**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.