Data Model and Storage Solution Using Data Vault 2.0 Methodology

1. Technology Stack To handle the scale, complexity, and real-time requirements of the proposed architecture, I recommend using the following technologies:

Data Storage:
Azure Data Lake Storage (ADLS) for storing raw and processed data.
Synapse Analytics (or Azure SQL Database) for structured data storage.
Data Processing:
Azure Synapse Pipelines and Azure Databricks for ETL/ELT processes.
Data Ingestion and Real-Time Processing:
Azure Event Hubs or Kafka for event streaming, followed by Stream Analytics
or Azure Databricks for real-time data transformation.
Data Modeling:
Implement Data Vault 2.0 using Synapse SQL or Databricks Delta Lake tables.

2. Data Vault 2.0 Table Structure

Data Vault 2.0 methodology splits data into three primary components:

Hubs, Links and Satellites.

a. Hubs

Hubs are the core business entities identified by unique keys. Each Hub contains:
- *Primary Key* (Business Key): Unique identifier for the entity (**PII data will be hashed)*
- *Load Date*: Timestamp when the data was loaded.
- *Record Source*: Source of the data.
We create Hubs for Player, Game, and Event

Hub_Player

Player_ID: INT --- The unique identifier for the player (mapped from uid in events).
Load_Date: DATETIME --- The timestamp when the data was loaded into the Hub.
Record_Source: VARCHAR --- The source of the data, Ex:"auth_event", "spin_event".

Hub_Game

Game_ID: VARCHAR --- The unique identifier for the game/application (ex: "app_3").
Load_Date: DATETIME --- The timestamp when the data was loaded into the Hub.
Record_Source: VARCHAR --- The source of the data.

Hub_Event

Event_ID: INT --- The unique identifier for the event (msg_id from each event).
Event_Type: VARCHAR --- The type of event, Ex:"auth_event", "spin_event", "purchase_event".
Load_Date: DATETIME --- The timestamp when the event was loaded into the Hub.
Record_Source: VARCHAR --- The source of the data.

b. Links

Links establish relationships between hubs. We define Links to establish connections between players, games, and events. Each link table contains:
- *Primary Key*: Composite key made up of the foreign keys from the linked hubs.
- *Load Date*
- *Record Source*

Link_Player_Game

Player_ID: INT --- Foreign key to Hub_Player.
Game_ID: VARCHAR--- Foreign key to Hub_Game.
Event_ID: INT --- Foreign key to Hub_Event (for linking specific events to this relationship).
Load_Date: DATETIME --- The timestamp when the link was created.
Record_Source: VARCHAR--- The source of the data.

Link_Player_Event

Player_ID: INT --- Foreign key to Hub_Player.
Event_ID: INT --- Foreign key to Hub_Event.
Load_Date: DATETIME --- The timestamp when the link was created.
Record_Source: VARCHAR--- The source of the data.

Link_Game_Event

Game_ID: VARCHAR--- Foreign key to Hub_Game.
Event_ID: INT --- Foreign key to Hub_Event.
Load_Date: DATETIME --- The timestamp when the link was created.
Record_Source: VARCHAR--- The source of the data.

c. Satellites

Satellites store descriptive information related to hubs or links and they may include SCD type 2 (attributes that may change over time)

Examples:

Satellite_Player_Profile

Player_ID: INT --- Foreign key to Hub_Player.
Email: VARCHAR--- The player's email address (PII, encrypted or hashed).
Phone: VARCHAR--- The player's phone number (PII, encrypted or hashed, or null if unavailable).
Load_Date: DATETIME --- The timestamp when the data was loaded.
Record_Source: VARCHAR--- The source of the data (Ex:"auth_event").

Satellite_Event_Details

Event_ID: INT --- Foreign key to Hub_Event.
Publish_Timestamp: DATETIME --- The timestamp when the event was published (publish_ts from the event).
Event_Type: VARCHAR--- The type of event, Ex:"auth_event", "spin_event", "purchase_event".
Load_Date: DATETIME --- The timestamp when the data was loaded.
Record_Source: VARCHAR--- The source of the data.

Satellite_Auth_Event

Event_ID: INT --- Foreign key to Hub_Event.
UID: INT --- User identifier for linking purposes (uid from auth_msg).
App: VARCHAR--- The app associated with the event (Ex:"app_3").
Email: VARCHAR--- The email address, encrypted or hashed.
Phone: VARCHAR--- The phone number, encrypted or hashed, or null if unavailable.
Load_Date: DATETIME --- The timestamp when the data was loaded.
Record_Source: VARCHAR--- The source of the data.

Satellite_Spin_Event

Event_ID: INT --- Foreign key to Hub_Event.
UID: INT --- User identifier (uid from spins_msg).
Spin_Amount: INT --- The amount or number of spins (spin from spins_msg).
App: VARCHAR--- The app where the spin occurred (Ex:"app_3").
Load_Date: DATETIME --- The timestamp when the data was loaded.
Record_Source: VARCHAR--- The source of the data.

Satellite_Purchase_Event

Event_ID: INT --- Foreign key to Hub_Event.
UID: INT --- User identifier (uid from purchase_msg).
Amount: DECIMAL(10, 2) --- The purchase amount (amount from purchase_msg).
App: VARCHAR--- The app where the purchase occurred (Ex:"app_3").
Load_Date: DATETIME --- The timestamp when the data was loaded.
Record_Source: VARCHAR--- The source of the data

Event Mapping to Data Vault Model

Message	*Example 1*

auth_msg	<pre>{
	  "msg_id": 124,
	  "publish_ts":
	"2024-10-12T14:00:00",
	  "type": "auth_event",
	  "payload": {
	  "uid": 453135,
	  "email": "SomeEmail@test.com",
	  "phone": null,
	  "app": "app_3"}
	}</pre>

spins_msg	<pre>{
	  "msg_id": 1275,
	  "publish_ts":
	"2024-10-12T14:02:00",
	  "type": "spin_event",
	  "payload": {
	  "uid": 125331,
	  "spin": 1400,
	  "app": "app_3"}
	}</pre>

purchase_msg	<pre>{
	  "msg_id": 2112,
	  "publish_ts":
	"2024-10-12T17:09:00",
	  "type": "purchase_event",
	  "payload": {
	  "uid": 124442,
	  "amount": 1499,
	  "app": "app_3"}
	}</pre>

auth_msg:

msg_id: 124 → Event_ID in Hub_Event.
publish_ts: "2024-10-12T14:00:00" → Publish_Timestamp in Satellite_Event_Details.
type: "auth_event" → Event_Type in Satellite_Event_Details.
uid: 453135 → Player_ID in Hub_Player.
email: "SomeEmail@test.com" → Email in Satellite_Player_Profile.
phone: null → Phone in Satellite_Player_Profile.
app: "app_3" → App in various tables.

spins_msg:

msg_id: 1275 → Event_ID in Hub_Event.
publish_ts: "2024-10-12T14:02:00" → Publish_Timestamp in Satellite_Event_Details.
type: "spin_event" → Event_Type in Satellite_Event_Details.
uid: 125331 → Player_ID in Hub_Player.
spin: 1400 → Spin_Amount in Satellite_Spin_Event.
app: "app_3" → App in various tables.

purchase_msg:

msg_id: 2112 → Event_ID in Hub_Event.
publish_ts: "2024-10-12T17:09:00" → Publish_Timestamp in Satellite_Event_Details.
type: "purchase_event" → Event_Type in Satellite_Event_Details.
uid: 124442 → Player_ID in Hub_Player.
amount: 1499 → Amount in Satellite_Purchase_Event.
app: "app_3" → App in various tables.

3. Additional Components

Data Ingestion Framework:
Implement a framework using Azure Event Hubs or Kafka for ingesting JSON events into the data lake.

Data Processing and Transformation:
Azure Synapse Pipelines or Azure Dtabricks to handle ETL/ELT, batch processing, and stream processing.

Data Governance and Security:
Use Azure Purview for data cataloging and governance. Implement encryption for PII data using Azure Key Vault.

Reporting and Analytics: Use Power BI to build dashboards and reports.

Data Monitoring & Alerts: Set up monitoring and alerting using Azure Monitor or Application Insights for real-time health checks and performance tracking.

4. Storage Format

Raw Data Layer: Store raw JSON events as they arrive in Azure Data Lake Storage

Structured Data Layer:
Use Delta Lake or Parquet format for processed data in Azure Synapse for optimized querying.

PII Management: Store encrypted or hashed PII data and manage encryption keys using Azure Key Vault.