StoreFlow — A1.2 Data Model & SQL Specification (Part 1)

1. Data Architecture Overview

This section outlines the complete data architecture of StoreFlow, describing how the database is structured, why the structure was chosen, and how it supports a multi-tenant SaaS application with strict isolation.

StoreFlow uses MySQL 8 as the primary relational database system. All tables are scoped by merchant_id and, where appropriate, store_id to ensure strict tenant isolation.
The system is fully normalized to minimize redundancy.

Goals:

• Maintain strict tenant separation.

• Ensure scalable order, product, and shipping queries.

• Support real-time updates without causing blocking queries.

• Maintain strong referential integrity across all domains.

• Allow horizontal scaling with minimal cross-node locking.

2. Multi-Tenant Data Isolation Model

StoreFlow uses a hybrid multi-tenant architecture:

Merchant Level Isolation:

• Every core table includes merchant_id.

• Queries are ALWAYS filtered by authenticated user's merchant_id.

• No user is ever allowed to operate outside their merchant's dataset.

Store Level Isolation:

• Store-specific data (orders, shipping config, products if store-scoped) contains store_id.

• Policies enforce store membership for manager/staff users.

Security Enforcement:

• Laravel global scopes or manual scoping.

• Policy-based authorization on read/write.

• Cross-tenant leakage is structurally impossible through DB design.

Data Integrity:

• Foreign keys always include merchant_id to prevent association with foreign tenant data.

• Cascading deletes protect consistency when stores or products are removed.

3. Core Entity Groups

StoreFlow’s schema is organized into several domain groups:

1. Merchant & Store Domain

- merchants

- stores

- users

- store_users

2. Product Catalog Domain

- products

- customization_groups

- customization_options

3. Customer Domain

- customers

- loyalty_accounts

4. Order & Fulfilment Domain

- orders

- order_items

- order_item_options

5. Shipping Domain

- shipping_zones

- shipping_methods

- shipping_rates

6. Logging & Analytics Domain

- audit_logs

7. Future domains (reserved)

- advanced analytics

- plugin system

- marketplace extensions

4. Entity Relationship Summary (High-Level)

High-Level ER Summary:

Merchant 1---N Stores

Merchant 1---N Users

Store 1---N Store Users

Store 1---N Orders

Customer 1---N Orders

Order 1---N Order Items

Order Item 1---N Order Item Options

Store 1---N Shipping Zones

Shipping Zone 1---N Shipping Rates

Store 1---N Shipping Methods

Product 1---N Customization Groups

Customization Group 1---N Customization Options

All major tables link back to merchants for top-level isolation.

Orders link to both stores and customers.

Shipping tables link only to stores.

Products can be merchant-wide or store-specific depending on setup.

5. Full Table Definitions — Merchant & Store Domain

TABLE: merchants

id BIGINT PK

name VARCHAR(255)

slug VARCHAR(255) UNIQUE

owner_user_id BIGINT FK → users.id

stripe_account_id VARCHAR(255) NULL (future)

created_at TIMESTAMP

updated_at TIMESTAMP

TABLE: stores

id BIGINT PK

merchant_id FK → merchants.id

name VARCHAR(255)

description TEXT

theme_key ENUM('classic','modern','minimal')

timezone VARCHAR(255)

shipping_enabled BOOLEAN DEFAULT TRUE

created_at, updated_at

TABLE: users

id BIGINT PK

merchant_id FK

username VARCHAR(150) UNIQUE

password_hash VARCHAR(255)

role ENUM('owner','manager','staff')

created_at, updated_at

TABLE: store_users

id BIGINT PK

store_id FK

user_id FK

role ENUM('manager','staff')

created_at, updated_at

Notes:

• store_users maps users to specific stores.

• Owners do not appear in store_users—they implicitly have access to all stores.

• Index on (merchant_id, username) improves login resolution in multi-merchant systems.