StoreFlow — A1.2 Data Model & SQL Specification (Part 5)

34. Shipping Domain – Overview

The Shipping Domain encapsulates all configuration and data required to calculate shipping costs dynamically based on destination postcode, cart weight, and cart value.

It contains:

• shipping_zones

• shipping_methods

• shipping_rates

Design requirements:

• Per-store configuration (shipping is always store-specific).

• Support for multiple methods (standard, express, courier).

• Flexible rate modeling based on weight and price ranges.

• Clean integration with the shipping engine and checkout logic.

35. Shipping Zones – Full Specification

TABLE: shipping_zones

---

id BIGINT PK

store_id BIGINT FK → stores.id

name VARCHAR(255) NOT NULL

country VARCHAR(2) NOT NULL -- ISO-3166

state VARCHAR(64) NULL -- optional state filter

postcode_from INT NOT NULL

postcode_to INT NOT NULL

created_at TIMESTAMP

updated_at TIMESTAMP

Zone Matching Rules:

• The engine finds a zone where:

- country matches customer country

- postcode_from <= customer_postcode <= postcode_to

• state is optional; when present, it should also match customer's state.

Indexes:

• idx_zones_store (store_id)

• idx_zones_range (store_id, country, postcode_from, postcode_to)

Notes:

• Non-overlapping postcode ranges are recommended but not enforced.

• For fine-grained control, merchants can define multiple small ranges per zone.

36. Shipping Methods – Full Specification

TABLE: shipping_methods

---

id BIGINT PK

store_id BIGINT FK → stores.id

name VARCHAR(255) NOT NULL -- e.g., 'Standard', 'Express'

type ENUM('flat','weight','price','formula') NOT NULL

is_active BOOLEAN DEFAULT TRUE

created_at TIMESTAMP

updated_at TIMESTAMP

Type meanings:

• flat: base fixed cost regardless of weight/price

• weight: cost influenced by total weight

• price: cost based on cart total

• formula: reserved for future carrier integrations

Notes:

• Methods are independent of zones; linkage occurs via shipping_rates.

• Methods can be toggled on/off at any time without losing config.

37. Shipping Rates – Full Specification

TABLE: shipping_rates

---

id BIGINT PK

method_id BIGINT FK → shipping_methods.id

zone_id BIGINT FK → shipping_zones.id

min_weight_grams INT NULL

max_weight_grams INT NULL

min_cart_total_cents INT NULL

max_cart_total_cents INT NULL

base_price_cents INT NOT NULL

price_per_kg_cents INT NULL

created_at TIMESTAMP

updated_at TIMESTAMP

Range Semantics:

• NULL in min_* means no lower bound.

• NULL in max_* means no upper bound.

• Both ranges must be satisfied if provided.

Cost Formula:

base_price_cents

+ (price_per_kg_cents * ceil(total_weight_grams / 1000)) when price_per_kg_cents is not NULL

Indexes:

• idx_rates_method_zone (method_id, zone_id)

• idx_rates_weight (method_id, min_weight_grams, max_weight_grams)

• idx_rates_price (method_id, min_cart_total_cents, max_cart_total_cents)

Notes:

• Multiple rates per method/zone allow for tiered pricing (e.g., 0–2kg, 2–5kg).

• Free shipping thresholds implemented with base_price_cents = 0 and min_cart_total_cents set accordingly.

38. Loyalty Configuration – Full Specification

The Loyalty Domain provides a simple points-based loyalty engine at the merchant level.

TABLE: loyalty_config

---

id BIGINT PK

merchant_id BIGINT FK → merchants.id

type ENUM('points') NOT NULL -- extensible for future types

points_per_dollar INT NOT NULL -- e.g., 1 point per $1

threshold INT NOT NULL -- points needed to redeem

reward_json JSON NOT NULL -- describes reward

created_at TIMESTAMP

updated_at TIMESTAMP

reward_json examples:

• { "type": "fixed_discount", "amount_cents": 1000 }

• { "type": "percentage_discount", "percent": 10 }

Notes:

• One config row per merchant.

• Deactivation implemented by setting points_per_dollar = 0 or threshold = very high value.

39. Loyalty Accounts – Specification (Revisited)

TABLE: loyalty_accounts

---

id BIGINT PK

merchant_id BIGINT FK → merchants.id

customer_id BIGINT FK → customers.id

points_balance INT NOT NULL DEFAULT 0

created_at TIMESTAMP

updated_at TIMESTAMP

Constraints:

• UNIQUE (merchant_id, customer_id)

• Points cannot be negative (enforced at service layer).

Loyalty Accrual Logic:

• On paid order completion:

- base_points = floor(items_total_cents / 100) * points_per_dollar

- points_balance += base_points

Loyalty Redemption Logic:

• If customer chooses to redeem:

- system checks if points_balance >= threshold

- applies reward_json rules to discount order

- deducts threshold (or more for multiple redemptions).

40. Audit Logs – Full Specification

TABLE: audit_logs

---

id BIGINT PK

merchant_id BIGINT FK → merchants.id

user_id BIGINT FK → users.id NULL -- NULL for system-generated events

entity VARCHAR(64) NOT NULL -- e.g., 'order','product','shipping_config'

entity_id BIGINT NULL

action VARCHAR(64) NOT NULL -- e.g., 'created','updated','deleted','status_changed'

meta_json JSON NOT NULL -- structured change context

created_at TIMESTAMP

Indexes:

• idx_audit_merchant (merchant_id, created_at)

• idx_audit_entity (entity, entity_id)

Meta_json contents:

• For orders:

- { "from_status": "...", "to_status": "...", "reason": "...", "ip": "..." }

• For shipping config:

- { "field": "base_price_cents", "old": 800, "new": 900 }

• For user management:

- { "user_id": 123, "old_role": "staff", "new_role": "manager" }

Use Cases:

• Owner/manager audit views.

• Forensics after incidents.

• Compliance reporting.

41. Indexing & Performance Strategy for These Domains

Performance considerations:

Shipping:

• Lookup: store_id + country + postcode range is common path.

• Preload active methods and rates for store in memory/Redis if needed.

• Indexes on range fields allow MySQL to prune candidates quickly.

Loyalty:

• Frequent reads on loyalty_accounts by (merchant_id, customer_id).

• Rare writes (only on paid order events).

• UNIQUE index ensures no duplicates.

Audit Logs:

• Writes are frequent but small.

• Reads typically filtered by merchant_id and created_at range.

• Partitioning by time (optional in the future) can keep queries fast.

Overall:

• MySQL 8 with InnoDB and proper composite indexes will handle the initial SaaS scale well.

42. Example SQL Queries – Shipping, Loyalty, Audit

Shipping – find matching rate:

```
SELECT r.*
FROM shipping_zones z
JOIN shipping_rates r ON r.zone_id = z.id
JOIN shipping_methods m ON m.id = r.method_id
WHERE z.store_id = ?
AND z.country = ?
AND z.postcode_from <= ?
AND z.postcode_to >= ?
AND m.is_active = 1
AND (r.min_weight_grams IS NULL OR r.min_weight_grams <= ?)
AND (r.max_weight_grams IS NULL OR r.max_weight_grams >= ?)
AND (r.min_cart_total_cents IS NULL OR r.min_cart_total_cents <= ?)
AND (r.max_cart_total_cents IS NULL OR r.max_cart_total_cents >= ?);
```

Loyalty – get account or create:

```
SELECT * FROM loyalty_accounts
WHERE merchant_id = ? AND customer_id = ?;
```

Audit – recent changes for a store:

```
SELECT * FROM audit_logs
WHERE merchant_id = ?
AND entity = 'order'
AND created_at >= DATE_SUB(NOW(), INTERVAL 7 DAY)
ORDER BY created_at DESC;
```