**MultiChain** setup while maintaining **a global ledger** that only records transaction summaries instead of full transaction data. Here’s how it works:

**Concept Explanation**

1. **Two Nodes**:
   * **Node 1 (Server/Full Node)**: This node owns an asset and has a wallet.
   * **Node 2 (Client Node)**: This node is connected as a client and interacts with the first node for asset transactions.
2. **Transaction Process**:
   * The first node sends an asset to the second node using wallet transactions.
   * Both nodes maintain **their individual ledgers**.
   * A **global ledger** is maintained, which logs **only the transaction details (e.g., sender, receiver, asset name, timestamp)** but not the full transaction data (such as inputs, outputs, signatures, etc.).
3. **Global Ledger Concept**:
   * The global ledger acts as a **summary ledger**.
   * It stores only the metadata of transactions, reducing storage overhead while maintaining transparency.
   * This approach is useful for audit purposes while keeping asset transfers confidential.

### ****Full Ledger is Not Automatically Shared****

* The **client node does NOT receive a full copy** of the blockchain unless explicitly configured.
* By default, each node **only sees transactions related to its own addresses** unless it has admin or read permissions.

### ****2️ Nodes Maintain Their Own Ledgers Privately****

* Each node **stores its own full copy of the ledger** but **only for transactions it is involved in** (unless explicitly set to receive full chain data).
* This means that Node 1 (Server) will have the full blockchain, but Node 2 (Client) will only have data relevant to its transactions.

### ****3️ Global Ledger (Optional, Summary Only)****

* Instead of syncing the entire blockchain across all nodes, a **separate global ledger** is maintained to track only transaction summaries (not full transaction data).
* This global ledger contains:
  + Transaction ID (TXID)
  + Sender and Receiver Addresses
  + Asset Name
  + Amount Transferred
  + Timestamp

### ****🔄 Example: What Happens When a Client Node Connects?****

#### Scenario:

* **Node 1 (Server)** issues an asset.
* **Node 2 (Client)** connects and makes a transaction.
* The **full transaction history stays on Node 1**.
* **Node 2 only sees transactions related to its own addresses**.
* The **global ledger logs only transaction metadata**, not full blockchain data.

### ****📝 What Gets Stored?****

| **Data Type** | **Stored in Node 1?** | **Stored in Node 2?** | **Stored in Global Ledger?** |
| --- | --- | --- | --- |
| Full Blockchain | ✅ Yes | ❌ No (Only relevant transactions) | ❌ No |
| Full Transaction Data | ✅ Yes | ❌ No | ❌ No |
| Wallet Transactions | ✅ Yes | ✅ Yes (Only its own) | ❌ No |
| Transaction Summary | ✅ Yes | ✅ Yes | ✅ Yes |

### ****🔍 Conclusion****

✔ **Each node keeps its own private ledger.**  
✔ **A client node does NOT receive the full blockchain unless explicitly allowed.**  
✔ **The global ledger is a manually maintained summary of transactions, NOT a full blockchain copy.**  
✔ **This improves privacy while allowing audits of transaction summaries.**

Would you like to implement a script that automates global ledger updates across nodes? 🚀

**Lab Question**

In a MultiChain setup, three different addresses (wallets) perform asset transactions among themselves. Each address maintains its **own transaction history** privately. However, a **global blockchain ledger** is required that allows all three addresses to see an **overall transaction summary** without revealing complete details of individual wallets.

**Task:**

1. **Set up a MultiChain Network** with a single-node blockchain.
2. **Create three different addresses (wallets).**
3. **Perform transactions** where assets are sent from one address to another.
4. **Maintain a private ledger** for each address, containing only transactions it is involved in.
5. **Design a global ledger system** that:
   * Records **only transaction summaries** (Sender, Receiver, Amount, Timestamp).
   * Allows each address to query the **global transaction history** without exposing full private details.
6. **Write a MultiChain script** to automate the process of updating the global ledger in real-time.

💡 **Bonus Challenge:** Ensure that each wallet can view the transaction history of others **without accessing private keys**.