KoraMarket - Description des Tables par Microservice

# Service Authentification & Utilisateurs

## users

Colonnes : id (PK), nom, prénom, email (unique), mot\_de\_passe, téléphone, date\_inscription, statut, last\_login

Contraintes :

- PK : clé primaire (id).

- Index unique sur les colonnes spécifiées.

## roles

Colonnes : id (PK), nom (unique), description

Contraintes :

- PK : clé primaire (id).

- Index unique sur les colonnes spécifiées.

## permissions

Colonnes : id (PK), nom (unique), description

Contraintes :

- PK : clé primaire (id).

- Index unique sur les colonnes spécifiées.

## role\_permissions

Colonnes : id (PK), role\_id (FK->roles.id), permission\_id (FK->permissions.id)

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

## user\_roles

Colonnes : id (PK), user\_id (FK->users.id), role\_id (FK->roles.id)

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

## sessions

Colonnes : id (PK), user\_id (FK->users.id), token (unique), ip\_adresse, date\_creation, date\_expiration, navigateur

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

- Index unique sur les colonnes spécifiées.

## oauth\_clients

Colonnes : id (PK), client\_id (unique), client\_secret, scopes, redirect\_uri

Contraintes :

- PK : clé primaire (id).

- Index unique sur les colonnes spécifiées.

## audit\_logs

Colonnes : id (PK), user\_id (FK->users.id), action, endpoint, date, ip\_adresse, details

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

**1. categories**

| **Colonne** | **Type** | **Contraintes** | **Commentaire** |
| --- | --- | --- | --- |
| id | BIGSERIAL | PK, NOT NULL | Identifiant unique |
| nom | VARCHAR(255) | NOT NULL, UNIQUE | Nom de la catégorie |
| description | TEXT |  | Description |
| parent\_id | BIGINT | FK → categories.id (nullable) | Pour sous-catégories |
| image\_url | VARCHAR(1024) | NULL | (Optionnel) Image de catégorie |
| ordre\_affichage | INTEGER | NULL | (Optionnel) Tri visuel |

* **PK** : id
* **FK** : parent\_id → categories(id)

**2. produits**

| **Colonne** | **Type** | **Contraintes** | **Commentaire** |
| --- | --- | --- | --- |
| id | BIGSERIAL | PK, NOT NULL | Identifiant unique |
| nom | VARCHAR(255) | NOT NULL | Nom du produit |
| description | TEXT |  | Description |
| prix | NUMERIC(15,2) | NOT NULL, CHECK (prix >= 0) | Prix du produit |
| stock | INTEGER | NOT NULL DEFAULT 0, CHECK (stock >= 0) | Stock disponible |
| statut | VARCHAR(32) | NOT NULL DEFAULT 'DISPONIBLE' | Statut du produit |
| category\_id | BIGINT | FK → categories.id, NOT NULL | Catégorie du produit |
| date\_creation | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Date de création |
| marque | VARCHAR(128) | NULL | (Optionnel) Marque |
| reference | VARCHAR(128) | NULL, UNIQUE | (Optionnel) Référence interne |
| slug | VARCHAR(255) | NULL, UNIQUE | (Optionnel) SEO URL |
| poids | NUMERIC(10,3) | NULL | (Optionnel) Poids (kg) |
| is\_active | BOOLEAN | NOT NULL DEFAULT true | Produit actif/inactif |

* **PK** : id
* **FK** : category\_id → categories(id)

**3. produit\_images**

| **Colonne** | **Type** | **Contraintes** | **Commentaire** |
| --- | --- | --- | --- |
| id | BIGSERIAL | PK, NOT NULL | Identifiant unique |
| produit\_id | BIGINT | FK → produits.id, NOT NULL | Produit lié |
| url\_image | VARCHAR(1024) | NOT NULL | URL de l’image |
| is\_default | BOOLEAN | DEFAULT FALSE | Image principale ? |

* **PK** : id
* **FK** : produit\_id → produits(id)

**4. attributs\_produits**

| **Colonne** | **Type** | **Contraintes** | **Commentaire** |
| --- | --- | --- | --- |
| id | BIGSERIAL | PK, NOT NULL | Identifiant unique |
| produit\_id | BIGINT | FK → produits.id, NOT NULL | Produit lié |
| attribut | VARCHAR(255) | NOT NULL | (ex : couleur, taille) |
| valeur | VARCHAR(255) | NOT NULL | (ex : rouge, XL) |

* **PK** : id
* **FK** : produit\_id → produits(id)

**5. avis\_clients**

| **Colonne** | **Type** | **Contraintes** | **Commentaire** |
| --- | --- | --- | --- |
| id | BIGSERIAL | PK, NOT NULL | Identifiant unique |
| produit\_id | BIGINT | FK → produits.id, NOT NULL | Produit évalué |
| user\_id | BIGINT | NOT NULL | Utilisateur (via auth-service) |
| note | INTEGER | NOT NULL, CHECK (note >= 0 AND note <= 5) | Note sur 5 |
| commentaire | TEXT |  | Commentaire |
| date | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Date de l’avis |
| is\_approved | BOOLEAN | DEFAULT FALSE | (Optionnel) Modération |

* **PK** : id
* **FK** : produit\_id → produits(id)
* **Note** : user\_id vient du auth-service, donc pas de contrainte FK stricte, mais logique.

# Service Commandes & Paiements

## commandes

Colonnes : id (PK), user\_id (FK->users.id), montant\_total, statut, date\_creation, date\_mise\_a\_jour

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

## commande\_details

Colonnes : id (PK), commande\_id (FK->commandes.id), produit\_id (FK->produits.id), quantite, prix\_unitaire

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

## paiements

Colonnes : id (PK), commande\_id (FK->commandes.id), montant, moyen\_paiement, statut, transaction\_id (unique), date\_paiement

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

- Index unique sur les colonnes spécifiées.

## factures

Colonnes : id (PK), commande\_id (FK->commandes.id), url\_facture, date\_emission

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

# Recommandations clés

1. **Pas de FK vers d’autres services**  
   En microservices, n’utilise pas de FK SQL vers users / products des autres BDD. Stocke plutôt des **références externes** (UUID/ID) + un **snapshot** utile (nom produit, prix au moment de la commande, vendeur, etc.).
2. **Monnaie**  
   Utilise soit INTEGER en **centimes** (reco) soit DECIMAL(12,2) + **currency** (ISO 4217) sur toutes les lignes (order, paiement, refund).
3. **Statuts en ENUM + machine d’état**
   * order\_status: PENDING, AWAITING\_PAYMENT, PAID, PARTIALLY\_SHIPPED, SHIPPED, COMPLETED, CANCELED
   * payment\_status: INITIATED, AUTHORIZED, CAPTURED, PARTIALLY\_REFUNDED, REFUNDED, FAILED, CANCELED
   * shipment\_status: READY, IN\_TRANSIT, DELIVERED, RETURNED
   * invoice\_status: ISSUED, CANCELED
4. **Split marketplace**  
   Si plusieurs vendeurs par commande, ajoute vendor\_id/vendor\_email sur **order\_items**. Tu pourras créer des **shipments** par vendeur.
5. **Adresses “snapshot”**  
   Ajoute une table order\_addresses (billing/shipping) pour figer l’adresse au moment de la commande.
6. **Taxes & remises**  
   Garde des montants au **niveau ligne** et **au niveau commande**: subtotal, tax\_total, shipping\_total, discount\_total, grand\_total. Mets tax\_rate / tax\_amount sur les items.
7. **Idempotence paiements**  
   Dans payments, ajoute idempotency\_key (unique) pour éviter double capture en cas de retry. Garde provider, external\_transaction\_id (unique), amount\_authorized, amount\_captured.
8. **Remboursements**  
   Table refunds séparée liée à payments (partiels/ complets). Facilite la compta et la traçabilité.
9. **Livraisons**  
   Tables shipments + shipment\_items (quantités partielles, tracking, carrier, dates).
10. **Événements / Outbox**  
    Table order\_events (audit) et éventuellement outbox\_messages pour publier vers d’autres services de façon fiable.

# Schéma proposé (PostgreSQL — squelette)

Ajuste types/longueurs selon tes conventions (UUID vs BIGINT, centimes vs DECIMAL).

sql

CopierModifier

-- ========= ORDERS =========

CREATE TABLE orders (

id UUID PRIMARY KEY,

order\_number VARCHAR(32) UNIQUE NOT NULL, -- humain-friendly

user\_id\_ext UUID NOT NULL, -- id utilisateur (auth-service)

currency CHAR(3) NOT NULL, -- "XOF","EUR","USD"

subtotal\_amount BIGINT NOT NULL DEFAULT 0, -- centimes

tax\_total\_amount BIGINT NOT NULL DEFAULT 0,

shipping\_total\_amount BIGINT NOT NULL DEFAULT 0,

discount\_total\_amount BIGINT NOT NULL DEFAULT 0,

grand\_total\_amount BIGINT NOT NULL, -- = subtotal + tax + shipping - discount

order\_status VARCHAR(32) NOT NULL, -- enum logique côté code

payment\_status VARCHAR(32) NOT NULL,

created\_at TIMESTAMP NOT NULL DEFAULT now(),

updated\_at TIMESTAMP NOT NULL DEFAULT now(),

version INT NOT NULL DEFAULT 0 -- optimistic locking

);

CREATE INDEX idx\_orders\_user ON orders(user\_id\_ext);

CREATE INDEX idx\_orders\_status ON orders(order\_status);

CREATE INDEX idx\_orders\_created ON orders(created\_at);

-- ========= ORDER ITEMS =========

CREATE TABLE order\_items (

id UUID PRIMARY KEY,

order\_id UUID NOT NULL REFERENCES orders(id) ON DELETE CASCADE,

product\_id\_ext UUID NOT NULL, -- id du produit dans product-service

product\_name\_snap TEXT NOT NULL, -- snapshot nom

product\_sku\_snap TEXT,

product\_image\_snap TEXT,

vendor\_id\_ext UUID, -- vendeur (si marketplace)

vendor\_email\_snap TEXT,

unit\_price\_amount BIGINT NOT NULL, -- centimes au moment de la commande

quantity INT NOT NULL CHECK (quantity > 0),

tax\_rate NUMERIC(5,2), -- ex: 18.00

tax\_amount BIGINT NOT NULL DEFAULT 0,

line\_total\_amount BIGINT NOT NULL -- = unit\_price\*qty + tax - remise ligne (si tu en as)

);

CREATE INDEX idx\_items\_order ON order\_items(order\_id);

CREATE INDEX idx\_items\_vendor ON order\_items(vendor\_id\_ext);

-- ========= ORDER ADDRESSES =========

CREATE TABLE order\_addresses (

id UUID PRIMARY KEY,

order\_id UUID NOT NULL REFERENCES orders(id) ON DELETE CASCADE,

addr\_type VARCHAR(16) NOT NULL, -- 'BILLING' | 'SHIPPING'

full\_name TEXT NOT NULL,

phone TEXT,

line1 TEXT NOT NULL,

line2 TEXT,

city TEXT NOT NULL,

state TEXT,

postal\_code TEXT,

country CHAR(2) NOT NULL

);

CREATE INDEX idx\_addr\_order ON order\_addresses(order\_id);

-- ========= PAYMENTS =========

CREATE TABLE payments (

id UUID PRIMARY KEY,

order\_id UUID NOT NULL REFERENCES orders(id) ON DELETE CASCADE,

provider VARCHAR(32) NOT NULL, -- ex: "stripe","paypal","momo"

status VARCHAR(32) NOT NULL, -- AUTHORIZED/CAPTURED/FAILED/...

amount\_authorized BIGINT NOT NULL DEFAULT 0,

amount\_captured BIGINT NOT NULL DEFAULT 0,

currency CHAR(3) NOT NULL,

external\_transaction\_id TEXT UNIQUE, -- id renvoyé par le PSP

idempotency\_key TEXT UNIQUE, -- pour nos retries

failure\_reason TEXT,

created\_at TIMESTAMP NOT NULL DEFAULT now(),

authorized\_at TIMESTAMP,

captured\_at TIMESTAMP

);

CREATE INDEX idx\_payments\_order ON payments(order\_id);

CREATE INDEX idx\_payments\_status ON payments(status);

-- ========= REFUNDS =========

CREATE TABLE refunds (

id UUID PRIMARY KEY,

payment\_id UUID NOT NULL REFERENCES payments(id) ON DELETE CASCADE,

order\_id UUID NOT NULL REFERENCES orders(id) ON DELETE CASCADE,

amount BIGINT NOT NULL,

currency CHAR(3) NOT NULL,

status VARCHAR(32) NOT NULL, -- INITIATED/COMPLETED/FAILED

external\_refund\_id TEXT UNIQUE,

reason TEXT,

created\_at TIMESTAMP NOT NULL DEFAULT now(),

completed\_at TIMESTAMP

);

CREATE INDEX idx\_refunds\_payment ON refunds(payment\_id);

-- ========= INVOICES =========

CREATE TABLE invoices (

id UUID PRIMARY KEY,

order\_id UUID NOT NULL REFERENCES orders(id) ON DELETE CASCADE,

invoice\_number VARCHAR(32) UNIQUE NOT NULL,

url\_pdf TEXT NOT NULL,

status VARCHAR(16) NOT NULL, -- ISSUED/CANCELED

issued\_at TIMESTAMP NOT NULL DEFAULT now()

);

-- ========= SHIPMENTS =========

CREATE TABLE shipments (

id UUID PRIMARY KEY,

order\_id UUID NOT NULL REFERENCES orders(id) ON DELETE CASCADE,

vendor\_id\_ext UUID, -- si tu split par vendeur

carrier TEXT,

tracking\_number TEXT,

status VARCHAR(32) NOT NULL, -- READY/IN\_TRANSIT/DELIVERED/RETURNED

shipped\_at TIMESTAMP,

delivered\_at TIMESTAMP

);

CREATE INDEX idx\_shipments\_order ON shipments(order\_id);

CREATE TABLE shipment\_items (

id UUID PRIMARY KEY,

shipment\_id UUID NOT NULL REFERENCES shipments(id) ON DELETE CASCADE,

order\_item\_id UUID NOT NULL REFERENCES order\_items(id) ON DELETE RESTRICT,

quantity INT NOT NULL CHECK (quantity > 0)

);

-- ========= EVENTS / OUTBOX (optionnels mais utiles) =========

CREATE TABLE order\_events (

id UUID PRIMARY KEY,

order\_id UUID NOT NULL REFERENCES orders(id) ON DELETE CASCADE,

event\_type VARCHAR(64) NOT NULL,

payload JSONB,

created\_at TIMESTAMP NOT NULL DEFAULT now(),

created\_by TEXT

);

CREATE INDEX idx\_order\_events\_order ON order\_events(order\_id);

CREATE TABLE outbox\_messages (

id UUID PRIMARY KEY,

aggregate\_id UUID NOT NULL, -- typiquement order\_id

topic VARCHAR(64) NOT NULL,

payload JSONB NOT NULL,

status VARCHAR(16) NOT NULL DEFAULT 'PENDING', -- PENDING/SENT/FAILED

created\_at TIMESTAMP NOT NULL DEFAULT now(),

sent\_at TIMESTAMP

);

CREATE INDEX idx\_outbox\_topic ON outbox\_messages(topic);

# Petites différences vs ta proposition initiale

* **commande\_details** → renommé **order\_items**, avec **snapshot produit** + **vendor** pour marketplace.
* **paiements** → enrichi avec **idempotency\_key**, provider, amount\_authorized/captured, failure\_reason, timestamps.
* Ajout **refunds**, **shipments**, **shipment\_items**, **order\_addresses**, **order\_events**, **outbox\_messages**.
* **factures** → **invoices** avec invoice\_number unique + url\_pdf + status.
* **Pas de FK cross-service** : seulement des IDs externes + snapshots.

# Flux standard (pour te guider)

1. **Create Order (PENDING)** → calcule subtotal/tax/shipping/grand\_total à partir des **snapshots** (prix figés).
2. **Payment Intent** → ligne payments (INITIATED/AUTHORIZED/CAPTURED) avec idempotency\_key.
3. **Order status** passe à PAID.
4. **Shipments** (un ou plusieurs, par vendeur si besoin).
5. **Delivery** → DELIVERED, puis COMPLETED.
6. **Invoice** → ISSUED (lien PDF).
7. **Refund** (si besoin) → table refunds, mise à jour du payment\_status/order\_status.

# Service Panier & Recommandations IA

## paniers

Colonnes : id (PK), user\_id (FK->users.id), date\_creation, last\_update

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

## panier\_items

Colonnes : id (PK), panier\_id (FK->paniers.id), produit\_id (FK->produits.id), quantite

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

## recommandations

Colonnes : id (PK), user\_id (FK->users.id), produit\_id (FK->produits.id), score, date\_generation

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

# Service Livraisons & Logistique

## livraisons

Colonnes : id (PK), commande\_id (FK->commandes.id), adresse\_id (FK->adresses\_livraison.id), transporteur\_id (FK->transporteurs.id), statut, tracking\_number (unique), date\_prevue

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

- Index unique sur les colonnes spécifiées.

## transporteurs

Colonnes : id (PK), nom, contact, delai\_estime

Contraintes :

- PK : clé primaire (id).

## adresses\_livraison

Colonnes : id (PK), user\_id (FK->users.id), rue, ville, pays, code\_postal

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

# Service Notifications & Gamification

## notifications

Colonnes : id (PK), user\_id (FK->users.id), type, message, statut, date

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

## badges

Colonnes : id (PK), nom, description, image

Contraintes :

- PK : clé primaire (id).

## user\_badges

Colonnes : id (PK), user\_id (FK->users.id), badge\_id (FK->badges.id), date\_obtention

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

## points\_fidelite

Colonnes : id (PK), user\_id (FK->users.id), points, date\_mise\_a\_jour

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

# Service Analytics & Rapports

## logs\_api

Colonnes : id (PK), microservice, endpoint, status\_code, temps\_reponse, date

Contraintes :

- PK : clé primaire (id).

## statistiques\_ventes

Colonnes : id (PK), produit\_id (FK->produits.id), total\_ventes, date

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

## rapports

Colonnes : id (PK), type, contenu, date\_generation

Contraintes :

- PK : clé primaire (id).

# Service Portefeuille (Wallet)

## wallets

Colonnes : id (PK), user\_id (FK->users.id), solde, date\_mise\_a\_jour

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

## transactions

Colonnes : id (PK), wallet\_id (FK->wallets.id), type, montant, statut, reference (unique), date

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

- Index unique sur les colonnes spécifiées.

# Service Localisation & Paramètres

## pays

Colonnes : id (PK), code\_iso (unique), nom

Contraintes :

- PK : clé primaire (id).

- Index unique sur les colonnes spécifiées.

## villes

Colonnes : id (PK), pays\_id (FK->pays.id), nom

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

## parametres\_systeme

Colonnes : id (PK), cle (unique), valeur, description

Contraintes :

- PK : clé primaire (id).

- Index unique sur les colonnes spécifiées.

# Loyalty-Service

## programmes\_fidelite

Colonnes : id (PK), nom, description, points\_par\_euro

Contraintes :

- PK : clé primaire (id).

## recompenses

Colonnes : id (PK), programme\_id (FK->programmes\_fidelite.id), type, valeur, points\_necessaires

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

## historique\_points

Colonnes : id (PK), user\_id (FK->users.id), points, motif, date

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

# Fraud-Detection-Service

## transactions\_suspectes

Colonnes : id (PK), transaction\_id (FK->transactions.id), raison, score\_risque, date\_detection

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

## modeles\_ml

Colonnes : id (PK), nom, version, date\_deploiement

Contraintes :

- PK : clé primaire (id).

## historique\_verifications

Colonnes : id (PK), user\_id (FK->users.id), resultat, date

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.

# Localization-Service

## traductions

Colonnes : id (PK), code\_langue, cle, valeur

Contraintes :

- PK : clé primaire (id).

## langues\_disponibles

Colonnes : id (PK), code (unique), nom

Contraintes :

- PK : clé primaire (id).

- Index unique sur les colonnes spécifiées.

## format\_localisation

Colonnes : id (PK), langue\_id (FK->langues\_disponibles.id), format\_date, devise

Contraintes :

- PK : clé primaire (id).

- FK : relations vers d'autres tables.