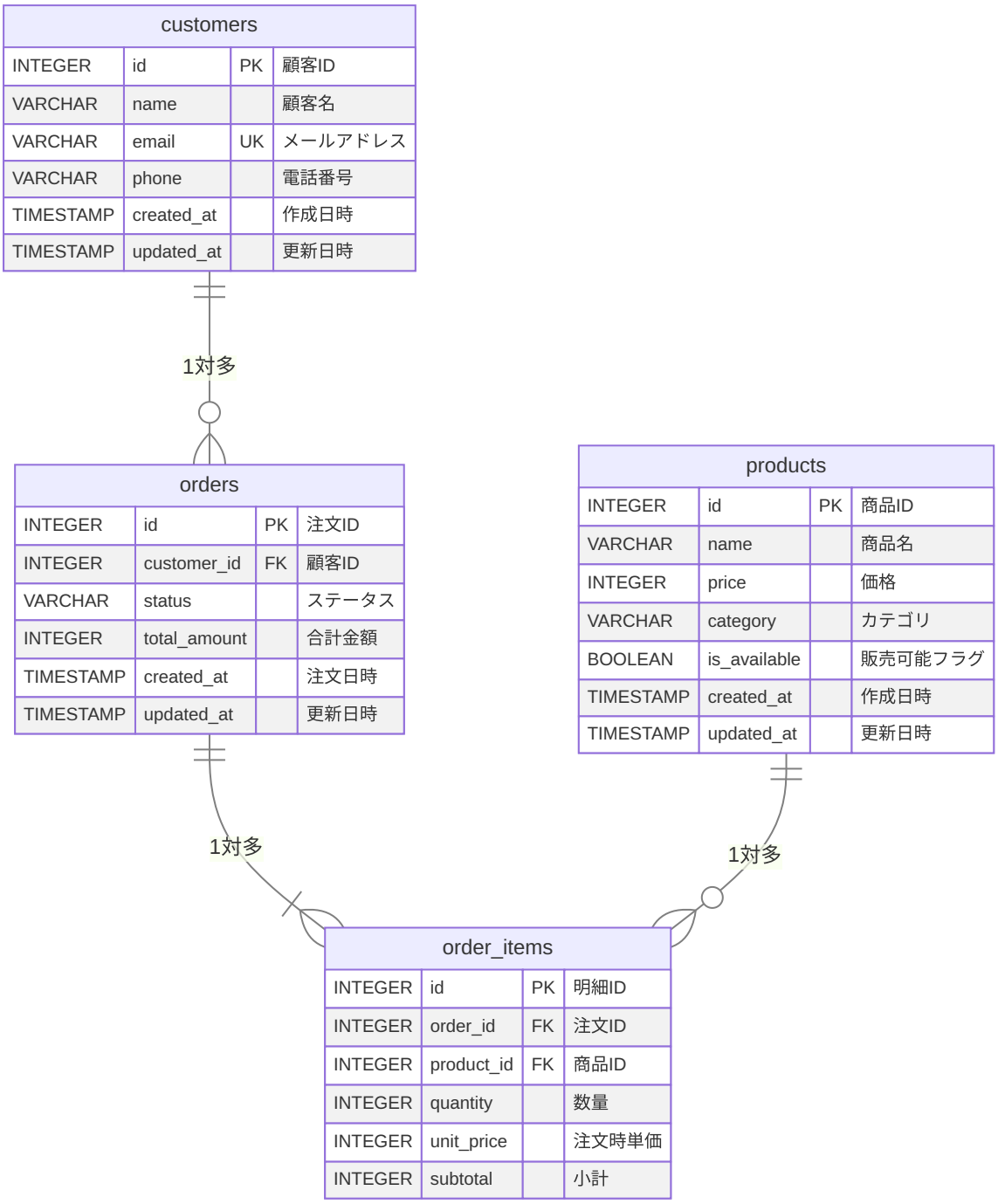
データベース設計

テーブル一覧

テーブル名	説明
customers	顧客情報
products	商品情報
orders	注文ヘッダー
order_items	注文明細

ER図


customers
INTEGER id PK 顧客ID
VARCHAR name 顧客名
VARCHAR email UK メールアドレス
VARCHAR phone 電話番号
TIMESTAMP created_at 作成日時
TIMESTAMP updated_at 更新日時
1対多
orders
INTEGER id PK 注文ID
INTEGER customer_id FK 顧客ID
VARCHAR status ステータス
INTEGER total_amount 合計金額
TIMESTAMP created_at 注文日時
TIMESTAMP updated_at 更新日時
products
INTEGER id PK 商品ID
VARCHAR name 商品名
INTEGER price 価格
VARCHAR category カテゴリ
BOOLEAN is_available 販売可能フラグ
TIMESTAMP created_at 作成日時
TIMESTAMP updated_at 更新日時
1対多
1対多
order_items
INTEGER id PK 明細ID
INTEGER order_id FK 注文ID
INTEGER product_id FK 商品ID
INTEGER quantity 数量
INTEGER unit_price 注文時単価
INTEGER subtotal 小計

リレーション説明

関係	説明
customers → orders	1対多: 1人の顧客は複数の注文を持てる
orders → order_items	1対多: 1つの注文は複数の明細を持つ
products → order_items	1対多: 1つの商品は複数の注文明細に含まれる

テーブル設計

customersテーブル（顧客）

項目名	データ型	制約	説明
id	整数	主キー、自動採番	顧客ID
name	文字列(100)	必須	顧客名
email	文字列(255)	一意、NULL許可	メールアドレス
phone	文字列(20)	NULL許可	電話番号
created_at	日時	自動設定	作成日時
updated_at	日時	自動更新	更新日時

productsテーブル（商品）

項目名	データ型	制約	説明
id	整数	主キー、自動採番	商品ID
name	文字列(100)	必須	商品名
price	整数	必須、1以上	価格（税込）
category	文字列(20)	必須	カテゴリ（food/drink/other）
is_available	真偽値	初期値: true	販売可能フラグ
created_at	日時	自動設定	作成日時
updated_at	日時	自動更新	更新日時

ordersテーブル（注文）

項目名	データ型	制約	説明
id	整数	主キー、自動採番	注文ID
customer_id	整数	外部キー、必須	顧客ID（customersへの参照）
status	文字列(20)	必須、初期値: pending	ステータス
total_amount	整数	必須	合計金額
created_at	日時	自動設定	注文日時
updated_at	日時	自動更新	更新日時

ステータス値: - `pending` - 受付 - `cooking` - 調理中 - `completed` - 完了 - `cancelled` - キャンセル

order_itemsテーブル（注文明細）

項目名	データ型	制約	説明
id	整数	主キー、自動採番	明細ID
order_id	整数	外部キー、必須	注文ID（ordersへの参照）
product_id	整数	外部キー、必須	商品ID（productsへの参照）
quantity	整数	必須、1以上	数量
unit_price	整数	必須	注文時の単価
subtotal	整数	必須	小計（unit_price × quantity）

設計ポイント: - `unit_price` を保持する理由: 商品価格が変更されても、注文時の価格を記録するため - `subtotal` を保持する理由: 計算済みの値を保持することで集計時のパフォーマンス向上

テーブル作成SQL

PostgreSQL

```sql
-- 顧客テーブル
CREATE TABLE customers (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(255) UNIQUE,
    phone VARCHAR(20),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- 商品テーブル
CREATE TABLE products (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    price INTEGER NOT NULL CHECK (price >= 1),
    category VARCHAR(20) NOT NULL CHECK (category IN ('food', 'drink', 'other')),
    is_available BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- 注文テーブル
CREATE TABLE orders (
    id SERIAL PRIMARY KEY,
    customer_id INTEGER NOT NULL REFERENCES customers(id),
    status VARCHAR(20) NOT NULL DEFAULT 'pending'
        CHECK (status IN ('pending', 'cooking', 'completed', 'cancelled')),
    total_amount INTEGER NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- 注文明細テーブル
CREATE TABLE order_items (
    id SERIAL PRIMARY KEY,
    order_id INTEGER NOT NULL REFERENCES orders(id) ON DELETE CASCADE,
    product_id INTEGER NOT NULL REFERENCES products(id),
    quantity INTEGER NOT NULL CHECK (quantity >= 1),
    unit_price INTEGER NOT NULL,
    subtotal INTEGER NOT NULL
);
```

MySQL / MariaDB

```sql
-- 顧客テーブル
CREATE TABLE customers (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(255) UNIQUE,
    phone VARCHAR(20),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);

-- 商品テーブル
CREATE TABLE products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    price INT NOT NULL CHECK (price >= 1),
    category ENUM('food', 'drink', 'other') NOT NULL,
    is_available BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);

-- 注文テーブル
CREATE TABLE orders (
    id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT NOT NULL,
    status ENUM('pending', 'cooking', 'completed', 'cancelled') NOT NULL DEFAULT 'pending',
    total_amount INT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (customer_id) REFERENCES customers(id)
);

-- 注文明細テーブル
CREATE TABLE order_items (
    id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL CHECK (quantity >= 1),
    unit_price INT NOT NULL,
    subtotal INT NOT NULL,
    FOREIGN KEY (order_id) REFERENCES orders(id) ON DELETE CASCADE,
    FOREIGN KEY (product_id) REFERENCES products(id)
);
```

インデックス

```
-- 顧客検索用
CREATE INDEX idx_customers_name ON customers(name);
CREATE INDEX idx_customers_email ON customers(email);

-- 商品検索用
CREATE INDEX idx_products_category ON products(category);
CREATE INDEX idx_products_is_available ON products(is_available);

-- 注文検索用
CREATE INDEX idx_orders_customer_id ON orders(customer_id);
CREATE INDEX idx_orders_status ON orders(status);
CREATE INDEX idx_orders_created_at ON orders(created_at DESC);

-- 注文明細検索用
CREATE INDEX idx_order_items_order_id ON order_items(order_id);
CREATE INDEX idx_order_items_product_id ON order_items(product_id);
```

サンプルデータ

```sql
-- 顧客データ
INSERT INTO customers (name, email, phone) VALUES
('山田太郎', 'yamada@example.com', '090-1234-5678'),
('佐藤花子', 'sato@example.com', '080-2345-6789'),
('田中一郎', 'tanaka@example.com', '070-3456-7890'),
('鈴木美咲', NULL, '090-4567-8901'),
('高橋健太', 'takahashi@example.com', NULL);

-- 商品データ
INSERT INTO products (name, price, category, is_available) VALUES
('カレーライス', 800, 'food', TRUE),
('ハンバーグ定食', 950, 'food', TRUE),
('サラダセット', 400, 'food', TRUE),
('オムライス', 850, 'food', TRUE),
('唐揚げ定食', 900, 'food', TRUE),
('コーヒー', 300, 'drink', TRUE),
('オレンジジュース', 250, 'drink', TRUE),
('ビール', 500, 'drink', TRUE),
('烏龍茶', 200, 'drink', TRUE),
('デザートセット', 450, 'other', FALSE);

-- 注文データ
INSERT INTO orders (customer_id, status, total_amount) VALUES
(1, 'completed', 1200),
(2, 'completed', 1550),
(3, 'cooking', 2100),
(1, 'pending', 1700),
(4, 'completed', 900);

-- 注文明細データ
INSERT INTO order_items (order_id, product_id, quantity, unit_price, subtotal) VALUES
-- 注文1: 山田太郎 - カレーライス + サラダセット
(1, 1, 1, 800, 800),
(1, 3, 1, 400, 400),
-- 注文2: 佐藤花子 - ハンバーグ定食 + オレンジジュース x2
(2, 2, 1, 950, 950),
(2, 7, 2, 250, 500),
-- 注文3: 田中一郎 - オムライス x2 + コーヒー
(3, 4, 2, 850, 1700),
(3, 6, 1, 300, 300),
-- 注文4: 山田太郎 - 唐揚げ定食 + ビール
(4, 5, 1, 900, 900),
(4, 8, 1, 500, 500),
-- 注文5: 鈴木美咲 - 唐揚げ定食
(5, 5, 1, 900, 900);
```

基本的なSQL操作

顧客操作

```sql
-- 顧客一覧取得
SELECT id, name, email, phone, created_at
FROM customers
ORDER BY created_at DESC;

-- 顧客検索（名前で部分一致）
SELECT id, name, email, phone
FROM customers
WHERE name LIKE '%山田%';

-- 顧客登録
INSERT INTO customers (name, email, phone)
VALUES ('新規顧客', 'new@example.com', '090-0000-0000');

-- 顧客更新
UPDATE customers
SET name = '更新後の名前', email = 'updated@example.com', updated_at = CURRENT_TIMESTAMP
WHERE id = 1;

-- 顧客削除（注文がない場合のみ）
DELETE FROM customers WHERE id = 1;
```

商品操作

```sql
-- 商品一覧取得（販売中のみ）
SELECT id, name, price, category, is_available
FROM products
WHERE is_available = TRUE
ORDER BY category, name;

-- カテゴリ別商品一覧
SELECT id, name, price, category
FROM products
WHERE category = 'food' AND is_available = TRUE;

-- 商品登録
INSERT INTO products (name, price, category)
VALUES ('新メニュー', 750, 'food');

-- 販売停止に変更
UPDATE products
SET is_available = FALSE, updated_at = CURRENT_TIMESTAMP
WHERE id = 10;
```

注文操作

```sql
-- 注文一覧取得（顧客名含む）
SELECT
    o.id,
    c.name AS customer_name,
    o.status,
    o.total_amount,
    o.created_at
FROM orders o
JOIN customers c ON o.customer_id = c.id
ORDER BY o.created_at DESC;

-- ステータスで絞り込み
SELECT o.id, c.name, o.status, o.total_amount, o.created_at
FROM orders o
JOIN customers c ON o.customer_id = c.id
WHERE o.status = 'pending'
ORDER BY o.created_at ASC;

-- 注文詳細取得（明細含む）
SELECT
    o.id AS order_id,
    c.name AS customer_name,
    o.status,
    o.total_amount,
    o.created_at,
    oi.id AS item_id,
    p.name AS product_name,
    oi.quantity,
    oi.unit_price,
    oi.subtotal
FROM orders o
JOIN customers c ON o.customer_id = c.id
JOIN order_items oi ON o.id = oi.order_id
JOIN products p ON oi.product_id = p.id
WHERE o.id = 1;

-- ステータス更新
UPDATE orders
SET status = 'cooking', updated_at = CURRENT_TIMESTAMP
WHERE id = 4;

-- 注文キャンセル
UPDATE orders
SET status = 'cancelled', updated_at = CURRENT_TIMESTAMP
WHERE id = 4;
```

集計クエリ

```sql
-- 本日の注文件数と売上
SELECT
    COUNT(*) AS order_count,
    COALESCE(SUM(total_amount), 0) AS total_sales
FROM orders
WHERE DATE(created_at) = CURRENT_DATE
AND status != 'cancelled';

-- ステータス別件数
SELECT
    status,
    COUNT(*) AS count
FROM orders
WHERE DATE(created_at) = CURRENT_DATE
GROUP BY status;

-- 期間指定の売上集計
SELECT
    DATE(created_at) AS order_date,
    COUNT(*) AS order_count,
    SUM(total_amount) AS daily_sales
FROM orders
WHERE created_at >= '2025-01-01' AND created_at < '2025-02-01'
AND status = 'completed'
GROUP BY DATE(created_at)
ORDER BY order_date;

-- 商品別売上ランキング
SELECT
    p.name AS product_name,
    SUM(oi.quantity) AS total_quantity,
    SUM(oi.subtotal) AS total_sales
FROM order_items oi
JOIN products p ON oi.product_id = p.id
JOIN orders o ON oi.order_id = o.id
WHERE o.status = 'completed'
GROUP BY p.id, p.name
ORDER BY total_sales DESC
LIMIT 10;

-- 顧客別注文回数
SELECT
    c.name AS customer_name,
    COUNT(o.id) AS order_count,
    SUM(o.total_amount) AS total_spent
FROM customers c
LEFT JOIN orders o ON c.id = o.customer_id AND o.status = 'completed'
GROUP BY c.id, c.name
ORDER BY total_spent DESC;
```

注文作成の処理フロー（トランザクション）

注文作成時は複数テーブルへの書き込みが必要なため、トランザクションを使用します。

```sql
BEGIN;

-- 1. 注文ヘッダーを作成
INSERT INTO orders (customer_id, status, total_amount)
VALUES (1, 'pending', 2000)
RETURNING id;
-- 例: id = 6 が返される

-- 2. 注文明細を作成
INSERT INTO order_items (order_id, product_id, quantity, unit_price, subtotal)
VALUES
(6, 1, 2, 800, 1600),  -- カレーライス x2
(6, 3, 1, 400, 400);   -- サラダセット x1

COMMIT;
```

削除時の制約

顧客削除

顧客に紐づく注文がある場合、削除前にチェックが必要です。

```sql
-- 注文履歴があるかチェック
SELECT COUNT(*) FROM orders WHERE customer_id = 1;

-- 履歴がない場合のみ削除可能
DELETE FROM customers WHERE id = 1;
```

商品削除

商品が注文明細に含まれている場合、削除ではなく販売停止にします。

```sql
-- 注文明細に含まれているかチェック
SELECT COUNT(*) FROM order_items WHERE product_id = 1;

-- 含まれている場合は販売停止
UPDATE products SET is_available = FALSE WHERE id = 1;

-- 含まれていない場合は削除可能
DELETE FROM products WHERE id = 1;
```

注文削除

注文を削除すると、`ON DELETE CASCADE`により明細も自動削除されます。 ただし、通常は削除ではなくキャンセルステータスに変更することを推奨します。

```
-- キャンセル処理（推奨）
UPDATE orders SET status = 'cancelled' WHERE id = 1;

-- 物理削除（明細も含めて削除される）
DELETE FROM orders WHERE id = 1;
```

備考

価格の保持について

`order_items`テーブルに`unit_price`を保持する理由： - 商品マスタの価格は変更される可能性がある - 注文時点での価格を記録することで、正確な売上記録を維持 - 履歴の整合性を保証

論理削除について

このシステムでは物理削除を基本としていますが、以下の場合は論理削除を検討： - 削除履歴を残したい場合 - 復元機能が必要な場合

論理削除を導入する場合は、各テーブルに`deleted_at`カラムを追加します。

タイムゾーンについ

タイムスタンプはサーバーのローカルタイム（日本時間）で統一。 本番環境ではUTCで保存し、表示時に変換することを推奨。