Database Schema Documentation

Overview

This document describes the database schema for the E-commerce Admin API. The database is designed using PostgreSQL and follows normalized database design principles to prevent redundancy and maintain data consistency.

Entity Relationship Diagram (Conceptual)

Table Definitions

1. categories

Purpose: Stores product categories for organizational and filtering purposes.

Column	Type	Constraints	Description
id	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Unique category identifier
name	VARCHAR(100)	NOT NULL, UNIQUE	Category name
description	TEXT	NULL	Category description
created_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Category creation timestamp

Indexes:

- PRIMARY KEY on `id`
- UNIQUE INDEX on `name`
- INDEX on `name` for fast lookups

Sample Data:

```sql
INSERT INTO categories (name, description) VALUES
('Electronics', 'Electronic devices and accessories'),
('Home & Kitchen', 'Home appliances and kitchen items'),
('Books', 'Books and educational materials');
```

2. products

Purpose: Central product catalog containing all product information.

Column	Type	Constraints	Description
id	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Unique product identifier
name	VARCHAR(200)	NOT NULL	Product name
description	TEXT	NULL	Product description
price	DECIMAL(10,2)	NOT NULL	Product price
sku	VARCHAR(50)	NOT NULL, UNIQUE	Stock Keeping Unit
category_id	INTEGER	NOT NULL, FOREIGN KEY	Reference to categories table
platform	VARCHAR(50)	NOT NULL	Sales platform (Amazon, etc.)
is_active	BOOLEAN	NOT NULL, DEFAULT TRUE	Product active status
created_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Product creation timestamp
updated_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Last update timestamp

Constraints:

- `FOREIGN KEY (category_id) REFERENCES categories(id)`
- `CHECK (price > 0)` - Ensures positive prices

Indexes:

- PRIMARY KEY on `id`
- UNIQUE INDEX on `sku`
- INDEX on `name` for search functionality
- INDEX on `category_id` for category filtering
- INDEX on `platform` for platform filtering

Sample Data:

```sql
INSERT INTO products (name, price, sku, category_id, platform) VALUES
('Samsung Galaxy Earbuds Pro', 199.99, 'SAM-EAR-001', 1, 'Amazon'),
('iPhone 15 Case', 24.99, 'APL-CAS-001', 1, 'Amazon');
```

3. inventory

Purpose: Tracks current stock levels and manages low stock alerts.

Column	Type	Constraints	Description
id	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Unique inventory identifier
product_id	INTEGER	NOT NULL, UNIQUE, FOREIGN KEY	Reference to products table
quantity	INTEGER	NOT NULL, DEFAULT 0	Current stock quantity
low_stock_threshold	INTEGER	NOT NULL, DEFAULT 10	Minimum stock alert level
last_updated	TIMESTAMP	NOT NULL, DEFAULT NOW()	Last inventory update

Constraints:

- `FOREIGN KEY (product_id) REFERENCES products(id)`
- `CHECK (quantity >= 0)` - Prevents negative inventory
- `CHECK (low_stock_threshold >= 0)` - Ensures valid threshold

Indexes:

- PRIMARY KEY on `id`
- UNIQUE INDEX on `product_id`
- INDEX on `quantity` for low stock queries

Business Logic:

- Low stock alert when `quantity <= low_stock_threshold`
- Automatic inventory reduction on sales

Sample Data:

```sql
INSERT INTO inventory (product_id, quantity, low_stock_threshold) VALUES
(1, 150, 20),
(2, 75, 15);
```

4. sales

Purpose: Records all sales transactions for analytics and reporting.

Column	Type	Constraints	Description
id	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Unique sale identifier
product_id	INTEGER	NOT NULL, FOREIGN KEY	Reference to products table
quantity	INTEGER	NOT NULL	Quantity sold
unit_price	DECIMAL(10,2)	NOT NULL	Price per unit at sale time
total_amount	DECIMAL(10,2)	NOT NULL	Total sale amount
sale_date	TIMESTAMP	NOT NULL	Date and time of sale
platform	VARCHAR(50)	NOT NULL	Platform where sale occurred
order_id	VARCHAR(100)	NULL	External order identifier
created_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Record creation timestamp

Constraints:

- `FOREIGN KEY (product_id) REFERENCES products(id)`
- `CHECK (quantity > 0)` - Ensures positive quantity
- `CHECK (unit_price > 0)` - Ensures positive price
- `CHECK (total_amount > 0)` - Ensures positive total

Indexes:

- PRIMARY KEY on `id`
- INDEX on `product_id` for product-specific queries
- INDEX on `sale_date` for time-based analytics
- INDEX on `platform` for platform-specific reports
- INDEX on `order_id` for order lookups
- COMPOSITE INDEX on `(sale_date, platform)` for common filter combinations

Calculated Fields:

- `total_amount` should equal `quantity * unit_price`

Sample Data:

```sql
INSERT INTO sales (product_id, quantity, unit_price, total_amount, sale_date, platform, order_i
(1, 2, 199.99, 399.98, '2024-01-15 14:30:00', 'Amazon', 'ORD-123456'),
(2, 1, 24.99, 24.99, '2024-01-15 16:45:00', 'Amazon', 'ORD-123457');
```

Relationships

1. Categories → Products (One-to-Many)

- One category can contain multiple products
- Each product belongs to exactly one category
- Enforced by foreign key constraint
- Allows for product categorization and filtering

2. Products → Inventory (One-to-One)

- Each product has exactly one inventory record
- Ensures stock tracking for every product
- Enforced by unique constraint on product_id in inventory table

3. Products → Sales (One-to-Many)

- One product can have multiple sales records
- Each sale record refers to exactly one product
- Enables sales history and analytics per product

Normalization

The database follows **Third Normal Form (3NF)**:

First Normal Form (1NF)

- All tables have atomic values
- No repeating groups

- Each column contains single values

Second Normal Form (2NF)

- Meets 1NF requirements
- No partial dependencies on composite keys
- All non-key attributes depend on the entire primary key

Third Normal Form (3NF)

- Meets 2NF requirements
- No transitive dependencies
- Non-key attributes don't depend on other non-key attributes

Data Integrity

Referential Integrity

- Foreign key constraints ensure valid references
- Cascade rules prevent orphaned records
- ON DELETE RESTRICT prevents deletion of referenced categories

Data Validation

- CHECK constraints ensure data quality
- NOT NULL constraints prevent missing critical data
- UNIQUE constraints prevent duplicates

Business Rules Enforcement

1. **SKU Uniqueness**: Prevents duplicate product identifiers
2. **Positive Values**: Prices, quantities, and amounts must be positive
3. **Stock Consistency**: Inventory automatically updates on sales
4. **Category Assignment**: Every product must belong to a category

Performance Considerations

Indexing Strategy

1. **Primary Keys**: Automatic clustered indexes for fast lookups
2. **Foreign Keys**: Indexes on all foreign key columns

3. **Search Fields**: Indexes on frequently searched columns (name, sku)
4. **Filter Fields**: Indexes on common filter columns (platform, sale_date)
5. **Composite Indexes**: For common multi-column filters

Query Optimization

- Use of proper JOIN operations
- Efficient WHERE clause ordering
- Pagination for large result sets
- Aggregate functions for analytics

Data Volume Expectations

Estimated Volumes (Production)

- **Categories**: ~50-100 records
- **Products**: ~10,000-100,000 records
- **Inventory**: Same as products (1:1 ratio)
- **Sales**: ~1,000-10,000 records per day

Growth Patterns

- Products: Steady growth with seasonal additions
- Sales: Daily accumulation with seasonal peaks
- Inventory: Static count, frequent updates
- Categories: Minimal growth after initial setup

Backup and Maintenance

Recommended Practices

1. **Daily Backups**: Full database backup
2. **Transaction Log Backups**: Every 15 minutes
3. **Index Maintenance**: Weekly rebuild/reorganize
4. **Statistics Updates**: Weekly on high-activity tables

Data Retention

- **Sales Data**: Retain for 7 years (compliance)
- **Product History**: Soft delete with is_active flag

- **Inventory History**: Consider separate audit table for tracking

Security Considerations

Access Control

- Read-only access for reporting users
- Full access only for admin operations
- Separate credentials for application vs. admin access

Data Protection

- Encrypt sensitive data at rest
- Use parameterized queries to prevent SQL injection
- Implement connection pooling with proper authentication

Future Enhancements

Potential Schema Extensions

1. **Users Table**: For authentication and user management
2. **Audit Tables**: For tracking all data changes
3. **Suppliers Table**: For supply chain management
4. **Price History**: For tracking price changes over time
5. **Product Images**: For storing product image references
6. **Customer Data**: For order and customer analytics

Scalability Considerations

- Table partitioning for large sales data
- Read replicas for analytics queries
- Caching layer for frequently accessed data
- Archive old sales data to separate tables