HRG Data Engineer Test Assignment

Task

As part of this test assignment, you need to develop and describe the data model structure and storage format for the specified data based on the Data Vault 2.0 methodology.

Technologies Used

To implement the Data Vault 2.0 storage solution, the following technologies will be used:

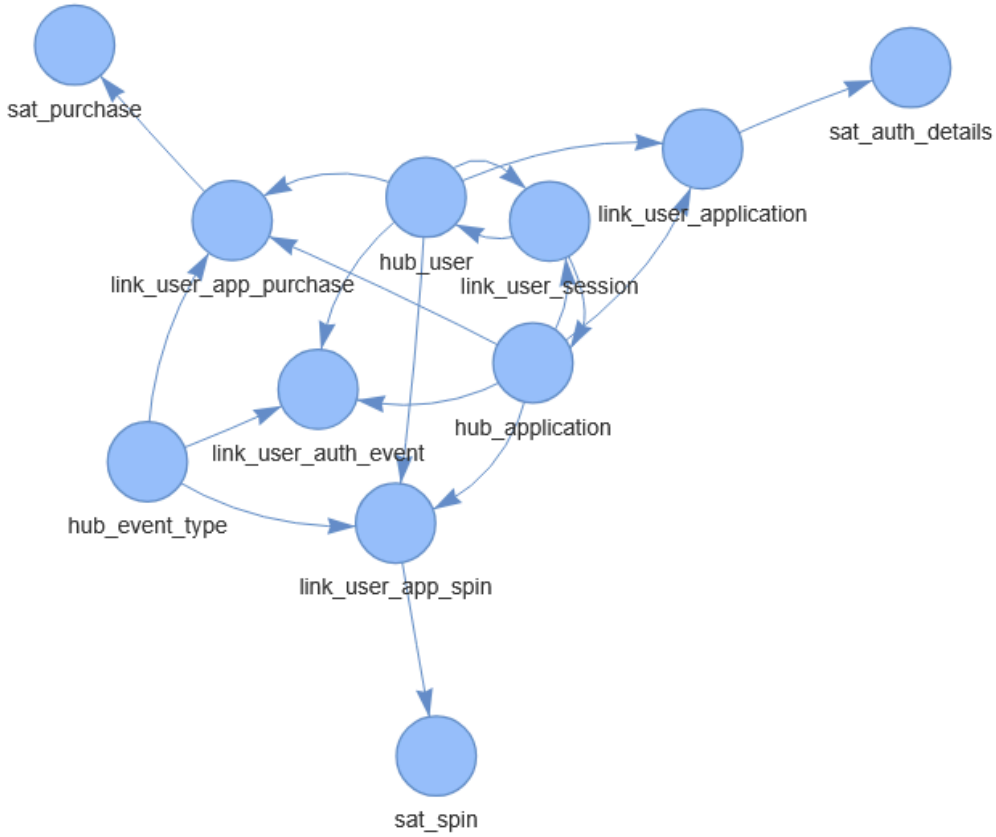
- **Storage**: BigQuery for scalable data warehousing.
- **Event Ingestion**: Kafka to capture real-time events.
- **ETL Processing**: dbt (Data Build Tool) for data transformations.
- **Orchestration**: Apache Airflow or dbt schduler for scheduling data pipelines.
- **Monitoring**: Grafana for system observability.

Data Vault Model

Hubs represent core business concepts, links represent relationships between hubs, and satellites store information about hubs and relationships between them.

ERD Diagram :

Diagram shows the relationship between hubs, links and satellites.



1. Hubs (Unique Business Keys)

Hub_user

- User hub would have a generated key uuid which is unique to the table user_hk, and a unique identifier for the user as user_id, load_dts is when the record is first loaded and record_src is source table info.

```
CREATE TABLE hub_user (
    user_hk STRING PRIMARY KEY,
    user_id STRING,
    load_dts TIMESTAMP,
    record_src STRING
);
```

Hub_application

- Application hub would have a generated uuid which is unique to the table app_hk, and a unique identifier for the user as app_id,, load_dts is when the record is first loaded and record_src is source table info.

```
CREATE TABLE hub_application (
    app_hk STRING PRIMARY KEY,
    app_id STRING,
    load_dts TIMESTAMP,
    record_src STRING
);
```

Hub_event_type

- event_type hub would have a generated uuid which is unique to the table event_type_hk, and a unique identifier for the user as event_type,, load_dts is when the record is first loaded and record_src is source table info.

```
CREATE TABLE hub_event_type (
    event_type_hk STRING PRIMARY KEY,
    event_type STRING,
    load_dts TIMESTAMP,
    record_src STRING
);
```

2. Links (Event-Based Relationships)

Link_user_application

- This table captures the relationship between users and applications. It links users to the applications they interact with, enabling analysis of user-application interactions.
- User_app_lk is a surrogate key that uniquely identifies each user-app relationship, additionally user_hk and app_hk are the foreign keys coming from the hubs.

```
CREATE TABLE link_user_application (
    user_app_lk STRING PRIMARY KEY,
    user_hk STRING,
    app_hk STRING,
    load_dts TIMESTAMP,
    record_src STRING,
```

```
    CONSTRAINT unique_user_app UNIQUE (user_hk, app_hk)
);

ALTER TABLE link_user_application
ADD CONSTRAINT fk_user_hk FOREIGN KEY (user_hk) REFERENCES hub_user(user_hk)
ON DELETE CASCADE,
ADD CONSTRAINT fk_app_hk FOREIGN KEY (app_hk) REFERENCES
hub_application(app_hk) ON DELETE CASCADE;
```

Link_user_app_purchase

- This table models the relationship between users, applications, and purchases. It allows us to track which users made purchases within specific applications.
- Purchase_lk is a surrogate key user_hk + app_hk + msg_id, and user_hk, app_kh are the foreign_key from the hub

```
CREATE TABLE link_user_app_purchase (
    purchase_lk STRING PRIMARY KEY,
    user_hk STRING,
    app_hk STRING,
    event_type_hk STRING,
    msg_id STRING,
    load_dts TIMESTAMP,
    record_src STRING,

    CONSTRAINT unique_user_app UNIQUE (user_hk, app_hk, msg_id)

);

ALTER TABLE link_user_app_purchase
ADD CONSTRAINT fk_user_hk FOREIGN KEY (user_hk) REFERENCES hub_user(user_hk)
ON DELETE CASCADE,
ADD CONSTRAINT fk_app_hk FOREIGN KEY (app_hk) REFERENCES
hub_application(app_hk) ON DELETE CASCADE,
ADD CONSTRAINT fk_event_type_hk FOREIGN KEY (event_type_hk) REFERENCES
hub_event_type(event_type_hk) ON DELETE CASCADE;
```

Link_user_app_spin

- This table represents the relationship between users, applications, and spin activities. It allows us to track spin events (e.g., spins in a game or a lottery) within applications by associating each spin activity with a specific user and application.
- spin_lk is a surrogate key user_hk + app_hk + msg_id, and user_hk, app_kh are the foreign_key from the hub

```
CREATE TABLE link_user_app_spin (
    spin_lk STRING PRIMARY KEY,
    user_hk STRING,
    app_hk STRING,
    event_type_hk STRING,
    msg_id STRING,
    load_dts TIMESTAMP,
    record_src STRING,

    CONSTRAINT unique_user_app UNIQUE (user_hk, app_hk, msg_id)
);

ALTER TABLE link_user_app_spin
ADD CONSTRAINT fk_user_hk FOREIGN KEY (user_hk) REFERENCES
hub_user(user_hk) ON DELETE CASCADE,
ADD CONSTRAINT fk_app_hk FOREIGN KEY (app_hk) REFERENCES
hub_application(app_hk) ON DELETE CASCADE,
ADD CONSTRAINT fk_event_type_hk FOREIGN KEY (event_type_hk) REFERENCES
hub_event_type(event_type_hk) ON DELETE CASCADE;
```

Link_user_auth_event

- This table captures the relationship between users, applications, and authentication events. It tracks the events where users authenticate into an application, such as login attempts.
- auth_lk is a surrogate key user_hk + app_hk + msg_id, and user_hk, app_kh are the foreign_key from the hub

```
CREATE TABLE link_user_auth_event (
    auth_lk STRING PRIMARY KEY,
    user_hk STRING,
    app_hk STRING,
    event_type_hk STRING,
    msg_id STRING,
    load_dts TIMESTAMP,
    record_src STRING,
```

```
    CONSTRAINT unique_user_app UNIQUE (user_hk, app_hk, msg_id)
);

ALTER TABLE link_user_auth_event ADD CONSTRAINT fk_user_hk FOREIGN KEY
(user_hk) REFERENCES hub_user(user_hk) ON DELETE CASCADE,
ADD CONSTRAINT fk_app_hk FOREIGN KEY (app_hk) REFERENCES
hub_application(app_hk) ON DELETE CASCADE,
ADD CONSTRAINT fk_event_type_hk FOREIGN KEY (event_type_hk) REFERENCES
hub_event_type(event_type_hk) ON DELETE CASCADE;
```

Link_user_session

- This table models the relationship between users, applications, and user sessions. It helps track when users are active in a session within an application, such as during an active browsing or gameplay session.
- user_session_lk is a surrogate key user_hk + app_hk + session_id, and user_hk, app_kh are the foreign_key from the hub

```
CREATE TABLE link_user_session (
    user_session_lk STRING PRIMARY KEY,
    user_hk STRING,
    app_hk STRING,
    session_id STRING,
    session_start TIMESTAMP,
    session_end TIMESTAMP,
    load_dts TIMESTAMP,
    record_src STRING,

    CONSTRAINT unique_user_app UNIQUE (user_hk, app_hk, session_id)
);

ALTER TABLE link_user_session
ADD CONSTRAINT fk_user_hk FOREIGN KEY (user_hk) REFERENCES
hub_user(user_hk) ON DELETE CASCADE,
ADD CONSTRAINT fk_app_hk FOREIGN KEY (app_hk) REFERENCES
hub_application(app_hk) ON DELETE CASCADE;
```

3. Satellites (Descriptive Attributes)

sat_auth_details

```
CREATE TABLE sat_auth_details (
    auth_lk STRING,
    email STRING,
    phone STRING,
    publish_ts TIMESTAMP,
    load_dts TIMESTAMP,
    record_src STRING

    CONSTRAINT fk_auth_event_lk FOREIGN KEY (auth_lk) REFERENCES
link_user_auth_event(auth_lk) ON DELETE CASCADE
);
```

sat_spin

```
CREATE TABLE sat_spin (
    spin_lk STRING PRIMARY KEY,
    spin_value INT,
    publish_ts TIMESTAMP,
    load_dts TIMESTAMP,
    record_src STRING,
    CONSTRAINT fk_spin_lk FOREIGN KEY (spin_lk) REFERENCES
link_user_app_spin(spin_lk) ON DELETE CASCADE
);
```

sat_purchase

```
CREATE TABLE sat_purchase (
    purchase_lk STRING,
    amount FLOAT,
    publish_ts TIMESTAMP,
    load_dts TIMESTAMP,
    record_src STRING,
    CONSTRAINT fk_purchase_event_lk FOREIGN KEY (purchase_lk) REFERENCES
link_user_purchase_event(purchase_lk) ON DELETE CASCADE
);
```

Business Marts for Analytical Queries

To answer business questions, the following marts are created:

1. Average Purchase Per Player Across All Applications

```
CREATE TABLE mart_avg_purchase AS

SELECT
    u.user_id,
    a.app_id,
    AVG(p.amount) AS avg_purchase
FROM hub_user u JOIN link_user_app_purchase lp ON u.user_hk = lp.user_hk
JOIN sat_purchase p ON lp.purchase_lk = p.purchase_lk
JOIN hub_application a ON lp.app_hk = a.app_hk
GROUP BY u.user_id, a.app_id;
```

2. Game a Player Spends the Most Time On

- Considering that the time taken is across all platforms we would have to group by the user and app_id to get the relevant total timespent overall.

```
CREATE TABLE mart_most_played_game AS
SELECT
    u.user_id,
    a.app_id,
    SUM(TIMESTAMP_DIFF(s.session_end, s.session_start, SECOND)) AS
total_play_time_seconds
FROM link_user_session s
JOIN hub_user u ON s.user_hk = u.user_hk
JOIN hub_application a ON s.app_hk = a.app_hk
GROUP BY u.user_id, a.app_id
ORDER BY total_play_time_seconds DESC;
```

3. How the Average Spin Changes Over Time

- Average spins based on the published date calculated from sat_spin

```
CREATE TABLE mart_avg_spin_trends AS
SELECT
    DATE(s.publish_ts) AS spin_date,
    AVG(s.spin_value) AS avg_spin
FROM sat_spin s
GROUP BY spin_date
ORDER BY spin_date;
```

4. User Profile with PII Data Across All Games

- User profiles across all games can be fetch using the sat_auth_details containing the PII information.

```
CREATE TABLE mart_user_profiles AS
SELECT
    u.user_id,
    a.app_id,
    ua.email,
    ua.phone
FROM sat_auth_details ua
JOIN hub_user u ON ua.auth_lk = u.user_hk
JOIN link_user_application lua ON u.user_hk = lua.user_hk
JOIN hub_application a ON lua.app_hk = a.app_hk;
```

Components needed to develop:

- In order for the solution to be fully developed there are several considerations to make.
- ETL Pipelines (with CDC Data Stream to stream data fed to source systems to BQ landing)
- Data Quality & Monitoring (Testing standards like unique, foreign, not null checks)
- Data Lineage & Metadata Management (ensure the lineage is readable and meta data is stored)
- Business Data Marts and Reporting (Reporting data marts)
- Data Governance & Security (Restricted access to views generated from data marts, sensitive PII data mask like personal information)
- Error Notifications & Alerts (Notification in case of errors in every and all stages)
- Interactive UI for Monitoring (Dashboards to monitor the health of systems)