Ecommerce Saved Search With Flexible Products Schema

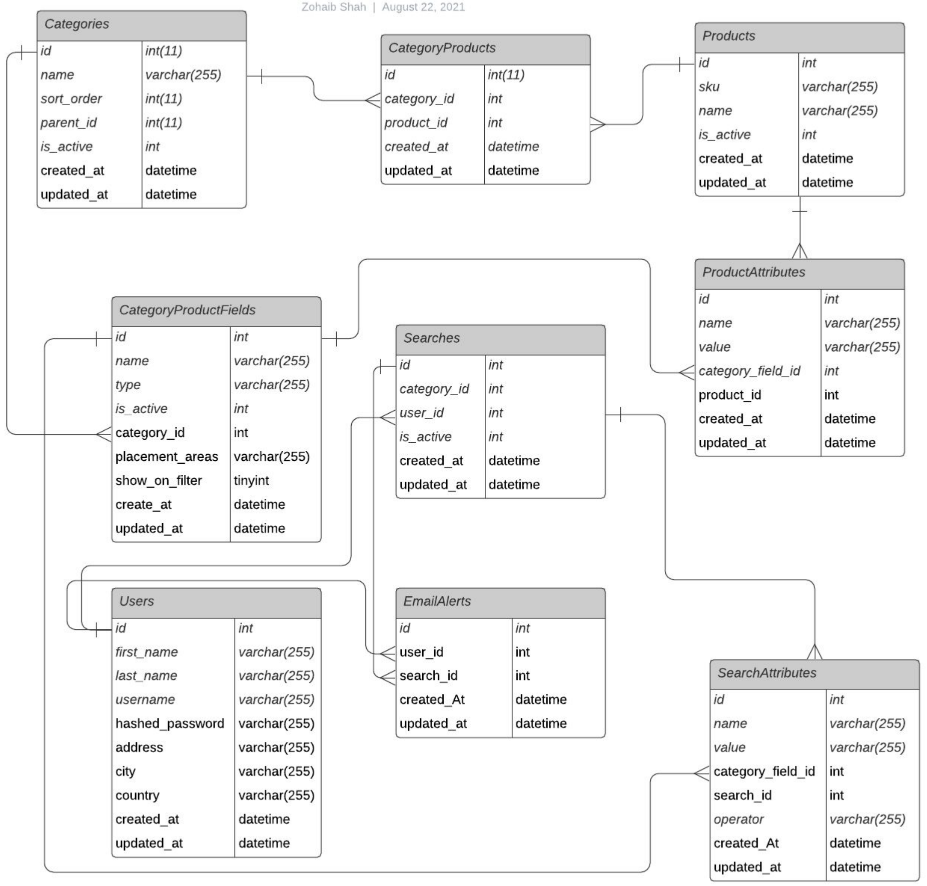
Created for the purpose of Assessment

T.O.C

1. ERD Diagram
2. Database Choice
3. Understanding of the mappings between entities
4. Achieving flexible Product’s Schema
5. Universal Save Search Mechanism
6. Future Considerations

Ecommerce-ERD-for-Zammen-Assessment

Zohaib Shah | August 22, 2021



Categories

id	int(11)
name	varchar(255)
sort_order	int(11)
parent_id	int(11)
is_active	int
created_at	datetime
updated_at	datetime

CategoryProducts

id	int(11)
category_id	int
product_id	int
created_at	datetime
updated_at	datetime

Products

id	int
sku	varchar(255)
name	varchar(255)
is_active	int
created_at	datetime
updated_at	datetime

ProductAttributes

id	int
name	varchar(255)
value	varchar(255)
category_field_id	int
product_id	int
created_at	datetime
updated_at	datetime

CategoryProductFields

id	int
name	varchar(255)
type	varchar(255)
is_active	int
category_id	int
placement_areas	varchar(255)
show_on_filter	tinyint
create_at	datetime
updated_at	datetime

Searches

id	int
category_id	int
user_id	int
is_active	int
created_at	datetime
updated_at	datetime

Users

id	int
first_name	varchar(255)
last_name	varchar(255)
username	varchar(255)
hashed_password	varchar(255)
address	varchar(255)
city	varchar(255)
country	varchar(255)
created_at	datetime
updated_at	datetime

EmailAlerts

id	int
user_id	int
search_id	int
created_At	datetime
updated_at	datetime

SearchAttributes

id	int
name	varchar(255)
value	varchar(255)
category_field_id	int
search_id	int
operator	varchar(255)
created_At	datetime
updated_at	datetime

2- Database Choice

As I was working keeping assessment in mind, I preferred Structural DB .i.e. MySQL. The same concept could be applied while shifting to MongoDB.

As MongoDB does not force us to follow specific schema, We could have a “Products” collection. In this collection, we could save products of different categories (with different attributes) without any problems.

3- Understanding the mapping between entities

1- **Categories:** A category will have multiple products associated with it. A category may have a parent category if the *parent_id* field is not equals to zero.

2- **Products:** A product may belong to multiple categories. It has Many-to-Many relation with **categories** entity. A **Product** hasMany **ProductAttributes.** A product may also be associated with a user if the project allowes users to add products (I didn’t cover this aspect). **Product attributes** are separated to allow flexible products fields. For example, A product in Electronics category will have different attributes as compared to a product in **Real Estate** category.

3- Understanding the mapping between entities

3. **CategoryProductFields:** This entity will contain the information about the possible fields / attributes a product of a specific category could have. Whenever a new Product is being added, application should refer to CategoryProductFields entity for possible product attributes. Following is the little definition about it's fields:

3.1 - **name** is the name of the attribute allowed for ProductAttributes

3.2 - **type** is the type of that particular attribute .i.e. String,integer etc.

3.3 - **placement_areas** comma separated field. Could be used to identify different parts of the website and check if the particular attribute is allowed to display in that part of the website.

3.4 - **show_on_filter** if set to 1, The particular attribute will be available in filtration area of the listing page.

3- Understanding the mapping between entities

4- **ProductAttributes:** This entity will contain information about a product. Whenever a product is being added in a specific category, application should first check **CategoriesProductField** and present relevant form to add **ProductAttributes** which is more like a key, value table and mainly depend on CategoryProductField. Ideally, ProductAttributes should be used for uncommon attributes of products belonging to different categories.

3- Understanding the mapping between entities

5- **Searches:** This entity should contain searches saved by the users. Every search belongs to a Category and each Category could have multiple searches saved against it.

6- **SearchAttributes** belongs to a search. It also belongs to a CategoryProductField. The **name, value and operator** denotes the filter criteria used by the user. Like if a user has filtered for *price < 20000,* This should go into the search attributes as name=”price”,value=”20000” and operator=”<”

7- **EmailAlerts** belongs to a search, every search could have multiple alerts sent against it.

4- Achieving the Flexible Products Schema

As a product could belong to different categories, their attributes have to be different as well. We could not have a different table for every product type like one table for electronics and other for Real Estate. To fix this, we have introduced **CategoryProductFields** table and made **ProductAttributes** a key,value table.

5- Universal Save Search Mechanism

With universal product entry setup, we have also made it possible to have a universal save search mechanism. Every Search belongsTo a category and every search attribute stores the information about the searched and filtered parameter.

6- Future Considerations

1- Choice of DB could easily be shifted to MongoDB which has flexible collection schema as it’s default.

2- Current working is obviously not complete as there are many other tables needed for complete Ecommerce / Marketplace website.

3- Though the proposed structure is flexible one but implementation on application layer is challenging. Many aspects needs to be considered, we may need a coding layer to normalize the data before creating a REST API.

Thank You

Prepared by : Zohaib Shah

Email : weblearningblog@gmail.com