<JAVA에서의 표준 입출력>

1. import java.util.**Scanner**
2. **public** class (변수이름)
   public static void main(매개변수)

<연산자>

1. 전치 : 변수 값 증감 후 연산
2. 후치 : 연산 후 변수 값 증감
3. 논리연산자/비트연산자 구분
 - && : and 논리연산자
 - || : or 논리연산자
 - & : and 비트연산자
 - ^ : xor 비트연산자
 - | : or 비트연산자

<연산자 우선순위>

* 콤마 구분 : 자기들끼리의 우선순위

1. 단항연산자 (! ~ ++ -- ...)
2. 이항연산자
 - 산술연산자 : * / %, + -
 - 시프트연산자 : << >>
 - 관계연산자 : < > <= >=, == !=
 - 비트연산자 : &, ^, |
 - 논리연산자 : &&, ||
3. 삼항연산자 : ? :
4. 대입연산자 : == += ...
5. 순서연산자 : ,(콤마)

<포인터>

1. a[]라는 배열이 있을 때

*a+i = a[0]+i

*(a+i) = a[0+i]

<헝가리안 표기법>

- 변수명 작성 시 변수의 자료형을 알 수 있도록 자료형을 의미하는 문자를 포함하여 작성하는 방법

8. SQL 응용

<GRANT/REVOKE>

- GRANT ... TO ... : 권한 부여를 위한 명령어
- REVOKE ... FROM ... : 권환 취소를 위한 명령어
- WITH GRANT OPTION : 부여받은 권한을 다른 사용자에게 다시 부여할 수 있는 권한 부여
- GRANT OPTION FOR : 다른 사용자에게 권한을 부여할 수 있는 권한을 취소
- CASECADE : 권한 취소 시 권한을 부여받았던 사용자가 다른 사용자에게 부여한 권한도 연쇄적으로 취소

<COMMIT/ROLLBACK/SAVEPOINT>

COMMIT	트랜잭션이 성공적으로 끝나면 데이터베이스가 새로운 일관성 상태를 가지기 위해 변경된 모든 내용을 데이터베이스에 반영해야하는데, 이 때 사용되는 명령어
ROLLBACK	아직 COMMIT되지 않은 변경된 모든 니용들을 취소하고 데이터베이스를 이전 상태로 되돌리는 명령어
SAVEPOINT	트랜잭션 내에 ROLLBACK 할 위치인 저장점을 지정하는 명령어, 저장점을 지정할 때는 이름을 부여하며, ROLLBACK시 저장된 저장점까지의 트랜잭션 처리 내용이 취소됨

<DML-JOIN>

- INNER JOIN : 관계가 설정된 두 테이블에서 조인된 필드가 일치하는 행만을 표시
- OUTER JOIN : 릴레이션에서 JOIN 조건에 만족하지 않는 튜플도 결과로 출력

<프로시저>

- 절차형 SQL을 활용하여 특정 기능을 수행하는 일종의 트랜잭션 언어, 호출을 통해 실행, 미리 저장해 놓은 SQL 작업을 수행
- 데이터베이스에 저장되어 수행됨
- 시스템의 일일 마감 작업, 일괄 작업 등에 주로 사용

<트리거>

- 데이터베이스 시스템에서 데이터의 삽입, 갱신, 삭제 등의 이벤트가 발생할 때마다 관련 작업이 자동으로 수행되는 절차형 SQL
- 데이터베이스에 저장되며, 데이터 변경 및 무결성 유지, 로그 메시지 출력 등의 목적으로 사용됨

<커서>

- 쿼리문의 처리 절차가 저장되어 있는 메모리 공간을 가리키는 포인터
- 커서의 수행 : 열기 Open, 패치 Fetch, 닫기 Close
- 묵시적 커서 : DBMS 자체적으로 열리고 패치되어 사용이 끝나면 닫히지만 커서의 속성을 조회하여 사용된 쿼리 정보를 열람하는 것 가능
- 명시적 커서 : 사용자가 직접 정의해서 사용, 명시적 커서로 사용하기 위해서는 열기 전에 선언(Declare)을 해야함

1. 요구사항 확인

<소프트웨어 생명 주기>

- 폭포수 모형 : 각 단계를 확실히 매듭짓고 그 결과를 철저히 검토하여 승인과정을 거친 후 다음 단계를 진행하는 개발 방법론
- 프로토타입 모형 : 실제 개발될 소프트웨어에 대한 프로토타입을 만들어 최종 결과물을 예측하는 모형
- 나선형 모형 : 여러 번의 소프트웨어 개발 과정을 거쳐 점진적으로 완벽한 소프트웨어를 개발하는 모형
  (계획수립) - (위험분석) - (개발 및 검증) - (고객평가)
- 애자일 모형 : 요구사항 변화에 유연하게 대응할 수 있도록 일정한 주기를 반복하면서 개발하는 모형 (스크럼, XP)

- 소프트웨어 공학 : 소프트웨어의 위기를 극복하기 위한 방안으로 연구된 학문

스크럼 기법	팀이 중심이 되어 개발의 효율성을 높이는 기법 (제품책임자-PO, 스크럼 마스터-SM, 개발팀-DT)
XP (extreme programming)	고객 요구사항에 유연하게 대응하기 위해 고객의 참여와 개발 과정의 반복을 극대화하여 생산성 향상

<개발 기술 환경>

운영체제	컴퓨터 시스템의 자원을 효율적으로 관리하며 사용자가 컴퓨터를 편리하고 효율적으로 사용할 수 있도록 환경을 제공하는 소프트웨어	가용성 성능 기술지원 주변기기 구축비용
데이터베이스 관리 시스템 (DBMS)	사용자와 DB 사이에서 사용자의 요구에 따라 정보를 생성해 주고, DB를 관리해주는 소프트웨어	가용성 성능 기술지원 상호호환성 구축비용
웹 어플리케이션 서버(WAS)	사용자의 요구에 따라 변하는 동적인 콘텐츠를 처리하기 위해 사용되는 미들웨어	가용성 성능 기술지원 구축비용
오픈소스	누구나 제한없이 사용할 수 있는 소스코드를 공개한 소프트웨어	라이선스 종류, 사용자 수, 기술의 지속 가능성

<요구사항 유형>

기능 요구사항	기능, 수행과 관련된 요구사항
비기능 요구사항	품질, 제약사항과 관련된 요구사항
사용자 요구사항	사용자 관점에서 본 시스템이 제공해야 할 요구사항
시스템 요구사항	개발자 관점에서 본 시스템 전체가 사용자와 다른 시스템에 제공해야 할 요구사항

<요구사항 개발 프로세스>

- 요구사항 도출 → 분석 → 명세 → 확인

<요구사항 분석>

- 자료 흐름도(DFD) : 자료의 흐름 및 변환 과정과 기능을 도형 중심으로 기술하는 방법

  프로세스, 자료 흐름(Data Flow), 자료 저장소(Data Store), 단말(Terminator)
- 자료 사전(DD) : 자료흐름도에 있는 자료를 정의하고 기록한 것

=	정의	+	연결	( )	생략	[ ]	선택
{ }	반복					**	주석

- 요구사항 분석용 CASE (자동화 도구) : 요구사항을 자동으로 분석하고 요구사항 분석 명세서를 기술하도록 개발된 도구
- HIPO(Hierachy Input Process Output) : 시스템 실행 과정인 입력, 처리, 출력의 기능을 표현한 것 (가시적 도표 / 총체적 도표 / 세부적 도표)

<UML>

- 시스템 개발 과정에서 개발자와 고객 등 상호간의 의사소통이 원활하게 이루어지도록 표준화한 대표적인 객체지향 모델링 언어이다.

사물 (things)	다이어그램 안에서 관계가 형성될 수 있는 대상 · 구조사물 : 시스템의 개념적, 물리적 요소 · 행동사물 : 시공간에 따른 요소들의 행위 · 그룹사물 : 요소들을 그룹으로 묶어서 표현 · 주해사물 : 부가적인 설명이나 제약조건 표현
관계 (relation-ships)	사물과 사물 사이의 연관성 표현 · 연관: 2개 이상의 사물이 서로 연관 · 집합: 하나의 사물이 다른 사물에 포함 · 포함: 포함하는 사물의 변화가 포함되는 사물에 영향 · 일반화: 하나의 사물이 다른 사물에 비해 더 일반적인지 구체적인지 · 의존: 필요에 의해 짧은 시간 동안만 연관 유지 · 실체화: 사물이 할 수 있거나 해야하는 기능으로 서로를 그룹화 할 수 있는 관계
다이어그램 (diagram)	사물과 관계를 도형으로 표현한 것

<구조적 다이어그램>

- 클래스 다이어그램 : 클래스와 클래스가 가지는 속성, 클래스 사이의 관계 표현 (클래스, 제약조건, 관계)
- 객체 다이어그램 : 클래스에 속한 인스턴스를 특정 시점의 객체와 객체 사이의 관계로 표현
- 컴포넌트 다이어그램 : 실제 구현 모듈인 컴포넌트 간의 관계나 컴포넌트 간의 인터페이스 표현
- 배치 다이어그램 : 결과물, 프로세스, 컴포넌트 등 물리적 요소들의 위치 표현
- 복합체 구조 다이어그램 : 클래스나 컴포넌트가 복합 구조를 가지는 경우
- 패키지 다이어그램 : 유스케이스나 클래스 등의 모델 요소들을 그룹화한 패키지들의 관계 표현

<행위 다이어그램>

- 유스케이스 다이어그램 : 사용자의 요구를 분석하는 것 (사용자-Actor, 사용 사례-Use Case)
- 시퀀스 다이어그램 : 시스템이나 객체들이 메시지를 주고받으며 상호작용하는 과정을 그림으로 표현한 것
- 커뮤니케이션 다이어그램 : 동작에 참여하는 객체들이 주고받는 메시지와 객체들 간의 연관 관계를 표현
- 상태 다이어그램 : 하나의 객체가 자신이 속한 클래스의 상태변화 혹은 다른 객체와의 상호작용에 따라 상태가 어떻게 변화하는지를 표현
- 활동 다이어그램 : 시스템이 어떤 기능을 수행하는지 객체의 처리 로직이나 조건에 따른 처리의 흐름을 순서에 따라 표현
- 상호작용 개요 다이어그램 : 상호작용 다이어그램 간의 제어 흐름 표현
- 타이밍 다이어그램 : 객체 상태 변화와 시간 제약을 명시적으로 표현

<스테레오 타입>

- UML에서 표현하는 기본 기능 외 추가적인 기능 표현

- 《include》, 《extend》, 《interface》, 《exception》, 《constructor》

<유스케이스 다이어그램>
- 개발될 시스템을 이용해 수행할 수 있는 기능을 사용자의 관점으로 표현한 것
- 시스템 / 시스템 범위 : 시스템 내부의 유스케이스들을 사각형으로 묶어 시스템 범위 표현
- 액터 : 시스템과 상호작용하는 모든 외부 요소(주액터/부액터)
- 유스케이스 : 사용자가 보는 관점에서 시스템이 엑터에게 제공하는 서비스나 기능
- 관계 : 액터-유스케이스, 유스케이스-유스케이스 / 포함관계, 확장관계, 일반화관계

<활동 다이어그램>
- 사용자의 관점에서 시스템이 수행하는 기능을 처리 흐름에 따라 순서대로 표현
- 액션, 액티비티, 노드, 스윔레인
- 액션 : 더 이상 분해할 수 없는 단일작업
- 액티비티 : 몇 개의 액션으로 분리 가능

<클래스 다이어그램>
- 정적 모델링 : 사용자가 요구한 기능을 구현하는데 필요한 자료들의 논리적인 구조를 표현
- 클래스 다이어그램 : 클래스, 클래스가 가지는 속성, 클래스 사이의 관계를 표현
- 클래스 : 각 객체들이 갖는 속성과 오퍼레이션(동작)을 표현
- 속성 : 클래스의 상태나 정보를 표현
- 오퍼레이션 : 클래스가 수행할 수 있는 동작, 함수(메소드)
- 제약조건 : 속성에 입력될 값에 대한 제약조건이나 오퍼레이션 수행 전후에 지정해야 할 조건이 있을 때 기술
- 관계 : 클래스와 클래스 사이의 연관성 표현 (연관 관계, 집합 관계, 포함 관계, 일반화 관계, 의존 관계)
- 연관 클래스 : 연관 관계에 있는 두 클래스에 추가적으로 표현해야 할 속성이나 오퍼레이션이 있는 경우 생성하는 클래스

<시퀀스 다이어그램>
- 동적 모델링 : 시스템의 내부 구성 요소들의 상태 변화 과정과 과정에서 발생하는 상호 작용을 표현
- 시퀀스 다이어그램 : 시스템이나 객체들이 메시지를 주고받으며 상호작용하는 과정을 그림으로 표현한 것
- 액터, 객체, 생명선, 실행상자, 메시지, 객체 소멸, 프레임

<그 외>
- 커뮤니케이션 다이어그램 : 시스템이나 객체들이 메시지를 주고받으며 상호작용하는 과정과 객체들 간의 연관을 그림으로 표현한 것 (액터, 객체, 링크, 메시지)
- 상태 다이어그램 : 객체들 사이에서 발생하는 이벤트에 의한 객체들의 상태 변화를 그림으로 표현한 것 (상태, 시작상태, 종료상태, 상태전환, 이벤트, 프레임)
- 패키지 다이어그램 : 요소들을 그룹화한 패키지 간의 의존관계를 표현한 것 (패키지, 객체, 의존 관계)

<SW공학의 발전적 추세>
- 소프트웨어 재사용 : 이미 개발되어 인정받은 소프트웨어를 다른 소프트웨어 개발이나 유지에 사용하는 것 (합성 중심, 생성 중심)
- 소프트웨어 재공학 : 기존 시스템을 이용하여 보다 나은 시스템을 구축하고 새로운 기능을 추가하여 소프트웨어 성능을 향상시키는 것
- CASE(Computer Aided Software Engineering) : 소프트웨어 개발 과정에서 사용되는 과정 전체 또는 일부를 컴퓨터와 전용 소프트웨어 도구를 사용하여 자동화하는 것

<비용 산정 기법 - 하향식>
- 하향식 비용 산정 : 과거의 유사한 경험을 바탕으로 전문지식이 많은 개발자들이 참여한 회의를 통해 비용을 산정
- 전문가 감정 기법 : 조직 내 경험이 많은 두 명 이상의 전문가에게 비용 산정 의뢰
- 델파이 기법 : 전문가 감정 기법의 주관적인 편견을 보완하기 위해 많은 전문가들의 의견을 종합하여 산정하는 기법

<비용 산정 기법 - 상향식>
- 상향식 비용 산정 : 세부적인 작업 단위별로 비용을 산정한 후 집계하여 전체 비용을 산정
- LOC(원시 코드 라인 수)
- 개발 단계별 인월수 : 기능을 구현시키는데 필요한 노력을 생명주기 각 단계별로 산정

<수학적 산정 기법(상향식 비용 산정 기법)>
- COCOMO : LOC를 예측한 후 소프트웨어의 종류에 따라 다르게 책정

COCOMO의 소프트웨어 개발 유형	COCOMO 모형의 종류
조직형 : 5만라인 이하의 소프트웨어 개발 / 사무 처리용, 업무용, 과학용 응용 SW 개발에 적합	기본형 : 소프트웨어 크기와 개발 유형만을 이용
반분리형 : 30만 라인 이하의 소프트웨어 개발 / 유틸리티 개발에 적합	중간형 : 기본형을 토대로 제품 · 컴퓨터 · 개발 요원 · 프로젝트 특성 고려
내장형 : 30만 라인 이상의 초대형 소프트웨어 개발 / 시스템 프로그램 개발에 적합	발전형 : 중간형 COCOMO 보완, 공정별로 자세하고 정확하게 노력 산출

- Putnam 모형 : 소프트웨어 생명 주기의 전 과정 동안 사용될 노력의 분포 예상
- 기능 점수(FP:Function Point) 모형 : 소프트웨어 기능을 증대시키는 요인별로 기능점수를 구한 후 비용 산정
- 비용 산정 자동화 추정 도구 : SLIM (Putnam 예측 모델 기초), ESTIMACS (FP 모형 기초)

<소프트웨어 개발 표준>
- ISO/IEC 12207
- CMMI : 소프트웨어 개발 조직의 업무 능력 및 조직의 성숙도를 평가하는 모델

초기	정의된 프로세스 없음	작업자의 능력에 따라 성공 여부 결정
관리	규칙화된 프로세스	특정 프로젝트 내의 프로세스 정의 및 수행
정의	표준화된 프로세스	조직의 표준 프로세스를 활용하여 업무 수행
정량적 관리	예측 가능한 프로세스	프로세스를 정량적으로 관리 및 통제
최적화	지속적 개선 프로세스	프로세스 역량 향상을 위해 지속적인 프로세스 개선

- SPICE : 소프트웨어의 품질 및 생산성 향상을 위해 소프트웨어 프로세스를 평가 및 개선하는 국제 표준

고객-공급자 프로세스	소프트웨어를 개발하여 고객에게 전달하는 것 지원, SW의 정확한 운용 및 사용을 위한 프로세스
공학 프로세스	시스템과 소프트웨어 제품의 명세화, 구현, 유지보수 하는데 사용되는 프로세스
지원 프로세스	소프트웨어 생명 주기에서 다른 프로세스에 의해 이용되는 프로세스로 구성
관리 프로세스	소프트웨어 생명 주기에서 프로젝트 관리자에 의해 사용되는 프로세스
조직 프로세스	조직의 업무 목적 수립과 조직의 업무 목표 달성을 위한 프로세스

불완전	프로세스가 구현되지 않았거나 목적을 달성하지 못함
수행	프로세스가 수행되고 목적 달성
관리	정의된 자원의 한도 내에서 그 프로세스가 작업 산출물을 인도하는 단계
확립	소프트웨어 공학 원칙에 기반하여 정의된 프로세스가 수행
예측	프로세스 목적 달성을 위해 통제되고 양적인 측정을 통해 일관되게 수행
최적화	프로세스 수행 최적화, 지속적인 개선을 통해 업무 목적 만족

<소프트웨어 개발 프레임워크>

- 소프트웨어 개발에 공통적으로 사용되는 구성요소와 아키텍처를 일반화하여 제공해주는 반제품 형태의 소프트웨어 시스템
- 스프링 프레임워크(자바), 전자정부 프레임워크(공공 사업), 닷넷 프레임워크(윈도우)

모듈화	캡슐화를 통해 모듈화 강화, 설계에 영향 최소화 유지보수 용이
재사용성	재사용 가능한 모듈 제공 예산 절감, 생산성 향상, 품질보증 가능
확장성	다형성을 통한 인터페이스 확장 가능 (다양한 형태, 기능)
제어의 역흐름	개발자가 관리하고 통제해야하는 객체들의 제어를 프레임워크에 넘겨 생산성 향상

2. 데이터 입출력 구현

<데이터베이스>

- 데이터베이스 : 공동으로 사용될 데이터를 중복을 배제하여 통합하고 저장장치에 저장하여 항상 사용할 수 있도록 운영하는 운영 데이터
- DBMS(database management system) : 사용자의 요구에 따라 정보를 생성해주고 데이터베이스를 관리해주는 소프트웨어 (정의기능, 조작기능, 제어기능)
- 스키마(schema) : 데이터베이스의 구조와 제약조건에 관한 전반적인 명세를 기술한 것이다.

외부 스키마	사용자나 응용프로그래머가 개인의 입장에서 필요로하는 데이터베이스의 논리적 구조 정의
개념 스키마	데이터베이스의 전체적인 논리적 구조 모든 응용프로그램이나 사용자들이 필요로 하는 데이터를 종합한 조직 전체의 데이터베이스
내부 스키마	물리적 저장장치의 입장에서 본 데이터베이스 구조 실제로 저장될 데이터 형식, 저장 데이터 항목의 표현 방법, 내부 레코드의 물리적 순서 등을 나타냄

<데이터베이스 설계>

무결성	연산 후에도 DB가 저장된 데이터가 정해진 제약조건을 항상 만족해야함
일관성	DB가 저장된 데이터들 사이, 특징 질의에 대한 응답이 처음부터 끝까지 변함없이 일정해야함
회복	시스템 장애가 발생했을 때 그 직전의 상태로 복구할 수 있어야함
보안	불법적인 데이터의 노출, 변경, 손실로부터 보호할 수 있어야함
효율성	응답시간의 단축, 시스템의 생산성, 저장공간의 최적화
데이터베이스 확장	DB운영에 영향을 주지 않으면서 지속적으로 데이터를 추가할 수 있어야함

- 설계 순서 : 요구조건 분석 → 개념적 설계 → 논리적 설계 → 물리적 설계 → 구현

<데이터 모델>

구성

- 개체(Entity) : DB에 표현하려는 것, 개념이나 정보 단위 같은 현실 세계의 대상체
- 속성(Attribute) : DB를 구성하는 가장 작은 논리적 단위
- 관계(Relationship) : 개체 간 관계 또는 속성 간의 논리적 연결을 의미

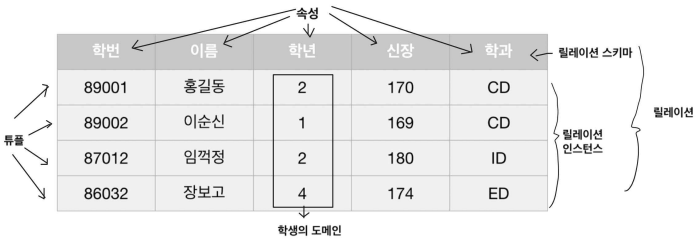
종류

- 개념적 데이터 모델 : 현실세계에 대한 인식을 추상적 개념으로 표현
- 논리적 데이터 모델 : 개념적 구조를 컴퓨터세계의 환경에 맞도록 변환
- 물리적 데이터 모델 : 실제 컴퓨터에 데이터가 저장되는 방법을 정의

표시할 요소

- 구조 : 개체타입들 간 관계
- 연산 : 실제 데이터 처리 작업에 대한 명세
- 제약조건 : 실제 데이터의 논리적인 제약조건

<관계형 데이터베이스>



- 튜플 : 릴레이션을 구성하는 각각의 행 (튜플의 수 : 카디널리티, 기수)

- 속성(Attribute) : 데이터베이스를 구성하는 가장 작은 논리적 단위 (속성의 수 : 디그리, 차수)
- 도메인 : 하나의 속성이 취할 수 있는 같은 타입의 원자의 집합
- 키

후보키	튜플을 유일하게 식별하기 위해 사용되는 속성들의 부분집합(유일성O, 최소성O)
기본키	후보키 중 특별히 선정된 Main Key
대체키	후보키가 둘 이상일 때 기본키를 제외한 나머지 후보키
슈퍼키	속성들의 집합으로 구성된 키 (유일성O, 최소성X)
외래키	다른 릴레이션의 기본키를 참조하는 속성 또는 속성들의 집합

<무결성>

- 데이터베이스에 저장된 데이터 값과 현실 세계의 실제 값이 일치하는 정확성

개체 무결성	기본 테이블의 기본키를 구성하는 어떤 속성도 NULL 값이나 중복값을 가질 수 없음
참조 무결성	외래키 값은 NULL이거나 참조 릴레이션의 기본키 값과 동일해야함 (릴레이션은 참조할 수 없는 외래키 값을 가질 수 없음)
도메인 무결성	주어진 속성 값이 정의된 도메인에 속한 값이어야함
사용자 정의 무결성	속성 값들이 사용자가 정의한 제약조건에 만족되어야 함
NULL 무결성	릴레이션의 특정 속성 값이 NULL이 될 수 없음
고유 무결성	릴레이션의 특정 속성에 대해 각 튜플이 갖는 속성값들이 서로 달라야함
키 무결성	하나의 릴레이션에는 적어도 하나의 키가 존재해야함
관계 무결성	릴레이션에 어느 한 튜플의 삽입 가능 여부 또는 한 릴레이션과 다른 릴레이션의 튜플들 사이의 관계에 대한 적절성 여부

<관계대수 및 관계해석>

- 관계 대수 : 관계형 DB에서 원하는 정보와 그 정보를 검색하기 위해서 어떻게 유도하는가를 기술하는 절차적 언어
- 관계해석 : 관계 데이터의 연산을 표현하는 방법, 원하는 정보가 무엇인지만 정의하는 비절차적 특성

<이상>

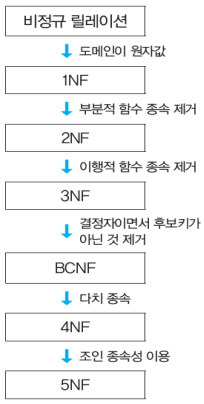
- 이상 : 테이블에서 일부 속성들의 종속으로 인해 데이터의 중복이 발생하고 이로 인해 테이블 조작 시 문제가 발생하는 현상
- 삽입 이상 : 테이블에 데이터를 삽입할 때 원하지 않는 값들로 인해 삽입할 수 없게 됨
- 삭제 이상 : 테이블에서 한 튜플을 삭제할 때 의도와 상관없는 값들도 함께 삭제됨
- 갱신 이상 : 테이블에서 튜플에 있는 속성 값을 갱신할 때 일부 튜플의 정보만 갱신되어 정보에 불일치성이 생기는 현상

<함수적 종속>

- 완전 함수적 종속 : 하나의 속성에 대해서만 종속일 때
- 부분 함수적 종속 : 하나의 속성에 대해서 종속이며 다른 속성의 임의의 진부분 집합에 대해 함수적 종속일 때

<정규화>

- 테이블의 속성들이 상호 종속적인 관계를 갖는 특성을 이용하여 테이블을 무손실 분해하는 과정



<반정규화>

- 시스템의 성능 향상, 개발 및 운영의 편의성 등을 위해 의도적으로 정규화 원칙을 위배하는 행위
- 테이블 통합 : 두 개의 테이블이 조인되는 경우 하나의 테이블로 합쳐 사용하는 것이 성능 향상에 도움이 될 경우 수행
- 테이블 분할 : 수평분할(레코드 기준으로 테이블 분할 - 레코드 별로 사용 빈도의 차이가 큰 경우), 수직분할(속성을 기준으로 테이블 분할 - 하나의 테이블에 속성이 너무 많을 경우)
- 중복 테이블 추가 : 여러 테이블에서 데이터를 추출하거나 다른 서버에 저장된 테이블을 이용해야 하는 경우
- 중복 속성 추가 : 데이터를 조회하는 경로를 단축하기 위해

<시스템 카탈로그>

- 다양한 객체에 관한 정보를 포함하는 시스템 데이터베이스
- 메타데이터 : 시스템 카탈로그에 저장된 정보
- 데이터 디렉터리 : 데이터 사전에 수록된 데이터에 접근하는 데 필요한 정보를 관리 유지하는 시스템

<인덱스>

- 데이터 레코드를 빠르게 접근하기 위해 <키 값, 포인터> 쌍으로 구성되는 데이터 구조
- 클러스터드 인덱스(데이터 정렬), 넌클러스터드 인덱스(인덱스 값만 정렬, 실제 데이터는 정렬 X)

트리 기반 인덱스	인덱스 저장 블록이 트리모양
비트맵 인덱스	인덱스 칼럼의 데이터를 0 or 1로 변환
함수 기반 인덱스	컬럼 값 대신 특정 함수나 수식 적용
비트맵 조인 인덱스	다수의 조인된 객체로 구성된 인덱스
도메인 인덱스	개발자가 필요한 인덱스를 직접 만들어 사용하는 것

<뷰 / 클러스터>

- 사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 하나 이상의 기본 테이블로부터 유도된 가상 테이블
- 저장장치 내에 물리적으로 존재하진 않음, 사용자에게는 있는 것처럼 간주
- CREATE, DROP
- 데이터의 논리적 독립성 제공
- 클러스터 : 동일한 성격의 데이터를 동일한 데이터 블록에 저장하는 물리적 저장 방법

<트랜잭션>

- 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을

수행하기 위한 작업의 단위
- 한꺼번에 모두 수행되어야 할 일련의 연산들

원자성 (atomicity)	트랜잭션 연산은 DB에 모두 반영되도록 완료되든지 전혀 반영되지 않도록 복구되든지
일관성 (consistency)	트랜잭션이 그 실행을 성공적으로 완료하면 언제나 일관성 있는 데이터베이스 상태로 변환
독립성 (isolation)	둘 이상의 트랜잭션이 동시에 병행 실행되는 경우 어느 하나의 트랜잭션 실행 중에 다른 트랜잭션의 연산이 끼어들 수 없음
지속성 (durability)	성공적으로 완료된 트랜잭션의 결과는 시스템이 고장나더라도 영구적으로 반영되어야 함

<파티션>
- 대용량 테이블이나 인덱스를 작은 논리적 단위로 나누는 것
- 범위 분할 : 지정한 열의 값을 기준으로 분할
- 해시 분할 : 해시함수를 적용한 결과값에 따라 데이터 분할
- 조립 분할 : 범위 분할로 분할한 다음 해시 함수를 적용하여 다시 분할
- 인덱스 파티션 : 파티션 된 테이블의 데이터를 관리하기 위해 인덱스를 나눈 것

<분산 데이터베이스 설계>
- 분산 데이터베이스 : 논리적으로는 하나의 시스템에 속하지만 물리적으로는 네트워크를 통해 연결된 여러 개의 사이트에 분산된 데이터베이스

위치 투명성	엑세스하려는 데이터베이스의 위치를 알 필요 없이 논리적 명칭으로만 엑세스할 수 있다
중복 투명성	동일 데이터가 여러곳에 중복되어 있더라도 마치 하나의 데이터만 존재하는 것처럼 사용하고 시스템은 자동으로 여러 자료에 대한 작업을 수행한다.
병행 투명성	분산 데이터베이스와 관련된 다수의 트랜잭션들이 동시에 실현되더라도 그 트랜잭션의 결과는 영향을 받지 않는다
장애 투명성	장애에도 트랜잭션을 정확하게 처리한다

<데이터베이스 이중화/서버 클러스터링>
- 데이터베이스 이중화 : 동일한 데이터베이스를 복제하여 관리(활동-대기, 활동-활동)
- 클러스터링 : 두 대 이상의 서버를 하나의 서버처럼 운영하는 기술

고가용성 클러스터링	하나의 서버에 장애가 발생하면 다른 노드가 받아 처리
병렬처리 클러스터링	하나의 작업을 여러 개의 서버에서 분산하여 처리

- RTO(recovery time objective, 목표 복구 시간) : 장애 발생 시점부터 복구되어 가동될 때 까지의 소요시간
- RPO(recovery point objective, 목표 복구 시점) : 장애 발생 기준점 의미

<스토리지>
- 대용량의 데이터를 저장하기 위해 서버와 저장장치를 연결하는 기술
- DAS(direct attached storage) : 서버와 저장장치를 전용 케이블로 직접 연결하는 방식
- NAS(network attached storage) : 서버와 저장장치를 네트워크를 통해 연결하는 방식
- SAN(storage area network) : 서버와 저장장치를 연결하는 전용 네트워크를 별도로 구성

3. 통합 구현

<XML (extensible markup language)>
- 특수한 목적을 갖는 마크업 언어를 만드는 데 사용되는 다목적 마크업 언어
- 웹브라우저 간 HTML 문법이 호환되지 않는 문제와 SGML의 복잡함을 해결하기 위해 개발

SOAP (Simple Object Access Protocol)	네트워크 상에서 HTTP/HTTPS, SMTP등을 이용하여 XML을 교환하기 위한 통신 규약
WSDL (Web Services Description Language)	웹 서비스와 관련된 서식이나 프로토콜 등을 표준적인 방법으로 기술하고 게시하기 위한 언어

4. 서버 프로그램 구현

<개발 환경 구축 - 하드웨어 환경>
- 개발을 위해 개발 프로젝트를 이해하고 소프트웨어 및 하드웨어 장비를 구축하는 것
- 하드웨어 환경 : 클라이언트(사용자와의 인터페이스 역할), 서버(클라이언트와 통신하여 서비스를 제공)
- 서버의 종류

웹 서버 (Web Server)	클라이언트로부터 직접 요청받아 처리
웹 어플리케이션 서버(WAS)	동적 서비스를 제공하거나 인터페이스 역할 수행
데이터베이스 서버	데이터베이스와 이를 관리하는 DBMS 운영
파일 서버	데이터베이스에 저장하기에는 비효율적이나, 서비스 제공을 목적으로 유지하는 파일들을 저장

<소프트웨어 환경>
- 클라이언트나 서버 운영을 위한 시스템 소프트웨어와 개발에 사용되는 개발 소프트웨어로 구성
- 운영체제, 웹서버, WAS 운용을 위한 서버 프로그램, DBMS 등

요구사항 관리 도구	요구사항 수집, 분석, 추적 등을 편리하게 도와주는 소프트웨어
설계/모델링 도구	UML 지원, 개발의 전 과정에서 설계 및 모델링을 도와주는 소프트웨어
구현 도구	개발 언어를 통해 어플리케이션의 실제 구현을 지원하는 소프트웨어
빌드 도구	구현 도구를 통해 작성된 소스의 빌드, 배포, 라이브러리 관리를 지원하는 소프트웨어
테스트 도구	모듈들이 요구사항에 적합하게 구현되었는지 테스트하는 소프트웨어
형상 관리 도구	산출물들을 버전별로 관리하여 품질 향상을 지원하는 소프트웨어

<소프트웨어 아키텍처>

- 소프트웨어를 구성하는 요소들 간의 관계를 표현하는 시스템의 구조 또는 구조체

모듈화	시스템의 기능들을 모듈 단위로 나누는 것
추상화	전체적이고 포괄적인 개념을 설계 후 차례로 세분화하여 구체화시켜 나가는 것
단계적 분해	상위의 중요 개념으로부터 하위의 개념으로 구체화시키는 분할 기법
정보 은닉	모듈 내부에 포함된 절차와 자료들의 정보가 감추어져 다른 모듈이 접근하거나 변경하지 못하도록 하는 기법

<아키텍처 패턴>

- 아키텍처를 설계할 때 참조할 수 있는 전형적인 해결 방식 또는 예제
- 레이어 패턴 : 시스템을 계층으로 구분하여 구성하는 패턴
- 클라이언트-서버 패턴 : 하나의 서버 컴포넌트와 다수의 클라이언트 컴포넌트로 구성되는 패턴
- 파이프-필터 패턴 : 데이터 스트림 절차의 각 단계를 필터로 캡슐화하여 파이프를 통해 전송하는 패턴
- 모델-뷰-컨트롤러 패턴 : 서브시스템을 모델, 뷰, 컨트롤러로 구조화하는 패턴
- 마스터-슬레이브 패턴, 브로커 패턴, 피어-투-피어 패턴, 이벤트-버스 패턴, 블랙보드 패턴, 인터프리티 패턴

<객체지향>

- 각 요소들을 객체로 만든 후 객체들을 조립해서 소프트웨어를 개발하는 기법

객체	데이터와 이를 처리하기 위한 함수를 묶어놓은 소프트웨어 모듈
클래스	공통된 속성과 연산을 갖는 객체의 집합
메시지	객체들 간 상호작용이 사용되는 수단, 객체의 동작이나 연산을 일으키는 외부의 요구사항
캡슐화	외부에서의 접근을 제한하기 위해 인터페이스를 제외한 세부 내용을 은닉
상속	상위 클래스의 모든 속성과 연산을 하위 클래스가 물려받는 것
다형성	하나의 메시지에 대해 각각의 객체가 고유한 방법으로 응답할 수 있는 능력
연관성	두 개 이상의 객체들이 상호 참조하는 관계

<객체지향 분석>

- 사용자의 요구사항과 관련된 객체, 속성, 연산, 관계 등을 정의하여 모델링하는 작업

럼바우 방법	모든 소프트웨어 구성 요소를 그래픽 표기법을 이용하여 모델링 - 객체 모델링 : 속성과 연산 식별 및 객체들 간의 관계를 규정하여 객체 다이어그램으로 표시 - 동적 모델링 : 상태 다이어그램을 이용하여 시간의 흐름에 따른 객체들 간의 동적인 행위 표현 - 기능 모델링 : 자료 흐름도(DFD)를 이용하여 다수의 프로세스들 간의 자료 흐름을 중심으로 처리 과정 표현
부치 방법	미시적 개발 프로세스와 거시적 개발 프로세스를 모두 사용
Jocobson 방법	유스케이스를 강조하여 사용
Coad, Yourdon 방법	E-R 다이어그램을 사용하여 객체의 행위를 모델링
Wirfs-Brock 방법	분석과 설계 간 구분이 없고, 고객 명세서를 평가해서 설계 작업까지 연속으로 수행

<모듈>

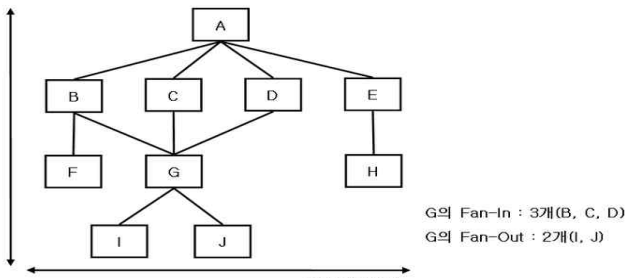
- 모듈화를 통해 분리된 시스템의 각 기능
- 모듈화 : 소프트웨어의 성능을 향상시키거나 시스템 수정, 재사용, 유지관리 등이 용이하도록 시스템 기능들을 모듈 단위로 분해, 모듈 간 결합도의 최소화, 응집도의 최대화가 목표
- 결합도 : 모듈 간 상호 의존하는 정도

내용 결합도	한 모듈이 다른 모듈의 내부 기능 및 그 내부 자료를 직접 참조하거나 수정할 때의 결합도
공통 결합도	공유되는 공통 데이터 영역을 여러 모듈이 사용할 때의 결합도
외부 결합도	어떤 모듈에서 선언한 데이터를 외부의 다른 모듈에서 참조할 때의 결합도
제어 결합도	어떤 모듈이 다른 모듈 내부의 논리적 흐름을 제어하기 위해 제어 신호나 제어 요소를 전달하는 결합도
스탬프 결합도	모듈 간의 인터페이스로 배열이나 레코드 등의 자료 구조가 전달될 때의 결합도
자료 결합도	모듈 간의 인터페이스가 자료 요소로만 구성될 때의 결합도

- 응집도 : 모듈의 내부 요소들이 서로 관련되어 있는 정도

기능적 응집도	모듈 내부의 모든 기능 요소들이 단일 문제와 연관되어 수행될 경우
순차적 응집도	모듈 내 하나의 활동으로부터 나온 출력 데이터를 그 다음 활동의 입력 데이터로 사용할 경우
통신적 응집도	동일한 입력과 출력을 사용하여 서로 다른 기능을 수행하는 구성 요소들이 모였을 경우
절차적 응집도	모듈이 다수의 관련 기능을 가질 때 모듈 안의 구성 요소들이 그 기능을 순차적으로 수행할 경우
시간적 응집도	특정 시간에 처리되는 몇 개의 기능을 모아 하나의 모듈로 작성할 경우
논리적 응집도	유사한 성격을 갖거나 특정 형태로 분류되는 처리 요소들로 하나의 모듈이 형성되는 경우
우연적 응집도	모듈 내부의 각 구성 요소들이 서로 관련 없는 요소로만 구성된 경우의 응집도

- N-S 차트 : 논리의 기술에 중점을 두고 도형을 이용해 표현하는 방법
- 팬 인(Fan-In), 팬 아웃(Fan-Out)



<공통 모듈>

- 여러 프로그램에서 공통으로 사용할 수 있는 모듈
- 공통 모듈 명세 기법의 종류

정확성	시스템 구현 시 해당 기능이 필요하다는 것을 알 수 있

	도록 정확히 작성
명확성	해당 기능을 이해할 때 중의적으로 해석되지 않도록 명확하게 작성
완전성	시스템 구현을 위해 필요한 모든 것을 기술
일관성	공통 기능들 간 상호 충돌이 발생하지 않도록 작성
추적성	기능에 대한 요구사항의 출처, 관련 시스템 등의 관계를 파악할 수 있도록 작성

<재사용>

- 이미 개발된 기능들을 새로운 시스템이나 기능 개발에 사용하기 적합하도록 최적화하는 작업

함수와 객체	클래스나 메소드 단위의 소스코드를 재사용
컴포넌트	컴포넌트 지체에 대한 수정 없이 인터페이스를 통해 통신하는 방식으로 재사용
애플리케이션	공통된 기능들을 제공하는 애플리케이션을 공유하는 방식으로 재사용

<코드>

- 자료의 분류, 조합, 집계, 추출을 용이하게 하기 위해 사용하는 기호

식별 기능	데이터 간 성격에 따라 구분이 가능
분류 기능	특정 기준이나 동일한 유형에 해당하는 데이터를 그룹화할 수 있음
배열 기능	의미를 부여하여 나열할 수 있음
표준화 기능	다양한 데이터를 기준에 맞추어 표현할 수 있음
간소화 기능	복잡한 데이터를 간소화할 수 있음

순차 코드	최초의 자료부터 차례로 일련번호를 부여하는 방법, 순서코드, 일련번호 코드
블록 코드	코드화 대상 항목 중에서 공통성이 있는 것끼리 블록으로 구분, 각 블록 내애서 일련번호 부여
10진 코드	코드화 대상 항목을 10진 분할(도서분류식)
그룹 분류 코드	코드화 대상 항목을 일정 기준에 따라 대, 중, 소분류 등으로 구분하여 각 그룹 안에서 일련번호 부여
연상 코드	코드화 대상 항목의 명칭이나 관계 있는 기호를 이용하여 코드 부여(TV-40 : 40인치)
표의 숫자 코드	코드화 대상 항목의 성질을 그대로 코드에 적용 유효숫자코드(120-720-1500)
합성 코드	필요한 기능을 하나의 코드로 수행하기 어려운 경우 2개 이상의 코드를 조합

<디자인 패턴>

- 모듈 간의 관계 및 인터페이스를 설계할 때 참조할 수 있는 전형적인 해결 방식 또는 예제
- 생성 패턴

추상 팩토리	구체적인 클래스에 의존하지 않고 인터페이스를 통해 연관, 의존하는 객체들의 그룹으로 생성하여 추상적으로 표현
빌더	작게 분리된 인스턴스를 건축하듯이 조합하여 객체를 생성
팩토리 메소드	객체 생성을 서브 클래스에서 처리하도록 분리하여 캡슐화한 패턴
프로토타입	원본 객체를 복제하는 방법으로 객체를 생성하는 패턴
싱글톤	하나의 객체를 생성하면 생성된 객체를 어디서든 참조할 수 있지만, 여러 프로세스가 동시에 참조할 수는 없음

- 구조 패턴

어댑터	호환성이 없는 클래스들의 인터페이스를 다른 클래스가 이용할 수 있도록 변환해주는 패턴
브리지	구현부에서 추상층을 분리하여, 서로가 독립적으로 확장할 수 있도록 구성
컴포지트	여러 객체를 가진 복합 객체와 단일 객체를 구분없이 다루고자 할 때 사용하는 패턴
데코레이터	객체 간 결합을 통해 능동적으로 기능들을 확장할 수 있는 패턴
퍼싸드	복잡한 서브 클래스들을 피해 더 상위에 인터페이스 구성
플라이웨이트	인스턴스가 필요할 때마다 매번 생성하는 것이 아니고 가능한 한 공유해서 사용함으로써 메모리를 절약하는 패턴
프록시	접근이 어려운 객체와 여기에 연결하려는 객체 사이에서 인터페이스 역할을 수행하는 패턴

- 행위 패턴

책임 연쇄	요청을 처리할 수 있는 객체가 둘 이상 존재하여 한 객체가 처리하지 못하면 다음 객체로 넘어가는 형태의 패턴
커맨드	요청을 객체의 형태로 캡슐화하여 재이용하거나 취소할 수 있도록 요청에 필요한 정보를 젖아하거나 로그에 남기는 패턴
인터프리터	언어에 문법 표현을 정의하는 패턴
반복자	자료 구조와 같이 접근이 잦은 객체에 대해 동일한 인터페이스를 사용하도록 하는 패턴
중재자	수많은 객체들 간의 복잡한 상호작용을 캡슐화하여 객체로 정의하는 패턴
메멘토	특정 시점에서의 객체 내부 상태를 객체화함으로써 이후 요청에 따라 객체를 해당 시점의 상태로 돌릴 수 있는 기능을 제공
옵서버	한 객체의 상태가 변화하면 객체에 상속되어 있는 다른 객체들에게 변화된 상태를 전달하는 패턴
상태	객체의 상태에 따라 동일한 동작을 다르게 처리해야 할 때 사용
젼략	동일 계열의 알고리즘들을 개별적으로 캡슐화하여 상호 교환할 수 있게 정의하는 패턴
템플릿 메소드	상위 클래스에서 골격을 정의하고 하위 클래스에서 세부 처리를 구체화하는 구조의 패턴
방문자	각 클래스들의 데이터 구조에서 처리 기능을 분리하여 별도의 클래스로 구성하는 패턴

<배치 프로그램>

- 배치 프로그램 : 여러 작업들을 미리 정해진 일련의 순서에 따라 일괄적으로 처리하도록 만든 프로그램

대용량 데이터	대량의 데이터 처리가 가능해야함
자동화	사용자의 개입 없이 수행되어야함(심각한 오류 상황 제외)
견고성	잘못된 데이터나 데이터 중복 등의 상황으로 중단되는 일 없이 수행되어야함
안전성/신뢰성	오류가 발생하면 오류 발생 위치, 시간 등을 추적할 수 있어야함
성능	다른 응용 프로그램의 수행을 방해하지 않아야함 지정된 시간 내 처리가 완료되어야함

- 배치 스케줄러 : 일괄 처리 작업이 설정된 주기에 맞춰 자동으로 수행되도록 지원해주는 도구

스프링 배치	- Spring Source, Accenture 사가 공동 개발한 오픈 소스 프레임워크 - 로그 관리, 추적, 트랜잭션 관리, 작업 처리 통계, 작업 재시작 등의 기능
Quartz	스프링 프레임워크로 개발하는 응용 프로그램들의 일괄 처리를 위한 다양한 기능을 제공하는 오픈 소스 라이브러리, 일괄 처리 작업에 유연성 제공
Cron	리눅스의 기본 스케줄러 도구

	crontab 명령어를 통해 작업을 예약

5. 인터페이스 구현

<인터페이스 요구사항>

- 인터페이스 요구사항 검증 방법

동료 검토	요구사항 명세서 작성자가 명세서 내용을 직접 설명하고 동료들이 이를 들으면서 결함을 발견하는 형태
워크스루	검토 회의 전에 요구사항 명세서를 미리 배포하여 사전 검토한 후에 짧은 검토 회의를 통해 결함 발견
인스펙션	요구사항 명세서 작성자를 제외한 다른 검토 전문가들이 요구사항 명세서를 확인하면서 결함을 발견하는 형태의 검토 방법

- 프로토타이핑, 테스트 설계, CASE 도구 활용
- 주요 항목

완전성	사용자의 모든 요구사항이 누락되지 않고 완전히 반영되어있는지
일관성	요구사항이 모순되거나 충돌되는 점 없이 일관성을 유지하는지
명확성	모든 참여자가 요구사항을 명확히 이해할 수 있는지
기능성	요구사항이 무엇에 중점을 두고 있는지
검증 가능성	요구사항이 사용자의 요구를 모두 만족하고 개발된 소프트웨어가 사용자의 요구 내용과 일치하는지
추적 가능성	요구사항 명세서와 설계서를 추적할 수 있는지
변경 용이성	요구사항 명세서의 변경이 쉽도록 작성되었는지

<미들웨어 솔루션>

- 미들웨어 : 운영체제와 응용 프로그램 사이에서 다양한 서비스를 제공하는 소프트웨어

DB	클라이언트에서 원격의 데이터베이스와 연결하는 미들웨어
RPC	원격 프로시저를 로컬 프로시저처럼 호출하는 미들웨어
MOM	비동기형 메시지를 전달하는 미들웨어
TP-Monitor	트랜잭션을 처리 및 감시하는 미들웨어
ORB	코바 표준 스펙을 구현한 객체 지향 미들웨어
WAS (웹 어플리케이션 서버)	사용자 요구에 따라 변하는 동적인 컨텐츠를 처리하기 위한 미들웨어

<모듈 연계를 위한 인터페이스 기능 식별>

- 모듈 연계 : 내부 모듈과 외부 모듈 또는 내부 모듈 간 데이터 교환을 위해 관계를 설정하는 것
- EAI : 기업 내 각종 애플리케이션 및 플랫폼 간의 상호 연동이 가능하게 해주는 솔루션

Point-to-Point	가장 기본적인 애플리케이션 통합 방식 애플리케이션을 1:1로 연결, 변경 및 재사용이 어려움
Hub&Spoke	허브 시스템을 통해 데이터를 전송하는 중앙 집중형 방식 확장 및 유지보수 용이, 허브 장애 발생 시 시스템 전체에 영향
Message Bus	애프리케이션 사이에 미들웨어를 두어 처리하는 방식 확장성이 뛰어나며 대용량 처리 가능
Hybrid	Hub&Spoke, Message Bus의 혼합 데이터 병목 현상 최소화 가능

- ESB : 애플리케이션 간 표준 기반의 인터페이스를 제공하는 솔루션
- 웹 서비스 : 네트워크의 정보를 표준화된 서비스 형태로 만들어 공유하는 기술

SOAP	HTTP, HTTPS, SMTP 등을 활용하여 XML 기반의 메시지를 네트워크 상에서 교환하는 프로토콜
UDDI	WSDL을 등록하여 서비스와 서비스 제공자를 검색하고 접근하는데 사용
WSDL	웹 서비스명, 서비스 제공 위치, 프로토콜 등 웹 서비스에 대한 상세 정보를 XML 형식으로 구현

<인터페이스 구현>

- 송수신 시스템 간의 데이터 교환 및 처리를 실현해주는 작업
- JSON(JavaScript Object Notation) : 데이터 객체를 속성, 값의 쌍 형태로 표현하는 개방형 표준 포맷
- AJAX(Asynchronous JavaScript and XML) : 클라이언트와 서버 간 XML 데이터를 주고받는 비동기 통신 기술

<인터페이스 보안>

- 인터페이스의 보안 취약점을 분석한 후 적절한 보안 기능을 적용하는 것

네트워크 영역	네트워크 트래픽에 대한 암호화 설정 IPSec, SSL, S-HTTP
애플리케이션 영역	소프트웨어 개발 보안 가이드를 참조하여 애플리케이션 코드 상의 보안 취약점을 보완하는 방향으로 애플리케이션 보안 기능 적용
데이터베이스 영역	데이터베이스, 스키마, 엔티티의 접근 권한과 프로시저, 트리거 등 데이터베이스 동작 객체의 보안 취약점에 보안 기능을 적용 개인 정보나 업무상 민감한 데이터의 경우 암호화나 익명화 등 데이터 자체 보안 방안도 고려

<인터페이스 구현 검증>

- 인터페이스 구현 검증 : 인터페이스가 정상적으로 문제 없이 작동하는지 확인하는 것

xUnit	다양한 언어를 지원하는 단위 테스트 프레임워크
STAF	서비스 호출 및 컴포넌트 재사용 등 다양한 환경을 지원하는 테스트 프레임워크
FitNesse	웹 기반 테스트 케이스 설계, 실행, 결과 확인 등을 지원하는 테스트 프레임워크
NTAF	FitNesse의 장점인 협업 기능과 STAF의 장점인 재사용 및 확장성을 통합한 NHN의 테스트 자동화 프레임워크
Selenium	다양한 브라우저 및 개발 언어를 지원하는 웹 애플리케이션 테스트 프레임워크
watir	Ruby를 사용하는 애플리케이션 테스트 프레임워크

6. 화면 설계

<사용자 인터페이스>

- 사용자 인터페이스 : 사용자와 시스템 간 상호작용이 원활하게 이루어지도록 도와주는 장치나 소프트웨어

CLI(Command Line Interface)	명령과 출력이 텍스트 형태로 이루어지는 인터페이스

GUI(Graphical User Interface)	아이콘이나 메뉴를 마우스로 선택하여 작업을 수행하는 그래픽 환경의 인터페이스
NUI(Natural User Interface)	사용자의 말이나 행동으로 기기를 조작하는 인터페이스

- 사용자 인터페이스의 기본 원칙

직관성	누구나 쉽게 이해하고 사용할 수 있어야 함
유효성	사용자의 목적을 정확하고 완벽하게 달성해야함
학습성	누구나 쉽게 배우고 익힐 수 있어야 함
유연성	사용자의 요구사항을 최대한 수용하고 실수를 최소화해야 함

<UI 설계 도구>

와이어프레임 (Wireframe)	페이지에 대한 개략적인 레이아웃이나 UI 요소 등에 대한 뼈대를 설계하는 도구
목업 (Mockup)	와이어프레임보다 좀 더 실제 화면과 유사하게 만든 정적인 형태의 모형
스토리보드 (Story Board)	와이어프레임에 콘텐츠에 대한 설명, 페이지 간 이동 흐름 등을 추가한 문서
프로토타입 (Prototype)	와이어프레임, 스토리보드 등에 인터랙션을 적용함으로써 실제 구현하는 것처럼 테스트가 가능한 동적인 형태의 모형 - 페이퍼 프로토타입 : 손으로 직접 작성하는 아날로그적인 방법으로, 제작 기간이 짧은 경우, 비용이 적을 경우, 업무 협의가 빠를 경우 사용 - 디지털 프로토타입 : 프로그램을 사용하여 작성하는 방법으로 재사용이 필요한 경우, 산출물과 비슷한 효과가 필요한 경우, 숙련된 전문가가 있을 경우 사용
유스케이스 (Usecase)	사용자의 요구사항을 기능 단위로 표현하는 것

<품질 요구사항>

- 품질 요구사항 : 소프트웨어에 대한 요구사항이 사용자의 입장에서 얼마나 충족하는가를 나타내는 소프트웨어 특성의 총체

ISO/IEC 9126	소프트웨어의 품질 특성과 평가를 위한 국제 표준
ISO/IEC 25010	ISO/IEC 9126에 호환성과 보안성 강화하여 개정
ISO/IEC 12119	패키지 소프트웨어의 일반적인 제품 품질 요구사항 및 테스트를 위한 국제표준
ISO/IEC 14598	소프트웨어 품질의 측정과 평가에 필요 절차를 구정한 표준

- ISO/IEC 9126 소프트웨어 품질 특성

기능성 (Functionality)	소프트웨어가 사용자의 요구사항을 정확하게 만족하는 기능을 제공하는지
신뢰성 (Reliability)	주어진 시간동안 주어진 기능을 오류 없이 수행할 수 있는 정도
사용성 (Usability)	사용자와 컴퓨터 사이에 발생하는 어떠한 행위에 대해 사용자가 정확하게 이해하고 사용하며, 향후 다시 사용하고 싶은 정도
효율성 (Efficiency)	사용자가 요구하는 기능을 얼마나 빠르게 처리할 수 있는지
유지보수성 (Maintainability)	환경의 변화 또는 새로운 요구사항이 발생했을 때 소프트웨어를 개선하거나 확장할 수 있는 정도
이식성 (Portability)	소프트웨어가 다른 환경에서도 얼마나 쉽게 적용할 수 있는지 정도

<HCI/UX/감성공학>

- HCI : 사람이 시스템을 보다 편리하고 안전하게 사용할 수 있도록 연구하고 개발하는 학문
- UX(User Experience) : 사용자가 시스템이나 서비스를 이용하면서 느끼고 생각하게 되는 총체적인 경험

주관성 (Subjectivity)	사람들이 개인적, 신체적, 인지적 특성에 따라 다르므로 주관적임
정황성 (Contextuality)	경험이 일어나는 상황 또는 주변 환경에 영향
총체성(Holistic)	개인이 느끼는 총체적인 심리적, 감성적 결과

- 감성공학 : 제품이나 작업 환경을 사용자의 감성에 알맞도록 설계 및 제작하는 기술

7. 애플리케이션 테스트 관리

<애플리케이션 테스트>

- 애플리케이션 테스트 : 애플리케이션에 잠재되어 있는 결함을 찾아내는 일련의 행위 또는 절차
- 애플리케이션 테스트의 기본 원리

완벽한 테스트 불가능	소프트웨어의 잠재적인 결함을 줄일 수 있지만 소프트웨어에 결함이 없다고 증명할 수는 없음
파레토 법칙	애플리케이션의 20%에 해당하는 코드에서 전체 결함의 80%가 발견된다는 법칙
살충제 패러독스	동일한 테스트 케이스로 동일한 테스트를 반복하면 더 이상 결함이 발견되지 않는 현상
테스팅은 정황 의존	소프트웨어의 특징, 테스트 환경, 테스터의 역량 등 정황에 따라 테스트 결과가 달라질 수 있으므로 정황에 따라 테스트를 다르게 수행해야 함
오류-부재의 궤변	소프트웨어의 결함을 모두 제거해도 사용자의 요구사항을 만족시키지 못하면 해당 소프트웨어는 품질이 높다고 말할 수 없는 것
테스트와 위험은 반비례	테스트를 많이 할수록 미래에 발생할 위험을 줄일 수 있음
테스트의 점진적 확대	테스트는 작은 부분에서 시작하여 점점 확대하며 진행해야함
테스트의 별도 팀 수행	테스트는 개발자와 관계없는 별도의 팀에서 수행해야 함

<애플리케이션 테스트의 분류>

- 프로그램 실행 여부에 따른 테스트

정적 테스트	- 프로그램을 실행하지 않고 명세서나 소스 코드를 대상으로 분석하는 테스트 - 워크스루, 인스펙션, 코드검사 등
동적 테스트	- 프로그램을 실행하여 오류를 찾는 테스트 - 소프트웨어 개발의 모든 단계에서 테스트 수행 - 블랙박스 테스트, 화이트박스 테스트

- 테스트 기반에 따른 테스트

명세 기반 테스트	- 사용자의 요구사항에 대한 명세를 빠짐없이 테스트 케이스로 만들어 구현하고 있는지 확인하는 테스트 - 동등분할, 경계 값 분석 등
구조 기반 테스트	- 소프트웨어 내부의 논리 흐름에 따라 테스트 케이스를 작성하고 확인하는 테스트 - 구문 기반, 결정 기반, 조건 기반 등
경험 기반 테스트	- 유사 소프트웨어나 기술 등에 대한 테스터의 경험을 기반으로 수행하는 테스트 - 사용자의 요구사항에 대한 명세가 불충분하거나 테스트 시간에 제약이 있는 경우 수행하면 효과적

	- 에러 추정, 체크 리스트, 탐색적 테스팅

- 시각에 따른 테스트

검증(Verification) 테스트	- 개발자의 시각에서 제품의 생산 과정을 테스트하는 것 - 제품이 명세서대로 완성되었는지를 테스트
확인(Validation) 테스트	- 사용자의 시각에서 생산된 제품의 결과를 테스트 - 사용자가 요구한대로 제품이 완성됐는지, 제품이 정상적으로 동작하는지 테스트

- 목적에 따른 테스트

회복 테스트 Recovery	시스템에 여러 가지 결함을 주어 실패하도록 한 후 올바르게 복구되는지를 확인하는 테스트
안전 테스트 Security	시스템에 설치된 시스템 보호 도구가 불법적인 침입으로부터 시스템을 보호할 수 있는지를 확인하는 테스트
강도 테스트 Stress	시스템에 과도한 정보량이나 빈도 등을 부과하여 과부화 시에도 소프트웨어가 정상적으로 실행되는지를 확인하는 테스트
성능 테스트 Performance	소프트웨어의 실시간 성능이나 전체적인 효율성을 진단하는 테스트, 소프트웨어의 응답시간, 처리량 등 테스트
구조 테스트 Structure	소프트웨어 내부의 논리적 경로, 소스코드와 복잡도 등을 평가하는 테스트
회귀 테스트 Regression	소프트웨어의 변경 또는 수정된 코드에 새로운 결함이 없음을 확인하는 테스트
병행 테스트 Parallel	변경된 소프트웨어와 기존 소프트웨어에 동일한 데이터를 입력하여 결과를 비교하는 테스트

<테스트 기법에 따른 애플리케이션 테스트>

- 화이트박스 테스트 : 원시 코드의 논리적인 모든 경로를 테스트하여 테스트 케이스를 설계하는 방법, 모듈 안의 작동 직접 관찰
- 화이트박스 테스트의 종류

기초 경로 검사	- 대표적인 화이트 테스트 기법 - 테스트 케이스 설계자가 절차적 설계의 논리적 복잡성을 측정할 수 있게 해주는 테스트 기법
제어 구조 검사	- 조건 검사 : 프로그램 모듈 내에 있는 논리적 조건을 테스트하는 테스트 케이스 설계 기법 - 루프 검사 : 프로그램의 반복 구조에 초점을 맞춰 실시하는 테스트 케이스 설계 기법 - 데이터 흐름 검사 : 프로그램에서 변수의 정의와 변수 사용의 위치에 초점을 맞춰 실시하는 테스트 케이스 설계 기법

- 화이트박스 테스트의 검증 기준

문장 검증 기준	소스 코드의 모든 구문이 한 번 이상 수행되도록 테스트 케이스를 설계
분기 검증 기준	소스 코드의 모든 조건문이 한 번 이상 수행되도록 테스트 케이스를 설계
조건 검증 기준	소스 코드의 모든 조건문에 대해 조건이 True인 경우와 False인 경우가 한 번 이상 수행되도록 테스트 케이스를 설계
분기/조건 기준	소스 코드의 모든 조건문과 각 조건문에 포함된 개별 조건식의 결과가 True인 경우와 False인 경우가 한 번 이상 수행되도록 테스트 케이스를 설계

- 블랙박스 테스트 : 소프트웨어가 수행할 특정 기능을 알기 위해 각 기능이 완전히 작동되는 것을 입증하는 테스트, 사용자의 요구사항 명세를 보며 테스트, 주로 구현된 기능을 테스트

동치 분할 검사	프로그램 입력 조건에 타당한 입력 자료와 타당하지 않은 입력 자료의 개수를 균등하게 하여 테스트 케이스를 정하고 해당 입력 자료에 맞는 결과가 출력되는지 확인하는 기법
경계값 분석	입력 조건의 중간값보다 경계값에서 오류가 발생할 확률이 높다는 점을 이용하여 입력 조건의 경계값을 테스트 케이스로 선정하여 검사
원인-효과 그래프 검사	입력 데이터 간의 관계와 출력에 영향을 미치는 상황을 체계적으로 분석한 다음 효용성이 높은 테스트 케이스를 선정하여 검사
오류 예측 검사	과거의 경험이나 확인자의 감각으로 테스트하는 기법
비교 검사	여러 버전의 프로그램에 동일한 테스트 자료를 제공하여 동일한 결과가 출력되는지 테스트

<개발 단계에 따른 애플리케이션 테스트>

단위 테스트	- 코딩 직후 소프트웨어 설계의 최소 단위인 모듈이나 컴포넌트에 초점을 맞춰 테스트 - 구조 기반 테스트(화이트박스), 명세 기반 테스트(블랙박스)
통합 테스트	단위 테스트가 완료된 모듈을 결합하여 하나의 시스템으로 완성시키는 과정에서의 테스트
시스템 테스트	개발된 소프트웨어가 해당 컴퓨터 시스템에서 완벽하게 수행되는가를 점검
인수 테스트	- 개발한 소프트웨어가 사용자의 요구사항을 충족하는지에 중점을 두고 테스트 - 사용자 인수 테스트 : 사용자가 시스템 사용 적절성 여부 확인 - 운영상의 인수 테스트 : 시스템 관리자가 시스템 인수 시 수행하는 테스트 - 계약 인수 테스트 : 계약 상의 인수/검수 조건을 준수하는지 여부 확인 - 규정 인수 테스트 : 소프트웨어가 정부 지침, 법규, 규정 등 규정에 맞게 개발되었는지 확인 - 알파 테스트 : 개발자의 장소에서 사용자가 개발자 앞에서 행하는 테스트 기법(통제된 환경, 문제점을 사용자와 개발자가 함께 확인) - 베타 테스트 : 선정된 최종 사용자가 여러 명의 사용자 앞에서 행하는 테스트 기법(실업무를 가지고 사용자가 직접 테스트)

<통합 테스트>

- 통합 테스트 : 단위 테스트가 끝난 모듈을 통합하는 과정에서 발생하는 오류 및 결함
- 비점진적 통합 방식 : 단계적으로 통합하는 절차 없이 모든 모듈이 미리 결합되어있는 프로그램 전체를 테스트하는 방법 (빅뱅 통합 테스트)
- 점진적 통합 방식 : 모듈 단위로 단계적으로 통합하면서 테스트하는 방법 (하향식 통합 테스트, 상향식 통합 테스트, 혼합식 통합 테스트)

하향식 통합 테스트	상위 모듈에서 하위 모듈 방향으로 통합하면서 테스트
상향식 통합 테스트	하위 모듈에서 상위 모듈 방향으로 통합하면서 테스트
혼합식 통합 테스트	하위 수준에서는 상향식 통합, 상위 수준에서는 하향식 통합을 사용하여 최적의 테스트 지원

- 회귀 테스팅 : 통합 테스트로 인해 변경된 모듈이나 컴포넌트에 새로운 오류가 있는지 확인

<테스트 케이스/테스트 시나리오/테스트 오라클>

- 테스트 케이스 : 사용자의 요구사항을 정확하게 준수했는지를 확인하기 위해 설계된 입력 값, 실행 조건, 기대 결과

등으로 구성된 테스트 항목에 대한 명세서
- 테스트 시나리오 : 테스트 케이스를 적용하는 순서에 따라 여러 개의 테스트 케이스를 묶은 집합
- 테스트 오라클 : 테스트 결과가 올바른지 판단하기 위해 사전에 정의된 참값을 대입하여 비교하는 기법 (특징 : 제한된 검증, 수학적 기법, 자동화 가능)

참(True) 오라클	- 모든 케이스의 입력 값에 대해 기대하는 결과 제공 - 발생된 모든 오류 검출 가능
샘플링 (Sampling) 오라클	- 특정 몇몇 케이스의 입력 값들에 대해서만 기대하는 결과를 제공하는 오라클 - 전수 테스트가 불가능한 경우 사용
추정(Heuristic) 오라클	특정 테스트 케이스의 입력 값에 대해 기대하는 결과를 제공하고, 나머지 입력 값들에 대해서는 추정으로 처리
일관성 검사 (Consistent) 오라클	애플리케이션에 변경이 있을 때 테스트 케이스의 수행 전과 후의 결과 값이 동일한지 확인

<테스트 자동화 도구>
- 테스트 자동화 : 사람이 반복적으로 수행하던 테스트 절차를 스크립트 형태로 구현하는 자동화 도구를 적용함으로써 쉽고 효율적으로 테스트를 수행할 수 있도록 한 것
- 정적 분석 도구 : 프로그램을 실행하지 않고 분석하는 도구
- 테스트 실행 도구 : 스크립트 언어를 사용하여 테스트를 실행하는 도구 (데이터 주도 접근 방식, 키워드 주도 접근 방식)
- 성능 테스트 도구 : 가상의 사용자를 만들어 테스트를 수행함으로써 성능의 목표 달성 여부를 확인하는 도구
- 테스트 통제 도구 : 테스트 계획 및 관리, 테스트 수행, 결함 관리 등을 수행하는 도구
- 테스트 하네스 도구 : 테스트가 실행될 환경을 시뮬레이션하여 컴포넌트 및 모듈이 정상적으로 테스트 되도록 하는 도구

테스트 드라이버	테스트 대상의 하위 모듈 호출, 파라미터 전달, 모듈 테스트 수행 후의 결과 도출
테스트 스텁	제어 모듈이 호출하는 타 모듈의 기능을 단순히 수행하는 도구, 일시적으로 필요한 조건만을 가지고 있는 테스트용 모듈
테스트 슈트	테스트 대상 컴포넌트나 모듈, 시스템에 사용되는 테스트 케이스의 집합
테스트 케이스	사용자 요구사항을 정확하게 준수했는지 확인하기 위한 입력값, 실행 조건, 기대 결과 등으로 만들어진 테스트 항목 명세서
테스트 스크립트	자동화된 테스트 실행 절차에 대한 명세서
목 오브젝트	사전에 사용자의 행위를 조건부로 입력해두면 그 상황에 맞는 예정된 행위를 수행하는 객체

<애플리케이션 성능 분석>
- 애플리케이션 성능 : 최소한의 자원을 사용하여 최대한 많은 기능을 신속하게 처리하는 정도

처리량 (Throughput)	일정 시간 내에 애플리케이션이 처리하는 일의 양
응답시간 (Response Time)	애플리케이션이 요청을 전달한 시간부터 응답이 도착할 때 까지 걸린 시간
경과 시간 (Turn Around Time)	애플리케이션에 작업을 의뢰한 시간부터 처리가 완료될 때 까지 걸린 시간
자원 사용률 (Resource Usage)	애플리케이션이 의뢰한 작업을 처리하는 동안의 CPU 사용량, 메모리 사용량, 네트
	워크 사용량 등 자원 사용률

<복잡도>
- 복잡도 : 시스템이나 시스템 구성 요소 또는 소프트웨어의 복잡한 정도를 나타내는 말
- 시간 복잡도 : 알고리즘을 수행하기 위해 프로세스가 수행하는 연산 횟수를 수치화한 것

빅오 표기법 (Big-O)	알고리즘의 실행 시간이 최악일 때
세타 표기법 (Big-θ)	알고리즘의 실행 시간이 평균일 때
오메가 표기법 (Big-Ω)	알고리즘의 실행 시간이 최상일 때

- 순환 복잡도 : 한 프로그램의 논리적인 복잡도를 측정하기 위한 소프트웨어의 척도 (맥 케이브 순환도, 맥 케이브 복잡도 메트릭스)

<애플리케이션 성능 개선>
- 소스 코드 최적화 : 나쁜 코드(프로그램의 로직이 복잡하고 이해하기 어려운 코드)를 배제하고 클린 코드(누구나 쉽게 이해하고 수정 및 추가할 수 있는 단순 명료한 코드)로 작성하는 것
- 클린 코드 작성 원칙

가독성	- 누구든지 코드를 쉽게 읽을 수 있도록 작성 - 이해하기 쉬운 용어, 들여쓰기 등
단순성	- 코드를 간단하게 작성 - 한번에 한 가지를 처리하도록 작성, 클래스/메소드/함수 등 최소 단위로 분리
의존성 배제	- 코드가 다른 모듈에 미치는 영향 최소화 - 코드 변경 시 다른 붑누에 영향이 없도록 작성
중복성 최소화	- 코드의 중복 최소화 - 중복된 코드는 삭제하고 공통된 코드 사용
추상화	상위 클래스/메소드/함수에서는 간략하게 애플리케이션의 특성을 나타내고 상세 내용은 하위 클래스/메소드/함수에서 구현

- 소스 코드 최적화 유형 : 클래스 분할 배치(하나의 클래스는 하나의 역할만 수행, 응집도 높이고 크기를 작게 작성), 느슨한 결합(인터페이스 클래스를 이용하여 추상화된 자료구조와 메소드를 구현함으로써 클래스 간 의존성 최소화)
- 소스 코드 품질 분석 도구

정적 분석 도구	작성한 소스 코드를 실행하지 않고 코딩 표준이나 코딩 스타일, 결함 등을 확인
동적 분석 도구	작성한 소스 코드를 실행하여 코드에 존재하는 메모리 누수, 스레드 결함 등 분석

9. 소프트웨어 개발 보안 구축

<Secure SDLC> (SDLC : 소프트웨어 개발 생명주기)
- 보안상 안전한 소프트웨어를 개발하기 위해 SDLC에서 보안 강화를 위한 프로세스를 포함한 것 (요구사항 분석, 설계, 구현, 테스트, 유지보수 등 전 단계에 걸쳐 수행되어야 할 보안 활동)
- 소프트웨어 개발 보안 요소

기밀성 Confidentiality	- 시스템 내의 정보와 자원은 인가된 사용자에게만 접근이 허용 - 정보가 전송 중에 노출되더라도 데이터를 읽을 수 없음
무결성	시스템 내의 정보는 오직 인가된 사용자만

Integrity	수정할 수 있음
가용성 Avaliability	인가받은 사용자는 시스템 내의 정보와 자원을 언제라도 사용할 수 있음
인증 Authentication	- 시스템 내의 정보와 자원을 사용하려는 사용자가 합법적인 사용자인지를 확인하는 모든 행위 - 패스워드, 인증용 카드, 지문 검사 등
부인 방지 NonRepudiation	데이터를 송수신한 자가 송수신 사실을 부인할 수 없도록 송수신 증거 제공

- 시큐어 코딩 : 구현 단계에서 발생할 수 있는 보안 취약점들을 최소화하기 위해 보안 요소들을 고려하며 코딩하는 것
- 세션 통제 : 세션의 연결과 연결로 인해 발생하는 정보를 관리하는 것

<입력 데이터 검증 및 표현>

- 입력 데이터 검증 및 표현의 보안 약점

SQL 삽입 Injection	- 웹 응용 프로그램에 SQL을 삽입하여 내부 DB 서버의 데이터를 유출 및 변조하고, 관리자 인증을 우회하는 보안 약점 - 동적 쿼리에 사용되는 입력 데이터에 예약어 및 특수문자가 입력되지 않게 필터링 되도록 설정하여 방지할 수 있음
경로 조작 및 자원 삽입	- 데이터 입출력 경로를 조작하여 서버 자원을 수정, 삭제할 수 있는 보안 약점 - 사용자 입력값을 식별자로 사용하는 경우, 경로 순회 공격을 막는 필터를 사용하여 방지할 수 있음
크로스사이트 스크립팅 (XSS)	- 웹페이지에 악의적인 스크립트를 삽입하여 방문자들의 정보를 탈취하거나, 비정상적인 기능 수행을 유발하는 보안 약점 - HTML 태그의 사용을 제한하거나 스크립트에 삽입되지 않도록 특수문자들을 다른 문자로 치환함으로써 방지
운영체제 명령어 삽입	- 외부 입력값을 통해 시스템 명령어의 실행을 유도함으로써 권한을 탈취하거나 시스템 장애를 유발하는 보안 약점 - 웹 인터페이스를 통해 시스템 명령아 전달되지 않도록 하고, 외부 입력값을 검증 없이 내부 명령어로 사용하지 않음으로써 방지할 수 있음
위험한 형식 파일 업로드	- 악의적인 명령어가 포함된 스크립트 파일을 업로드함으로써 시스템에 손상을 주거나, 시스템을 제어할 수 있는 보안 약점 - 업로드 되는 파일의 확장자 제한, 파일명의 암호화, 웹사이트와 파일 서버의 경로 분리, 실행 속성을 제거하는 등의 방법으로 방지
신뢰되지 않는 URL 주소로 자동접속 연결	- 입력 값으로 사이트 주소를 받는 경우 이를 조작하여 방문자를 피싱 사이트로 유도 - 연결되는 외부 사이트의 주소를 화이트 리스크로 관리함으로써 방지할 수 있음
메모리 버퍼 오버플로	- 연속된 메모리 공간을 사용하는 프로그램에서 할당된 메모리의 범위를 넘어선 위치에서 자료를 읽거나 쓰려고 할 때 발생하는 보안 약점 - 메모리 버퍼를 사용할 경우 적절한 버퍼의 크기를 설정하고 설정된 범위의 메모리 내에서 올바르게 읽거나 쓸 수 있도록 함으로써 방지할 수 있음

<보안 기능>

- 보안 기능 : 소프트웨어 개발의 구현 단계에서 코딩하는 기능인 인증, 접근, 제어, 기밀성, 암호화 등을 올바르게 구현하기 위한 보안 점검 항목

적절한 인증 없이 중요기능 허용	- 보안검사를 우회하여 인증과정 없이 중요한 정보 또는 기능에 접근 및 변경 가능 - 중요 정보나 기능을 수행하는 페이지에서는 재인증 기능을 수행하도록 하여 방지
부적절한 인가	- 접근제어 기능이 없는 실행 경로를 통해 정보 또는 권한을 탈취할 수 있음 - 모든 실행경로에 대해 접근제어 검사를 수행하고, 사용자에게는 반드시 필요한 접근 권한만을 부여하여 방지할 수 있음
중요한 자원에 대한 잘못된 권한 설정	- 권한 설정이 잘못된 자원에 접근하여 해당 자원을 임의로 사용할 수 있다 - 소프트웨어 관리자만 자원들을 읽고 쓸 수 있도록 설정, 인가되지 않은 사용자의 중요 자원에 대한 접근 여부를 검사함으로써 방지할 수 있음
취약한 암호화 알고리즘 사용	- 암호화된 환경설정 파일을 해독하여 비밀번호 등의 중요정보를 탈취할 수 있음 - 안전한 암호화 알고리즘 이용, 안전성이 확인된 암호 모듈을 이용함으로써 방지
중요정보 평문 저장 및 전송	- 암호화되지 않은 평문 데이터를 탈취하여 중보한 정보를 획득할 수 있음 - 중요한 정보를 저장하거나 전송할 때는 반드시 암호화 과정을 거치도록 하고, HTTPS 또는 SSL과 같은 보안 채널을 이용함으로써 방지할 수 있음
하드코드 된 암호화 키	- 암호화된 키도 하드코드 된 경우 유출 시 역계산 또는 무차별 대입 공격에 의해 탈취될 수 있음 - 상수 형태의 암호키를 사용하지 않고 암호화 키 생성 모듈 또는 보안이 보장된 외부 공간을 이용함으로써 방지할 수 있음

<암호 알고리즘>

- 암호 알고리즘 : 중요 정보를 보호하기 위해 평문을 암호화된 문장으로 만드는 절차, 방법
- 개인키 암호화 기법 : 동일 키로 데이터를 암호화하고 복호화하는 암호화 기법 (스트림 암호화/블록 암호화)
- 공개키 암호화 기법 : 암호화할 때 사용하는 공개키는 사용자에게 공개하고, 복호화할 때의 비밀키는 관리자가 비밀리에 관리하는 암호화 기법

SEED	1999년 한국인터넷진흥원에서 개발한 블록 암호화 알고리즘
ARIA	2004년 국가정보원, 산학연합회가 개발한 블록 암호화 알고리즘
DES	1975년 미국 NBS에서 발표한 개인키 암호화 알고리즘
AES	2001년 미국 NIST에서 발표한 개인키 암호화 알고리즘
RSA	1987년 MIT에 의해 제안된 공개키 암호화 알고리즘 큰 숫자를 소인수분해하기 어렵다는 것에 기반하여 만들어짐

- 해시 : 임의의 길이의 입력 데이터나 메시지를 고정된 길이의 값이나 키로 변환하는 것 (MD5 : MD4를 대체하기 위해 고안한 암호화 해시함수)

<서비스 공격 유형>

- 서비스 거부 공격 (DOS) : 대량의 데이터를 한 곳의 서버에 집중적으로 전송함으로써 서버의 정상적인 기능을 방해하는 것
- Ping of Death : 패킷의 크기를 인터넷 프로토콜 허용범위 이상으로 전송하여 네트워크를 마비시키는 서비스 거부 공격 방법
- SMURFING(스머핑) : IP나 ICMP의 특성을 악용하여 엄청난 양의 데이터를 한 사이트에 집중적으로 보냄으로써 네트워크를 불능 상태로 만드는 공격 방법
- SYN Flooding : 3-way-handshake 과정을 의도적으로 중단시킴으로써 서버가 대기 상태에 놓여 정상적인 서비스를 수행하지 못하게 하는 공격 방법
- TearDrop : Offset값을 변경시켜 수신 측에서 과부화를 발생시킴으로써 시스템이 다운되도록 하는 공격 방법

- LAND Attack : 패킷을 전송할 때 송신 IP 주소와 수신 IP 주소를 모두 공격 대상의 IP 주소로 하여 자신에 대해 무한히 응답하게 하는 공격
- DDos(분산 Dos) : 여러 곳에 분산된 공격 지점에서 한 곳의 서버에 대해 분산 서비스 공격을 수행하는 것

Trin00	가장 초기 형태의 데몬, 주로 UDP Flooding 공격 수행
TFN (Tribe Flood Network)	UDP Flooding 뿐 아니라 TCP SYN Flood 공격, ICMP 응답 요청, 스머핑 공격 등 수행
TFN2K	TFN 확장판
Stacheldraht	이전 툴들의 기능을 유지하면서 공격자, 마스터, 에이전트가 쉽게 노출되지 않도록 암호화된 통신 수행

, 디도스 공격을 위한 툴을 데몬이라고 함

- 네트워크 침해 공격 관련 용어 : 스미싱, 스피어 피싱

스미싱 Smishing	문자 메시지를 이용해 사용자의 개인 신용정보를 빼내는 수법
스피어 피싱 Spear Phishing	사회공학의 한 기법, 특정 대상 선정 후 그 대상에게 일반적인 이메일로 위장한 메일을 지속적으로 발송하여 발송 메일의 본문 링크나 첨부된 파일을 클릭하도록 유도해 사용자의 개인 정보 탈취
APT 지능형 지속 위협	조직적으로 특정 기업이나 조직 네트워크에 침투해 활동 거점을 마련한 뒤 때를 기다리면서 보안을 무력화시키고 정보를 수집한 다음 외부로 빼돌리는 형태의 공격
무작위 대입 공격	암호화된 문서의 암호키를 찾아내기 위해 적용 가능한 모든 값을 대입하여 공격하는 방식
큐싱 Qshing	QR코드를 통해 악성 앱의 다운로드를 유도하거나 악성 프로그램을 설치하도록 하는 금융 사기기법
SQL 삽입 공격	전문 스캐너 프로그램 혹은 봇넷 등을 이용해 웹사이트를 무차별적으로 공격하는 과정에서 취약한 사이트가 발견되면 데이터베이스 등의 데이터를 조작하는 일련의 공격 방식
크로스 사이트 스크립팅 (XSS)	- 네트워크를 통한 컴퓨터 보안 공격, 웹 페이지의 내용을 사용자 브라우저에 표현하기 위해 사용되는 스크립트의 취약점을 악용한 해킹기법 - 사용자가 특정 게시물이나 이메일 링크를 클릭하면 악성 스크립트가 실행되어 페이지가 깨지거나 사용자의 컴퓨터에 있는 로그인 정보나 개인정보, 내부 자료 등이 해커에게 전달
스니핑 Sniffing	네트워크의 중간에서 남의 패킷 정보를 도청하는 해킹 유형의 하나, 수동적 공격

- 정보 보안 침해 공격 관련 용어

좀비PC	악성코드에 감염되어 다른 프로그램이나 컴퓨터를 조종하도록 만들어진 컴퓨터, C&C 서버의 제어, 주로 DDos 공격 등에 이용
C&C 서버	해커가 원격지에서 감염된 좀비PC에 명령을 내리고 악성코드를 제어하기 위한 용도로 사용되는 서버
봇넷Botnet	악성 프로그램에 감염되어 악의적인 의도로 사용될 수 있는 다수의 컴퓨터들이 네트워크로 연결된 형태
웜 Worm	네트워크를 통해 연속적으로 자신을 복제하여 시스템의 부하를 높임, 시스템을 다운시키는 바이러스의 일종(DDos, 버퍼오버플로, 슬래머 등)
제로 데이 공격	보안 취약점이 발견되었을 때 발견된 취약점
	의 존재 자체가 공표되기 전에 해당 취약점을 통해 이루어지는 보안 공격, 공격의 신속성을 의미
키로거 공격	컴퓨터 사용자의 키보드 움직임을 탐지해 개인정보를 몰래 빼가는 해킹 공격
랜섬웨어	인터넷 사용자의 컴퓨터에 잠입해 내부 문서나 파일 등을 암호화해 사용자게 열지 못하게 함
백도어	- 시스템 설계자가 서비스 기술자나 유지보수 프로그램 작성자의 액세스 편의를 위해 시스템 보안을 제거하여 만들어놓은 비밀 통로 - 백도어 탐지 방법 : 무결성 검사, 열린 포트 확인, 로그 분석, SetUID 파일 검사 등
트로이 목마	정상적인 기능을 하는 프로그램으로 위장하여 프로그램 내에 숨어 있다가 해당 프로그램이 동작할 때 활성화하여 부작용을 일으키는 것

- 보안 아키텍처 : 무결성, 가용성, 기밀성을 확보하기 위해 보안 요소 및 보안 체계를 식별하고 이들 간 관계를 정의한 구조
- 보안 프레임워크 : 안전한 정보 시스템 환경을 유지하고 보안 수준을 향상시키기 위한 체계
- 로그 : 시스템 사용에 대한 모든 내역을 기록해놓은 것

<보안 솔루션>

- 보안 솔루션 : 외부로부터의 불법적인 침입을 막는 기술 및 시스템

방화벽 Firewall	내부의 네트워크와 인터넷 간에 전송되는 정보를 선별하여 수용, 거부, 수정하는 기능을 가진 침입 차단 시스템
침입 탐지 시스템 (IDS; Intruction Detection System)	컴퓨터 시스템의 비정상적인 사용, 오용, 남용 등을 실시간으로 탐지하는 것 - 오용 탐지 : 미리 입력해 둔 공격 패턴이 감지되면 이를 알려줌 - 이상 탐지 : 평균적인 시스템의 상태를 기준으로 비정상적인 행위나 자원의 사용이 감지되면 이를 알려줌
침입 방지 시스템 (IPS; Intruction Prevention System)	비정상적인 트래픽을 능동적으로 차단하고 격리하는 등의 방어 조취를 취하는 보안 솔루션
데이터 유출 방지(DLP)	내부 정보의 외부 유출을 방지하는 보안 솔루션
웹 방화벽	웹 기반 공격을 방어할 목적으로 만들어진 웹 서버에 특화된 방화벽
VPN(가상 사설 통신망)	공중 네트워크와 암호화 기술을 이용하여 사용자가 마치 자신의 전용 회선을 사용하는 것처럼 해주는 보안 솔루션
NAC(Network Access Control)	네트워크에 접속하는 내부 PC의 MAC 주소를 IP관리 시스템에 등록한 후 일관된 보안 관리 기능을 제공하는 보안솔루션
ESM(Enterprise Security Management)	다양한 장비에서 발생하는 로그 및 보안 이벤트를 통합하여 관리하는 보안 솔루션

11. 응용 SW 기초 기술 활용

<운영체제의 개념>

- 운영체제 : 컴퓨터 시스템의 자원들을 효율적으로 관리하

며, 사용자가 컴퓨터를 편리하고 효과적으로 사용할 수 있도록 환경을 제공하는 여러 프로그램의 모임

처리 능력 Throughput	일정 시간 내 시스템이 처리하는 일의 양
반환 시간 Turn Around Time	시스템에 작업을 의뢰한 시간부터 처리가 완료될 때 까지 걸린 시간
사용 가능도 Availability	시스템을 사용할 필요가 있을 때 즉시 사용 가능한 정도
신뢰도 Reliability	시스템이 주어진 문제를 정확하게 해결하는 정도

<운영체제의 종류>

- Windows : 마이크로소프트 개발
- UNIX : 시분할 시스템, 개방형 시스템, 주로 C언어 사용 → 이식성 높음

커널 Kernel	- 하드웨어를 보호하고 프로그램과 하드웨어 간 인터페이스 역할 담당 - UNIX의 가장 핵심 부분 - 프로세스 관리, 기억장치 관리, 파일 관리, 입출력 관리, 프로세스 간 통신, 데이터 전송 및 변환 등
쉘 Shell	- 사용자 명령어를 인식하여 프로그램 호출, 명령을 수행 - 시스템과 사용자 간 인터페이스 담당
유틸리티 프로그램	- 일반 사용자가 작성한 응용 프로그램을 처리하는데 사용 - DOS에서의 외부 명령어에 해당

- LINUX : 리누스 토발즈가 UNIX 기반으로 개발된 운영체제
- MacOS : 애플사가 UNIX를 기반으로 개발한 운영체제
- Android : 구글사에서 개발한 리눅스 커널 기반의 개방형 모바일 운영체제
- iOS : 애플 사에서 개발한 유닉스 기반의 모바일 운영체제

<기억장치 관리>

- 반입 전략 Fetch : 프로그램이나 데이터를 언제 주기억장치로 적재할 것인지 결정

요구 반입 Demand Fetch	실행중인 프로그램이 특정 프로그램이나 데이터 등의 참조를 요구할 때 적재
예상 반입 Anticipatory Fetch	실행중인 프로그램에 의해 참조될 프로그램이나 데이터를 미리 예상하여 적재하는 방법

- 배치 전략 Placement : 프로그램이나 데이터를 주기억장치의 어디에 위치시킬 것인가를 결정

최초 적합 First Fit	프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 첫 번째 분할 영역에 배치
최적 적합 Best Fit	프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중 단편화를 가장 작게 남기는 분할 영역에 배치
최악 적합 Worst Fit	프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 많이 남기는 분할 영역에 배치

- 교체 전략 Replacement : 이미 사용되고 있는 영역 중에서 어느 영역을 교체하여 사용할 것인지를 결정

<가상 기억장치>

- 가상기억장치 (Virtual Memory) : 보조기억장치의 일부를 주기억장치처럼 사용하는 것

페이징 Paging	- 프로그램과 주기억장치의 영역을 동일한 크기로 나눈 후 주기억장치의 영역에 적재시켜 실행하는 기법 - 외부 단편화는 발생하지 않으나 내부 단편화는 발생할 수 있다
세그먼테이션 Segmentation	- 기억장치에 보관되어 있는 프로그램을 다양한 크기의 논리적인 단위로 나눈 후 주기억장치에 적재시켜 실행시키는 방법 - 내부 단편화는 발생하지 않으나 외부 단편화는 발생할 수 있다

<페이지 교체 알고리즘>

- 페이지 부재가 발생하면 이때 주기억장치의 모든 페이지 프레임이 사용중이면 어떤 페이지 프레임을 선택하여 교체할 것인지를 결정하는 기법

OPT 최적 교체 optimal replacement	앞으로 가장 오랫동안 사용하지 않을 페이지 교체
FIFO first in first out	가장 먼저 들어와서 가장 오래 있었던 페이지 교체
LRU least recently used	chlrsdp 가장 오랫동안 사용하지 않은 페이지를 교체
LFU least frequnetly used	사용 빈도가 가장 적은 페이지를 교체
NUR not used recently	최근에 사용하지 않은 페이지를 교체, 참조비트와 변형비트
SCR second chance replacement	가장 오랫동안 주기억장치에 있던 페이지 중 자주 사용되는 페이지의 교체를 방지

<가상기억장치 기타 관리사항>

- 페이지 크기

페이지 크기가 작을 경우	- 페이지 단편화가 감소되고 한 개의 페이지를 주기억장치로 이동시키는 시간이 줄어듦 - 불필요한 내용이 주기억장치에 적재될 확률이 적어 효율적인 워킹셋을 유지할 수 있음 - 페이지 정보를 갖는 페이지 맵 테이블의 크기가 커지고, 매핑 속도가 늦어짐 - 디스크 접근 횟수가 많아져서 전체적인 입출력 시간은 늘어남
페이지 크기가 클 경우	- 페이지 정보를 갖는 페이지 맵 테이블의 크기가 작아지고 매핑 속도가 빨라짐 - 디스크 접근 횟수가 줄어들어 전체적인 입출력의 효율성이 증가됨 - 페이지 단편화가 증가되고 한 개의 페이지를 주기억장치로 이동시키는 시간이 늘어남

- Locality : 프로세스가 실행되는 동안 주기억장치를 참조할 때 일부 페이지만 집중적으로 참조하는 성질

시간 구역성 Temporal Locality	프로세스가 실행되면서 하나의 페이지를 일정 시간 동안 집중적으로 엑세스
공간 구역성 Spatial Locality	프로세스 실행 시 일정 위치의 페이지를 집중적으로 액세스하는 현상

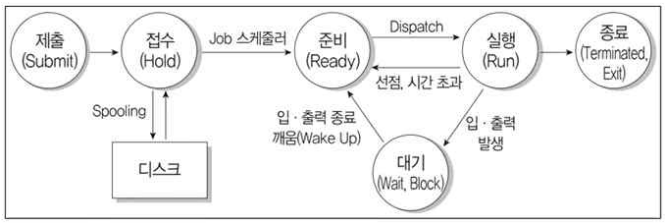
- 워킹셋(Working set) : 프로세스가 일정 시간 동안 자주 참조하는 페이지들의 집합, 자주 참조되는 워킹 셋을 주기억장치에 상주시킴으로써 페이지 부재 및 페이지 교체 현상이 줄어들어 프로세스의 기억장치 사용이 안정됨
- 스래싱(Thrashing) : 프로세스의 처리 시간보다 페이지 교체에 소요되는 시간이 더 많아지는 현상

<프로세스의 개요>

- 프로세스 : 프로세서에 의해 처리되는 사용자 프로그램, 시

스템 프로그램, 실행중인 프로그램

- PCB(Process Control Block, 프로세스 제어 블록) : 운영체제가 프로세스에 대한 중요한 정보를 저장해 놓는 곳
- 프로세스 상태 전이



제출 submit	작업을 처리하기 위해 사용자가 작업을 시스템에 제출한 상태
접수 hold	제출된 작업이 스풀 공간인 디스크의 할당 위치에 저장된 상태
준비 ready	프로세스가 프로세서를 할당받기 위해 기다리고 있는 상태
실행 run	준비상태 큐에 있는 프로세스가 프로세서를 할당받아 실행되는 상태
대기 wait, 블록 block	프로세스에 입출력 처리가 필요하면 현재 실행 중인 프로세스가 중단되고, 입출력 처리가 완료될 때 까지 대기하고 있는 상태
종료 terminated, exit	프로세스의 실행이 끝나고 프로세스 할당이 해제된 상태

Dispatch	준비 상태에서 대기하고 있는 프로세스 중 하나가 프로세서를 할당받아 실행 상태로 전이되는 과정
Wake Up	입출력 작업이 완료되어 프로세스가 대기 상태에서 준비 상태로 전이 되는 과정
Spooling	입출력장치의 공유 및 상대적으로 느린 입출력장치의 처리 속도를 보완하고 다중 프로그래밍 시스템의 성능을 향상시키기 위해 입출력할 데이터를 직접 입출력 장치에 보내지 않고 나중에 한꺼번에 입출력하기 위해 디스크에 저장하는 과정
교통량 제어기 Traffic Controller	프로세스의 상태에 대한 조사와 통보 담당

- 스레드 : 시스템의 여러 자원을 할당받아 실행하는 프로그램의 단위, 프로세스 내에서의 작업 단위

<스케줄링>

- 스케줄링 : 프로세스가 생성되어 실행될 때 필요한 시스템의 여러 자원을 해당 프로세스에게 할당하는 작업

장기 스케줄링	어떤 프로세스가 시스템의 자원을 차지할 수 있도록 할 것인가를 결정하여 준비상태 큐로 보내는 작업
중기 스케줄링	어떤 프로세스들이 CPU를 할당받을 것인지 결정하는 작업
단기 스케줄링	프로세스가 실행되기 위해 CPU를 할당받는 시기와 특정 프로세스를 지정하는 작업

- 스케줄링의 목적

공정성	모든 프로세스에 공정하게 할당
처리율 증가	단위 시간당 프로세스를 처리하는 비율 증가
CPU 이용률 증가	CPU가 순수하게 프로세스를 실행하는 데 사용되는 시간 비율을 증가
우선순위 제도	우선순위가 높은 프로세스를 먼저 실행
오버헤드 최소화	오버헤드를 최소화함
응답시간 최소화	작업을 지시하고 반응하기 시작하는 시간을 최소화
반환시간 최소화	프로세스를 제출한 시간부터 실행이 완료될 때까지 걸리는 시간을 최소화
대기시간 최소화	프로세스가 준비상태 큐에서 대기하는 시간을 최소화
균형있는 자원의 사용	메모리, 입출력장치 등의 자원을 균형 있게 사용
무한 연기 회피	자원을 사용하기 위해 무한정 연기되는 상태를 회피

- 비선점 스케줄링 : 이미 할당된 CPU를 다른 프로세스가 강제로 빼앗아 사용할 수 없는 스케줄링 기법(FCFS, SJF, 우선순위, HRN, 기한부 등)
- 선점 스케줄링 : 우선순위가 높은 다른 프로세스가 CPU를 강제로 빼앗아 사용할 수 있는 스케줄링 기법 (RR, SRT, 선점 우선순위, 다단계 큐, 다단계 피드백 큐 등)

<주요 스케줄링 알고리즘>

- FSFC(=FIFO) : 준비상태 큐에 도착한 순서에 따라 차례로 CPU를 할당하는 기법
- SJF(Shortest Job First) : 준비상태 큐에서 기다리고 있는 프로세스들 중 실행시간이 가장 짧은 프로세스에 먼저 CPU를 할당하는 기법
- HRN : 우선순위 = (대기시간+서비스시간) / 서비스시간

<인터넷>

- 인터넷 : 전 세계 수많은 컴퓨터와 네트워크들이 연결된 광범위한 통신망
- IP주소 : 인터넷에 연결된 모든 컴퓨터 자원을 구분하기 위한 고유한 주소

Class A	국가나 대형 통신망(16,777,216개 호스트)
Class B	중대형 통신망(65,536개 호스트)
Class C	소규모 통신망(256개 호스트)
Class D	멀티캐스트용
Class E	실험용으로 공용되지 않음

- 서브네팅 : 할당된 네트워크 주소를 여러개의 작은 네트워크로 나누어 사용하는 것
- IPv6 : 현재 사용하고 있는 IP 주소 체계인 IPv4 주소 부족 문제를 해결하기 위해 개발, 128비트의 주소, IPv4에 비해 자료 전송 속도가 빠름, 인증성·기밀성·데이터 무결성의 지원

유니캐스트 Unicast	단일 송신자와 단일 수신자 간의 통신 (1:1 통신)
멀티캐스트 Multicast	단일 송신자와 다중 수신자 간의 통신 (1:다 통신)
애니캐스트 Anycast	단일 송신자와 가장 가까이 있는 단일 수신자 간의 통신 (1:1 통신)

- 도메인 네임 : 숫자로 된 IP 주소를 사람이 이해하기 쉬운 문자 형태로 표현한 것

<OSI 참조 모델>

- OSI 참조모델 : 다른 시스템 간의 원활한 통신을 위해 ISO에서 제안한 통신 규약

물리 계층 Physical	전송에 필요한 두 장치 간의 기계적, 전기적, 기능적, 절차적 특성에 대한 규칙

데이터 링크 계층 Data Link	- 두 개의 인접한 개방 시스템들 간 신뢰성 있고 효율적이 정보 전송을 할 수 있도록 - 흐름 제어, 프레임 동기화, 오류 제어, 순서 제어
네트워크 계층 Network	- 개방 시스템들 간의 네트워크 연결을 관리하는 기능과 데이터의 교환 및 중계 - 경로설정(라우팅), 트래픽 제어, 패킷 정보 전송
전송계층 Transport	- 종단 시스템 간의 전송 연결 설정, 데이터 전송, 연결 해제 기능 - 주소 설정, 다중화, 오류제어, 흐름제어
세션 계층 Session	- 송수신 측 간 관련성을 유지하고 대화 제어를 담당 - 대화 구성 및 동기 제어, 데이터 교환 관리
표현 계층 Presentation	- 응용 계층으로부터 받은 데이터를 세션 계층에 맞게, 세션 계층에서 받은 데이터는 응용 계층에 맞게 변환하는 기능 - 코드 변환, 데이터 암호화, 데이터 압축, 구문 검색, 정보 형식 변환, 문맥 관리
응용 계층 Application	- 사용자가 OSI 환경에 접근할 수 있도록 응용 프로세스 간의 정보 교환, 전자 사서함, 파일 전송, 가상 터미널 등의 서비스 제공

<네트워크 관련 장비>

네트워크 인터페이스 카드(NIC)	컴퓨터와 컴퓨터 또는 컴퓨터와 네트워크를 연결하는 장치/(=이더넷 카드, 네트워크 어댑터)
허브(Hub)	한 사무실이나 가까운 거리의 컴퓨터들을 연결하는 장치, 각각의 회선을 통합하여 관리, 신호 증폭 기능을 하는 리피터 역할 포함 더미 허브: 네트워크에 흐르는 모든 데이터를 단순히 연결하는 기능만 제공 스위칭 허브: 네트워크 상에 흐르는 데이터의 유무 및 흐름을 제어하여 각각의 노드가 허브의 최대 대역폭을 사용할 수 있는 지능형 허브
브리지	수신한 신호를 재생시키거나 출력 전압을 높여 전송
리피터	디지털 신호의 장거리 전송을 위해 수신한 신호를 재생시키거나 출력 전압을 높여 전송하는 장치 브리지 : LAN과 LAN을 연결, LAN 안에서의 컴퓨터 그룹 연결
라우터	네트워크 계층의 장비, LAN과 LAN의 연결 및 경로 선택, 서로 다른 LAN이나 LAN과 WAN 연결
게이트웨이	OSI 전 계층의 프로토콜 구조가 다른 네트워크를 연결하는 장치
스위치	LAN과 LAN을 연결하여 더 큰 LAN을 만드는 장치

<TCP/IP>

- 프로토콜 : 데이터 교환을 원활하게 수행할 수 있도록 표준화시켜 놓은 통신 규약

구문 Syntax	전송하고자 하는 데이터의 형식, 부호화, 신호레벨 등을 규정
의미 Semantics	두 기기 간의 효율적이고 정확한 정보 전송을 위한 협조 사항, 오류 관리를 위한 제어 정보 규정
시간 Timing	두 기기 간의 통신 속도, 메시지의 순서 제어 등을 규정

- TCP/IP : 인터넷에 연결된 서로 다른 기종의 컴퓨터들이 데이터를 주고받을 수 있도록 하는 표준 프로토콜

TCP(Transmission Control Protocol)	- OSI 7계층의 전송 계층 - 신뢰성 있는 연결형 서비스 - 패킷의 다중화, 순서 제어, 오류 제어, 흐름 제어 기능
IP (Internet Protocol)	- OSI 7계층의 네트워크 계층 - 데이터그램을 기반으로 하는 비연결형 서비스 - 패킷의 분해/조립, 주소 지정, 경로 선택 기능

- TCP/IP의 구조

OSI	TCP/IP	기능
응용계층 표현계층 세션계층	응용 계층	- 응용 프로그램 간의 데이터 송수신 제공 - TELENET, FTP, SMTP, DNS, HTTP 등
전송계층	전송 계층	- 호스트들 간의 신뢰성 있는 통신 제공 - TCP, UDP, RTCP
네트워크 계층	인터넷 계층	- 데이터 전송을 위한 주소 지정, 경로 설정 제공 - IP, ICMP, IGMP, ARP, RARP
데이터링크 계층 물리계층	네트워크 액세스 계층	- 실제 데이터를 송수신 - 이더넷, IEEE802, HDLC, X.25, RS-232C, ARQ 등

- 응용계층의 주요 프로토콜

FTP	컴퓨터-컴퓨터, 컴퓨터-인터넷 사이에서 파일을 주고받을 수 있도록 하는 원격 파일 전송 프로토콜
SMTP	전자 우편을 교환하는 서비스
TELENET	- 멀리 떨어져 있는 컴퓨터에 접속하여 자신의 컴퓨터처럼 사용할 수 있도록 해주는 서비스 - 시스템 관리 작업을 할 수 있는 가상의 터미널 기능 수행
SNMP	TCP/IP의 네트워크 관리 프로토콜, 라우터나 허브 등 네트워크 기기의 네트워크 정보를 네트워크 관리 시스템에 보내는데 사용되는 표준 통신 규약
DNS	도메인 네임을 IP 주소로 매핑하는 시스템
HTTP	WWW에서 HTML주소를 송수신하기 위한 표준 프로토콜

- 전송 계층의 주요 프로토콜

TCP	- 양방향 연결형 서비스 - 가상 회선 연결 형태의 서비스 제공 - 스트림 위주 전달 - 신뢰성 있는 경로 확립, 메시지 전송 감독 - 순서 제어, 오류 제어, 흐름 제어 - 패킷의 분실, 손상, 지연 등 오류 발생 시 투명성이 보장되는 통신 제공 - 헤더 : 보통 20~60Byte, 최대 100Byte
UDP	- 데이터 전송 전 연결을 설정하지 않는 비연결형 서비스 - TCP에 비해 상대적으로 단순한 헤더 구조, 오버헤드가 적고 흐름제어, 순서제어가 없어 전송속도가 빠름 - 실시간 전송에 유리, 신뢰성<속도
RTCP	- RTP(Real-Time Transport Protocol) 패킷의 전송 품질을 제어하기 위한 제어 프로토콜 - 세션에 참여한 각 참여자들에게 주기적으로 제어 정보 전송 - 하위 프로토콜은 데이터 패킷과 제어 패킷의 다중화를 제공

- 인터넷 계층의 주요 프로토콜

IP	- 전송할 데이터에 주소를 지정하고 경로 설정 - 비연결형인 데이터그램 방식 사용, 신뢰성 보장 x
ICMP	- IP와 조합하여 통신 중 발생하는 오류의 처리와 전송 경로 변경 등을 위한 제어 메시지를 관리하는 역할 - 헤더 8바이트
IGMP	멀티캐스트를 지원하는 호스트나 라우터 사이에서 멀티캐스트 그룹 유지를 위해 사용
ARP	호스트의 IP 주소를 호스트와 연결된 네트워크 접속장치의 물리적 주소(MAC)로 바꿈
RARP	ARP와 반대, 물리적 주소를 IP로 변환

- 네트워크 액세스 계층의 주요 프로토콜

Ethernet (IEEE 802.3)	CSMA/CD 방식의 LAN
IEEE 802	LAN을 위한 표준 프로토콜
HDLC	비트 위주의 데이터 링크 제어 프로토콜
X.25	패킷 교환망을 통한 DTE와 DCE 간의 인터페이스 제공하는 프로토콜
RS-232C	공중 전화 교환망을 통한 DTE와 DCE 간의 인터페이스를 제공하는 프로토콜

<네트워크 관련 신기술>

1. IoT(사물 인터넷) : 다양한 사물들을 인터넷으로 서로 연결하여 진보된 서비스 제공
2. M2M(사물 통신) : 무선 통신을 이용한 기계-기계 통신
3. 모바일 컴퓨팅 : 휴대형 기기로 이동하면서 자유로이 네트워크에 접속하여 업무를 처리할 수 있음
4. 클라우드 컴퓨팅 : 각종 컴퓨팅 자원을 중앙 컴퓨터에 두고 인터넷 기능을 갖는 단말기
5. 그리드 컴퓨팅 : 지리적으로 분산되어 있는 컴퓨터를 초고속 인터넷망으로 연결하여 공유함으로써 하나의 고성능 컴퓨터처럼 활용하는 기술
6. 모바일 클라우드 컴퓨팅(MCC) ; 소비자와 소비자의 파트너가 클라우드 서비스를 이용하여 모바일 기기로 클라우드 컴퓨팅 인프라를 구성하여 여러 가지 정보와 자원 공유
7. 인터클라우드 컴퓨팅 : 각기 다른 클라우드 서비스를 연동하거나 컴퓨팅 자원의 동적 할당이 가능하도록 여러 클라우드 서비스 제공자들이 제공하는 클라우드 서비스나 자원 연결
8. 메시 네트워크 : 차세대 이동통신, 홈네트워킹, 공공안전 등 특수 목적을 위한 네트워크 기술 (대규모 디바이스의 네트워크 생성)
9. 와이선(Wi-SUN) : 장거리 무선통신을 필요로 하는 사물인터넷 서비스를 위한 저전력 장거리 통신 기술
10. NDN(Named Data Networking) : 콘텐츠 자체의 정보와 라우터 기능만으로 데이터 전송을 수행하는 기술
11. NGN(Next Generation Networking) : ITU-T에서 개발하고 있는 유선망 기반의 차세대 통신망 (이동사용 목표)
12. SDN(Software Defined Networking) : 네트워크를 컴퓨터처럼 모델링하여 여러 사용자가 각각의 소프트웨어로 네트워킹을 가상화하여 제어하고 관리하는 네트워크
13. NFC(Near Filed Communication) : 고주파를 이용한 근거리 무선통신 기술 (RFID 기술의 일종)
14. UWB(Ultra WideBand, 초광대역) : 짧은 거리에서 많은 양의 디지털 데이터를 낮은 전력으로 전송하기 위한 무선 기술, 무선 디지털 펄스
15. 피코넷(PICONET) : 여러 개의 독립된 통신장치가 블루투스 기술이나 UWB 통신 기술을 이용하여 통신망을 형성하는 무선 네트워크 기술
16. WBAN : 웨어러블 또는 임플란트(몸에 심는) 형태의 센서나 기기를 무선으로 연결하는 개인 영역 네트워킹 기술
17. GIS(지리 정보 시스템) : 지리적인 자료를 수집, 저장, 분석, 출력할 수 있는 컴퓨터 응용 시스템, 위성을 이용해 사물의 위치정보 제공
18. USN(유비쿼터스 센서 네트워크) : 각종 센서로 수집한 정보를 무선으로 수집할 수 있도록 구성한 네트워크, 필요한 모든 것에 RFID 태그를 부착하여 정보를 탐지함
19. SON(자동 구성 네트워크) : 주변 상황에 맞추어 스스로 망 구성, 망의 운영과 관리의 효율성을 높이는 것이 목표
20. 애드 혹 네트워크 : 별도의 고정된 유선망을 구축할 수 없는 상황에서 모바일 호스트만을 이용하여 구성한 네트워크
21. 네트워크 슬라이싱 : 네트워크에서 하나의 물리적인 코어 네트워크 인프라를 독립된 다수의 가상 네트워크로 분리하여 각각의 네트워크를 통해 다양한 고객 맞춤형 서비스를 제공하는 것이 목적
22. 저전력 블루투스 기술(BLE) : 일반 블루투스와 동일한 주파수 대역을 사용하지만 대기상태에서는 절전모드 유지
23. 지능형 초연결망 : 4차 산업혁명을 맞아 급격히 증가하는 데이터 트래픽을 효과적으로 수용하기 위해 시행되는 과학기술정보통신부 주관 사업
24. 파장 분할 다중화 : 광섬유를 이용한 통신 기술, 파장이 서로 다른 복수의 신호를 보냄으로 여러 대의 단말기가 동시에 통신회선을 사용할 수 있음
25. 소프트웨어 정의 데이터 센터(SDDC) : 데이터 센서의 모든 자원을 가상화하여 인력의 개입 없이 소프트웨어의 조작만으로 고나리 및 제어되는 데이터 센서를 의미
26. 개방형 링크드 데이터(LOD) : Linked Data + Open Data, 누구나 사용할 수 있도록 웹상에 공개된 연계 데이터

<네트워크 구축>

- 네트워크 : 두 대 이상의 컴퓨터를 연결하여 자원을 공유하는 것
- 성형(Star), 링형(Ring), 버스형(Bus), 계층형(Tree), 망형(Mesh, 모든 지점의 컴퓨터와 단말을 서로 연결)
- 네트워크의 분류

근거리 통신망 LAN	비교적 가까이에 있는 자원 연결, 주로 자원 공유 목적, 데이터 전송 속도가 빠르고 에러 발생률이 낮음
광대역 통신망 WAN	멀리 떨어진 사이트들을 연결하여 구성, 사이트 간 거리가 멀어 통신 속도가 느리고 에러 발생률이 높음

<경로 제어>

- 경로 제어(Routing) : 송수신측 간 전송 경로 중 최적 패킷 교환 경로를 결정하는 기능
- IGP(내부 게이트웨이 프로토콜) : 하나의 자율 시스템 내의 라우팅에 사용되는 프로토콜

RIP(Routing Information Protocol)	- 현재 가장 널리 사용되는 라우팅 프로토콜, bellman-ford 알고리즘 - 최대 홉수를 15로 제한
OSPF(Open Shortest Path First protocol)	- 대규모 네트워크에서 많이 사용되는 라우팅 프로토콜 - 라우팅 정보에 변화가 생길 경우 변화된 정보만 네트워크 내의 모든 라우터에 알리며 RIP에 비해 홉수 제한이 없음

- EGP(외부 게이트웨이 프로토콜) : 자율 시스템 간의 라우팅, 게이트웨이 간 라우팅에 사용
- BGP(Border Gateway Protocol) : 자율 시스템 간의 라우팅 프로토콜, EGP의 단점 보완 (초기에 라우터들이 연결될 때는 전체 경로 제어표를 교환, 이후에는 변화된 정보만을 교환)
- 트래픽 제어 : 전송되는 패킷의 흐름 또는 양을 조절하는 기능
- 흐름 제어 Flow : 송수신 측 사이에 전송되는 패킷의 양이나 속도를 규제하는 기능

정지-대기 stop-and-wait	- ACK를 받은 후 다음 패킷 전송 - 한 번에 하나의 패킷만 전송
슬라이딩 윈도우	- ACK 없이도 보낼 수 있는 패킷의 최대치를 미리 약속받음(윈도우 크기) - 한 번에 여러 개의 패킷을 전송할 수 있어 전송 효율이 좋음 - 윈도우 크기는 ACK가 전달되면 증가하고 NAK가 전달되면 감소함

- 폭주 제어 Congestion : 네트워크 내의 패킷 수를 조절하여 네트워크의 오버플로를 방지하는 기능

느린 시작 slow start	- 윈도우의 크기를 크기를 1,2,4,8,... 씩 증가시켜 초기에는 느리지만 갈수록 빨라짐 - 전송 데이터의 크기가 임계 값에 도달하면 혼잡 회피로 넘어감
혼잡 회피 congestion avoidance	slow start의 지수적 증가가 임계값에 도달하면 윈도우의 크기를 1씩 선형적으로 증가시켜 혼잡 예방

<SW 관련 신기술>

1. 인공지능(AI) : 컴퓨터 스스로 추론, 학습, 판단 등 인간지능적인 작업을 수행하는 시스템 (패턴 인식, 전문가 시스템, 로봇공학 등)
2. 뉴럴링크 : 신경 레이스 - 작은 전극을 뇌에 이식함으로써 생각을 업로드하고 다운로드 하는 것 목표
3. 딥 러닝 : 인간의 두뇌를 모델로 만들얼진 인공 신경망을 기반으로 하는 기계 학습 기술
4. 전문가 시스템 : 특정 분야의 전문가가 수행하는 고도의 업무를 지원하기 위한 컴퓨터 응용 프로그램
5. 증강현실(AR) : 실제 촬영한 화면에 가상의 정보를 부여하여 보여주는 기술
6. 블록체인 : P2P 네트워크를 이용하여 온라인 금융 거래 정보를 온라인 네트워크 참여자의 디지털 장비에 분산 저장
7. 분산 원장 기술(있) : 중앙 관리자나 중앙 데이터 저장소에 존재하지 않고 P2P 망 내의 참여자들에게 모든 거래 목록이 분산 저장되어 거래가 발생할 때마다 지속적으로 갱신되는 디지털 원장
8. 해시 : 임의의 길이의 입력 데이터나 메시지를 고정된 길이의 값이나 키로 변환
9. 양자 암호키 분배(QKD) : 양자 통신을 위해 비밀키를 분배하여 관리하는 기술
10. 프라이버시 강화 기술(PET) : 개인정보 침해 위험을 관리하기 위한 핵심 기술
11. 공통 평가 기준 (CC) : 정보화 순기능 역할을 보장하기 위해 정보화 제품의 정보보호 기능과 이에 대한 사용 환경 등급을 정한 기준
12. 개인정보 영향평가 제도 (PIA) : 개인 정보를 활용하는 새로운 정보시스템의 도인 및 기존 정보시스템의 중요한 변경 시 프로그램의 구축, 운영이 기업의 고객은 물론 국민의 사생활에 미칠 영향에 대해 미리 조사, 분석, 평가하는 제도
13. 그레이웨어(Grayware) : 소프트웨어를 제공하는 입장에서는 악의적이지 않는 유용한 소프트웨어라고 주장할 수 있지만 사용자 입장에서는 유용할수도 있고 악의적일 수도 있는 악성 코드나 악성 공유웨어
14. 매시업(Mashup) : 웹에서 제공하는 정보 및 서비스를 이용하여 새로운 소프트웨어나 서비스, 데이터베이스 등을 만드는 기술
15. 리치 인터넷 애플리케이션(RIA) : 플래시 애니메이션 기술과 웹 서버 애플리케이션 기술을 통합하여 기존 HTML보다 역동적이고 인터랙티브한 웹 페이지를 제공하는 신개념 플래시 웹 페이지 제작 기술
16. 시맨틱 웹 : 컴퓨터가 사람을 대신하여 정보를 읽고 이해하고 가공하여 새로운 정보를 만들어 낼 수 있도록 이해하기 쉬운 의미를 가진 차세대 지능형 웹
17. 증발품(Vaporware) : 판매 계획 또는 배포 계획은 발표되었으나 실제로 고객에게 판매되거나 배포되지 않고 있는 소프트웨어
18. 오픈 그리드 서비스 아키텍처(OGSA) : 애플리케이션 공유를 위한 웹 서비스를 그리도 상에서 제공하기 위해 만든 개방형 표준
19. 서비스 지향 아키텍처(SOA) : 기업의 소프트웨어 인프라인 정보시스템을 공유와 재사용이 가능한 서비스 단위나 컴포넌트 중심으로 구축하는 정보기술 아키텍처
20. 서비스형 소프트웨어(SaaS) : 소프트웨어의 여러 기능 중에서 사용자가 필요로 하는 서비스만 이용할 수 있도록 한 소프트웨어
21. 소프트웨어 에스크로 : 소프트웨어 개발자의 지식재산권을 보호하고 사용자는 저렴한 비용으로 소프트웨어를 안정적으로 사용 및 유지보수 할 수 있도록 소스 프로그램과 기술 정보 등을 제3의 기관에 보관
22. 복잡 이벤트 처리(CEP) : 실시간으로 발생하는 많은 사건들 중 의미가 있는 것만을 추출할 수 있도록 사건 발생 조건을 정의하는 데이터 처리 방법
23. 디지털 트윈 : 현실 속의 사물을 소프트웨어로 가상화한 모델, 실제 물리적인 자산을 소프트웨어로 가상화함으로써 실제 자산의 특성에 대한 정확한 정보를 얻을 수 있다

<HW 관련 신기술>

1. 고가용성(HA) : 긴 시간동안 안정적인 서비스 운영을 위해 장애 발생 시 즉시 다른 시스템으로 대체 가능한 환경을 구축하는 매커니즘
2. 3D Printing : 대상을 손으로 만질 수 있는 실제 두께로 만들어내는 것
3. 4D Printing : 특정 시간이나 환경 조건이 갖추어지면 스

스로 형태를 변화시키거나 제조되는 자가 조립 기술이 적용된 제품을 3D Printing하는 기술

4. RAID : 여러개의 하드디스크로 디스크 배열을 구성하여 파일을 구성하고 있는 데이터 블록들을 서로 다른 디스크들에 분산 저장할 경우, 그 블록들을 여러 디스크에서 동시에 읽거나 쓸 수 있으므로 디스크의 속도가 매우 향상되는 것
5. 4K 해상도 : 차세대 고화질 모니터의 해상도
6. 앤 스크린 : N개의 서로 다른 단말기에서 동일한 콘텐츠를 자유롭게 이용할 수 있는 서비스
7. 컴패니언 스크린 : TV 방송 시청 시 방송 내용을 공유하며 추가적인 기능을 수행할 수 있는 스마트폰, 태블릿 PC
8. 신 클라이언트 PC : 하드디스크나 주변장치 없이 기본적인 메모리만 갖추고 서버와 네트워크로 운용되는 개인용 컴퓨터, 서버 기반 컴퓨팅과 깊은 관계
9. 패블릿 : 폰과 태블릿의 합성어, 태블릿 기능을 포함한 5인치 이상의 대화면 스마트폰
10. C형 USB : 범용 인터페이스 규격인 USB 표준
11. 멤스(MEMS) : 초정밀 반도체 제조 기술을 바탕으로 센서, 액추에이터 등 기계 구조를 다양한 기술로 미세 가공하여 전기기계적 동작을 할 수 있도록 한 초미세 장치
12. 트러스트존 기술 : 하나의 프로세서 내에 일반 애플리케이션을 처리하는 일반 구역과 보안이 필요한 애플리케이션을 처리하는 보안 구역으로 분할하여 관리하는 하드웨어 기반의 보안 기술
13. 엠디스크(M-DISC) : 한 번의 기록만으로 자료를 영구 보관할 수 있는 광 저장장치
14. 멤리스터 : 메모리+레지스터, 전류의 방향과 양 등 기존의 경험을 모두 기억하는 특별한 소자

<Secure OS>

-Secure OS : 보안 기능을 갖춘 커널을 이식하여 외부의 침입으로부터 시스템 자원을 보호하는 운영체제
참조 모니터 : 보호 대상 객체에 대한 접근 통제를 수행하는 추상머신 (격리성, 검증가능성, 완전성)

<DB 관련 신기술>

1. 빅데이터 : 기존의 관리 방법이나 분석 체계로는 처리하기 어려운 막대한 양의 정형, 비정형 데이터 집합
2. 브로드 데이터 : 다양한 채널에서 소비자와 상호 작용을 통해 생성된 것, 기업 마케팅에 있어 효율적이고 다양함
3. 메타 데이터 : 일련의 데이터를 정의하고 설명해주는 데이터(컴퓨터 - 데이터사전 내용, 스키마 / HTML - 메타 태그 내 내용)
4. 디지털 아카이빙 : 디지털 정보 자원을 장기적으로 보존하기 위한 작업(아날로그-디지털로 변환 후 압축하여 저장 / 디지털-체계적으로 분류, 메타데이터를 만들어 DB화)
5. 하둡(Hadoop) : 오픈 소스를 기반으로 한 분산 컴퓨팅 플랫폼, 일반 PC급 컴퓨터들로 가상화된 대형 스토리지를 형성하고 그 안에 보관된 거대한 데이터 세트를 병렬로 처리할 수 있도록 개발된 자바 소프트웨어 프레임워크
6. 맵리듀스(MapReduce) : 대용량 데이터를 분산 처리하기 위한 목적으로 개발된 프로그래밍 모델, Map- 흩어져 있는 데이터를 연관성 있는 데이터 분류로 묶는 작업 / Reduce- 중복 데이터를 제거하고 원하는 데이터를 추출
7. 타조 : 오픈소스 기반 분산 컴퓨팅 플랫폼인 아피치 하둡 기반의 분산 데이터 웨어하우스 프로젝트
8. 데이터 다이어트 : 데이터를 압축하고 중복된 정보는 중복을 배제하고 새로운 기준에 따라 나누어 저장하는 작업
9. 데이터 마이닝 : 대량의 데이터를 분석하여 데이터에 내재된 변수 사이의 상호 관계를 구명하여 일정한 패턴을 찾아내는 기법
10. OLAP(Online Analytical Processing) : 다차원으로 이루어진 데이터로부터 통계적인 요약 정보를 분석하여 의사결정에 활용하는 방식

<회복/병행제어>

- 회복 : 데이터베이스가 손상되었을 때 손상되기 이전의 정상 상태로 복구하는 작업

연기 갱신 기법 Deferred Update	- 트랜잭션이 성공적으로 완료될 때까지 DB에 대한 실질적인 갱신을 연기 - 트랜잭션이 수행되는 동안 갱신 내용은 일단 Log에 보관 - 트랜잭션의 부분 완료 시점에 Log에 보관한 내용을 실제 DB에 기록
즉각 갱신 기법 Immediate Update	- 트랜잭션이 데이터를 갱신하면 트랜잭션이 부분 완료되기 전이라도 즉시 실제 DB에 반영 - 장애가 발생하여 회복 작업을 할 경우를 대비하여 갱신된 내용들은 Log에 보관
그림자 페이지 대체 기법 Shadow Paging	- 갱신 이전의 DB를 일정 크기의 페이지 단위로 구성하여 각 페이지마다 복사본인 그림자 페이지를 별도 보관 - 실제 페이지를 대상으로 갱신 작업을 수행하다 장애가 발생하여 rollback을 할 때에는 갱신 이후의 실제 페이지 부분을 그림자 페이지로 대체
검사점 기법 Check Point	트랜잭션 실행 중 특정 단계에서 재실행할 수 있도록 갱신 내용이나 시스템에 대한 상황 등에 관한 정보와 함께 검사점을 로그에 보관해두고 장애 발생 시 트랜잭션 전체를 철회하지 않고 검사점부터 회복 작업을 수행하여 회복시간을 절약하도록 하는 기법

- 병행제어 : 동시에 여러 개의 트랜잭션을 병행수행할 때 동시에 실행되는 트랜잭션들이 데이터베이스의 일관성을 파괴하지 않도록 트랜잭션 간 상호작용 제어

로킹 Locking	- 트랜잭션들이 어떤 로킹 단위를 엑세스하기 전에 lock을 요청하여 lock이 허락되어야만 그 로킹 단위를 액세스할 수 있음 - 주요 데이터의 엑세스를 상호 배타적으로 함
타임 스탬프 순서 Time Stamp Ordering	- 트랜잭션이 실행을 시작하기 전에 시간표를 부여하여 부여된 시간에 따라 트랜잭션 작업을 수행하는 기법 - 직렬성 순서를 결정하기 위해 트랜잭션 간의 처리 순서를 미리 선택하는 기법들 중에서 가장 보편적인 방법
최적 병행수행 (검증, 확인,	병행수행하고자 하는 대부분의 트랜잭션이 판독 전용 트랜잭션일 경우, 트랜잭션 간

낙관적 기법)	의 충돌률이 매우 낮아 병행제어 기법을 사용하지 않고 실행되어도 이 중의 많은 트랜잭션은 시스템의 상태를 일관성 있게 유지한다는 점을 이용
다중 버전 기법	- 타임 스탬프 개념 이용 - 갱신될 때마다의 버전을 부여하여 관리

- 로킹 단위 : 병행제어에서 한꺼번에 로킹할 수 있는 객체의 크기 의미

<교착 상태>

- 교착 상태 : 상호배제에 의해 나타나는 문제점, 서로 다른 프로세스가 점유하고 있는 자원을 요구하며 무한정 기다리는 현상
- 필요충분조건

상호 배제 Mutual Exclusion	한 번에 하나의 프로세스만이 공유 자원을 사용할 수 있어야 함
점유와 대기 Hold and Wait	최소한 하나의 자원을 점유하고 있으면서 다른 프로세스에 할당되어 사용하고 있는 자원을 추가로 점유하기 위해 대기하는 프로세스가 있어야 함
비선점 Non-preemption	다른 프로세스에 할당된 자원은 사용이 끝날 때 까지 강제로 빼앗을 수 없어야 함
환형 대기 Circular Wait	공유 자원과 대기하는 프로세스들이 원형으로 구성되어 있어 자신에게 할당된 자원을 점유하면서 앞이나 뒤에 있는 프로세스의 자원을 요구해야 함

- 해결방법

예방 기법	- 교착상태가 발생하지 않도록 사전에 시스템 제어 - 네가지 조건 중 하나를 제거 - 자원낭비가 가장 심함
회피 기법	- 교착상태가 발생하면 적절히 피해감 - 은행원 알고리즘
발견 기법	- 교착상태가 발생했는지 점검 - 교착상태 발견 알고리즘, 자원 할당 그래프
회복 기법	교착상태를 일으킨 프로세스를 종료하거나 교착상태의 프로세스에 할당된 자원을 선점하여 프로세스나 자원 회복

12. 제품 소프트웨어 패키징

<소프트웨어 패키징>

- 소프트웨어 패키징 : 모듈별로 생성한 실행 파일들을 묶어 배포용 설치 파일을 만드는 것, 사용자 중심
- 작업 순서 : 기능 식별 → 모듈화 → 빌드 진행 → 사용자 환경 분석 → 패키징 적용 및 시험 → 패키징 변경 개선 → 배포

<릴리즈 노트 작성>

- 릴리즈 노트 : 개발 과정에서 정리된 릴리즈 정보를 소프트웨어의 최종 사용자인 고객과 공유하기 위한 문서
- 작성 순서 : 모듈 식별 → 릴리즈 정보 확인 → 릴리즈 노트 개요 작성 → 영향도 체크 → 정식 릴리즈 노트 작성 → 추가 개선 항목 식별
- Header(머릿말) : 릴리즈 노트 이름, 소프트웨어 이름, 릴리즈 버전, 릴리즈 날짜, 릴리즈 노트 날짜, 릴리즈 노트 버전 등 표시

<디지털 저작권 관리(DRM)>

- DRM : 저작권자가 배포한 디지털 콘텐츠가 저작권자가 의도한 용도로만 사용되도록 디지털 콘텐츠의 생성, 유통, 이용까지의 전 과정에 걸쳐 사용되는 디지털 콘텐츠 관리 및 보호 기술
- 디지털 저작권 관리의 흐름 및 구성 요소

클리어링 하우스 Clearing House	저작권에 대한 사용 권한, 라이선스 발급, 암호화된 키 관리, 사용량에 따른 결제 관리 등을 수행
콘텐츠 제공자 Contents Provider	콘텐츠를 제공하는 저작권자
패키저 Packager	콘텐츠를 메타 데이터와 함께 배포 가능한 형태로 묶어 암호화하는 프로그램
콘텐츠 분배자 Contents Distributor	암호화된 콘텐츠를 유통하는 곳, 사람
콘텐츠 소비자 Customer	콘텐츠를 구매해서 사용하는 주체
DRM 컨트롤러	배포된 콘텐츠의 이용 권한을 통제하는 프로그램
보안 컨테이너 Security Container	콘텐츠 원본을 안전하게 유통하기 위한 전자적 보안 장치

- 디지털 저작권 관리의 기술 요소

암호화 Encryption	콘텐츠 및 라이선스를 암호화하고 전자서명을 할 수 있는 기술
키 관리 Key Management	콘텐츠를 암호화한 키에 대한 저장 및 분배 기술
암호화 파일 생성 Packager	콘텐츠를 암호화된 콘텐츠로 생성하기 위한 기술
식별 기술 Identification	콘텐츠에 대한 식별 체계 표현 기술
저작권 표현 Right Expression	라이선스의 내용 표현 기술
정책 관리 Policy Management	라이선스 발급 및 사용에 대한 정책 표현 및 관리 기술
크랙 방지 Tamper Resistance	크랙에 의한 콘텐츠 사용 방지 기술
인증 Authenfication	라이선스 발급 및 사용의 기준이 되는 사용자 인증 기술

<소프트웨어 버전 등록>

- 형상 관리 : 소프트웨어의 개발 과정에서 소프트웨어의 변경 사항을 관리하기 위해 개발된 일련의 활동

형상 식별	형상 관리 대상에 이름과 관리 번호 부여, 계층 구조고 구분하여 수정 및 추적이 용이하도록 하는 작업
버전 제어	소프트웨어 업그레이드, 유지보수 과정에서 생성된 다른 버전의 형상 항목 관리, 이를 위해 특정 절차와 도구를 결합
형상 통제	식별된 형상 항목에 대한 변경 요구를 검토하여 현재의 기준선이 잘 반영될 수 있도록 조정
형상 감사	기준선의 무결성을 평가하기 위해 확인, 검증, 검열 과정을 통해 공식적으로 승인
형상 기록	형상의 식별, 통제, 감사 작업의 결과를 기록, 관리, 보고서 작성

- 소프트웨어 버전 등록 관련 주요 기능

저장소 repository	최신 버전의 파일들과 변경 내역에 대한 정보들이 저장되어있는 곳
가져오기 import	버전 관리가 되고 있지 않은 아무것도 없는 저장소에 처음으로 파일 복사
체크아웃	프로그램을 수정하기 위해 저장소에서 파일을 받아옴 소스파일과 함께 버전관리를 위한 파일들도 받아옴
체크인	체크아웃 한 파일의 수정을 완료한 후 저장소의 파일을 새로운 버전으로 갱신

커밋	체크인을 수행할 때 이전에 갱신된 내용이 있는 경우 충돌을 알리고 diff 도구를 이용해 수정 후 갱신 완료
동기화	저장소에 있는 최신 버전으로 자신의 작업 공간을 동기화함

-소프트웨어 버전 등록 과정 : 가져오기→인출→예치→동기화→차이

<소프트웨어 버전 관리 도구>

- 공유 폴더 방식 : 버전 관리 자료가 로컬 컴퓨터의 공유 폴더에 저장되어 관리되는 방식
- 클라이언트/서버 방식 : 버전 관리 자료가 중앙 시스템(서버)에 저장되어 관리되는 방식
- 분산 저장소 방식 : 버전 관리 자료가 하나의 원격 저장소와 분산된 개발자 PC의 로컬 저장소에 함께 저장되어 관리되는 방식
- Subversion(SVN) : CVS를 개선한 것, 클라이언트/서버 구조, 아파치 소프트웨어 재단에서 2000년에 발표
- Git : 리누스 토발즈가 2005년 리눅스 커널 개발에 사용할 관리 도구로 개발한 이후 주니오 하마노에 의해 유지보수

<빌드 자동화 도구>

- 빌드 : 소스 코드 파일들을 컴파일한 후 여러 개의 모듈을 묶어 실행 파일로 만드는 과정
- 빌드 자동화 도구 : 빌드를 포함하여 테스트 및 배포를 자동화하는 도구
- Jenkins : Java 기반의 오픈소스, 가장 많이 사용되는 빌드 자동화 도구, 대부분의 형상관리 도구와 연동 가능, 여러대의 컴퓨터를 이용한 분산 빌드나 테스트 가능
- Gradle : Groovy를 기반으로 한 오픈소스 형태의 자동화 도구, 안드로이드 앱 개발 환경에서 사용, 실행할 처리 명령들을 모아 태스크로 만든 후 태스크 단위로 실행, 안드로이드, 자바, C/C++, 파이썬 등의 언어 빌드 가능