Database / Table Design Document

1. Database Design Objective

Database Design นี้ออกแบบเพื่อรองรับระบบคำนวณราคาสุทธิของคำสั่งซื้อ โดยมีเป้าหมายหลักคือ:

- รองรับ Product และ Order
- รองรับ Promotion หลายรูปแบบ
- รองรับ Promotion แบบซ้อนกัน
- รองรับการเพิ่ม Promotion ใหม่ในอนาคต
- รองรับการตรวจสอบย้อนหลัง
- รองรับ Usage Limit และ Coupon Limit
- ใช้กับ MySQL + GORM ได้ง่าย
- ลดการแก้ Schema เมื่อมี Promotion ใหม่

แนวคิดหลักคือแยก Promotion ออกเป็น 4 ส่วน:

```
Promotion = ข้อมูลหลักของ Promotion
Condition = ใช้ได้เมื่อไหร่
Target    = ใช้กับอะไร
Action    = ลดอย่างไร
```

---

2. Table Overview

ระบบจะแบ่ง Table ออกเป็น 4 กลุ่มหลัก

2.1 Product Tables

```
product_categories
products
```

2.2 Order Tables

```
orders
order_items
```

2.3 Promotion Configuration Tables

```
promotions
promotion_conditions
promotion_targets
promotion_actions
```

2.4 Runtime / Audit Tables

```
promotion_usages
promotion_calculation_logs
```

---

3. ERD Overview

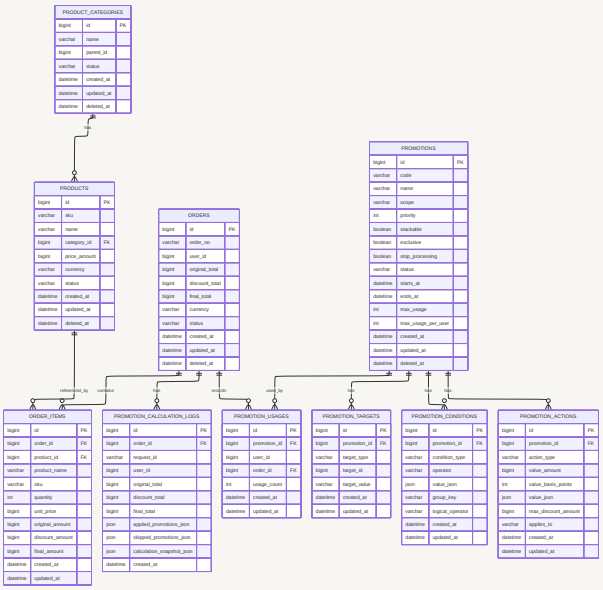
```
erDiagram
    PRODUCT_CATEGORIES ||--o{ PRODUCTS : has
    ORDERS ||--o{ ORDER_ITEMS : contains
    PRODUCTS ||--o{ ORDER_ITEMS : referenced_by

    PROMOTIONS ||--o{ PROMOTION_CONDITIONS : has
    PROMOTIONS ||--o{ PROMOTION_TARGETS : has
    PROMOTIONS ||--o{ PROMOTION_ACTIONS : has
    PROMOTIONS ||--o{ PROMOTION_USAGES : used_by
```

```
ORDERS ||--o{ PROMOTION_USAGES : records
ORDERS ||--o{ PROMOTION_CALCULATION_LOGS : has

PRODUCT_CATEGORIES {
    bigint id PK
    varchar name
    bigint parent_id
    varchar status
    datetime created_at
    datetime updated_at
    datetime deleted_at
}

PRODUCTS {
    bigint id PK
    varchar sku
    varchar name
    bigint category_id FK
    bigint price_amount
    varchar currency
    varchar status
    datetime created_at
    datetime updated_at
    datetime deleted_at
}

ORDERS {
    bigint id PK
    varchar order_no
    bigint user_id
    bigint original_total
    bigint discount_total
    bigint final_total
    varchar currency
    varchar status
```

```
        datetime created_at
        datetime updated_at
        datetime deleted_at
    }

    ORDER_ITEMS {
        bigint id PK
        bigint order_id FK
        bigint product_id FK
        varchar product_name
        varchar sku
        int quantity
        bigint unit_price
        bigint original_amount
        bigint discount_amount
        bigint final_amount
        datetime created_at
        datetime updated_at
    }

    PROMOTIONS {
        bigint id PK
        varchar code
        varchar name
        varchar scope
        int priority
        boolean stackable
        boolean exclusive
        boolean stop_processing
        varchar status
        datetime starts_at
        datetime ends_at
        int max_usage
        int max_usage_per_user
        datetime created_at
        datetime updated_at
```

```
        datetime deleted_at
    }

    PROMOTION_CONDITIONS {
        bigint id PK
        bigint promotion_id FK
        varchar condition_type
        varchar operator
        json value_json
        varchar group_key
        varchar logical_operator
        datetime created_at
        datetime updated_at
    }

    PROMOTION_TARGETS {
        bigint id PK
        bigint promotion_id FK
        varchar target_type
        bigint target_id
        varchar target_value
        datetime created_at
        datetime updated_at
    }

    PROMOTION_ACTIONS {
        bigint id PK
        bigint promotion_id FK
        varchar action_type
        bigint value_amount
        int value_basis_points
        json value_json
        bigint max_discount_amount
        varchar applies_to
        datetime created_at
        datetime updated_at
```

```
}

PROMOTION_USAGES {
    bigint id PK
    bigint promotion_id FK
    bigint user_id
    bigint order_id FK
    int usage_count
    datetime created_at
    datetime updated_at
}

PROMOTION_CALCULATION_LOGS {
    bigint id PK
    bigint order_id FK
    varchar request_id
    bigint user_id
    bigint original_total
    bigint discount_total
    bigint final_total
    json applied_promotions_json
    json skipped_promotions_json
    json calculation_snapshot_json
    datetime created_at
}
```



---

4. Money Design

ระบบต้องเก็บเงินเป็น `BIGINT` หน่วยเล็กสุด เช่น **satang**

```
100 บาท = 10000 satang
1,000 บาท = 100000 satang
```

เหตุผล:

- ป้องกันปัญหา Floating Point Precision
- คำนวณง่าย
- เปรียบเทียบง่าย
- เหมาะกับ Go `int64`
- เหมาะกับ MySQL `BIGINT`

ไม่ควรใช้:

```
FLOAT
DOUBLE
```

ควรใช้:

```
BIGINT
```

สำหรับ Percentage แนะนำใช้ **basis points**

```
10% = 1000 basis points
10.50% = 1050 basis points
100% = 10000 basis points
```

---

5. Product Tables

5.1 `product_categories`

ใช้เก็บหมวดหมู่สินค้า เพื่อรองรับ Promotion ตาม Category ในอนาคต

Column	Type	Required	Description
`id`	BIGINT UNSIGNED	Yes	Primary Key
`name`	VARCHAR(255)	Yes	ชื่อหมวดหมู่
`parent_id`	BIGINT UNSIGNED	No	หมวดหมู่แม่
`status`	VARCHAR(30)	Yes	ACTIVE / INACTIVE
`created_at`	DATETIME	Yes	วันที่สร้าง
`updated_at`	DATETIME	Yes	วันที่แก้ไข
`deleted_at`	DATETIME	No	Soft delete

เหตุผลที่ต้องมี:

```
อนาคตอาจมี Promotion เช่น ลด 10% สำหรับสินค้าหมวด Electronics
```

5.2 `products`

ใช้เก็บข้อมูลสินค้า

Column	Type	Required	Description
`id`	BIGINT UNSIGNED	Yes	Primary Key
`sku`	VARCHAR(100)	Yes	รหัสสินค้า
`name`	VARCHAR(255)	Yes	ชื่อสินค้า
`category_id`	BIGINT UNSIGNED	Yes	อ้างอิง product_categories
`price_amount`	BIGINT	Yes	ราคาสินค้า หน่วย satang
`currency`	VARCHAR(10)	Yes	THB / USD
`status`	VARCHAR(30)	Yes	ACTIVE / INACTIVE
`created_at`	DATETIME	Yes	วันที่สร้าง
`updated_at`	DATETIME	Yes	วันที่แก้ไข
`deleted_at`	DATETIME	No	Soft delete

ตัวอย่าง:

```
id = 1
sku = PRODUCT-001
name = Product 1
price_amount = 100000
currency = THB
```

หมายความว่า:

```
Product 1 ราคา 1,000 บาท
```

Index ที่ควรมี:

```
CREATE UNIQUE INDEX idx_products_sku ON products(sku);
CREATE INDEX idx_products_category_id ON products(category_id);
CREATE INDEX idx_products_status ON products(status);
```

---

6. Order Tables

6.1 `orders`

ใช้เก็บข้อมูลคำสั่งซื้อหลัก

Column	Type	Required	Description
`id`	BIGINT UNSIGNED	Yes	Primary Key
`order_no`	VARCHAR(100)	Yes	เลข Order
`user_id`	BIGINT UNSIGNED	Yes	รหัส User
`original_total`	BIGINT	Yes	ยอดรวมก่อนลด
`discount_total`	BIGINT	Yes	ส่วนลดรวม
`final_total`	BIGINT	Yes	ยอดสุทธิ
`currency`	VARCHAR(10)	Yes	สกุลเงิน
`status`	VARCHAR(30)	Yes	DRAFT / CONFIRMED / PAID / CANCELLED
`created_at`	DATETIME	Yes	วันที่สร้าง
`updated_at`	DATETIME	Yes	วันที่แก้ไข
`deleted_at`	DATETIME	No	Soft delete

เหตุผลที่ต้องแยกยอด:

```
original_total = ตรวจยอดก่อนลด
discount_total = ตรวจส่วนลดรวม
final_total    = ตรวจยอดที่ลูกค้าต้องจ่ายจริง
```

Index:

```sql
CREATE UNIQUE INDEX idx_orders_order_no ON orders(order_no);
CREATE INDEX idx_orders_user_id ON orders(user_id);
CREATE INDEX idx_orders_status ON orders(status);
CREATE INDEX idx_orders_created_at ON orders(created_at);
```

6.2 `order_items`

ใช้เก็บสินค้าแต่ละรายการใน Order

Column	Type	Required	Description
`id`	BIGINT UNSIGNED	Yes	Primary Key
`order_id`	BIGINT UNSIGNED	Yes	อ้างอิง orders
`product_id`	BIGINT UNSIGNED	Yes	อ้างอิง products
`product_name`	VARCHAR(255)	Yes	Snapshot ชื่อสินค้า
`sku`	VARCHAR(100)	Yes	Snapshot SKU
`quantity`	INT	Yes	จำนวนสินค้า
`unit_price`	BIGINT	Yes	ราคาต่อชิ้นก่อนลด
`original_amount`	BIGINT	Yes	`unit_price * quantity`
`discount_amount`	BIGINT	Yes	ส่วนลดของ Item
`final_amount`	BIGINT	Yes	ราคาสุทธิของ Item
`created_at`	DATETIME	Yes	วันที่สร้าง
`updated_at`	DATETIME	Yes	วันที่แก้ไข

ทำไมต้อง Snapshot?

เพราะข้อมูลสินค้าอาจเปลี่ยนภายหลัง เช่น:

- ชื่อสินค้าเปลี่ยน
- ราคาเปลี่ยน
- SKU เปลี่ยน

แต่ Order เก่าต้องยังแสดงข้อมูล ณ ตอนที่สั่งซื้อได้ถูกต้อง

Index:

```
CREATE INDEX idx_order_items_order_id ON order_items(order_i
d);
CREATE INDEX idx_order_items_product_id ON order_items(produc
t_id);
```

---

7. Promotion Configuration Tables

หัวใจของ Database Design คือไม่สร้าง Table แยกตาม Promotion Type เช่น:

```
percentage_promotions
fixed_amount_promotions
bogo_promotions
```

เพราะจะขยายยาก

ควรใช้โครงสร้าง:

```
promotions
promotion_conditions
promotion_targets
promotion_actions
```

7.1 `promotions`

ใช้เก็บข้อมูลหลักของ Promotion

Column	Type	Required	Description
`id`	BIGINT UNSIGNED	Yes	Primary Key
`code`	VARCHAR(100)	No	Promotion Code / Coupon Code
`name`	VARCHAR(255)	Yes	ชื่อ Promotion
`description`	TEXT	No	รายละเอียด
`scope`	VARCHAR(30)	Yes	ITEM / CART / COUPON / SHIPPING
`priority`	INT	Yes	ลำดับคำนวณ
`stackable`	BOOLEAN	Yes	ใช้ร่วมกับ Promotion อื่นได้ไหม
`exclusive`	BOOLEAN	Yes	ใช้แล้วห้าม Promotion อื่นหรือไม่
`stop_processing`	BOOLEAN	Yes	ใช้แล้วหยุด Rule ถัดไปหรือไม่
`status`	VARCHAR(30)	Yes	ACTIVE / INACTIVE

Column	Type	Required	Description
`starts_at`	DATETIME	Yes	วันที่เริ่ม
`ends_at`	DATETIME	Yes	วันที่หมดอายุ
`max_usage`	INT	No	จำนวนครั้งสูงสุดทั้งหมด
`max_usage_per_user`	INT	No	จำนวนครั้งสูงสุดต่อ User
`created_at`	DATETIME	Yes	วันที่สร้าง
`updated_at`	DATETIME	Yes	วันที่แก้ไข
`deleted_at`	DATETIME	No	Soft delete

ตัวอย่าง Promotion A:

```
Product 1 ลด 10%

scope = ITEM
priority = 1
stackable = true
exclusive = false
stop_processing = false
status = ACTIVE
```

Index:

```sql
CREATE UNIQUE INDEX idx_promotions_code ON promotions(code);
CREATE INDEX idx_promotions_active
ON promotions(status, starts_at, ends_at, priority);
CREATE INDEX idx_promotions_scope_priority
ON promotions(scope, priority);
```

หมายเหตุ: ถ้า `code` อนุญาตให้ NULL หลายแถว ใน MySQL สามารถมี NULL ซ้ำได้ แต่ต้องระวังตอนใช้ Coupon Code จริง

7.2 `promotion_conditions`

ใช้เก็บเงื่อนไขว่า Promotion ใช้ได้เมื่อไหร่

Column	Type	Required	Description
`id`	BIGINT UNSIGNED	Yes	Primary Key
`promotion_id`	BIGINT UNSIGNED	Yes	FK ไป promotions
`condition_type`	VARCHAR(50)	Yes	ประเภท Condition
`operator`	VARCHAR(30)	Yes	วิธีเปรียบเทียบ
`value_json`	JSON	Yes	ค่า Condition
`group_key`	VARCHAR(50)	No	ใช้จัดกลุ่ม Condition
`logical_operator`	VARCHAR(10)	Yes	AND / OR
`created_at`	DATETIME	Yes	วันที่สร้าง
`updated_at`	DATETIME	Yes	วันที่แก้ไข

ตัวอย่าง `condition_type`:

```
PRODUCT_ID
CATEGORY_ID
MIN_ORDER_AMOUNT
MAX_ORDER_AMOUNT
COUPON_CODE
USER_SEGMENT
FIRST_ORDER
PAYMENT_METHOD
```

ตัวอย่าง `operator`:

```
EQ
NEQ
IN
NOT_IN
GTE
LTE
BETWEEN
```

ตัวอย่าง Condition: ซื้อครบ 1,000 บาท

```
{
  "condition_type": "MIN_ORDER_AMOUNT",
  "operator": "GTE",
```

```
  "value_json": {
    "amount": 100000
  },
  "logical_operator": "AND"
}
```

เหตุผลที่ใช้ `value_json`:

```
Condition แต่ละประเภทใช้ข้อมูลไม่เหมือนกัน
การใช้ JSON ทำให้เพิ่ม Condition ใหม่ได้โดยไม่ต้องแก้ Schema
```

Index:

```
CREATE INDEX idx_promotion_conditions_promotion_id
ON promotion_conditions(promotion_id);

CREATE INDEX idx_promotion_conditions_type
ON promotion_conditions(condition_type);
```

7.3 `promotion_targets`

ใช้ระบุว่า Promotion นี้ Apply กับอะไร

Column	Type	Required	Description
`id`	BIGINT UNSIGNED	Yes	Primary Key
`promotion_id`	BIGINT UNSIGNED	Yes	FK ไป promotions
`target_type`	VARCHAR(50)	Yes	PRODUCT / CATEGORY / CART / USER_SEGMENT
`target_id`	BIGINT UNSIGNED	No	ID ของ Target
`target_value`	VARCHAR(255)	No	ใช้กรณี Target เป็น String
`created_at`	DATETIME	Yes	วันที่สร้าง

Column	Type	Required	Description
`updated_at`	DATETIME	Yes	วันที่แก้ไข

ตัวอย่าง Target:

Product 1:

```
{
  "target_type": "PRODUCT",
  "target_id": 1
}
```

Category 10:

```
{
  "target_type": "CATEGORY",
  "target_id": 10
}
```

Cart ทั้งหมด:

```
{
  "target_type": "CART",
  "target_id": null
}
```

Target ต่างจาก Condition อย่างไร?

```
Target = Promotion ใช้กับอะไร
Condition = Promotion ใช้ได้เมื่อไหร่
```

ตัวอย่าง:

```
ลด 10% สินค้าหมวด Electronics เมื่อซื้อครบ 1,000 บาท

Target    = CATEGORY Electronics
```

```
Condition = MIN_ORDER_AMOUNT >= 1,000
Action    = PERCENTAGE_DISCOUNT 10%
```

Index:

```
CREATE INDEX idx_promotion_targets_promotion_id
ON promotion_targets(promotion_id);

CREATE INDEX idx_promotion_targets_lookup
ON promotion_targets(target_type, target_id);
```

7.4 `promotion_actions`

ใช้เก็บว่า Promotion นี้ลดอย่างไร

Column	Type	Required	Description
`id`	BIGINT UNSIGNED	Yes	Primary Key
`promotion_id`	BIGINT UNSIGNED	Yes	FK ไป promotions
`action_type`	VARCHAR(50)	Yes	ประเภท Action
`value_amount`	BIGINT	No	จำนวนเงินที่ลด หน่วย satang
`value_basis_points`	INT	No	เปอร์เซ็นต์แบบ basis points
`value_json`	JSON	No	Config สำหรับ Action ซับซ้อน
`max_discount_amount`	BIGINT	No	จำกัดส่วนลดสูงสุด
`applies_to`	VARCHAR(30)	Yes	ITEM / CART / SHIPPING
`created_at`	DATETIME	Yes	วันที่สร้าง
`updated_at`	DATETIME	Yes	วันที่แก้ไข

ตัวอย่าง `action_type`:

```
PERCENTAGE_DISCOUNT
FIXED_AMOUNT_DISCOUNT
FREE_SHIPPING
BUY_X_GET_Y
BUNDLE_DISCOUNT
```

ตัวอย่างลด 10%:

```
{
  "action_type": "PERCENTAGE_DISCOUNT",
  "value_basis_points": 1000,
  "value_amount": null,
  "applies_to": "ITEM"
}
```

ตัวอย่างลด 100 บาท:

```
{
  "action_type": "FIXED_AMOUNT_DISCOUNT",
  "value_amount": 10000,
  "value_basis_points": null,
  "applies_to": "ITEM"
}
```

ตัวอย่าง Buy 2 Get 1:

```
{
  "action_type": "BUY_X_GET_Y",
  "value_json": {
    "buy_qty": 2,
    "free_qty": 1
  },
  "applies_to": "ITEM"
}
```

เหตุผลที่มี `value_amount`, `value_basis_points`, `value_json`:

```
Promotion แต่ละแบบใช้ข้อมูลคนละชนิด
Fixed Amount ใช้ value_amount
Percentage ใช้ value_basis_points
Complex Promotion ใช้ value_json
```

Index:

```sql
CREATE INDEX idx_promotion_actions_promotion_id
ON promotion_actions(promotion_id);

CREATE INDEX idx_promotion_actions_type
ON promotion_actions(action_type);
```

---

8. Runtime / Audit Tables

8.1 `promotion_usages`

ใช้เก็บจำนวนครั้งที่ Promotion ถูกใช้จริง

Column	Type	Required	Description
`id`	BIGINT UNSIGNED	Yes	Primary Key
`promotion_id`	BIGINT UNSIGNED	Yes	FK ไป promotions
`user_id`	BIGINT UNSIGNED	No	User ที่ใช้ Promotion
`order_id`	BIGINT UNSIGNED	No	Order ที่ใช้ Promotion
`usage_count`	INT	Yes	จำนวนครั้งที่ใช้
`created_at`	DATETIME	Yes	วันที่สร้าง
`updated_at`	DATETIME	Yes	วันที่แก้ไข

ใช้สำหรับ:

```
max_usage
max_usage_per_user
coupon usage limit
```

Index:

```sql
CREATE INDEX idx_promotion_usages_promotion_id
ON promotion_usages(promotion_id);
```

```sql
CREATE INDEX idx_promotion_usages_promo_user
ON promotion_usages(promotion_id, user_id);

CREATE INDEX idx_promotion_usages_order_id
ON promotion_usages(order_id);
```

หมายเหตุ Production:

```
ตอน Confirm Order ต้องตรวจและ Consume Usage ภายใต้ Transaction
```

8.2 `promotion_calculation_logs`

ใช้เก็บ Snapshot การคำนวณ เพื่อ Debug และ Audit

Column	Type	Required	Description
`id`	BIGINT UNSIGNED	Yes	Primary Key
`order_id`	BIGINT UNSIGNED	No	อ้างอิง Order ถ้ามี
`request_id`	VARCHAR(100)	Yes	Trace Request
`user_id`	BIGINT UNSIGNED	No	User
`original_total`	BIGINT	Yes	ยอดก่อนลด
`discount_total`	BIGINT	Yes	ส่วนลดรวม
`final_total`	BIGINT	Yes	ยอดสุทธิ
`applied_promotions_json`	JSON	Yes	Promotion ที่ถูก Apply
`skipped_promotions_json`	JSON	Yes	Promotion ที่ถูก Skip
`calculation_snapshot_json`	JSON	Yes	Snapshot ทั้งหมด
`created_at`	DATETIME	Yes	วันที่สร้าง

ตัวอย่าง Applied Promotion:

```json
{
  "promotion_id": 1,
  "code": "ITEM1_10PERCENT",
  "scope": "ITEM",
```

```
  "before_amount": 100000,
  "discount_amount": 10000,
  "after_amount": 90000
}
```

ตัวอย่าง Skipped Promotion:

```
{
  "promotion_id": 3,
  "code": "CART_1000_MINUS_100",
  "reason": "min_order_amount_not_reached"
}
```

เหตุผลที่ต้องมี Calculation Log:

- Debug ราคาย้อนหลัง
- Support ลูกค้า
- Audit
- Replay Calculation
- ตรวจสอบว่า Promotion ใดถูก Apply หรือ Skip

Index:

```
CREATE INDEX idx_calculation_logs_order_id
ON promotion_calculation_logs(order_id);

CREATE INDEX idx_calculation_logs_user_id
ON promotion_calculation_logs(user_id);

CREATE INDEX idx_calculation_logs_request_id
ON promotion_calculation_logs(request_id);

CREATE INDEX idx_calculation_logs_created_at
ON promotion_calculation_logs(created_at);
```

---

9. Example: Store Promotion ตามโจทย์

Product Data

```
Product 1:
id = 1
price_amount = 100000
meaning = 1,000 บาท

Product 2:
id = 2
price_amount = 50000
meaning = 500 บาท
```

Promotion A: Product 1 ลด 10%

promotions

```
id = 1
code = ITEM1_10PERCENT
name = Product 1 Discount 10%
scope = ITEM
priority = 1
stackable = true
exclusive = false
stop_processing = false
status = ACTIVE
```

promotion_targets

```
promotion_id = 1
target_type = PRODUCT
target_id = 1
```

promotion_actions

```
promotion_id = 1
action_type = PERCENTAGE_DISCOUNT
value_basis_points = 1000
applies_to = ITEM
```

Promotion B: Product 2 ลด 100 บาท

promotions

```
id = 2
code = ITEM2_100THB
name = Product 2 Discount 100 THB
scope = ITEM
priority = 1
stackable = true
exclusive = false
stop_processing = false
status = ACTIVE
```

promotion_targets

```
promotion_id = 2
target_type = PRODUCT
target_id = 2
```

promotion_actions

```
promotion_id = 2
action_type = FIXED_AMOUNT_DISCOUNT
value_amount = 10000
applies_to = ITEM
```

Calculation Result

```
Product 1:
original = 100000
discount = 10000
final = 90000

Product 2:
original = 50000
discount = 10000
final = 40000

Order:
original_total = 150000
discount_total = 20000
final_total = 130000
```

แปลว่า:

```
Product 1: 1,000 - 100 = 900
Product 2: 500 - 100 = 400
Final Total = 1,300 บาท
```

---

10. How This Design Supports New Promotions

Case 1: Cart Discount ซื้อครบ 1,000 ลด 100

ไม่ต้องเพิ่ม Table ใหม่

เพิ่มข้อมูลใน:

```
promotions
promotion_conditions
promotion_targets
promotion_actions
```

Condition:

```json
{
  "condition_type": "MIN_ORDER_AMOUNT",
  "operator": "GTE",
  "value_json": {
    "amount": 100000
  }
}
```

Action:

```json
{
  "action_type": "FIXED_AMOUNT_DISCOUNT",
  "value_amount": 10000,
  "applies_to": "CART"
}
```

Case 2: Category Discount ลด 5%

Target:

```json
{
  "target_type": "CATEGORY",
  "target_id": 10
}
```

Action:

```json
{
  "action_type": "PERCENTAGE_DISCOUNT",
  "value_basis_points": 500,
  "applies_to": "ITEM"
}
```

Case 3: Buy 2 Get 1

Action:

```
{
  "action_type": "BUY_X_GET_Y",
  "value_json": {
    "buy_qty": 2,
    "free_qty": 1
  },
  "applies_to": "ITEM"
}
```

ฝั่ง Code เพิ่ม:

```
BuyXGetYStrategy
```

โดยไม่ต้องแก้ Table หลัก

---

11. Validation Rules

promotions

- `name` ต้องไม่ว่าง
- `scope` ต้องเป็น ITEM / CART / COUPON / SHIPPING
- `priority` ต้องมากกว่าหรือเท่ากับ 0
- `starts_at` ต้องน้อยกว่า `ends_at`
- `status` ต้องเป็น ACTIVE / INACTIVE
- `max_usage` ถ้ามี ต้องมากกว่า 0
- `max_usage_per_user` ถ้ามี ต้องมากกว่า 0

promotion_actions

- `action_type` ต้องเป็น Type ที่ระบบรองรับ
- ถ้า `PERCENTAGE_DISCOUNT` ต้องมี `value_basis_points`
- `value_basis_points` ต้องมากกว่า 0 และไม่เกิน 10000
- ถ้า `FIXED_AMOUNT_DISCOUNT` ต้องมี `value_amount`
- `value_amount` ต้องมากกว่า 0
- `max_discount_amount` ถ้ามี ต้องมากกว่า 0

promotion_conditions

- `condition_type` ต้องเป็น Type ที่รองรับ
- `operator` ต้องเป็น Operator ที่รองรับ
- `value_json` ต้อง Parse ได้
- JSON schema ต้องตรงกับ `condition_type`

promotion_targets

- `target_type` ต้องเป็น Type ที่รองรับ
- ถ้า `target_type = PRODUCT` ต้องมี `target_id`
- ถ้า `target_type = CATEGORY` ต้องมี `target_id`
- ถ้า `target_type = CART` สามารถไม่มี `target_id` ได้

---

12. Query Pattern

Load Products

```sql
SELECT *
FROM products
WHERE id IN (...)
  AND status = 'ACTIVE';
```

Load Active Promotions

```sql
SELECT *
FROM promotions
WHERE status = 'ACTIVE'
  AND starts_at <= NOW()
  AND ends_at >= NOW()
ORDER BY scope, priority, created_at, id;
```

ควร Preload:

```
conditions
targets
actions
```

Save Calculation Log

```sql
INSERT INTO promotion_calculation_logs (
  request_id,
  user_id,
  original_total,
  discount_total,
  final_total,
  applied_promotions_json,
  skipped_promotions_json,
  calculation_snapshot_json,
  created_at
)
VALUES (...);
```

---

13. Index Strategy

Index สำคัญสำหรับ Performance:

```sql
CREATE INDEX idx_promotions_active
ON promotions(status, starts_at, ends_at, priority);

CREATE INDEX idx_promotions_scope_priority
ON promotions(scope, priority);

CREATE INDEX idx_promotion_conditions_promotion_id
ON promotion_conditions(promotion_id);

CREATE INDEX idx_promotion_actions_promotion_id
ON promotion_actions(promotion_id);

CREATE INDEX idx_promotion_targets_lookup
ON promotion_targets(target_type, target_id);

CREATE INDEX idx_promotion_usages_promo_user
ON promotion_usages(promotion_id, user_id);

CREATE INDEX idx_calculation_logs_request_id
ON promotion_calculation_logs(request_id);
```

---

14. Things We Should Not Do

14.1 ไม่ควร Hardcode Product ID ใน Code

ไม่ดี:

```
if product_id == 1:
    discount = price * 10%
```

ดี:

```
promotion_targets ระบุ product_id
promotion_actions ระบุ discount type
```

```
strategy เป็นคนคำนวณ
```

14.2 ไม่ควรสร้าง Table แยกตาม Promotion Type

ไม่ดี:

```
percentage_promotions
fixed_amount_promotions
buy_x_get_y_promotions
```

ดี:

```
promotions
promotion_conditions
promotion_targets
promotion_actions
```

14.3 ไม่ควรใช้ Float กับเงิน

ไม่ดี:

```
price = 1000.50
```

ดี:

```
price_amount = 100050
```

14.4 ไม่ควร Query DB ใน Loop

ไม่ดี:

```
for each item:
```

```
    query promotions
```

ดี:

```
load active promotions once
build map in memory
calculate in memory
```

---

15. Final Database Design Summary

Database Design นี้ใช้แนวคิด **Rule-based Promotion Config**

```
promotions             = ข้อมูลหลักของ Promotion
promotion_conditions   = เงื่อนไขการใช้งาน
promotion_targets      = ใช้กับอะไร
promotion_actions      = ลดอย่างไร
promotion_usages       = ใช้ไปกี่ครั้ง
promotion_logs         = ตรวจสอบย้อนหลัง
```

ข้อดีของ Design นี้:

- รองรับ Product 1 ลด 10%
- รองรับ Product 2 ลด 100 บาท
- รองรับ Cart-level Promotion
- รองรับ Promotion ซ้อนกันด้วย Priority / Stackable / Exclusive
- เพิ่ม Promotion ใหม่ได้โดยไม่ต้องแก้ Schema บ่อย
- ใช้กับ MySQL + GORM ได้ง่าย
- Debug และ Audit ได้
- Scale ได้ด้วย Index, Preload และ Cache
- ไม่ผูก Business Logic กับ Database Table เฉพาะ Promotion Type

---

สรุปสั้นสำหรับใส่สไลด์

> Database ถูกออกแบบเป็น Rule-based Schema โดยแยก Promotion ออกเป็น Metadata, Condition, Target และ Action ทำให้สามารถเพิ่ม Promotion ใหม่ เช่น Cart Discount, Category Discount หรือ Buy X Get Y ได้ผ่าน Configuration และ Strategy ใหม่ โดยไม่ต้องสร้าง Table ใหม่ทุกครั้ง พร้อมรองรับ Calculation Log เพื่อ Audit และ Debug ราคาย้อนหลังได้.