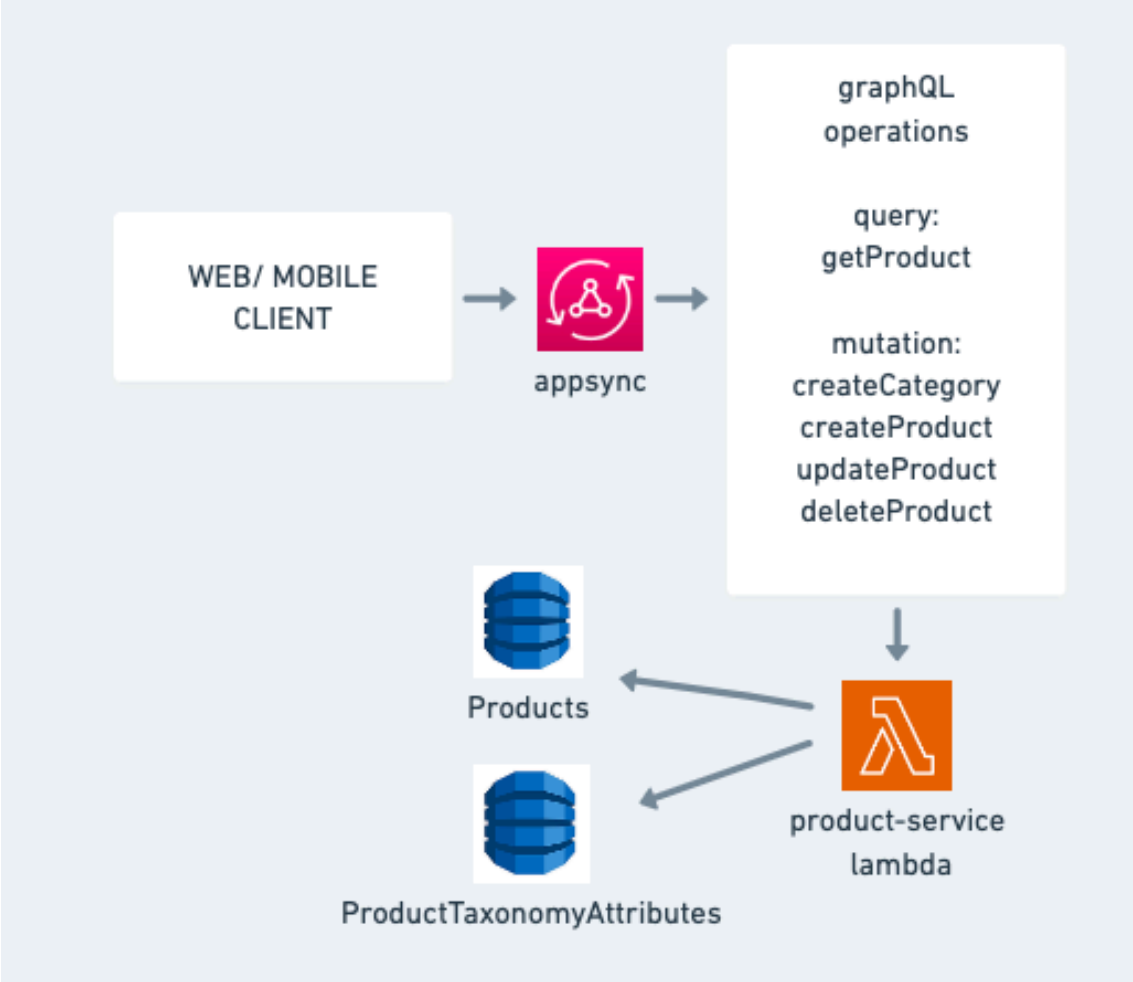
Product Data Model in DynamoDB

1. Overview

This document describes the **Product Data Model** in **AWS DynamoDB**, including table structure, indexing strategies, and category management for an eCommerce application.



2. Tables and Schema

The database consists of two tables:

1. **Products** - Stores product details.
2. **ProductTaxonomyAttributes** - Stores category and subcategory information.

---

3. Products Table

Schema:

Attribute	Type	Description
ProductId	String (PK)	Unique identifier for each product.
Name	String	Name of the product.
Description	String	Detailed product description.
Price	Number	Price of the product.
Category	String	Reference to TaxonomyId (category).
Stock	Number	Inventory count.
CreatedAt	Timestamp	Timestamp when the product was added.
UpdatedAt	Timestamp	Timestamp of the last update.

Primary Key:

- **Partition Key:** `ProductId`

Secondary Indexes (Optional):

- **GSI - CategoryIndex** (For querying products by category)
    - **Partition Key:** `Category`
    - **Sort Key:** `CreatedAt`

---

4. ProductTaxonomyAttributes Table

Schema:

Attribute	Type	Description
TaxonomyId	String (PK)	Unique identifier for each category/tag.
Name	String	Category or tag name.
Description	String	Brief description of the category/tag.
ParentId	String	Parent category reference (NULL for top-level).

Primary Key:

- **Partition Key:** `TaxonomyId`

Secondary Indexes:

- **GSI - ParentIndex** (For hierarchical category lookup)
  - **Partition Key:** `ParentId`
  - **Sort Key:** `Name`

---

5. Category Management in Products

To ensure **consistent category references**, products should only store a **TaxonomyId** that exists in `ProductTaxonomyAttributes`.

Predefined Category Workflow:

Enums for Categories & Subcategories:

```
enum ProductCategory {
  ELECTRONICS
  CLOTHING
  BOOKS
}

enum ProductSubCategory {
  # Electronics
  SMARTPHONES
  LAPTOPS
  TABLETS
  ACCESSORIES

  # Clothing
  MENS_WEAR
  WOMENS_WEAR
  KIDS_WEAR
  FOOTWEAR

  # Books
  FICTION
  NON_FICTION
  EDUCATIONAL
  COMICS
}
```

---

6. GraphQL Schema for Product Management

```
type Product {
  ProductId: String
  Name: String
  Description: String
  Price: Float
  Category: String
  Stock: Int
  CreatedAt: String
  UpdatedAt: String
}
type Query {
  getProduct(input: ProductInput!): Product
}
type Mutation {
  createProduct(input: ProductInput): Response
  updateProduct(input: ProductInput): Response
  deleteProduct(input: ProductInput): Response
  createCategory(input: CategoryInput): Response
}
input ProductInput {
  ProductId: String
  Name: String
  Description: String
  Price: Float
  Category: String
  Stock: Int
}
input CategoryInput {
  TaxonomyId: String
  Name: String
  Description: String
  ParentId: String
}
type Response {
  statusCode: Int
  message: String
  data: AWSJSON
}
```

7. DB Schema

structured for **GraphQL API operations**, **Access Patterns**, and **Sample Data**

API	Access Pattern	Sample Data
Query `getProduct (input: {ProductId: ""}) { Category CreatedAt Description Name Price ProductId Stock UpdatedAt }`	**PK EQUALS** `<ProductId>`	`{ "ProductId": "12345", "Name": "Sample Product", "Category": "Electronics", "Price": 99.99, "Stock": 50, "CreatedAt": "2025-02-03T00:00:00Z", "UpdatedAt": "2025-02-03T00:00:00Z" }`
Mutation `createCategory (input: { Description: "", Name: "", ParentId: ""}) { data message statusCode }`	**PK EQUALS** `<TaxonomyId>`	`{ "Name": "Electronics", "Description": "Electronic Items", "ParentId": "root" }`
Mutation `createProduct ( input: {Category: "", Description: "", Name: "", Price: 1.5, Stock: 10} ) { data message statusCode }`	**PK EQUALS** `<ProductId>`	`{ "Name": "Laptop", "Category": "Electronics", "Price": 1200.50, "Stock": 10, "Description": "Gaming Laptop" }`

Mutation `deleteProduct (input: {ProductId: ""}) { data message statusCode }`	**PK EQUALS** `<ProductId>`	`{ "ProductId": "12345" }`
Mutation `updateProduct (input: {Category: "", Description: "", Name: "", Price: 1.5, ProductId: "", Stock: 10}) { data message statusCode }`	**PK EQUALS** `<ProductId>`	`{ "ProductId": "12345", "Name": "Updated Laptop", "Price": 1150.00, "Stock": 8, "Description": "Updated Gaming Laptop" }`

---

8. Summary

- Use **Products Table** to store product details.
- Use **ProductTaxonomyAttributes Table** for structured categories.
- Implement **GSIs** for optimized queries.
- GraphQL API allows CRUD operations on products and categories.

This approach ensures **scalability, performance, and data consistency** for an eCommerce product catalog in DynamoDB.

Repo link:
https://github.com/vishalgl98/expedite-commerce.git