utilisateur, les types de données structurés, les tables de référence, et les opérations sur les types de données structurés.
- Informix : Informix est un autre ORDBMS produit par IBM qui offre des fonctionnalités objet-relationnelles, y compris le support pour les types de données définis par l'utilisateur, les types de données structurés, et les requêtes qui retournent des objets ou des attributs d'objets.
- Microsoft SQL Server : SQL Server offre des fonctionnalités objet-relationnelles, y compris le support pour les types de données définis par l'utilisateur et les types de données structurés.

Veuillez noter que bien que ces systèmes de gestion de bases de données offrent des fonctionnalités objet-relationnelles, ils sont souvent utilisés comme des systèmes de gestion de bases de données relationnelles traditionnels. Le degré d'utilisation des fonctionnalités objet-relationnelles dépend souvent des besoins spécifiques de l'application.

III. Comprendre la différence avec le modèle relationnel et quand utiliser chaque modèle

A. Éléments de ORD

Définition

Les bases de données objet-relationnelles (ORD) intègrent des concepts de la programmation orientée objet (OOP ou POO en français), ce qui permet une plus grande flexibilité et une meilleure encapsulation des données. Par exemple, une ORD peut prendre en charge l'héritage, ce qui signifie que vous pouvez définir des types de données qui sont des sous-types d'autres types de données.

Les bases de données orientées objet sont étroitement liées aux concepts de la programmation orientée objet. Les quatre principales idées de la programmation orientée objet sont :

- Polymorphisme
- Héritage
- Encapsulation
- Abstraction

Ces quatre notions décrivent les caractéristiques essentielles des systèmes de gestion orientés objet. Ces quatre notions de POO ne sont pas utilisables dans une base de données relationnelles classique, alors qu'elles sont implémentables dans une base de données relationnelles orientée objet.

Polymorphisme

Le polymorphisme est la capacité d'un objet à prendre plusieurs formes. Cette capacité permet au même code de programme de travailler avec différents types de données.

Exemple

Une voiture et un vélo peuvent tous deux freiner, mais le mécanisme est différent. Dans cet exemple, l'action de freiner est un polymorphisme. L'action définie est polymorphe : le résultat change en fonction du véhicule qui la réalise.

Héritage

L'héritage crée une relation hiérarchique entre les classes apparentées tout en rendant des parties du code réutilisables. La définition de nouveaux types hérite de tous les champs et méthodes de la classe existante et les étend. La classe existante est la classe parente, tandis que la classe enfant étend la classe parente.

Exemple

Une classe parente appelée Véhicule aura des classes enfants Voiture et Vélo. Les deux classes enfants héritent des informations de la classe parente et étendent la classe parente avec de nouvelles informations en fonction du type de véhicule.

Encapsulation

L'encapsulation est la capacité de regrouper des données et des mécanismes en un seul objet pour assurer la protection de l'accès. Grâce à ce processus, des morceaux d'information et les détails du fonctionnement d'un objet sont cachés, ce qui se traduit par une sécurité des données et des fonctions. Les classes interagissent entre elles par le biais de méthodes sans qu'il soit nécessaire de savoir comment fonctionnent ces méthodes.

Exemple

Une voiture a des caractéristiques descriptives et des actions. Vous pouvez changer la couleur d'une voiture, mais le modèle ou la marque sont des exemples de propriétés qui ne peuvent pas changer. Une classe encapsule toutes les informations sur la voiture en une seule entité, où certains éléments sont modifiables tandis que d'autres ne le sont pas.

Abstraction

L'abstraction est la procédure de représentation uniquement des caractéristiques de données essentielles pour la fonctionnalité requise. Le processus sélectionne les informations vitales tandis que les informations inutiles restent cachées. L'abstraction aide à réduire la complexité des données modélisées et permet la réutilisabilité.

Exemple

Il existe différentes façons pour un ordinateur de se connecter au réseau. Un navigateur web a besoin d'une connexion internet. Cependant, le type de connexion est sans importance. Une connexion établie à internet représente une abstraction, tandis que les différents types de connexions représentent différentes mises en œuvre de l'abstraction.

B. Différence avec le modèle relationnel

Le modèle relationnel de base de données est basé sur l'idée de stocker des données dans des tables avec des lignes et des colonnes. Chaque ligne d'une table représente une entité (par exemple, un livre dans une bibliothèque), et chaque colonne représente un attribut de cette entité (par exemple, le titre du livre).

En revanche, le modèle objet-relationnel permet de stocker des données plus complexes et d'encapsuler à la fois les données et les opérations qui peuvent être effectuées sur ces données en un seul objet. Cela permet une plus grande flexibilité dans la modélisation des données. Le choix entre le modèle relationnel et le modèle objet-relationnel dépend de la nature de vos données et de vos besoins en matière de traitement des données. Le modèle relationnel peut être plus facile à comprendre et à utiliser pour des données simples, tandis que le modèle objet-relationnel peut offrir plus de flexibilité pour des données complexes.

En termes de SQL, les requêtes dans un modèle relationnel sont souvent plus simples, car elles sont basées sur des tables et des jointures. Par exemple, pour obtenir une liste de tous les auteurs et de leurs livres dans un modèle relationnel, nous pourrions faire une jointure entre la table Livre et la table Auteur.

Cependant, dans un modèle objet-relationnel, la même requête pourrait nécessiter de naviguer à travers les objets et leurs attributs. Par exemple, pour obtenir la même liste d'auteurs et de livres, nous pourrions avoir besoin d'appeler une méthode sur chaque objet Auteur pour obtenir ses livres.

	SGBD (Modèle relationnel)	SGBDRO (Modèle objet-relationnel)
Représentation des données	Les données sont représentées sous forme de tables, chaque ligne représentant une entité et chaque colonne un attribut.	Les données peuvent être représentées comme des objets, qui peuvent avoir des attributs de types simples ou complexes, y compris d'autres objets.
Manipulation des données	Les données sont manipulées à l'aide de SQL, en utilisant des opérations telles que SELECT, INSERT, UPDATE et DELETE.	Les données peuvent être manipulées à l'aide de SQL étendu, qui permet d'effectuer des opérations sur les attributs des objets, ainsi que d'appeler des méthodes définies sur les objets.

	SGBD (Modèle relationnel)	SGBDRO (Modèle objet-relationnel)
Relations	Les relations entre les données sont représentées par des clés étrangères.	Les relations peuvent être représentées par des liens entre objets, ce qui permet une encapsulation plus naturelle des relations.
Types de données	Supporte des types de données scalaires tels que INT, VARCHAR, DATE, etc.	Supporte des types de données scalaires ainsi que des types de données complexes et définis par l'utilisateur.
Flexibilité	Moins flexible pour représenter des données complexes.	Plus flexible pour représenter et manipuler des données complexes.
Complexité	Moins complexe à comprendre et à utiliser pour des données simples.	Plus complexe en raison de la prise en charge des concepts de la programmation orientée objet.
Exemples de SGBD/SGBDRO	MySQL, Oracle	PostgreSQL, Oracle (avec extensions OR), IBM DB2.

Méthode Création d'une table

Reprenons l'exemple de la gestion d'une bibliothèque. Avec un SGBDR classique, voici comment nous aurions pu créer nos deux tables :

```

1 CREATE TABLE Auteur (
2     id SERIAL PRIMARY KEY,
3     nom VARCHAR(100) NOT NULL,
4     prenom VARCHAR(100) NOT NULL,
5     date_naissance DATE NOT NULL
6 );
7
8 CREATE TABLE Livre (
9     id SERIAL PRIMARY KEY,
10    titre VARCHAR(255) NOT NULL,
11    annee_publication INT NOT NULL,
12    auteur_id INT,
13    FOREIGN KEY (auteur_id) REFERENCES Auteur (id)
14 );

```

Dans cet exemple, chaque Livre a un auteur_id qui fait référence à un Auteur. C'est une base de données relationnelle simple avec une relation un-à-plusieurs (one-to-many) entre Auteur et Livre.

Pour l'approche orientée objet, nous pouvons créer et utiliser le type composite Livre :

```

1 CREATE TYPE Livre AS (
2     titre VARCHAR(255),
3     annee_publication INT
4 );
5
6 CREATE TABLE Auteur (
7     id SERIAL PRIMARY KEY,
8     nom VARCHAR(100) NOT NULL,

```

```

9      prenom VARCHAR(100) NOT NULL,
10     date_naissance DATE NOT NULL,
11     livres Livre[]
12 );

```

Ici, chaque Auteur a un tableau de Livres. Chaque Livre est un type composite avec des champs titre et annee_publication. Cette conception a ses propres avantages et inconvénients. Un inconvénient majeur est que la recherche ou la modification de livres individuels dans le tableau de livres peut être plus compliquée que dans une base de données relationnelle.

Méthode Insérer des données

Avec notre base de données relationnelles, nous devons exécuter ces requêtes :

```

1 INSERT INTO Auteur (nom, prenom, date_naissance) VALUES ('Doe', 'John', '1970-01-01');
2 INSERT INTO Livre (titre, annee_publication, auteur_id) VALUES ('Mon livre', 2020, 1);

```

Avec une Base de Données Objet-Relationnelle, cette possibilité s'ouvre à nous :

```

1 INSERT INTO Auteur (nom, prenom, date_naissance, livres) VALUES ('Doe', 'John', '1970-01-01',
  ARRAY[('Mon livre', 2020) ::Livre]);

```

En utilisant une base de données relationnelles, nous devons insérer séparément dans Auteur et Livre tandis que dans une base de données relationnelles orientée objet, nous pouvons insérer un Auteur avec ses Livres en une seule opération.

Méthode Sélectionner des données

Avec notre base de données relationnelles, nous devons exécuter ces requêtes :

```

1 SELECT a.nom, a.prenom, l.titre FROM Auteur a JOIN Livre l ON a.id = l.auteur_id WHERE a.nom =
  'Doe';

```

Avec une Base de Données Objet-Relationnelle, cette possibilité s'ouvre à nous :

```

1 SELECT a.nom, a.prenom, l.titre FROM Auteur a, UNNEST(a.livres) l WHERE a.nom = 'Doe';

```

En utilisant RDBMS, nous utilisons une jointure pour obtenir les livres d'un auteur. En ORDBMS, nous utilisons la fonction UNNEST () pour extraire les livres de l'attribut livres de Auteur.

IV. Essentiel

Ce cours sur les bases de données objet-relationnelles (ORD) est important, car il nous permet de comprendre comment gérer des données complexes et non scalaires de manière plus flexible. Les ORD combinent les fonctionnalités des bases de données relationnelles et des bases de données orientées objet, offrant ainsi une solution adaptée pour répondre aux besoins de gestion de données complexes dans différents domaines d'application. Les ORD permettent de représenter les données sous forme d'objets, d'utiliser des types de données définis par l'utilisateur et de bénéficier d'autres fonctionnalités avancées de la programmation orientée objet.

Ce cours nous sera utile dans les cas où nous devons gérer des données complexes qui ne peuvent pas être facilement représentées dans les bases de données relationnelles traditionnelles. En comprenant les principes et les fonctionnalités des ORD, nous serons en mesure de choisir le bon modèle de base de données en fonction de la complexité de nos données et de nos besoins de manipulation. Nous pourrions également exploiter les avantages du modèle relationnel existant tout en tirant parti des fonctionnalités supplémentaires offertes par les ORD.

V. Auto-évaluation

A. Exercice

Vous travaillez en tant que développeur dans une bibliothèque. Vous avez été chargé d'améliorer le système de gestion des données. Le but est de créer une base de données contenant la liste des auteurs, et les livres de ces auteurs.

Question 1

[solution n°1 p.13]

Écrivez un mail à votre responsable (John Doe) pour lui expliquer en quoi un ORDBMS pourrait être une bonne solution. Pensez à lui préciser aussi les points négatifs, pour qu'il puisse prendre la décision finale lui-même. Votre responsable n'est pas technique, donc il va falloir lui présenter les avantages, sans rentrer dans les détails techniques.

Question 2

[solution n°2 p.13]

Créez la base de données sur PostgreSQL.

B. Test

Exercice 1 : Quiz

[solution n°3 p.14]

Question 1

Les ORDBMS combinent les caractéristiques des bases de données :

- ☐ Relationnelles et hiérarchiques
- ☐ Relationnelles et orientées objet
- ☐ Relationnelles et NoSQL

Question 2

Quel avantage offrent les ORDBMS par rapport aux bases de données relationnelles classiques ?

- ☐ La gestion des types de données définis par l'utilisateur
- ☐ Une syntaxe SQL plus simple
- ☐ Un logiciel pour nous passer du SQL

Question 3

Quelle est la différence entre le modèle relationnel et le modèle objet-relationnel ?

- ☐ Le modèle relationnel ne supporte pas les requêtes SQL, tandis que le modèle objet-relationnel le fait
- ☐ Le modèle relationnel utilise des tables, tandis que le modèle objet-relationnel permet d'utiliser des objets pour représenter les données
- ☐ C'est la même chose, il n'y a pas de différence

Question 4

Quel est l'avantage de l'héritage dans les ORDBMS ?

- ☐ Permettre la réutilisation du code
- ☐ Permettre une meilleure encapsulation des données
- ☐ Permettre une meilleure performance des requêtes

Question 5

Quels sont quelques exemples de systèmes de gestion de bases de données objet-relationnelles (ORDBMS) ?

- ☐ MySQL, SQLite
- ☐ PostgreSQL, Oracle (avec extensions OR), IBM DB2
- ☐ MongoDB

Solutions des exercices

p. 11 Solution n°1

Bonjour John,

J'espère que vous allez bien. J'aimerais vous proposer d'envisager l'utilisation d'un système de gestion de base de données objet-relationnelle (ORDBMS) pour améliorer notre système de gestion des données de la bibliothèque.

Un ORDBMS serait une excellente solution pour notre bibliothèque, car il nous permettrait de mieux organiser et gérer les informations sur les auteurs et les livres. Cela signifie que nous pourrions facilement retrouver les livres d'un auteur spécifique et obtenir des informations détaillées sur chaque livre. Cette organisation améliorée nous aidera à mieux répondre aux besoins de nos utilisateurs en matière de recherche et de recommandations.

En utilisant un ORDBMS, nous pourrions également simplifier les processus de mise à jour et de maintenance de la base de données. Les informations sur les auteurs et les livres seront mieux structurées, ce qui réduira les erreurs et facilitera les tâches administratives. De plus, nous pourrions plus facilement suivre les prêts et les retours des livres, améliorant ainsi notre efficacité opérationnelle.

Cependant, il est important de noter que l'utilisation d'un ORDBMS peut nécessiter une formation supplémentaire pour l'équipe et que la complexité de l'ORDBMS peut rendre certaines opérations plus délicates. Je suis disponible pour discuter plus en détail des avantages et des inconvénients de cette proposition afin de vous aider à prendre une décision éclairée.

Pour résumer, on peut dire qu'utiliser un ORDBMS revient à utiliser un outil plus puissant, mais plus lourd.

Je suis disponible pour toute question.

Cordialement,

p. 11 Solution n°2

Sur Windows, vous pouvez installer PostgreSQL à l'aide de l'installateur interactif disponible sur le site officiel de PostgreSQL. Pour installer PostgreSQL, rendez-vous sur le site officiel de PostgreSQL à l'adresse PostgreSQL¹ et téléchargez l'installateur interactif pour Windows.

Suivez les instructions de l'installateur. Au cours de cette installation, vous définirez un mot de passe pour l'utilisateur « *postgres* ». Assurez-vous de le retenir, car vous en aurez besoin pour accéder à la base de données. Une fois l'installation terminée, vous pouvez lancer l'outil « *pgAdmin* » qui vous permettra de gérer vos bases de données via une interface graphique.

Vous pouvez créer la base de données à partir de l'outil pgAdmin ou via l'invite de commandes. Pour créer la base de données via pgAdmin, lancez pgAdmin puis connectez-vous avec l'utilisateur « *postgres* » et le mot de passe que vous avez défini pendant l'installation. Faites un clic droit avec votre souris sur « *Databases* », puis cliquez sur « *Create* » → « *Database...* ». Enfin, donnez un nom à votre base de données, par exemple « *Bibliotheque* », puis cliquez sur « *Save* ».

Maintenant que vous avez créé la base de données, vous pouvez créer les tables. Voici les commandes SQL pour ce faire :

```
1 CREATE TYPE Livre AS (  
2     titre VARCHAR(255),  
3     annee_publication INT,  
4     prix NUMERIC  
5 );  
6  
7 CREATE TABLE Auteur (  
8     id SERIAL PRIMARY KEY,  
9     nom VARCHAR(100) NOT NULL,  
10    prenom VARCHAR(100) NOT NULL,
```

1 <https://www.postgresql.org/download/windows/>

```
11     date_naissance DATE NOT NULL,
12     livres Livre[]
13 );
```

Il est maintenant possible d'ajouter des données à cette table.


```
1 INSERT INTO Auteur (nom, prenom, date_naissance, livres) VALUES
2 ('Hugo', 'Victor', '1802-02-26', ARRAY[('Les Misérables', 1862, 25.00) ::Livre, ('Notre-Dame
3 de Paris', 1831, 20.00) ::Livre]),
4 ('de Saint-Exupéry', 'Antoine', '1900-06-29', ARRAY[('Le Petit Prince', 1943, 15.00) ::Livre,
5 ('Vol de nuit', 1931, 10.00) ::Livre]);
```

Vous pouvez exécuter ces commandes SQL via l'outil pgAdmin ou via l'invite de commandes.

Exercice p. 11 Solution n°3


Question 1

Les ORDBMS combinent les caractéristiques des bases de données :

- ☐ Relationnelles et hiérarchiques
- ☒ Relationnelles et orientées objet
- ☐ Relationnelles et NoSQL
-  Les ORDBMS fusionnent les capacités des bases de données relationnelles et des bases de données orientées objet. Ils offrent un moyen plus flexible de manipuler des données, en particulier des données complexes ou non scalaires.


Question 2

Quel avantage offrent les ORDBMS par rapport aux bases de données relationnelles classiques ?

- ☒ La gestion des types de données définis par l'utilisateur
- ☐ Une syntaxe SQL plus simple
- ☐ Un logiciel pour nous passer du SQL
-  Les ORDBMS offrent la possibilité de définir des types de données personnalisés par l'utilisateur. Cela permet de créer des structures de données plus complexes et adaptées aux besoins spécifiques de l'application.


Question 3

Quelle est la différence entre le modèle relationnel et le modèle objet-relationnel ?

- ☐ Le modèle relationnel ne supporte pas les requêtes SQL, tandis que le modèle objet-relationnel le fait
- ☒ Le modèle relationnel utilise des tables, tandis que le modèle objet-relationnel permet d'utiliser des objets pour représenter les données
- ☐ C'est la même chose, il n'y a pas de différence
-  Le modèle relationnel représente les données sous forme de tables avec des lignes et des colonnes, tandis que le modèle objet-relationnel permet de représenter les données à l'aide d'objets qui peuvent avoir des attributs de types simples ou complexes, y compris d'autres objets.


Question 4

Quel est l'avantage de l'héritage dans les ORDBMS ?

- ☒ Permettre la réutilisation du code
- ☐ Permettre une meilleure encapsulation des données
- ☐ Permettre une meilleure performance des requêtes
-  L'héritage dans les ORDBMS crée une relation hiérarchique entre les classes et permet d'étendre les fonctionnalités d'une classe existante vers une classe enfant. Cela permet la réutilisation du code et facilite la gestion des données.

Question 5

Quels sont quelques exemples de systèmes de gestion de bases de données objet-relationnelles (ORDBMS) ?

- ☐ MySQL, SQLite
- ☒ PostgreSQL, Oracle (avec extensions OR), IBM DB2
- ☐ MongoDB
-  PostgreSQL, Oracle (avec extensions OR) et IBM DB2 sont des exemples de systèmes de gestion de bases de données objet-relationnelles (ORDBMS) largement utilisés.