Solution Based on Data Vault 2.0 Methodology

Overview of Data Vault 2.0

Data Vault 2.0 organizes the data warehouse into three main components:

- **Hubs**: Store unique business keys (e.g., Player ID, Game ID).
- **Links**: Define relationships between hubs (e.g., Player-Game relationships).
- **Satellites**: Store descriptive data (e.g., player activity, game metadata).

This approach provides flexibility, scalability, and auditability, which is crucial for an event-driven architecture and analytics requirements.

---

1. Table Structures

1.1 Hubs

Hubs store unique business keys and metadata. They represent the core entities in the system.

Hub_Player

Column Name	Data Type	Description
Player_ID	VARCHAR(36)	Unique identifier for the player (UUID).
Load_DTS	TIMESTAMP	Timestamp when the record was loaded.
Record_Source	VARCHAR(50)	Source system of the record.

Hub_Game

Column Name	Data Type	Description
Game_ID	VARCHAR(36)	Unique identifier for the game.
Load_DTS	TIMESTAMP	Timestamp when the record was loaded.
Record_Source	VARCHAR(50)	Source system of the record.

1.2 Links

Links define relationships between hubs, capturing the associations between entities.

Link_Player_Game

Column Name	Data Type	Description
Link_ID	VARCHAR(36)	Unique identifier for the link (UUID).
Player_ID	VARCHAR(36)	Foreign key referencing Hub_Player.
Game_ID	VARCHAR(36)	Foreign key referencing Hub_Game.

Load_DTS	TIMESTAMP	Timestamp when the record was loaded.
Record_Source	VARCHAR(50)	Source system of the record.

1.3 Satellites

Satellites store descriptive information and context for hubs or links. They allow tracking changes over time.

Satellite_Player_Purchase

Column Name	Data Type	Description
Player_ID	VARCHAR(36)	Foreign key referencing Hub_Player.
Purchase_ID	VARCHAR(36)	Unique identifier for the purchase.
Purchase_Amount	DECIMAL(10, 2)	Amount of the purchase.
Currency	VARCHAR(10)	Currency of the purchase.
Event_Timestamp	TIMESTAMP	Timestamp of the purchase event.
Load_DTS	TIMESTAMP	Timestamp when the record was loaded.
Record_Source	VARCHAR(50)	Source system of the record.

Satellite_Game_Spin

Column Name	Data Type	Description
Game_ID	VARCHAR(36)	Foreign key referencing Hub_Game.
Spin_ID	VARCHAR(36)	Unique identifier for the spin.
Spin_Amount	DECIMAL(10, 2)	Amount bet in the spin.
Win_Amount	DECIMAL(10, 2)	Amount won in the spin.
Event_Timestamp	TIMESTAMP	Timestamp of the spin event.
Load_DTS	TIMESTAMP	Timestamp when the record was loaded.
Record_Source	VARCHAR(50)	Source system of the record.

Satellite_Player_Game_Time

Column Name	Data Type	Description
Link_ID	VARCHAR(36)	Foreign key referencing Link_Player_Game.
Session_Start	TIMESTAMP	Start time of the session.
Session_End	TIMESTAMP	End time of the session.
Total_Time	INTERVAL	Total time spent in the session.

Load_DTS	TIMESTAMP	Timestamp when the record was loaded.
Record_Source	VARCHAR(50)	Source system of the record.

---

2. Storage Format

2.1 Raw Event Storage

- **Format**: JSON, stored in AWS S3 (or equivalent) for immutable storage.
- **Structure**: Events are organized by service and date, e.g., s3://events/authorization/2025-01-01/.

2.2 Processed Data in Data Warehouse

- **Format**: Tables stored in Snowflake (or equivalent data warehouse).
- **Schema**: Data Vault schema separates raw, business vault, and reporting layers:
    - **Raw Vault**: Contains hubs, links, and satellites in normalized form.
    - **Business Vault**: Contains derived tables for specific metrics.
    - **Reporting Layer**: Aggregated views for BI tools.

---

3. Additional Components

3.1 ETL/ELT Pipelines

- **Tools**: dbt (Data Build Tool) for ELT and SQL-based transformations.
- **Process**:
    1. Extract events from Kafka or S3.
    2. Load raw data into staging tables.
    3. Transform data into hubs, links, and satellites.
    4. Load business vault and reporting layer tables.

3.2 Data Quality and Monitoring

- **Tools**: Great Expectations for automated data quality checks.
- **Features**: Ensure schema consistency, validate key relationships, and track data freshness.

3.3 BI Integration

- **Tools**: Tableau, Power BI, or Looker for reporting and visualization.
- **Example Reports**:
    - Average purchase per player.
    - Game with the most time spent by players.

- Trends in spin amounts over time.

---

4. Mapping Message Examples to Data Vault Tables

Example Message 1: auth_msg

Mapping:

- auth_msg.payload.uid populates Hub_Player.Player_ID.
- auth_msg.payload.app populates Hub_Game.Game_ID.

Example Message 2: spins_msg

Mapping:

- spins_msg.payload.uid populates Hub_Player.Player_ID.
- spins_msg.payload.spin populates Satellite_Game_Spin.Spin_Amount.
- spins_msg.payload.app populates Hub_Game.Game_ID.

Example Message 3: purchase_msg

Mapping:

- purchase_msg.payload.uid populates Hub_Player.Player_ID.
- purchase_msg.payload.amount populates Satellite_Player_Purchase.Purchase_Amount.
- purchase_msg.payload.app populates Hub_Game.Game_ID.

Summary

This Data Vault 2.0 implementation offers a robust, scalable, and auditable solution for aggregating, processing, and analyzing event-driven data from multiple applications. The combination of hubs, links, and satellites ensures flexibility in adapting to future changes while maintaining a clear lineage and history of data changes.

Data Flow

The **data flow** in the context of the provided architecture and the Data Vault 2.0 methodology can be summarized as follows:

---

1. Event Generation and Collection

- **Source**: Events are generated by various backend services (auth, spins, purchase) for different games and players.

- **Event Bus**: These events (e.g., auth_msg, spins_msg, purchase_msg) are published as structured JSON messages to the event bus, which acts as a central communication pipeline for all events.
- **Storage**: Raw JSON events are stored in a cloud storage solution like AWS S3 for immutable archiving.

---

2. ETL/ELT Pipelines

- **Extraction**:
    - Events are continuously ingested from the event bus or cloud storage into a staging area in the data warehouse (e.g., Snowflake).
    - Tools like Kafka, Kinesis, or a similar data streaming platform can facilitate real-time ingestion.
- **Loading (Raw Vault)**:
    - Events are parsed and transformed into their corresponding **Hubs**, **Links**, and **Satellites** based on the Data Vault 2.0 schema.
    - The staging area is used to clean and validate data before populating the raw vault tables:
        - Unique business keys (e.g., Player_ID, Game_ID) are extracted into **Hubs**.
        - Relationships (e.g., Player-Game) are recorded in **Links**.
        - Event details (e.g., spins, purchases) are added to **Satellites**.
- **Transformations (Business Vault)**:
    - Business logic is applied to derive metrics such as "average purchase per player" or "time spent per game."
    - Derived tables are created for specific use cases (e.g., aggregated trends over time).

---

3. Data Storage

- **Raw Vault**: Stores normalized tables (hubs, links, and satellites) for all ingested events.
- **Business Vault**: Contains transformed and derived data for specific business needs.
- **Reporting Layer**: Aggregated and denormalized views are created for easy consumption by BI tools.

---

4. Reporting and Analytics

- **BI Tools**:
    - Data from the reporting layer is consumed by tools like Tableau, Power BI, or Looker.

- Examples of reports include:
    - Average purchase per player across all applications.
    - Games with the highest engagement by players.
    - Spin trends over time.

- **Ad-hoc Queries**:
    - Analysts can query raw or derived data directly for exploratory analytics or operational needs.

Example of the Flow with a Message

Input: A purchase_msg


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


Flow:

1. **Event Bus**:
    - The purchase_msg is published to the event bus.
    - The raw event is stored in cloud storage (e.g., S3) for archival.
2. **Raw Vault**:
    - The uid (124442) is used to populate Hub_Player.
    - The app (app_3) is used to populate Hub_Game.
    - The event relationship is recorded in Link_Player_Game.
    - The purchase details (amount: 1499) are stored in Satellite_Player_Purchase.
3. **Business Vault**:
    - Metrics like "total purchases by player" or "average purchase amount by game" are calculated and stored in derived tables.

4. **Reporting Layer**:
    - A BI tool visualizes the total revenue per player or trends in purchase amounts over time.

Advantages of this Flow

1. **Scalability**: Can handle large volumes of event data with minimal schema changes.
2. **Auditability**: Historical records and lineage of all changes are preserved in the raw vault.
3. **Flexibility**: New business metrics or reports can be added without redesigning the core data model.
4. **Real-time Analytics**: Integration with real-time data streaming platforms enables up-to-date reporting.