📐 E-Commerce Database 논리 스키마 (Logical Schema)

목차

논리 스키마란?

논리 스키마(Logical Schema)는 데이터베이스의 논리적 구조를 형식적으로 표현한 것입니다.

- 개념적 설계를 관계형 모델로 변환
- 테이블 구조, 속성, 키, 제약조건을 명시
- DBMS 독립적인 설계 (특정 제품에 종속되지 않음)

관계형 스키마 표현

📋 표준 관계형 표기법

```
CUSTOMERS(customer_id, customer_name, customer_email, customer_phone)
    Primary Key: customer_id
    Unique: customer_email

PRODUCTS(product_id, product_name, product_category, product_price)
    Primary Key: product_id

ORDERS(order_id, order_date, customer_id, shipping_address, billing_address,
       payment_method, payment_txn_id, order_status)
    Primary Key: order_id
    Foreign Key: customer_id REFERENCES CUSTOMERS(customer_id)
    Unique: payment_txn_id

ORDER_ITEMS(order_item_id, order_id, product_id, quantity, discount)
    Primary Key: order_item_id
    Foreign Key: order_id REFERENCES ORDERS(order_id)
    Foreign Key: product_id REFERENCES PRODUCTS(product_id)
    Unique: (order_id, product_id)

REVIEWS(review_id, order_id, product_id, review_rating, review_comment)
```

Primary Key: review_id
Foreign Key: order_id REFERENCES ORDERS(order_id)
Foreign Key: product_id REFERENCES PRODUCTS(product_id)

🔤 간략 표기법 (밑줄 = Primary Key)

CUSTOMERS(<u>customer_id</u>, customer_name, customer_email, customer_phone)

PRODUCTS(<u>product_id</u>, product_name, product_category, product_price)

ORDERS(<u>order_id</u>, order_date, customer_id*, shipping_address, billing_address,
payment_method, payment_txn_id, order_status)

ORDER_ITEMS(<u>order_item_id</u>, order_id*, product_id*, quantity, discount)

REVIEWS(<u>review_id</u>, order_id*, product_id*, review_rating, review_comment)

* = Foreign Key

테이블 상세 정의

1️⃣ CUSTOMERS (고객)

목적: 고객의 기본 정보를 저장

속성명	데이터 타입	제약조건	설명
customer_id	INTEGER	PK, NOT NULL	고객 고유 식별자
customer_name	VARCHAR(100)	NOT NULL	고객 이름
customer_email	VARCHAR(255)	NOT NULL, UNIQUE	이메일 주소
customer_phone	VARCHAR(20)	NOT NULL	전화번호

키 정의:

- **Primary Key**: customer_id
- **Candidate Key**: customer_email (이메일은 고유해야 함)
- **Alternate Key**: customer_email

비즈니스 규칙:

- 한 명의 고객은 여러 개의 주문을 할 수 있다
- 이메일 주소는 중복될 수 없다
- 고객 정보는 주문 이력과 독립적으로 존재한다

2️⃣ PRODUCTS (제품)

목적: 판매 제품의 정보를 저장

속성명	데이터 타입	제약조건	설명
product_id	INTEGER	PK, NOT NULL	제품 고유 식별자
product_name	VARCHAR(200)	NOT NULL	제품명
product_category	VARCHAR(100)	NOT NULL	제품 카테고리
product_price	DECIMAL(10,2)	NOT NULL, CHECK(product_price >= 0)	제품 가격

키 정의:

- **Primary Key**: product_id
- **Candidate Key**: product_id (단일 키)

비즈니스 규칙:

- 하나의 제품은 여러 주문에 포함될 수 있다
- 제품 가격은 0 이상이어야 한다
- 제품 정보는 주문과 독립적으로 관리된다

3️⃣ ORDERS (주문)

목적: 고객의 주문 기본 정보를 저장

속성명	데이터 타입	제약조건	설명
order_id	INTEGER	PK, NOT NULL	주문 고유 식별자
order_date	DATE	NOT NULL	주문 날짜
customer_id	INTEGER	FK, NOT NULL	고객 ID (CUSTOMERS 참조)
shipping_address	VARCHAR(500)	NOT NULL	배송지 주소
billing_address	VARCHAR(500)	NOT NULL	청구지 주소
payment_method	VARCHAR(50)	NOT NULL, CHECK(payment_method IN ('Card', 'BankTransfer', 'KakaoPay', 'Payco'))	결제 수단
payment_txn_id	VARCHAR(100)	NOT NULL, UNIQUE	결제 거래 ID
order_status	VARCHAR(50)	NOT NULL, CHECK(order_status IN ('Processing', 'Shipped', 'Delivered', 'Cancelled'))	주문 상태

키 정의:

- **Primary Key**: order_id
- **Foreign Key**: customer_id → CUSTOMERS(customer_id)
- **Candidate Key**: payment_txn_id (결제 거래 ID는 고유)

비즈니스 규칙:

- 하나의 주문은 한 명의 고객에게 속한다 (N:1)
- 하나의 주문은 여러 제품을 포함할 수 있다
- 결제 거래 ID는 고유해야 한다
- 주문 상태는 정해진 값만 가능하다

참조 무결성:

```
customer_id REFERENCES CUSTOMERS(customer_id)
    ON DELETE RESTRICT
    ON UPDATE CASCADE
```

4️⃣ ORDER_ITEMS (주문 항목)

목적: 주문과 제품 간의 다대다(M:N) 관계를 해결하는 연결 테이블

속성명	데이터 타입	제약조건	설명
order_item_id	INTEGER	PK, NOT NULL	주문 항목 고유 식별자
order_id	INTEGER	FK, NOT NULL	주문 ID (ORDERS 참조)
product_id	INTEGER	FK, NOT NULL	제품 ID (PRODUCTS 참조)
quantity	INTEGER	NOT NULL, CHECK(quantity > 0)	주문 수량
discount	DECIMAL(10,2)	DEFAULT 0, CHECK(discount >= 0)	할인 금액

키 정의:

- **Primary Key**: order_item_id
- **Foreign Key**:
  - order_id → ORDERS(order_id)
  - product_id → PRODUCTS(product_id)
- **Alternate Key**: (order_id, product_id) - 한 주문에 같은 제품은 한 번만

비즈니스 규칙:

- 한 주문에 같은 제품이 중복으로 들어갈 수 없다
- 주문 수량은 1 이상이어야 한다

- 할인 금액은 0 이상이어야 한다

참조 무결성:

```
order_id REFERENCES ORDERS(order_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE

product_id REFERENCES PRODUCTS(product_id)
    ON DELETE RESTRICT
    ON UPDATE CASCADE
```

5️⃣ REVIEWS (리뷰)

목적: 고객의 제품 리뷰를 저장

속성명	데이터 타입	제약조건	설명
review_id	INTEGER	PK, NOT NULL	리뷰 고유 식별자
order_id	INTEGER	FK, NOT NULL	주문 ID (ORDERS 참조)
product_id	INTEGER	FK, NOT NULL	제품 ID (PRODUCTS 참조)
review_rating	INTEGER	NOT NULL, CHECK(review_rating BETWEEN 1 AND 5)	평점 (1-5)
review_comment	TEXT	NULL	리뷰 내용

키 정의:

- **Primary Key**: review_id
- **Foreign Key**:
  - order_id → ORDERS(order_id)
  - product_id → PRODUCTS(product_id)
- **Alternate Key**: (order_id, product_id) - 한 주문의 제품당 하나의 리뷰

비즈니스 규칙:

- 리뷰는 주문과 제품에 연결되어야 한다
- 평점은 1~5 사이의 값이어야 한다
- 리뷰 내용은 선택사항이다
- 한 주문의 특정 제품에 대해 하나의 리뷰만 작성 가능

참조 무결성:

```
order_id REFERENCES ORDERS(order_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE

product_id REFERENCES PRODUCTS(product_id)
    ON DELETE RESTRICT
    ON UPDATE CASCADE
```

무결성 제약조건

1. 도메인 무결성 (Domain Integrity)

각 속성은 정의된 도메인 내의 값만 가질 수 있습니다.

```sql
-- 가격은 음수가 될 수 없음
product_price >= 0

-- 수량은 양수여야 함
quantity > 0

-- 평점은 1~5 사이
review_rating BETWEEN 1 AND 5

-- 주문 상태는 정해진 값만 가능
order_status IN ('Processing', 'Shipped', 'Delivered', 'Cancelled')

-- 결제 수단은 정해진 값만 가능
payment_method IN ('Card', 'BankTransfer', 'KakaoPay', 'Payco')
```

---

2. 개체 무결성 (Entity Integrity)

모든 테이블의 기본키는 NULL 값을 가질 수 없습니다.

```
CUSTOMERS: customer_id IS NOT NULL
PRODUCTS: product_id IS NOT NULL
ORDERS: order_id IS NOT NULL
ORDER_ITEMS: order_item_id IS NOT NULL
REVIEWS: review_id IS NOT NULL
```

---

3. 참조 무결성 (Referential Integrity)

외래키는 참조하는 테이블의 기본키 값이거나 NULL이어야 합니다.

ORDERS 테이블

```
∀ o ∈ ORDERS:
  o.customer_id ∈ CUSTOMERS.customer_id
```

"모든 주문의 customer_id는 CUSTOMERS 테이블에 존재해야 한다"

ORDER_ITEMS 테이블

```
∀ oi ∈ ORDER_ITEMS:
  oi.order_id ∈ ORDERS.order_id ∧
  oi.product_id ∈ PRODUCTS.product_id
```

"모든 주문 항목은 존재하는 주문과 제품을 참조해야 한다"

REVIEWS 테이블

```
∀ r ∈ REVIEWS:
  r.order_id ∈ ORDERS.order_id ∧
  r.product_id ∈ PRODUCTS.product_id
```

"모든 리뷰는 존재하는 주문과 제품을 참조해야 한다"

---

4. 키 무결성 (Key Integrity)

Primary Key 제약

- 모든 PK는 유일(UNIQUE)하고 NOT NULL

Unique 제약

```
CUSTOMERS.customer_email: 고유해야 함
ORDERS.payment_txn_id: 고유해야 함
ORDER_ITEMS.(order_id, product_id): 복합 유일키
REVIEWS.(order_id, product_id): 복합 유일키
```

---

5. 비즈니스 규칙 제약

```
sql
```

```
-- 할인 금액은 제품 가격보다 클 수 없음
CHECK (discount <= (SELECT product_price * quantity
                    FROM PRODUCTS
                    WHERE product_id = ORDER_ITEMS.product_id))

-- 주문 날짜는 미래일 수 없음
CHECK (order_date <= CURRENT_DATE)

-- 전화번호 형식 검증
CHECK (customer_phone ~ '^\d{3}-\d{4}-\d{4}$')

-- 이메일 형식 검증
CHECK (customer_email ~ '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}$')
```

함수 종속성 (Functional Dependencies)

CUSTOMERS

customer_id → customer_name, customer_email, customer_phone
customer_email → customer_id, customer_name, customer_phone

완전 함수 종속: 모든 비키 속성이 PK에 완전히 종속됨

PRODUCTS

product_id → product_name, product_category, product_price

완전 함수 종속: 모든 비키 속성이 PK에 완전히 종속됨

ORDERS

order_id → order_date, customer_id, shipping_address, billing_address,
payment_method, payment_txn_id, order_status

payment_txn_id → order_id (Alternate Key)

완전 함수 종속: 모든 비키 속성이 PK에 완전히 종속됨
이행적 종속 없음: customer_id를 통한 이행 종속이 제거됨

ORDER_ITEMS

order_item_id → order_id, product_id, quantity, discount
(order_id, product_id) → order_item_id, quantity, discount

완전 함수 종속: 모든 비키 속성이 PK에 완전히 종속됨

REVIEWS

review_id → order_id, product_id, review_rating, review_comment
(order_id, product_id) → review_id, review_rating, review_comment

완전 함수 종속: 모든 비키 속성이 PK에 완전히 종속됨

정규형 분석

제1정규형 (1NF) ✅

- 모든 속성이 원자값(atomic value)을 가짐
- 반복 그룹이 없음
- 각 속성이 단일 값만 저장

제2정규형 (2NF) ✅

- 1NF를 만족
- 부분 함수 종속이 제거됨
- 모든 비키 속성이 전체 기본키에 완전히 종속

제3정규형 (3NF) ✅

- 2NF를 만족
- 이행적 함수 종속이 제거됨
- 비키 속성 간의 종속성이 없음

보이스-코드 정규형 (BCNF) ✅

- 3NF를 만족
- 모든 결정자가 후보키임
- 현재 스키마는 BCNF를 만족함

관계 대수 표현

고객의 모든 주문 조회

σ(customer_id = 1)(ORDERS)

특정 주문의 모든 상품

```
π(product_name, quantity, discount)(
    ORDER_ITEMS ⋈ PRODUCTS
    WHERE order_id = 1001
)
```

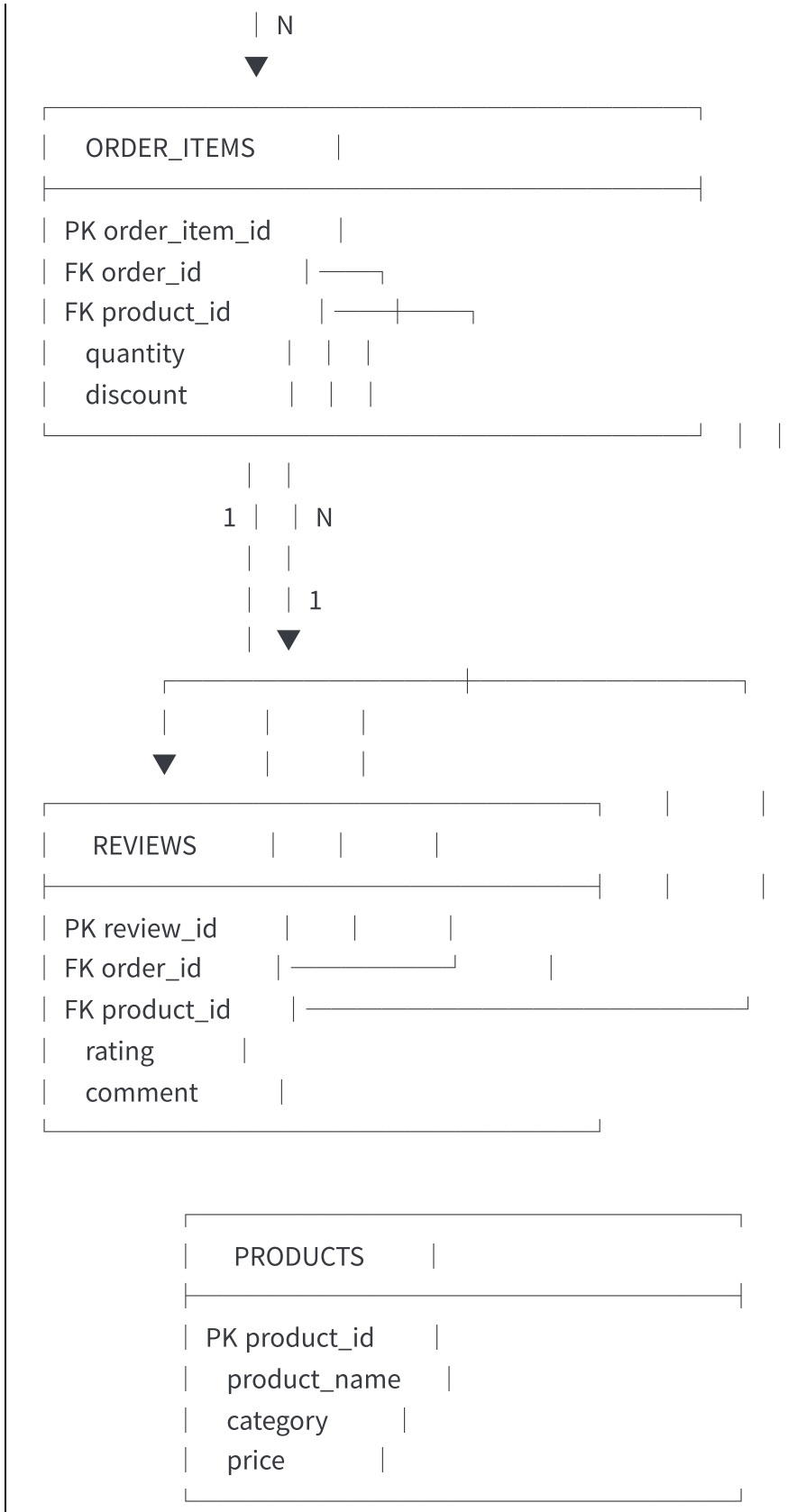
고객별 총 주문 금액

```
γ(customer_id; SUM(product_price * quantity - discount) AS total)(
    CUSTOMERS ⋈ ORDERS ⋈ ORDER_ITEMS ⋈ PRODUCTS
)
```

평점이 4점 이상인 제품

```
π(product_name)(
    PRODUCTS ⋈ (σ(review_rating >= 4)(REVIEWS))
)
```

스키마 다이어그램 (텍스트)



카디널리티 요약

관계	카디널리티	설명
CUSTOMERS ↔ ORDERS	1:N	한 고객이 여러 주문
ORDERS ↔ ORDER_ITEMS	1:N	한 주문에 여러 항목
PRODUCTS ↔ ORDER_ITEMS	1:N	한 제품이 여러 주문에

관계	카디널리티	설명
ORDER_ITEMS ↔ REVIEWS	1:0..1	항목당 리뷰 0개 또는 1개

논리 스키마 검증 체크리스트

✅ 정규화 검증

- [x] 제1정규형 만족 (원자값)
- [x] 제2정규형 만족 (부분 종속 제거)
- [x] 제3정규형 만족 (이행 종속 제거)
- [x] BCNF 만족 (모든 결정자가 후보키)

✅ 무결성 제약조건

- [x] 도메인 무결성 정의됨
- [x] 개체 무결성 정의됨 (PK NOT NULL)
- [x] 참조 무결성 정의됨 (FK 관계)
- [x] 키 무결성 정의됨 (UNIQUE 제약)

✅ 비즈니스 규칙

- [x] 모든 비즈니스 규칙이 제약조건으로 표현됨
- [x] 데이터 타입이 적절히 선택됨
- [x] 필수/선택 속성이 명확히 정의됨

✅ 관계 정의

- [x] 모든 관계가 명확히 정의됨
- [x] 카디널리티가 올바르게 설정됨
- [x] 참조 무결성 액션이 정의됨 (CASCADE/RESTRICT)

💡 논리 스키마 설계 원칙

1. **명확성**: 모든 엔티티, 속성, 관계가 명확히 정의됨
2. **완전성**: 비즈니스 요구사항을 모두 반영함
3. **일관성**: 명명 규칙과 데이터 타입이 일관됨
4. **무결성**: 모든 제약조건이 명시됨
5. **독립성**: DBMS에 독립적인 설계
6. **확장성**: 향후 변경이 용이한 구조

📝 요약

이 논리 스키마는:

- ✅ **5개의 엔티티**로 구성됨
- ✅ **BCNF까지 정규화**되어 데이터 중복이 최소화됨
- ✅ **명확한 제약조건**으로 데이터 무결성 보장
- ✅ **함수 종속성**이 명확히 정의됨
- ✅ **물리 스키마로 전환** 준비 완료

다음 단계는 이 논리 스키마를 특정 DBMS(MySQL, PostgreSQL 등)에 맞는 **물리 스키마**로 변환하는 것입니다.