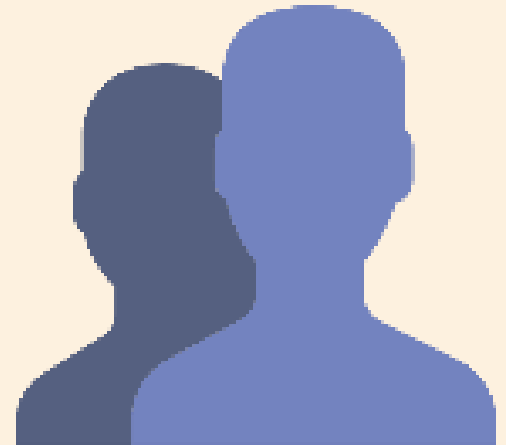


# *DataBase Programming*

## *Team Proejet*

- 편의점 통합 가격 비교 시스템 -

〈 TEAM 날개 - 편의점모아 〉 노현영, 안윤주, 이다연



# 시스템 개요

## 1. 물품 관리

- 물품 추가
- 물품 수정
- 물품 삭제
- 물품 리스트 보기(카테고리별)

## 2. 입고 관리

- 입고 리스트 보기
- 내 브랜드에 입고 물품 추가(다중선택)
- 입고 물품 삭제

## 3. 이벤트 관리

- 이벤트 종류 관리
- 이벤트 리스트 보기
- 이벤트 추가(물품 다중선택)
- 이벤트 삭제

핵심 기능 - 관리자

## 1. 회원 관리

- 회원 가입
- 로그인
- 회원 정보 수정

## 2. 장바구니 관리

- 장바구니 리스트 보기
- 물품 목록에서 장바구니로 추가(다중 선택)
- 물품 삭제

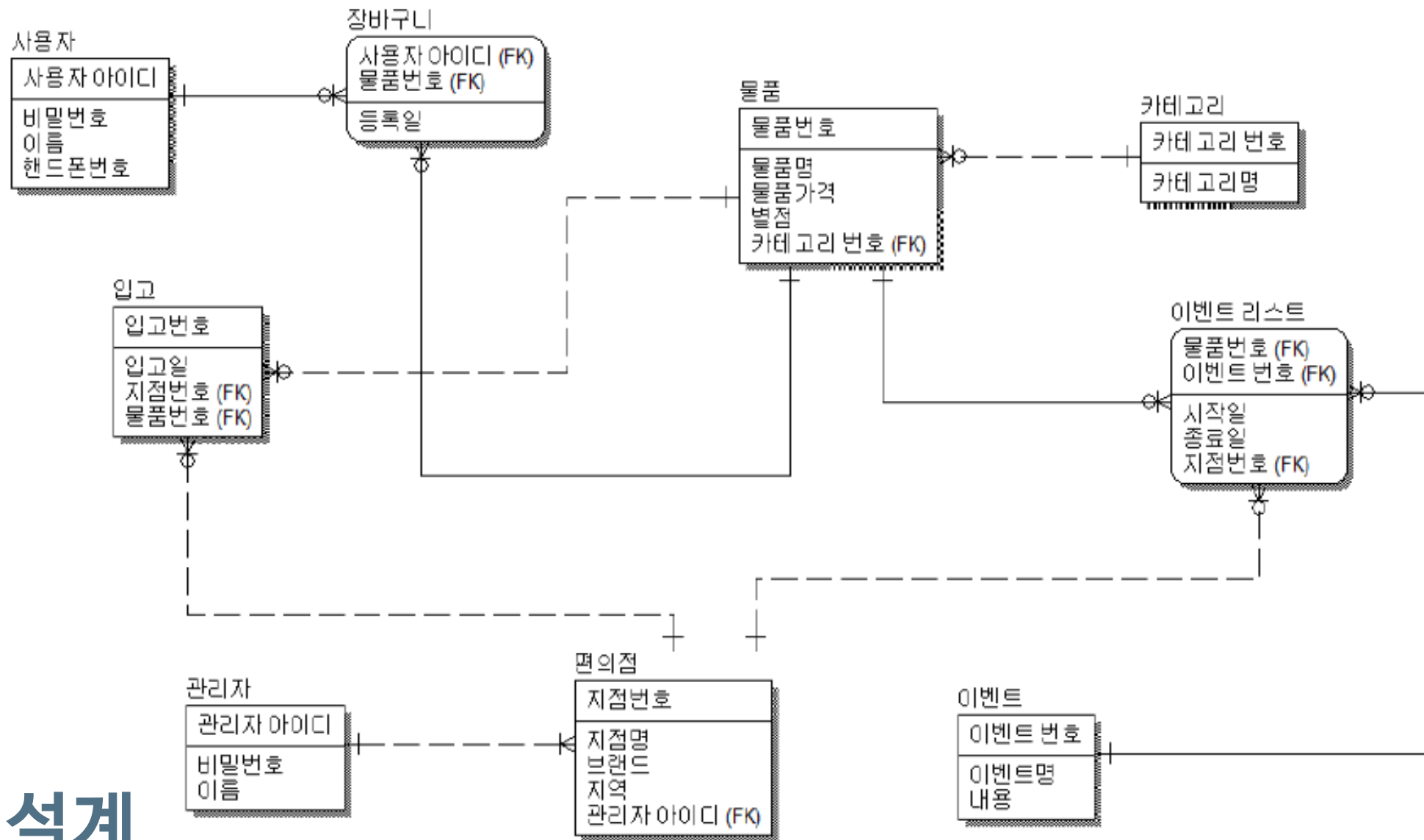
## 3. 추천 상품

- 좋아요가 많은 순서, 찜한 횟수가 많은 순서 별로 상품 검색
- 오늘의 핫딜(찜한상품 중 이벤트인 상품)
- 관심 카테고리의 신상품 추천

핵심 기능 - 사용자

# 모든 핵심 기능 구현 완료

## 구현 완료된 기능



ERWin

# 시스템 구조 설계

### BASKET

- BASKET\_DATE
- USER\_ID
- PRODUCT\_ID

### BRAND

- BRAND\_ID
- BRAND\_NAME
- MANAGERZZZ  
ZZ\_ID

### CATEGORY

- CATEGORY\_ID
- CATEGORY\_NAME

## 시스템 구조 설계

## ELIST

- ELIST\_ID
- EVENT\_ID
- BRAND\_ID
- REC\_ID
- ELIST\_CONTENT
- START\_DATE
- END\_DATE

## EVENT

- EVENT\_ID
- EVENT\_NAME
- CONTENT
- DISCOUNT

## MANAGER

- MANAGER\_ID
- MANAGER\_PWD
- MANAGER\_NAME

# 시스템 구조 설계



## PRODUCT

- PRODUCT\_ID
- PRODUCT\_NAME
- PRODUCT\_PRICE
- CATEGORY\_ID
- SOCRE

## RECEIVING

- REC\_ID
- REC\_DATE
- BRAND\_ID
- PRODUCT\_ID

## USERINFO

- USER\_ID
- USER\_PWD
- USER\_NAME
- PHONE

# 시스템 구조 설계

basket\_add.jsp  
basket\_list.jsp  
basket\_product\_view\_for\_add.jsp  
basket\_product\_view.jsp

login.jsp  
loginCheck.jsp  
manager\_main.jsp  
customer\_main.jsp

user.css

elist\_addlist.jsp  
elist\_list.jsp  
elist\_modify.jsp  
elist\_view.jsp  
elist\_write.jsp

product\_list.jsp  
product\_modify.jsp  
product\_view.jsp  
product\_write.jsp

searchPage.jsp  
searchPageForBasket.jsp  
searchPageForScore.jsp

event\_list.jsp  
event\_modify.jsp  
event\_view.jsp  
event\_write.jsp

receiving\_list.jsp  
receiving\_product\_view.jsp  
receiving\_write.jsp  
recommend\_list.jsp

user\_list.jsp  
user\_modify.jsp  
user\_view.jsp  
user\_write.jsp

## 시스템 구조 설계(View)

## controller.action

- ▶ Action.java
- ▶ DeleteAction.java
- ▶ IllegalCommandException.java
- ▶ InsertAction.java
- ▶ InsertFormAction.java
- ▶ ListAction.java
- ▶ LoginAction.java
- ▶ LoginFormAction.java
- ▶ LogoutAction.java
- ▶ MlistAction.java
- ▶ UpdateAction.java
- ▶ UpdateFormAction.java
- ▶ ViewAction.java

## controller.action.basket

- ▶ BasketAddFormAction.java
- ▶ BasketAddOnlyOneAction.java
- ▶ BasketByCategoryAction.java
- ▶ BasketDeleteAction.java
- ▶ BasketInsertAction.java
- ▶ BasketListAction.java
- ▶ BasketProductViewAction.java
- ▶ ClistAction.java
- ▶ ScorePlusAction.java

## controller.action.elist

- ▶ CustomerElistListAction.java
- ▶ ElistDeleteAction.java
- ▶ ElistInsertAction.java
- ▶ ElistInsertFormAction.java
- ▶ ElistListAction.java
- ▶ ElistRecAddFormAction.java
- ▶ ElistRecInsertAction.java
- ▶ ElistSearchByCategoryAction.java
- ▶ ElistUpdateAction.java
- ▶ ElistUpdateFormAction.java
- ▶ ElistViewAction.java
- ▶ RecSearchByCategoryAction.java

## controller.action.event

- ▶ EventDeleteAction.java
- ▶ EventInsertAction.java
- ▶ EventInsertFormAction.java
- ▶ EventListAction.java
- ▶ EventUpdateAction.java
- ▶ EventUpdateFormAction.java
- ▶ EventViewAction.java

## controller.action.product

- ▶ ProductDeleteAction.java
- ▶ ProductInsertAction.java
- ▶ ProductInsertFormAction.java
- ▶ ProductListAction.java
- ▶ ProductListSearchByCategoryAction.java
- ▶ ProductUpdateAction.java
- ▶ ProductUpdateFormAction.java
- ▶ ProductViewAction.java

## controller.action.receiving

- ▶ ReceivingAddAction.java
- ▶ ReceivingDeleteAction.java
- ▶ ReceivingInsertAction.java
- ▶ ReceivingInsertListAction.java
- ▶ ReceivingListAction.java
- ▶ ReceivingListSearchByCategory.java

## controller.action.search

- ▶ BasketProductViewForAddAction.java
- ▶ SearchHighBasketAction.java
- ▶ SearchHighScoreAction.java
- ▶ SearchNewProductAction.java

# 시스템 구조 설계(Controller)

## model

- ExistedUserException.java
- PasswordMismatchException.java
- Userinfo.java
- UserinfoAnalysis.java
- UserinfoDAO.java
- UserinfoManager.java
- UserNotFoundException.java

## model.basket

- Basket.java
- BasketDAO.java
- BasketManager.java

## model.event

- Event.java
- EventDAO.java
- EventManager.java

## model.mnginfo

- Mnginfo.java
- MnginfoDAO.java
- MnginfoManager.java

## model.elist

- Elist.java
- ElistDAO.java
- ElistManager.java

## model.product

- Product.java
- ProductDAO.java
- ProductManager.java

## model.receiving

- Receiving.java
- ReceivingDAO.java
- ReceivingManager.java

## model.search

- Search.java
- SearchDAO.java
- SearchManager.java

## function

- RecommendDAO.java
- RecommendListAction.java

# 시스템 구조 설계(Model)

- 카테고리 별 물품 검색
  - 콤보박스에 category\_id를 value값으로 주어 선택하게 함
- 신상품 검색
  - SYSDATE between SYSDATE -7 AND SYSDATE
- 많이 찜한 횟수 검색
  - COUNT(\*) 이용
- 오늘의 핫딜
  - 찜한 상품(장바구니에 있는 상품) 중에서 이벤트가 진행 중인 상품이 있으면 할인 가격을 함께 출력해줌
  - ELIST에 있는 이벤트이며 SYSDATE가 start\_date와 end\_date 사이에 존재
  - product\_price를 query문으로 받아 온 후 switch문을 통해 할인가를 산정

## 구현 방법

```
List<Product> productList = null;
if ( (start >= 0) && rs.absolute(start) ) {
    productList = new ArrayList<Product>();
    do {
        Product product = new Product();
        product.setProduct_id(rs.getInt("product_id"));
        product.setProduct_price(rs.getInt("product_price"));
        product.setProduct_name(rs.getString("product_name"));
        product.setBrand_name(rs.getString("brand_name"));
        product.setEvent_id(rs.getInt("event_id")); //일반 이벤트 아이디
        product.setElist_content(rs.getString("elist_content")); //진행중인 이벤트 내용
        product.setContent(rs.getString("content")); //일반 이벤트 이름

        switch(product.getEvent_id()) {
            case 112:
                product.setDc_price(product.getProduct_price() / 2);
                break;

            case 212:
                product.setDc_price(product.getProduct_price() / 3);
                break;

            case 312:
                product.setDc_price(product.getProduct_price() / 4);
                break;

            case 101:
                product.setDc_price((int) (product.getProduct_price() * 0.9));
                break;

            case 201:
                product.setDc_price((int) (product.getProduct_price() * 0.8));
                break;
        }
    } while(rs.next());
}
```



```
//새로운 상품 등록
public int create(String userID, int product_id) throws SQLException {
    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    int brand_id = 0;
    try {
        con = ds.getConnection();

        StringBuffer findQuery = new StringBuffer();
        findQuery.append("SELECT brand_id from BRAND where manager_id = ?");
        pstmt = con.prepareStatement(findQuery.toString());
        pstmt.setString(1, userID);
        rs = pstmt.executeQuery();

        while(rs.next()) {
            brand_id = rs.getInt("brand_id");
        }

        pstmt = null;

        StringBuffer insertQuery = new StringBuffer();
        insertQuery.append("INSERT INTO RECEIVING VALUES ");
        insertQuery.append("(receiving_seq.nextval, SYSDATE, ?, ?)");

        pstmt = con.prepareStatement(insertQuery.toString());
        pstmt.setInt(1, brand_id);
        pstmt.setInt(2, product_id);

        int result = pstmt.executeUpdate();
        return result;
    } finally {
        if ( pstmt != null ){
            pstmt.close();
        }
        if ( con != null ){
            con.close();
        }
    }
}
```

## 문제 해결 방법

한 query로 원하는 결과가 나오지 않으면 2번 이상 query를 돌려 결과를 얻어냄

- 체크박스로 다중 선택하여 값 넘겨 받아오기(request에 저장이 되지 않아 session에 저장하여 가져옴)
- 사용자 관심 카테고리의 신상품
  - 사용자가 장바구니에 담은 물품 중 해당하는 물품이 가장 많은 카테고리를 뽑아낸다.
  - 그 카테고리의 입고 신상품을 가져온다.

```
findCQuery.append("select cate from ");
findCQuery.append("(select c.category_id as cate, count(*) as cnt ");
findCQuery.append("from category c, basket b, product p ");
findCQuery.append("where p.category_id=c.category_id and b.user_id=? and b.product_id=p.product_id ");
findCQuery.append("group by c.category_id) ");
findCQuery.append("where cnt = (select max(cnt) ");
findCQuery.append("from (select c.category_id, count(*) as cnt ");
findCQuery.append("from category c, basket b, product p ");
findCQuery.append("where p.category_id=c.category_id and b.user_id=? and b.product_id=p.product_id ");
findCQuery.append("group by c.category_id))");

pstmt = con.prepareStatement(findCQuery.toString());

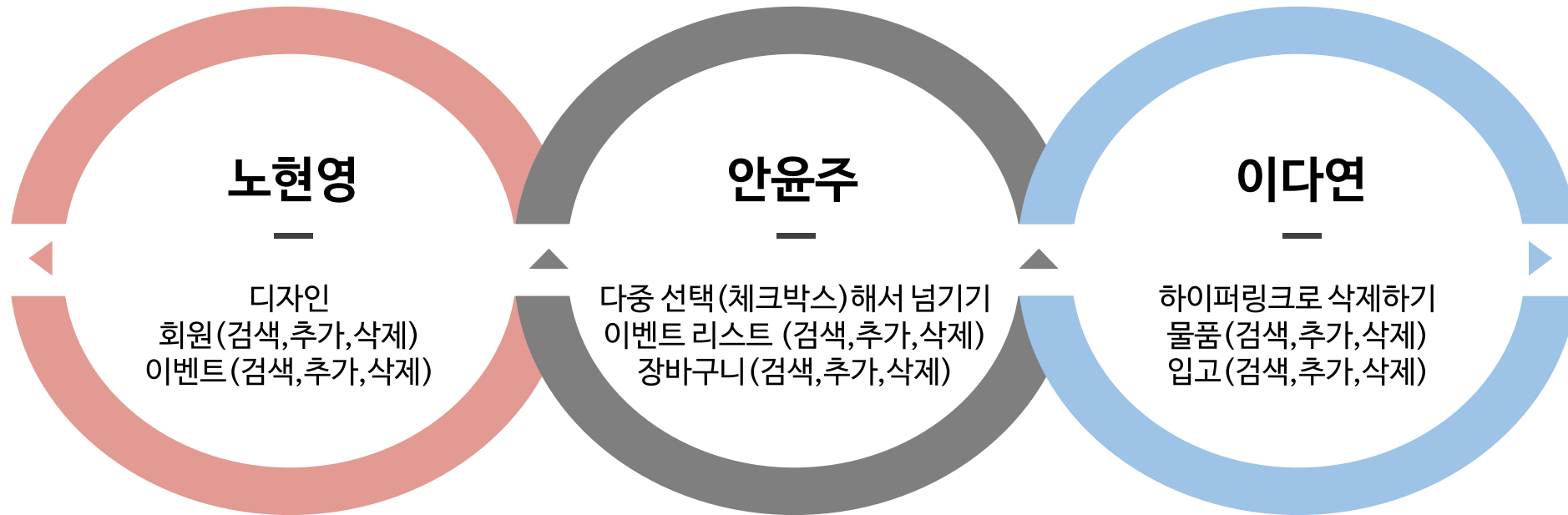
pstmt.setString(1, userId);
pstmt.setString(2, userId);

rs = pstmt.executeQuery();
rs.next();
cate = rs.getInt("cate");

System.out.print(cate);
pstmt = null;
StringBuffer findQuery = new StringBuffer();
findQuery.append("select p.product_id, p.product_name, b.brand_name, p.score, c.category_name, p.product_price ");
findQuery.append("from product p, receiving r, brand b, category c ");
findQuery.append("where r.rec_date between sysdate-30 and sysdate and p.category_id=c.category_id "
+ "and b.brand_id=r.brand_id and r.product_id = p.product_id and p.category_id = ?");
```

## 구현 방법





## 역할 분담

# 편의점 통합 가격 비교 시스템



관리자 혹은 사용자  
를 체크해주세요!



● 관리자 ○ 사용자	
사용자 아이디	<input type="text"/>
비밀번호	<input type="password"/>

[로그인](#) [회원가입](#)

시연(Demo)

*Thank You*