

ICE3020 알고리즘설계

<제목>

Term Project

보고서 작성 서약서

1. 나는 타학생의 보고서를 베끼거나 여러 보고서의 내용을 짜집기하지 않겠습니다.

2. 나는 보고서의 주요 내용을 인터넷사이트 등을 통해 얻지 않겠습니다.

3. 나는 보고서의 내용을 조작하지 않겠습니다.

4. 나는 보고서 작성에 참고한 문헌의 출처를 밝히겠습니다.

5. 나는 나의 보고서를 제출 전에 타학생에게 보여주지 않겠습니다.

나는 보고서 작성시 윤리에 어긋난 행동을 하지 않고 정보통신공학인으로서 나의 명예를 지킬 것을 맹세합니다.

2024년 12 월 16일

학부 정보통신공학

학년 3

성명 이유진

학번 12221510



1. 개요

* 해당 실습 또는 과제에서 구현해야 하는 목표 또는 개괄적인 설명 작성

1. 상세 설계 내용

* 해당 실습 또는 과제에서 구현한 내용에 대하여 중요한 부분의 코드 캡처 및 상세한 설명 작성
* 알고리즘 동작 과정 및 어떤 생각을 가지고 어떤 방식으로 구현했는지에 대하여

<ERD >

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

book은 reservation과 연결되어있는데, 하나의 예약은 하나의 책만 가능하고 같은 책이 서로 다른 customer들이 예약할 수 있으므로 n:1로 연결된다. 예약한 해당 책인 book\_ISBN을 FK로 받는다. 모든 책이 예약이 되어있는 상태가 아니므로 partial participation이다. 각각의 table이 독립적으로 존재할수있으므로 non identifying relationship으로 연결하였다. ISBN, title은 null일수없고 나머지는 null이 가능하도록 설정하였다. ISBN은 INT타입 범위를 넘어가므로 varchar(45)로 설정하였다.

reservation에 reservation\_date는 예약을 설정한 시점, pickup\_date는 pickup할 날짜, pickup\_time은 pickup할 시간이다. pickup\_date, pickup\_time은 예약을 확정안했을때에는 null이기 때문에 null이 가능하도록 설정하였다. ID는 book의 ISBN값이므로 VARCHAR 타입을 사용하였다.

shopping\_basket은 구매전에는 order\_date값이 없으므로 null이가능하도록 설정하였다.

contains는 관계형 table로 book과 shopping\_basket을 1:1으로 받는다. 한 장바구니안에 한 종류의 책이 있을 수 있고 쇼핑카트에 책이 없을 수 도 있고 모든 책이 쇼핑카트에 있는 것이 아니기 때문에 partial participation이다. number는 0일수있지만 null일수는 없으므로 null이 불가능하도록 설정하였다.

basket\_of은 관계형 table로 shopping\_basket와 customer를 1:n로 받는다. 한 고객은 책 종류마다 장바구니를 가질수있고 모든 고객이 장바구니를 가져야하므로 total participation이다.

customer 와 reservation은 1:1이고, 한 고객당 하나의 예약만 할수있고 각각의 table이 독립적으로 존재할수있으므로 non identifying relationship으로 연결하였다. reservation에서 어떤 customer가 예약한건지 확인하기위해 추가적으로 customer의 Email과 동일한 Email을 FK로 받아 예약항목에 저장한다. 모든 customer가 예약을 한 생태가 아니므로 partial participation이다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

Inventory는 관계형 table로 book, warehouse을 받는다. 여러 개의 책이 창고에 들어가있고, 한 책은 여러 창고에 존재할수있으므로 N:M으로 연결된다. 창고에 책이 없을수있지만 모든 책은 창고에 존재해야하므로 partial participation으로 연결된다. 책의 수가 null이 될수없으므로 null이 불가능하도록 설정하였다. 창고 번호는 없을 수 있지만, 주소는 없을수없으므로 위와 같이 설정하였다. inventory는 book\_ISBN, warehouse\_code가 n:m이므로 pk를 두개갖도록 설정하였다.

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

award의 연도가 불분명할 수 도있고 애매할 수 있어서 null이 가능하도록 설정하였다.

author의 url은 존재하지않을 수 도 있고, address는 공개하지않을 수 있어 null이 가능하도록 설정하였다.

awarded\_to는 관계형 table로 book과 award를 받는다. 한 책에서 여러 개의 상을 받을 수 있고, 한 상에 여러권의 책이 상을 받을 수 있으므로 n:m으로 연결하였다. 수상을 하지않은 책도 있으므로 partial participation으로 연결하였다.

received\_by는 관계형 table로 award와 author를 받는다. 한 작가가 여러 상을 수상받을 수 있고, 한 상에는 여러명의 작가가 상을 받을수있으므로 n:m으로 받는다. 수상을 받지않은 작가도 있으므로 partial participation으로 연결하였다.

written\_by 는 관계형 table로 author와 book을 받는다. 한 책에는 공동저자가 있을수있고, 한 작가가 여러 책을 썼을 수 있으므로 n:m으로 연결하였다. 모든 책에는 작가가있으므로 total participation이다.

텍스트, 폰트, 번호, 라인이(가) 표시된 사진

자동 생성된 설명

추가적으로 user table을 생성하였다. user에는 로그인할 때 쓰이는 id, password column이 있고, role에는 customer와 super(관리자)가 있다. 이중에 role=customer인 id는 customer의 name을 FK로 받아 1:1로 관계를 형성하였다. 관리자는 이 관계에 포함되지않으므로 partial participation이다. 로그인시 password와 role이 명확해야하므로 null이 불가능하도록 설정하였다.

<소스코드>

웹코드에서 트랜잭션은

트랜잭션시작: await transactionSql.startTransaction();

성공시: await transactionSql.commitTransaction();

오류시: await transactionSql.rollbackTransaction();

실행된다. 각 함수는 sql.js파일에서 설명할것이다.

<login.js>

-get요청

 사용자가 로그인 페이지에 접근했을 때 실행.

 만약 사용자가 이미 로그인된 상태(req.session.user && req.session.user.checkLogin)라면, 역할(role)에 따라 자동으로 리다이렉트:

* customer: /home으로 이동.
* super (관리자): /administer로 이동.

 로그인이 안 된 상태면 로그인 페이지를 렌더링(res.render('login')).

-post요청

 로그인 폼에서 제출된 사용자 id와 password를 인증.

 데이터베이스에서 가져온 사용자 목록(selectSql.getUsers())과 대조.

 일치하는 사용자 정보가 있으면:

* 세션(req.session.user)에 사용자 정보 저장.
* 쿠키(user와 role)를 설정하여 1시간 동안 유효.
* 사용자 역할에 따라 페이지 리다이렉트:
  + customer: /home.
  + super: /administer.

 인증 실패 시:

* 브라우저에 경고 메시지와 함께 로그인 페이지로 돌아가도록 처리.

<logout.js>

 **세션 삭제**:

* req.session.destroy((err) => { ... }): 현재 세션을 삭제합니다.
* 세션 삭제 실패 시, 오류 메시지를 기록하고 500 상태 코드와 함께 오류 응답을 보냅니다.

 **쿠키 삭제**:

* res.clearCookie('user');
* res.clearCookie('role');
* user와 role 쿠키를 삭제합니다.

 **로그인 페이지로 리다이렉트**:

* res.redirect('/');: 로그아웃 후, 로그인 페이지로 리다이렉트합니다.

<home.js>

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

<book\_search.js>

/페이지 post

startTransaction으로 트랜잭션 시작

 searchType 값에 따라 검색 쿼리를 실행:

* **book**: 책 제목으로 검색.
* **author**: 작가 이름으로 검색.
* **award**: 수상 내역으로 검색.

 selectSql 모듈의 검색 함수(searchByBookTitle, searchByAuthorName, searchByAwardName)를 호출하여 데이터베이스에서 결과를 가져온다.

검색 작업이 성공적으로 완료되면 트랜잭션을 커밋하여 데이터베이스에 변경 사항을 확정.

검색 중 오류가 발생하면 트랜잭션을 롤백하여 작업을 취소

검색 결과를 **book\_search\_results** 템플릿에 렌더링:

* title: 검색 결과 페이지의 제목.
* books: 검색 결과 목록.
* searchType: 사용자가 선택한 검색 유형.
* searchValue: 사용자가 입력한 검색어.

/reserve페이지

 데이터베이스에서 userId를 기반으로 고객 이메일 정보를 가져옴.

 고객 정보가 없으면 잘못된 세션으로 간주하고 오류 반환.

스크린샷, 텍스트, 폰트이(가) 표시된 사진

자동 생성된 설명 사용자의 이메일로 이미 예약된 책이 있는지 확인.

 중복 예약이 존재하면 경고 메시지를 표시하고 /book\_search로 리다이렉트.

스크린샷이(가) 표시된 사진

자동 생성된 설명

 ISBN을 사용해 해당 책의 재고 상태를 조회.

 재고가 없거나 책이 존재하지 않으면 예약을 차단.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

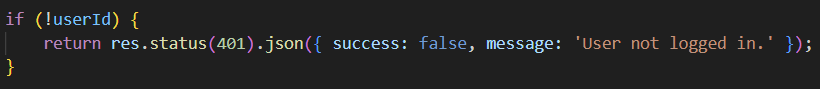
책을 예약하고 예약 정보를 데이터베이스에 저장:

* 예약된 책의 ISBN.
* 고객 이메일.
* 픽업 시간 및 날짜 (초기값 null).

/shopping\_basket

try { await transactionSql.startTransaction();

데이터베이스 트랜잭션을 시작합니다. 이는 데이터 일관성을 보장하기 위해 사용됩니다.



사용자가 로그인되어 있지 않으면 401 상태 코드와 함께 오류 메시지를 반환합니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

 userId를 사용하여 고객의 이메일을 가져옵니다.

 고객이 존재하지 않으면 404 상태 코드와 함께 오류 메시지를 반환합니다

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

 ISBN을 사용하여 책의 재고를 확인합니다.

 재고가 없으면 해당 메시지를 반환합니다.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

 고객의 이메일과 ISBN을 사용하여 장바구니를 확인합니다.

 장바구니에 해당 책이 이미 있는 경우 수량을 증가시킵니다.

 장바구니에 책이 없으면 새 장바구니를 생성합니다.

 재고가 부족하거나 구매내역이 있는 경우 알림 메시지를 반환합니다.



성공시 commit

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

오류가 발생하면 트랜잭션을 롤백하고 오류 메시지를 반환합니다

<reservation.js>

/예약 페이지 GET (조회)

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

트랜잭션 시작 후

 req.session.user?.id를 통해 현재 로그인된 사용자의 ID를 가져옴.

 사용자가 로그인되지 않은 상태면 **로그아웃 처리**(/logout으로 리다이렉트)

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

 userId를 기반으로 데이터베이스에서 해당 고객 정보를 검색.

 고객 정보가 없으면 **404 에러**를 반환하며, 잘못된 사용자 세션으로 간주.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

 고객 이메일(customerEmail)을 사용해 데이터베이스에서 예약 정보를 조회.

 예약 정보는 reservations 배열로 반환됩니다.

이후 reservation 템플릿에 다음 데이터를 전달:

* **title**: 페이지 제목 (Manage Reservations).
* **reservations**: 조회된 예약 정보.

 클라이언트는 예약 내역을 확인할 수 있는 페이지를 보게 됩니다.

/update

스크린샷, 텍스트, 폰트이(가) 표시된 사진

자동 생성된 설명

클라이언트로부터 reservationId, reservationDate, pickupTime 값을 받아옵니다.

 주어진 날짜와 시간에 다른 예약이 있는지 확인합니다.

 시간 충돌이 발생하면 경고 메시지를 표시하고, 예약 페이지로 리다이렉트합니다.

스크린샷, 텍스트, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

예약 정보를 업데이트합니다. 이 부분에서는 데이터베이스에서 해당 예약 ID에 대한 날짜와 시간을 수정합니다.

 예약 업데이트가 성공하면 트랜잭션을 커밋하고 성공 메시지를 출력합니다.

 성공 메시지와 함께 예약 페이지로 리다이렉트합니다.

 오류가 발생하면 트랜잭션을 롤백하고 오류 메시지를 출력합니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

 클라이언트로부터 reservationId(=Email) 값을 받아옵니다.

 req.body를 출력하여 요청 본문 데이터를 로그에 기록합니다.

reservationId를 사용하여 예약을 삭제합니다. 이는 데이터베이스에서 해당 예약 ID에 대한 기록을 제거합니다.

<shopping\_basket.js>

/페이지 // 쇼핑 바구니 페이지 렌더링

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

 이 라우터는 / 경로로의 GET 요청을 처리합니다.

 세션에서 사용자 ID를 가져옵니다.

 사용자가 로그인하지 않은 경우 로그아웃 페이지로 리다이렉트합니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

데이터베이스 트랜잭션을 시작합니다. 이는 데이터 조작 중 오류가 발생할 경우 롤백할 수 있도록 합니다.

 userId를 사용하여 현재 로그인된 사용자의 이메일을 가져옵니다.

 고객이 존재하지 않으면 404 상태 코드와 함께 "Customer not found." 메시지를 반환합니다.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

고객의 이메일을 사용하여 쇼핑 바구니 항목들을 조회합니다.

 조회한 쇼핑 바구니 항목들을 사용하여 shopping\_basket 페이지를 렌더링합니다.

 렌더링에 성공하면 트랜잭션을 커밋합니다.

 오류가 발생하면 트랜잭션을 롤백하고, 오류 메시지를 기록하며 500 상태 코드와 함께 내부 서버 오류 메시지를 반환합니다.

/purchase

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

 사용자가 요청한 **책의 ISBN**, **창고 코드**, **구매 수량**을 요청 본문에서 가져옵니다.

 warehouseCode와 purchaseQuantity는 정수로 변환하며, 기본값은 0입니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

* bookISBN과 warehouseCode로 데이터베이스에서 재고 상태를 조회.
* 재고가 없거나 total\_inventory가 0 이하이면 구매를 차단하고 알림을 표시.
* Math.max를 사용해 결과가 0 이하로 내려가지 않도록 처리.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

 inventory.number에서 purchaseQuantity를 차감하여 재고를 계산.

쇼핑 바구니 테이블(contains)에서 해당 책의 수량을 0으로 설정.

사용자의 구매를 완료한 날짜를 업데이트하여 구매 기록을 남김.

<administer.js>

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

관리자가 관리할 수 있는 테이블의 이름 목록을 배열로 정의:

* **book**: 책 정보 관리.
* **author**: 작가 정보 관리.
* **award**: 수상 정보 관리.
* **warehouse**: 창고 정보 관리.
* **inventory**: 책 재고 관리.
* **contains**: 쇼핑 바구니 항목 관리.

administer 템플릿에 다음 데이터를 전달하여 렌더링:

* **title**: 페이지 제목 ("Administer Page").
* **tables**: 관리 가능한 테이블 이름 배열.

**/select**

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

 table 이름을 사용하여 해당 테이블의 데이터를 조회합니다.

 데이터베이스에서 해당 테이블의 모든 데이터를 가져옵니다.

 조회한 데이터를 사용하여 administer 페이지를 렌더링합니다.

 페이지의 제목을 Administer - ${table}로 설정하고, 조회한 데이터를 table과 data 변수로 전달합니다.

/insert텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

warehouse는 복합키이고 상식적으로 warehouse삽입하는데 inventory도 같이 추가 되어야하므로 insert를 같이 진행한다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

warehouse 이외의 테이블에 데이터를 삽입:

* insertData 함수 호출.
* table과 data를 전달하여 지정된 테이블에 데이터를 삽입.

/update

스크린샷, 텍스트이(가) 표시된 사진

자동 생성된 설명

 테이블의 기본 키 컬럼을 가져옵니다.

 기본 키 값이 요청 데이터에 없는 경우 경고 메시지를 표시하고, 관리 페이지로 리다이렉트합니다.

 기본 키 값을 primaryKeyValues 객체에 저장하고, 수정 데이터에서 기본 키 컬럼을 제거합니다.

스크린샷, 텍스트, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

 첫 번째 기본 키 컬럼의 값을 잠금 키로 사용합니다.

 잠금 키가 없는 경우 경고 메시지를 표시하고, 관리 페이지로 리다이렉트합니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

 데이터 수정 작업을 수행하기 전에 해당 데이터에 대해 잠금을 시도합니다.

 잠금을 획득하지 못한 경우 경고 메시지를 표시하고, 관리 페이지로 리다이렉트합니다.

 updateSql.updateData 함수를 호출하여 데이터를 업데이트합니다.

 업데이트 작업이 완료된 후 잠금을 해제합니다.

/delete

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

클라이언트로부터 table 값을 비롯한 나머지 데이터를 받아옵니다. 여기서 ...data는 삭제할 데이터의 기본 키 값을 포함합니다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

 테이블의 기본 키 컬럼을 가져옵니다.

 기본 키 값이 요청 데이터에 없는 경우 경고 메시지를 표시하고, 관리 페이지로 리다이렉트합니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

 기본 키 값을 primaryKeyValues 객체에 저장하고, 수정 데이터에서 기본 키 컬럼을 제거합니다.  
deleteSql.deleteData 함수를 호출하여 데이터를 삭제합니다. 여기서 table과 lockKey를 사용하여 해당 테이블의 데이터를 삭제합니다.

<sql.js>

getUsers

user 테이블에서 모든 사용자 정보를 조회

SQL 쿼리를 통해 모든 사용자를 조회하고, 결과를 반환합니다.

getReservations

특정 고객의 예약 정보를 조회

고객 이메일을 사용하여 해당 고객의 예약 정보를 조회하고, 결과를 반환

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

checkTimeConflict

예약 시간 충돌 여부를 확인

예약 시간 충돌 여부를 확인

updateReservation

예약 정보를 업데이트

예약 ID를 사용하여 예약의 픽업 날짜와 시간을 업데이트

deleteReservation

예약을 삭제

예약 ID를 사용하여 예약을 삭제

searchByBookTitle

책 제목을 기준으로 검색

책 제목을 사용하여 책 정보를 조회하고, 각 책의 총 재고 수량을 함께 반환

searchByAuthorName

저자 이름을 기준으로 검색

저자 이름을 사용하여 책 정보를 조회하고, 각 책의 총 재고 수량을 함께 반환

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

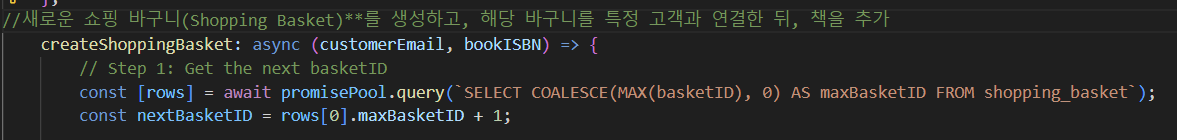
자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명



*  shopping\_basket 테이블에서 가장 큰 basketID를 조회 (MAX(basketID)).
* 만약 shopping\_basket에 데이터가 없다면 COALESCE를 사용해 기본값 0을 반환.
* 다음에 사용할 basketID는 maxBasketID + 1로 계산.
* 새로 생성할 쇼핑 바구니의 고유 ID(nextBasketID)가 결정됩니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

*  shopping\_basket 테이블에 새 바구니를 생성.
* basketID: 앞에서 계산한 nextBasketID.
* order\_date: 기본값 NULL로 설정.
* 새 쇼핑 바구니가 데이터베이스에 추가

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

*  basket\_of 테이블에 새 쇼핑 바구니와 고객 이메일 간의 관계를 추가.
* customer\_Email: 함수의 입력값으로 전달된 고객 이메일.
* shopping\_basket\_basketID: 앞서 생성된 nextBasketID.
* 새로 생성된 쇼핑 바구니가 특정 고객과 연결

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

*  contains 테이블에 책을 추가.
* book\_ISBN: 함수 입력값으로 전달된 책 ISBN.
* shopping\_basket\_basketID: 앞서 생성된 nextBasketID.
* number: 추가할 책의 수량(여기서는 기본값 1).
* 특정 책이 새로 생성된 쇼핑 바구니에 추가

getContainsEntry

contains 테이블에서 특정 book\_ISBN과 basketID를 가진 행을 조회

조회된 행 중 첫 번째 행을 반환

updateContainsQuantity

contains 테이블에서 특정 book\_ISBN과 basketID를 가진 행의 수량(number)을 업데이트

insertIntoContains

contains 테이블에 새로운 행을 삽입

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

 SUM(inventory.number)로 모든 창고의 재고 합계를 계산.

 COALESCE를 사용해 재고가 없는 경우에도 기본값 0 반환.

 GROUP\_CONCAT으로 책이 저장된 창고 코드를 쉼표로 연결.

 창고 코드 데이터를 배열로 변환하여 사용하기 쉽게 처리.

WHERE shopping\_basket.order\_date IS NULL 조건으로 주문이 완료되지 않은 쇼핑 바구니만 선택.

텍스트, 스크린샷, 소프트웨어, 운영 체제이(가) 표시된 사진

자동 생성된 설명

updateInventory

inventory 테이블에서 특정 book\_ISBN과 warehouseCode를 가진 행의 재고 수량(number)을 업데이트

updateContainsNumber

contains 테이블에서 특정 book\_ISBN을 가진 행의 수량(number)을 업데이트

updateOrderDate

shopping\_basket 테이블의 order\_date를 현재 날짜로 업데이트

getTableData

주어진 테이블에서 모든 데이터를 조회

insertsql

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

insertData

 table: 데이터를 삽입할 테이블의 이름.

 data: 삽입할 데이터 객체. 키는 컬럼 이름, 값은 컬럼 값

 data 객체의 키를 모아서 columns 문자열을 생성합니다.

 data 객체의 값을 모아서 values 배열을 생성합니다.

 값의 수만큼 ?를 사용하여 placeholders 문자열을 생성합니다.

 INSERT INTO ${table} (${columns}) VALUES (${placeholders}) 쿼리를 실행

insertWarehouse

warehouse 테이블에 새로운 창고 데이터를 삽입

 code: 창고 코드.

 phone: 창고 전화번호.

 address: 창고 주소.

insertInventory

inventory 테이블에 새로운 재고 데이터를 삽입

 book\_ISBN: 책의 ISBN 번호.

 warehouse\_code: 창고 코드.

 number: 재고 수량.

updatesql

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

 GET\_LOCK 함수를 호출하며, 락의 이름은 ${table}\_${lockKey}로 생성.

 락 대기 시간은 10초.

쿼리 결과에서 lockAcquired 값이 1이면 락 획득 성공, 그렇지 않으면 실패.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

 RELEASE\_LOCK 함수 호출.

 락 이름은 ${table}\_${lockKey}로 지정.

쿼리 결과에서 lockReleased 값이 1이면 락 해제 성공, 그렇지 않으면 실패.

deletesql

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

복합키와 단일키 구분

<transactionsql>

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

<create.sql>

database이름: term\_project

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

[book table]

 ISBN: 책의 고유 번호. PRIMARY KEY로 설정 (유일, NULL 불가).

 title: 책 제목.

 category: 책의 카테고리 (NULL 허용).

 price: 책 가격 (NULL 허용).

 year: 출판 연도 (4자리).

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

[warehouse 테이블]

 책을 보관하는 창고 정보를 저장합니다.

code: 창고의 고유 코드. PRIMARY KEY 설정.

phone: 창고 연락처.

address: 창고 주소.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

[inventory 테이블]

각 창고에 보관된 책의 수량을 저장

* + book\_ISBN: 책의 ISBN.
  + warehouse\_code: 창고 코드.
  + number: 보관된 책의 수량.

 **복합 PRIMARY KEY**: book\_ISBN과 warehouse\_code를 조합한 키입니다.

 **FOREIGN KEY**:

* book\_ISBN → book 테이블의 ISBN 참조.
* warehouse\_code → warehouse 테이블의 code 참조.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

[shopping\_basket 테이블]

고객의 쇼핑 바구니 정보를 저장

 basketID: 바구니 고유 ID. PRIMARY KEY.

 order\_date: 주문 날짜.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

[contains 테이블]

쇼핑 바구니에 담긴 책 정보를 저장

 book\_ISBN: 책의 ISBN.

 shopping\_basket\_basketID: 바구니 ID.

 number: 바구니에 담긴 책 수량.

**제약조건**:

* **복합 PRIMARY KEY**: book\_ISBN과 shopping\_basket\_basketID.
* **FOREIGN KEY**:
  + book\_ISBN → book 테이블 참조.
  + shopping\_basket\_basketID → shopping\_basket 테이블 참조.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

[user 테이블]

사용자 정보를 저장

 Id: 사용자 ID. PRIMARY KEY.

 Password: 비밀번호.

 Role: 사용자 역할 (예: 관리자, 고객 등).

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

[customer 테이블]

고객 정보를 저장

 Email: 고객 이메일.

 phone: 고객 전화번호.

 address: 고객 주소.

 name: 고객 이름.

**제약조건**:

* **복합 PRIMARY KEY**: Email과 name.
* **FOREIGN KEY**:
  + name → user 테이블의 Id를 참조합니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

[reservation 테이블]

책 예약 정보를 저장

 Email: 고객 이메일.

 book\_ISBN: 책의 ISBN.

 reservation\_date: 예약 날짜.

 pickup\_time: 픽업 시간.

 pickup\_date: 픽업 날짜.

**제약조건**:

* **PRIMARY KEY**: Email.
* **FOREIGN KEY**:
  + Email → customer 테이블 참조.
  + book\_ISBN → book 테이블 참조.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

[award 테이블]

 name: 상 이름. PRIMARY KEY.

 year: 수상 연도.

텍스트, 스크린샷, 폰트, 디스플레이이(가) 표시된 사진

자동 생성된 설명

[author 테이블]

 name: 작가 이름. PRIMARY KEY.

 url: 작가 웹사이트.

 address: 작가 주소.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

[written\_by 테이블]

 author\_name: 작가 이름.

 book\_ISBN: 책의 ISBN

**제약조건**:

* **복합 PRIMARY KEY**: author\_name, book\_ISBN.
* **FOREIGN KEY**:
  + author\_name → author 테이블 참조.
  + book\_ISBN → book 테이블 참조.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

[received\_by 테이블]

 author\_name: 작가 이름.

 award\_name: 상 이름.

**제약조건**:

* **복합 PRIMARY KEY**: author\_name, award\_name.
* **FOREIGN KEY**:
  + author\_name → author 테이블 참조.
  + award\_name → award 테이블 참조.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

[awarded\_to 테이블]

 award\_name: 상 이름.

 book\_ISBN: 책 ISBN.

 **복합 PRIMARY KEY**: award\_name, book\_ISBN.

 **FOREIGN KEY**:

* award\_name → award 테이블 참조.
* book\_ISBN → book 테이블 참조.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

[basket\_of 테이블]

customer\_Email: 고객 이메일. customer 테이블의 Email을 참조.

shopping\_basket\_basketID: 쇼핑 바구니 ID. shopping\_basket 테이블의 basketID를 참조.

**PRIMARY KEY**:

* **복합 키**(customer\_Email, shopping\_basket\_basketID)를 사용하여 각 고객-바구니 관계를 고유하게 식별합니다.

**FOREIGN KEY**:

* **customer\_Email**:
  + customer 테이블의 Email을 참조합니다.
  + 고객이 삭제되거나 수정될 때 basket\_of 테이블의 데이터는 영향을 받지 않도록 **ON DELETE NO ACTION**, **ON UPDATE NO ACTION** 설정이 적용됩니다.
* **shopping\_basket\_basketID**:
* shopping\_basket 테이블의 basketID를 참조합니다.
* 쇼핑 바구니가 삭제되거나 수정될 때 basket\_of 테이블의 데이터는 영향을 받지 않습니다.

<index.sql>

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

성능비교는 실행화면 부분에서..

<insert\_data.py>

//insert.sql쿼리파일만드는 파이썬코드

import random  
import string  
from datetime import datetime  
  
# ISBN 생성기  
def generate\_isbn():  
 return ''.join(random.choices(string.digits, k=13))  
  
# 랜덤 제목 생성기  
def generate\_title():  
 titles = ["Great Adventure", "Mystery Tales", "Science Wonders", "Fantasy World", "Classic Love Story"]  
 return random.choice(titles) + " " + ''.join(random.choices(string.ascii\_letters, k=5))  
  
# 랜덤 카테고리 생성기  
def generate\_category():  
 categories = ["Fiction", "Non-Fiction", "Science", "Fantasy", "Mystery", "History", "Biography"]  
 return random.choice(categories)  
  
# 랜덤 가격 생성기  
def generate\_price():  
 return random.randint(1000, 50000)  
  
# 랜덤 연도 생성기  
def generate\_year():  
 return random.randint(1980, datetime.now().year)  
  
# SQL 파일 생성  
def generate\_sql\_file(file\_name, total\_records):  
 try:  
 with open(file\_name, 'w', encoding='utf-8') as file:  
 # SQL 파일 헤더  
 file.write("-- SQL Script to Insert Data into Book Table\nSTART TRANSACTION;\n")  
 file.write("INSERT INTO Book (ISBN, title, category, price, year) VALUES\n")  
  
 for i in range(total\_records):  
 isbn = generate\_isbn()  
 title = generate\_title()  
 category = generate\_category()  
 price = generate\_price()  
 year = generate\_year()  
  
 # INSERT 데이터 생성  
 row = f"('{isbn}', '{title}', '{category}', '{price}', {year})"  
  
 # 마지막 줄이 아닌 경우 쉼표 추가  
 if i < total\_records - 1:  
 row += ",\n"  
 else:  
 row += ";\n" # 마지막 줄은 세미콜론으로 종료  
  
 # 파일에 기록  
 file.write(row)  
 file.write("COMMIT;\n")  
 print(f"SQL file '{file\_name}' created successfully with {total\_records} records.")  
  
 except Exception as e:  
 print("Error creating SQL file:", e)  
  
# 실행  
if \_\_name\_\_ == "\_\_main\_\_":  
 file\_name = "insert\_data1.sql" # 출력 SQL 파일 이름  
 total\_records = 100000 # 생성할 데이터 개수  
 generate\_sql\_file(file\_name, total\_records)

<insert\_data3.sql>

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

트랜잭션 실제로되는지 확인

로그

<정규화>

이 데이터베이스는 **제3정규형(3NF)** 이상의 정규화를 만족하도록 설계됨

**book 테이블**

* + ISBN: 기본키, 책의 고유 식별자.
  + title, category, price, year: 책의 속성.
* **정규화 상태**:
  + 1NF: 모든 컬럼이 원자값을 가짐.
  + 2NF: 비프라이머리 속성(title, category, price, year)이 기본키 ISBN에 완전 종속.
  + 3NF: 모든 속성이 기본키 이외의 속성에 종속되지 않음.

**warehouse 테이블**

* + code: 기본키, 창고의 고유 식별자.
  + phone, address: 창고의 속성.
* **정규화 상태**:
  + 1NF: 모든 컬럼이 원자값.
  + 2NF: 비프라이머리 속성(phone, address)이 기본키 code에 완전 종속.
  + 3NF: 모든 속성이 기본키 이외의 속성에 종속되지 않음.

**inventory 테이블**

* + 복합 기본키: book\_ISBN, warehouse\_code.
  + number: 재고 수량.
* **정규화 상태**:
  + 1NF: 모든 컬럼이 원자값.
  + 2NF: number는 기본키(book\_ISBN, warehouse\_code)의 모든 속성에 완전 종속.
  + 3NF: number는 기본키 외의 속성에 종속되지 않음.

**shopping\_basket 테이블**

* + basketID: 기본키, 쇼핑 바구니의 고유 식별자.
  + order\_date: 주문 날짜.
* **정규화 상태**:
  + 1NF: 모든 컬럼이 원자값.
  + 2NF: order\_date는 기본키 basketID에 완전 종속.
  + 3NF: order\_date는 기본키 외의 속성에 종속되지 않음.

**contains 테이블**

* + 복합 기본키: book\_ISBN, shopping\_basket\_basketID.
  + number: 해당 바구니에 담긴 책 수량.
* **정규화 상태**:
  + 1NF: 모든 컬럼이 원자값.
  + 2NF: number는 기본키(book\_ISBN, shopping\_basket\_basketID)에 완전 종속.
  + 3NF: number는 기본키 외의 속성에 종속되지 않음.

**customer 테이블**

* + 복합 기본키: Email, name.
  + phone, address: 고객의 속성.
* **정규화 상태**:
  + 1NF: 모든 컬럼이 원자값.
  + 2NF: phone, address는 기본키(Email, name)에 완전 종속.
  + 3NF: phone, address는 기본키 외의 속성에 종속되지 않음.

**reservation 테이블**

* + 기본키: Email.
  + book\_ISBN, reservation\_date, pickup\_time, pickup\_date: 예약 정보.
* **정규화 상태**:
  + 1NF: 모든 컬럼이 원자값.
  + 2NF: 모든 비프라이머리 속성이 기본키(Email)에 완전 종속.
  + 3NF: 모든 속성이 기본키 외의 속성에 종속되지 않음.

**award 테이블**

* + name: 기본키, 상 이름.
  + year: 수상 연도.
* **정규화 상태**:
  + 1NF: 모든 컬럼이 원자값.
  + 2NF: year는 기본키 name에 완전 종속.
  + 3NF: year는 기본키 외의 속성에 종속되지 않음.

**관계 테이블 (written\_by, received\_by, awarded\_to, basket\_of)**

* **정규화 상태**:
  + 모든 관계 테이블은 복합 기본키를 사용하여 두 엔터티 간의 관계를 정의.
  + 1NF, 2NF, 3NF를 모두 만족.

1. 실행 화면

* 프로그램을 실행시켰을 때의 결과 화면을 캡처하여 붙여넣기
* 실제로 실행했을 때와 결과가 같아야 하며, 해당 실습 또는 과제에서 요구하는 사항이 반드시 포함되어 있어야 함(없을 시 감점)

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 도표, 흑백이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 도표이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 흑백, 도표이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명

로그인화면 /페이지

텍스트, 스크린샷, 폰트, 정보기기이(가) 표시된 사진

자동 생성된 설명

login 실패시 실패했다는 알림띄우고 login화면을 다시보여준다.

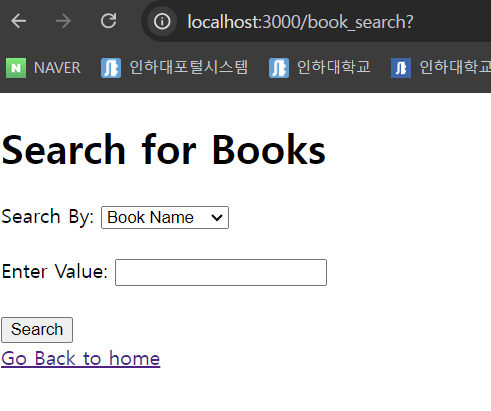
텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

customer 로그인 성공시 해당 사용자의 이름과 logout버튼, book search, shopping basket, reservation페이지로 이동할수있는 버튼이 화면에 보인다.



book search를 눌러 book\_search페이지로 이동하면 book, author, award를 선택할 수 있는 list가 있고 찾으려는 값을 입력할 수 있는 입력창이 있다. go back to home을 누르면 이전페이지로 돌아간다.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

hugo award를 검색해보자.  
텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

search를 누르면 다음과 같은 결과가 조회된다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

재고가 없으면 이런식으로 예약, 쇼핑카트에 추가하지못한다.

<shopping basket>

텍스트, 스크린샷, 폰트, 원이(가) 표시된 사진

자동 생성된 설명

구매내역이 있는 책은 다시구매하지 못하도록 설정하였다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 멀티미디어, 폰트이(가) 표시된 사진

자동 생성된 설명

쇼핑카트로 정상적으로 추가되었을 때 위와같이 알림이 뜬다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

조회할수있다.

구매수량을 선택하고 원하는 warehouse를 선택하여 구매를 누르면

텍스트, 스크린샷, 멀티미디어, 폰트이(가) 표시된 사진

자동 생성된 설명

구매가 성공적으로 진행되었다는 알림이뜨고

텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

자동 생성된 설명

구매한 목록은 쇼핑카트에 보이지않도록 설정하였다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

실제로 total inventory가 2권이 줄어든 것을 볼수있다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

총 개수를 초과하여 구매하지못하도록 설정

<reservation>

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

book\_search페이지에서 reservation선택

텍스트, 스크린샷, 폰트, 멀티미디어이(가) 표시된 사진

자동 생성된 설명

한번 더 누르면 이미 예약된 기록이 있다는 알림을 띄운다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

/book\_search에서 reservation할때에는 날짜,시간=null 텍스트, 스크린샷, 폰트, 멀티미디어이(가) 표시된 사진

자동 생성된 설명

예약성공시 알람

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

예약했을떄 위와같이 pickup date와 pickup time이 뜬다.

텍스트, 스크린샷, 폰트, 정보기기이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명  
cancel누르면 성공적으로 취소되면서 화면에 예약기록이 삭제된다.

<administer>

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

administer로그인 성공화면

원하는 table을 선택할 수 있는 list가있다.

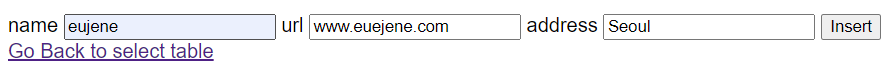
텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

author를 선택하면 해당 페이지처럼 조회한 결과나온다.



eujene을 insert해보면

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

성공적으로 insert되었다고 나오고

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

추가된것을볼수있다.

텍스트, 라인, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

seoul부분을 incheon으로 update해보겠다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

성공했다는 알림이 뜨고

텍스트, 라인, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

실제로 조회했을 때 incheon으로 뜨는 것을 볼 수있다.

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

delete버튼을 누르면 성공했다는 알림이 뜨고

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

delete이후 화면 eujene이 사라진 것을 볼 수 있다.

<indexing 성능비교>

[index전]

스크린샷, 텍스트, 블랙, 라인이(가) 표시된 사진

자동 생성된 설명

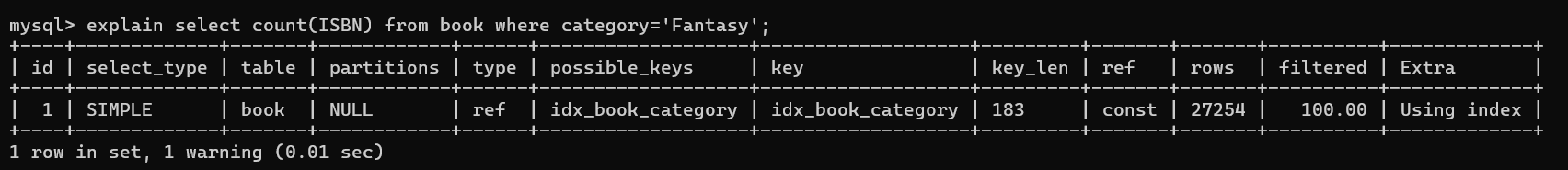
텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명

[index후]

텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명



isbn, title, category에 대한 index를 설정 후 category에 대한 cardinality는 6으로 매우 낮고 중중복 많은 것을 알수있다. 나머지 부분은 중복이 적고 고유하다고 볼 수 있다.

시간은 0.03초에서 0.02초로 줄어들었다.

key가 생성됨 rows 27254, filtered가 100%으로 되어 효율적으로 테이블을 스캔하고있는 것을 볼수있다. 모든 검색된 행이 결과에 포함됨

1. 결론

* 구현을 못했을 때의 문제점(어떤 점이 부족해서 구현을 못하였는지에 대한 분석)
* 구현하였을 경우 결과에 대한 분석

termproject에서는 관리자 및 고객 웹 페이지를 구성하고 관련 데이터베이스를 설계 및 구현하며, 주어진 요구사항을 만족하기 위한 주요 기능들을 실습하였다. 이를 위해 데이터베이스를 설계하고 트랜잭션, 제약조건, 정규화, 인덱스를 활용하여 데이터 무결성과 성능을 보장하며, 관리와 사용의 편의성을 충족하는 구조를 구축해보았다.

먼저 관리자 페이지에서는 Book, Author, Award, Warehouse, Inventory, Contains와 같은 테이블의 데이터를 입력, 수정, 삭제할 수 있도록 구현했다. 또한, MySQL의 GET\_LOCK과 RELEASE\_LOCK을 사용하여 동일 데이터를 여러 관리자가 동시에 수정하지 못하도록 설계해 데이터 충돌을 방지했다. 이를 통해 관리자가 안전하게 데이터를 관리할 수 있는 환경을 제공했다.

고객 페이지에서는 검색, 예약, 구매와 같은 다양한 기능을 구현했다. 고객은 Book, Author, Award의 이름을 통해 검색할 수 있으며, 검색된 책의 총 재고를 확인할 수 있다. 예약 기능에서는 재고 확인, 예약 시간 충돌 방지 로직을 적용하여 예약 과정에서의 오류를 최소화했다. 또한, 고객은 예약된 데이터를 조회하고 픽업 시간을 수정하거나 예약을 취소할 수 있었다. 구매 기능에서는 쇼핑 바구니를 통해 여러 권의 책을 담아 구매할 수 있도록 했으며, 재고가 부족한 경우 구매를 제한하여 데이터 무결성을 유지했다.

데이터베이스 설계는 ERD를 기반으로 각 엔터티와 관계를 명확히 정의했으며, 모든 테이블은 제3정규형(3NF)을 만족하도록 설계하였다. 이를 통해 데이터 중복을 제거하고, 모든 속성이 기본키에만 종속되도록 하여 데이터 무결성을 유지했다. 또한, 외래 키 제약조건과 인덱스를 활용해 데이터 참조 관계를 관리하고 검색 성능을 최적화하였다.

트랜잭션은 데이터 안정성을 보장하기 위해 예약 및 구매와 같은 중요한 작업에 적용되었다. 트랜잭션은 작업 중 오류가 발생하면 데이터를 원래 상태로 복구하며, 성공적인 작업만 데이터베이스에 반영되도록 설계되었다. 예를 들어, 예약 과정에서는 재고 확인, 시간 충돌 검토, 데이터 삽입이 하나의 트랜잭션으로 처리되어 작업이 안전하게 이루어졌다.