Détail Rapport Technique : Base de Données E-commerce SportShop

Table des matières

5. Workflows d’Utilisation Typiques

5.1 Inscription et Authentification
5.2 Navigation et Sélection de Produits
5.3 Processus d’Achat
5.4 Post-vente et Support Client

6. Conclusion et Recommandations

6.1 Points Forts de l’Architecture
6.2 Améliorations Potentielles
6.3 Bonnes Pratiques Recommandées
6.4 Checklist de Déploiement

1. Introduction et Contexte du Projet :

1.1 Objectif du Projet:

La base de données **SportShop** a été conçue pour supporter une plateforme e-commerce spécialisée dans les produits sportifs. Elle doit gérer un catalogue de produits variés, des utilisateurs avec différents rôles, un système de commandes complet et des fonctionnalités avancées comme les favoris et les évaluations.

1.2 Portée et Fonctionnalités:

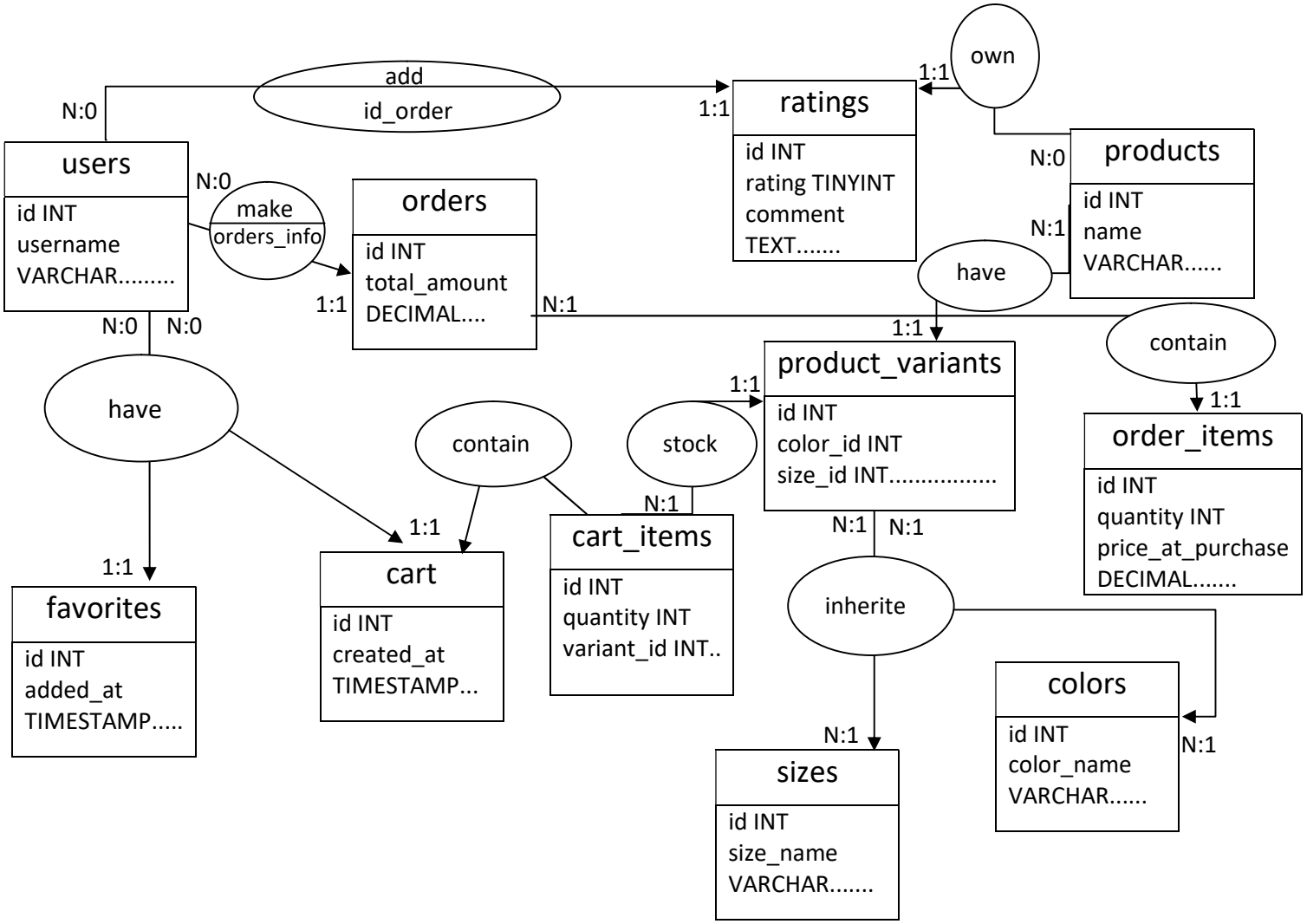
- Gestion multi-utilisateurs (clients et administrateurs)
- Authentification double (email/mot de passe et Google OAuth)
- Catalogue de produits avec variations de couleurs et tailles
- Système de panier d'achat
- Processus de commande complet
- Système d'évaluation et de commentaires
- Gestion des favoris/wishlist
- Suivi d'inventaire détaillé
- Historique des commandes et statuts

1.3 Technologies Utilisées:

- **SGBD**: MySQL 8.0+
- **Moteur de Stockage**: InnoDB
- **Contraintes**: Clés étrangères, contraintes UNIQUE, CHECK
- **Indexation**: Index primaires, secondaires et composites
- **Types de Données**: JSON pour flexibilité, ENUM pour validation

2. Architecture Globale de la Base de Données :

2.1 Schéma Relationnel:



2.2 Groupes de Tables:

- **Gestion Utilisateurs**: users, favorites.
- **Catalogue Produits**: products, product_variants, colors, sizes.
- **Processus Achat**: cart, cart_items, orders, order_items.
- **Post-vente**: ratings, orders_info, order_status_history, order_confirmations .

3. Analyse Détaillée des Tables :

3.1 Table users - Gestion des Utilisateurs:

```
CREATE TABLE users (
   id INT PRIMARY KEY AUTO_INCREMENT,
   username VARCHAR(50) UNIQUE NOT NULL,
   email VARCHAR(100) UNIQUE NOT NULL,
   password VARCHAR(255),
   auth_provider ENUM('email', 'google') DEFAULT 'email',
   google_id VARCHAR(100) UNIQUE,
   user_type ENUM('customer', 'admin') DEFAULT 'customer',
   verification_token VARCHAR(255),
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Table centrale pour tous les utilisateurs du système.

Attributs Clés:
auth_provider: Support d'authentification multiple (email/Google).
google_id: Identifiant unique pour les utilisateurs Google.
verification_token: Pour la vérification d'email.
user_type: Différenciation clients/administrateurs.
Index:
UNIQUE(username): Garantit l'unicité des noms d'utilisateur
UNIQUE(email): Empêche les doublons d'email
UNIQUE(google_id): Identifiants Google uniques

3.2 Tables colors et sizes - Gestion des Attributs Produits:

```
CREATE TABLE colors (
   id INT PRIMARY KEY AUTO_INCREMENT,
   color_name VARCHAR(50) NOT NULL UNIQUE,
   color_code VARCHAR(7),
```

```
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE sizes (
  id INT PRIMARY KEY AUTO_INCREMENT,
  size_name VARCHAR(20) NOT NULL UNIQUE,
  size_type ENUM('clothing', 'shoes', 'unisex') DEFAULT 'unisex',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Tables de référence pour normalisation des données.

Avantages:

- Évite la redondance des données
- Facilite les mises à jour globales
- Permet des jointures efficaces
- Supporte l'internationalisation future

3.3 Table products - Catalogue Principal:

```
CREATE TABLE products (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(100) NOT NULL,
  description TEXT,
  base_price DECIMAL(10, 2) NOT NULL,
  product_type ENUM('clothing', 'accessories') NOT NULL,
  sub_type VARCHAR(50) NOT NULL,
  image_url VARCHAR(255),
  is_active BOOLEAN DEFAULT TRUE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Hiérarchie de Produits:

Niveau 1: product_type (clothing, accessories)
Niveau 2: sub_type (sneakers, shirts, nutrition, tools, etc.)
Niveau 3: Variations dans product_variants
Gestion d'État:
is_active = TRUE: Produit visible sur le site
is_active = FALSE: Produit masqué (soft delete)

3.4 Table product_variants - Gestion des Variations:

```
CREATE TABLE product_variants (
   id INT PRIMARY KEY AUTO_INCREMENT,
   product_id INT NOT NULL,
   color_id INT,
   size_id INT,
   sku VARCHAR(50) UNIQUE NOT NULL,
   price_adjustment DECIMAL(10, 2) DEFAULT 0,
   stock_quantity INT DEFAULT 0,
   image_url VARCHAR(255),
   is_active BOOLEAN DEFAULT TRUE,

   FOREIGN KEY (product_id) REFERENCES products(id) ON DELETE CASCADE,
   FOREIGN KEY (color_id) REFERENCES colors(id) ON DELETE SET NULL,
   FOREIGN KEY (size_id) REFERENCES sizes(id) ON DELETE SET NULL,
   UNIQUE KEY unique_product_variant (product_id, color_id, size_id)
);
```

Séparation des variations de stock et prix.

Cas d'Usage:

- Même produit, couleurs différentes avec stocks séparés
- Ajustements de prix par taille/couleur
- Images spécifiques par variation
- Gestion indépendante du statut actif

Contraintes Uniques:

- UNIQUE(sku): Code produit unique global
- UNIQUE(product_id, color_id, size_id): Évite les doublons de combinaisons

3.5 Tables cart et cart_items - Panier d'Achat:

```
CREATE TABLE cart (
   id INT PRIMARY KEY AUTO_INCREMENT,
   user_id INT UNIQUE NOT NULL,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

   FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

CREATE TABLE cart_items (
```

```
  id INT PRIMARY KEY AUTO_INCREMENT,
  cart_id INT NOT NULL,
  variant_id INT NOT NULL,
  quantity INT NOT NULL DEFAULT 1,

  FOREIGN KEY (cart_id) REFERENCES cart(id) ON DELETE CASCADE,
  FOREIGN KEY (variant_id) REFERENCES product_variants(id) ON DELETE
CASCADE,
  UNIQUE KEY unique_cart_variant (cart_id, variant_id)
);
```

One-to-one User-Cart avec One-to-many Cart-Items.

Fonctionnalités:

- Panier persistant par utilisateur
- Prévention des doublons dans le panier
- Quantités ajustables
- Suppression en cascade lors de la suppression d'utilisateur

3.6 Tables orders et order_items - Gestion des Commandes:

```
CREATE TABLE orders (
  id INT PRIMARY KEY AUTO_INCREMENT,
  user_id INT NOT NULL,
  order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  total_amount DECIMAL(10, 2) NOT NULL,
  status ENUM('pending', 'processing', 'shipped', 'delivered', 'cancelled')
DEFAULT 'pending',
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

CREATE TABLE order_items (
  id INT PRIMARY KEY AUTO_INCREMENT,
  order_id INT NOT NULL,
  variant_id INT NOT NULL,
  quantity INT NOT NULL,
  price_at_purchase DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (order_id) REFERENCES orders(id) ON DELETE CASCADE,
  FOREIGN KEY (variant_id) REFERENCES product_variants(id) ON DELETE
RESTRICT
);
```

Préservation des Données:

- price_at_purchase: Capture le prix au moment de l'achat
- DELETE RESTRICT sur variant_id: Empêche la suppression de variants référencés

Workflow des Statuts:

pending → Commande créée
processing → En préparation
shipped → Expédiée
delivered → Livrée
cancelled → Annulée

3.7 Table ratings - Système d'Évaluation:

```
CREATE TABLE ratings (
   id INT PRIMARY KEY AUTO_INCREMENT,
   user_id INT NOT NULL,
   product_id INT NOT NULL,
   order_id INT NOT NULL,
   rating TINYINT NOT NULL CHECK (rating >= 0 AND rating <= 5),
   comment TEXT,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
   FOREIGN KEY (product_id) REFERENCES products(id) ON DELETE CASCADE,
   FOREIGN KEY (order_id) REFERENCES orders(id) ON DELETE CASCADE,
   UNIQUE KEY unique_user_product_rating (user_id, product_id, order_id)
);
```

Sécurité Intégrée:
Contrainte CHECK: Notes entre 0 et 5
UNIQUE(user_id, product_id, order_id): Une évaluation par achat
Référence à order_id: Garantit l'achat effectif

3.8 Tables de Support (orders_info, order_status_history, order_confirmations):

Table orders_info:

```
CREATE TABLE orders_info (
   id INT PRIMARY KEY AUTO_INCREMENT,
   order_id INT NOT NULL,
```

```
    user_id INT NOT NULL,
    customer_name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    wilaya VARCHAR(50) NOT NULL,
    commune VARCHAR(50) NOT NULL,
    address TEXT NOT NULL,
    instructions TEXT,
    delivery_type VARCHAR(20) DEFAULT 'standard',
    payment_method VARCHAR(20) DEFAULT 'cash_on_delivery',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (order_id) REFERENCES orders(id) ON DELETE CASCADE,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

Spécificités Régionales:

- Champs wilaya et commune pour l'Algérie
- *payment_method*: Adapté au contexte local
- Séparation des informations de livraison pour flexibilité

Table order_status_history:

```
CREATE TABLE order_status_history (
    id INT PRIMARY KEY AUTO_INCREMENT,
    order_id INT NOT NULL,
    user_id INT NOT NULL,
    status ENUM('pending', 'processing', 'shipped', 'delivered', 'cancelled')
DEFAULT 'processing',
    changed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    changed_by VARCHAR(50) DEFAULT 'system',
    notes TEXT,
    FOREIGN KEY (order_id) REFERENCES orders(id) ON DELETE CASCADE,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

4. Relations et Contraintes d'Intégrité :

4.1 Relations Principales:

(dans la page de schema)

4.2 Contraintes FOREIGN KEY:

Politiques de Suppression:

- ON DELETE CASCADE: Suppression en cascade (users → cart)
- ON DELETE SET NULL: Mise à NULL préservant l'enregistrement (colors → product_variants)
- ON DELETE RESTRICT: Empêche la suppression (variants → order_items)

Justification des Choix:

- **CASCADE**: Pour les relations fortement dépendantes
- **SET NULL**: Pour les données référentielles optionnelles
- **RESTRICT**: Pour préserver l'intégrité historique

4.3 Contraintes UNIQUE :

-- Empêche les doublons fonctionnels
UNIQUE(username) -- Dans users
UNIQUE(email) -- Dans users
UNIQUE(google_id) -- Dans users
UNIQUE(color_name) -- Dans colors
UNIQUE(size_name) -- Dans sizes
UNIQUE(sku) -- Dans product_variants
UNIQUE(product_id, color_id, size_id) -- Dans product_variants
UNIQUE(user_id, product_id) -- Dans favorites
UNIQUE(cart_id, variant_id) -- Dans cart_items
UNIQUE(user_id, product_id, order_id) -- Dans ratings

4.4 Contraintes CHECK et Validation:

-- Validation des notes
CHECK (rating >= 0 AND rating <= 5) -- Dans ratings

-- Validation via ENUM
ENUM('email', 'google') -- Dans users.auth_provider
ENUM('customer', 'admin') -- Dans users.user_type
ENUM('clothing', 'accessories') -- Dans products.product_type
ENUM('clothing', 'shoes', 'unisex') -- Dans sizes.size_type
ENUM('pending', 'processing', 'shipped', 'delivered', 'cancelled') -- Dans orders.status

5. Workflows d'Utilisation Typiques :

5.1 Inscription et Authentification :

1. User visite le site → Vérifie la session
2. Option Google OAuth → Vérifie/Crée compte dans `users`
3. Option Email/Password → Crée compte avec `verification_token`
4. Email de vérification envoyé → Token validé
5. Panier automatiquement créé dans `cart`

5.2 Navigation et Sélection Produit :

1. User parcourt `products` actifs
2. Filtre par `product_type` et `sub_type`
3. Sélectionne produit → Charge `product_variants`
4. Choisit couleur/taille → Vérifie `stock_quantity`
5. Ajoute aux favoris (`favorites`) ou au panier (`cart_items`)

5.3 Processus d'Achat :

1. User vérifie `cart_items`
2. Passe commande → Crée `orders` avec statut 'pending'
3. Remplit `orders_info` (adresse, contact)
4. Système vérifie stocks → Met à jour `product_variants`
5. Transfert `cart_items` vers `order_items`
6. Vide `cart_items`
7. Envoi confirmation → Enregistre dans `order_confirmations`

5.4 Post-vente et Support :

1. Suivi commande via `order_status_history`
2. Livraison → Statut mis à 'delivered'
3. User peut évaluer → `ratings` avec vérification `order_id`
4. Support client accède à l'historique complet

6. Conclusion et Recommandations :

6.1 Points Forts de l'Architecture:

1. **Normalisation Appropriée** : Équilibre entre normalisation et performance
2. **Flexibilité Produits** : Système de variations sophistiqué

3. **Intégrité des Données** : Contraintes robustes et cascades bien pensées
4. **Sécurité Intégrée** : Authentification multiple et validation stricte
5. **Évolutivité** : Design permettant le scaling futur

6.2 Améliorations Potentielles:

Court Terme:

1. Ajout d'index sur les champs fréquemment recherchés
2. Implémentation de triggers pour le calcul automatique des prix moyens
3. Ajout de contraintes CHECK supplémentaires pour la validation

Moyen Terme:

1. Implémentation de full-text search pour les produits
2. Ajout de support multi-langues
3. Intégration de système de coupons et promotions

Long Terme:

1. Migration vers une architecture microservices
2. Implémentation de cache Redis pour les données fréquentes
3. Intégration avec un data warehouse pour l'analytique

6.3 Bonnes Pratiques Recommandées:

1. **Tests Réguliers** :
    - Tests de charge avec données réalistes
    - Tests de récupération après crash
    - Validation des contraintes d'intégrité
2. **Documentation** :
    - Mettre à jour le schéma après chaque changement
    - Documenter les workflows métier complexes
    - Maintenir un dictionnaire de données à jour
3. **Formation** :
    - Former l'équipe sur les requêtes optimisées
    - Établir des procédures pour les opérations courantes
    - Créer des guides de dépannage

6.4 Checklist de Déploiement:

- Backup de la base existante
- Validation de toutes les contraintes
- Tests de performance avec données réelles
- Configuration des utilisateurs et permissions
- Mise en place du monitoring
- Documentation des procédures d'urgence
- Plan de rollback en cas de problème