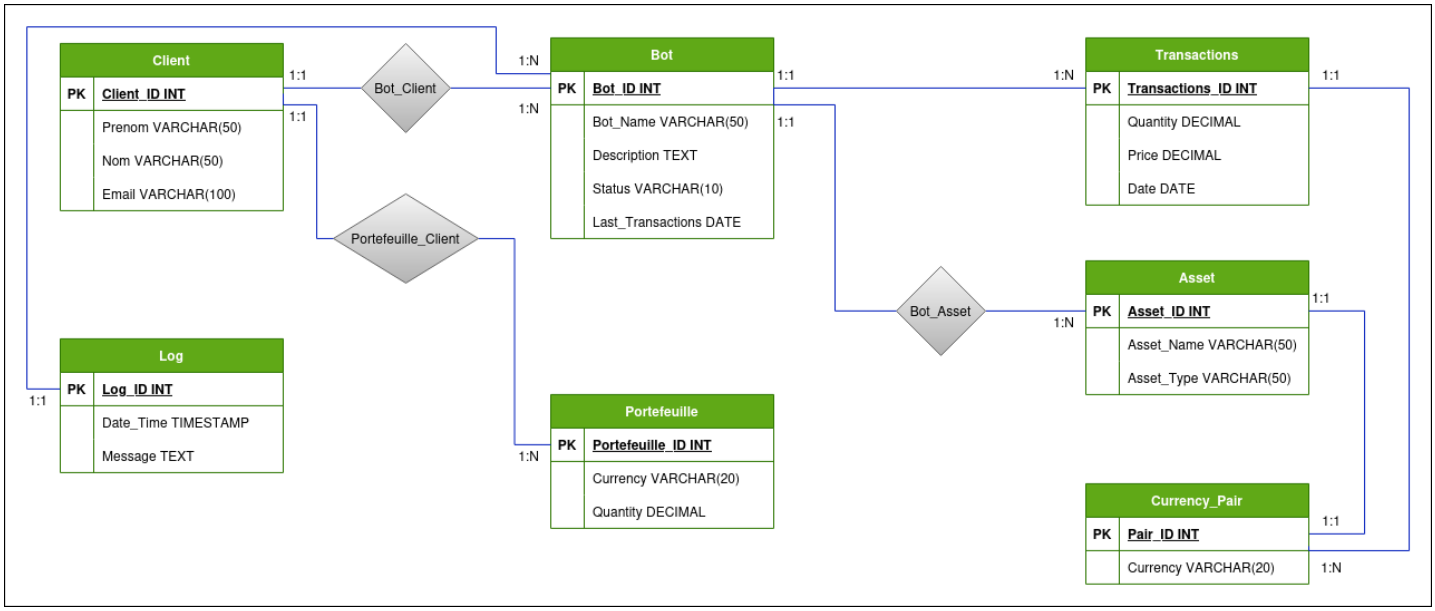
Projet BDD Soufiane **EL KHARMOUDI** 22-03-2023

Ce projet concerne la conception d'une base de données pour une application de trading de crypto-monnaies. Dans un monde où le trading de crypto-monnaies gagne de plus en plus d'importance, cette base de données offre une solution efficace et organisée pour gérer les nombreuses transactions effectuées par des bots de trading en fonction des clients. L'objectif principal de cette base de données est de gérer et suivre avec précision les diverses transactions, tout en assurant une liaison efficace entre les clients, les bots de trading et leurs activités respectives.

Le système propose une interaction entre divers éléments clés tels que les clients, les bots, les transactions, les logs, les portefeuilles et les actifs. Chaque client peut posséder plusieurs bots, qui sont capables de gérer de multiples transactions sur différents actifs. De plus, les bots maintiennent une trace de leurs activités par le biais de logs, garantissant ainsi une transparence totale du processus de trading.

La base de données est conçue pour être flexible et évolutive, capable de gérer une augmentation du nombre de clients, de bots et d'opérations de trading. Sa structure assure également l'intégrité et la cohérence des données, évitant ainsi les redondances et les incohérences. En outre, elle offre la possibilité de faire des requêtes complexes, permettant ainsi d'extraire des informations détaillées et utiles pour l'analyse des performances de trading.

Modèle EA:



Description:

1. `Client` : Cette table stocke les informations sur les clients. Chaque client a un ID unique, un prénom, un nom et une adresse e-mail (qui est également unique).
2. `Bot` : Cette table stocke les informations sur les bots. Chaque bot a un ID unique, un nom, une description, un statut et la date de la dernière transaction.
3. `Bot_Client` : Cette table est une table de relation qui lie les clients et les bots. Elle contient l'ID du bot et l'ID du client.
4. `Transaction` : Cette table stocke les informations sur les transactions. Chaque transaction a un ID unique, l'ID du bot qui a effectué la transaction, l'ID de la paire de devises utilisée pour la transaction, la quantité, le prix et la date de la transaction.
5. `Log` : Cette table stocke les logs. Chaque log a un ID unique, l'ID du bot concerné, la date et l'heure du log, et le message du log.
6. `Portefeuille` : Cette table stocke les informations sur les portefeuilles. Chaque portefeuille a un ID unique, la devise et la quantité.
7. `Portefeuille_Client` : Cette table est une table de relation qui relie les clients et les portefeuilles. Elle contient l'ID du client et l'ID du portefeuille.
8. `Asset` : Cette table stocke les informations sur les actifs. Chaque actif a un ID unique, un nom et un type.
9. `Bot_Asset` : Cette table est une table de relation qui relie les bots et les actifs. Elle contient l'ID du bot et l'ID de l'actif.
10. `CurrencyPair` : Cette table stocke les informations sur les paires de devises. Chaque paire a un ID unique, l'ID de l'actif associé et la devise.

Modèle relationnel:

```
Client(ClientID,Prenom,Nom,Email)

Bot(BotID,BotName,Description,Status,LastTransactions)

Bot_Client(BotID,ClientID)
    BotID ⊆ Bot.BotID
    ClientID ⊆ Client.ClientID

Transaction(TransactionID,BotID,CurrencyPair,Quantity,Price,Date)
    BotID ⊆ Bot.BotID
    CurrencyPair ⊆ CurrencyPair.PairID

Log(LogID,BotID,DateTime,Message)
    BotID ⊆ Bot.BotID

Portefeuille(PortefeuilleID,Currency,Quantity)

Portefeuille_Client(ClientID,PortefeuilleID)
    ClientID ⊆ Client.ClientID
    PortefeuilleID ⊆ Portefeuille.PortefeuilleID

Asset(AssetID,AssetName,AssetType)

Bot_Asset(BotID,AssetID)
    BotID ⊆ Bot.BotID
    AssetID ⊆ Asset.AssetID

CurrencyPair(PairID,AssetID,Currency)
    AssetID ⊆ Asset.AssetID
```

Relation:

1. `Client - Bot_Client`: 1:N - Un client peut posséder plusieurs bots, mais chaque enregistrement dans la table `Bot_Client` se réfère à un seul client.
2. `Bot - Bot_Client`: 1:N - Un bot peut être attribué à plusieurs clients, mais chaque enregistrement dans la table `Bot_Client` se réfère à un seul bot.
3. `Bot - Transaction`: 1:N - Un bot peut réaliser de nombreuses transactions, mais chaque transaction est effectuée par un seul bot.
4. `Bot - Log`: 1:N - Un bot peut générer plusieurs logs, mais chaque log est associé à un seul bot.
5. `Bot - Bot_Asset`: 1:N - Un bot peut être associé à plusieurs actifs, mais chaque enregistrement dans la table `Bot_Asset` se réfère à un seul bot.
6. `Asset - Bot_Asset`: 1:N - Un actif peut être associé à plusieurs bots, mais chaque enregistrement dans la table `Bot_Asset` se réfère à un seul actif.
7. `Client - Portefeuille_Client`: 1:N - Un client peut posséder plusieurs portefeuilles, mais chaque enregistrement dans la table `Portefeuille_Client` se réfère à un seul client.
8. `Portefeuille - Portefeuille_Client`: 1:1 - Un portefeuille peut appartenir à un seul client, et chaque client peut posséder un seul portefeuille. (Si un portefeuille peut appartenir à plusieurs clients, alors cette serait une relation 1:N)
9. `Asset - CurrencyPair`: 1:1 - Chaque actif est associé à une unique paire de devises, et chaque paire de devises est associée à un unique actif.
10. `CurrencyPair - Transaction`: 1:N - Chaque paire de devises peut être utilisée dans de nombreuses transactions, mais chaque transaction utilise une unique paire de devises.

File SQL Create Table / Index / Trigger:

- File_sql / *CreateTable_Index_Trigger.sql*

Index:

Les indices peuvent être utiles pour accélérer les recherches dans la base de données.

```
CREATE INDEX idx_client_name ON Client (Nom);
CREATE INDEX idx_bot_id ON Bot (Bot_ID);
```

Trigger:

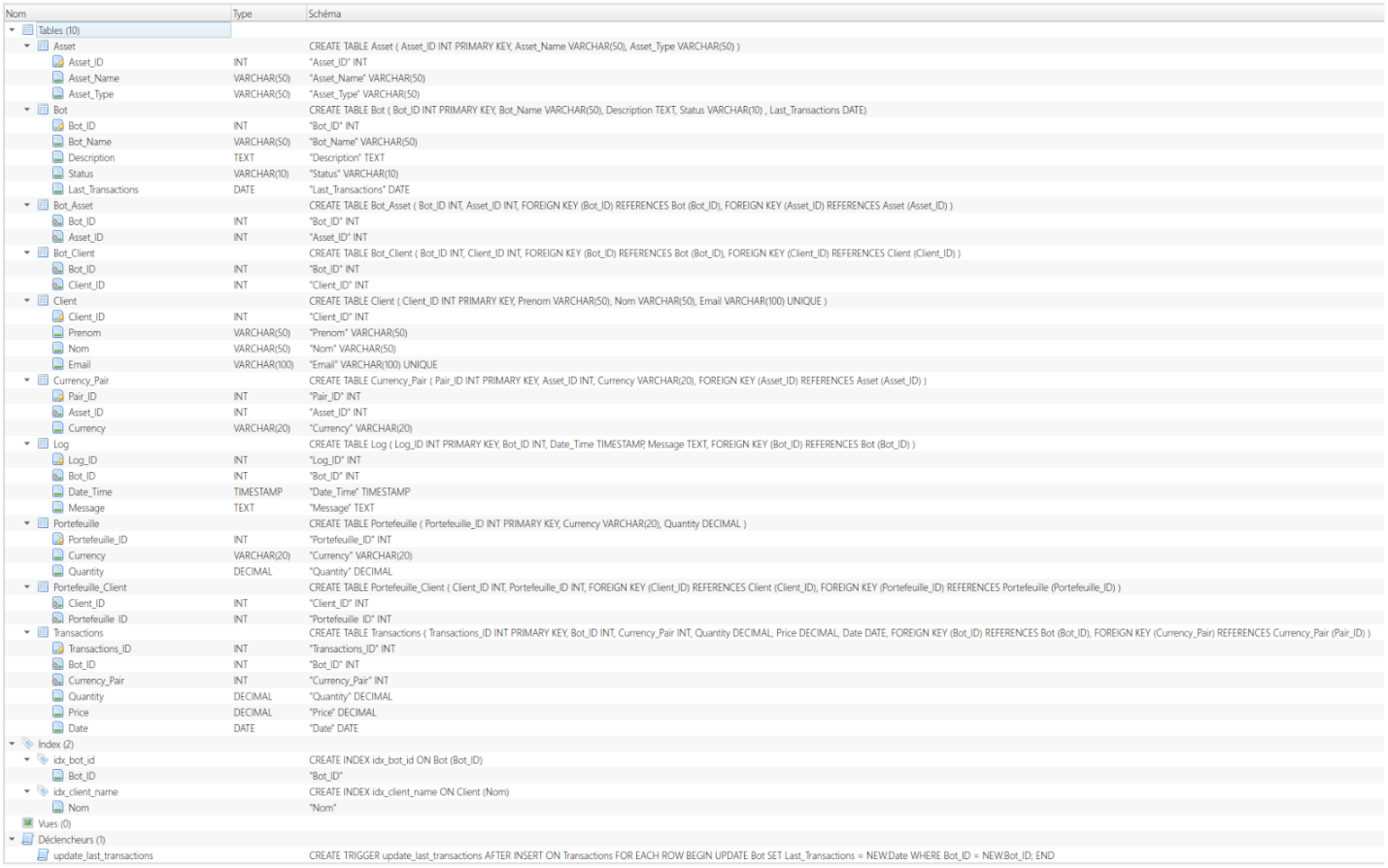
Nous pouvons créer un déclencheur pour mettre à jour le champ 'Last_Transactions' dans la table 'Bot' chaque fois qu'une nouvelle transaction est insérée.

```
CREATE TRIGGER update_last_transactions AFTER INSERT ON Transactions
FOR EACH ROW
BEGIN
    UPDATE Bot
    SET Last_Transactions = NEW.Date
    WHERE Bot_ID = NEW.Bot_ID;
END;
```

File Population Database: *File_sql / Populate.sql*

File Requêtes SQL testées : *File_sql / Requetes.sql*

Database: *Database / projetBOT*



Nom	Type	Schéma
Tables (10)		
Asset		CREATE TABLE Asset (Asset_ID INT PRIMARY KEY, Asset_Name VARCHAR(50), Asset_Type VARCHAR(50))
Asset_ID	INT	"Asset_ID" INT
Asset_Name	VARCHAR(50)	"Asset_Name" VARCHAR(50)
Asset_Type	VARCHAR(50)	"Asset_Type" VARCHAR(50)
Bot		CREATE TABLE Bot (Bot_ID INT PRIMARY KEY, Bot_Name VARCHAR(50), Description TEXT, Status VARCHAR(10) , Last_Transactions DATE)
Bot_ID	INT	"Bot_ID" INT
Bot_Name	VARCHAR(50)	"Bot_Name" VARCHAR(50)
Description	TEXT	"Description" TEXT
Status	VARCHAR(10)	"Status" VARCHAR(10)
Last_Transactions	DATE	"Last_Transactions" DATE
Bot_Asset		CREATE TABLE Bot_Asset (Bot_ID INT, Asset_ID INT, FOREIGN KEY (Bot_ID) REFERENCES Bot (Bot_ID), FOREIGN KEY (Asset_ID) REFERENCES Asset (Asset_ID))
Bot_ID	INT	"Bot_ID" INT
Asset_ID	INT	"Asset_ID" INT
Bot_Client		CREATE TABLE Bot_Client (Bot_ID INT, Client_ID INT, FOREIGN KEY (Bot_ID) REFERENCES Bot (Bot_ID), FOREIGN KEY (Client_ID) REFERENCES Client (Client_ID))
Bot_ID	INT	"Bot_ID" INT
Client_ID	INT	"Client_ID" INT
Client		CREATE TABLE Client (Client_ID INT PRIMARY KEY, Prenom VARCHAR(50), Nom VARCHAR(50), Email VARCHAR(100) UNIQUE)
Client_ID	INT	"Client_ID" INT
Prenom	VARCHAR(50)	"Prenom" VARCHAR(50)
Nom	VARCHAR(50)	"Nom" VARCHAR(50)
Email	VARCHAR(100)	"Email" VARCHAR(100) UNIQUE
Currency_Pair		CREATE TABLE Currency_Pair (Pair_ID INT PRIMARY KEY, Asset_ID INT, Currency VARCHAR(20), FOREIGN KEY (Asset_ID) REFERENCES Asset (Asset_ID))
Pair_ID	INT	"Pair_ID" INT
Asset_ID	INT	"Asset_ID" INT
Currency	VARCHAR(20)	"Currency" VARCHAR(20)
Log		CREATE TABLE Log (Log_ID INT PRIMARY KEY, Bot_ID INT, Date_Time TIMESTAMP, Message TEXT, FOREIGN KEY (Bot_ID) REFERENCES Bot (Bot_ID))
Log_ID	INT	"Log_ID" INT
Bot_ID	INT	"Bot_ID" INT
Date_Time	TIMESTAMP	"Date_Time" TIMESTAMP
Message	TEXT	"Message" TEXT
Portefeuille		CREATE TABLE Portefeuille (Portefeuille_ID INT PRIMARY KEY, Currency VARCHAR(20), Quantity DECIMAL)
Portefeuille_ID	INT	"Portefeuille_ID" INT
Currency	VARCHAR(20)	"Currency" VARCHAR(20)
Quantity	DECIMAL	"Quantity" DECIMAL
Portefeuille_Client		CREATE TABLE Portefeuille_Client (Client_ID INT, Portefeuille_ID INT, FOREIGN KEY (Client_ID) REFERENCES Client (Client_ID), FOREIGN KEY (Portefeuille_ID) REFERENCES Portefeuille (Portefeuille_ID))
Client_ID	INT	"Client_ID" INT
Portefeuille_ID	INT	"Portefeuille_ID" INT
Transactions		CREATE TABLE Transactions (Transactions_ID INT PRIMARY KEY, Bot_ID INT, Currency_Pair INT, Quantity DECIMAL, Price DECIMAL, Date DATE, FOREIGN KEY (Bot_ID) REFERENCES Bot (Bot_ID), FOREIGN KEY (Currency_Pair) REFERENCES Currency_Pair (Pair_ID))
Transactions_ID	INT	"Transactions_ID" INT
Bot_ID	INT	"Bot_ID" INT
Currency_Pair	INT	"Currency_Pair" INT
Quantity	DECIMAL	"Quantity" DECIMAL
Price	DECIMAL	"Price" DECIMAL
Date	DATE	"Date" DATE
Index (2)		
idx_bot_id		CREATE INDEX idx_bot_id ON Bot (Bot_ID)
Bot_ID		"Bot_ID"
idx_client_name		CREATE INDEX idx_client_name ON Client (Nom)
Nom		"Nom"
Vues (0)		
Déclencheurs (1)		
update_last_transactions		CREATE TRIGGER update_last_transactions AFTER INSERT ON Transactions FOR EACH ROW BEGIN UPDATE Bot SET Last_Transactions = NEW.Date WHERE Bot_ID = NEW.Bot_ID; END