Database Design Document: Store Price Tracking Application

Problem Description: (Nouns, Verbs)

In neighborhoods or cities without online platforms, users find it challenging to identify the store selling specific items at the most affordable prices. The solution is to develop a web-based application where users can contribute prices for common items when visiting various stores. The app enables users to view prices per unit for each store, aiding them in making informed decisions about where to shop. The application incorporates crowdsourced price tracking for offline stores.

Rules:

1. Each location can have more than one store.
2. There are multiple locations.
3. Customers can contribute to the list of items.
4. Each item is categorized into one category.
5. One category can encompass multiple items.
6. Each store has at least one location.
7. Multiple stores can have the same name but different locations.
8. Each item has a specific price in one store.
9. Many stores can have the same item with the same/different prices.
10. Each store is associated with an expense rating.
11. Users can filter items by expense rating, category, specialty, etc.
12. A store cannot carry multiple quantities of the same item with different prices.
13. The application should track the timestamp of when a price is contributed.
14. Users can update the price of an item at a location.
15. Stores may or may not have a cultural speciality.

Important Nouns:

User
Contributor
Reviewer
Item
Category
Price
Store
Location
Expense Rating
Cultural Specialty
Crowdsourced
Price Tracking

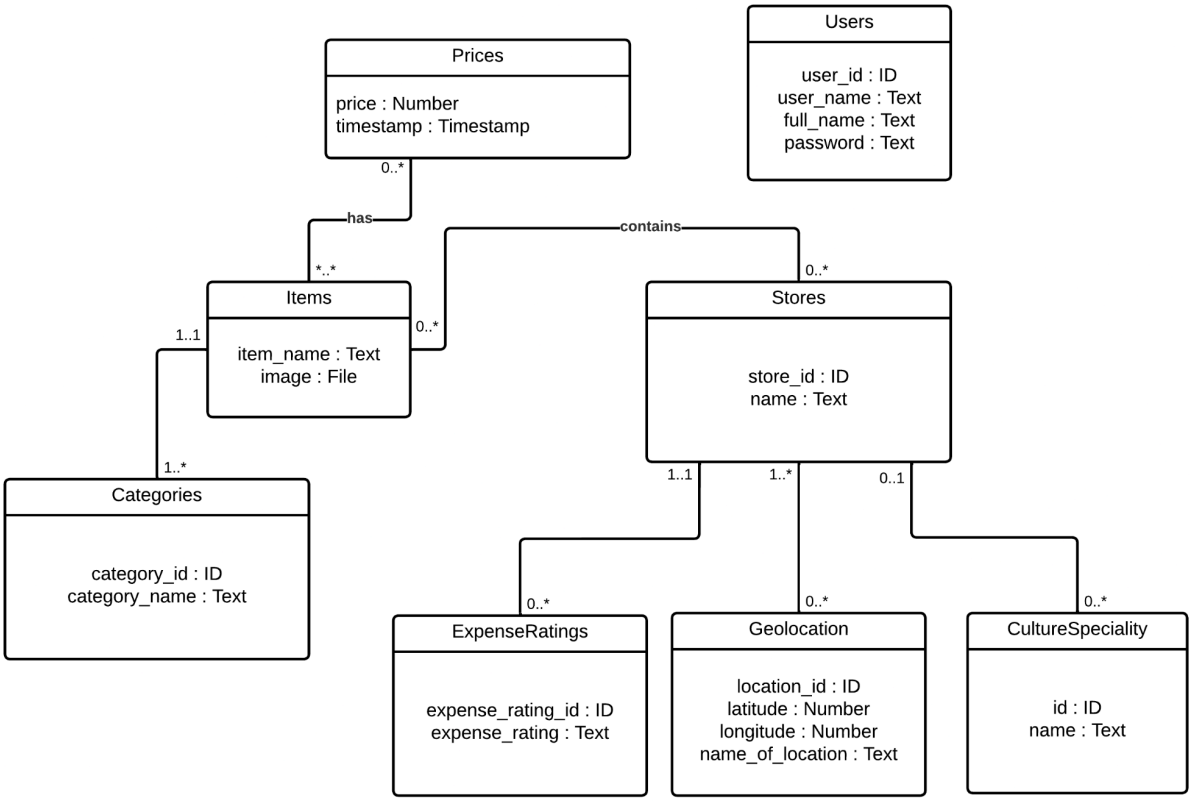
Important Verbs:

Contribute

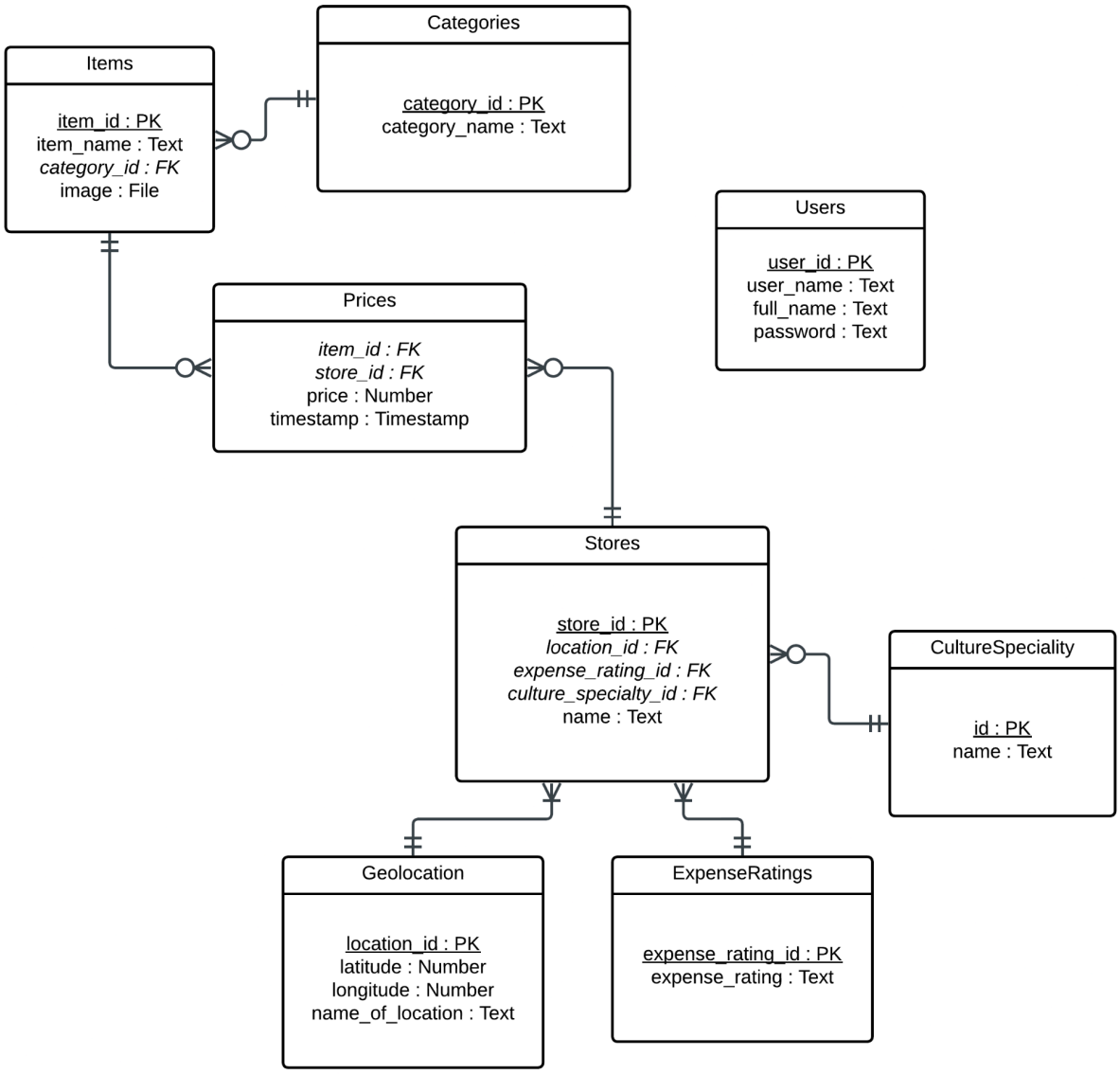
View
Discover
Filter
Categorize
Contribute
View
Track geolocation
Assign expense rating
Associate cultural specialty
Contribute prices
View prices
Record prices

LucidChart Diagram for UML & ERD - https://lucid.app/lucidchart/9e780024-8e62-4ee2-b3da-6eb836497986/edit?viewport_loc=-554%2C-961%2C3291%2C1622%2C0_0&invitationId=inv_c3d652b1-1f3a-4206-92a1-5e0b44ae9c4f

UML Diagram:



Crows Foot ERD:



Schema:

- Items(item_id, item_name, category_id, image)
- Categories(category_id, category_name)
- Prices(item_id, store_id, price, timestamp)
- Stores(store_id, location_id, expense_rating_id, culture_specialty_id, name)
- Users(user_id, user_name, full_name, password)
- CultureSpecialty(id, name)
- Geolocation(location_id, latitude, longitude, name_of_location)

- ExpenseRatings(expense_rating_id, expense_rating)

BCNF

Category - No non-trivial functional dependencies other than the superkey “category_id”

CultureSpecialty: - No non-trivial functional dependencies other than the superkey “id”

ExpenseRatings - No non-trivial functional dependencies other than the superkey “expense_rating_id”

Geolocation - No non-trivial functional dependencies other than the superkey “location_id”

Items - No non-trivial functional dependencies other than the superkey “item_id”

Prices - No non-trivial functional dependencies other than the superkey “item_id”

Stores - No non-trivial functional dependencies other than the superkey “store_id”

Users - No non-trivial functional dependencies other than the superkey “user_id”

Also, there are no many-to-many relationships. Therefore, our schema is indeed in BCNF.