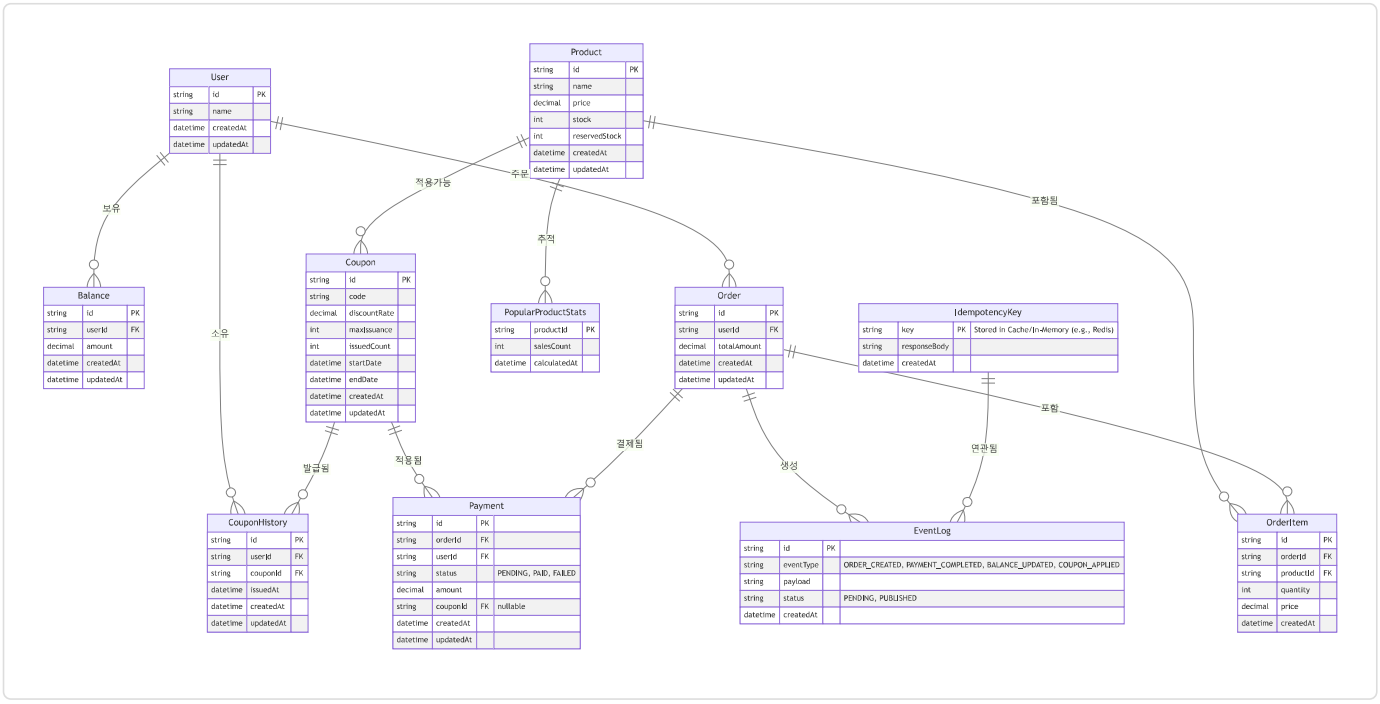
e-커머스 시스템 ERD

ERD 다이어그램



설명

1. 엔티티와 속성

요구사항과 설계서를 기반으로 간소화된 엔티티를 정의함.

- **User**: 사용자 정보.
  - `id` : (PK) 기본 키.
  - `name` : 사용자 이름.
  - `createdAt` , `updatedAt` : 생성/수정 시간.

  관계:
  - **User ↔ Balance**: 1:1, 한 사용자는 하나의 잔액을 보유함.
  - **User ↔ Order**: 1:N, 한 사용자가 여러 주문을 할 수 있음.
  - **User ↔ CouponHistory**: 1:N, 한 사용자가 여러 쿠폰 발급 기록을 가질 수 있음.

- **Balance**: 사용자별 잔액.
  - `id` : (PK) 기본 키.
  - `userId` : (FK) `User`와 1:1 관계.
  - `amount` : 잔액.
  - `createdAt`, `updatedAt` : 생성/수정 시간.

  관계:
  - **Balance ↔ User**: 1:1, 잔액은 한 명의 사용자에게 귀속됨.

- **Product**: 상품 정보.
  - `id` : (PK) 기본 키.
  - `name` : 상품 이름.
  - `price` : 가격.
  - `stock` : 실제 재고.
  - `reservedStock` : 주문 생성 시 임시 예약된 재고.
  - `createdAt`, `updatedAt` : 생성/수정 시간.

  관계:
  - **Product ↔ OrderItem**: 1:N, 하나의 상품이 여러 주문 항목에 포함될 수 있음.
  - **Product ↔ Coupon**: 1:N, 하나의 상품에 여러 쿠폰이 적용될 수 있음 (확장성 고려).
  - **Product ↔ PopularProductStats**: 1:1, 상품별 인기 통계 데이터와 연결됨.

- **Coupon**: 쿠폰 정보.
  - `id` : (PK) 기본 키.
  - `code` : 쿠폰 코드.
  - `discountRate` : 할인율.
  - `maxIssuance` : 최대 발급 수량.
  - `issuedCount` : 현재 발급된 수량.
  - `startDate`, `endDate` : 유효 기간.
  - `createdAt`, `updatedAt` : 생성/수정 시간.

  관계:
  - **Coupon ↔ CouponHistory**: 1:N, 하나의 쿠폰이 여러 사용자에게 발급될 수 있음.
  - **Coupon ↔ Payment**: 1:N, 하나의 쿠폰이 여러 결제에 사용될 수 있음.
  - **Coupon ↔ Product**: N:1, 여러 쿠폰이 하나의 특정 상품에만 적용될 수 있음 (또는 전역 쿠폰).

- **CouponHistory**: 사용자별 쿠폰 발급 기록.
  - `id` : (PK) 기본 키.
  - `userId` : (FK) `User` 참조.
  - `couponId` : (FK) `Coupon` 참조.
  - `issuedAt` : 발급 시점.
  - `createdAt` , `updatedAt` : 생성/수정 시간.

  관계:
  - **CouponHistory ↔ User**: N:1, 여러 발급 기록이 한 명의 사용자에게 속함.
  - **CouponHistory ↔ Coupon**: N:1, 여러 발급 기록이 하나의 쿠폰에 속함.

- **Order**: 주문 정보.
  - `id` : (PK) 기본 키.
  - `userId` : (FK) `User` 참조.
  - `totalAmount` : 주문 총액.
  - `createdAt` , `updatedAt` : 생성/수정 시간.

  관계:
  - **Order ↔ User**: N:1, 여러 주문이 한 명의 사용자에게 속함.
  - **Order ↔ OrderItem**: 1:N, 하나의 주문은 여러 주문 항목을 포함함.
  - **Order ↔ Payment**: 1:N, 하나의 주문에 대해 여러 번의 결제 시도(성공 또는 실패)가 있을 수 있음.
  - **Order ↔ EventLog**: 1:N, 주문 생성과 관련된 이벤트 로그가 생성될 수 있음.

- **OrderItem**: 주문 항목.
  - `id` : (PK) 기본 키.
  - `orderId` : (FK) `Order` 참조.
  - `productId` : (FK) `Product` 참조.
  - `quantity` : 상품 수량.
  - `price` : 주문 시점의 상품 가격.
  - `createdAt` : 생성 시간.

  관계:
  - **OrderItem ↔ Order**: N:1, 여러 주문 항목이 하나의 주문에 속함.
  - **OrderItem ↔ Product**: N:1, 여러 주문 항목이 하나의 상품을 참조할 수 있음.

- **Payment**: 결제 정보.
  - `id` : (PK) 기본 키.
  - `orderId` : (FK) `Order` 참조.
  - `userId` : (FK) `User` 참조.
  - `status` : 결제 상태(PENDING, PAID, FAILED).
  - `amount` : 결제 금액.
  - `couponId` : (FK, nullable) 적용된 `Coupon` 참조.
  - `createdAt` , `updatedAt` : 생성/수정 시간.

  관계:
  - **Payment** ↔ **Order**: N:1, 여러 결제 시도가 하나의 주문에 연결될 수 있음.
  - **Payment** ↔ **User**: N:1, 여러 결제가 한 명의 사용자에 의해 이루어질 수 있음.
  - **Payment** ↔ **Coupon**: N:1, 여러 결제에 하나의 쿠폰이 사용될 수 있음.

- **EventLog**: 이벤트 기록.
  - `id` : (PK) 기본 키.
  - `eventType` : 이벤트 종류(ORDER_CREATED, PAYMENT_COMPLETED 등).
  - `payload` : 이벤트 데이터.
  - `status` : 전송 상태(PENDING, PUBLISHED).
  - `createdAt` : 생성 시간.

  관계:
  - **EventLog** ↔ **Order**: N:1, 여러 이벤트가 하나의 주문과 연관될 수 있음.
  - **EventLog** ↔ **IdempotencyKey**: N:1, 멱등성 키와 관련된 여러 이벤트가 있을 수 있음.

- **PopularProductStats**: 인기 상품 통계.
  - `productId` : (PK) `Product` 참조.
  - `salesCount` : 판매량.
  - `calculatedAt` : 집계 시간.

  관계:
  - **PopularProductStats** ↔ **Product**: 1:1, 각 통계는 하나의 상품에 해당함.

- **IdempotencyKey**: 멱등성 키 저장.
  - `key` : (PK) 클라이언트 제공 UUID, Redis와 같은 캐시/인메모리 저장소에 저장됨.
  - `responseBody` : 처리 결과.
  - `createdAt` : 생성 시간.

  관계:
  - **IdempotencyKey ↔ EventLog**: 1:N, 하나의 멱등성 키가 여러 이벤트 로그를 생성할 수 있음.

3. 설계 고려사항

- **Order-Payment 분리**: `Order` 는 주문 생성, `Payment`는 결제 처리를 담당. 시퀀스 다이어그램 반영.

- 멱등성:
  - `IdempotencyKey` 는 `CachePort` 에 저장
  - `TTL(예: 24시간)`로 관리하여 `DB` 부하 감소.

- **EventLog**:
  - `DB` 저장 필수?
  - 예: `OrderCreatedEvent` , `PaymentCompletedEvent` 저장.

- 동시성:
  - `Balance.amount` , `Product.stock` , `reservedStock` , `Coupon.issuedCount`는 `LockingPort` 로 동시성 제어.

- 캐싱:
  - `Balance` , `Product` , `PopularProductStats` 는 `CachePort`로 캐싱.
  - `IdempotencyKey` 도 캐시에 저장.

- 테스트:
  - 동시성: `Balance` , `Product` , `Coupon` 업데이트 무결성.
  - 멱등성: 동일 `IdempotencyKey` 로 요청 시 동일 응답.
  - 캐싱: `CachePort`와 `StoragePort` 데이터 일관성.