2025 – 2학기 DB응용 프로젝트 결과보고서

기본 사항			
분반 및 조	(1)분반 (2) 조	**담당교수**	이중화
프로젝트 주제	영화 예약 시스템		
팀구성원 (학번/성명)	20212979 임진호	20212977 이규찬	
	20233065 이지민	20213034 임민욱	
개발 환경	Visual Studio 2022, Proc, Oracle SQL Developer19.2.1		

1. 프로젝트 개요

1.1 프로젝트 개요

본 프로젝트는 사용자가 손쉽게 영화 정보를 확인하고, 좌석을 선택하여 예매할 수 있는 데이터베이스 기반 영화 예매 시스템을 구축하는 것을 목표로 한다.
사용자는 회원가입을 통해 개인 계정을 생성하고, 현재 상영 중인 영화 목록과 상영 일정을 직관적인 인터페이스로 조회할 수 있다. 또한 원하는 상영 시간과 좌석을 선택하여 예매를 진행하고, 기존 예매 내역을 확인하거나 변경·취소하는 등 다양한 기능을 편리하게 이용할 수 있다. 시스템은 영화 정보, 상영 스케줄, 좌석 상태, 사용자 예매 내역을 데이터베이스로 체계적으로 관리하여 데이터의 무결성과 정확성을 보장한다. 이를 통해 중복 예매를 방지하고, 좌석 배정의 신뢰성과 안정성을 확보한다.
본 프로그램은 영화 예매 과정 전반을 자동화하고 간소화하여 사용자가 보다 빠르고 정확하게 서비스를 이용할 수 있도록 지원하는 것을 주요 목표로 한다.

1.2 프로젝트 목적

- **효율적인 영화 예매 처리**: 영화 정보와 상영 일정을 체계적으로 관리하여 사용자가 빠르고 정확하게 예매를 진행할 수 있도록 지원한다.
- **사용자 편의성 증대**: 직관적인 인터페이스를 제공하여 영화 검색, 좌석 선택, 예매 확인 및 취소 과정을 쉽게 이용할 수 있도록 사용자 경험을 향상시킨다.
- **좌석 중복 예약 방지**: 데이터베이스 기반 좌석 관리로 동일 좌석의 중복 예약을 방지하고, 정확한 좌석 배정을 보장한다.
- **예매 관리 기능 강화**: 사용자가 자신의 예매 내역을 언제든지 조회하고, 필요한 경우 변경 또는 취소할 수 있도록 함으로써 관리 편의성을 높인다.

2. 프로젝트 구성요소

목표/기준 설정	합 성	분 석	제 작	시 험	평 가	결과 도출	기 타
√		√	√		√	√	

목표/기준 설정	본 프로젝트의 최종 목표는 사용자가 영화 정보 조회부터 좌석 선택, 예매, 변경, 취소까지 모든 기능을 직관적이고 정확하게 이용할 수 있는 **영화 예매 관리 시스템**을 구축하는 것이다. 이를 위해 영화(MOVIES), 상영 스케줄(SCHEDULES), 좌석(SEATS), 회원(USERS), 예매(BOOKINGS) 정보를 데이터베이스 기반으로 통합 관리한다. 특히 **중복 예매 방지, 정확한 좌석 배정, 사용자 편의성 제공**을 핵심 기준으로 설정하며, 데이터의 무결성과 시스템의 안정성을 확보하는 데 주력한다.
분 석	프로젝트 요구 사항 분석을 통해 영화 예매 시스템에서 반드시 필요한 핵심 기능을 다음과 같이 정의하였다. • **사용자 관리**: 회원가입 및 사용자 정보 저장 • **영화 정보 관리**: MOVIES 테이블의 영화 제목, 등급, 상영 시간을 기반으로 영화 목록 제공 • **상영 일정 선택:** SCHEDULES 테이블을 이용해 영화별 일정/상영관/시작시간/가격 제공 • **좌석 선택:** SEATS 테이블의 좌석 행/열 정보를 기반으로 시각적인 좌석 선택 지원 • **예매 트랜잭션 처리**: BOOKINGS 테이블을 통해 예매 생성, 변경, 취소 처리 또한 중복 예매 방지를 위해 동일 일정(schedule_id)과 좌석(seat_id)의 예약 여부를 검증하고, PK/FK/UNIQUE 제약 및 CHECK 조건을 고려하는 전략을 수립하였다. 이러한 분석 과정을 통해 각 테이블 간의 참조 관계(FK)를 설정하고, 실제 프로그램에서 필요한 메뉴 흐름(예매, 변경, 조회 등)을 구체적으로 계획하였다.
제 작	데이터베이스 설계를 기반으로 실제 프로그램을 구현하였다. 우선 USERS, MOVIES, SCHEDULES, SEATS, BOOKINGS **테이블을 생성**하고, 각 테이블에 필요한 PK, FK, CHECK **제약 조건을 설정**하였다. **메뉴 화면**은 회원가입, 영화 예매, 나의 예매 내역 조회, 예매 변경, 예매 취소로 구성하였다. C 프로그램에서 입력값을 받아 DB와 연동하는 프로시저 및 쿼리를 작성하였으며, 예매 처리 시 자동 증가 PK, 예약 상태, 생성일 등을 자동 관리하도록 설계하였다. 이를 통해 실제로 영**화 선택, 일정 선택, 좌석 선택, 예매 완료의 흐름**이 작동하도록 제작하였다.
결과 도출	프로그램 실행 결과, 영화 예매, 일정 조회, 좌석 선택, 예매 내역 조회/변경/취소 기능이 정상적으로 작동함을 확인하였다. 출력 화면은 프로젝트 요구에 맞게 구성되었으며, 데이터베이스와의 연동을 통해 실시간으로 정확한 예매 처리와 좌석 관리가 가능함을 확인하였다. 또한 BOOKING 테이블에 저장된 결과 데이터를 기반으로, 좌석 중복 예약이 발생하지 않도록 무결성이 유지되는 부분도 검증하였다.
평 가	프로젝트의 기능들을 초기 목표·기준과 비교하여 평가한 결과, 사용자 중심의 예매 흐름 제공, 중복 예매 방지 처리, 직관적인 데이터베이스 구조 구축 등 핵심 목표를 대부분 충족하였다. 테스트 과정에서 확인된 오류나 개선사항은 차후 버전에서 보완할 수 있으며, 본 프로젝트는 DB 기반 예매 시스템의 구조적 설계와 기능 구현 측면에서 의미 있는 성과를 도출하였다.

3. 현실적 제한 조건

경제성	안전성	신뢰성	미 학	윤리성	사회적 영향	생산성/내구성	산업표준	기타
		√	√			√		

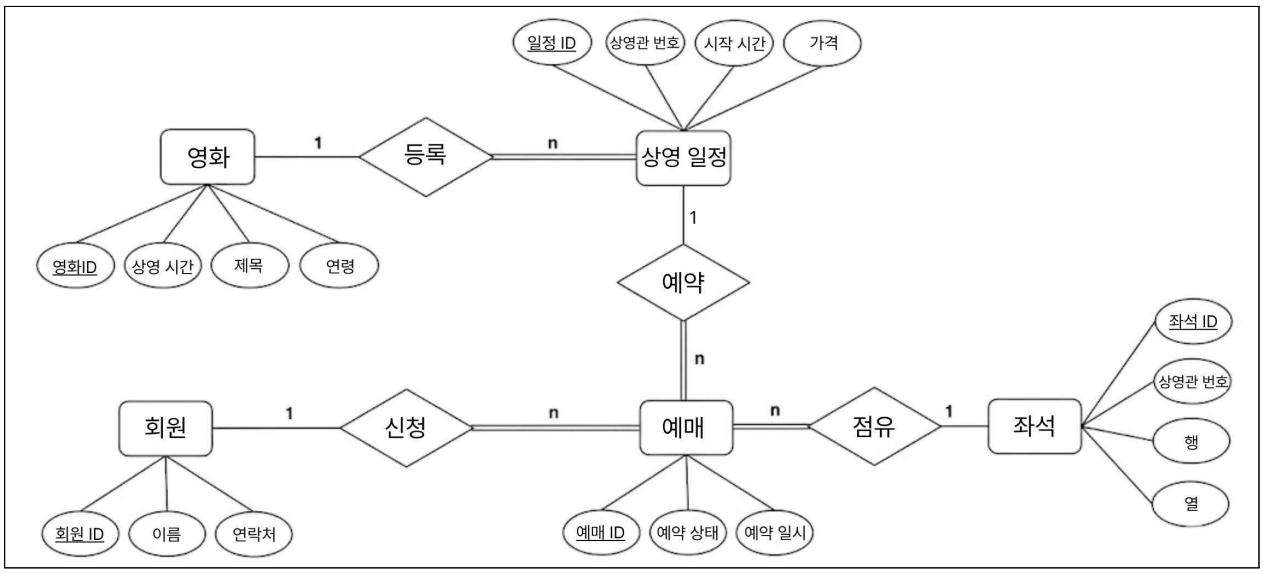
신뢰성	본 영화 예매 시스템은 데이터베이스를 기반으로 안정적이고 정확한 예매 처리를 수행하는 것을 목표로 한다. MOVIES, SCHEDULES, SEATS, USERS, BOOKINGS 테이블이 상호 참조(FK) 관계를 맺고 있어 예매 생성, 변경, 취소의 전 과정에서 데이터 무결성이 철저히 유지된다. 특히 중복 예약을 방지하기 위해 동일 일정(schedule_id)에 동일 좌석(seat_id)이 이미 예약되어 있는지 확인하는 검증 로직을 적용하여, 잘못된 데이터가 저장되는 것을 원천 차단하였다. 이러한 설계 후 다양한 테스트를 통해 시스템이 의도대로 작동함을 검증함으로써, 전체 예매 절차에 대한 안정성과 신뢰성을 확보하였다.
미 학	본 시스템은 콘솔 환경에서 구동되지만, 메뉴 배치, 텍스트 정렬, 구분선 활용 등을 통해 사용자에게 명확한 정보를 제공하도록 설계하였다. 영화 목록, 좌석 현황, 상영 일정 등은 표 형태로 정렬하여 가독성을 높였으며, 예매 정보 입력 화면은 단계별 구성으로 직관적인 흐름을 유지하였다. 또한 불필요한 텍스트나 복잡한 메뉴는 최소화하여 깔끔하고 정돈된 UI를 구현하였으며, 이를 통해 콘솔 환경의 제약 안에서도 사용자 경험(UX)을 최우선으로 고려한 화면 구성을 제공한다.
생산성	실제 영화관의 좌석 관리 및 예매 흐름을 기반으로 설계되어, 사용자 관리부터 영화 정보 조회, 일정 및 좌석 선택, 예매·변경·취소 기능이 하나의 프로그램 안에서 통합적으로 처리된다. 데이터베이스와 연동된 구조를 통해 좌석 상태, 예매 이력, 일정 정보 등이 실시간으로 자동 갱신되므로, 운영 과정에서의 반복적이고 수동적인 작업을 줄여 업무 효율성을 높였다. 아울러 SEQUENCE, TRIGGER, CHECK 제약 조건 등을 활용하여 데이터의 자동 증가, 형식 검증, 값 제약 등을 시스템적으로 처리함으로써, 추가적인 오류를 방지하고 향후 시스템의 확장성과 유지보수 편의성을 확보하였다.

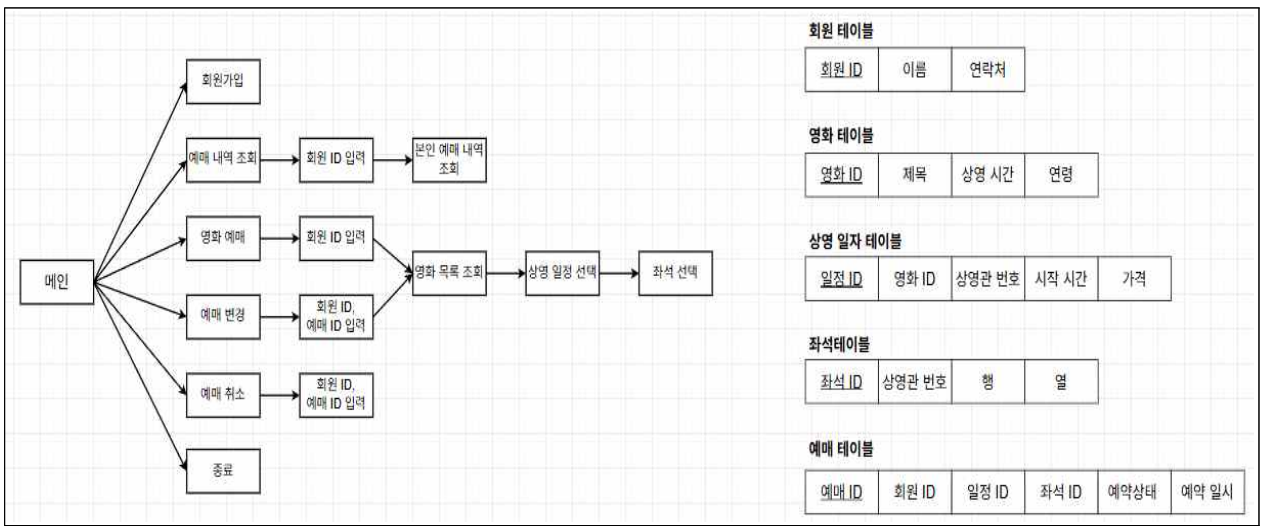
4. 프로젝트 설계 및 구현 내용

4.1 응용 분석

본 프로젝트는 사용자 요구를 충족하는 직관적이고 체계적인 영화 예매 관리 시스템 구축을 목표로 한다. 사용자는 회원가입 후 실시간으로 상영 중인 영화를 조회하고, 영화별 상영 일정 및 가격 정보를 확인할 수 있다. 이를 바탕으로 원하는 시간대를 선택하고, 시각화된 좌석 배치 화면을 통해 직관적으로 좌석을 지정하여 예매를 진행하게 된다. 또한, 예매 내역 조회 기능을 통해 본인의 예약 정보를 확인하고, 필요시 일정 변경, 좌석 변경, 예매 취소 등의 기능을 이용하여 예매 정보를 유연하게 수정할 수 있도록 설계하였다. 시스템 측면에서는 영화 정보, 상영 일정, 좌석 정보, 예매 트랜잭션을 데이터베이스 기반으로 통합 관리한다. 이를 통해 좌석 중복 예약을 시스템적으로 방지하고 정확한 예매 처리를 보장한다. 결과적으로 본 시스템은 전체 예매 흐름의 자동화를 통해 운영의 안정성과 효율성을 확보하며, 사용자에게 언제나 편리하고 신뢰성 있는 서비스를 제공하는 기반을 마련한다.

4.2 E-R 다이어그램





4.3 테이블 명세서

USERS

테이블명	USERS	Table 기술서	작성일	2025.11.25	Page
System	영화 예매 시스템		작성자	**임진호**	/
테이블 설명	영화 예매 서비스를 이용하는 사용자의 식별 정보와 연락처를 관리하는 테이블				

No	Attribute	Data Type	NN	Ky	Default	Description
1	USER_ID	NUMBER	Y	PK	Null	회원ID
2	NAME	VARCHAR2(50)	Y	-	Null	이름
3	CONTACT	VARCHAR2(20)	Y	-	Null	연락처

제약 조건

constraint USERS_USER_ID_PK primary key (USER_ID)
constraint USERS_CONTACT_CK check (REGEXP_LIKE(CONTACT, '^010-[0-9]{4}-[0-9]{4}$'))

MOVIES

테이블명	MOVIES	Table 기술서	작성일	2025.11.25	Page
System	영화 예매 시스템		작성자	**임진호**	/
테이블 설명	영화의 기본 정보(제목, 관람 등급, 상영 시간 등)를 정의하고 관리하는 정적 데이터 테이블				

No	Attribute	Data Type	NN	Ky	Default	Description
1	MOVIE_ID	NUMBER	Y	PK	NULL	영화 ID
2	TITLE	VARCHAR2(100)	Y	-	NULL	제목
3	RATING	VARCHAR2(20)	N	-	NULL	등급
4	DURATION	NUMBER	Y	-	NULL	상영시간

제약 조건

constraint MOVIES_MOVIE_ID_PK primary key (MOVIE_ID)
constraint MOVIES_DURATION_CK check (DURATION > 0)

SEATS

테이블명	SEATS	Table 기술서	작성일	2025.11.25	Page
System	영화 예매 시스템		작성자	**임민욱**	/
테이블 설명	각 상영관에 배치된 물리적인 좌석 위치(행, 열) 정보를 관리하는 정적 테이블				

No	Attribute	Data Type	NN	Ky	Default	Description
1	SEAT_ID	NUMBER	Y	PK	Null	좌석 ID
2	SCREEN_NO	VARCHAR2(20)	Y	-	Null	상영관 번호
3	ROW_CODE	NUMBER	Y	-	Null	행 코드
4	COL_CODE	NUMBER	Y	-	Null	열 코드

제약 조건

constraint SEATS_SEAT_ID_PK primary key (SEAT_ID)

SCHEDULES

테이블명	SCHEDULES	Table 기술서	작성일	2025.11.25	Page
System	영화 예매 시스템		작성자	**이지민**	/
테이블 설명	특정 영화가 언제, 어느 상영관에서 상영되는지와 티켓 가격 정보를 관리하는 테이블.				

No	Attribute	Data Type	NN	Ky	Default	Description
1	SCHEDULE_ID	NUMBER	Y	PK	Null	일정ID
2	MOVIE_ID	NUMBER	Y	FK	Null	영화ID
3	SCREEN_NO	VARCHAR2(10)	Y	-	Null	상영관 번호
4	START_TIME	VARCHAR2(10)	Y	-	Null	시작 시간
5	PRICE	NUMBER	Y	-	Null	가격

비고

constraint SCHEDULES_SCHEDULE_ID_PK primary key (SCHEDULE_ID)
constraint SCHEDULES_MOVIE_ID_FK foreign key (MOVIE_ID) references MOVIES(MOVIE_ID)
constraint SCHEDULES_PRICE_CK check (PRICE >= 0)

BOOKINGS

테이블명	BOOKINGS	Table 기술서	작성일	2025.11.25	Page
System	영화 예매 시스템		작성자	**이규찬**	/
테이블 설명	회원이 특정 상영 일정과 좌석을 선택하여 결제한 예매 내역(트랜잭션)을 관리하는 테이블				

No	Attribute	Data Type	NN	Ky	Default	Description
1	BOOKING_ID	NUMBER	Y	PK	Null	예매 ID
2	USER_ID	NUMBER	Y	FK	Null	회원 ID
3	SCHEDULE_ID	NUMBER	Y	FK	Null	일정 ID
4	SEAT_ID	NUMBER	Y	FK	Null	좌석 ID
5	STATUS	VARCHAR2(10)	Y	-	Null	예약 상태
6	CREATED_AT	DATE	Y	-	SYSDATE	예약 일시

비 고

constraint BOOKINGS_BOOKING_ID_PK primary key (BOOKING_ID)
constraint BOOKINGS_USER_ID_FK foreign key (USER_ID) references USERS(USER_ID)
constraint BOOKINGS_SEAT_ID_FK foreign key (SEAT_ID) references SEATS(SEAT_ID)
constraint BOOKINGS_STATUS_CK check (STATUS IN ('결제완료', '취소됨', '예약중'))
constraint BOOKINGS_SCHEDULE_ID_FK foreign key (SCHEDULE_ID)
references SCHEDULES(SCHEDULE_ID)

4.4 기능 분석

ID	기능 요구사항	기능 상세 설명
SFR-100	**회원가입**	
SFR-101	회원가입	사용자는 개인 정보를 입력하여 회원가입을 할 수 있다.
SFR-200	**사용자**	
SFR-201	영화 예매	사용자는 영화관에 등록된 영화 목록을 조회할 수 있다.
SFR-202	예매한 내역 보기	사용자는 본인이 예매한 영화 내역을 조회할 수 있다.
SFR-203	예매 변경	사용자는 본인이 예매한 내역에 대해 영화, 시간, 좌석 등을 변경할 수 있다.
SFR-204	영화 및 좌석, 상영관	시스템은 영화 리스트, 시간대, 좌석 정보를 제공하며, 이미 예약된 좌석은 '선택 불가(X)'로 표시하여 중복 예매를 시스템적으로 차단한다.
SFR-205	예매 취소	사용자는 예매를 취소할 수 있으며, 취소 완료 시 해당 좌석은 즉시 '예약 가능(O)' 상태로 전환되어 타 사용자가 예매할 수 있다.

4.5 프로그램 수행 내용

4.5.1 DB 연결

DB 연결 코드

```
void db_connect()
{     // ... 변수 선언 생략 ...
      // DB 연결 정보 설정
      strcpy((char *)uid.arr, "se20212979@//sedb.deu.ac.kr:1521/orcl");
      strcpy((char *)pwd.arr, "20212979");

      // DB 접속 시도
      EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;

      if (Error_flag == 1){
            printf("DB 연결 실패!₩n");
            exit(-1);
      }
}
```

설명

- DB 연결: EXEC SQL CONNECT 구문을 사용하여 지정된 계정 정보로 Oracle 데이터베이스 세션을 수립한다. 연결 실패 시 즉시 프로그램을 종료하여 예외 상황을 방지한다.

4.5.2 메인 화면

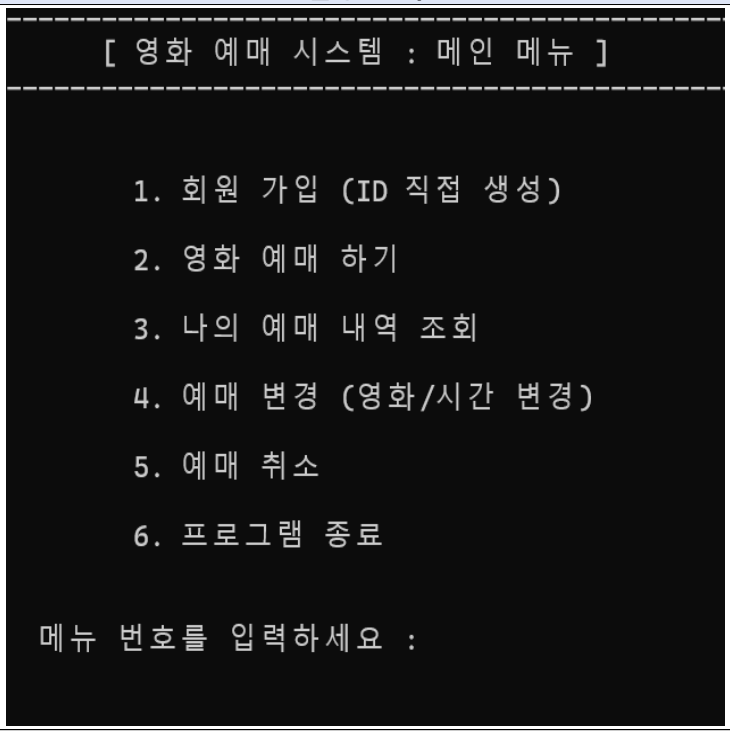
main 메뉴 처리

```
void main()
{     db_connect(); // DB 연결
     while( c != '6') {
          clrscr();
          print_screen("scr_main.txt"); // UI 파일 로드

               switch (c) {
               case '1': fn_signup();              break;  // 1. 회원 가입
               case '2': fn_booking_flow();        break;  // 2. 영화 예매 하기
               case '3': fn_my_booking();          break;  // 3. 나의 예매 내역 조회
               case '4': fn_change_booking(); break;  // 4. 예매 변경
               case '5': fn_cancel();              break;  // 5. 예매 취소
               case '6':                           break;  // 6. 프로그램 종료
               default:                            break;  // 그 외 입력은 무시
          }
     }
     EXEC SQL COMMIT WORK RELEASE; // 종료 시 세션 해제
}
```

설명

- 프로그램 흐름 제어: 프로그램 시작 후 while 반복문을 통해 메인 메뉴 화면을 지속적으로 출력한다. scr_main.txt를 로드하여 UI를 구성하며, switch-case 문을 통해 사용자 입력(1~6)에 따라 각 기능 모듈로 분기 처리한다.

실행 결과



4.5.3 회원가입

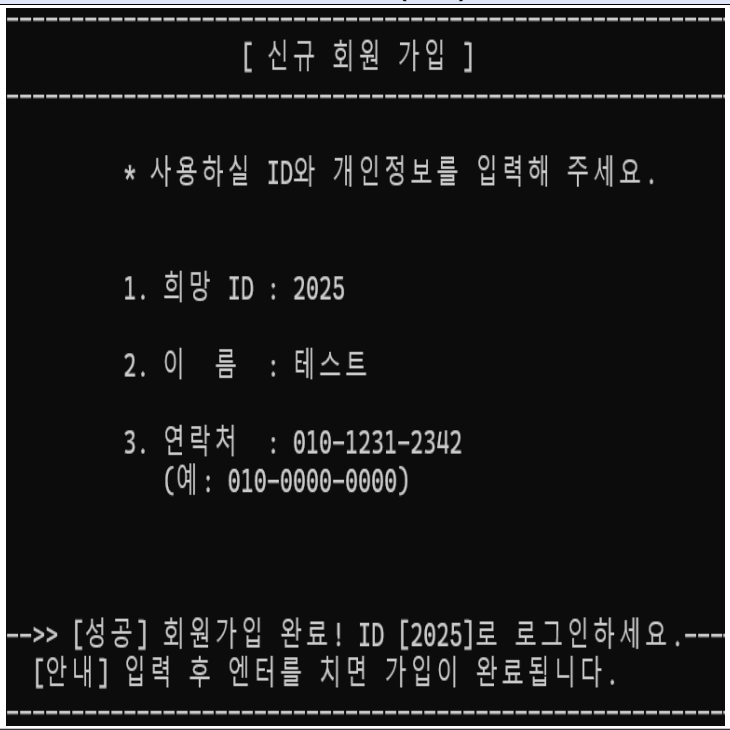
회원가입 코드

```
// proc_sample_all.pc (fn_signup 함수)

// 1. 사용자 입력 처리
gotoxy(x, y);
if(fgets(temp, sizeof(temp), stdin) == NULL) return;
v_id = atoi(temp);

// 2. 중복 ID 검사 (제약조건 위배 방지)
EXEC SQL SELECT count(*) INTO :check_dup FROM Users WHERE user_id = :v_id;
if (check_dup > 0) {
    gotoxy(10, 16);
    printf(">> [오류] 이미 사용 중인 ID입니다. (%d)", v_id);
    getch(); return;
}
if(fgets(v_name, sizeof(v_name), stdin) == NULL) return;
    cleanup_input(v_name);
// 연락처 입력 생략 ...

// 3. DB 삽입 및 트랜잭션 확정
EXEC SQL INSERT INTO Users (user_id, name, contact) VALUES (:v_id, :v_name, :v_contact);
if (sqlca.sqlcode == 0) {
    EXEC SQL COMMIT WORK;
    printf(">> [성공] 회원가입 완료! ID [%d]로 로그인하세요.", v_id);
} else {
    EXEC SQL ROLLBACK WORK;
}
```

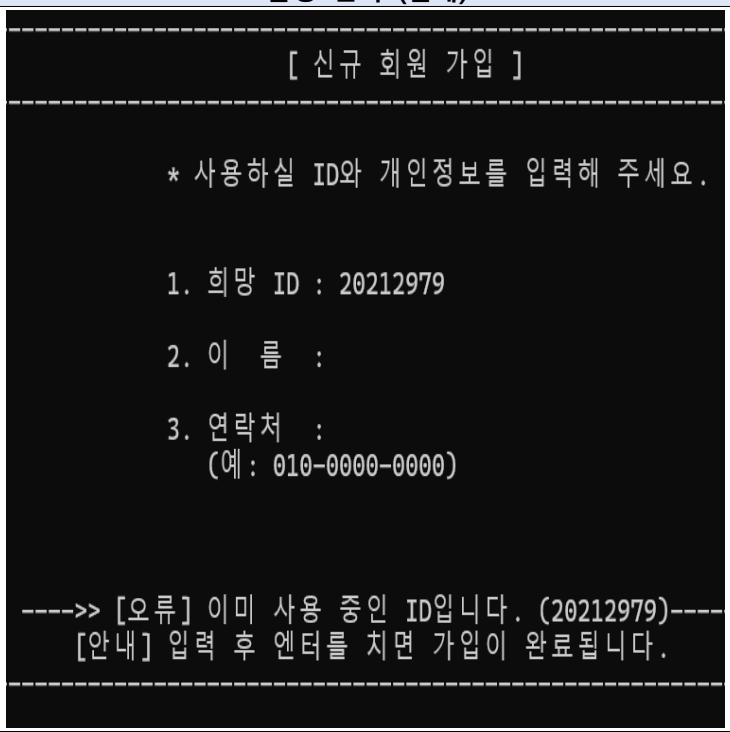
설명

- 중복 검사: INSERT 수행 전 SELECT count(*)를 통해 입력한 ID가 이미 존재하는지 확인합니다. 이는 PK 중복 에러(ORA-00001)를 사전에 방지하여 사용자에게 친절한 에러 메시지를 제공하기 위함입니다.
- 트랜잭션 처리: 데이터 삽입 성공 시 COMMIT을 수행하여 저장을 확정하고, 실패 시 ROLLBACK 하여 데이터 무결성을 유지합니다.

실행 결과 (성공)



실행 결과 (실패)



4.5.4 스케줄 및 좌석 선택

스케줄 및 좌석선택 코드

```
// 1. 영화 선택 후 해당 영화의 스케줄 조회 (CURSOR 사용)
EXEC SQL DECLARE c_sch_sub CURSOR FOR
    SELECT s.schedule_id, ...
    FROM Schedules s, Movies m
    WHERE s.movie_id = :input_mid ...;

// 2. 좌석 상태 표시 (예약 여부 확인)
while(1) {
    // 해당 상영관의 모든 좌석을 가져옴
    EXEC SQL FETCH c_seat_sub INTO :v_seatid, ...;

    // 해당 좌석이 현재 스케줄에 예약되어 있는지 확인 (서브쿼리 역할)
    EXEC SQL SELECT count(*) INTO :v_is_booked
            FROM Bookings
            WHERE schedule_id = :v_selected_sid AND seat_id = :v_seatid;

    // UI 출력 로직
    if (v_is_booked > 0) printf("[X] 예약됨");
    else printf("[O] 가능");
}
```

설명

- Cursor 활용: Schedules와 Movies 테이블을 조인하여 선택한 영화의 상영 시간표를 커서(c_sch_sub)를 통해 리스트 형태로 조회한다.
- 좌석 상태 시각화: Seats 테이블의 전체 좌석을 순회하면서 Bookings 테이블을 조회하여 예약 여부(v_is_booked)를 확인한다. 결과에 따라 [X](예약됨) 또는 [O](가능)로 상태를 구분하여 출력한다.

실행 결과

```
------------------------------------------------------
                  [ 좌석 선택 ]
------------------------------------------------------
  ID       상태          상영관          행        열
------------------------------------------------------

  1    [X] 예약됨       1관         A    열    1번
  2    [X] 예약됨       1관         A    열    2번
  3    [O] 가능         1관         B    열    1번
  4    [O] 가능         1관         B    열    2번
  5    [O] 가능         1관         B    열    3번


>> 선택할 [좌석 ID] :
```

4.5.5 중복 예약 방지

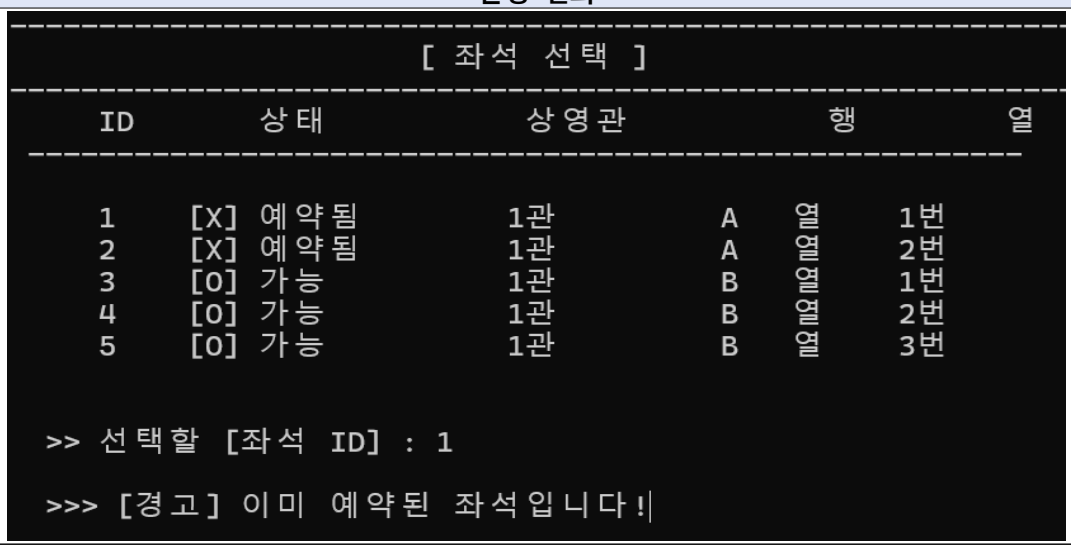
이미 예약된 좌석 방어 로직 코드

```
// 1. 좌석 상태 표시 (예약 여부 시각화)
EXEC SQL SELECT count(*) INTO :v_is_booked
         FROM Bookings
         WHERE schedule_id = :v_selected_sid AND seat_id = :v_seatid;

if (v_is_booked > 0) printf("[X] 예약됨");
else printf("[O] 가능");

// ... (사용자 입력 후) ...

// 2. 선택한 좌석 재검증 (Double Check)
// 사용자가 실수로 [X] 좌석을 입력했을 때 DB에서 다시 한 번 확인하여 차단
EXEC SQL SELECT count(*) INTO :v_is_booked
         FROM Bookings
         WHERE schedule_id = :v_selected_sid AND seat_id = :input_seat_temp;

if (v_is_booked > 0) {
    gotoxy(2, y+4);
    printf(">>> [경고] 이미 예약된 좌석입니다!");
    getch();
    continue; // 다시 입력받기 위해 루프 처음으로 이동
}
```

설명

- 서버 사이드 검증: 사용자가 화면상의 [X] 표시를 무시하고 강제로 해당 번호를 입력하더라도, 로직 내부에서 SELECT count(*)를 통해 한 번 더 검증(Double Check)하여 중복 예약을 원천 봉쇄한다.

실행 결과



4.5.6 예매 변경

예매 변경 코드

```
void fn_change_booking() {

    // [화면 1] 본인 확인 및 변경 권한 검증 (Security)
    // - 입력한 예매번호(target_bid)가 실제 로그인한 유저(target_uid)의 것인지 확인
    EXEC SQL SELECT count(*) INTO :check_exists
             FROM Bookings
             WHERE booking_id = :target_bid AND user_id = :target_uid;

    if (check_exists == 0) {
        printf(">>> [오류] 본인의 예매 내역이 아닙니다.₩n"); // 권한 없음 차단
        return;
    }

    // [화면 2] 신규 영화 및 일정 선택 (Reuse)
    // - 기존 예매 로직(select_schedule_logic)을 재사용하여 영화/일정/좌석 선택
    // - 성공 시 새로운 일정ID(new_sid)와 좌석ID(new_seatid)를 받아옴
    if (select_schedule_logic(&new_sid, &new_seatid, &dummy) == 0) return;

    // [화면 3] 최종 변경 및 DB 반영 (Atomic Update)
    // - 기존 예매 건의 일정과 좌석 정보를 동시에 수정
    EXEC SQL UPDATE Bookings
             SET schedule_id = :new_sid, seat_id = :new_seatid
             WHERE booking_id = :target_bid;

    // 트랜잭션 확정 (Commit)
    if (sqlca.sqlcode == 0) {
        EXEC SQL COMMIT WORK;
        printf(">>> 예매 변경 완료 (티켓 재발급)₩n");
    } else {
        EXEC SQL ROLLBACK WORK; // 실패 시 원상복구
        printf(">>> 변경 실패 (DB 오류)₩n");
    }
}
```

설명

- 소유권 검증: 예매 번호와 회원 ID를 대조하여 본인의 예매 내역만 수정할 수 있도록 보안 로직을 적용하였다.
- 모듈 재사용: 신규 일정 선택 시 기존 예매 함수를 재사용하여 코드 중복을 최소화하였다.
- 트랜잭션 처리: 일정과 좌석 정보를 하나의 UPDATE 문으로 처리하고, 오류 발생 시 ROLLBACK 하여 데이터 불일치를 방지한다.

실행 결과 (예매 변경)

```
------------------------------------------------------------------
                  [ 예매 변경 (영화/일정) ]
------------------------------------------------------------------

            * 본인 확인을 위해 정보를 입력해주세요.


            1. 회원 ID : 1010

            2. 예매 ID : 46



  >> 확인 완료! 엔터를 누르면 새 일정을 선택합니다.
   ------------------------------------------------------
     [안내] 정보가 일치하면 새로운 영화/좌석 선택으로 이동합니다.
------------------------------------------------------------------
--------------- [ 회원님의 예매 목록 (ID: 1010) ] ----------------
예매ID   영화제목                        좌석
  46     범죄도시4                       B-1
------------------------------------------------------------------
```

실행 결과 (변경할 영화 목록 선택)

```
------------------------------------------------------------------
                  [ 현재 상영중인 영화 목록 ]
------------------------------------------------------------------
  ID      영화제목                       등급      상영시간 (분)
 ---------------------------------------------------------------

    1     범죄도시4                      15세         109분
    2     인사이드 아웃 2                전체          96분
    3     아바타 3                       12세         180분
    4     파묘                           15세         134분
    5     위니 더 푸: 피와 꿀            18세          90분


 >> 선택할 [영화 ID] : 1
```

실행 결과 (상영 일정 선택)

```
--------------------------------------------------------------------------
                    [ 상영 일정 선택 ]
--------------------------------------------------------------------------
ID      영화제목                  상영관     시작시간          가격
-------------------------------------------------------------------

  4     위니 더 푸: 피와 꿀       2관       2025-12-25 22:00    10000원
  8     위니 더 푸: 피와 꿀       1관       2025-12-25 23:00    10000원

>> 선택할 [일정 ID] :
```

실행 결과 (예매 변경 완료)

```
-----------------------------------------------------
                  [ 좌석 선택 ]
-----------------------------------------------------
 ID       상태              상영관              행          열
-----------------------------------------------------

 1     [O] 가능            1관            A    열     1번
 2     [O] 가능            1관            A    열     2번
 3     [O] 가능            1관            B    열     1번
 4     [O] 가능            1관            B    열     2번
 5     [O] 가능            1관            B    열     3번

> 선택할 [좌석 ID] : 5

>>> 예매 변경 완료! 티켓이 재발급되었습니다. <<<
```

4.5.7 예매 내역 조회

나의 예매 내역 조회 코드

```
EXEC SQL DECLARE c_list CURSOR FOR
    SELECT b.booking_id, m.title, to_char(sch.start_time, 'MM-DD HH24:MI'),
           s.row_code || '-' || s.col_code, b.status
    FROM Bookings b, Schedules sch, Movies m, Seats s
    WHERE b.schedule_id = sch.schedule_id
      AND sch.movie_id = m.movie_id
      AND b.seat_id = s.seat_id
      AND b.user_id = :search_uid
    ORDER BY b.booking_id DESC;
```

설명

- 다중 조인(Join): Bookings(예매), Schedules(일정), Movies(영화 정보), Seats(좌석 정보) 등 4개의 테이블을 조인하여 예매 ID만으로는 알 수 없는 영화 제목, 시간, 좌석 위치(행-열)를 한 번에 조회한다.
- 데이터 포맷팅: Oracle의 to_char 함수를 사용하여 날짜 데이터를 가독성 높은 포맷(MM-DD HH:MI)으로 변환하여 출력한다.

실행 결과

```
--------------------------------------------------------------------------------
                    [ 나의 예매 내역 ]
--------------------------------------------------------------------------------
  예매ID      영화제목                          일정시간          좌석          상태
--------------------------------------------------------------------------------

   32       범죄도시4                        12-25 10:00          A-1        결제완료




--------------------------------------------------------------------------------
```

4.5.8 예매 취소

예매 취소 코드

```
// 1. 본인 예매 내역 확인 (권한 검증)
EXEC SQL SELECT count(*) INTO :check_exists
        FROM Bookings
        WHERE booking_id = :target_bid AND user_id = :target_uid;

if (check_exists == 0) {
    printf(">>> [오류] 예매번호 [%d]는 회원 [%d]님의 예약이 아닙니다.", target_bid, target_uid);
    return;
}
// 2. 삭제 수행
EXEC SQL DELETE FROM Bookings WHERE booking_id = :target_bid;
```

설명

- 무결성 검증: 삭제 대상인 booking_id가 현재 로그인한 user_id의 소유인지 검증하여 타인의 티켓 오삭제를 방지한다.
- 데이터 삭제: 검증 완료 후 DELETE 문을 통해 데이터를 영구적으로 삭제하며, 이에 따라 해당 좌석은 즉시 예약 가능 상태로 전환된다.

실행 결과

```
--------------------------------------------------------------------------------
                        [ 예매 변경 (영화/일정) ]
--------------------------------------------------------------------------------

                * 본인 확인을 위해 정보를 입력해주세요.


                1. 회원 ID : 20212979

                2. 예매 ID : 32



        정말 취소하시겠습니까? (y/n): Y
      ------------------------------------------------------------
        [안내] 정보가 일치하면 새로운 영화/좌석 선택으로 이동합니다.
--------------------------------------------------------------------------------
---------------- [ 회원님의 예매 목록 (ID: 20212979) ] ----------------
 예매ID    영화제목                          좌석
   32      범죄도시4                         A-1
----------------------------------------------------------------
```

4.5.9 예매 내역 실시간 조회

예매 내역 실시간 조회 코드

```
int show_booking_list(int uid, int mode) {
    // 1. 다중 테이블 조인 (Join): 예매ID뿐만 아니라 영화제목, 시간, 좌석까지 한 번에 조회
    EXEC SQL DECLARE c_list CURSOR FOR
        SELECT b.booking_id, m.title, to_char(sch.start_time, 'MM-DD HH24:MI'),
               s.row_code || '-' || s.col_code, b.status
        FROM Bookings b, Schedules sch, Movies m, Seats s
        WHERE b.schedule_id = sch.schedule_id
          AND sch.movie_id = m.movie_id
          AND b.seat_id = s.seat_id
          AND b.user_id = :search_uid
        ORDER BY b.booking_id DESC;

    EXEC SQL OPEN c_list;

    // 2. 데이터 순차 조회 (Fetch Loop)
    while(1) {
        EXEC SQL FETCH c_list INTO :v_bid, :v_title, :v_time, :v_seat, :v_status;

        if(sqlca.sqlcode == 1403) break; // 데이터가 더 없으면 종료

        // 3. UI 출력 위치 제어 (mode 1: 화면 하단에 리스트 출력)
        if (mode == 1) {
            gotoxy(2, y++); // 기존 화면을 유지한 채 아래쪽에 출력
            printf("%d    %s    %s    %s", v_bid, v_title, v_seat, v_status);
        }
    }
    EXEC SQL CLOSE c_list;
}
```

설명

- 다중 조인(Multi-table Join): Bookings 테이블에는 ID만 저장되어 있으므로, 제목, 시간, 좌석 테이블을 모두 조인하여 사용자에게 친절한 정보를 제공한다.
- 조건부 렌더링(Conditional Rendering): mode 파라미터를 사용하여, 단순 조회 메뉴에서는 전체 화면에 출력하고, 변경/취소 메뉴에서는 화면 하단(mode=1) 에만 출력하여 입력 폼을 가리지 않도록 구현했다.

실행 결과 (회원 ID 입력 전)

```
------------------------------------------------------------
              [ 예매 변경 (영화/일정) ]
------------------------------------------------------------

        * 본인 확인을 위해 정보를 입력해주세요.

        1. 회원 ID :

        2. 예매 ID :




------------------------------------------------------
[안내] 정보가 일치하면 새로운 영화/좌석 선택으로 이동합니다.
------------------------------------------------------------
```

실행 결과 (회원 ID 입력 후)

```
------------------------------------------------------------
                    [ 예매 변경 (영화/일정) ]
------------------------------------------------------------

              * 본인 확인을 위해 정보를 입력해주세요.

              1. 회원 ID : 1010

              2. 예매 ID :




    ------------------------------------------------------
       [안내] 정보가 일치하면 새로운 영화/좌석 선택으로 이동합니다.
------------------------------------------------------------
---------------- [ 회원님의 예매 목록 (ID: 1010) ] ----------------
예매ID    영화제목                          좌석
  46      아바타 3                          B-3
------------------------------------------------------------
```

4.5.10 UI 커서 제어

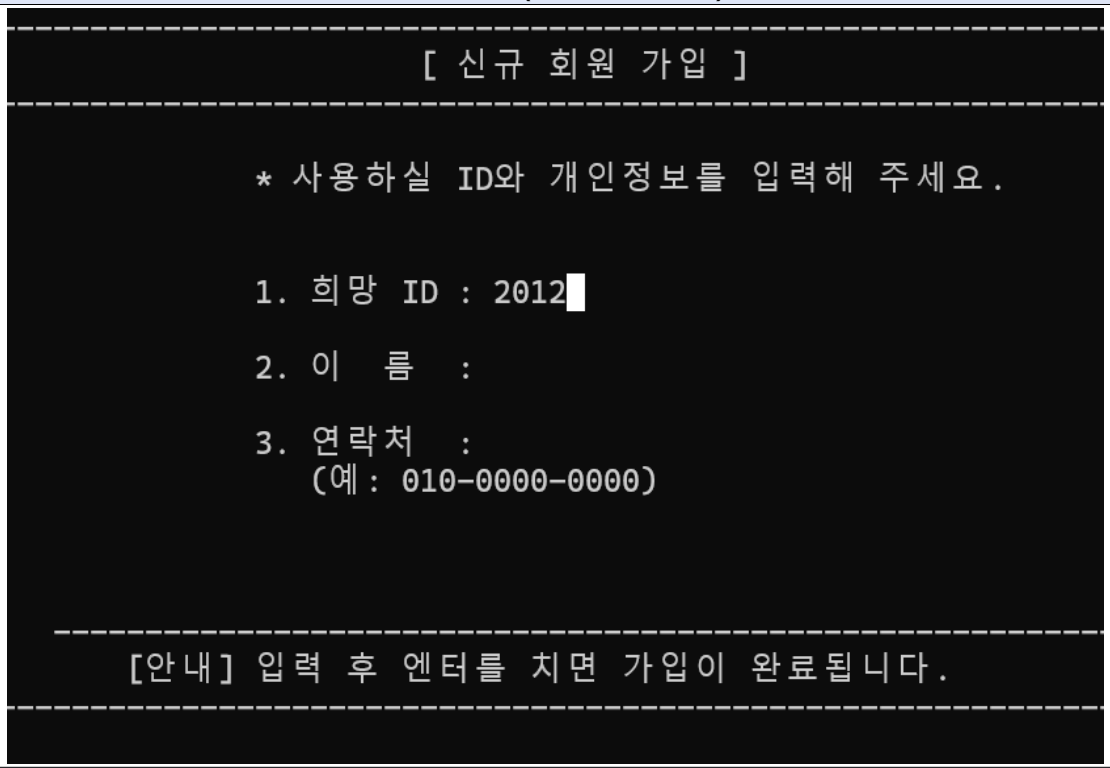
UI 커서 제어 및 입력 흐름 코드

```
void fn_signup() {
    int x = 30, y = 7; // 초기 입력 시작 좌표 설정

    // 1. 첫 번째 필드 (ID) 입력 대기
    gotoxy(x, y);
    if(fgets(temp, sizeof(temp), stdin) == NULL) return;  // Enter 입력 시까지 대기

    // ... (ID 중복 체크 로직 수행) ...

    // 2. 다음 필드로 커서 이동 (좌표 계산)
    y += 2; // Y 좌표를 2칸 아래로 이동 (화면상 다음 항목 위치)
    gotoxy(x, y); // 커서를 해당 위치로 강제 이동

    // 3. 두 번째 필드 (이름) 입력 대기
    fgets(v_name, sizeof(v_name), stdin);

    // 4. 세 번째 필드 (연락처) 입력 대기
    y += 2;
    gotoxy(x, y);
    fgets(v_contact, sizeof(v_contact), stdin);
}

// Windows API를 이용한 커서 이동 헬퍼 함수
void gotoxy(int x, int y) {
    COORD Cur = {x, y};
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), Cur);
}
```

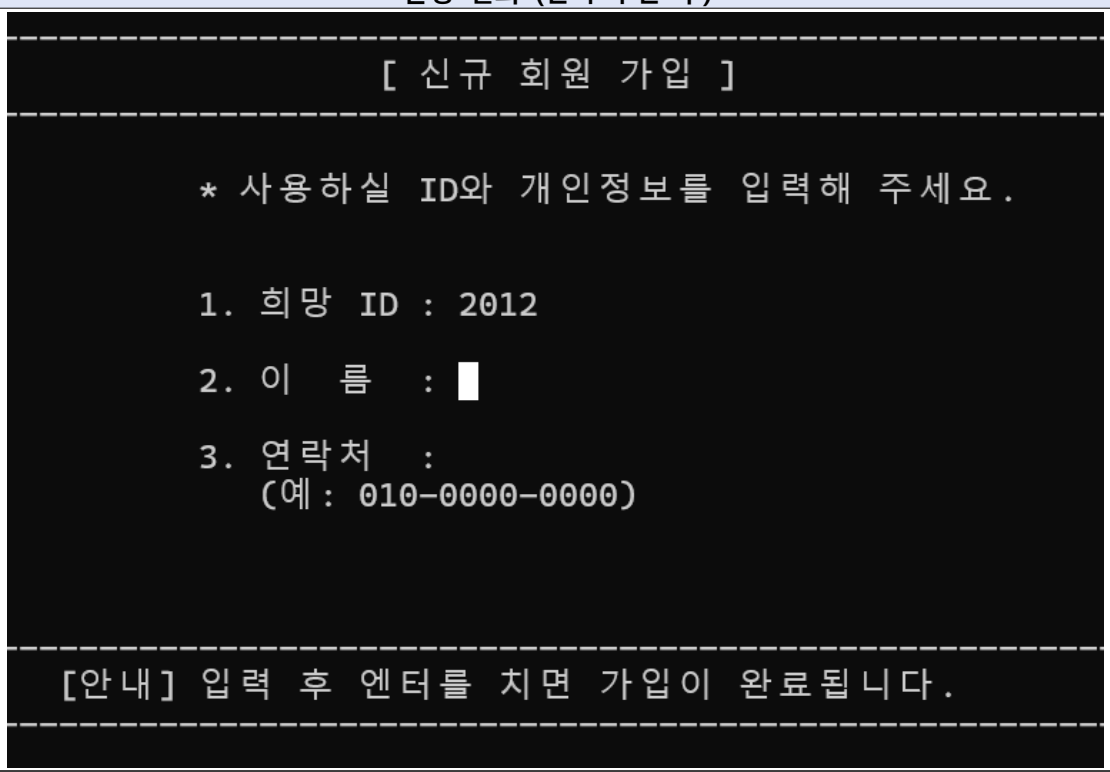
설명

- 좌표 제어: gotoxy(x, y) 함수를 통해 콘솔 커서를 특정 좌표로 이동시킨다.
- 순차적 흐름: 입력이 완료되면 y += 2 연산을 수행해 다음 필드로 커서를 자동 이동시켜, 사용자가 체계적으로 정보를 입력할 수 있는 직관적인 흐름을 구현하였다.

실행 결과 (엔터 누르기 전)



실행 결과 (엔터 누른 후)



4.5.11 종료

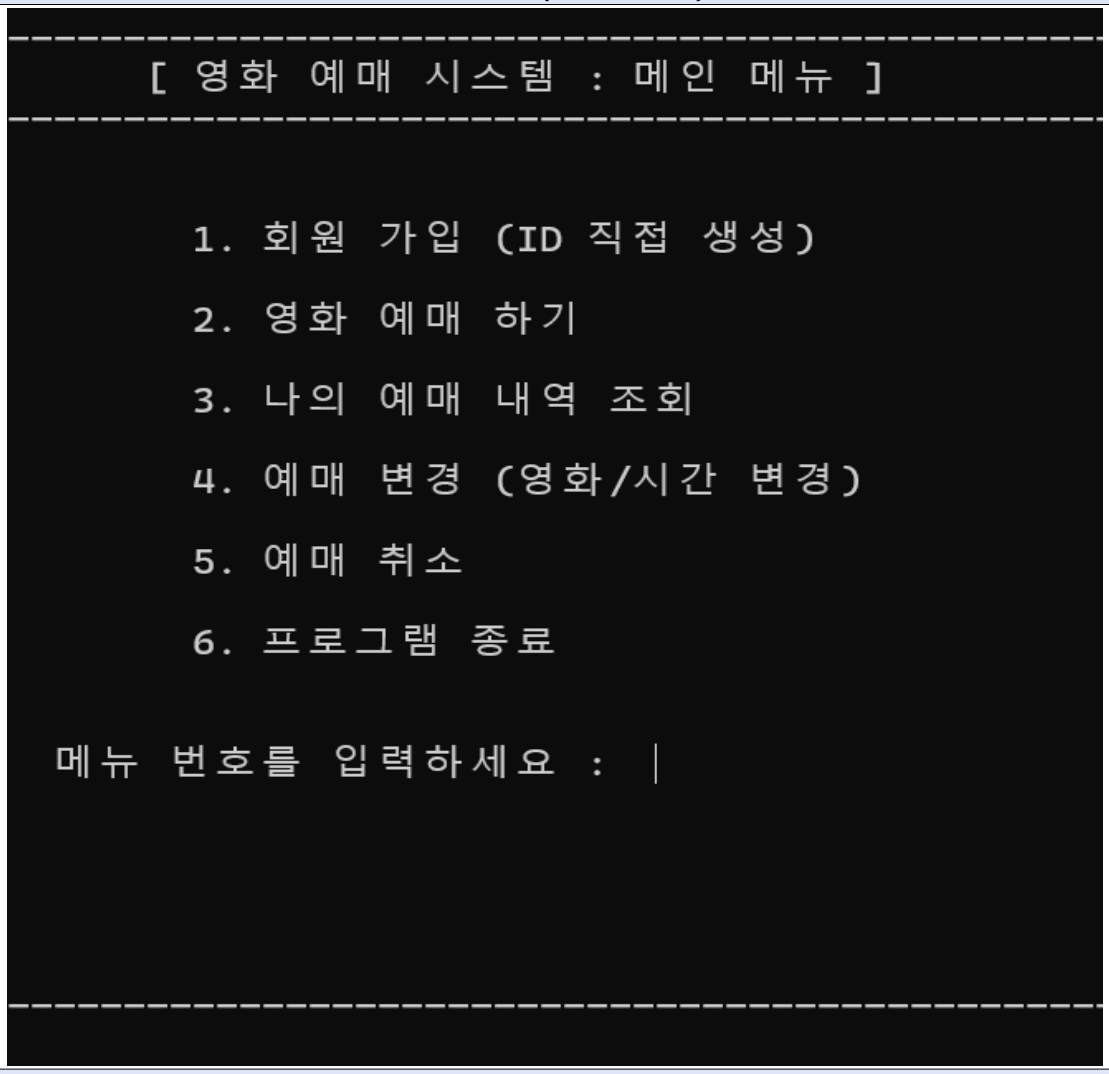
종료 및 리소스 해제 코드

```
void main() {
    // 1. 메인 루프: 사용자가 '6'을 입력하기 전까지 계속 반복
    while (c != '6') {
        print_screen("scr_main.txt"); // 메뉴 화면 출력

        // ... (사용자 입력 받기) ...

        switch(c) {
            case '1': fn_signup(); break;
            // ... (다른 기능들) ...
            case '6': break; // switch문을 빠져나감 -> while 조건(c!=6) 체크 후 루프 탈출
        }
    }

    // 2. 종료 메시지 출력
    clrscr();
    printf("₩n 시스템을 종료합니다.₩n");

    // 3. [핵심] DB 세션 해제 (Resource Release)
    // 현재까지의 작업을 최종 확정(COMMIT)하고, 서버와의 연결을 안전하게 끊음(RELEASE)
    EXEC SQL COMMIT WORK RELEASE;
}
```

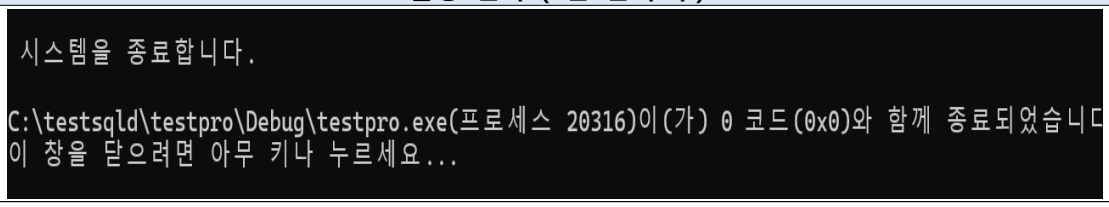
설명

- 루프 탈출: 사용자가 6번을 선택하면 while 루프를 종료한다.
- 리소스 해제 (Release Connection): 프로그램 종료 직전 EXEC SQL COMMIT WORK RELEASE를 호출하여 트랜잭션을 확정하고 서버 연결을 명시적으로 끊는다. 이는 유령 세션(Ghost Session) 발생을 방지하는 필수적인 안정화 코드이다.

실행 결과 (6번 선택 전)



실행 결과 (6번 선택 후)



5. 프로젝트 결과

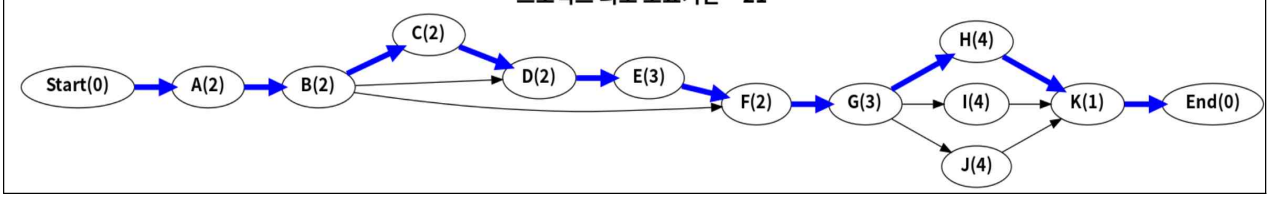
5.1 프로젝트 완성도

기능	완성도	평가
회원가입 및 데이터 검증	100 / 100	사용자 입력 정보를 정확하게 처리하며, ID 중복 사전 검사 및 전화번호 형식 제약검증을 통해 데이터 무결성을 확보하였다.
조인 기반 정보 제공	100 / 100	정규화된 여러 테이블을 효율적으로 조인하여, 사용자가 예매 상세 정보를 한눈에 파악할 수 있도록 직관적인 뷰를 제공하였다.
사용자 기능	100 / 100	사용자의 필요에 따라 데이터의 생성, 조회, 변경, 삭제가 가능하도록 전 기능을 구현하여, 유연하고 자유로운 서비스 이용 환경을 구축하였다.

5.2 일정 계획

Activity No.	소작업	소요기간	선행작업
Start	시작	0	-
A	주제 선정	2	Start
B	요구 사항 분석 및 정리	2	A
C	데이터 베이스 설계	2	B
D	E-R 다이어그램 작성	2	B,C
E	테이블 명세서 작성	3	D
F	회원가입 기능 구현	2	B,E
G	영화 예매 기능 구현	3	F
H	예매 내역 조회 기능 구현	4	G
I	예매 내역 변경 기능 구현	4	G
J	예매 취소 기능 구현	4	G
K	테스팅	1	H,I,J
End	종료	0	K

프로젝트 최소 소요기간 = 21



5.3 역할 수행

	이름	역할
팀장	이지민	- 프로젝트 설계 (물리적 설계) - 예매 변경 및 취소 트랜잭션 처리 로직 구현
팀원	이규찬	- 기능 요구사항 명세서(SRS) 작성 및 분석 - 회원가입 프로세스, 데이터 유효성 검사, 프로그램 종료 및 리소스 해제
	임민욱	- 데이터 무결성 유지를 위한 제약 조건 설계 - 메인 인터페이스 구성 및 화면 전환 로직 구현
	임진호	- E-R 다이어그램 작성 및 데이터 모델링 - 조인을 이용한 실시간 내역 조회 및 예매 확정 로직 구현

5.4 위험 처리

문제점/ 위험요소	설명	해결방안
데이터 무결성 위배 (중복 예매)	동일한 상영 스케줄의 같은 좌석을 다수의 사용자가 동시에 예매를 시도할 경우, 중복 예약이 발생할 위험이 있음.	• 데이터 무결성 강화: 테이블 설계 시 PK 및 Unique Key 제약 조건을 설정하고, 예매 확정 전 SELECT count(*)를 통한 이중 검증 수행.
Pro*C 컴파일 및 링킹 오류	일반 C 코드와 달리 Oracle 내장 SQL 문법 오류는 디버깅이 까다롭고, Pre-compiler를 거치는 과정에서 시간이 많이 소요됨.	• 개발 및 디버깅 효율화: SQL Developer를 활용한 쿼리 문법 사전 검증 및 sqlca.sqlcode를 이용한 실시간 에러 코드 출력으로 원인 파악 시간 단축.
콘솔(CLI) 환경의 UI 한계	텍스트 기반(CLI) 환경 특성상 영화 좌석 배치도나 복잡한 예매 내역을 사용자에게 직관적으로 시각화하여 전달하기 어려움.	• gotoxy() 함수를 활용한 화면 좌표 제어로 좌석 배치도를 구현하고, 예약 상태를 [O], [X]로 기호화하여 직관성 확보.
사용자 입력 예외	메뉴 선택이나 ID 입력 시, 자료형 불일치(문자 입력 등)나 범위 초과 값을 입력할 경우 프로그램이 비정상 종료될 위험이 있음.	• 입력 버퍼 초기화 로직을 적용하고, 조건문(if)을 활용한 유효성 검사를 통해 잘못된 입력 시 재입력을 유도하는 방어 코드 작성.

6. 소감

팀원	소감
임진호	이번 영화 예매 시스템 프로젝트를 진행하면서 단순히 기능 구현을 넘어서, 실제 서비스 구조가 어떻게 동작하는지 깊이 이해할 수 있는 경험을 얻었습니다. 특히 Pro*C를 활용해 SQL과 C 로직을 연동하면서, 데이터베이스 설계가 프로그램 안정성과 사용성에 얼마나 큰 영향을 미치는지 실감하게 되었습니다. 좌석 중복 예약 방지나 전화번호 형식 검증처럼 사소해 보이는 기능도 실제로는 정확한 트랜잭션 처리와 제약 조건 설계가 뒷받침되어야만 제대로 작동한다는 점을 배우게 되었다는 점이 인상 깊었습니다. 또한 사용자 흐름에 맞는 화면 구성과 오류 처리 과정을 직접 구현하면서 프로그램 완성도의 중요성을 느꼈고, 문제가 발생할 때마다 디버깅하며 논리적인 사고력을 키울 수 있었습니다. 이번 프로젝트는 기술적인 성장뿐 아니라 구조적으로 사고하는 능력을 키워준 의미 있는 경험이었습니다.
이규찬	이번 프로젝트는 기존의 단순한 SQL문 실습 단계를 넘어서, 처음으로 저장 프로시저를 활용해 데이터베이스 내부에서 직접 비즈니스 로직을 처리하는 방식을 깊이 있게 경험하는 시간이었습니다. 도입 초기에는 기존 SQL과는 다른 문법과 흐름이 낯설게 느껴졌지만, 반복되는 쿼리 작업을 하나의 프로시저로 모듈화하고 복잡한 조건 처리를 DB 레벨에서 수행해 보면서, 전체적인 코드 로직이 훨씬 체계적으로 정리되고 시스템 구조 또한 한층 더 안정적으로 변화한다는 것을 확실히 느낄 수 있었습니다. 결과적으로 이번 프로젝트를 통해 프로시저가 팀원 간의 협업 효율성을 높이고 유지보수성을 확보하는 데 필수적인 도구임을 깨닫게 되었으며, 엔지니어로서 성장하는 소중한 계기가 되었습니다.
이지민	팀장으로서 프로젝트의 기반이 되는 DB 물리적 설계를 주도하고, 시스템의 핵심 기능인 예매 변경 및 취소 트랜잭션을 구현했습니다. 초기 단계부터 제약 조건을 엄격히 설정하여 데이터 불일치 문제를 미연에 방지했으며, 상황에 따라 COMMIT과 ROLLBACK 시점을 정교하게 제어하여 트랜잭션의 원자성을 보장하는 데 주력했습니다. 또한 팀원들이 개발한 모듈을 하나의 시스템으로 통합하는 과정에서 원활한 소통과 조율의 중요성을 깊이 체감했으며, 이러한 경험은 향후 프로젝트를 이끄는 데 큰 자산이 될 것입니다.
임민욱	이번 프로젝트에서 Pro*C를 활용해 영화 예매 시스템을 구현하며, 단순 테이블 구성을 넘어 실제 서비스 흐름을 DB 내에서 효율적으로 처리하는 방법을 깊이 있게 경험했습니다. 예매 프로시저를 통해 좌석 중복 방지 등 다양한 조건을 통합 처리하며 데이터 무결성 유지의 중요성을 절실히 깨달았습니다. 구현 중 발생한 오류들을 분석하고 로직을 개선하는 과정은 DB 설계 및 논리적 사고력을 크게 향상시키는 계기가 되었습니다. 또한 팀원들과 의견을 조율하며 구조를 다듬었던 협업의 경험은 향후 더 복잡한 시스템 개발에 있어 든든한 기반이 될 것입니다.