Nadia Paquin
IST659 Database Administration
Final Project

An Inventory Database: Frank

As a lifelong hobbyist, I have become a collector of tools, materials, supplies, for everything I've been interested in for as long as I can remember. Collecting can become troublesome for me over time as I have yet to find a good system for accessible storage and efficient retrieval, and I often find myself yearning for a 'command F' button for my life. When I worked as a researcher in a biochemistry lab, for the first-time I witnessed storage management databases built for volume, efficiency, and accessibility. In attempt to recreate my own DBMS for my own things, I've built Frank.

My goal is to keep a digital log of all my things stored in a physical storage system:

- Three physical shelving units with 25 cubes each (5x5 grid)
- A list of users who can check in and check out items from the shelves
- Ability to see a view of all items and their locations, if they're checked out or in storage, when the last interaction was, and who did it
- A log of all the past interactions, with date and user
- A way to search for items based on name or category

Here are the tables I decided on:

- Shelves (with names decided arbitrarily)
- Bins (with names: A1:E5)
- Users (assuming me and my brothers are the only ones who access my storage)
- Categories (general terms to categorize each item)
- Items (all items and locations table)
- Checkout_history (all check-ins, check-outs, new additions)

Their relationships can be modeled by:

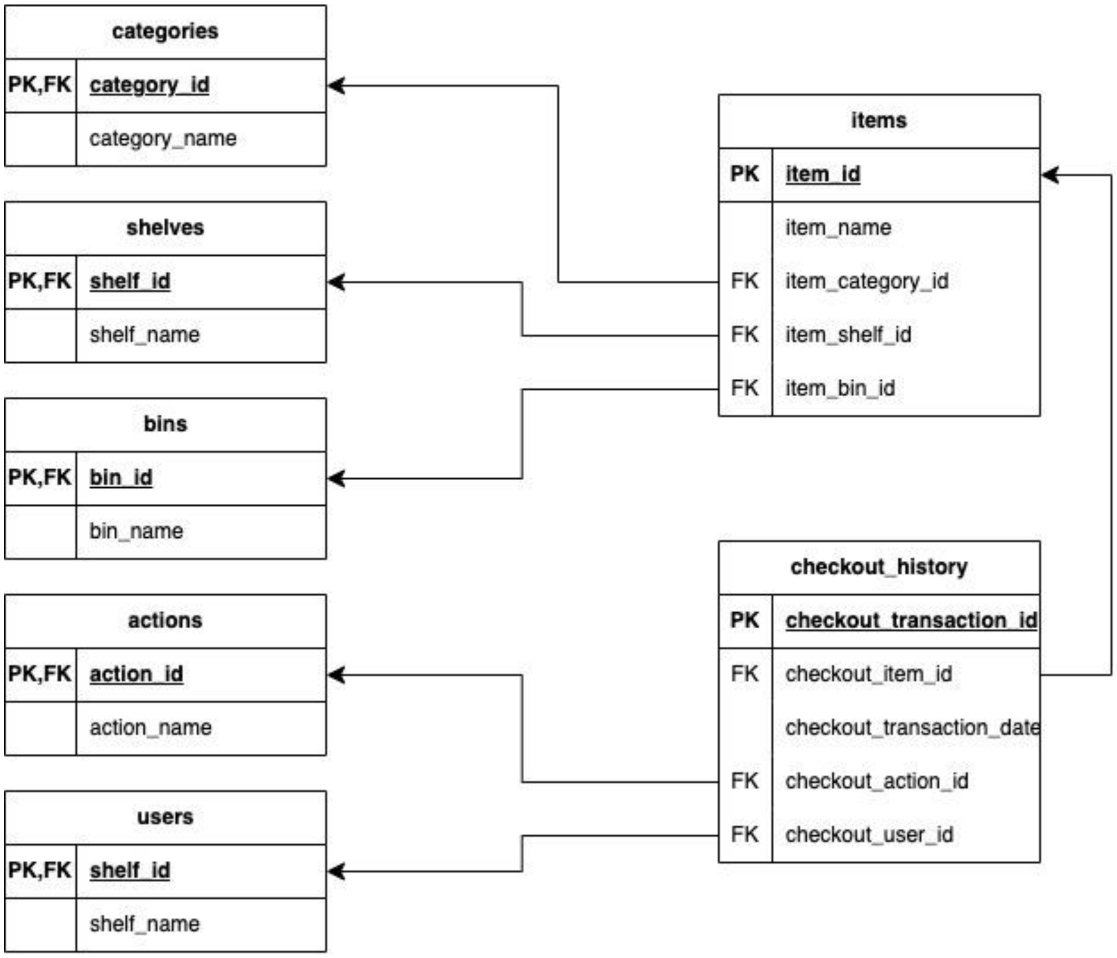

Figure 1. Frank Logical Model

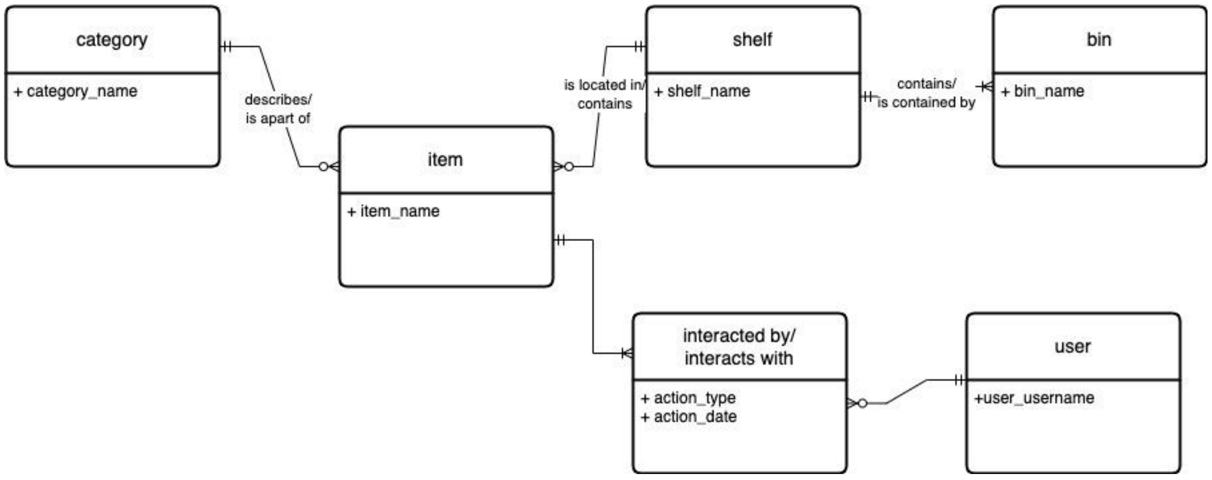

Figure 2. Frank Entity Relationship Diagram

Views:

- all_things_and_homes:
  - Displays item details including item_id, item_name, category_name, and location composed of shelf_name and bin_name.
- checkout_history_recent:
  - Shows the most recent checkout transaction details for each item, including checkout_item_id, most_recent_date, checkout_action_id, and checkout_user_id.
- all_things_history:
  - Provides a user-friendly view of the complete checkout history, showing checkout_transaction_id, checkout_transaction_date, checkout_item_id, action (action_name), and last_used_by (user_username).
- items_inventory:
  - Combines checkout history with item details to present a comprehensive inventory view, including checkout_item_id, item_name, category_name, location, most_recent_date, action, and last_used_by.
- items_inventory_cat_keys:
  - Extends items_inventory with keyword search functionality based on category_name keywords.
- items_inventory_item_keys:
  - Extends items_inventory with keyword search functionality based on item_name keywords.

Constraints:

- Primary Keys:
  - pk_shelves_shelf_id: Primary key constraint on shelf_id column in shelves table.
  - pk_bins_bin_id: Primary key constraint on bin_id column in bins table.
  - pk_actions_action_id: Primary key constraint on action_id column in actions table.
  - pk_categories_category_id: Primary key constraint on category_id column in categories table.
  - pk_users_user_id: Primary key constraint on user_id column in users table.
  - pk_items_item_id: Primary key constraint on item_id column in items table.
  - pk_checkout_history_transaction_id: Primary key constraint on checkout_transaction_id column in checkout_history table.
- Unique Constraints:
  - ck_shelves_shelf_name: Unique constraint on shelf_name column in shelves table.
  - ck_bins_bin_name: Unique constraint on bin_name column in bins table.
  - ck_actions_action_name: Unique constraint on action_name column in actions table.
  - ck_categories_category_name: Unique constraint on category_name column in categories table.
  - ck_users_username: Unique constraint on user_username column in users table.
  - u_items_item_name: Unique constraint on item_name column in items table.

- Foreign Key Constraints:
    - fk_items_item_shelf_id: Foreign key constraint referencing shelf_id column in shelves table from item_shelf_id column in items table.
    - fk_items_item_category_id: Foreign key constraint referencing category_id column in categories table from item_category_id column in items table.
    - fk_items_item_bin_id: Foreign key constraint referencing bin_id column in bins table from item_bin_id column in items table.
    - fk_checkout_history_item_id: Foreign key constraint referencing item_id column in items table from checkout_item_id column in checkout_history table.
    - fk_checkout_history_action_id: Foreign key constraint referencing action_id column in actions table from checkout_action_id column in checkout_history table.
    - fk_checkout_history_user_id: Foreign key constraint referencing user_id column in users table from checkout_user_id column in checkout_history table.

Functions:

- search_categories:
    - Parameters: Accepts a varchar parameter @search.
    - Returns: Returns a table containing rows from items_inventory_cat_keys where the keyword matches @search.
- search_items:
    - Parameters: Accepts a varchar parameter @search.
    - Returns: Returns a table containing rows from items_inventory_item_keys where the keyword matches @search.

Procedures:

- checkout_item:
    - Parameters: Takes @item_name (varchar) and @username (varchar).
    - Functionality: Updates checkout_history to mark an item as checked out by a specific user, with error handling for items already checked out or invalid user/item combinations.
- return_item:
    - Parameters: Takes @item_name (varchar) and @username (varchar).
    - Functionality: Updates checkout_history to mark an item as returned by a specific user, with error handling for items not checked out or invalid user/item combinations.
- add_item:
    - Parameters: Takes @item (varchar), @category (varchar), @shelf (char), and @bin (varchar).
    - Functionality: Inserts a new item into the items table with specified details including category, shelf, and bin, ensuring data integrity through foreign key references.
- remove_item:
    - Parameters: Takes @item (varchar).

    - Functionality: Deletes an item from the items table based on its name, ensuring data integrity and handling errors if the item cannot be found.

Data:

- Shelves:
    - Data Inserted: 'avila', 'shell', 'pismo'
    - Purpose: These are names of shelves where items are stored.
- Bins:
    - Data Inserted: A total of 25 bins ('A1' to 'E5')
    - Purpose: Bins where items can be stored, structured as a 5x5 grid for organization.
- Actions:
    - Data Inserted: 'new addition', 'checked out', 'returned'
    - Purpose: Types of actions recorded in the checkout history, indicating whether an item was added, checked out, or returned.
- Categories:
    - Data Inserted: 'tech equipment', 'surf accessories', 'art supplies', 'tools', 'misc'
    - Purpose: Categories to classify different types of items stored in the inventory.
- Users:
    - Data Inserted: 'nadia', 'anton', 'davos', 'dante'
    - Purpose: Usernames of individuals who interact with the inventory system by checking out or returning items.
- Items:
    - Data Inserted:
        - 'surf wax': Category - 'surf accessories', Shelf - 'avila', Bin - 'A1'
        - 'hdmi cable': Category - 'tech equipment', Shelf - 'avila', Bin - 'B1'
        - 'ceramic stamps': Category - 'art supplies', Shelf - 'shell', Bin - 'A1'
        - 'screwdriver': Category - 'tools', Shelf - 'pismo', Bin - 'D5'
        - 'paintbrushes': Category - 'art supplies', Shelf - 'avila', Bin - 'E2'
        - 'fins': Category - 'surf accessories', Shelf - 'shell', Bin - 'B5'
    - Purpose: Specific items stored in the inventory system, categorized by type, shelf location, and bin location.
- Checkout History:
    - Data Inserted:
        - Records various transactions where items were checked out or returned by users on specific dates.
        - Includes actions such as new additions, checkouts, and returns.

Error Handling:

- Error 50100: Raised if attempting to checkout items that are already checked out or return items that are not checked out.
- Error 50101: Raised if there is a failure to checkout or return an item, indicating issues with user registration or spelling.

- Error 50102: Raised if there is an issue with adding or removing items, ensuring actions are completed successfully and data integrity is maintained.