Stored programs pseudocode

parse_event Procedure

1. Initialize variables for event attributes (e.g., event type, account ID, event ID, etc.)
2. Determine the maximum data feed ID from **SM_data_feed** table.
3. Iterate over each row in the **SM_data_feed** table using a while loop.
    - For each iteration, retrieve event-related data (e.g., SMUID, event ID, event category codes, event time, hashtags, content) for the current **data_feed_id**.
    - Parse the event category codes (**evcatcodes**) to extract two category values, adjusting for out-of-range values.
    - Format **event_time** to a proper **DATETIME** format or set a default value if null.
    - Use a **CASE** statement to decide which upsert procedure to call based on the event category (**evcat1**):
        - If **evcat1** is 0 or 4, call **UpsertSocialEvent**.
        - If **evcat1** is between 1 and 3, call **UpsertDeviceLog**.
        - Otherwise, call **UpsertActions**.
    - Increment the index to move to the next row.

UpsertSocialEvent Procedure

1. Check if an entry with the given **event_id** exists in the **social_event** table.
2. If the entry exists, update the existing record with the new data.
3. If the entry does not exist, insert a new record into the **social_event** table with the provided data.

UpsertDeviceLog Procedure

1. Check if an entry with the given **event_id** exists in the **device_event_log** table.
2. If the entry exists, update the existing record with the new data.
3. If the entry does not exist, insert a new record into the **device_event_log** table with the provided data.

UpsertActions Procedure

1. Check if an entry with the given **event_id** exists in the **social_action** table.
2. If the entry exists, update the existing record with the new data.
3. If the entry does not exist, insert a new record into the **social_action** table with the provided data.

process_hashtag Procedure

1. Determine the maximum data feed ID from **SM_data_feed** table.
2. Iterate over each row in the **SM_data_feed** table using a while loop.
    - For each iteration, retrieve the event ID and hashtags for the current **data_feed_id**.
    - Split the hashtags string into individual tags.
    - For each tag, call **UpsertHashtags** to either update an existing tag or insert a new one, linking it with the event.
    - Increment the index to move to the next row.

UpsertHashtags Procedure

1. Check if the provided hashtag already exists in the **hashtags** table.
2. If the hashtag exists, retrieve its ID.
3. If the hashtag does not exist, insert it into the **hashtags** table and retrieve the new ID.
4. Check if an association between the event and the hashtag already exists in the **event_tag** link table.
5. If the association does not exist, insert a new record into the **event_tag** table to link the event with the hashtag.

process_ip_address Procedure

1. Declare a variable to hold the IP address (**v_ip_address**).
2. Retrieve an IP address from the **SM_data_feed** table and store it in **v_ip_address**.
3. Check if **v_ip_address** is null.
    - If yes, set **v_ip_address** to the default value of '0.0.0.0'.
4. Remove all non-numeric characters from **v_ip_address**, except for periods.
5. Insert the processed **v_ip_address** into the **ip_address_table**.

process_location Procedure

1. Declare variables for person ID, account ID, location, and location ID.
2. Retrieve **person_id** and **account_id** from the **social_account** table where **SMUID** matches **p_SMUID**.
3. Retrieve **location** from the **SM_data_feed** table where **SMUID** matches **p_SMUID**.
4. If **v_location** is not null:
    - Check if the location already exists in the **address** table for the given **person_id** and **location**.
    - If the location does not exist (**v_location_id** is null), insert a new record into the **address** table with **person_id**, **account_id**, and **location**.

process_SMUID Procedure

1. Declare a variable for the social account ID (**SM_social_account_id**).
2. Retrieve **social_account_id** from the **social_account** table where **SMUID** matches **p_SMUID**.
3. If **SM_social_account_id** is null:
    - Insert a new record into the **social_account** table with **SMUID**.
    - Set **SM_social_account_id** to the last inserted ID.

add_device Procedure

1. Declare variables for account ID and device ID.
2. Call **processSMUID** with **p_SMUID** to process the social media user ID and retrieve the account ID.
3. Set **v_account_id** to the output of **processSMUID**.
4. If **p_device** is not null:
    - Retrieve **device_id** from the **device_make** table where **make_name** matches **p_device**.
    - If **v_device_id** is null (the device make does not exist):
        - Insert a new make into **device_make** with **p_device**.

- Insert a new model into **device_model** with "UNKNOWN" as the model name and the last inserted make ID as **make_id**.

Insert a new device record into **device** with **p_ip_address**, the last inserted model ID as **model_id**, and **v_account_id**.