01 소프트웨어 구축

Section 1. 소프트웨어 공학 개념

1. 소프트웨어 공학

(1) 소프트웨어 공학의 정의

- 소프트웨어 위기를 극복하고 품질 높은 소프트웨어를 개발하기 위한 학문

(2) 소프트웨어 위기의 원인

- 소프트웨어 특성에 대한 이해 부족
- 소프트웨어 관리 방법론 부재
- 올바른 설계 없이 프로그래밍에만 치중
- 소프트웨어 개발에 대한 전문적 교육 부족
- 작업일정과 비용의 추정치가 부정확

(3) 소프트웨어의 위기의 결과

- 개발 인력의 부족과 인건비 상승
- 소프트웨어 성능 및 신뢰성 부족
- 개발 기간 및 비용의 증가
- 소프트웨어 품질저하 및 유지보수 비용 증가
- 소프트웨어의 생산성 저하

2. 소프트웨어 공학의 3R

(1) 역공학(Reverse Engineering)

- 기존 개발된 시스템을 CASE도구를 이용하여 사양서, 설계서 등의 문서로 추출하는 작업

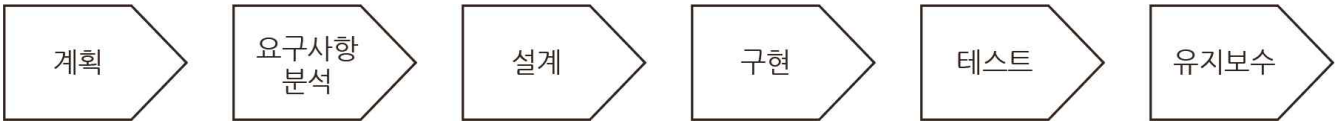
(2) 재공학(Re-engineering)

- 기존 시스템의 기능이나 성능을 업그레이드 하는 작업

(3) 재사용(Reuse)

- 이미 개발되어 그 기능, 성능 및 품질을 인정받았던 소프트웨어의 전체 또는 일부분을 다시 사용
- 재사용 방법
  - 합성 중심(Composition Based) : 소프트웨어 모듈끼워 맞추어 소프트웨어를 완성시키는 방법
  - 생성 중심(Generation Based) : 추상화 형태로 쓰여진 명세를 구체화하여 프로그램을 만드는 방법

3. 소프트웨어 개발 단계



Section 2. 소프트웨어 개발 방법론

1. 소프트웨어 개발 방법론 종류

⑴ 구조적 방법론

- 절차지향 소프트웨어 개발 방법론
- 요구사항 분석 → 구조적 분석 → 구조적 설계 → 구조적 프로그래밍
- 구조적 방법론 구성요소 : 데이터 흐름도(DFD), 자료사전(DD), 상태전이도(STD), 소단위 명세서(Minispec)

⑵ 정보공학 방법론

- 기업의 주요 부분을 계획, 분석, 설계, 구축에 정형화된 기법들을 상호 연관성 있게 통합, 적용하는 데이터 중심 방법론
- 정보전략계획 수립단계 → 업무영역 분석단계 → 시스템 설계단계 → 시스템 구축단계

⑶ 객체지향 개발 방법론

- 현실세계의 개체(Entity)를 속성(Attribute)과 메서드(Method)형태로 표현

⑷ CBD(Component Based Development) 분석 방법론

- 재사용 가능한 컴포넌트의 개발 또는 상용 컴포넌트를 조합해 어플리케이션 개발

⑸ 애자일 방법론

- 기존 방법론들이 절차를 중시한 나머지 변화에 빠른 대응을 할 수 없는 단점 개선을 위해 등장

2. 소프트웨어 개발 모델

⑴ 폭포수 모델(Waterfall Model)

- 계획, 분석, 설계, 구현, 테스트, 운영 등 전 과정을 순차적으로 접근하는 개발모델

⑵ 프로토타이핑 모델(Prototyping Model)

- 고객이 요구한 주요 기능을 프로토타입으로 구현하여 완성해가는 모델
- 계획수립 → 프로토타입 개발 → 사용자 평가 → 구현 → 인수

⑶ 나선형 모델(Spiral Model)

- 폭포수 모델과 프로토타이핑 모델의 장점을 수용하고, 위험 분석을 추가한 점증적 개발 모델
- 계획수립(planning) → 위험분석(Risk Analysis) → 공학적 개발(Development) → 평가(Evaluation)

⑷ RAD(Rapid Application Development) 모델

- 매우 짧은 개발 주기를 강조하는 점진적 소프트웨어 개발 방식
- CASE(Computer Adied Software Engineering) 도구를 이용해 시스템을 개발

⑸ V 모형

- 폭포수 모델에 시스템 검증과 테스트 작업을 강조

⑹ 4세대 기법(4th Generation Techniques)

- CASE등의 자동화도구를 이용하여 요구사항 명세로부터 원시코드를 자동으로 생성

3. 애자일(Agile) 방법론

(1) 애자일 방법론의 개념

- 애자일 방법론은 소프트웨어 개발 방법에 있어서 아무런 계획이 없는 개발 방법과, 계획이 지나치게 많은 개발 방법들 사이에서 타협점을 찾고자 하는 방법론

(2) 애자일 선언문

- 공정과 도구보다 개인과 상호작용을
- 포괄적인 문서보다 작동하는 소프트웨어를
- 계약 협상보다 고객과의 협력을
- 계획을 따르기보다 변화에 대응하기를
- 우리는 왼쪽 항목의 가치를 인정하면서도 오른쪽 항목을 더 중요하게 여긴다.

(3) 애자일 방법론 종류

① XP(eXtream Programming)

- XP 5가지 핵심가치
  - 용기 : 고객의 요구사항 변화에 능동적인 대처
  - 존중 : 개발자의 역량을 존중하고 충분한 권한과 권리를 부여
  - 의사소통 : 개발자, 관리자, 고객 간의 원활한 의사소통
  - 피드백 : 의사소통에 따른 즉각적인 피드백
  - 단순성 : 부가적 기능, 사용되지 않는 구조와 알고리즘 배제

② 스크럼 (SCRUM)

- 개발 주기는 30일 정도(스프린트)로 조절하고 개발 주기마다 실제 동작할 수 있는 결과를 제공
- 날마다 15분 정도의 회의

③ 그 외 애자일 방법론

- 크리스털 패밀리(Crystal)
- Feature-Driven Development (FDD)
- Adaptive Software Development, ASD

02 프로젝트 계획 및 분석

Section 1. 프로젝트 계획

1. 프로젝트 관리

(1) 프로젝트 관리의 개념

- 특정한 목적을 가진 프로젝트를 한정된 기간, 예산, 자원 내에서 사용자가 만족할 만한 제품을 개발하도록 행하는 기술적, 관리적 활동

(2) 프로젝트 핵심 관리대상(3P)

- 사람(People)
- 문제(Problem)
- 프로세스(Process)

(3) PMBOK(Project Management Body of Knowledge)

- PMI(Project Management Institute)에서 제작한 프로젝트 관리 프로세스 및 지식 체계
- PMBOK 5단계 프로세스 그룹
  - 1단계 : 프로젝트 착수
  - 2단계 : 프로젝트 계획
  - 3단계 : 프로젝트 실행
  - 4단계 : 프로젝트 통제
  - 5단계 : 프로젝트 종료

2. 개발 비용 산정

- 하향식
  - 전문가 기법
  - 델파이 기법
- 상향식
  - LOC(원시코드 라인수) 기법
  - 단계별 인원수(M/M) 기법
- 수학적
  - COCOMO 기법
  - PUTNAM 기법
  - FP(기능점수) 기법

3. 개발 일정 산정

(1) 소프트웨어 개발 일정 계획

- 작업 순서
  - 작업분해(Work Breakdown Structure)
  - CPM 네트워크 작성
  - 최소 소요 기간을 구함
  - 소요 M/M, 기간 산정하여 CPM 수정
  - 간트 차트로 표현

(2) WBS(Work Breakdown Structure)

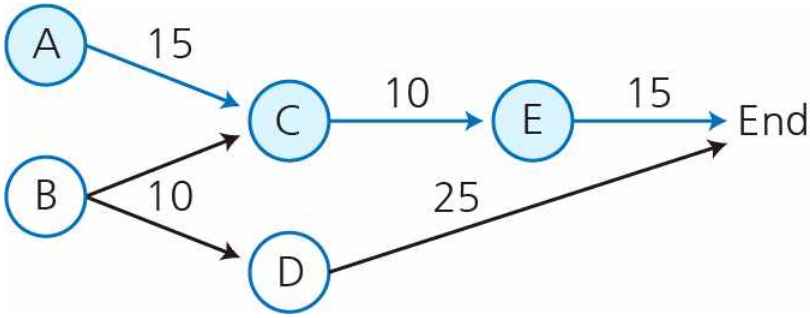
- 프로젝트 목표를 달성하기 위해 필요한 활동과 업무를 세분화 하는 작업

⑶ Network Chart(PERT/CPM)

- CPM 소작업 리스트

작업	선행작업	소요기간(일)
A	-	15
B	-	10
C	A, B	10
D	B	25
E	C	15

- CPM 네트워크 작성



- 소요기간 산정
  - 임계경로(Critical Path) : 40일
  - D의 가장 빠른 착수일 : 10일
  - D의 가장 늦은 착수일 : 15일
  - D의 여유기간 : 5일

⑷ 간트 차트(Gantt chart)

- 일정 계획의 최종 산출물
- 프로젝트 일정관리를 위한 바(bar)형태의 도구

Section 2. 요구사항 분석

1. 현행 시스템 분석

(1) 현행 시스템 파악

- 현행 시스템이 어떤 하위 시스템으로 구성되어 있는지
- 제공하는 기능이 무엇인지
- 다른 시스템들과 어떤 정보를 주고받는지
- 어떤 기술요소를 사용하고 있는지
- 사용하고 있는 소프트웨어 및 하드웨어는 무엇인지
- 네트워크는 어떻게 구성되어 있는지

(2) 플랫폼 기능 분석

① 플랫폼 정의

- 어플리케이션을 구동시키는데 필요한 하드웨어와 소프트웨어의 결합

② 플랫폼의 유형

- 싱글 사이드 플랫폼(single-side platform)
  - 제휴 관계를 통해 소비자와 공급자를 연결하는 형태
- 투 사이드 플랫폼(two-side platform)
  - 두 그룹을 중개하고 모두에게 개방하는 형태
- 멀티 사이드 플랫폼(multi-side platform)
  - 다양한 이해관계 그룹을 연결하여 중개하는 형태

(3) 미들웨어(Middleware) 분석

① 미들웨어 개념

- 양 쪽을 연결하여 데이터를 주고 받을 수 있도록 중간에서 매개 역할을 하는 소프트웨어

② 미들웨어 종류

- RPC(Remote Procedure Call)
  - 클라이언트가 원격에서 동작하는 프로시저를 호출하는 시스템
- MOM(Message Oriented Middleware)
  - 응용 소프트웨어 간의 데이터 통신을 위한 소프트웨어
  - 메시지 기반의 비동기형 메시지를 전달하는 방식의 미들웨어
- ORB (Object Request Broker)
  - 객체지향 시스템에서 객체 및 서비스를 요청하고 전송할 수 있도록 지원하는 미들웨어
- DB 접속 미들웨어
  - 애플리케이션과 데이터베이스 서버를 연결해주는 미들웨어
- TP 모니터
  - 온라인 트랜잭션을 처리 및 감시하는 미들웨어

- WAS(Web Application Server)
  - 동적인 콘텐츠를 처리하기 위한 미들웨어
- ESB(Enterprise Service Bus)
  - 메시지 기반으로 느슨한 결합형태의 표준 인터페이스 통신을 지원하는 미들웨어

2. 요구 공학

(1) 요구공학 개념

- 고객 요구를 체계적으로 수집, 분석, 명세화, 검증하고 추적, 변경되는 요구사항을 도출하고 관리하는 기법

(2) 요구공학의 필요성

- 분석의 어려움
- 요구사항 변화
- 관점별 차이 발생

(3) 요구사항의 분류

- 기능적 요구사항
  - 소프트웨어를 구성하는 기능들이 무엇인지를 정의한 것
- 비기능적 요구사항
  - 소프트웨어의 기능들에 대한 조건과 제약 사항들이 무엇인지 정의한 것
  - 보안, 성능, 품질, 안정성 등

(4) 요구사항 개발 프로세스

① 도출

- 소프트웨어가 해결해야 할 문제를 이해하고 요구사항이 어디에 있고, 어떻게 수집할 것인가를 확인

② 분석

- 요구사항들 간에 상충 되는 것을 해결
- 구조적 분석 도구
  - DFD(Data Flow Diagram) : 자료 흐름도
  - Data Dictionary : 자료 사전
  - Mini-Spec : 소단위 명세서
  - ERD(Entity Relationship Diagram) : 개체 관계도
  - STD(State Transition Diagram) : 상태 전이도
- 객체지향 분석 도구
  - UML(Unified Modeling Language)
  - 모델링

③ 명세

- 정형 명세 기법
- 비정형 명세 기법

④ 확인

- 분석가가 요구사항을 이해했는지 확인(Validation)
- 요구사항 문서가 일관성 있고 완전한지 검증(Verification)

(5) 요구사항 분석 도구

① 요구사항 분석 CASE(Computer Aided Software Engineering) 도구

분류	설명
상위 CASE	- 생명주기 전반부에 사용되며, 소프트웨어의 계획과 요구분석, 설계 단계를 지원한다. - 모순검사, 오류검사, 자료흐름도 작성 등의 기능을 수행한다.
하위 CASE	- 생명 주기 후반부에 사용되며, 코드의 작성과 테스트, 문서화 하는 과정을 지원한다. - 구문 편집기, 코드 생성기 등의 기능을 수행한다.
통합 CASE	- 소프트웨어 생명주기에 포함되는 전체 과정을 지원한다.

② HIPO(Hierarchy Inut Process Output)

- HIPO 의 개념
  - 하향식 소프트웨어 개발을 위한 문서화 도구
- HIPO Chart 종류
  - 가시적 도표 (Visual Table of Content)
  - 총체적 도표 (Overview Diagram)
  - 세부적 도표 (Detail Diagram)

3. 요구사항 분석 모델링

(1) 모델링의 개념

- 복잡한 시스템을 쉽게 이해하기 위해 불필요한 부분을 생략하고 추상화하여 간단한 모델로 표현하는 것을 의미

(2) 구조적 분석 모델

- 자료 흐름도(DFD, Data Flow Diagram)
  - 자료의 흐름과 처리 과정을 도형 중심으로 기술
- 자료사전(Data Dictionary, DD)
  - 자료흐름도에 기술된 모든 자료들에 대한 사항을 자세히 정의
  - 이름 + 주소 + [남|여] + {대여목록}
- 소단위 명세서(Mini-Specification)
  - 자료 흐름도에서 어떤 일이 수행되는지를 정의하기 위해 각 처리들이 수행하는 업무를 상세하게 작성
- 개체 관계도(Entity Relationship Diagram, ERD)
  - 시스템에서 처리되는 개체(자료)와 개체의 구성과 속성, 개체 간의 관계를 표현하여 자료를 모델화하는데 사용
- 상태 전이도(State Transition Diagram, STD)
  - 시스템에 어떤 일이 발생할 경우 시스템의 상태와 상태 간의 전이를 모델화한 것

⑶ 객체 지향 분석 모델

① 객체 지향 분석

- 사용자의 요구사항을 분석하여 요구된 문제와 관련된 모든 클래스, 이와 연관된 속성과 연산, 그들 간의 관계 등을 정의하여 모델링하는 작업

② 객체지향 분석 방법론

- Rumbaugh(럼바우) 방법
  - 객체 모델링 (Object Modeling) : 객체 다이어그램
  - 동적 모델링 (Dynamic Modeling) : 상태 다이어그램
  - 기능 모델링 (Functional Modeling) : 자료 흐름도(DFD)
- Booch(부치) 방법
  - 미시적 개발 프로세스와 거시적 개발 프로세스를 모두 사용하는 분석 방법
- Jacobson 방법
  - Use case를 강조하여 사용하는 분석 방법
- Coad와 Yourdon 방법
  - E-R 다이어그램을 사용하는 기법
- Wirfs-Brock 방법
  - 분석과 설계간 구분 없음

03 소프트웨어 설계

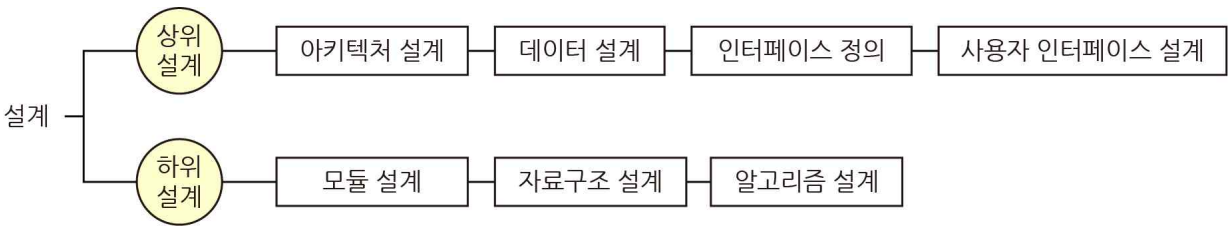
Section 1. 소프트웨어 설계의 기본 원칙

1. 소프트웨어 설계

(1) 소프트웨어 설계의 개념

- 요구사항 명세서를 참조하여 소프트웨어의 구체적인 설계서를 작성하는 단계

(2) 소프트웨어 설계의 종류



(3) 소프트웨어 설계의 원리

- 분할과 정복(Divide & Conquer)
- 추상화(Abstraction)
  - 추상화 기법

추상화 기법	설명
과정 추상화	자세한 단계를 고려하지 않고 상위 수준에서 수행 흐름만 먼저 설계
데이터 추상화	데이터 구조를 대표할 수 있는 표현으로 대체하는 것이다.
제어 추상화	여러 명령들을 간단한 표현으로 대체하는 것이다.

- 단계적 분해(Gradual Decomposition)
- 모듈화(Modulization)
- 정보은닉(Information Hiding)

2. 설계 모델링

(1) 설계 모델링 개념

- 소프트웨어를 구성하는 모듈들을 식별하고, 이것들의 연결을 그림으로 표현한 것

(2) 설계 모델링 유형

- 구조 모델링
  - 소프트웨어를 구성하는 컴포넌트들의 유형, 인터페이스, 내부 설계 구조 및 이들의 연결 구조를 모델링
  - UML 정적 다이어그램
- 행위 모델링
  - 소프트웨어의 구성요소들의 기능들이 언제, 어떠한 순서로 기능을 수행해야 작용하는지를 모델링
  - UML 동적 다이어그램

⑶ 소프트웨어 설계 절차 및 유형

- 아키텍처 설계
- 데이터베이스 설계
- 서브시스템 설계
- 컴포넌트 설계
- 자료구조와 알고리즘 설계
- 협약에 의한 설계

  - 선행 조건, 결과 조건, 불변 조건

Section 2. 소프트웨어 아키텍처

1. 소프트웨어 아키텍처

(1) 소프트웨어 아키텍처(SoftWare Architecture) 개념

- 소프트웨어의 골격이 되는 기본구조

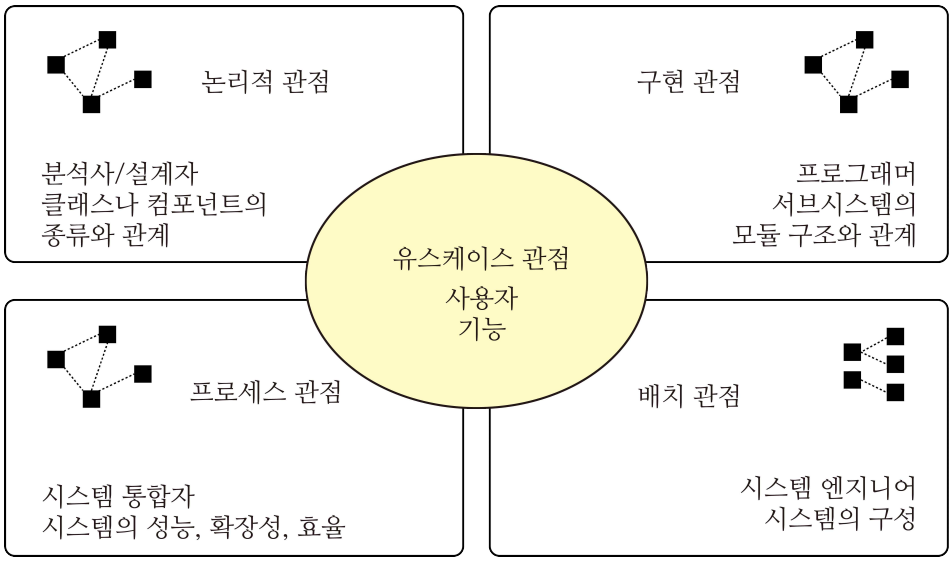
(2) 소프트웨어 아키텍처의 특징

- 간략성
- 추상화
- 가시성
- 관점 모형
- 의사소통수단

(3) 소프트웨어 아키텍처 프레임워크 구성요소

- 아키텍처 명세서 (Architecture Description)
- 이해관계자 (Stakeholder)
- 관심사 (Concerns)
- 관점 (Viewpoint)
- 뷰(View)

(4) 소프트웨어 아키텍처 4+1 뷰



(5) 소프트웨어 아키텍처 품질속성

- 정확성 (Correctness)
- 신뢰성 (Reliability)
- 효율성 (Efficiency)
- 무결성 (Integrity)
- 사용 용이성 (Usability)
- 유지보수성 (Maintainability)
- 시험 용이성 (Testability)
- 유연성 (Flexibility)
- 이식성 (Potability)
- 재사용성 (Reusability)
- 상호 운용성 (Interoperability)

(6) 소프트웨어 아키텍처 평가

관점	유형	내용
가시성	가시적 평가	Inspection, Review, Validation & Verification
	비가시적 평가	SAAM, ATAM, CBAM, ARID, ADR
시점	이른 평가	아키텍처 구축과정 중 어느 때나 평가 가능
	늦은 평가	기존 시스템의 요구사항에 대한 아키텍처의 적합성을 판단할 때 사용

2. 소프트웨어 아키텍처 패턴

(2) 소프트웨어 아키텍처 패턴 종류

- 계층화 패턴(Layered pattern)
- 클라이언트-서버 패턴(Client-Server Pattern)
- 마스터-슬레이브 패턴(Master-Slave Pattern)
- 파이프-필터 패턴(Pipe-Filter Pattern)
- 브로커 패턴(Broker Pattern)
- 피어 투 피어 패턴(Peer to Peer Pattern)
- 이벤트-버스 패턴(Event-Bus Pattern)
- 모델-뷰-컨트롤러 패턴(Model-View-Controller Pattern : MVC Pattern)
- 블랙보드 패턴(Blackboard Pattern)
- 인터프리터 패턴(Interpreter Pattern)

Section 3. UML

1. UML(Unified Modeling Language)

⑴ UML 개념

- 프로그램 설계를 표현하기 위해 사용하는 표기법

⑵ UML 특징

- 가시화 언어
- 명세화 언어
- 구축 언어
- 문서화 언어

2. UML 구성요소

⑴ 사물(Things)

- 구조사물
- 행동사물
- 그룹사물
- 주해사물

⑵ 관계(Relationships)

- 일반화 관계(Generalization)
  - 한 클래스가 다른 클래스를 포함하는 상위 개념일 때의 관계
- 연관관계(Accociation)
  - 2개 이상 사물이 서로 관련된 관계
- 의존관계(Dependency)
  - 연관 관계와 같이 한 클래스가 다른 클래스에서 제공하는 기능을 사용할 때 표시
  - 연관 관계와 차이점은 두 클래스의 관계가 한 메서드를 실행하는 동안과 같이 매우 짧은 시간만 유지
- 실체화 관계 (Realization)
  - 인터페이스를 구현받아 추상 메서드를 오버라이딩 하는 것을 의미
- 집합 관계 - 집약관계 (Aggregation)
  - 전체 객체가 사라진다 해도 부분 객체는 사라지지 않음
- 집합관계 - 합성관계 (Composition)
  - 전체 객체가 없어지면 부분 객체도 없어짐

⑶ 다이어그램(Diagram)

- 구조 다이어그램
  - 클래스 다이어그램
  - 객체 다이어그램
  - 컴포넌트 다이어그램
  - 배치 다이어그램
  - 복합체 다이어그램
  - 패키지 다이어그램
- 행위 다이어그램
  - 유스케이스 다이어그램
  - 시퀀스 다이어그램
  - 커뮤니케이션 다이어그램
  - 상태 다이어그램
  - 활동 다이어그램
  - 상호작용 다이어그램
  - 타이밍 다이어그램

04 화면 설계

Section 1. UI 설계

1. UI (User Interface) 개념

(1) UI 개념

- 컴퓨터, 웹 사이트, 시스템 등의 정보기기와 사용자가 서로 상호작용을 할 수 있도록 연결해주는 매개체

(2) UX (User eXperience) 개념

- 사용자가 컴퓨터, 웹 사이트, 시스템 등 정보기기의 UI를 직/간접적으로 이용하여 경험한 모든 것

(3) UI 유형

- CLI (Command Line Interface)
- GUI (Graphical User Interface)
- AUI (Auditory User Interface)
- NUI (Natural User Interface)

2. UI 설계

(1) UI 요구사항 구분

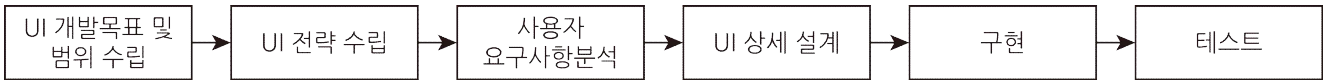
① 기능적 요구사항

- 시스템이 제공해야 하는 기능에 대한 요구사항

② 비기능적 요구사항

- 사용성, 효율성, 신뢰성, 유지 보수성, 재사용성 등 품질에 관한 요구사항

(2) UI 설계 절차



① UI 개발목표 및 범위 수립

② UI 전략 수립

③ 사용자 요구사항 분석

④ UI 상세 설계

⑤ 구현

⑥ 테스트

⑶ UI 설계 원칙

- 직관성
  - 누구나 쉽게 이해하고 사용할 수 있어야 한다.
- 유효성
  - 사용자의 목적을 정확하게 달성해야 한다.
- 학습성
  - 누구나 쉽게 배우고 익힐 수 있어야 한다.
- 유연성
  - 사용자의 요구사항을 최대한 수용하며, 오류를 최소화해야 한다.

⑷ UI 설계 도구

① 와이어프레임(Wireframe)

② 스토리보드

③ 프로토타입

3. 감성공학

⑴ 감성공학의 개념

- 인간의 심상을 구체적인 물리적 설계 요소로 번역하여 이를 실현하는 기술
- 요소화 → 형상화 → 구현 → 생산

⑵ 제품과 관련된 인간의 감성

- 감각적 감성
- 기능적 감성
- 문화적 감성

⑶ 감성공학의 접근 방법

- 1류 접근 방법
  - 인간의 감성을 표현하는 어휘를 이용하여 제품에 대한 이미지를 조사하고, 그 분석을 통해 제품 디자인 요소 연계
- 2류 접근 방법
  - 문화적 감성의 일부를 반영한 개념
- 3류 접근 방법
  - 특정 시제품을 사용하여 감각 척도를 계측하고, 정량화된 값을 환산

4. UI 설계 지침

(1) 한국 HCI 연구회 설계 지침

- 가시성의 원칙 (Visibility)
  - 소프트웨어의 기능을 노출 시켜 최대한 조작이 쉽도록 한다.
- 조작 결과 예측의 원칙 (Natural Mapping)
  - 사용자가 소프트웨어를 조작하여 작동시킨 결과를 조작 부위만 보고도 예측 가능하게 설계 한다.
- 일관성의 원칙 (Consistency)
  - 소프트웨어의 조작방식에 일관성을 제공하여 사용자가 쉽게 기억하고 빠르게 적응할 수 있게 한다.
- 단순성의 원칙 (Simplicity)
  - 소프트웨어의 기능구조를 단순화시켜 조작에 요구되는 노력을 최소화한다.
- 지식 배분의 원칙 (Knowledge in World & Head)
  - 학습하기 쉽고 기억하기 쉽게 설계해야 한다.
- 조작오류의 원칙 (Design for Error)
  - 사용간 발생한 오류를 쉽게 발견하고 수정 또한 쉽게 이루어져야 한다.
- 제한사항 선택사용의 원칙 (Constraints)
  - 소프트웨어를 조작할 때 선택의 여지를 줄여 조작 방법이 명확하도록 유도한다.
- 표준화의 원칙 (Standardization)
  - 소프트웨어의 기능 구조와 디자인을 한 번 학습한 이후 보다 효과적으로 서비스를 사용할 수 있어야 한다.
- 행동 유도성의 원칙 (Affordance)
  - 사용자가 디자인을 보고 기능 및 조작법을 유추할 수 있도록 해야 한다.
- 접근성의 원칙 (Accessibility)
  - 사용자의 성별, 연령등 다양한 계층의 사용자가 받아드릴 수 있는 사용자인터페이스를 구축해야 한다.

Section 2. UI 구현

1. 화면 레이아웃 구성

⑴ 레이아웃(Layout)의 개념

- 특정 공간에 여러 구성 요소를 보기 좋게 효과적으로 배치하는 작업

⑵ HTML5

① HTML5 개념

- 월드와이드웹(World Wide Web)을 통해 제공되는 정보를 나타낼 목적으로 사용되는 마크업 언어

② 시맨틱 요소

- header
- nav
- section
- article
- aside
- footer

③ INPUT 요소

- 텍스트 입력 (text, textarea)
  - text : 한 줄의 텍스트 입력
  - textarea : 여러 줄의 텍스트 입력
- 비밀번호 입력 (password)
  - 입력받은 문자를 별표나 작은 원으로 표시
- 라디오 버튼(radio)
  - 여러 개의 라디오 버튼 옵션 중에서 단 하나만의 값을 선택
- 체크 박스 (checkbox)
  - 여러 개의 체크박스 중에서 여러 개의 옵션 값을 선택
- 파일 선택 (file)
  - 사용자 컴퓨터의 파일을 입력
- 선택 입력 (select)
  - 여러 개의 드롭다운 리스트(drop-down-list) 중에서 한 개의 옵션을 선택
- 버튼 (button)
  - 사용자가 클릭했을 때 작업을 수행
- 전송 (submit)
  - 입력 받은 데이터를 서버로 전송
- 필드셋 (fieldset)
  - 관련된 데이터를 하나로 묶어준다.

⑶ CSS(Cascading Style Sheet)

① CSS 개념

- HTML과 함께 웹을 구성하는 기본 프로그래밍 요소

② CSS 특징

- HTML로 부터 디자인적인 요소를 분리해 정의할 수 있다.

⑷ JavaScript

① JavaScript 개념

- 모질라 재단의 프로토타입 기반의 프로그래밍 언어로, 스크립트 언어에 해당된다.
- 클라이언트 단에서 웹 페이지가 동작하는 것을 담당

② JavaScript 프레임워크

- React
  - 유저 인터페이스를 만드는 데 사용되는 오픈 소스 자바스크립트 라이브러리
  - 페이스북에서 개발
  - 싱글 페이지 애플리케이션(SPA)이나 모바일 애플리케이션 개발에 사용될 수 있다.
- Vue.js
  - 자바스크립트로 개발된 컴포넌트 구조 기반 프론트엔드 프레임워크
  - 고성능의 싱글 페이지 애플리케이션(SPA)을 구축하는데 이용가능하다.
  - Evan You에 의해 개발
- AngularJS
  - 자바스크립트 기반의 오픈 소스 프론트엔드 웹 애플리케이션 프레임워크
  - 구글에서 개발
- Ajax(Asynchronous JavaScript and XML)
  - 비동기적인 웹 애플리케이션의 제작을 위한 웹 개발 기법

2. UI 관련용어

용어	설명
웹 표준	월드 와이드 웹의 측면을 서술하고 정의하는 공식 표준이나 다른 기술 규격을 말한다.
웹 호환성	이용자의 단말기 (PC, 모바일 기기 등)의 하드웨어 및 소프트웨어 환경이 다른 경우에도 동등한 서비스를 제공할 수 있는 것을 말한다.
웹 접근성	장애인과 비장애인 모두가 동등하게 웹 사이트에 접근하여 이용할 수 있도록 보장하는 방식을 말한다.
반응형 웹	PC, Mobile 등 다용한 디바이스에서 화면크기에 맞춰 하나의 사이트를 보여준다.
인포그래픽 (Infographic)	정보(Information)와 그래픽(Graphic)의 합성어로, 복잡한 정보를 쉽고 빠르게 전달하기 위해 정보를 분석, 정리하여 차트, 그래프, 아이콘, 그래픽스, 이미지 등을 활용하여 시각화한 것을 말한다.
브랜드 아이덴티티(BI)	사용자에게 전달하고자 하는 특정 브랜드의 가치와 의미를 반영한 심적 표상을 말한다.
네비게이션 (Navigation)	하이퍼링크를 따라 웹 공간의 정보를 요청하고 받아오는 웹 브라우징을 의미하며, 웹 사이트를 탐색하기 위한 도구를 뜻하기도 한다.
아코디언(Accordion)	사용자가 원하는 정보만 선택적으로 볼 수 있게 접을 수 있는 내용 패널을 말한다.
플레이스 홀더 (Placeholder)	사용자가 값을 입력하는 데 참고할 수 있도록, 입력 필드에 제공되는 간략한 텍스트 도움말을 말한다.
필터링(Filtering)	원하지 않는 데이터를 차단하거나, 원하는 데이터만 볼 수 있도록 해주는 기능을 말한다.
입력 폼(Input Form)	다양한 입력 필드로 구성되어, 사용자가 웹 서버로 전송할 정보를 입력 할 수 있는 웹 문서의 일부를 말한다.
입력 필드(Input Field)	사용자가 정보를 입력하거나 선택하는데 이용되는 사용자 인터페이스 요소를 말한다.
썸네일(Thumbnail)	커다란 이미지를 축소하여 제공한 이미지를 말한다.
레이블(Label)	입력폼을 구성하는 다양한 입력 필드를 식별하기 위해 사용하는 명칭을 말한다.
대체 텍스트 (Alternative Text)	콘텐츠를 대신하기 위해 제공되는 텍스트를 의미한다.
초점(Focus)	웹 페이지에서 사용자가 선택한 해당 요소에 있을 때, 해당 요소에 Focus가 있다고 한다.

05 서버 프로그램 구현

Section 1. 개발 환경 구축

1. 서버 환경 구축

(1) 웹 서버 (WEB)

- 클라이언트에게 정적 파일(HTML, CSS, JS, 이미지)을 제공하는 웹서버 어플리케이션이 설치된 하드웨어
- Apache Web Server, IIS, nginx, GWS 등

(2) 웹 어플리케이션 서버 (WAS)

- 동적인 웹 서비스를 제공하기 위한 미들웨어가 설치된 하드웨어
- Web Logic, Web Spere, Jeus, Tomcat 등

(3) 데이터베이스 서버 (DBMS)

- 데이터의 저장과 관리를 위한 데이터베이스 소프트웨어가 설치된 하드웨어
- Oracle, MySQL, MS-SQL 등

(4) 파일서버

- 사용자의 파일을 저장하고, 파일을 공유할 목적으로 구성된 하드웨어

(5) Load Balancer

- 여러 대의 서버가 존재할 경우 요청을 적절히 분배해주는 역할
- 분배 방식 : Random, Least loaded, Round Robin

(6) CDN(Content Delivery Network)

- 용량이 큰 컨텐츠 데이터(이미지, 비디오 등)를 빠른 속도로 제공하기 위해 사용자와 가까운 곳에 분산되어 있는 데이터 저장 서버

(7) 시스템 아키텍처 고려사항

- 확장성 (Scalability)
- 성능 (Performance)
- 응답 시간 (Latency)
- 처리량 (Throughput)
- 접근성 (Availability)
- 일관성 (Consistency)

2. 개발 소프트웨어 환경

(1) 시스템 소프트웨어

① 운영체제(OS, Operation System)

② JVM(Java Virtual Machine)

③ Web Server

④ WAS(Web Application Server)

⑤ DBMS(Database Management System)

(2) 개발 소프트웨어

① 요구사항 관리 도구

- 고객의 요구사항을 수집, 분석, 추적을 쉽게 할 수 있도록 지원한다.

② 설계/모델링 도구

- 기능을 논리적으로 표현할 수 있는 통합 모델링 언어(UML) 지원

③ 구현도구

- 소프트웨어 언어를 통해 구현 및 개발을 지원하는 도구

④ 테스트 도구

- 개발된 모듈들에 대하여 요구 사항에 적합하게 구현되어 있는지 테스트를 지원하는 도구

⑤ 형상관리 도구

- 산출물 및 소스코드의 변경 사항을 버전별로 관리하여, 목표 시스템의 품질 향상을 지원하는 도구
- Git, CVS, SVN 등

3. IDE(Integrated Development Environment) 도구

(1) IDE 도구의 개념

- 소프트웨어 개발에 필요한 많은 도구의 기능을 하나로 묶어 활용하는 소프트웨어

(2) IDE 도구의 기능

- 텍스트 에디터, 컴파일, 디버거, 배포 등

(3) IDE 도구 선정시 고려 사항

기준	설명
적정성	- 대상 업무에 적절한 도구 선정
효율성	- 프로그래밍의 효율성 고려
이식성	- 여러 OS에 개발환경 설치 가능
친밀성	- 프로그래머가 익숙한 언어 및 도구
범용성	- 다양한 개발 사례가 존재

4. 협업 도구

(1) 협업 도구의 개념

- 여러 사용자가 각기 별개의 작업 환경에서 통합된 하나의 프로젝트를 동시에 수행할 수 있도록 도와주는 소프트웨어

(2) 협업 도구의 기능

- 전사관리 : 전자결재, 조직도 등
- 프로젝트 관리 : 캘린더, 타임라인, 간트차트, 대시보드 등
- 자체 드라이브 공간
- 문서 공유 지원
- 커뮤니케이션
- 다국어지원
- 타 협업툴간 연동 지원

5. 형상 관리 도구

(1) 형상 관리 도구의 개념

- 소프트웨어 생명주기 동안 발생하는 변경사항을 통제하기 위한 관리 방법

(2) 형상 관리의 필요성

- 개발 도중 소스코드를 이전 상태로 되돌릴 필요가 있을 경우
- 각 변경점에 대한 이력 확인
- 여러 개발자의 동시 개발에 따른 충돌 해결
- 버그 및 문제점 발생시 추적이 용이
- 기타 산출물의 이력관리도 용이

(3) 변경 관리/버전 관리/형상 관리

① 변경 관리

- 소스의 변경 상황을 관리

② 버전 관리

- 변경을 관리하기 위한 효과적인 방법

③ 형상 관리

- 변경 관리와 버전 관리가 포함되고, 프로젝트 진행상황, 빌드와 릴리즈까지 모두 관리할 수 있는 통합 시스템

(4) 형상 관리 절차

- 형상 식별
  - 형상 관리의 시작으로 시스템을 구성하는 요소들 중 형상 관리의 대상들을 구분하고 관리 목록의 번호를 정의하여 부여한다.

- 형상 통제
  - 소프트웨어 형상 변경 요청을 검토하고 승인하여 현재의 베이스라인에 반영될 수 있도록 통제
  - 형상통제가 이루어지기 위해서는 형상 통제 위원회(Configuration Control Board, CCB)의 승인을 통한 변경 통제가 이루어져야 한다.
- 형상 감사
  - 형상 항목의 변경이 계획에 따라 제대로 이뤄졌는지를 검토하고 승인
- 형상 기록/보고
  - 프로젝트 팀, 회사, 클라이언트 등에게 소프트웨어 개발 상태에 대한 보고서를 제공

6. 버전 관리 도구

(1) 소프트웨어 버전 관리 도구 개념

- 동일한 소스 코드에 대한 여러 버전을 관리하는 것

(2) 소프트웨어 버전 관리 도구 유형

① 공유 폴더 방식 (RCS, SCCS)

- 매일 개발 완료 파일은 약속된 위치의 공유 폴더에 복사

② 클라이언트/서버 방식 (CVS, SVN)

- 중앙에 버전 관리 시스템이 항시 동작

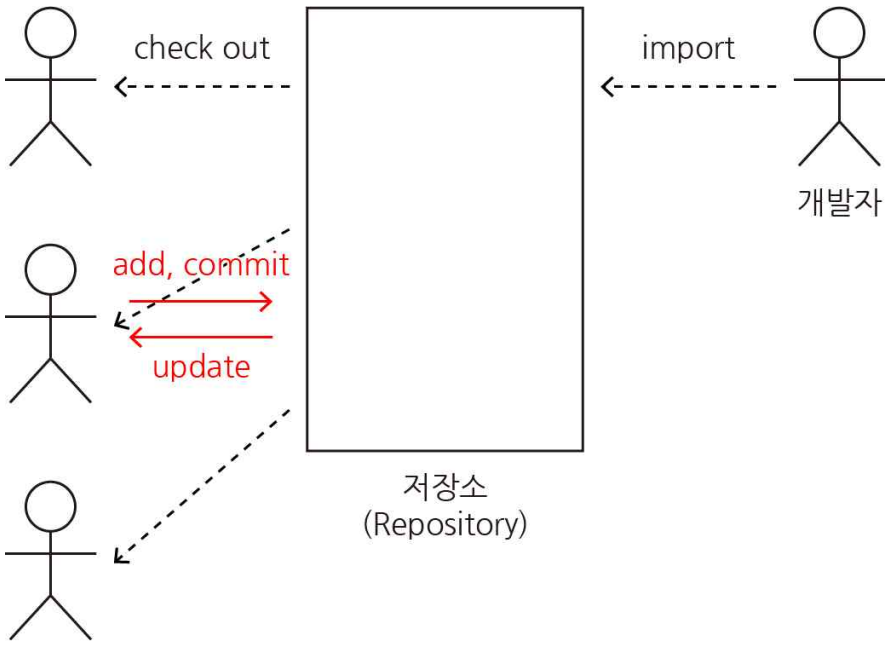
③ 분산 저장소 방식 (Git, Betkeeper)

- 로컬 저장소와 원격저장소 구조

(3) 버전 관리 도구별 특징

- CVS
  - 오랜 기간 사용된 형상 관리 도구로, 다양한 운영체제를 지원
- SVN
  - CVS의 단점을 보완하기 위해 만들어졌다.
  - 최초 1회에 한해 파일 원본을 저장하고, 그 이후에는 실제 파일이 아닌 원본과 차이점을 저장하는 방식
  - 언제든지 원하는 시점으로 복구가 가능
- Git
  - 리누스 토발즈가 리눅스 커널의 개발을 위해 만들었다.
  - 원격 서버 Git Repository에 push 하지 않은 채 여러 branch 생성이 가능하다.
- Clear Case
  - IBM에서 개발된 유료 버전의 형상 관리 툴
  - 서버가 부족할 때 서버를 하나씩 늘려 확장할 수 있다.
- BitKeeper
  - SVN과 비슷한 중앙 통제 방식으로 대규모 프로젝트에서 빠른 속도를 내도록 개발된 버전관리 도구
- RCS(Revision Control System)
  - 소스 파일의 수정을 한 사람만으로 제한하여 다수의 사람이 파일의 수정을 동시에 할 수 없도록 파일을 잠금하는 방식으로 버전 컨트롤을 수행

(4) 버전 관리 소프트웨어 사용 방식



(5) 버전 관리 주요 용어

용어	설명
Repository	저장소
Checkout	Repository에서 로컬로 프로젝트를 복사
Commit	로컬의 변경된 내용을 Repository에 저장
Update	Repository에 있는 내용을 로컬에 반영
Add	로컬에서 새로운 파일이 추가되었을 때 Repository에 등록
Trunk	Root 프로젝트
Branch	Root 프로젝트에서 파생된 프로젝트
Merge	Branch에서 진행하던 작업을 Root 프로젝트와 합침
Diff	파일의 비교

7. 빌드 도구

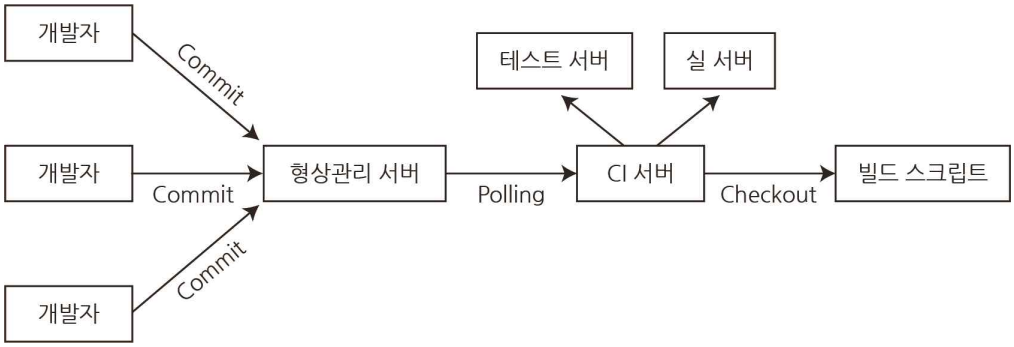
(1) 빌드의 개념

- 소스코드 파일들을 컴퓨터에서 실행할 수 있는 소프트웨어로 변환하는 일련의 과정

(2) 빌드 자동화 도구 특징

- 빌드, 테스트, 배포를 자동으로 수행하는 도구

(3) 빌드 자동화 프로세스



(4) 빌드 자동화 도구 종류

- Make
  - 유닉스 계열 운영체제에서 주로 사용되는 프로그램 빌드 도구이다.
- Ant
  - Java 기반의 빌드 도구로 다른 빌드 도구 보다 역사가 오래 되었다.
  - 개발자가 원하는 형태로 개발을 할 수 있다는 유연성에 장점이 있다.
  - XML 기반의 빌드 스크립트로 개발한다.
  - 스크립트의 재사용이 어렵다.
  - Remote Repository를 가져올 수 없다.
- Maven
  - 프로젝트에 필요한 모든 Dependency를 리스트 형태로 Maven에게 알려 관리할 수 있도록 돕는 방식이다.
  - 필요한 라이브러리를 특정 파일(pom.xml) 에 정의해 놓으면 해당 라이브러리와 관련된 라이브러리까지 네트워크를 통해 자동으로 다운받는다.
  - 정해진 라이프사이클에 의하여 작업 수행하며, 전반적인 프로젝트 관리 기능까지 포함한다.
- Jenkins
  - Java 기반의 오픈소스로, 소프트웨어 개발 시 지속적 통합(continuous integration) 서비스를 제공하는 툴
  - 서블릿 컨테이너에서 실행되는 서버 기반 도구
  - SVN, Git 등 대부분의 형상 관리 도구와 연동이 가능
  - 여러 대의 컴퓨터를 이용한 분산 빌드나 테스트가 가능
- Gradle
  - Groovy를 기반으로 한 오픈 소스 형태의 자동화 도구로 안드로이드 앱 개발 환경에서 사용

Section 2. 개발 프레임워크

1. 프레임워크의 개념

- 소프트웨어 개발에 공통적으로 사용되는 구성 요소와 아키텍처를 일반화하여 손쉽게 구현할 수 있도록 여러 가지 기능들을 제공해주는 반제품 형태의 소프트웨어

2. 프레임워크의 특징

- 모듈화 (modularity)
  - 프레임워크는 구현을 인터페이스 뒤에 감추는 캡슐화를 통해서 모듈화를 강화
  - 설계와 구현의 변경에 따르는 영향을 최소화시킴으로써 쉽게 소프트웨어의 품질을 향상시킴
- 재사용성 (reusability)
  - 프레임워크가 제공하는 인터페이스는 여러 어플리케이션에서 반복적으로 사용할 수 있는 일반적인 컴포넌트를 정의할 수 있게 함으로써 재사용성을 높여준다.
  - 소프트웨어의 품질, 성능, 신뢰성, 상호 운용성을 향상시키고, 프로그래머의 생산성을 높여준다.
- 확장성(extensibility)
  - 다형성(polymorphism)을 통해 애플리케이션의 프레임워크의 인터페이스를 확장할 수 있게 한다.
- 제어의 역흐름(inversion of control)
  - 프레임워크가 외부의 이벤트에 대해 애플리케이션이 어떠한 메소드들을 수행해야 하는지 결정

3. 라이브러리(Library)

- 컴퓨터 프로그램에서 빈번하게 사용되는 사전 컴파일된 루틴 또는 리소스(클래스, 템플릿, 설정 데이터 등)를 모아둔 것

4. API(Application Programming Interface)

- 일종의 소프트웨어 인터페이스이며 다른 종류의 소프트웨어에 서비스를 제공한다.

Section 3. 모듈 구현

1. 단위 모듈 구현

⑴ 단위 모듈 구현의 개념

- 소프트웨어를 기능 단위로 분해하여 구현하는 기법

⑵ 효과적인 모듈화

- 결합도를 줄이고 응집도를 높여 모듈의 독립성을 높임
- FAN-OUT 최소화, FAN-IN 증가

⑶ 단위 모듈 설계의 원리

- 단계적 분해
  - 처음엔 간단히 작성하고, 점점 세밀하게 작성
- 추상화
  - 복잡한 문제를 일반화하여, 쉽게 이해할 수 있도록 한다.
- 독립성
  - 모듈은 응집도는 높이고 결합도는 낮춰 독립성을 가져야 한다.
- 정보은닉
  - 모듈 내부의 데이터를 외부에 은폐
- 분할과 정복
  - 큰 문제를 작게 나누어 하나씩 해결

⑷ 단위 모듈 작성 원칙

- 정확성 (Correctness)
  - 해당 기능이 실제 시스템 구현 시 필요한지 여부를 알 수 있도록 정확하게 작성
- 명확성 (Clarity)
  - 해당 기능에 대해 일관되게 이해되고 한 가지로 해석될 수 있도록 작성
- 완전성 (Completeness)
  - 시스템이 구현될 때 필요하고 요구되는 모든 것을 기술
- 일관성 (Consistency)
  - 공통 기능들 간에 상호 충돌이 없도록 작성
- 추적성 (Traceability)
  - 공통 기능에 대한 요구사항 출처와 관련 시스템 등의 유기적 관계에 대한 식별이 가능하도록 작성

2. 결합도

⑴ 결합도(Coupling)의 개념

- 어떤 모듈이 다른 모듈에 의존하는 정도

⑵ 결합도 유형

구분	설명
자료 결합도 (Data Coupling)	모듈 간의 인터페이스로 값이 전달되는 경우
스탬프 결합도 (Stamp Coupling)	모듈 간의 인터페이스로 배열이나 오브젝트, 스트럭처 등이 전달되는 경우
제어 결합도 (Control Coupling)	단순 처리할 대상인 값만 전달되는 게 아니라 어떻게 처리를 해야 한다는 제어 요소가 전달되는 경우
외부 결합도 (External Coupling)	어떤 모듈에서 선언한 데이터(변수)를 외부의 다른 모듈에서 참조하는 경우
공통 결합도 (Common Coupling)	파라미터가 아닌 모듈 밖에 선언되어 있는 전역 변수를 참조하고 전역 변수를 갱신하는 식으로 상호 작용하는 경우
내용 결합도 (Content Coupling)	다른 모듈 내부에 있는 변수나 기능을 다른 모듈에서 사용하는 경우

3. 응집도

⑴ 응집도(Cohesion)의 개념

- 모듈의 독립성을 나타내는 개념으로, 모듈 내부 구성요소 간 연관 정도

⑵ 응집도 유형

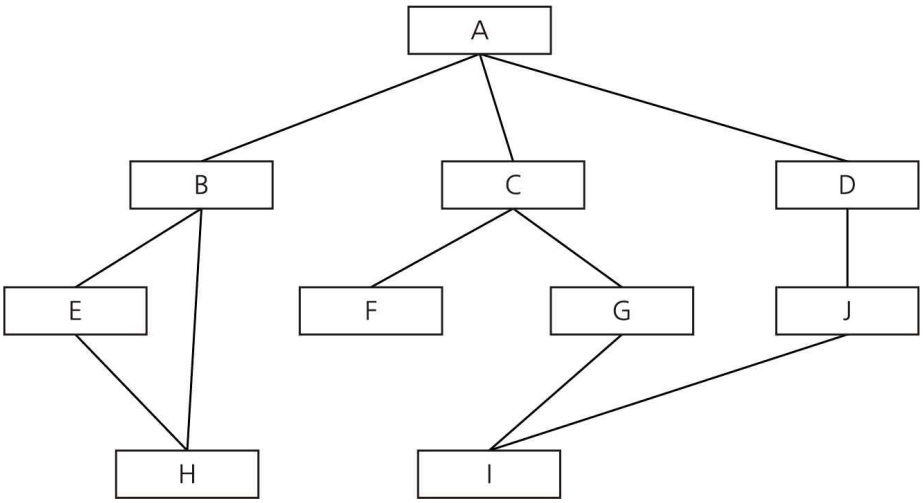
구분	설명
기능적 응집도 (Functional Cohesion)	모듈 내부의 모든 기능이 단일한 목적을 위해 수행되는 경우
순차적 응집도 (Sequential Cohesion)	모듈 내에서 한 활동으로부터 나온 출력값을 다른 활동이 사용할 경우
통신적 응집도 (Communication Cohesion)	동일한 입력과 출력을 사용하여 다른 기능을 수행하는 활동들이 모여 있을 경우
절차적 응집도 (Procedural Cohesion)	모듈이 다수의 관련 기능을 가질 때 모듈 안의 구성 요소들이 그 기능을 순차적으로 수행할 경우
시간적 응집도 (Temporal Cohesion)	연관된 기능이라기보다는 특정 시간에 처리되어야 하는 활동들을 한 모듈에서 처리할 경우
논리적 응집도 (Logical Cohesion)	유사한 성격을 갖거나 특정 형태로 분류되는 처리 요소들이 한 모듈에서 처리되는 경우
우연적 응집도 (Coincidental Cohesion)	모듈 내부의 각 구성 요소들이 연관이 없을 경우

4. 팬인(Fan-in), 팬아웃(Fan-out)

⑴ 팬인(Fan-in), 팬아웃(Fan-out)의 개념

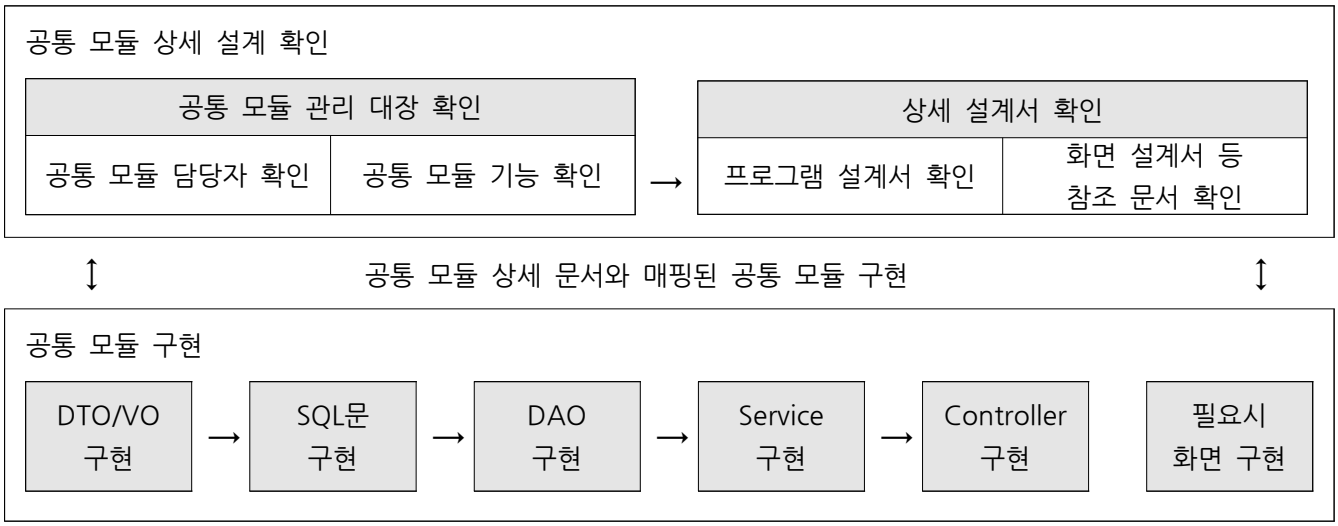
- 소프트웨어의 구성요소인 모듈을 계층적으로 분석하기 위해 활용

⑵ 팬인/팬아웃 계산법



5. 공통 모듈 구현

⑴ 공통 모듈 구현 순서



⑵ 공통 모듈 구현요소

구현 요소	설명
DTO (Data Transfer Object)	- 프로세스 사이에서 데이터를 전송하는 객체 - Getter, Setter 메서드만 포함한다.
VO (Value Object)	- 도메인에서 속성들을 묶어서 특정 값을 나타내는 객체 - DTO와 동일한 개념이나 차이점은 Read-Only 속성 객체이다
DAO (Data Access Object)	- 실질적으로 DB에 접근하는 객체 - DataBase에 접근하기 위한 로직 & 비지니스 로직을 분리하기 위해 사용
Service	- DAO 클래스를 호출하는 객체
Controller	- 비즈니스 로직을 수행하는 객체

⑶ Annotation

- 사전적으로는 "주석"이라는 의미를 가지고 있다.
- 자바코드에 주석처럼 달아 특수한 의미를 부여한다.
- 컴파일 또는 런타임에 해석된다.

Section 4. 서버 프로그램 구현

1. 서버 프로그램 구현

⑴ 업무 프로세스

- 개인이나 조직이 한 개 이상의 자원 입력을 통해 가치 있는 산출물을 제공하는 활동

⑵ MVC 모델의 계층

① 프리젠테이션 계층(Presentation Layer)

- 사용자 인터페이스
- 사용자가 선택할 수 있는 기능 및 부가정보를 전달할 양식을 표현한다.
- JSP 와 JSTL 태그 라이브러리를 결합하는 방식(JAVA의 경우)

② 제어 계층(Control Layer)

- 프리젠테이션 계층과 비즈니스 로직 계층을 분리하기 위한 컨트롤러를 제공
- 어떤 요청이 들어왔을 때 어떤 로직이 처리해야 하는지를 결정한다.

③ 비즈니스 로직 계층(Business Logic Layer)

- 핵심 업무를 어떻게 처리하는지에 대한 방법을 구현 하는 계층
- 핵심 업무 로직의 구현과 그에 관련된 데이터의 적합성 검증, 트랜잭션 처리 등을 담당한다.

④ 퍼시스턴스 계층(Persistence Layer)

- 데이터 처리를 담당하는 계층
- 데이터의 생성/수정/삭제/선택(검색)과 같은 CRUD 연산을 수행한다.

⑤ 도메인 모델 계층(Domain Model Layer)

- 각 계층 사이에 전달되는 실질적인 비즈니스 객체
- 데이터 전송 객체(DTO) 형태로 개발자가 직접 제작해서 데이터를 넘기게 된다.

2. DBMS 접속기술

⑴ DBMS 접속기술의 개념

- 프로그램에서 DB에 접근하여 DML을 사용 가능하게 하는 기술

⑵ DBMS 접속기술 종류

① 소켓통신

- 응용프로그램과 DBMS가 주고 받는 통신

② Vender API

- DBMS 사에서 공개한 API를 이용해 DBMS와 통신

③ JDBC(Java DataBase Connectivity)

- Java 에서 DB에 접속하고 SQL문을 수행할 때 사용되는 표준 API

④ ODBC(Open DataBase Connectivity)

- 데이터베이스에 접근하기 위한 표준 규격
- 개발언어에 관계없이 사용할 수 있다.

3. ORM(Object-Relational Mapping) 프레임워크

(1) ORM 프레임워크의 개념

- 객체와 관계형 데이터베이스의 데이터를 자동으로 매핑(연결)해주는 것

(2) 매핑 기술 비교

- SQL Mapper
  - SQL을 명시하여 단순히 필드를 매핑 시키는 것이 목적
  - SQL 의존적인 방법
  - 종류 : iBatis, Mybatis, jdbc Templetes 등
- OR Mapping (=ORM)
  - 객체를 통해 간접적으로 데이터베이스를 다룬다.
  - ORM을 이용하면 SQL Query 가 아닌 직관적인 코드로 데이터를 조작할 수 있다.
  - 종류 : JPA(Java Persistent API),Hibernate

Section 5. 배치 프로그램 구현

1. 배치 프로그램

⑴ 배치의 개념

- 데이터를 일괄적으로 모아서 처리하는 대량의 작업을 처리

⑵ 배치 프로그램의 필수 요소

- 대용량 데이터
- 자동화
- 견고함
- 안정성
- 성능

⑶ 스케줄 관리 종류

① 크론탭 (crontab)

- UNIX, LINUX 계열에서 사용
- 크론탭 형식

분 시 일 월 요일 명령어

- 항목의 범위

필드	의미	범위
첫 번째	분	0 ~ 59
두 번째	시	0 ~ 23
세 번째	일	1 ~ 31
네 번째	월	1 ~ 12
다섯 번째	요일	0 ~ 6 (0:일요일, 1:월요일)
여섯 번째	명령어	실행할 명령

- 허용 특수문자

특수문자	설명
*	모든 값 (매시, 매일, 매주)
?	특정 값이 아닌 어떤 값이든 상관 없음
-	범위를 지정할 때 (12-14 : 12시부터 14시)
,	여러 값을 지정할 때 (12, 14 : 12시, 14시)
/	증분값, 즉 초기값과 증가치 설정 (*/20 : 매 20분 마다)

- 설정 예

형식	설명
* * * * * 명령	매분 실행
30 4 * * 0 명령	매주 일요일 4시 30분 실행
10-30 4 * * * 명령	매일 오전 4시 10분부터 30분까지 매분 실행
0,10,20 * * * * 명령	매일 매시간 0분, 10분, 20분 실행
*/30 * * * * 명령	매 30분 마다 실행
30 0 1 1,6 * 명령	1월과 6월, 1일, 0시 30분에 실행

② Spring Batch

- 백엔드의 배치처리 기능을 구현하는데 사용하는 프레임워크

③ Quartz Job Scheduler

- 표준 자바 프로그램으로 만들어진 작업을 지정된 일정에 따라 실행시키는데 사용하는 Java 패키지

06 인터페이스 구현

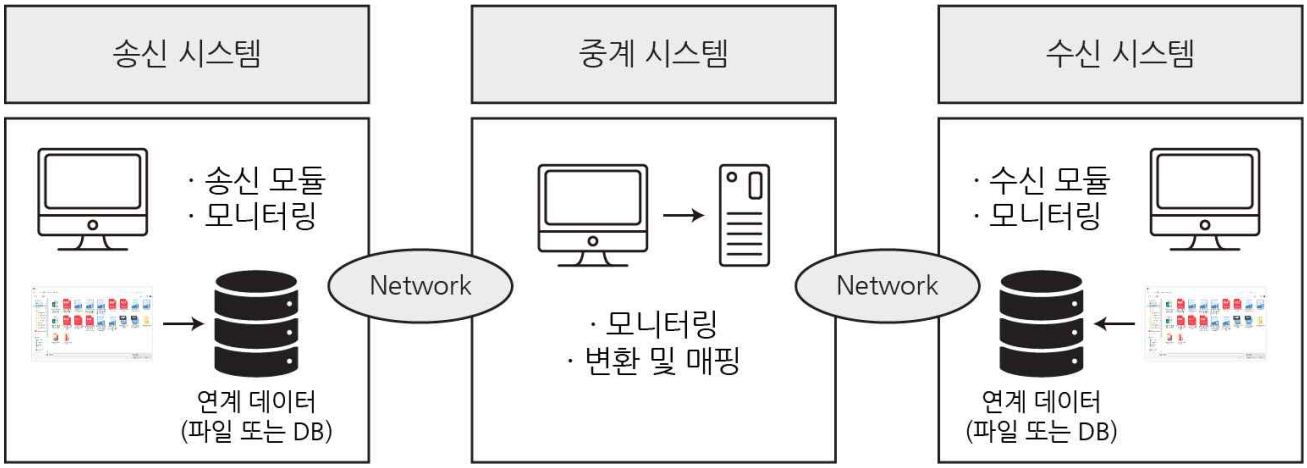
Section 1. 인터페이스 개요

1. 인터페이스 시스템

(1) 인터페이스 시스템의 개념

- 서로 다른 시스템, 장치 사이에서 정보나 신호를 주고 받을 수 있도록 도움을 주는 시스템

(2) 인터페이스 시스템 구성



2. 연계 시스템 분류와 데이터 식별

(1) 시스템 분류 체계

- 기업 내부에서 사용하고 있는 시스템 분류 체계를 기반으로 대내외 인터페이스 시스템의 식별자를 정의

(2) 연계 시스템 식별 정보

- 대내외 구분 정보
- 기관명
- 시스템 ID
- 한글명
- 영문명
- 시스템 설명
- 시스템 위치
- 네트워크 특성
- 전용 회선 정보
- IP/URL
- Port
- Login 정보
- DB 정보
- 담당자 정보

⑶ 송수신 데이터 식별

- 송수신 시스템 사이에서 교환되는 데이터는 규격화된 표준 형식에 따라 전송
- 송수신 전문 구성

구성	설명
전문 공통부	- 인터페이스 표준 항목을 포함 - 인터페이스 ID, 서비스 코드, 접속 IP 등
전문 개별부	- 업무처리에 필요한 데이터를 포함
전문 종료부	- 전송 데이터의 끝을 표시하는 문자 포함

Section 2. 인터페이스 설계서 확인

1. 인터페이스 설계서 구성

⑴ 인터페이스 목록

- 연계 업무와 연계에 참여하는 송수신 시스템의 정보, 연계 방식과 통신 유형 등에 대한 정보

⑵ 인터페이스 정의서

- 데이터 송신 시스템과 수신 시스템 간의 데이터 저장소와 속성 등의 상세 내역을 포함

Section 3. 인터페이스 기능 구현

1. 내·외부 모듈 연계 방식

⑴ EAI(Enterprise Application Integration)

① EAI의 개념

- 기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션들 간의 정보 전달, 연계, 통합을 가능하게 해주는 솔루션

② EAI의 구축유형

- Point-to-Point
- Hub & Spoke
- Message Bus (ESB 방식)
- Hybrid

⑵ ESB(Enterprise Service Bus)

- 웹 서비스 중심으로 표준화된 데이터, 버스를 통해 이 기종 애플리케이션을 유연(loosely-coupled)하게 통합하는 핵심 플랫폼(기술)

2. 인터페이스 연계 기술

- DB Link
  - 데이터베이스에 제공하는 DB Link 객체를 이용
- DB Connection
  - 수신 시스템의 WAS에서 송신 시스템 DB로 연결하는 DB Connection Pool을 생성하고 연계 프로그램에서 해당 DB Connection Pool명을 이용
- JDBC
  - 수신 시스템의 프로그램에서 JDBC 드라이버를 이용하여 송신 시스템 DB와 연결
- API / OpenAPI
  - 송신 시스템의 애플리케이션 프로그래밍 인터페이스 프로그램
- Web Service
  - WSDL(Web Services Description Language), UDDI(Universal Description, Discovery and Integration), SOAP(Simple Object Access Protocol) 프로토콜을 이용하여 연계
- Hyper Link
  - 웹 애플리케이션에서 하이퍼링크(Hyper Link) 이용
- Socket
  - 통신을 위한 소켓(Socket)을 생성하여 포트를 할당하고 클라이언트의 통신 요청 시 클라이언트와 연결하고 통신하는 네트워크 기술

3. 인터페이스 전송 데이터

(1) JSON (JavaScript Object Notation)

- Javascript 객체 문법으로 구조화된 데이터를 표현하기 위한 문자 기반의 표준 포맷
- JSON 문법

```
{
  "firstName": "Kwon",
  "lastName": "YoungJae",
  "email": "kyoje11@gmail.com",
  "hobby": ["puzzles","swimming"]
}
```

(2) XML (eXtensible Markup Language)

- 웹에서 구조화한 문서를 표현하고 전송하도록 설계한 마크업 언어
- XML 예제

```
<?xml version="1.0" encoding="UTF-8"?>
<information type="필기">
    <subject>
        <no>1</no>
        <name>소프트웨어설계</name>
        <point>80</point>
    </subject>
    <subject>
        <no>2</no>
        <name>소프트웨어개발</name>
        <point>60</point>
    </subject>
</shop>
```

(3) CSV (Comma Separated Values)

- 몇 가지 필드를 쉼표(,)로 구분한 텍스트 데이터 및 텍스트 파일
- 표 형태의 데이터를 저장하는 파일 형식
- CSV 예제

```
1,소프트웨어설계,80
2,소프트웨어개발,60
3,데이터베이스구축,85
```

4. 인터페이스 구현

(1) AJAX(Asynchronous JavaScript and XML)

① AJAX의 개념

- 자바스크립트를 이용해 서버와 브라우저가 비동기 방식으로 데이터를 교환할 수 있는 통신 기능

② 비동기 방식

- 웹페이지를 리로드하지 않고 데이터를 불러오는 방식

③ AJAX 예제

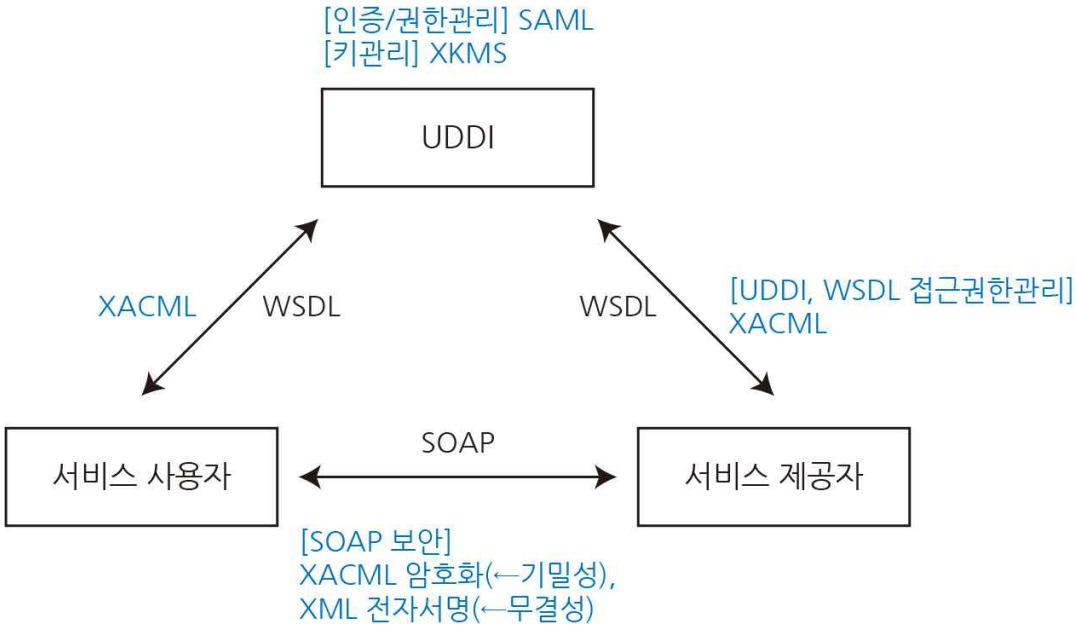
```
$.ajax({
    url: 'https://www.njobler.net/test',
    type: 'GET',
    success: function onData (data) {
        console.log(data);
    },
    error: function onError (error) {
        console.error(error);
    }
});
```

(2) SOAP(Simple Object Access Protocol)

① SOAP의 개념

- HTTP, HTTPS, SMTP 등을 통해 XML 기반의 메시지를 컴퓨터 네트워크 상에서 교환하는 프로토콜
- SOAP은 웹 서비스에서 기본적인 메시지를 전달하는 기반이 된다.

② SOAP 구성



③ SOAP 기본 구조

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
<soap:Fault>
...
</soap:Fault>
</soap:Body>

</soap:Envelope>
```

(3) REST

① REST의 개념

- HTTP URI를 통해 자원을 명시하고, HTTP Method(POST, GET, PUT, DELETE)를 통해 해당 자원에 대한 CRUD Operation을 적용하는 것을 의미한다.
- 자원 기반의 구조(ROA, Resource Oriented Architecture)의 개념으로 구현되었다.

② REST 구성요소

- 자원(Resource), URI
  - 서버에 존재하는 데이터의 총칭
  - 모든 자원은 고유의 URI를 가지며 클라이언트는 이 URI를 지정하여 해당 자원에 대해 CRUD 명령을 수행한다.
- 행위(Verb), Method
  - 클라이언트는 URI를 이용해 자원을 지정하고 자원을 조작하기 위해 Method를 사용한다.
  - HTTP 프로토콜에서는 GET , POST , PUT , DELETE 같은 Method를 제공한다.
- 표현(Representation)
  - REST에서 하나의 자원은 JSON , XML , TEXT , RSS 등 여러 형태의 Representation으로 나타낼 수 있다.

③ REST API 예제

구분	표현

강의 생성	HTTP POST, http://njobler.net/lecture
	{
	"lec":{
	"name":"정보처리실기"
	"price":"70,000"
	}
	}

강의 조회	HTTP GET, http://njobler.net/lecture/정보처리실기

강의 수정	HTTP PUT, http://njobler.net/lecture
	{
	"lec":{
	"name":"정보처리 실기"
	"price":"65,000"
	}
	}

강의 삭제	HTTP DELETE, http://njobler.net/lecture/정보처리실기

④ RESTful

- REST의 원리를 따르는 시스템

5. 인터페이스 보안

(1) 인터페이스 보안 취약점 분석

- 인터페이스의 보안 취약점을 분석
- 분석된 보안 취약점을 근거로 인터페이스 보안 기능을 적용

(2) 인터페이스 보안 기능 적용

① 네트워크 영역

- 인터페이스 송/수신 간 스니핑 등 이용한 데이터 탈취 및 변조 위협 방지 위해 네트워크 트래픽에 대한 암호화 설정
- IPSec, SSL, S-HTTP 등 다양한 방식으로 적용

② 애플리케이션 영역

- 시큐어코딩 가이드를 참조하여 애플리케이션 코드 상 보안 취약점을 보완하는 방향으로 보안 기능 적용

③ DB 영역

- DB, 스키마, 엔티티의 접근 권한과 프로시저, 트리거 등 DB 동작 객체의 보안 취약점에 보안 기능을 적용
- 민감 데이터를 암호화, 익명화 등을 통해 데이터 자체 보안 방안도 고려

Section 4. 인터페이스 구현 검증

1. 인터페이스 검증

(1) 인터페이스 구현 검증 도구

- xUnit
  - 다양한 언어를 지원하는 단위 테스트 프레임 워크
- STAF
  - 서비스 호출 및 컴포넌트 재사용 등 다양한 환경을 지원하는 테스트 프레임워크
- FitNesse
  - 웹 기반 테스트케이스 설계, 실행, 결과 확인 등을 지원하는 테스트 프레임워크
- NTAF
  - FitNesse의 장점과 STAF의 장점을 통합한 Naver의 테스트 자동화 프레임워크
- Selenium
  - 다양한 브라우저 및 개발 언어를 지원하는 웹 어플리케이션 테스트 프레임워크
- watir
  - Ruby를 사용하는 애플리케이션 테스트 프레임워크

(2) 인터페이스 구현 감시 도구

- APM(Application Performance Management)을 사용하여 동작상태 감시
- 데이터베이스, 웹 애플리케이션의 다양한 정보를 조회하고 분석하여 시각화 함
- 종류 : 스카우터(Scouter), 제니퍼(Jennifer) 등

2. 인터페이스 오류 처리

(1) 인터페이스 오류 발생 알림

- 화면을 통한 오류 메시지 표시
- 오류 발생시 SMS 발송
- 오류 발생시 이메일 발송

(2) 인터페이스 오류 발생 확인

- 인터페이스 오류 로그 확인
- 인터페이스 오류 테이블 확인
- 인터페이스 감시도구로 확인

07 객체지향 구현

Section 1. 객체지향 설계

1. 객체지향 (Object Oriented Programming-OOP)

(1) 객체지향 개념

- 현실 세계의 유형, 무형의 모든 대상을 객체(Object)로 나누고, 객체의 행동(method)과 고유한 값(Attribute)을 정의하여 설계하는 방법

(2) 객체지향 구성요소

- 클래스 (Class)
  - 유사한 종류의 유형/무형의 존재를 속성과 연산을 정의해서 만든 틀
- 객체 (Object)
  - 클래스의 인스턴스
- 속성 (Attribute)
  - 객체들이 가지고 있는 고유한 데이터를 단위별로 정의한 것
- 메서드 (Method)
  - 어떤 특정한 작업을 수행하기 위한 명령문의 집합
- 메시지 (Message)
  - 객체에게 어떤 행위를 하도록 지시

(3) 객체지향언어의 특징

- 캡슐화(Encapsulation)
- 정보은닉(Information Hiding)
- 상속(Inheritance)
- 다형성(Polymorphism)
- 추상화(Abstraction)

(4) 객체지향 설계원칙(SOLID)

- 단일 책임 원칙(SRP, Single responsibility principle)
  - 한 클래스는 하나의 책임만을 가져야한다.
- 개방 폐쇄 원칙(OCP, Open-closed principle)
  - 확장에는 열려 있고, 수정에는 닫혀 있어야 한다.
- 리스코프 치환 원칙(LSP, Liskov substitution principle)
  - 자식 클래스는 언제나 자신의 부모 클래스를 대체할 수 있어야 한다.
- 인터페이스 분리 원칙(ISP, Interface Segregation Principle)
  - 자신이 사용하지 않는 인터페이스는 구현하지 말아야 한다.
- 의존성 역전 원칙(DIP, Dependency Inversion Principle)
  - 의존 관계를 맺을 때 자주 변화하는 것보다, 변화가 거의 없는 것에 의존해야 한다.

2. 디자인패턴

(1) 디자인 패턴(Design Pattern) 개념

- 객체 지향 프로그래밍 설계를 할 때 자주 발생하는 문제들에 대해 재사용할 수 있도록 만들어놓은 패턴들의 모음

(2) 디자인 패턴 구조

- 패턴의 이름과 유형
- 문제 및 배경
- 솔루션
- 결과
- 사례
- 샘플 코드

(3) GoF 디자인 패턴

- GoF(Gang of Four) 의 디자인 패턴
  - 에리히 감마(Erich Gamma), 리차드 헬름(Richard Helm), 랄프 존슨(Ralph Johnson), 존 블리시디스(John Vissides) 에 의해 개발 영역에서 디자인 패턴을 구체화 하고 체계화 시킴
- Gof 디자인 패턴 분류

생성 패턴	- 객체 생성과 관련한 패턴 - 객체 생성에 있어서 프로그램 구조에 영향을 크게 주지 않는 유연성 제공
구조 패턴	- 클래스나 객체를 조합해서 더 큰 구조를 만드는 패턴
행위 패턴	- 객체나 클래스 사이의 알고리즘이나 책임 분배에 관련된 패턴

생성(Creational) 패턴	구조(Structural) 패턴	행위(Behavioral) 패턴
- Abstract Factory - Builder - Factory Method - Prototype - Singleton	- Adapter - Bridge - Composite - Decorator - Facade - Flyweight - Proxy	- Chain of Responsibility - Command - Interpreter - Iterator - Mediator - Memento - Observer - State - Strategy - Template Method - Visitor

08 애플리케이션 테스트 관리

Section 1. 애플리케이션 테스트케이스 설계

1. 소프트웨어 테스트

(1) 소프트웨어 테스트의 개념

- 구현된 소프트웨어에서, 사용자가 요구하는 기능의 동작과 성능, 사용성, 안정성 등을 만족하기 위하여 소프트웨어의 결함을 찾아내는 활동

(2) 소프트웨어 테스트의 필요성

- 오류 발견 관점
- 오류 예방 관점
- 품질 향상 관점

(3) 소프트웨어 테스트의 기본 원칙

- 테스팅은 결함이 존재함을 밝히는 활동이다.
- 완벽한 테스팅은 불가능하다.
- 테스팅은 개발 초기에 시작해야 한다.
- 결함 집중(Defect Clustering) - 파레토 법칙
- 살충제 패러독스(Presticide Paradox)
- 테스팅은 정황(Context)에 의존한다.
- 오류-부재의 궤변(Absence of Errors Fallacy)

(4) 테스트 산출물

- 테스트 계획서
- 테스트 케이스
  - 케이스를 작은 단위로 나누어 각 단위의 입력 값, 테스트 조건, 기대 결과를 기술한다.
- 테스트 시나리오
  - 테스트를 위한 절차를 명세한 문서
- 테스트 결과서
  - 테스트 결과를 정리한 문서

2. 테스트 오라클

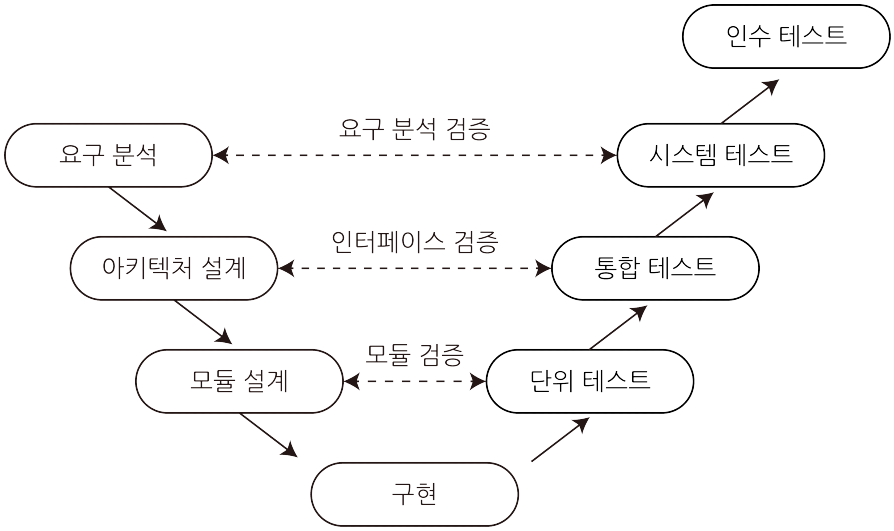
(1) 테스트 오라클의 개념

- 테스트의 결과가 참인지 거짓인지를 판단하기 위해서 사전에 정의된 참 값을 입력하여 비교하는 기법 및 활동

(2) 테스트 오라클의 유형

- 참 오라클 (True)
- 샘플링 오라클 (Sampling)
- 휴리스틱 오라클 (Heuristic)
- 일관성 검사 오라클 (Consistent)

3. 테스트 레벨



4. 소프트웨어 테스트 기법

(1) 프로그램 실행 여부

① 정적 테스트

- 소프트웨어의 실행 없이 소스 코드의 구조를 분석하여 논리적으로 검증하는 테스트
- 경로 분석, 제어 흐름 분석, 데이터 흐름 분석 등을 수행

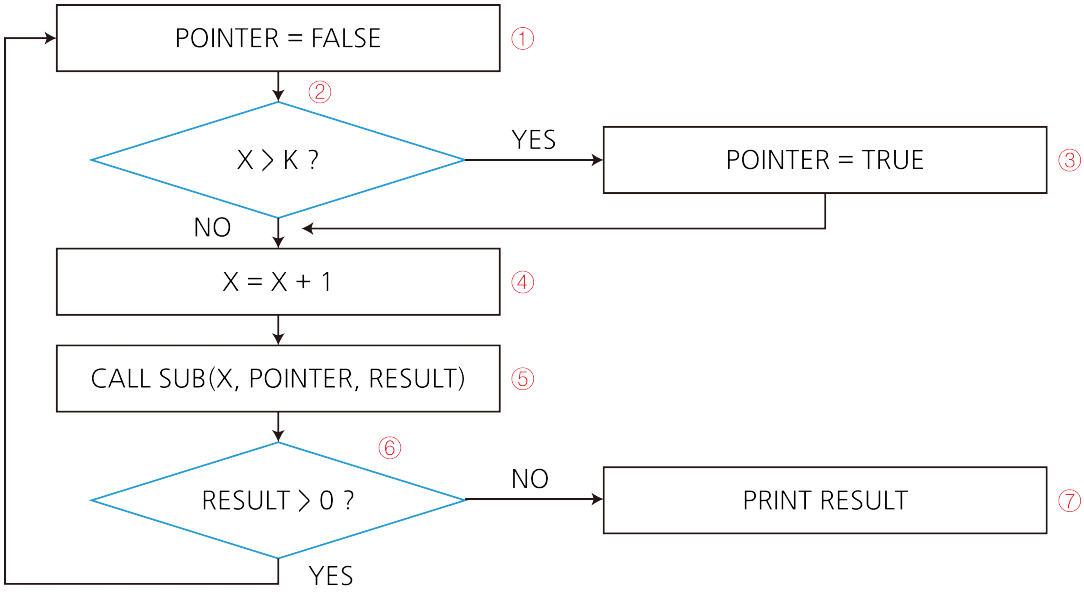
② 동적 테스트

- 소프트웨어를 실행 하여 실제 발생하는 오류를 발견하여 문제를 해결하는 분석 기법
- 다양한 운영 환경에서 소프트웨어를 분석

(2) 테스트 기법

① 화이트박스 테스트

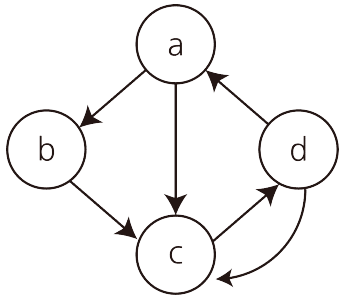
- 소프트웨어의 내부 구조와 동작을 검사하는 테스트 방식
- 화이트박스 테스트 기법



기법	설명	검증 방법
문장 검증	프로그램의 모든 문장을 한번 수행하여 검증	1,2,3,4,5,6,7
선택(분기) 검증	선택하는 부분만 검증	1,2,3,4,5,6,7 1,2,4,5,6,1
경로 검증	수행 가능한 모든 경로 검사	1,2,3,4,5,6,7 1,2,3,4,5,6,1 1,2,4,5,6,7 1,2,3,5,6,1
조건 검증	조건이나 반복문 내 조건식을 검사	x>1 or y<10 일 경우 x>1 조건과 y<10 모두 테스트

② 기초 경로 검사 (Basic Path Test)

- McCabe가 제안한 것으로 대표적인 화이트박스 테스트 기법
- 계산식 : V(G)=E-N+2



③ 블랙박스 테스트

- 프로그램의 외부 사용자 요구사항 명세를 보면서 테스트, 주로 구현된 기능을 테스트
- 블랙박스 테스트 기법

기법	설명
동등 분할 기법 (Equivalence Partitioning Testing)	- 입력 자료에 초점을 맞춰 테스트 케이스를 만들어 검사하는 방법 - 입력 데이터의 영역을 유사한 도메인별로 유효값과 무효값을 그룹핑하여 나누어서 검사
경계값 분석 (Boundary Value Analysis)	- 입력 값의 중간값보다 경계값에서 오류가 발생할 확률이 높다는 점을 이용해 입력 조건의 경계값을 테스트 케이스로 선정한다.
원인-효과 그래프 검사 (Cause-Effect Graphing Testing)	- 입력 데이터 간의 관계와 출력에 영향을 미치는 상황을 체계적으로 분석한 다음 효용성이 높은 테스트 케이스를 선정하여 검사하는 기법
오류 예측 검사 (Error Guessing)	- 과거의 경험이나 테스터의 감각으로 테스트하는 기법
비교 검사 (Comparison Testing)	- 여러 버전의 프로그램에 동일한 테스트 자료를 제공하여 동일한 결과가 출력되는지 테스트하는 기법이다.

(3) 테스트에 대한 시각

① 검증(Verification)

- 소프트웨어의 개발 과정을 테스트
- 올바른 소프트웨어가 만들어지고 있는지 검증

② 확인(Validation)

- 완성된 소프트웨어의 결과를 테스트
- 완성된 소프트웨어가 정상적으로 동작하는지 확인
- 완성된 소프트웨어가 사용자의 요구사항을 만족하는지 확인

(4) 테스트 목적

- 회복(Recovery) 테스트
  - 시스템에 고의로 실패를 유도하고 시스템이 정상적으로 복귀하는지 테스트
- 안전(Security) 테스트
  - 불법적인 소프트웨어가 접근하여 시스템을 파괴하지 못하도록 소스코드 내의 보안적인 결함을 미리 점검하는 테스트
- 강도(Stress) 테스트
  - 시스템에 과다 정보량을 부과하여 과부하 시에도 시스템이 정상적으로 작동되는지를 검증하는 테스트
- 성능(Performance) 테스트
  - 사용자의 이벤트에 시스템이 응답하는 시간, 특정 시간 내에 처리하는 업무량, 사용자 요구에 시스템이 반응하는 속도 등을 테스트
- 구조(Structure) 테스트
  - 시스템의 내부 논리 경로, 소스코드의 복잡도를 평가하는 테스트
- 회귀(Regression) 테스트
  - 변경 또는 수정된 코드에 대하여 새로운 결함 발견 여부를 평가하는 테스트
- 병행(Parallel) 테스트
  - 변경된 시스템과 기존 시스템에 동일한 데이터를 입력 후 결과를 비교하는 테스트

(5) 테스트 종류

① 명세 기반 테스트

- 주어진 명세를 빠짐없이 테스트 케이스로 구현하고 있는지 확인하는 테스트

② 구조 기반 테스트

- 소프트웨어 내부 논리 흐름에 따라 테스트 케이스를 작성하고 확인하는 테스트

③ 경험 기반 테스트

- 유사 소프트웨어나 유사 기술 평가에서 테스터의 경험을 토대로 한, 직관과 기술 능력을 기반으로 수행하는 테스트

5. 테스트 커버리지

(1) 테스트 커버리지의 개념

- 주어진 테스트 케이스에 의해 수행되는 소프트웨어의 테스트 범위를 측정하는 테스트 품질 측정 기준
- 테스트를 얼마나 수행했는지 측정하는 기준

(2) 테스트 커버리지 유형

① 기능 기반 커버리지

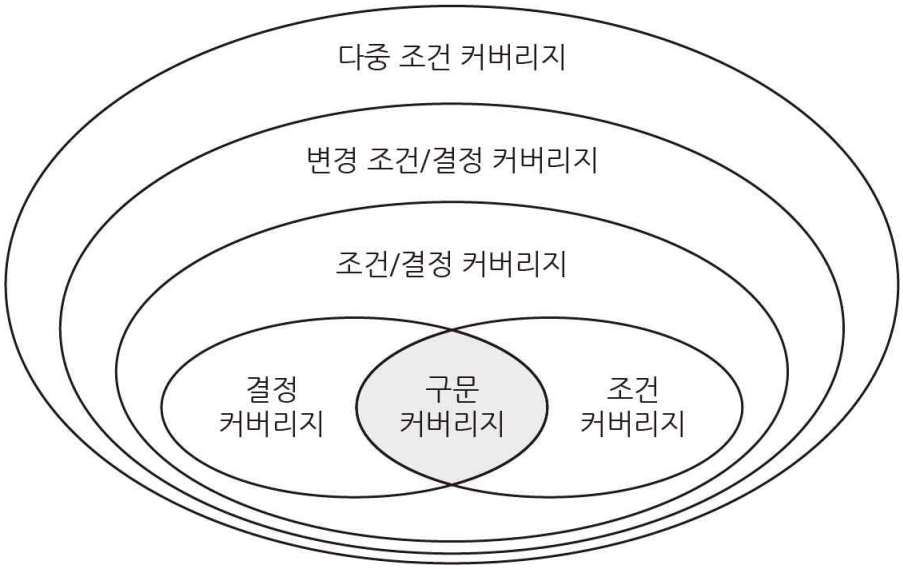
- 테스트 대상 애플리케이션의 전체 기능을 모수로 설정하고, 실제 테스트가 수행된 기능의 수를 측정하는 방법

② 라인 커버리지(Line Coverage)

- 애플리케이션 전체 소스 코드의 Line 수를 모수로 테스트 시나리오가 수행한 소스 코드의 Line 수를 측정하는 방법

③ 코드 커버리지(Code Coverage)

- 소프트웨어 테스트 충분성 지표 중 하나로 소스 코드의 구문, 조건, 결정 등의 구조 코드 자체가 얼마나 테스트되었는지를 측정하는 방법



Section 2. 애플리케이션 통합 테스트

1. 결함관리 도구

(1) 결함관리 도구의 개념

- 각 단계별 테스트 수행 후 발생한 결함의 재발 방지를 위해, 유사 결함 발견 시 처리 시간 단축을 위해 결함을 추적하고 관리할 수 있게 해주는 도구

(2) 결함관리 프로세스

- 에러 발견
- 에러 등록
- 에러 분석
- 결함 확정
- 결함 할당
- 결함 조치
- 결함 조치 검토 및 승인

(3) 결함 추이 분석

- 결함 관리 측정 지표
  - 결함 분포 : 각 애플리케이션 모듈 또는 컴포넌트의 특정 속성에 해당하는 결함의 수를 측정하여 결함의 분포를 분석할 수 있다.
  - 결함 추세 : 테스트 진행 시간의 흐름에 따른 결함의 수를 측정하여 결함 추세를 분석할 수 있다.
  - 결함 에이징 : 등록된 결함에 대해 특정한 결함 상태의 지속 시간을 측정하여 분석할 수 있다.

(4) 결함 관리 도구 종류

- S/W 테스트 관리
  - TestLink, GanttProject, OpenProj, Redmine
- 결함 추적 관리
  - Mantis, Bugzilla, Trac

2. 테스트 자동화 도구

(1) 테스트 자동화 도구의 개념

- 테스트 도구를 활용하여 반복적인 테스트 작업을 스크립트 형태로 구현함으로써, 테스트 시간 단축과 인력 투입 비용을 최소화하는 한편, 쉽고 효율적인 테스트를 수행할 수 있는 방법

(2) 테스트 자동화 도구 유형

① 정적 분석 도구(Static Analysis Tools)

- 정적 분석 도구는 만들어진 애플리케이션을 실행하지 않고 분석하는 방법

② 테스트 실행 도구(Test Execution Tools)

- 테스트를 위해 작성된 스크립트를 실행

③ 성능 테스트 도구(Performance Test Tools)

- 애플리케이션의 처리량, 응답 시간, 경과 시간, 자원 사용률에 대해 가상의 사용자를 생성하고 테스트를 수행함으로써 성능 목표를 달성하였는지를 확인하는 도구

④ 테스트 통제 도구(Test Control Tools)

- 테스트 관리 도구 : 테스트 계획 및 관리
- 형상 관리 도구 : 테스트 수행에 필요한 데이터와 도구를 관리
- 결함 추적/관리 도구 : 테스트에서 발생한 결함에 대해 관리하거나 협업을 지원

⑤ 테스트 장치(Test Harness)

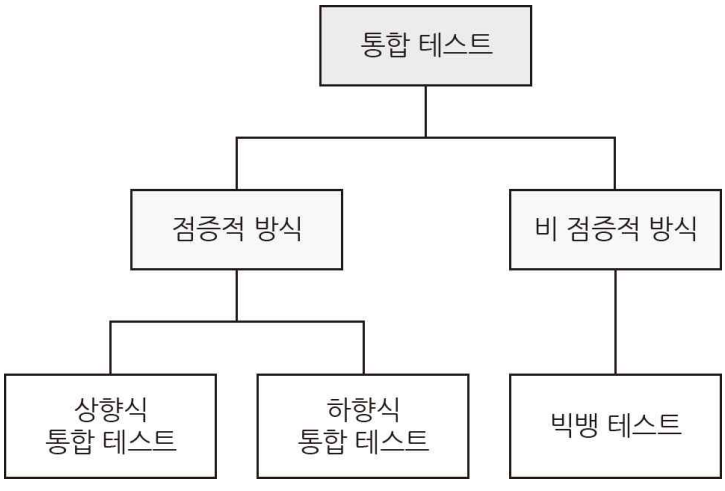
구성요소	설명
테스트 드라이버 (Test Driver)	- 테스트 대상 하위 모듈을 호출하고, 파라미터를 전달하고, 모듈 테스트 수행 후의 결과를 도출하는 등 상향식 테스트에 필요하다.
테스트 스텁 (Test Stub)	- 제어 모듈이 호출하는 타 모듈의 기능을 단순히 수행하는 도구로 하향식 테스트에 필요하다.
테스트 슈트 (Test Suites)	- 테스트 대상 컴포넌트나 모듈, 시스템에 사용되는 테스트 케이스의 집합을 말한다.
테스트 케이스 (Test Case)	- 입력 값, 실행 조건, 기대 결과 등의 집합을 말한다.
테스트 스크립트 (Test Script)	- 자동화된 테스트 실행 절차에 대한 명세를 말한다.
목 오브젝트 (Mock Object)	- 사용자의 행위를 조건부로 사전에 입력해 두면, 그 상황에 예정된 행위를 수행하는 객체를 말한다.

3. 통합 테스트

(1) 통합 테스트의 개념

- 소프트웨어 각 모듈 간의 인터페이스 관련 오류 및 결함을 찾아내기 위한 체계적인 테스트 기법

(2) 통합 테스트 수행 방법의 분류



Section 3. 애플리케이션 성능 개선

1. 애플리케이션 성능 저하 원인

(1) 데이터베이스 관련 성능 저하

- 데이터베이스 락(DB Lock)
- 불필요한 패치(DB Fetch)
- 연결 누수(Connection Leack)

(2) 내부 로직으로 인한 성능 저하 원인

(3) 외부 호출로 인한 성능 저하

2. 애플리케이션 성능 분석

(1) 애플리케이션 성능 분석 지표

- 처리량(Throughput)
- 응답 시간(Response Time)
- 경과 시간(Turn Around Time)
- 자원 사용률(Resource Usage)

(2) 성능 분석 도구

- JMeter
  - HTTP, FTP등 다양한 프로토콜을 지원하는 부하 테스트 도구
- LoadUI
  - 서버 모니터링, Drag&Drop 등 사용자 편리성이 강화된 도구
- OpenSTA
  - HTTP, HPPTS 프로토콜에 대한 부하 테스트 및 생산품 모니터링 도구

(3) 모니터링 도구

- Scouter
  - 단일 뷰 통합/실시간 모니터링
- NMon
  - 리눅스 서버 자원에 대한 모니터링 도구
- Zabbix
  - 웹기반 서버, 서비스, 애플리케이션 모니터링 도구
- Jeniffer
  - 애플리케이션에서 서버로 유입되는 트랜잭션 수량, 처리시간, 응답시간, 자원 활용율 등을 모니터링

3. 정형 기술 검토회의(FTR, Formal Technical Review)

(1) FTR의 개념

- 소프트웨어 엔지니어가 수행하는 소프트웨어 품질보증 활동
- 개발단계에서 제작되는 문서나 프로그램의 문제점을 찾고, 문제해결을 촉구하는 일반적인 용어
- S/W 개발 산출물을 대상으로 오류를 발견하기 위한 공식적인 활동

(2) FTR의 목적

- 산출물 요구사항 일치여부
- 소프트웨어가 미리 정한 기준에 따라 표현 되었는가를 확인
- 소프트웨어의 표현에 대한 기능, 논리적 오류
- 프로젝트를 보다 관리하기 쉽게 만든다.

4. 소스코드 품질 분석

(1) 동료 검토(Peer Review)

- 2~3명이 진행하는 리뷰의 형태
- 작성자가 코드를 설명하고 이해 관계자들이 설명을 들으면서 결함을 발견하는 형태로 진행하는 기법

(2) 워크스루(Walkthrough)

- 계획된 개발자 검토 회의
- 검토 자료를 회의 전에 배포해서 사전검토 한 후 짧은 시간 동안 회의를 진행하는 형태

(3) 인스펙션(Inspection)

- 공식적 검사 회의
- 작업자 외 다른 전문가가 검사하는 가장 공식적인 리뷰 기법

5. 소스코드 품질 분석 도구

구분	도구명	설명
정적 분석 도구	PMD	주로 Java에서 사용하지만, Javascript, PLSQL, XML 등의 언어도 지원
	checkstyle	자바 코드에 대한 소스 코드 표준 준수 검사, 다양한 개발 도구에 통합 가능
	SonarQube	중복코드, 복잡도, 코딩 설계 등을 분석하는 소스 분석 통합 플랫폼
	cppcheck	C, C++코드에 대한 메모리 누수, 오버플로우 등 문제 분석
	ccm	다양한 언어의 코드 복잡도 분석 도구
	cobertura	자바 언어의 소스 코드 복잡도 분석 및 테스트 커버리지 측정
동적 분석 도구	Avalanche	프로그램에 대한 결함 및 취약점 분석
	Valgrind	프로그램 내 존재하는 메모리 및 스레드 결함 등 분석

6. 애플리케이션 성능 개선하기

(1) 코드 최적화의 개념

- 주어진 코드에 대해 같은 작업을 수행 하면서, 실행 시간을 줄이거나 메모리를 줄이는 것

(2) 코드 스멜(Code Smell)

- 같은 코드가 여러 곳에 존재하는 중복된 코드로 코드 스멜(Code Smell)이 발생

용어	설명
스파게티 코드 (spaghetti code)	- 소스 코드가 복잡하게 얽힌 모습을 스파게티의 면발에 비유한 표현이다.
외계인 코드	- 아주 오래되거나 참고 문서 또는 개발자가 없어 유지보수 작업이 어려운 프로그램 코드이다.

(3) 리팩토링

- 외부 동작을 바꾸지 않으면서 내부 구조를 개선하는 방법

(4) 클린코드

- 의존성을 최소로 하고 사람이 이해할 수 있는 가독성, 목적성이 뛰어난 명확한 코드
- 클린코드 작성원칙

- 가독성
- 단순성
- 의존성 배제
- 중복성 최소화
- 추상화

09 소프트웨어 유지보수

Section 1. 소프트웨어 유지보수

1. 소프트웨어 유지보수

(1) 소프트웨어 유지보수의 개념

- 개발 완료 시점부터 폐기될 때까지, 지속적으로 수행하는 작업
- 소프트웨어의 수명을 연장하기 위한 활동

(2) 유지보수가 어려운 이유

- 업무 프로세스와 구축된 시스템을 이해해야 함
- 유지보수 계약이 개발과 계획과 별개인 경우 시간이 지나면서 소프트웨어 구조와 가독성이 떨어짐

2. 유지보수의 구분

(1) 수정 보수(Corrective Maintenance)

- 소프트웨어 구축 시 테스트 단계에 미처 발견하지 못한 잠재적인 오류를 찾아 수정한다.

(2) 적응 보수(Adaptive Maintenance)

- 운영체제, 하드웨어와 같은 프로그램 환경변화에 맞추기 위해 수행하는 유지보수

(3) 향상 보수(Perfective Maintenance)

- 기존 기능과 다른 새로운 기능을 추가하거나, 기존 기능을 개선

(4) 예방 보수(Preventive Maintenance)

- 장래에 유지보수성 또는 신뢰성을 보장하기 위해 선제적으로 하는 유지보수

3. 유지보수 관련 용어

- 레거시 시스템(legacy system)
  - 낡은 기술이나 방법론, 컴퓨터 시스템, 소프트웨어 등을 말한다.
  - 더이상 쓰이지 않더라도 현대의 기술에 영향을 주는 경우도 포함한다.
- 외계인 코드(Alien Code)
  - 아주 오래되거나 참고 문서 또는 개발자가 없어 유지보수 작업이 어려운 프로그램 코드
  - 프로그램 문서화(Documentation)을 통해 외계인 코드를 방지 가능
- 스파게티 코드(Spaghetti Code)
  - 복잡한 프로그래밍 소스 코드
  - 작동 자체는 제대로 하거나 하는 것처럼 보이지만, 추후 유지보수가 매우 어려워진다.

10 제품 소프트웨어 패키징

Section 1. 국제 표준 제품 품질 특성

1. 제품 품질 국제 표준

(1) 제품 품질 국제 표준의 개념

- 소프트웨어 개발 공정 각 단계에서 산출되는 제품이 사용자 요구를 만족하는지 검증하기 위한 국제 표준

(2) 소프트웨어 품질 관련 국제 표준

① ISO/IEC 9126 의 소프트웨어 품질 특성

품질 특성	설명
기능성 (Functionality)	- 소프트웨어가 특정 조건에서 사용될 때, 명시된 요구와 내재된 요구를 만족하는 기능에 대한 소프트웨어 제품의 능력 - 부특성 : 적합성, 정확성, 상호 운용성, 보안성, 준수성
신뢰성 (Reliability)	- 소프트웨어가 명세된 조건에서 사용될 때, 성능 수준을 유지할 수 있는 소프트웨어 제품의 능력 - 부특성 : 성숙성, 결함 허용성, 복구성
사용성 (Usability)	- 사용자에 의해 이해 이해되고, 학습되고, 사용되고, 선호될 수 있는 소프트웨어 제품의 능력 - 부특성 : 이해성, 학습성, 운영성, 선호도, 준수성
효율성 (Efficiency)	- 사용되는 자원의 양에 따라 요구된 성능을 제공하는 소프트웨어 제품의 능력 - 부특성 : 시간 반응성, 자원 활용성, 준수성
유지보수성 (Maintainability)	- 소프트웨어 제품이 변경되는 능력 - 소프트웨어의 수정, 개선 등이 포함된다. - 부특성 : 분석성, 변경성, 안정성, 시험성, 준수성
이식성 (Portability)	- 현재 환경에서 다른 환경으로 이전될 수 있는 소프트웨어 제품의 능력 - 부특성 : 적응성, 설치성, 공존성, 대체성, 준수성

② ISO/IEC 14598 평가 특성

평가 특성	설명
반복성 (Repetability)	- 특정 제품에 대해 동일 평가자가 동일 사양에 대해 평가 했을 때 동일한 결과가 나와야 한다.
재현성 (Reproducibility)	- 특정 제품에 대해 다른 평가자가 동일 사양에 대해 평가 했을 때 동일하다고 여길 수 있는 결과가 나와야 한다.
공정성 (Impartiality)	- 평가가 특정 결과에 편향되지 않아야 한다.
효율성 (Efficiency)	- 평가 결과가 평가자의 감정이나 의견에 의해 영향을 받지 않아야 한다.

③ ISO/IEC 12119

- 패키지 SW 품질 요구사항 및 테스트

④ ISO/IEC 25000

- ISO 9126과 ISO 14598, ISO 12119, ISO 15288 표준을 5개 영역 중심으로 통합한 소프트웨어 평가 모델 국제 표준

2. 프로세스 품질 국제 표준

(1) 프로세스 품질 국제 표준의 개념

- 소프트웨어 개발 프로세스 등 소프트웨어 관련 업체의 프로세스 관리능력을 평가하고 프로세스를 개선하는데 활용할 수 있는 표준

(2) 국제 프로세스 품질 표준

① ISO/IEC 12207 구성

생명주기 프로세스	세부 프로세스
기본 생명주기 프로세스	- 획득, 공급, 개발, 운영, 유지보수
지원 생명주기 프로세스	- 문서화, 형상관리, 품질보증, 검증, 확인, 합동검토, 감사, 문제해결
조직 생명주기 프로세스	- 관리, 기반구조, 개선, 교육훈련

② ISO/IEC 15504(SPICE)

- SPICE 프로세스 능력 수준

수준	단계	설명
0	불안정 단계(Incomplete)	미구현 또는 목표 미달성
1	수행 단계(Performed)	프로세스 수행 및 목적 달성
2	관리 단계(Managed)	프로세스 수행 계획 및 관리
3	확립 단계(Established)	표준 프로세스의 사용
4	예측 단계(Predictable)	프로세스의 정량적 이해 및 통제
5	최적화 단계(Optimizing)	프로세스의 지속적인 개선

③ CMM(Capability Maturity Model)

- 소프트웨어 개발 업체들의 업무능력평가 기준을 세우기 위한 평가 모형
- CMM 성숙도 5단계

수준	단계	설명
1	초기 단계(Initial)	- 소프트웨어를 개발하고 있으나 관리는 하고 있지 않은 상태 - 프로세스의 성과를 예측할 수 없는 상태
2	반복 단계(Repeatable)	- 이전의 성공적인 프로젝트의 프로세스를 반복하고 있는 상태 - 같은 것을 반복적으로 실행하며 어느 정도의 통계적 관리가 가능한 상태
3	정의 단계(Defined)	- 프로세스 작업이 잘 정의/이해되고, 프로세스 데이터에 의한 프로젝트 관리도 실행하고 있는 상태 - 프로세스의 기초가 정립되어 계속 진보되고 있는 상태
4	관리 단계(Managed)	- 프로세스 성과를 측정/분석하여 개선시키고, 이를 바탕으로 관리하고 있는 상태 - 정량적 프로세스 관리, 소프트웨어 품질 관리
5	최적화 단계(Optimizing)	- 질적, 양적으로 지속적인 개선이 이루어지고 있는 상태

④ CMMi(Capability Maturity Model Integration)

- 시스템과 소프트웨어 영역을 하나의 프로세스 개선 툴로 통합시켜 기업의 프로세스 개선 활동에 광범위한 적용성을 제공하는 모델
- CMMi 성숙도 5단계

수준	단계	설명
1	초기 단계(Initial)	- 구조화된 프로세스를 갖고 있지 않는 조직
2	관리 단계(Managed)	- 기본적인 프로세스를 갖고 있는 조직 - 기본 프로세스에 따라 업무가 수행되고 기본적인 관리 활동들로부터 구체적인 특정 영역으로 프로세스의 체계가 확대 발전하는 조직
3	정의 단계(Defined)	- 조직 차원의 표준 프로세스를 보유하고 있으며 프로젝트를 수행할 경우 프로젝트의 특성에 따라 적절하게 조정하여 사용
4	정량적 관리 단계 (Quantitatively Managed)	- 프로세스들을 통계적이고 정략적으로 관리하는 조직
5	최적화 단계(Optimizing)	- 질적, 양적으로 지속적인 개선이 이루어지고 있는 상태

Section 2. 제품 소프트웨어 패키징

1. 애플리케이션 패키징

(1) 애플리케이션 패키징의 개념

- 개발이 완료된 제품 소프트웨어를 고객에게 전달하기 위한 형태로 패키징하고, 설치와 사용에 필요한 제반 절차 및 환경 등 전체 내용을 포함하는 매뉴얼을 작성하는 활동

(2) 사용자 중심의 패키징 작업

- 사용자 실행 환경의 이해
- 사용자 관점에서의 패키징 고려 사항

(3) 애플리케이션 패키징 수행 순서

- 기능 식별
- 모듈화
- 빌드 진행
- 사용자 환경 분석
- 패키징 적용 시험
- 패키징 변경 개선

2. 릴리즈 노트

(1) 릴리즈 노트의 개념

- 소프트웨어 제품과 함께 배포되는 문서들을 말한다.

(2) 릴리즈 노트 작성 항목

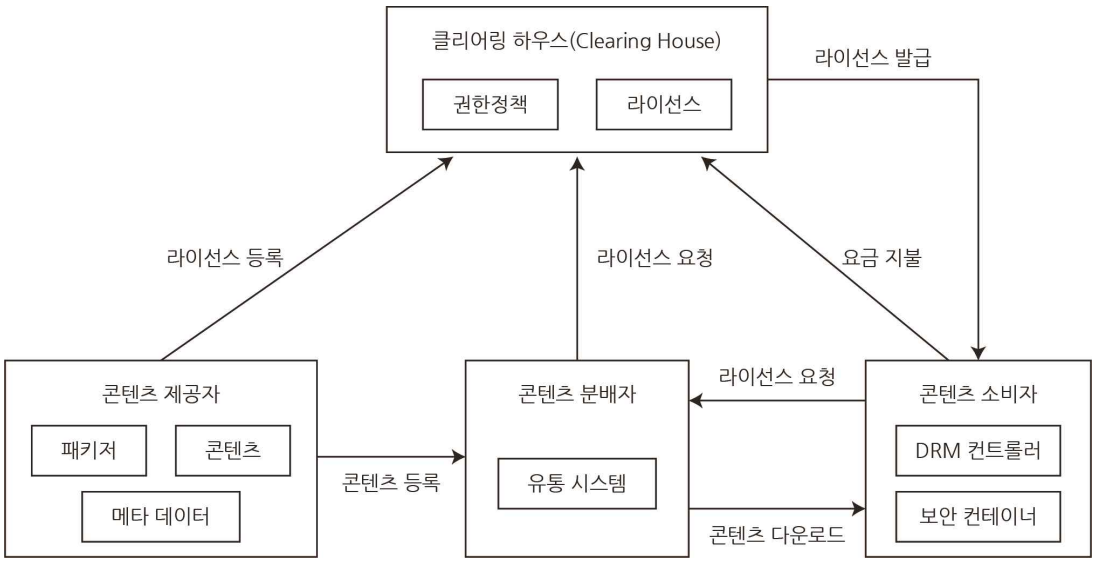
작성항목	설명
Header	- 문서 이름(릴리스 노트 이름), 제품 이름, 버전 번호, 릴리즈 날짜, 참고 날짜, 노트 버전 등
개요	- 제품 및 변경에 대한 간략한 전반적 개요
목적	- 릴리스 버전의 새로운 기능목록과 릴리스 노트의 목적에 대한 간략한 개요. - 버그 수정 및 새로운 기능 기술.
이슈 요약	- 버그의 간단한 설명 또는 릴리즈 추가 항목 요약
재현 항목	- 버그 발견에 따른 재현 단계 기술
수정/개선 내용	- 수정 / 개선의 간단한 설명 기술
사용자 영향도	- 버전 변경에 따른 최종 사용자 기준의 기능 및 응용 프로그램 상의 영향도 기술
SW 지원 영향도	- 버전 변경에 따른 SW의 지원 프로세스 및 영향도 기술
노트	- SW 및 HW Install 항목, 제품, 문서를 포함한 업그레이드 항목 메모
면책 조항	- 회사 및 표준 제품과 관련된 메시지. 프리웨어, 불법 복제 방지, 중복 등 참조에 대한 고지 사항
연락 정보	- 사용자 지원 및 문의 관련한 연락처 정보

3. DRM

(1) DRM(Digital Rights Management)의 개념

- 각종 디지털 콘텐츠의 불법적인 사용을 제한하고, 승인된 사용자의 콘텐츠 사용을 저작권 소유자의 의도에 따라 제어하는 기술

(2) DRM의 구성 및 흐름



(3) DRM 사용 규칙 제어 기술

① 콘텐츠 식별 체계 (Identificayion)

- 디지털 콘텐츠에 고유 식별 번호를 부여하여 관리하고 운영
- 대표적으로 DOI(Digital Object Identifier), URI 가 있다.

② 메타데이터 (Meta Data)

- 콘텐츠에 관한 구조화된 데이터
- 콘텐츠의 속성정보

③ 권리표현기술 (Right Expression)

- 콘텐츠에 대한 규칙 설정
- 어느 사용자가 어떠한 권한과 어떠한 조건으로 콘텐츠를 이용할 수 있는지 정의
- 콘텐츠의 사용조건(기간, 횟수) 등에 의해 사용이 제한 될 수 있고, 주로 XML 기반으로 권한 표현 언어가 개발
- XrML(eXtensible rights mark-up language) 기술이 대표적

④ 권리표현 종류

권리표현 종류	설명
Render Permission	사용자에게 콘텐츠가 표현되고 이용되는 권리 형태를 정의
Transport Permission	사용자들 간에 권리의 교환이 이루어지는 권리 형태를 정의
Derivative Permission	콘텐츠의 추출 변형이 가능한 권리 형태를 정의

⑷ 저작권 보호 기술

① 암호화 기술

- 특정 키를 가진 사용자만이 해당 콘텐츠를 사용할 수 있도록 한다.
- 암호화 키와 복호화 키가 서로 다른 비대칭키 방식과 두 키가 동일한 대칭키 방식 있다.

② 위변조 방지(Tamper-proofing)

- 콘텐츠에 승인되지 않은 조작이 가해졌을 때, 위변조 사항을 감지할 수 있도록 하고, 오류 동작을 일으키게끔 하는 기술
- 부정 조작에 대한 방어 기술

③ 워터마킹(Watermarking)

- 콘텐츠에 저작권 정보를 은닉하여, 향후 저작권 분쟁이 일어날 경우, 추척을 통해 저작권자를 확인할 수 있게 해주는 기술
- 워터마킹(Watermarking), 핑거프린팅(Fingerprinting) 으로 구분

관점	워터마킹	핑거프린팅
목적	불법 복제 방지	불법 유통 방지
삽입정보	저작권 정보	저작권 정보 + 구매자 정보
컨텐츠 변화 시점	최초 저작 시점	구매시점 마다
취약점	불법 유통	공모 공격

⑸ DRM 구성요소

구성요소	설명
암호화 (Encryption)	- 콘텐츠 및 라이선스를 암호화하고, 전자 서명을 할 수 있는 기술 - PKI, Symmetric/Asymmetric Encryption, Digital Signature
키 관리 (Key Manangement)	- 콘텐츠를 암호화한 키에 대한 저장 및 배포 기술 (Centralized, Enveloping)
암호화 파일 생성 (Packager)	- 콘텐츠를 암호화된 콘텐츠로 생성하기 위한 기술 - Pre-packaging, On-the-fly Packaging
식별 기술 (Identification)	- 콘텐츠에 대한 식별 체계 표현 기술 - DOI, URI
저작권 표현 (Right Expression)	- 라이선스의 내용 표현 기술 - XrML/MPGE-21 REL, ODRL
정책 관리 (Policy management)	- 라이선스 발급 및 사용에 대한 정책표현 및 관리기술 - XML, Contents Management System
크랙 방지 (Tamper Resistance)	- 크랙에 의한 콘텐츠 사용 방지 기술 - Secure DB, Secure Time Management, Encryption
인증 (Authentication)	- 라이선스 발급 및 사용의 기준이 되는 사용자 인증 기술 - User/Device Authentication, SSO, DiGital Certificate

Section 3. 제품 소프트웨어 매뉴얼 작성

1. 제품 소프트웨어 매뉴얼 작성

⑴ 제품 소프트웨어 매뉴얼 개념

- 사용자가 제품 구매 후 최초 설치 시 참조하게 되는 매뉴얼
- 제품 소프트웨어 소개, 설치 파일, 설치 절차 등이 포함

⑵ 제품 소프트웨어 설치 매뉴얼

구성 요소	설명
제품 소프트웨어 개요	- 제품 소프트웨어의 주요 기능 및 UI 설명 - UI 및 화면 상의 버튼, 프레임 등을 도식화하여 설명
설치 관련 파일	- 제품 소프트웨어를 설치하기 위한 관련 파일 설명 - 설치 구동을 위한 exe 실행 - ini나 log 파일 같은 관련 파일
설치 절차	- 소프트웨어 설치 방법을 순서대로 상세히 설명
설치 아이콘	- Windows 구동용 설치 아이콘 설명
프로그램 삭제	- 해당 소프트웨어 삭제 시 원래대로 삭제하는 방법을 설명
설치 환경	- CPU, Memory, OS 등 환경설명
설치 버전 및 작성자	- 소프트웨어 릴리즈 버전 및 작성자 정보
고객 지원 방법 및 FAQ	- 실제 설치 시 자주 발생하는 어려움들을 FAQ로 정리 - 유선 및 E-mail, Website URL

⑶ 제품 소프트웨어 사용자 매뉴얼

작성 항목	설명
목차 및 개요	- 매뉴얼 전체의 내용을 순서대로 요약 - 제품 소프트웨어의 주요 특징에 대해 정리 - 사용자 매뉴얼에서의 구성과 실행 방법, 메뉴에 대한 설명을 비롯하여 사용법, 각 항목에 따른 점검 기준, 그리고 설정 방법 등에 대해 기술함
문서 이력 정보	- 사용자 매뉴얼 변경 이력 정보
사용자 매뉴얼의 주석	- 주의 사항: 사용자가 반드시 숙지해야 하는 중요한 정보의 주석 표시 - 참고 사항: 특별한 사용자 환경 및 상황에 대한 내용의 주석 표시
기록 항목	- 제품명칭, 모델명, 기록 항목에 대한 문서 번호, 제품 번호, 구입 날짜 등을 기재
기본 사항	- 소프트웨어 개요, 사용방법, 모델/버전별 특징, 기능 및 인터페이스 특징, 구동 환경 등을 기재
고객 지원 방법 및 FAQ	- 소프트웨어 사용시 자주 발생하는 어려움들을 FAQ로 정리 - 유선 및 E-mail, Website URL
준수 정보 & 제한 보증	- 시리얼 보존, 불법 등록 사용금지 등의 준수 사항 권고 - 저작권 정보 관련 작성

11 데이터베이스 구축

Section 1. 데이터베이스 개념

1. 데이터베이스 개념

(1) 데이터와 정보

- 데이터
  - 현실 세계에서 단순히 관찰하거나 측정하여 수집한 사실이나 값
- 정보
  - 데이터를 의사 결정에 유용하게 활용할 수 있도록 처리하여 체계적으로 정리한 결과물

(2) 데이터베이스의 정의

- 통합 데이터(Integrated Data)
  - 검색의 효율성을 위해 중복이 최소화 된 데이터의 모임
- 저장 데이터(Stored Data)
  - 컴퓨터가 접근 가능한 저장 매체에 저장된 데이터
- 운영 데이터(Operational Data)
  - 조직의 목적을 위해 존재 가치가 확실하고 반드시 필요한 데이터
- 공유 데이터(Shared Data)
  - 여러 응용 프로그램들이 공동으로 사용하는 데이터

(3) 데이터베이스의 특징

- 실시간 접근성(Real Time Accessibility)
- 계속적인 변화(Continuous Evolution)
- 동시 공유(Concurrent Sharing)
- 내용에 의한 참조(Content Reference)
- 데이터의 독립성(Independence)

(4) 데이터 언어

① DDL(Data Definition Language : 데이터 정의어)

- DB의 구조, 데이터 형식, 접근 방식 등 DB의 구축과 변경 목적으로 사용하는 언어

② DML(Data Manipulation Language : 데이터 조작어)

- 데이터의 검색, 삽입, 삭제, 갱신 연산 등을 포함한 집합

③ DCL(Data Control Language : 데이터 제어어)

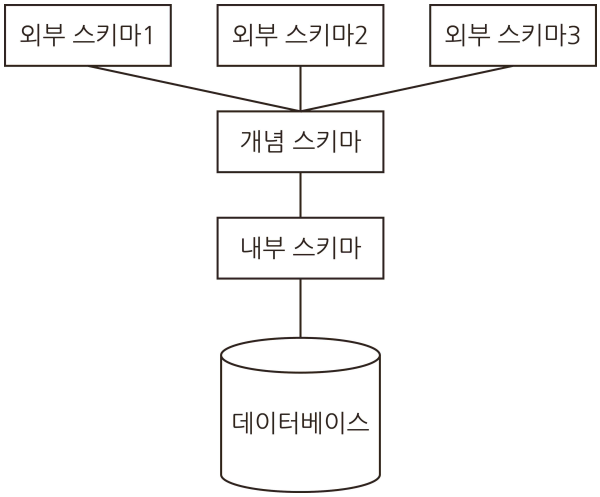
- 보안 및 권한 제어, 무결성, 회복, 병행 제어를 위한 언어

⑸ 스키마(schema)

① 스키마의 정의

- 데이터베이스의 구조와 제약조건에 관해 전반적인 명세를 기술한 것

② 3계층 스키마



- 외부 스키마 (external schema) - 사용자 뷰
- 개념 스키마 (Conceptual schema) - 전체적인 뷰
- 내부 스키마 (Internal schema) - 저장 스키마

2. 데이터베이스 관리 시스템 (Database Management System)

⑴ DBMS의 정의

- 데이터베이스를 조작하는 별도의 소프트웨어

⑵ DBMS의 기능

- 데이터 정의
- 데이터 조작
- 데이터 제어
- 데이터 공유
- 데이터 보호
- 데이터 구축
- 유지보수

⑶ DBMS의 종류

- 계층형(Hierarchical DataBase)
- 네트워크형(Network DataBase)
- 관계형(Relational DataBase)
- 객체 지향형(Object-Oriented DataBase)
- 객체 관계형(Object-Relational DataBase)
- NoSQL
- NewSQL

Section 2. 데이터베이스 설계

1. 데이터베이스 설계 개요

(1) 데이터베이스 설계 정의

- 요구조건에서부터 데이터베이스 구조를 도출해 내는 과정

(2) 데이터베이스 설계 목적

- 이해관계자의 데이터 관점 요구사항에 대한 정확한 이해 및 추상화

(3) 데이터베이스 설계 시 고려사항

- 제약조건
- 데이터베이스 무결성
- 일관성
- 회복
- 보안
- 효율성
- 데이터베이스 확장성

2. 데이터베이스 설계 단계

(1) 요구조건 분석

- 데이터베이스의 사용자, 사용목적, 사용범위, 제약조건 등에 대한 내용을 정리하고 명세서를 작성
- 트랜잭션 유형, 트랜잭션 실행빈도와 같은 동적 DB 처리 요구조건 정의

(2) 개념적 설계

- 현실세계를 데이터관점으로 추상화 단계
- DBMS 에 독립적으로 설계
- 데이터베이스의 개념적 스키마 구성(E-R 다이어그램)

(3) 논리적 설계

- 자료를 컴퓨터가 이해할 수 있도록 특정 DBMS의 논리적 자료 구조로 변환하는 과정
- 특정 데이터모델(계층형, 관계형, 객체지향형 등)을 적용한 설계
- 데이터베이스의 논리적 스키마 생성
- 관계형 데이터베이스인 경우 이 단계에서 테이블을 설계하고, 정규화 과정
- 트랜잭션 인터페이스 설계

(4) 물리적 설계

- 특정 DBMS의 물리적 구조와 내부적인 저장구조, 분산형태, 데이터타입의 특징, 인덱스의 특징 등을 구체화 하는 설계단계
- 레코드 집중의 분석 및 설계
- 오브젝트, 접근방법, 트랜잭션분석, 인덱스, 뷰, 데이터베이스 용량설계 등을 수행
- 데이터베이스의 물리적 스키마 생성
- 트랜잭션 세부 설계

(5) 구현

- 목표 DBMS의 DDL로 기술된 명령문을 컴파일하고, 실행시켜 데이터베이스 스키마 생성

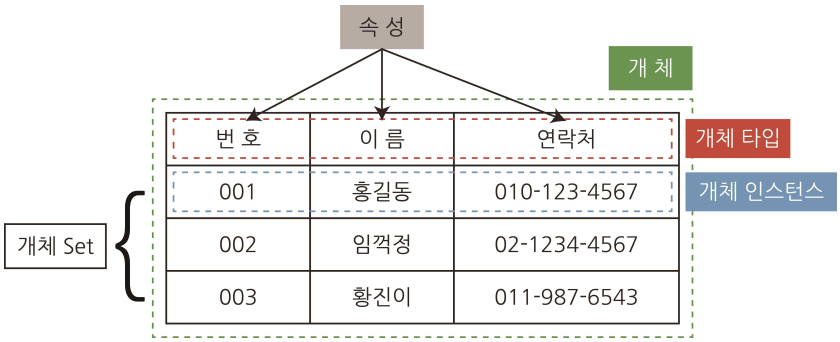
Section 3. 데이터 모델링

1. 데이터모델 개념

(1) 데이터모델 개념

- 현실세계의 요소를 인간과 컴퓨터가 이해할 수 있는 정보로 표현한 것

(2) 데이터모델 구조



- 개체(Entity)
  - 데이터베이스에 데이터로 표현하려고 하는 현실 세계의 대상체
- 개체 타입 (Entity type)
  - 개체를 구성하고 있는 속성들의 집합
- 개체 인스턴스 (Entity instance)
  - 데이터베이스에 저장되는 구체적인 객체
- 개체 세트 (Entity set)
  - 개체 인스턴스의 집합
- 속성(Attribute)
  - 데이터의 가장 작은 논리적 단위
- 관계(Relation)
  - 개체와 개체가 맺고 있는 의미 있는 연관성

(3) 데이터모델 표시해야 할 요소

- 구조(Structure)
  - 데이터베이스에 표현될 대상으로서의 개체 타입과 개체 타입들간의 관계
- 연산(Operation)
  - 데이터베이스에 저장될 실제 데이터를 처리하는 방법
- 제약조건(Constraint)
  - 저장될 수 있는 데이터의 논리적인 제약조건

2. 개체-관계 모델 (Entity Relation Model)

(1) 개체-관계 모델 개념

- 데이터베이스에 대한 요구 사항을 그래픽적으로 표현하는 방법

(2) 개체 (Entity)

- 현실 세계에서 꼭 필요한 사람이나 사물과 같이 구별되는 모든 것

(3) 애트리뷰트, 속성(Attribute)

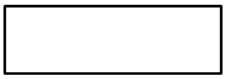
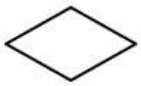
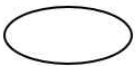
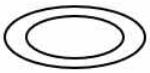
- 개체나 관계가 가지고 있는 고유의 특성
- 속성의 유형

속성 유형	설명
단일 값 속성	- 값을 하나만 가질 수 있는 속성 (ex. 이름, 학번 등)
다중 값 속성	- 값을 여러 개 가질 수 있는 속성 (ex. 취미 등)
단순 속성	- 의미를 더는 분해할 수 없는 속성 (ex. 성별 등)
복합 속성	- 의미를 분해할 수 있는 속성 (ex. 주소, 생년월일 등)
유도 속성	- 기존의 다른 속성의 값에서 유도되어 결정되는 속성 (ex. 주민번호와 성별)
널 속성	- 아직 결정되지 않은 존재 하지 않는 값
키 속성	- 각 개체를 식별하는데 사용하는 속성

(4) 관계 (Relationship)

- 서로 다른 개체가 맺고 있는 의미 있는 연관성

(5) E-R 다이어그램 기호

기호	기호 이름	설명
	사각형	- 개체(Entity)
	마름모	- 관계(Relationship)
	타원	- 속성(Attribute)
	밑줄 타원	- 기본키 속성
	이중 타원	- 복합속성
	선 링크	- 개체와 속성 연결

3. 데이터 모델의 품질 기준

기준항목	설명
정확성	데이터 모델이 표기법에 따라 정확하게 표현되었고, 업무영역 또는 요구사항이 정확하게 반영되었음을 의미함
완전성	데이터 모델의 구성 요소를 정의하는데 있어서 누락을 최소화하고, 요구사항 및 업무영역 반영에 있어서 누락이 없음을 의미함
준거성	제반 준수 요건들이 누락 없이 정확하게 준수되었음을 의미함
최신성	데이터 모델이 현행 시스템의 최신 상태를 반영하고 있고, 이슈사항들이 지체없이 반영되고 있음을 의미
일관성	여러 영역에서 공통 사용되는 데이터 요소가 전사 수준에서 한 번만 정의되고 이를 여러 다른 영역에서 참조·활용되면서, 모델 표현상의 일관성을 유지하고 있음을 의미함
활용성	작성된 모델과 그 설명 내용이 이해관계자에게 의미를 충분하게 전달할 수 있으면서, 업무 변화 시에 설계 변경이 최소화되도록 유연하게 설계되어 있음을 의미

Section 4. 논리 데이터베이스 설계

1. 논리적 데이터 모델링

⑴ 논리적 모델링

- 개념적 설계에서 추출된 실체와 속성들의 관계를 구조적으로 설계하는 단계

2. 데이터베이스 정규화(Normalization)

⑴ 정규화의 개념

- 관계형 데이터베이스의 설계에서 중복을 최소화하게 데이터를 구조화

⑵ 정규화의 목적

- 데이터의 중복을 최소화
- 정보의 무손실 : 정보가 사라지지 않아야 함
- 독립적인 관계는 별개의 릴레이션으로 표현
- 정보의 검색을 보다 용이하게 함
- 이상현상 최소화

⑶ 이상(Anomaly) 현상

- 데이터 중복으로 인해 릴레이션 조작 시 예상하지 못한 곤란한 현상이 발생
- 이상의 종류
  - 삽입 이상 : 데이터를 삽입할 때 불필요한 데이터가 함께 삽입되는 현상
  - 삭제 이상 : 한 튜플을 삭제할 때 연쇄 삭제 현상으로 인해 정보 손실
  - 갱신 이상 : 튜플의 속성값을 갱신할 때 일부 튜플의 정보만 갱신되어 정보에 모순이 생기는 현상

⑷ 함수적 종속 (Functional Dependency)

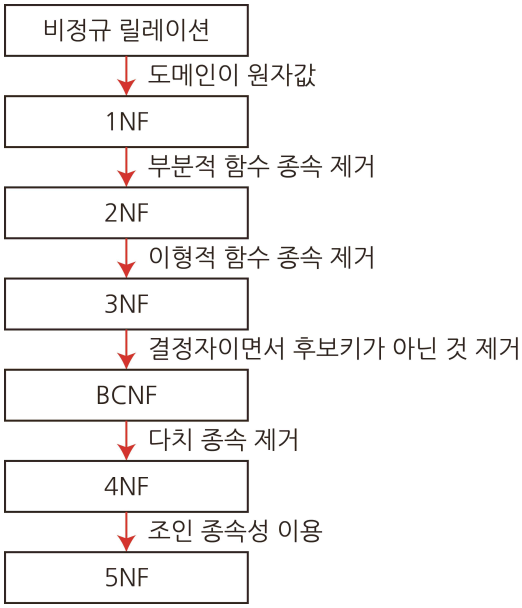
① 함수적 종속의 개념

- 어떤 릴레이션 R이 있을때 X와 Y를 각각 속성의 부분집합이라고 가정했을 때
  - X의 값을 알면 Y의 값을 바로 식별할 수 있고, X의 값에 Y의 값이 달라질 때, Y는 X에 함수적 종속이라고 함
  - 이를 기호로 표현하면 X→Y

② 함수적 종속 관계

- 완전 함수적 종속 (Full Functional Dependency)
- 부분 함수적 종속 (Partial Functional Dependency)
- 이행적 함수 종속 (Transitive Functional Dependecy)

(5) 정규화 과정


비정규 릴레이션
도메인이 원자값
1NF
부분적 함수 종속 제거
2NF
이형적 함수 종속 제거
3NF
결정자이면서 후보키가 아닌 것 제거
BCNF
다치 종속 제거
4NF
조인 종속성 이용
5NF

Section 5. 물리 데이터베이스 설계

1. 물리 데이터베이스 설계

⑴ 물리 데이터베이스 설계 과정

- 사용자 DBMS 결정
- 데이터 타입 크기 결정
- 데이터 용량, 설계 및 업무 프로세스 분석
- 역정규화(반정규화)
- 인덱스 정의
- 데이터베이스 생성

⑵ 물리 데이터베이스 설계시 고려사항

- 반응시간
  - 응답시간을 최소화 해야 한다.
- 트랜잭션 처리량
  - 얼마나 많은 트랜잭션을 동시에 발생시킬 수 있는지 검토한다.
- 공간활용
  - 데이터가 저장될 공간을 효율적으로 배치한다.

⑶ 데이터베이스 암호화 방식

- API방식
  - 데이터베이스 솔루션 외부의 어플리케이션에서 데이터의 암/복화화를 수행
- Plug-in 방식
  - 데이터베이스 서버에 제품을 설치 → 암/복호화 수행
- TDE(Transparent Data Encryption)방식
  - DBMS 암호화 기능을 이용하여 데이터 파일 저장시 암호화
  - 파일에 저장된 내용을 메모리로 가져올 때 DBMS에 의해 복호화
- 파일암호화 방식
  - 데이터뿐만 아니라 비정형 데이터 암호화 적용가능
- 하드웨어방식
  - 별도의 하드웨어 장비를 외부에 설치

2. 반정규화

(1) 반정규화의 개념

- 데이터베이스 정규화 이후, 성능향상과 개발 편의성 등 정규화 기법에 위배되는 수행 기법

(2) 반정규화시 고려사항

- 데이터의 중복이 발생하여 데이터 수정시 무결성이 깨질 수 있다.
- 읽기 속도는 향상되지만, 삽입/삭제/수정 속도는 느려짐
- 저장공간의 효율이 떨어짐
- 테이블이 크고 복잡해져 유지보수가 어려움
- 과도한 반정규화는 오히려 성능을 저하시킴
- 반정규화를 위해서는 사전에 데이터의 일관성과 무결성을 우선으로 할지, 데이터베이스의 성능과 단순화를 우선으로 할지를 결정해야 함

(3) 반정규화의 적용순서

- 반정규화의 대상을 조사한다.
- 다른 방법으로 유도한다.
- 반정규화 수행

(4) 반정규화의 유형

구분	유형	설명
테이블 분할	수평분할	- 레코드 단위로 분할
	수직분할	- 컬럼 단위로 분할
테이블 중복	통계 테이블 추가	- DW, OLAP 데이터 용
	진행 테이블 추가	- 업무 프로세스 상태
컬럼기반 분할	조회 빈도 기반	- 고빈도 컬럼 분리
	크기 기반 분할	- 일정 용량 컬럼 분리
컬럼 중복	중복 컬럼 추가	- 자주 조회되는 컬럼 추가
	파생 컬럼 추가	- 연산 결과 별도 저장

4. 데이터베이스 이중화

(1) 데이터베이스 이중화 구성

- 장애발생시 데이터베이스를 보호하기 위한 방법으로 동일한 데이터베이스를 중복시켜 동시에 갱신하여 관리하는 방법
- 서버와 네트워크, 프로그램 등의 정보 시스템이 지속적으로 정상 운영이 가능한 고가용성(HA, High Availability) 서버로 구성하는 것

(2) 데이터베이스 이중화의 분류

- Eager 기법
  - 트랜잭션 수행 중에 발생한 변경은 발생 즉시 모든 이중화서버로 전달되어 연쇄적으로 변경 내용이 반영
- Lazy 기법
  - 트랜잭션의 수행이 완전히 완료된 후에 그 변경 사실에 대한 새로운 트랜잭션을 작성하여 각 노드에게 전달하는 기법

(3) 데이터베이스 이중화의 종류

① Active-Active

- 다중화된 장비가 모두 가동되는 방식

② Active-Standby

- 두 대 중 하나는 가동이 되고, 하나는 장애 상황의 경우를 대비해서 준비 상태로 대기
- Active-Standby 타입

Hot Standby	- Standby 장비가 가동되었을 때 즉시 사용가능
Warm Standy	- Standby 장비가 가동되었을 때 설정에 대한 준비가 필요함
Cold Standby	- Standby 장비를 평소에는 정지시켜두며 필요에 따라서 직접 켜서 구성을 함

5. 데이터베이스 백업

(1) 백업 방식

- 전체 백업(Full Backup)
  - 선택된 폴더의 DATA를 모두 백업하는 방식
- 증분 백업(Incremental Backup)
  - Full 백업 이후 변경/추가된 Data만 백업하는 방식
- 차등 백업(Differential Backup)
  - Full 백업 이후 변경/추가된 Data를 모두 포함하여 백업
- 실시간 백업(RealTime Backup)
  - 즉각적으로 모든 변경사항을 분리된 스토리지 디바이스에 복사
- 트랜잭션 로그 백업(Transaction log Backup)
  - 데이터베이스에서 실행되는 모든 SQL문을 기록한 로그
  - REDO(다시 실행), UNDO(원상태로 복구) 로 복원
  - CHECK POINT : 설정한 지점 이전 까지는 트랜잭션이 성공적으로 수행이 돼서 disk에 확실히 저장된 상태

⑵ 복구시간목표/복구시점 목표

- 복구 시간 목표(RTO)
  - 서비스 중단 시점과 서비스 복원 시점 간에 허용되는 최대 지연 시간
- 복구 시점 목표(RPO)
  - 마지막 데이터 복구 시점 이후 허용되는 최대 시간

Section 6. 데이터베이스 물리속성 설계

1. 파티셔닝

⑴ 파티셔닝 개념

- 데이터베이스를 여러 부분으로 분할하는 것

⑵ 분할 기준

① 범위 분할 (range partitioning)

- Partition Key 의 연속된 범위로 파티션을 정의
- 파티션 키 위주로 검색이 자주 실행될 경우 유용
- 예) 월별, 분기별 등

② 목록 분할 (list partitioning)

- 특정 Partition 에 저장 될 Data 에 대한 명시적 제어
- 많은 SQL 에서 해당 Column 의 조건이 많이 들어오는 경우 유용
- ex) [한국, 일본, 중국 → 아시아] [노르웨이, 스웨덴, 핀란드 → 북유럽]

③ 해시 분할 (hash partitioning)

- 파티션 키 값에 해시 함수를 적용하고, 거기서 반환된 값으로 파티션 매핑
- 데이터가 모든 파티션에 고르게 분산되도록 DBMS가 관리
- 병렬처리 시 성능효과 극대화

④ 라운드 로빈 분할 (round robin partitioning)

- Data를 균일하게 분배해서 저장하는 방식

⑤ 합성 분할 (composite partitioning)

- 위의 기술들을 복합적으로 사용하는 방법
- ex) 범위 분할 후 분할 된 데이터를 해시 분할하는 등

2. 클러스터 설계

⑴ 클러스터의 개념

- 디스크로부터 데이터를 읽어오는 시간을 줄이기 위해서 조인이나 자주 사용되는 테이블의 데이터를 디스크의 같은 위치에 저장시키는 방법

⑵ 클러스터 대상 테이블

- 분포도가 넓은 테이블
- 대량의 범위를 자주 조회하는 테이블
- 입력, 수정, 삭제가 자주 발생하지 않는 테이블
- 자주 JOIN 되어 사용되는 테이블
- ORDER BY, GROUP BY, UNION 이 빈번한 테이블

3. 인덱스(Index)

⑴ 인덱스의 개념

- 추가적인 저장 공간을 활용하여 데이터베이스 테이블의 검색 속도를 향상시키기 위한 자료구조

⑵ 인덱스의 종류

- 클러스터 인덱스
  - 테이블당 1개만 허용되며, 해당 컬럼을 기준으로 테이블이 물리적으로 정렬
- 논클러스터 인덱스
  - 테이블당 약 240개의 인덱스 생성 가능
  - 레코드의 원본은 정렬되지 않고, 인덱스 페이지만 정렬
- 밀집 인덱스
  - 데이터 레코드 각각에 대해 하나의 인덱스가 만들어짐
- 희소 인덱스
  - 레코드 그룹 또는 데이터 블록에 대해 하나의 인덱스가 만들어짐

⑶ 인덱스 생성/삭제

① 인덱스 생성

```
--문법
CREATE INDEX [인덱스명] ON [테이블명](컬럼1, 컬럼2, 컬럼3.....)

--예제
CREATE INDEX USER_INDEX ON TB_USER(NAME,HP);
```

② 인덱스 삭제

```
--문법
DROP INDEX 인덱스 명;

--예제
DROP INDEX USER_INDEX;
```

4. 뷰(View)

(1) 뷰의 개념

- 하나 이상의 기본 테이블로부터 유도된, 이름을 가지는 가상 테이블

(2) 뷰 생성/삭제

① 뷰 생성

```
--문법
CREATE VIEW [view_name]AS
SELECT [field_name_1], [field_name_2]
FROM [table_name]
WHERE [조건];

--예제
CREATE OR REPLACE VIEW USER_VIEW AS
SELECT
    id, user
FROM TB_USER
WHERE name like '이%'
;
```

② 뷰 삭제

```
--문법
DROP VIEW 뷰 명;

--예제
DROP VIEW USER_VIEW;
```

5. 시스템 카탈로그

(1) 시스템 카탈로그

- 데이터베이스 관리자의 도구로, 데이터베이스에 저장되어 있는 모든 데이터 개체들에 대한 정의나 명세에 대한 정보가 수록되어 있는 시스템 테이블

(2) 시스템 카탈로그의 내용

- 릴레이션 관련 정보
- 인덱스 관련 정보
- 뷰 관련 정보
- 통계 관련 정보
- 사용자 관련 정보

(3) 시스템 카탈로그의 특징

- 시스템 카탈로그 자체도 시스템 테이블로 구성되어 있어 사용자가 SQL문을 이용하여 내용을 검색해 볼 수 있다.
- 시스템 카탈로그는 데이터베이스 관리 시스템에 의해 생성되고 유지된다.
- 사용자가 시스템 카탈로그를 직접 갱신하는 것은 허용되지 않는다.

(4) 시스템 카탈로그의 종류

- SYSTABLES
  - 기본 테이블 및 뷰 테이블의 정보를 저장하는 테이블
- SYSCOLUMNS
  - 모든 테이블에 대한 정보를 열(속성) 중심으로 저장하는 테이블
- SYSVIEW
  - 뷰에 대한 정보를 저장하는 테이블
- SYSTABAUTH
  - 테이블에 설정된 권한 사항들을 저장하는 테이블
- SYSCOLAUTH
  - 각 속성에 설정된 권한 사항들을 저장하는 테이블

Section 7. 관계 데이터베이스 모델

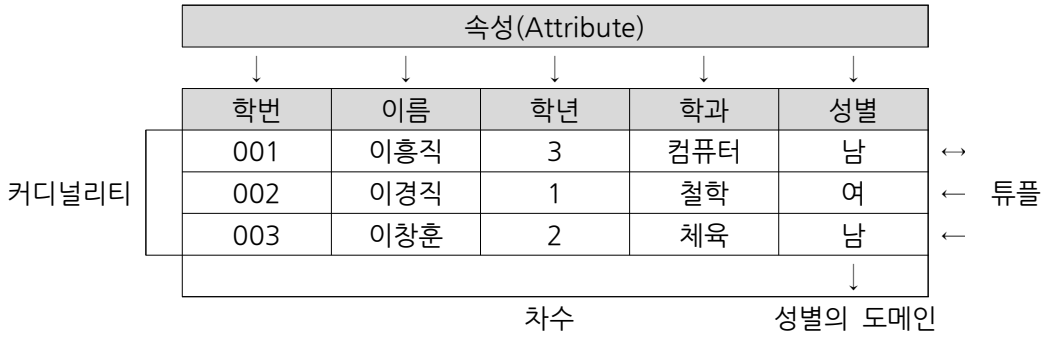
1. 관계 데이터 모델

(1) 관계 데이터 모델 개념

- 데이터의 논리적 구조가 릴레이션, 즉 테이블 형태의 평면 파일로 표현되는 데이터 모델

(2) 관계 데이터 릴레이션의 구조

〈학생 릴레이션〉



2. 관계데이터 언어(관계대수, 관계해석)

(1) 관계 대수의 개념

- 원하는 데이터를 얻기 위해, 데이터를 어떻게 찾는지에 대한 처리 과정을 명시하는 절차적인 언어

(2) 관계해석

- 관계 데이터 모델의 제안자인 코드(E. F. Codd)가 수학의 Predicate Calculus(술어 해석)에 기반을 두고 관계 데이터베이스를 위해 제안
- 관계해석은 원하는 정보가 무엇이라는 것만 정의하는 비절차적 특성

Section 8. 키와 무결성 제약조건

1. 컬럼

(1) 컬럼의 개념

- 릴레이션에서 정보를 나타내는 최소 단위로, 각 열의 상태나 특성을 나타내는 항목을 말한다.

(2) 속성의 분류

- 기본속성
  - 업무로부터 추출한 모든 속성
- 설계속성
  - 코드성 데이터, 릴레이션 식별용 일련번호
- 파생속성
  - 다른 속성에 영향을 받아 발생하는 속성
  - 계산값, 합계, 재고 등

(3) 세부 의미에 따른 분류

- 단순 속성(Simple Attribute)
  - 나이, 성별같이 다른 속성들로 구성될 수 없는 단순한 속성
- 복합 속성(Composite Attribute)
  - 주소와 같이 시, 구, 동처럼 여러 세부 속성들로 구성될 수 있는 속성

(4) 구성방식 따른 분류

- PK(Primary Key) 속성
  - 릴레이션에서 튜플을 유일하게 구분할 수 있는 속성
- FK(Foreign Key) 속성
  - 다른 릴레이션과의 관계에서 참조하고 있는 속성
- 일반 속성
  - 릴레이션에 포함된 속성 중, PK와 FK가 아닌 속성

(5) 도메인

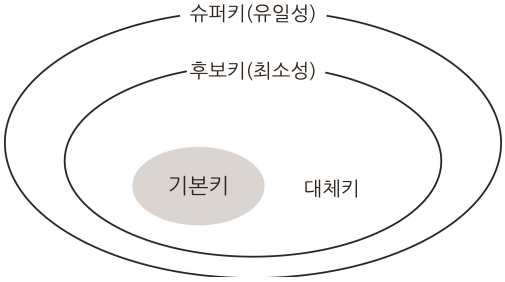
- 속성이 가질 수 있는 값의 범위

2. 키 종류

⑴ 키(Key)의 개념

- 데이터베이스에서 조건에 만족하는 튜플을 찾거나 순서대로 정렬할 때 다른 튜플들과 구별할 수 있는 유일한 기준이 되는 컬럼

⑵ 키(Key)의 종류



① 후보키 (Candidate Key)

- 릴레이션을 구성하는 속성들 중에서 튜플을 유일하게 식별할 수 있는 속성들의 부분집합
- 튜플에 대한 유일성과 최소성을 만족시켜야 한다.

② 기본키 (Primary Key)

- 후보키 중에서 선택한 주키(Main Key)
- Null 값을 가질 수 없다. (개체 무결성)
- 기본키로 정의된 속성에는 동일한 값이 중복되어 저장될 수 없다. (개체 무결성)

③ 대체키 (Alternate Key)

- 후보키가 둘 이상일 때 기본키를 제외한 나머지 후보키

④ 슈퍼키 (Super Key)

- 한 릴레이션 내에 있는 속성들의 집합으로 구성된 키

⑤ 외래키 (Foreign Key)

- 관계(Relation)를 맺고 있는 릴레이션 R1, R2에서 릴레이션 R1이 참조하고 있는 릴레이션 R2의 기본키와 같은 R1 릴레이션의 속성

3. 데이터베이스 무결성

⑴ 데이터베이스 무결성 개념

- 데이터의 정확성, 일관성, 유효성이 유지되는 것

⑵ 데이터베이스 무결성 종류

① 개체 무결성 (Entity integrity)

- 모든 릴레이션은 기본 키(primary key)를 가져야 한다
- 기본키는 중복되지 않은 고유한 값을 가져야한다.
- 릴레이션의 기본키는 NULL 값을 허용하지 않는다.

② 참조 무결성 (Referential integrity)

- 외래키 값은 NULL이거나 참조하는 릴레이션의 기본키 값과 동일해야 한다.
- 각 릴레이션은 참조할 수 없는 외래키 값을 가질 수 없다.
- 참조 무결성 제약조건

종류	설명
제한(Restricted)	- 문제가 되는 연산을 거절
연쇄(Cascade)	- 참조되는 릴레이션에서 튜플을 삭제하고 참조하는 릴레이션에서 이 튜플을 참조하는 튜플도 함께 삭제
널값(Nullify)	- 참조되는 릴레이션에서 튜플을 삭제하고 참조하는 릴레이션에서 이 튜플을 참조하는 튜플들의 외래키에 NULL 등록
기본값(Default)	- Null을 넣는 대신에 디폴트 값을 등록

③ 도메인 무결성 (Domain integrity)

- 속성들의 값은 정의된 도메인에 속한 값이어야 한다.
- 성별이라는 컬럼에는 '남', '여'를 제외한 데이터는 제한되어야 한다.

Section 9. 물리데이터 모델 품질 검토

1. CRUD 분석

(1) CRUD의 개념

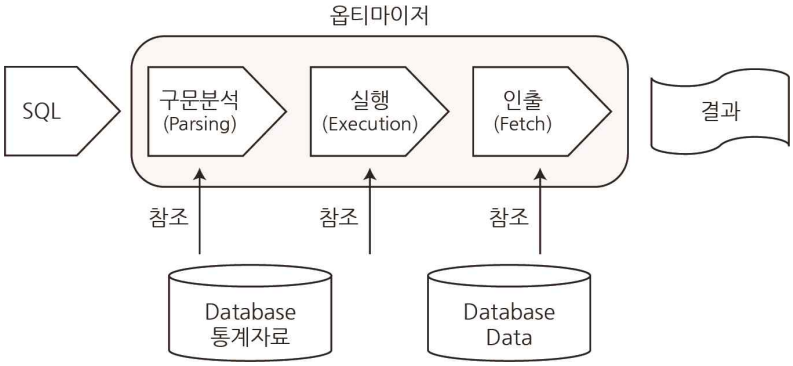
- 데이터 처리 기능인 Create(생성), Read(읽기), Update(갱신), Delete(삭제)를 묶어서 표현한 말이다.

(2) CRUD의 필요성

- 모델링 작업검증
- 중요 산출물
- 테스트 시 사용
- 인터페이스 현황 파악

2. 옵티마이저

(1) SQL 처리 흐름



(2) 옵티마이저 개념

- 사용자가 질의한 SQL 문에 대해 최적의 실행 방법을 결정하는 역할을 수행
- 옵티마이저의 구분

㉠ 규칙기반 옵티마이저 (Rule Based Optimizer)

- 규칙(우선순위)를 가지고 실행 계획을 생성
- 인덱스 유무, 연산자, 객체 등을 참조하여 우선순위를 부여

㉡ 비용기반 옵티마이저 (Cost Based Optimizer)

- SQL문을 처리하는데 필요한 비용이 가장 적은 실행계획을 선택하는 방식
- 소요시간 이나 자원 사용량을 가지고 실행계획 생성
- 테이블, 인덱스, 컬럼 등의 다양한 객체 통계정보와 시스템 통계정보 활용

3. SQL 성능 튜닝

⑴ 튜닝의 개념

- SQL 문을 최적화하여 빠른 시간내에 원하는 결과값을 얻기 위한 작업

⑵ SQL 성능 최적화를 위한 유틸리티

- SQL Trace
- TKPROF(Trace Kernel PROFile)
- EXPLAIN PLAN

Section 10. 분산 데이터베이스

1. 분산 데이터베이스

⑴ 분산 데이터베이스(Distribute Database)의 정의

- 여러 곳으로 분산되어있는 데이터베이스를 하나의 가상 시스템으로 사용할 수 있도록 한 데이터베이스

⑵ 분산 데이터베이스 구성요소

- 분산 처리기
  - 자체적으로 처리 능력을 가지며, 지리적으로 분산되어 있는 컴퓨터 시스템
- 분산 데이터베이스
  - 지리적으로 분산되어 있는 데이터베이스로서 해당 지역의 특성에 맞게 데이터베이스가 구성
- 통신 네트워크
  - 분산처리기들을 통신망으로 연결하여 논리적으로 하나의 시스템처럼 작동할 수 있도록 하는 통신 네트워크

⑶ 투명성 조건

- 위치 투명성 (Location Transparency)
  - 액세스하려는 데이터베이스의 실제 위치를 알 필요없이 단지 데이터베이스의 논리적인 명칭만으로 액세스할 수 있음
- 분할 투명성 (Division Transparency)
  - 하나의 논리적 테이블이 여러 단편으로 분할되어 각 단편의 사본이 여러 위치에 저장
- 지역사상 투명성 (Local Mapping transparency)
  - 지역DBMS와 물리적 DB사이의 Mapping 보장. 각 지역시스템 이름과 무관한 이름 사용 가능
- 중복 투명성 (Replication Transparency)
  - 동일 데이터가 여러 곳에 중복되어 있더라도 사용자는 마치 하나의 데이터만 존재하는 것처럼 사용하고, 시스템은 자동으로 여러 자료에 대한 작업을 수행함
- 병행 투명성 (Concurrency Transparency)
  - 분산 데이터베이스와 관련된 다수의 트랜잭션들이 동시에 실현되더라도 그 트랜잭션의 결과는 영향을 받지 않음
- 장애 투명성 (Failure Transparency)
  - 트랜잭션, DBMS, 네트워크, 컴퓨터 장애에도 불구하고 트랜잭션을 정확하게 처리함

⑷ CAP 이론

① 개념

- 어떤 분산환경에서도 일관성(C), 가용성(A), 분할내성(P) 세 가지 속성 중, 두 가지만 가질 수 있다는 것
- 3가지 모두 만족할 수는 없다.

② 특징의 의미

- 일관성(Consistency)
  - 모든 노드들은 같은 시간에 동일한 항목에 대하여 같은 내용의 데이터를 사용자에게 보여준다.
- 가용성(Availability)
  - 모든 사용자들이 읽기 및 쓰기가 가능해야 하며, 몇몇 노드의 장애 시에도 다른 노드에 영향을 미치면 안된다.
- 분할내성(Partition tolerance)
  - 메시지 전달이 실패하거나 시스템 일부가 망가져도 시스템이 계속 동작할 수 있어야 한다.

2. 트랜잭션

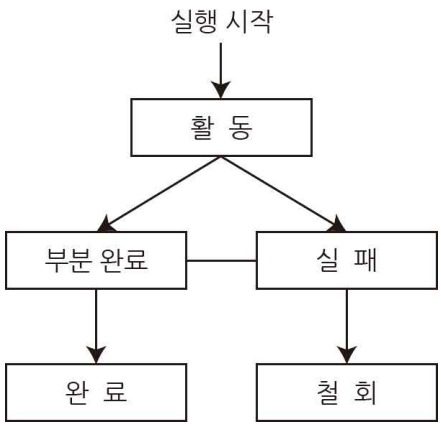
⑴ 트랜잭션의 개념

- 데이터베이스의 상태를 변환시키는 하나의 논리적인 기능을 수행하는 작업 단위

⑵ 트랜잭션의 성질

- 원자성 (Atomicity)
  - 트랜잭션의 연산은 데이터베이스에 모두 반영되든지 아니면 전혀 반영되지 않아야 한다.
  - Commit과 Rollback 명령어에 의해 보장 받는다.
- 일관성 (Consistency)
  - 트랜잭션이 그 실행을 성공적으로 완료하면 언제나 일관성 있는 데이터베이스 상태로 변환한다.
- 독립성, 격리성 (Isolation)
  - 둘 이상의 트랜잭션이 동시에 병행 실행되는 경우 어느 하나의 트랜잭션 실행중에 다른 트랜잭션의 연산이 끼어들 수 없다.
- 영속성 (Durablility)
  - 성공적으로 완료된 트랜잭션의 결과는 시스템이 고장나더라도 영구적으로 반영되어야 한다.

⑶ 트랜잭션의 상태



12 SQL 활용

Section 1. 기본 SQL 작성

1. SQL (Structured Query Language)

⑴ SQL의 개념

- 데이터베이스 시스템에서 자료를 처리하는 용도로 사용되는 구조적 데이터 질의 언어

⑵ SQL 문법의 종류

① Data Definition Language (DDL) - 데이터 정의어

- 데이터가 저장되는 테이블이나 각종 개체들을 정의하는데 사용되는 명령
- CREATE, ALTER, DROP, RENAME, TRUNCATE

② Data Manipulation Language (DML) - 데이터 조작어

- 데이터베이스 내의 데이터를 조작(추출, 생성, 수정, 삭제) 하는 명령
- SELECT, INSERT, UPDATE, DELETE

③ Data Control Language (DCL) - 데이터 제어어

- 데이터베이스에 접근하고, 객체들을 사용하도록 권한을 주고 회수하는 명령
- GRANT, REVOKE

④ Transaction Control Language (TCL) - 트랜잭션 제어어

- 논리적인 작업의 단위를 묶어 이에 의해 조작된 결과를 작업단위로별로 제어하는 명령어
- COMMIT, ROLLBACK, SAVEPOINT

Section 4. 절차형 SQL

1. 저장 프로시저 (Stored Procedure)

(1) 저장 프로시저의 개념

- 일련의 쿼리를 마치 하나의 함수처럼 실행하기 위한 쿼리의 집합

(2) 저장 프로시저의 구조

```
CREATE OR REPLACE PROCEDURE 프로시저명
( 변수1 IN 변수타입, 변수2 OUT 변수타입, 변수3 IN OUT 변수타입.... )
IS
    변수 처리부
BEGIN
    처리내용
EXCEPTION
    예외처리부
END;
```

2. 트리거

(1) 트리거의 개념

- 테이블에 대한 이벤트에 반응해 자동으로 실행되는 작업

(2) 트리거의 유형

- 행 트리거
  - 테이블 안의 영향을 받은 행 각각에 대해 실행
  - FOR EACH ROW 옵션 사용
- 문장 트리거
  - INSERT, UPDATE, DELETE 문에 대해 단 한번만 실행

(3) 트리거의 실행 시기

- BEFORE : 이벤트 전
- AFTER : 이벤트 후

(4) 트리거 생성 예

```
CREATE TRIGGER TRIGGER_GOODS_STOCK
AFTER INSERT ON TB_GOODS_STOCK FOR EACH ROW
BEGIN
    UPDATE TB_GOODS
    SET
        nStock = nStock + NEW.nStock
    WHERE idx = NEW.p_idx;
END
```

3. 사용자 정의 함수

(1) 사용자 정의 함수의 개념

- 프로시저와 사용자 정의 함수 모두 호출하게 되면 미리 정의 해놓은 기능을 수행하는 모듈
- 프로그램 로직을 도와주는 역할을 한다.
- 파라미터는 입력 파라미터만 가능하고, 리턴값이 하나이다.

(2) 사용자 정의 함수의 구조

```
CREATE OR REPLACE FUNCTION 함수명
( 매개변수1, 매개변수2, 매개변수3,..... )
RETURN 데이터 타입
IS
    변수 처리부
BEGIN
    처리내용
    RETURN 반환값;
EXCEPTION
    예외처리부
END;
```

13 병행제어와 데이터전환

Section 1. 병행제어와 회복

1. 병행제어

(1) 병행제어의 개념

- 여러 트랜잭션들이 동시에 실행되면서도 데이터베이스의 일관성을 유지할 수 있게 하는 기법

(2) 병행제어의 문제점

① 갱신 분실 (Lost Update)

- 두 개 이상의 트랜잭션이 같은 자료를 공유하여 갱신할 때 갱신 결과의 일부가 없어지는 현상

② 비완료 의존성 (Uncommitted Dependency)

- 하나의 트랜잭션 수행이 실패한 후 회복되기 전에 다른 트랜잭션이 실패한 갱신 결과를 참조하는 현상

③ 모순성 (Inconsistency)

- 두 개의 트랜잭션이 병행수행될 때 원치 않는 자료를 이용함으로써 발생하는 문제
- 갱신 분실과 비슷해 보이지만 여러 데이터를 가져올 때 발생하는 문제

④ 연쇄 복귀 (Cascading Rollback)

- 병행수행되던 트랜잭션들 중 어느 하나에 문제가 생겨 Rollback하는 경우 다른 트랜잭션도 함께 Rollback되는 현상

(3) 병행제어 기법

① 로킹(Locking)

- 트랜잭션이 어떤 데이터에 접근하고자 할 때 로킹 수행
- 로킹 단위에 따른 구분

구분	로크 수	병행성	오버헤드
로킹 단위가 크면	적어짐	낮아짐	감소
로킹 단위가 작으면	많아짐	높아짐	증가

② 2단계 로킹 규약(Two-Phase Locking Protocol)

- Lock과 Unlock이 동시에 이루어지면 일관성이 보장되지 않으므로 Lock만 가능한 단계와 Unlock만 가능한 단계를 구분
- 확장단계: 새로운 Lock은 가능하고 Unlock은 불가능하다.
- 축소단계: Unlock 은 가능하고 새로운 Lock은 불가능하다.

③ 타임스탬프(Time Stamp)

- 데이터에 접근하는 시간을 미리 정하여서 정해진 시간(Time Stamp)의 순서대로 데이터에 접근하여 수행

④ 낙관적 병행제어(Optimistic Concurrency Control)

- 트랜잭션 수행 동안은 어떠한 검사도 하지 않고, 트랜잭션 종료 시에 일괄적으로 검사

⑤ 다중 버전 병행제어(Multi-version, Concurrency Control)

- 여러 버전의 타임스탬프를 비교하여 스케줄상 직렬가능성이 보장되는 타임스탬프를 선택

2. 회복 (Database Recovery)

(1) 회복의 개념

- 트랜잭션들을 수행하는 도중 장애로 인해 손상된 데이터베이스를 손상되기 이전의 정상적인 상태로 복구시키는 작업

(2) 장애의 유형

- 트랜잭션 장애
- 시스템 장애
- 미디어 장애

(3) Undo와 Redo

- Undo
  - 트랜잭션 로그를 이용하여 오류와 관련된 모든 변경을 취소하여 복구 수행
- Redo
  - 트랜잭션 로그를 이용하여 오류가 발생한 트랜잭션을 재실행하여 복구 수행

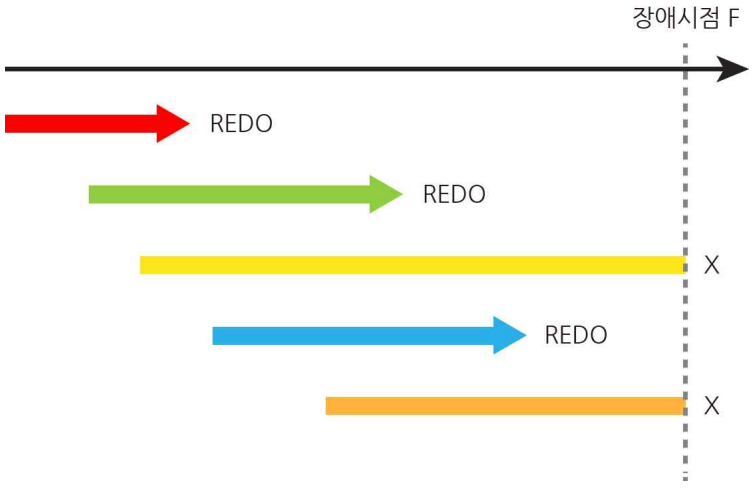
(4) 로그 파일

- 트랜잭션이 반영한 모든 데이터의 변경사항을 데이터베이스에 기록하기 전에 미리 기록해두는 별도의 파일

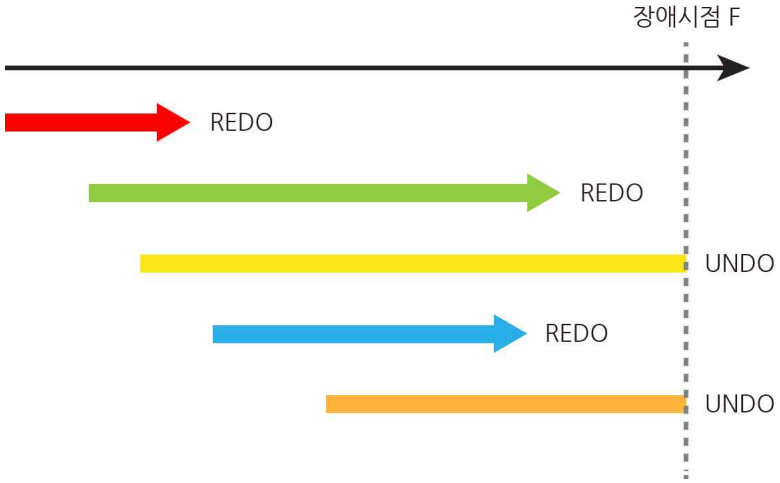
(5) 회복 기법

① 로그 기반 회복 기법

- 지연갱신 회복 기법(Deferred Update)
  - 트랜잭션의 부분 완료 상태에선 변경 내용을 로그 파일에만 저장
  - 중간에 장애가 생기더라도 데이터베이스에 기록되지 않았으므로 UNDO가 필요 없음(미실행 된 로그 폐기)



- 즉시갱신 회복 기법(Immediate Update)
  - 트랜잭션 수행 도중에도 변경 내용을 즉시 데이터베이스에 기록
  - 커밋 발생 이전의 갱신은 원자성이 보장되지 않는 미완료 갱신이므로 장애 발생 시 UNDO 필요



② 검사점 회복 기법 (Checkpoint Recovery)

- 장애 발생 시 검사점(Checkpoint) 이전에 처리된 트랜잭션은 회복에서 제외하고 검사점 이후에 처리된 트랜잭션은 회복 작업 수행

③ 그림자 페이징 회복 기법 (Shadow Paging Recovery)

- 트랜잭션이 실행되는 메모리상의 Current Page Table과 하드디스크의 Shadow Page Table 이용

④ 미디어 회복 기법 (Media Recovery)

- 디스크와 같은 비휘발성 저장 장치가 손상되는 장애 발생을 대비한 회복 기법

⑤ ARIES 회복 기법

Section 2. 데이터 전환

1. ETL(Extraction, Transformation, Loading)

(1) ETL 개념

- 다양한 소스 시스템(source system)으로부터 필요한 데이터를 추출(extract)하여 변환(transform) 작업을 거쳐 타깃 시스템(target system)으로 전송 및 로딩(loading)하는 모든 과정

(2) ETL 기능

- Extraction (추출)
  - 하나 또는 그 이상의 데이터 원천들로부터 데이터 획득
- Transformation (변환)
  - 데이터 클렌징, 형식 변환 및 표준화, 통합 또는 다수 애플리케이션에 내장된 비즈니스룰 적용 등
- Load (적재)
  - 변형 단계의 처리가 완료된 데이터를 특정 목표 시스템에 적재

14 운영체제

Section 1. 운영체제 기초

1. 기억장치

(1) 기억장치의 개념

- 데이터, 프로그램, 연산의 중간 결과 등을 일시적 또는 영구적으로 저장하는 장치

(2) 기억장치의 종류

① 레지스터

- 중앙처리장치 내부에 존재하는 기억장치

② 캐시 메모리

- 중앙처리장치가 주기억장치에 접근할 때 속도 차이를 줄이기 위해 사용

③ 주기억장치

- 중앙처리장치가 직접 데이터를 읽고 쓸 수 있는 장치
- 종류

종류	설명
ROM (Read Only Memory)	- 읽기만 가능한 읽기 전용 메모리 - 비 휘발성 메모리 - 종류 : mask-ROM, PROM, EPROM, EEPROM
RAM (Random Access Memory)	- 기억장소를 임의로 접근할 수 있는 메모리 - 읽고 쓰기가 가능한 휘발성 메모리 - SRAM : 전원이 공급되는 중에 내용이 사라지지 않음 (캐쉬메모리로 사용) - DRAM : 일반적인 주기억장치로, 일정 시간이 지나면 내용이 사라지는 RAM

④ 보조기억장치

- 주기억장치에 비해 접근 시간은 느리지만 기억 용량이 크다.
- 종류 : HDD, SSD(Solid State Drive), CD, USB, 플로피 디스크 등

2. 시스템 소프트웨어

(1) 시스템 소프트웨어의 개념

- 응용 소프트웨어를 실행하기 위한 플랫폼을 제공

(2) 시스템 소프트웨어의 구성

① 제어 프로그램

- 감시 프로그램 (Superviosr Program)
- 작업관리 프로그램 (Job Control Program)
- 데이터 관리 프로그램

② 처리 프로그램

- 서비스 프로그램(Service Program)
- 문제 프로그램(Problem Program)
- 언어 번역 프로그램(Language Translator Program)

3. 운영체제

(1) 운영체제의 개념

- 응용프로그램이 실행되는 과정에서 하드웨어들을 제어하여 응용프로그램을 실행시키고 실행 결과를 보일 수 있도록 컴퓨터 내부 동작을 관리하는 소프트웨어

(2) 운영체제의 기능

- 프로세스 관리
- 메모리 관리
- 파일 관리
- 입출력 관리
- 보조기억장치 관리
- 네트워킹
- 정보 보안 관리
- 명령해석 시스템

(3) 운영체제 운용 기법

- 일괄 처리 시스템(batch processing system)
- 다중 프로그래밍 시스템(multi programming system)
- 시분할 시스템 (time sharing system)
- 다중 처리 시스템 (multi-processing system)
- 실시간 처리 시스템 (real time processing system)
- 다중 모드 시스템 (multi-mode system)
- 분산 처리 시스템(distribute processing system)

⑷ 운영체제 성능 평가 기준

- 처리량(Throughput)
  - 일정 시간 내에 시스템이 처리하는 일의 양
- 반환시간(Turnaround Time)
  - 요청한 작업에 대하여 결과를 돌려줄 때까지 소요되는 시간
  - 대기시간 + 실행시간 + 응답시간
- 신뢰도(Reliability)
  - 작업의 결과가 얼마나 정확하고 믿을 수 있는가의 기준
- 사용 가능도(Availability)
  - 시스템을 사용할 필요가 있을 때 즉시 사용 가능한 정도

4. 운영체제의 종류

⑴ 윈도우(Windows)

- MS-DOS의 멀티태스킹 기능과 GUI 환경을 제공하는 운영체제

⑵ 리눅스(Linux)

- 1991년 리누스 토발즈에 의해 오픈소스로 개발된 유닉스 호환 OS

⑶ 유닉스(Unix)

① 유닉스 개요

- 1969년 미국 AT&T 벨연구소(Bell Lab.)가 개발한 공개형 오픈소스 운영체제(OS)
- 1969년 벨연구소 소속의 켄 톰슨(Ken Tompson)이 어셈블리 언어를 사용하여 개발했으며, 1972년 데니스 리치(Dennis Ritchie)가 C 언어를 사용하여 다시 작성

② Unix 시스템의 구성

- 커널(Kernel)
  - UNIX의 가장 핵심적인 부분
  - 컴퓨터가 부팅될 때, 주 기억장치에 적재된 후 상주하면서 실행
- 쉘(Shell)
  - 명령어 해석기
  - 시스템과 사용자 간의 인터페이스 담당
- Utility Program
  - 일반 사용자가 작성한 응용 프로그램을 처리하는 데 사용
  - 에디터, 컴파일러, 인터프리터, 디버거 등

③ Unix 파일 시스템

- Unix 파일 시스템의 구조
  - 부트블록 : 부팅시 필요한 코드를 저장하고 있는 블록
  - 슈퍼블록 : 전체 파일 시스템에 대한 정보를 저장하고 있는 블록
  - I-node 블록 : 각 파일이나 디렉터리에 대한 모든 정보를 저장하고 있는 블록
  - 데이터 블록 : 실제 파일에 대한 데이터가 저장된 블록

④ 파일 디스크립터 (FD, File Descriptor)

- 파일 디스크립터 특징
  - 유닉스 시스템에서 프로세스가 파일들을 접근할 때 이용
- 파일 디스크립터 정보
  - 파일 이름 및 파일 크기
  - 보조기억장치에서의 파일 위치
  - 파일 구조(순차파일/색인순차파일/색인파일 등)
  - 보조기억장치의 유형(자기 디스크/자기테이프 등)
  - 액세스 제어 정보
  - 파일 유형(텍스트파일, 목적프로그램파일 등)
  - 생성 날짜와 시간, 제거 날짜와 시간
  - 최종 수정 날짜 및 시간
  - 액세스한 횟수(파일 사용 횟수)

⑷ MacOS

- 애플사가 개발한 유닉스 기반의 운영체제

5. 운영체제의 명령어

(1) Unix 주요 명령어

명령어	설명
access	파일의 접근 가능 여부 결정
chmod	파일 또는 디렉토리에 대한 접근권한을 변경
close	FCB(File Control Block)를 닫는다.
chgrp	파일의 그룹명 변경
chown	파일의 소유자 변경
chdir	디렉터리 변경 명령
mkdir	디렉터리 생성 명령
rmdir	디렉터리 삭제 명령
mount	파일 시스템에 새로운 파일 시스템을 연결할 때 사용
umount	파일 시스템에서 서브 디렉터리 제거시 사용
exit	프로세스 종료
kill	프로세스 제거
fork	새로운 프로세스를 생성, 복제하는 명령
getpid	자신의 프로세스 명, 그룹명, 부 프로세스의 정보를 얻는다.
getppid	부모 프로세스의 ID를 얻는다.
sleep	프로세스를 일정 시간 동안 중단
uname	현재 운영체제의 버전 정보를 확인
ps	프로세스 상태 출력
exec	새로운 프로그램을 수행시키는 명령
vi	편집기 명령어
cat	파일 내용을 화면에 출력
rm	파일이나 디렉터리 삭제
cp	파일을 복사
mv	파일 이동
grep	파일이나 프로세스를 찾는 명령
ls	파일 목록 확인
du	파일의 사용량 출력
finger	사용자 정보 표시

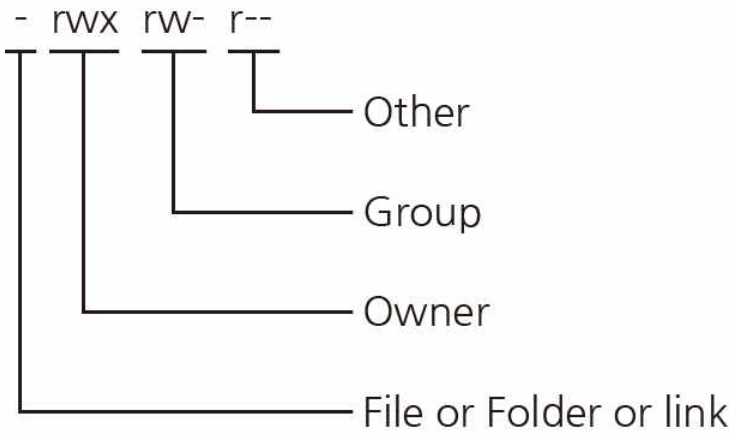
⑵ Linux, Unix 파일 접근 권한 관리

① 파일 확인 방법

- ls -al 명령을 이용하여 파일의 상세정보 및 파일 접근 권한을 확인할 수 있다.

```
[root@dbl data]# ls -al
합 계  2959848
drwxrwxr-x 2 srtplay srtplay         51  1월   7  2021 .
drwxrwxr-x 4 srtplay srtplay         30  1월   7  2021 ..
-rw-r--r-- 1 srtplay srtplay 1438807183  5월  18  2020 dump_0515.0834.sql
-rw-rw-r-- 1 srtplay srtplay 1592073446  5월  28  2020 dump_0527.sql
```

② 필드별 의미



1	2	3	4	5	6	7	8	9	10
-	R	W	-	R	-	-	R	-	-

- 1번 필드 : 타입
  - 파일 : -
  - 디렉터리 : d
  - 다른 파일을 가리키는 링크 : l
  - 두 개의 프로그램을 연결하는 파이프 파일 : p
  - 블록 단위로 하드웨어와 반응하는 파일 : b
  - 스트림 단위로 하드웨어와 반응하는 파일 : c
- 2~4번 필드 : 소유주(USER) 권한
- 5~7번 필드 : 그룹(GROUP) 권한
- 8~10번 필드 : 나머지(OTHER) 권한

③ 권한별 값

구분	값	설명
R	4	읽기 권한
W	2	쓰기 권한
X	1	실행 권한
-	0	권한 없음

④ 권한 변경

- chmod 0751 file명
  - 해당 파일에 대해서 소유주는 읽기, 쓰기, 실행 권한이 있다.
  - 해당 파일에 대해서 그룹은 읽기, 실행 권한이 있다.
  - 해당 파일에 대해서 나머지는 실행 권한이 있다.
- chmod 0775 file명
  - 해당 파일에 대해서 소유주는 읽기, 쓰기, 실행 권한이 있다.
  - 해당 파일에 대해서 그룹은 읽기, 쓰기, 실행 권한이 있다.
  - 해당 파일에 대해서 나머지는 읽기, 실행 권한이 있다.

⑤ umask (접근 권한 마스크)

- 앞으로 만들어질 파일 권한에 대한 설정
- umask 로 지정한 8진수는 새로 만들어질 파일에서 제거될 권한을 명시
- umask 022
  - 앞으로 만들어지는 파일은 644, 디렉터리는 755 권한을 가진다.
  - 소유주는 읽기, 쓰기권한
  - 그룹은 읽기 권한
  - 나머지는 읽기 권한

⑥ chown (소유주 변경)

- chown hungjik file명
  - 해당 파일의 소유주를 hungjik 로 변경

Section 2. 메모리 관리

1. 기억장치 관리 전략

⑴ 기억장치 관리 전략의 개념

- 보조기억장치의 프로그램이나 데이터를 주기억장치에 적재시키는 시기, 위치 등을 지정하여, 한정된 주기억장치의 공간을 효율적으로 사용한다.

⑵ 기억장치 관리 전략

① 반입(Fetch) 전략

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

② 배치(Placement) 전략

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 중에서 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

③ 교체(Replacement) 전략

- 주기억장치의 모든 영역이 이미 사용중인 상태에서 새로운 프로그램이나 데이터를 주기억장치에 배치하려고 할 때, 이미 사용되고 있는 영역 중에서 어느 영역을 교체하여 사용할 것인지를 결정하는 전략
- 종류 : FIFO, OPT, LRU, LFU, NUR, SCR 등

2. 단편화

⑴ 단편화의 개념

- 주기억장치에 프로그램을 할당하고 반납하는 과정에서 발생하는 사용되지 않는 작은 조각 공간

⑵ 단편화의 종류

① 내부 단편화

- 주기억장치 공간이 프로그램보다 커서 프로그램의 사용 공간을 할당 후 사용되지 않고 남아있는 공간

② 외부 단편화

- 주기억장치 공간 보다 프로그램이 커서 프로그램이 할당될 수 없어 사용되지 않고 남아있는 공간

③ 단편화 계산

영역	분할 크기	작업 크기	단편화 크기
1	20K	10K	내부:10K
2	50K	60K	외부:50K
3	120K	160K	외부:120K
4	200K	100K	내부:100K
5	300K	150K	내부:150K

⑶ 단편화 해결 방법

① 통합(Coalescing) 기법

- 인접해 있다면 두 개의 빈 분할 공간을 하나로 통합하여 효율성을 높이는 작업

② 압축(Compaction) 기법

- 주기억장치 내 분산되어 있는 단편화 공간들을 통합하여 하나의 커다란 빈 공간을 만드는 작업

③ 재배치 기법(Relocation)

- 압축을 실행하여 이 과정에서 프로그램의 주소를 새롭게 지정해주는 기법

Section 3. 가상 기억 장치

1. 가상 기억 장치

⑴ 가상 기억 장치의 개념

- 보조기억장치(하드디스크)의 일부를 주기억장치처럼 사용하는 것

⑵ 블록 분할 방법

① 페이징(Paging) 기법

- 가상기억장치를 모두 같은 크기의 블록으로 편성하여 운용하는 기법
- 주소 변환을 위해서 페이지 맵 테이블이 필요
- 페이지 크기별 비교

페이지 크기	기억장소 효율	단편화	입출력 시간	맵 테이블
클수록	감소	증가	감소	감소
작을수록	증가	감소	증가	증가

② 세그먼테이션(Segmentation) 기법

- 가상 메모리를 서로 크기가 다른 논리적 단위인 세그먼트로 분할하고 메모리를 할당하는 기법
- 아래 세그먼트 테이블에서 S=(2, 100) 의 실제 주소는 2100 임

세그먼트 번호	크기	시작주소
0	1200	4000
1	800	5700
2	1000	2000
3	500	3200

2. 가상기억장치 기타 관리 사항

⑴ 페이지 부재

- 프로세스 실행 시 참조할 페이지가 주기억 장치에 없는 현상

⑵ 지역성(Locality)

- 프로세스가 실행되는 동안 주기억장치를 참조할 때 일부 페이지만 집중적으로 참조하는 성질
- 지역성의 종류

시간 구역성 (Temporal Locality)	- 프로세스가 실행되면서 하나의 페이지를 일정 시간 동안 집중적으로 액세스 하는 현상 - 한 번 참조한 페이지는 가까운 시간 내에 계속 참조할 가능성이 높음 - Loop(반복), Stack(스택), 부 프로그램(Sub Routine) 등
공간 구역성 (Spatial Locality)	- 프로세스 실행 시 일정 위치의 페이지를 집중적으로 액세스 하는 현상 - 어느 하나의 페이지를 참조하면 그 근처의 페이지를 계속 참조할 가능성이 높음 - 배열순회, 순차적 코드 실행 등

⑶ 워킹 셋(Working Set)

- 프로세스가 일정 시간 동안 자주 참조하는 페이지들의 집합

⑷ 스래싱(Thrashing)

- 프로세스의 처리 시간보다 페이지 교체에 소요되는 시간이 더 많아지는 현상

3. 페이지 교체 알고리즘

⑴ FIFO (First In First Out)

- 가장 먼저 메모리에 적재된 페이지를 먼저 교체하는 기법
- 프레임 개수를 늘리면 부재 발생이 감소해야 하나, 오히려 더 늘어나는 Belady's Anomaly 이상현상 발생
- 페이지 프레임이 3개일 때, 페이지 결함 예

참조 페이지	1	2	3	1	2	4	1	2	5
페이지 프레임	1	1	1	1	1	4	4	4	5
		2	2	2	2	2	1	1	1
			3	3	3	3	3	2	2
페이지 부재	O	O	O	X	X	O	O	O	O

⑵ OPT (Optimal replacement, 최적 교체)

- 앞으로 가장 사용 안 될 페이지를 교체
- 페이지 부재 횟수가 가장 적게 발생하는 가장 효율적인 알고리즘이나 참조 상황을 예측하기 어려움

⑶ LRU (Least Recently Used)

- 최근에 가장 오랫동안 사용되지 않은 페이지를 교체
- 페이지 프레임이 3개일 때, 페이지 결함 예

참조 페이지	1	2	1	0	4	1	3
페이지 프레임	1	1	1	1	1	1	1
		2	2	2	4	4	4
				0	0	0	3
페이지 부재	O	O	X	O	O	X	O

⑷ LFU (Least Frequently Used)

- 사용 빈도가 가장 적은 페이지를 교체
- 페이지 프레임이 3개일 때, 페이지 결함 예

참조 페이지	1	2	3	1	2	4	1	2	5
페이지 프레임	1	1	1	1	1	1	1	1	1
		2	2	2	2	2	2	2	2
			3	3	3	4	4	4	5
페이지 부재	O	O	O	X	X	O	X	X	O

⑸ NUR(Not Used Recently)

- 최근의 사용여부를 확인하기 위해 각 페이지마다 두 개의 비트 사용
- 참조비트와 변형비트를 이용해서 페이지 교체
- 다음 중 가장 나중에 교체될 페이지는 4번

번호	1	2	3	4
참조비트	0	0	1	1
변형비트	0	1	0	1

⑹ SCR(Second Chance Replacement)

- 가장 오랫동안 주기억장치에 있던 페이지 중 자주 사용되는 페이지의 교체를 방지하기 위한 것으로, FIFO 기법의 단점을 보완하는 기법

Section 4. 프로세스

1. 프로세스

(1) 프로세스의 개념

- 컴퓨터에서 연속적으로 실행되고 있는 컴퓨터 프로그램

(2) 스레드(Thread)

① 스레드의 개념

- 프로세스 내에서 실행되는 흐름의 단위
- 프로그램은 하나 이상의 프로세스를 가지고 있고, 하나의 프로세스는 반드시 하나 이상의 스레드를 갖는다.

(3) 메모리상의 프로세스 영역

① 코드 영역

- 실행할 프로그램의 코드가 저장되는 영역

② 데이터 영역

- 전역변수와 정적변수(static) 변수가 할당 되는 부분

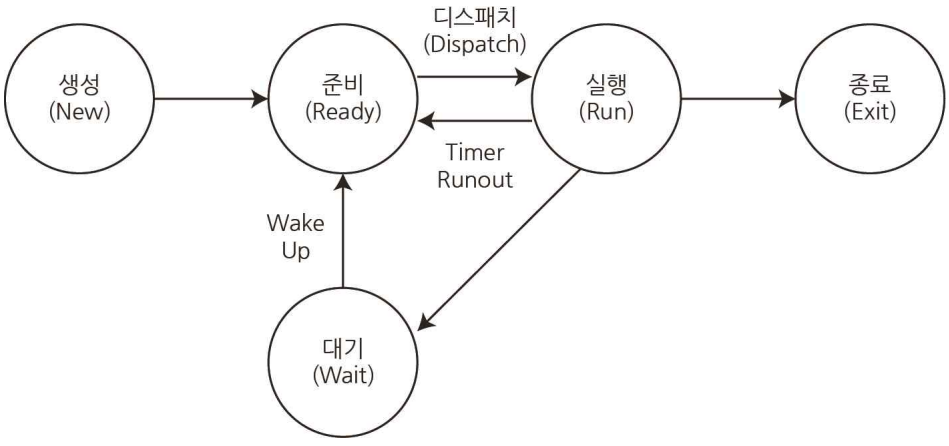
③ 힙 영역

- 프로그래머가 할당/해제하는 메모리 공간

④ 스택 영역

- 프로그램이 자동으로 사용하는 임시 메모리 영역

(4) 프로세스 상태 전이



(5) 문맥 교환(Context Switching)

- 하나의 프로세스가 CPU를 사용 중인 상태에서 다른 프로세스가 CPU를 사용하도록 하기 위해 이전의 프로세스의 상태를 PCB에 보관하고 또 다른 프로세스의 정보를 PCB에서 읽어 레지스터에 적재하는 과정

(6) PCB (Process Control Block, 프로세스 제어 블록)

- 운영체제가 프로세스에 대한 정보를 저장해 놓는 공간

2. 프로세스 스케줄링

⑴ 스케줄링(Scheduling)의 개념

- 메모리에 올라온 프로세스들 중 어떤 프로세스를 먼저 처리할지 순서를 정하는 것

⑵ 스케줄링의 목적

- 공평성
- 효율성
- 안정성
- 반응 시간 보장
- 무한 연기 방지

⑶ 스케줄링 기법

① 선점형 스케줄링 (Preemptive)

- 다른 프로세스가 실행 중이더라도 운영체제가 CPU를 강제로 뺏을 수 있는 방식
- 종류 : Round Robin, SRT, 다단계 큐, 다단계 피드백 큐 등

② 비선점형 스케줄링 (Non-Preemptive)

- 프로세스가 CPU를 점유하고 있다면 이를 빼앗을 수 없는 방식
- 종류 : FCFS, SJF, HRN, 우선순위, 기한부 등

Section 5. 병행 프로세스와 교착상태

1. 병행 프로세스

⑴ 병행 프로세스의 개념

- 두 개 이상의 프로세스들이 동시에 존재하며 실행 상태에 있는 것

⑵ 문제점과 해결책

① 문제점

- 동시에 2개 이상의 프로세스를 병행 처리하면 한정된 자원(CPU, 메모리, 디스크, I/O 장치 등)에 대한 사용 순서 등 여러 가지 문제가 발생

② 문제 해결책

- 임계구역
- 상호배제 기법
- 동기화 기법

2. 병행 프로세스 문제 해결책

⑴ 임계구역(Critical Section)

- 공유 자원에 대해서, 한 순간에는 반드시 하나의 프로세스만 사용되도록 지정한 영역

⑵ 상호 배제(Mutual Exclusion)

- 하나의 프로세스가 공유 메모리 혹은 공유 파일을 사용하고 있을 때 다른 프로세스들이 사용하지 못하도록 배제시키는 제어 기법

⑶ 동기화 기법

- 스레드들에게 하나의 자원에 대한 처리 권한을 주거나 순서를 조정해주는 기법
- 세마포어(Semaphore)
  - 각 프로세스에 제어 신호를 전달하여 순서대로 작업을 수행하도록 하는 기법
  - P와 V라는 2개의 연산에 의해서 동기화를 유지시키고, 상호 배제의 원리를 보장
- 모니터(Monitor)
  - 프로그래밍 언어 수준에서 동시성을 제어하여 타이밍 오류를 해결한 상호 배제 기법

3. 교착상태(Dead Lock)

(1) 교착상태의 개념

- 상호 배제에 의해 나타나는 문제점으로, 둘 이상의 프로세스들이 자원을 점유한 상태에서 서로 다른 프로세스가 점유하고 있는 자원을 요구하며 무한정 기다리는 현상

(2) 교착상태 발생 조건

- 상호배제(Mutual Exclusion)
  - 한 번에 한 개의 프로세스만이 공유 자원을 사용할 수 있어야 함
- 점유와 대기(Hold & Wait)
  - 자원을 점유하고 있으면서 다른 프로세스에 할당되어 있는 자원을 추가로 요구하며 대기
- 비선점(nonpreemption)
  - 프로세스에 할당된 자원은 사용이 끝날 때까지 강제로 빼앗을 수 없음
- 환형대기(Circular Wait)
  - 각 프로세스가 순차적으로 다음 프로세스가 요구하고 있는 자원을 가지고 있는 상태

(3) 교착상태 해결 방법

- 예방 기법(Prevention)
  - 교착 상태가 발생되지 않도록 사전에 시스템을 제어하는 방법
- 회피 기법(Avoidance)
  - 교착 상태가 발생하려고 할 때, 교착상태 가능성을 피해가는 방법
  - 주로 은행원 알고리즘(Banker's Algorithm)이 사용
- 발견 기법(Detection)
  - 시스템에 교착 상태가 발생했는지 점검하여 교착 상태에 있는 프로세스와 자원을 발견하는 것
- 회복 기법(Recovery)
  - 교착상태의 프로세스에 할당된 자원을 선점하여 프로세스나 자원을 회복하는 것
  - 교착 상태를 일으킨 프로세스를 종료시킴으로 선점을 해제한다.

Section 6. 디스크 스케줄링(Disk Scheduling)

1. 디스크 스케줄링

⑴ 디스크 스케줄링 개념

- 사용할 데이터가 디스크 상의 여러 곳에 저장되어 있을 경우, 데이터를 액세스하기 위해 디스크 헤드를 움직이는 경로를 결정하는 기법

⑵ 디스크 스케줄링 목표

- 하드 디스크 검색으로 낭비되는 시간을 최소화
- 특정한 프로세스의 입출력 요청의 우선순위를 정함
- 디스크 대역을 실행중인 각 프로세스에 할당
- 정해진 기한까지 요청을 처리

⑶ 디스크 스케줄링 종류

① FCFS 스케줄링(First Come First Served)

- 요청이 들어온 순서대로 처리
- FCFS 적용 예(현재 헤드의 위치가 53이라고 가정할 경우)

큐의 내용	98	183	37	122	14	124	65	67	합계
이동 순서	1	2	3	4	5	6	7	8	
이동 거리	45	85	146	85	108	110	59	2	640

② SSTF(Shortest Seek Time First)

- 현재 헤드에서 가장 가까운 트랙의 요청을 먼저 처리한다.
- SSTF 적용 예(현재 헤드의 위치가 53이라고 가정할 경우)

큐의 내용	98	183	37	122	14	124	65	67	합계
정렬	14	37	65	67	98	122	124	183	
이동 순서	4	3	1	2	5	6	7	8	
이동 거리	23	30	12	2	84	24	2	59	236

③ SCAN

- 헤드의 진행방향에 있는 요청을 처리하고, 다시 반대 방향으로 틀어 반대방향에 있는 요청들을 처리한다.

④ C-SCAN

- 항상 한쪽 방향에서 반대방향으로 진행하며 트랙의 요청을 처리한다.

⑤ LOOK

- SCAN 기법을 기초로 사용하며, 진행 방향의 마지막 요청을 처리한 후 반대 방향으로 처리하는 기법

⑥ C-LOOK

- C-SCAN 기법을 기초로 사용하며, 바깥쪽에서 안쪽 방향의 모든 요청을 처리한 후, 가장 바깥쪽으로 이동한 후 다시 안쪽 방향으로 서비스 하는 기법

⑦ N-STEP SCAN

- SCAN 기법을 기초로 두고 있으며, 시작하기 전 대기하고 있는 요청들을 우선적으로 처리하고, 처리하는 과정 중에 요청이 들어오는 것들은 이후에 모아서, 반대방향으로 진행할 때 서비스한다.

⑧ 에션바흐(Eschenbach)기법

- 부하가 매우 큰 항공 예약 시스템을 위해 개발
- 탐색 시간과 회전 지연 시간을 최적화하기 위한 최초의 기법

Section 7. 환경변수와 로그 파일

1. 환경변수

⑴ 환경변수의 개념

- 프로세스가 컴퓨터에서 동작하는 방식에 영향을 미치는 동적인 값들의 모임

⑵ UNIX/Linux 환경변수

- env, set, printenv 명령어들을 사용하여 환경 변수와 그에 따른 모든 값을 볼 수 있다.
- export 명령을 이용하여 사용자 환경 변수를 전역 변수로 설정할 수 있다.
- 환경변수 종류

환경변수	설명
$PATH	디렉터리의 경로
$HOME	사용자의 홈 디렉터리
$LANG	기본 지원되는 언어
$USER	사용자의 이름
$TERM	로그인 터미널 타입
$PS1	1차 명령 프롬프트 변수
$HISTFILE	히스토리 파일의 절대 경로
$MAIL	도착한 메일이 저장되는 경로
$TMOUT	로그인 후 일정시간 작업을 하지 않을 경우 로그아웃 시키는 시간
$UID	사용자의 UID
$PWD	사용자의 현재 작업 디렉토리
$DISPLAY	X 윈도우에서 프로그램 실행시 출력되는 창
$OSTYPE	운영체제 타입

2. 로그 파일

(1) 로그의 개념

- 시스템의 모든 기록을 담고 있는 데이터

(2) 로그 데이터 정보

- 외부로부터의 침입 감지 및 추적
- 시스템 성능관리
- 마케팅 전략으로 활용
- 시스템의 장애 원인 분석
- 시스템 취약점 분석

(3) 리눅스 로그 종류

종류	설명
messages	시스템 로그 파일
secure	보안인증에 관한 메세지 로그파일
maillog	메일 로그 파일
xferlog	ftp 로그파일
dmesg	부팅 시의 시스템 로그
wtmp	시스템에 로그인 기록이 저장되는 파일(전체 로그인 기록)
utmp	시스템에 로그인 기록이 저장되는 파일(현재 로그인 사용자에 대한 기록)
lastlog	각 계정들의 가장 최근 로그인 기록

15 네트워크

Section 1. 네트워크 기본

1. 네트워크

(1) 네트워크 개념

- 컴퓨터와 같은 노드들이 통신 기술을 이용하여 그물망처럼 연결된 통신 이용 형태

(2) 거리 기반 네트워크

- PAN(Personal Area Network)
  - 5m 전후의 인접 통신
- LAN(Local Area Network)
  - 근거리 네트워크로 사무실과 같은 소규모 공간 내의 고속 통신 회선
- MAN(Metropolitan Area Network)
  - LAN과 WAN의 중간 형태
- WAN(Wide Area Network)
  - 광대역 네트워크망으로 유관한 LAN간의 연결

2. 네트워크 토폴리지(Network Topology)

(1) 계층형(Tree)

(2) 버스형(Bus)

(3) 성형(Star)

(4) 링형(Ring)

(5) 망형(Mesh)

3. 데이터 전송

(1) 아날로그/디지털 전송

① 아날로그 전송

- 전송 매체를 통해 전달되는 신호가 아날로그 형태인 것
- 신호 감쇠 현상이 심하고, 오류의 확률이 높음

② 디지털 전송

- 전송 매체를 통해 전달되는 신호가 디지털 형태인 것
- 제한된 거리에서의 감쇄 현상은 없으나 전송거리의 제한을 극복하기 위해 리피터(Repeater) 사용
- 장거리 전송이 가능

⑵ 방향에 따른 구분

① 단방향 통신(Simplex)

- 일방적으로 'A → B' 의 통신만 가능한 전송 방식(ex. 라디오, TV)

② 반이중 통신(Half Duplex)

- 서로 데이터를 전송할 수 있지만, 하나의 회선을 사용하기 때문에 동시에 전송은 불가능(ex. 무전기)

③ 전이중 통신(Full Duplex)

- 서로 언제나 필요한 데이터를 동시에 송수신 할 수 있는 전송(ex. 전화)

⑶ 직렬전송/병렬전송

① 직렬전송(Serial Transmission)

- 한 번에 한 비트씩 순서대로 전송
- 데이터 전송 속도 느림
- 구축이 쉽고 경제적

② 병렬전송(Parallel Transmission)

- 문자 단위 등 여러 비트를 동시에 전송하는 방식
- 데이터 전송 속도 빠름
- 흐름제어 필요

⑷ 동기 전송/비동기 전송

① 동기식 전송 방식(Synchronous Transmission)

- 한 문자단위가 아니라 여러 문자를 수용하는 데이타블럭 단위로서 전송하는 방식
- 양측에 설치된 모뎀이나 다중화기 등과 같은 기기에 의해 타이밍 조정

② 비동기식 전송 방식(Asynchronous Transmission)

- 작은 비트블럭의 앞뒤에 각각 start bit와 stop bit를 삽입하여 동기화하는 방식

③ 동기/비동기 비교

구분	동기식 전송 방식	비동기식 전송 방식
통신 속도	고속	저속
회로 복잡도	복잡	단순
구축 비용	고가	저가
동기 제어 방식	클럭 동기	Start bit, Stop bit
전송 단위	블록 단위 전송	문자 단위 전송
적용 예	전화 교환망, ATM, 데이터 통신망	RS-232C

Section 2. 근거리 통신망(LAN, Local Area Network)

1. LAN

⑴ LAN의 개념

- 여러 대의 컴퓨터와 주변장치 등이 통신 네트워크를 구성하여 통신하는 망

⑵ LAN의 전송방식

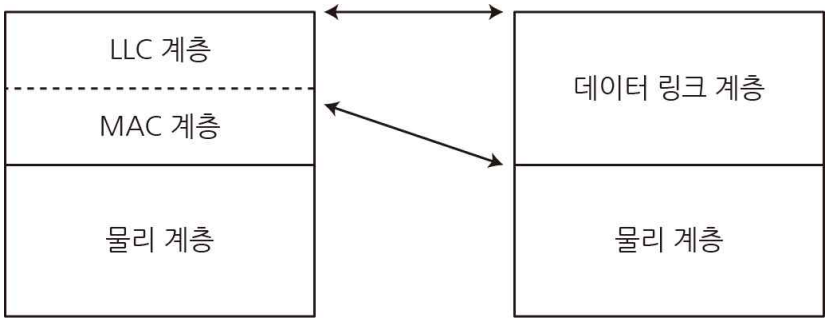
① 베이스밴드(Base Band)

- 컴퓨터나 통신장치의 디지털 신호를 변조하지 않고 전송로를 이용하여 그대로 전송하는 방식

② 브로드 밴드(Broad Band)

- 디지털 데이터를 모뎀을 이용하여 아날로그 데이터로 변조하여 전송하는 방식

⑶ LAN의 프로토콜



- LLC(Logical Link Control)
  - OSI에서 데이터 링크 계층 기능 담당 (흐름제어, 오류처리 등)
- MAC(Medium Access Control)
  - 물리적 전송 선로의 특징과 매체 간 연결 방식 제어
  - CSMA/CD, 토큰 링, 토큰 버스

2. LAN의 표준 802.X 시리즈

표준	설명
802.1	- 전체의 구성, OSI 참조 모델과의 관계, 표준 규약
802.2	- 논리링크제어(LLC)에 관한 규약
802.3	- CSMA/CD에 관한 규약
802.4	- 토큰 버스에 관한 규약
802.5	- 토큰 링에 관한 규약
802.11	- 무선 LAN에 관한 규약
802.15	- 블루투스에 관한 규약

⑴ CSMA/CD(Carrier Sense Multiple Access with Collision Detection)

① CSMA/CD 개념

- IEEE 802.3 이더넷 LAN에서 사용되는 매체접근방식
- 유선 네트워크에서 충돌을 확인할 수 있는 방식

② 용어의 의미

- CS(Carrier Sense Multiple Access)
  - 채널 사용 전 다른 이용자가 있는지 확인하는 방식
- MA(Multiple Access)
  - 누구든 동시에 접근할 수 있는 방식
- CD(Collision Detection)
  - 충돌을 검사하여 제어하는 통신 방식

⑵ CSMA/CA(Carrier Sense Multiple Access with Collision Avoidance)

① CSMA/CA 개념

- IEEE 802.11 무선 LAN에서 사용되는 매체접근방식
- 무선 네트워크에서 충돌을 감지하기 힘들기 때문에 CSMA/CD대신 CSMA/CA 사용
- CSMA 방식에 충돌 회피 기능 추가
- 다른 컴퓨터가 네트워크를 사용중인지 판단하여, 사용중이라면 일정시간 동안 대기한다.

② 용어의 의미

- CS(Carrier Sense Multiple Access)
  - 채널 사용 전 다른 이용자가 있는지 확인하는 방식
- MA(Multiple Access)
  - 누구든 동시에 접근할 수 있는 방식
- CA(Collision Avoidance)
  - 충돌을 검사하여 피하는 통신 방식

⑶ 토큰 버스(Token Bus)

- 버스형(Bus) LAN에서 사용하는 방식으로, 토큰이 논리적으로 형성된 링(Ring)을 따라 각 노드들을 차례로 옮겨 다니는 방식

⑷ 토큰 링(Token Ring)

- 링형(Ring) LAN에서 사용하는 방식으로, 물리적으로 연결된 링(Ring)을 따라 순환하는 토큰(Token)을 이용하여 송신 권리를 제어
- 토큰 상태
  - 프리 토큰(Free Token) : 회선을 사용할 수 있는 상태
  - 비지 토큰(Busy Token) : 회선이 데이터 전송에 사용 중

⑸ 블루투스 규약(802.15)

버전	내용
802.15.1	- 'Bluetooth'를 기반으로 한 WPAN(Wireless Personal Area Network) 규격
802.15.2	- WPAN 및 WLAN을 동시에 사용, 상호 간섭 해소 등 공존
802.15.3	- 20Mbps 이상의 고속의 WPAN 규격
802.15.4	- 저속의, 저전력, 저가형 WPAN 규격
802.15.5	- WPAN에 의한 `Mesh Network` 구성
802.15.6	- Body Area Network (체온, 심전도, 맥박, 삼축 가속도 등의 측정 기능)
802.15.7	- 가시광선 통신 (Visible Light Communication)

Section 3. 데이터 교환 방식과 다중화

1. 데이터 교환 방식

⑴ 회선망의 종류

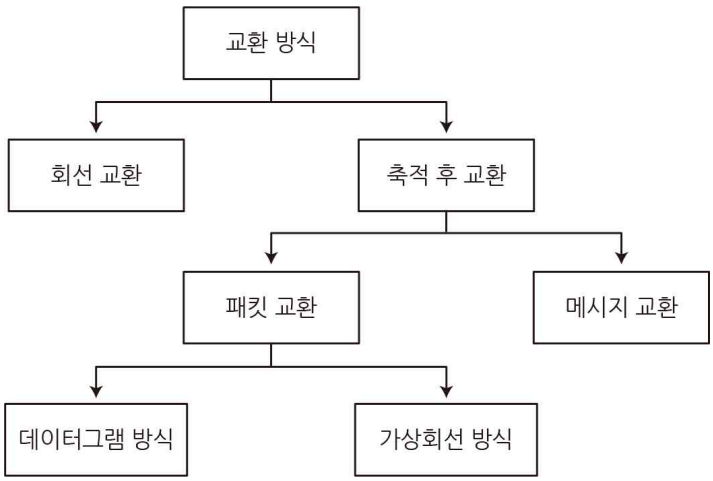
① 전용회선

- 통신회선이 항상 고정되어 있는 방식
- 송수신 측을 일대일로 연결
- 고가의 비용이 발생

② 교환회선

- 교환기에 의해 송/수신 상호간이 연결되는 방식
- 정보 보안을 위해 기밀성, 무결성을 고려해야 함
- 통신 장치와 회선 비용을 줄일 수 있다.

⑵ 데이터 교환 방식



2. 다중화(Multiplexing)

⑴ 다중화의 개념

- 하나의 통신 회선을 여러 가입자들이 동시에 사용하도록 하는 기능

⑵ 다중화기(MUX, MultipleXer)

- 여러 개의 터미널 신호를 하나의 통신 회선을 통해 전송할 수 있도록 하는 장치장치

⑶ 다중화기 종류

① 주파수 분할 다중화기 (FDM, Frequency Division Multiplexer)

- 하나의 물리적 통신 채널을 여러 주파수 채널로 나누어 사용하는 다중화 방식

② 시분할 다중화기 (TDM, Time Division Multiplexer)

- 한 전송로의 데이터 전송 시간을 일정한 시간 폭으로 나누어 차례로 분배하는 방식

- 다중화 방식

동기식 시분할 다중화 (Synchronous TDM)	- 실제 송신할 데이터의 존재 유무에 관계없이 타임슬롯을 할당하여 전송 - 전송할 데이터가 없는 장치도 타임슬롯이 할당되므로 효율성이 떨어진다.
비동기식 시분할 다중화 (Asynchronous TDM)	- 실제로 전송할 데이터가 있는 장치에만 타임슬롯을 할당 - 동기식 다중화에 비해 전송 효율이 높다. - 통계적 시분할 다중화 방식, 지능형 다중화 방식이라고도 한다.

③ 코드 분할 다중화 (Code Division Multiplexer, CDM)

- 고유의 코드를 이용한 다중화 방식

④ 파장 분할 다중화 (Wavelength Division Multiplexing, WDM)

- 여러 파장대역을 통해, 동시에 전송하는 광 다중화 방식

⑤ 공간 분할 다중화 (Space-Division Multiplexing, SDM)

- 시간(TDM) 또는 주파수(FDM)가 아닌 공간 차원(SDM)에서 다중화하는 기술

(4) 역다중화기와 집중화기

① 역다중화기 (Inverse MUX)

- 하나의 고속회선으로부터 데이터를 받아 여러 개의 저속회원으로 쪼개어 전송하는 것

② 집중화기(Concentrator)

- 여러 개의 저속회선으로부터 전송된 데이터를 버퍼에 축적한 후 이를 모아서 고속회선으로 전송하는 것

Section 4. 인터넷

1. 인터넷

(1) 인터넷(Internet)의 개념

- TCP/IP 프로토콜을 기반으로 하여 전 세계 수많은 컴퓨터와 네트워크들이 연결된 광범위한 컴퓨터 통신망

(2) 인터넷 서비스

- WWW (Wold Wide Web)
- 전자우편 (E-MAIL)
  - SMTP, POP3, MIME 프로토콜을 사용
- 텔넷 (Telnet)
  - 멀리 떨어져 있는 컴퓨터에 접속하여 자신의 컴퓨터처럼 사용할 수 있도록 해주는 서비스
- HTTP (Hyper Text Transfer Protocol)
  - 하이퍼 텍스트 문서를 전송하기 위해 사용되는 프로토콜
- FTP (File Transfer Protocol)
  - 파일 전송 프로토콜
- 아키 (Archie)
  - 익명의 FTP 사이트에 있는 FTP 서버와 그 안의 파일 정보를 데이터베이스에 저장해 두었다가 FTP서버의 리스트와 파일을 제공함으로써 정보를 쉽게 검색할 수 있도록 하는 서비스
- 고퍼 (Gopher)
  - 메뉴 방식을 이용해 손쉽게 정보 검색을 할 수 있도록 하는 서비스
- 유즈넷(USENET)
  - 분야별로 공통의 관심사를 가진 인터넷 사용자들이 서로의 의견을 주고받을 수 있게 하는 서비스

2. IP

(1) IP(Internet Protocol address) 주소

① IP의 개념

- 인터넷에서 컴퓨터를 식별할 수 있는 고유한 번호
- 네트워크 부분과 호스트 부분을 구분하기 위해 서브넷 마스크(Subnet Mask)를 사용한다.

② IP 주소 클래스

클래스	옥텟 IP	최상비트	호스트 수	네트워크 수	용도
A Class	0 ~ 127	0	16,777,216	128	국가/대형 통신망
B Class	128 ~ 191	10	65,536	16,384	중대형 통신망
C Class	192 ~ 223	110	256	2,097,152	소규모 통신망
D Class	224 ~ 239	1110			멀티캐스트용
E Class	240 ~ 255	1111			실험용

⑵ IPv6

① IPv6의 개념

- IPv4의 주소 고갈 문제를 해결하기 위하여 기존의 IPv4주소 체계를 128비트 크기로 확장한 차세대 인터넷 프로토콜 주소

② 특징

- 헤더의 내용을 확인하는 데 소요되는 오버헤드를 최소화하도록 설계
- 주소를 128비트로 표현한 확장된 주소 공간
- 계층적 주소 할당 체계
- 자동화된 주소 설정
- 기본으로 제공되는 보안기능
- 개선된 QoS 지원
- 헤더를 추가할 수 있는 확장성
- 패킷 크기 확장

③ 표시형식

- 16비트씩 8부분, 128비트로 구성되며, 콜론(:)으로 구분한다.
- EX) 2001:0DB8:1000:0000:0000:0000:1111:2222

④ IPv4/IPv6 전환기술

- 듀얼 스택(Dual Stack)
  - 장비들이 IPv4 및 IPv6 모두 지원, 동시 처리 가능
- 터널링(Tunneling)
  - IPv6 패킷을 IPv4 패킷 속에 캡슐화하여 사용하는 기술
- 주소 변환(Address Translation)
  - IPv6 시스템이 IPv4 수신자가 이해할 수 있는, 또는 그 반대로 헤더 변환하는 기술

⑶ IPv4 와 IPv6 비교

구분	IPv4	IPv6
주소길이	32비트	128비트
표시방법	8비트씩 4부분, 10진수로 표시	16비트씩 8부분, 16진수로 표시
주소개수	약 43억개	43억 * 43억 * 43억 * 43억
주소할당	비 순차적 할당	순차적 할당
품질제어	지원 수단 없음	품질보장이 용이
보안기능	IPSec 프로토콜 별도 설치	확장기능에서 기본으로 제공
플로그앤플레이	지원 수단 없음	지원 수단 있음
모바일 IP	곤란	용이
주소 유형	유니캐스트, 멀티캐스트, 브로드캐스트	유니캐스트, 멀티캐스트, 애니캐스트

3. 서브넷

(1) 서브넷, 서브넷 마스크

① 서브넷(Subnet)

- 하나의 네트워크가 분할되어 나눠진 작은 네트워크

② 서브네팅(Subnetting)

- 네트워크 성능 보장, 자원을 효율적으로 분배하기 위해 네트워크 영역과 호스트 영역을 쪼개는 작업

③ 서브넷 마스크(Subnet Mask)

- IP 주소에 대한 네트워크 아이디와 호스트 아이디를 구분하기 위해서 사용
- Mask 연산(AND 연산)을 활용하여 구분

4. IP 기타기술

(1) NAT(Network Address Translation)

① NAT의 개념

- 외부서 알려진 공인 IP 주소와 사설 IP 주소를 사용하는 내부 네트워크에서 IP 주소를 변환
- 제한된 수의 인터넷 IPv4 주소 문제를 해결하기 위해 개발

② 주소 할당 방식에 따른 NAT 종류

- Static NAT
  - 공인 IP주소와 사설 IP주소가 1:1로 매칭되는 방식
- Dynamic NAT
  - 여러 개의 공인 IP 주소 대비 사설 IP 개수가 많을 경우 사용하는 방식
- PAT(Port Address Translation)
  - 공인 IP 주소 1개에 사설 IP 주소 여러 개가 매칭되는 방식

(2) DNS(Domain Name System)

- Domain Name을 IP Address로 바꾸어 주거나, 그 반대의 작업을 처리하는 시스템

Section 5. 프로토콜

1. 프로토콜

(1) 프로토콜의 개념

- 컴퓨터나 통신장비들 사이에서 원활한 데이터 교환을 수행하기 위해 표준화한 통신 규약

(2) 통신 프로토콜의 기본요소

- 구문(Syntax)
  - 전송하고자 하는 데이터의 형식, 부호화, 신호 레벨 등을 규정
- 의미(Semantics)
  - 두 기기 간의 효율적이고 정확한 정보 전송을 위한 협조 사항과 오류 관리를 위한 제어 정보를 규정
- 타이밍(Timing)
  - 두 기기 간의 통신 속도, 메시지의 순서 제어 등을 규정

(3) 프로토콜의 기능

- 단편화와 재결합
- 캡슐화(Encapsulation)
- 흐름제어(Flow Control)
- 오류제어(Error Control)
- 혼잡제어(Congestion Control)
- 동기화(Synchronization)
- 순서 제어(Sequencing)
- 주소 지정(Addressing)
- 다중화(Multiplexing)
- 경로 제어(Routing)

2. 흐름제어와 오류제어

(1) 흐름제어

① 흐름제어의 개념

- 수신 측의 처리 능력에 따라 송신 측에서 송신하는 데이터의 전송량이나 전송 속도를 조절하는 기능
- Stop and Wait 방식, Sliding Window 방식

② 피기배킹(piggybacking)

- 양방향으로 동시에 정보 프레임과 응답 프레임을 교차하여 전송하는 경우를 사용하는 방식

(2) 오류제어

① 오류제어의 개념

- 전송중에 발생하는 오류를 검출하고 정정하여 데이터나 제어 정보의 파손에 대비하는 기능
- TCP는 기본적으로 ARQ(Automatic Repeat Request), 재전송 기반 오류 제어를 사용한다.
- Stop and Wait 방식, Go Back N 방식, Selective Repeat, Adaptive ARQ 방식

⑶ 오류 발생 원인

① 감쇠(Attenuation)

② 지연 왜곡(Delay Distortion)

③ 상호 변조 잡음(Intermodulation Noise)

④ 충격 잡음(Impluse Noise)

⑷ 전송 오류 제어 방식

① 전진 오류 수정(Forward Error Correction, FEC)

- 재전송 요구 없이 수신 측에서 스스로 오류 검출 및 수정하는 방식
- 에러 발생할 경우 송신측에 통보하지 않음
- 오류정정을 위한 제어비트가 추가되어 효율이 떨어짐
- 해밍코드, 상승코드 방식

② 후진 오류 수정(Backward Error Correction, FEC)

- 송신 측에 재전송을 요구하는 방식
- 패리티 검사, CRC, 블록 합 방식으로 오류를 검출하고, 오류 제어는 ARQ에 의해 이루어진다.

⑸ 오류 검출

① 패리티(Parity) 검사

- 데이터 한 블록 끝에 1비트의 검사 비트인 패리티 비트를 추가하여 전송 에러를 검출하는 방식
- 홀수 개의 오류만 검출할 수 있고, 짝수 개의 오류는 검출하지 못하는 문제점이 발생
- 이를 해결하기 위해 Two-dimensional Parity Check(2차원 패리티 검사)가 있다.

② 순환 중복 검사(Cyclic Redundancy Chcek, CRC)

- 데이터에 오류가 발생했는지 확인하는 코드를 데이터 뒤에 확장 데이터를 덧붙여 보내는 방식
- 프레임 단위로 오류 검출을 위한 코드를 계산하여 프레임 끝에 FCS(Frame Check Sequence)를 추가
- 집단적으로 발생하는 오류에 대해 신뢰성 있는 오류검출

③ 체크섬(Checksum)

- 간단하게 에러검출을 하는 방법
- 간단한 방식이기는 하나, 워드의 순서가 바꾸어지는 오류에 대한 검출은 하지 못함

④ 해밍코드(Hamming code)

- 수신측에서 직접 자기 정정 부호의 하나로 오류를 검출하고 수정까지 함
- 1Bit 의 오류만 수정가능
- 검출 가능한 최대 오류의 수 : 해밍거리 - 1
- 정정 가능한 최대 오류의 수 : (해밍거리 - 1) / 2

⑤ 상승코드

- 순차적 디코딩과 한계값 디코딩을 사용하여 오류수정
- 해밍코드처럼 검출과 정정 가능
- 여러 비트의 오류도 수정 가능

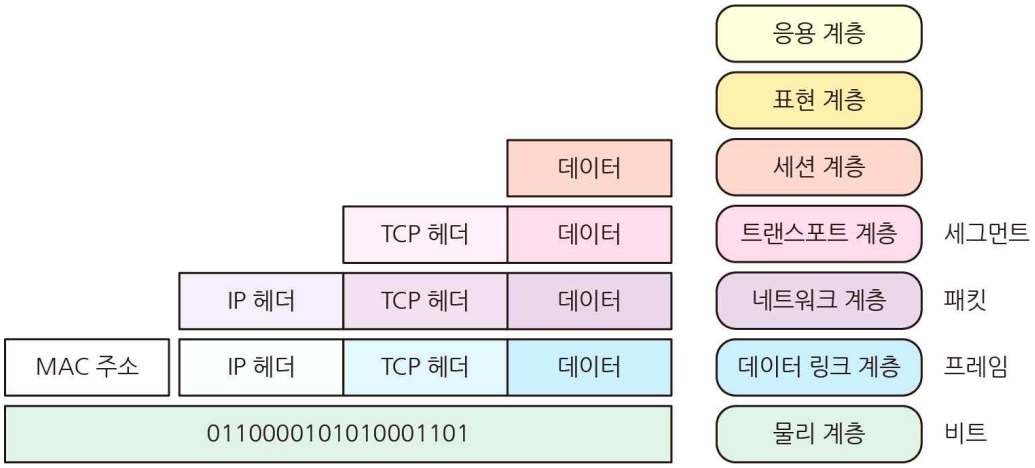
Section 6. OSI 7계층

1. OSI (Open System Interconnection) 7계층

(1) OSI 7계층 개념

- 네트워크 프로토콜 디자인, 통신을 7개의 계층으로 나누어 설명한 모델

(2) OSI 7계층 구조



2. 계층별 특징

(1) 물리계층(Physical Layer)

- 전기적, 기계적, 기능적인 특성을 이용해서 통신 케이블로 데이터를 전송
- 장비 : 통신 케이블, 리피터, 허브

(2) 데이터 링크계층(DataLink Layer)

- 포인트 투 포인트(Point to Point) 간 신뢰성 있는 전송을 보장하기 위한 계층
- 장비 : 스위치, 브리지

(3) 네트워크 계층(Network Layer)

- 데이터를 목적지까지 가장 안전하고 빠르게 전달하는 기능(라우팅)
- 주소부여(IP), 경로설정(Route)
- 장비 : 라우터, L3 스위치

(4) 전송 계층(Transport Layer)

- 양 종단간(End to end)의 사용자들이 신뢰성있는 데이터를 주고받을 수 있도록 해준다.
- 시퀀스 넘버 기반의 오류 제어 방식을 사용
- TCP, UDP 프로토콜이 있는 계층

(5) 세션 계층(Session Layer)

- 양 끝단의 응용 프로세스가 통신을 관리하기 위한 방법을 제공

(6) 표현 계층(Presentation Layer)

- 데이터 표현이 상이한 응용 프로세스의 독립성을 제공하고, 암호화 한다.

⑺ 응용 계층(Application Layer)

- 데이터의 최종 목적지로서 HTTP, FTP, SMTP, POP3, IMAP, Telnet 등과 같은 프로토콜이 있다.
- 브라우저나, 메일 프로그램은 프로토콜을 보다 쉽게 사용하게 해주는 응용프로그램이다.

3. 네트워크 장비

- Lan 카드
  - PC 혹은 네트워크에서 전달되어 오는 정보를 상호 교환할 수 있도록 만들어준다.
- 허브(Hub)
  - 집중화 장비라고도 하며, 단순히 노드들을 연결시켜주는 역할
- 리피터(Repeter)
  - 디지털 신호를 증폭시켜주는 역할
- 브리지(Bridge)
  - 두 개의 근거리 통신망을 서로 연결해주는 장치
- 스위칭허브
  - 스위치 기능을 가진 허브
- 라우터(Router)
  - 패킷의 위치를 추출하여, 그 위치에 대한 최적의 경로 지정
- 게이트웨이
  - 프로토콜을 서로 다른 통신망에 접속할 수 있게 해주는 장치

4. 백본(BackBone)

⑴ 백본 네트워크

- 기간망으로 불리는 대규모 패킷 통신망이다.

⑵ 백본 스위치

- 네트워크 중심에 위치하며 모든 패킷이 지나가는 역할

⑶ 스위치의 종류

① L2 스위치

- 데이터 링크 계층에서 운용되는 스위치

② L3 스위치

- 인터넷 계층에서 운용되는 스위치
- 라우팅 기능이 탑재되어 있다.

③ L4 스위치

- 전송 계층에서 운용되는 스위치
- 서버나 네트워크의 트래픽을 로드밸런싱 한다.

④ L7 스위치

- 응용 계층까지 운용되는 스위치
- 응용 계층 패킷까지 분석하여 보안 장비에 주로 사용된다.

Section 7. TCP/IP

1. TCP/IP(Transmission Control Protocol / Internet Protocol)

(1) TCP/IP 개념

- 현재 인터넷에서 사용되는 프로토콜로 시스템간 네트워크 연결과, 데이터를 전송하는데 사용하는 모델

(2) TCP/IP 4계층 구조

OSI 7계층	TCP/IP 4계층	프로토콜
응용계층	응용 계층	HTTP, FTP, SMTP, DNS, RIP, SNMP, DHCP
표현계층		
세션계층		
전송계층	전송 계층	TCP, UDP
네트워크 계층	인터넷 계층	IP, ICMP, IGMP, ARP, RARP
데이터 링크 계층	네트워크 엑세스 계층	Ethernet, X.25, RS-232C
물리 계층		

2. 계층별 특징

(1) 네트워크 액세스 계층(Network Access Layer)

- OSI 7계층의 물리계층과 데이터 링크 계층에 해당함
- 물리적인 주소로 MAC을 사용한다.
- 프로토콜

프로토콜	설명
Ethernet	- 물리 계층과 데이터 링크 계층의 통신 회선의 접근 제어를 정의하는 IEEE 표준
X.25	- DTE와 DCE간의 인터페이스를 제공, 패킷 교환망을 통해 패킷을 원활히 전달하기 위한 통신 프로토콜 - TCP/IP보다 느리지만 안정성과 보안성은 더 뛰어나다.
RS-232C	- 공중전화 교환망(PSTN)을 통한 DTE/DCE 접속 규격

⑵ 인터넷 계층(Internet Layer)

- OSI 7계층의 네트워크 계층에 해당함
- 여러 개의 패킷 교환망들의 상호 연결을 위한 비연결성 프로토콜
- 통신 노드 간의 IP패킷을 전송하는 기능과 라우팅 기능을 담당한다.
- 프로토콜

프로토콜	설명
IP	- 여러 개의 패킷 교환망들의 상호 연결을 위한 비연결성 프로토콜
ICMP	- 인터넷 제어 메시지 프로토콜 - IP 패킷 전송 중 에러 발생 시, 에러 발생 원인을 알려주거나 네트워크 상태를 진단해주는 기능 제공
IGMP	- 호스트가 멀티캐스트 그룹 구성원을 인접한 라우터에게 알리는 프로토콜
ARP	- IP 주소를 MAC 주소로 변환한다. - 네트워크 상에서 IP주소를 물리적 네트워크 주소(MAC)로 대응시키기 위해 사용되는 프로토콜
RARP	- 호스트의 물리적 주소로부터 IP 주소를 구할 수 있도록 하는 프로토콜

⑶ 전송 계층(Transport Layer)

- OSI 7계층의 전송 계층에 해당함
- 통신 노드 간의 연결을 제어하고, 신뢰성 있는 데이터 전송을 담당한다.
- 프로토콜

프로토콜	설명
TCP	- 클라이언트와 서버가 연결된 상태에서 데이터를 주고받는 프로토콜 - TCP는 데이터를 정확하고 안정적으로 전달할 수 있다. - 데이터의 전송 순서를 보장한다. - 연결의 설정 (3-way handshaking) - 연결의 해제 (4-way handshaking) - UDP 보다 전송 속도가 느리다. - 헤더정보 : 송수신자의 포트번호, 시퀀스 번호, 응답번호, 데이터 오프셋, 예약필드, 제어비트, 윈도우 크기, 체크섬, 긴급위치
UDP	- 데이터를 주고받을 때 연결 절차를 거치지 않고 발신자가 일방적으로 데이터를 발신하는 프로토콜 - TCP보다는 빠른 전송을 할 수 있지만 데이터 전달의 신뢰성은 떨어진다. - 중간에 패킷이 유실이나 변조가 되어도 재전송을 하지 않는다. - 헤더정보 : 송수신자의 포트번호, 데이터의 길이, 체크섬

(4) 응용 계층(Application Layer)

- 사용자와 가장 가까운 계층으로 사용자가 소프트웨어 application과 소통할 수 있게 해준다.
- 응용프로그램(application)들이 데이터를 교환하기 위해 사용되는 프로토콜
- 프로토콜

프로토콜		설명
TCP 프로토콜	HTTP	- 서버와 클라이언트 간에 하이퍼텍스트 문서를 송수신하는 프로토콜 - 80 포트 사용
	FTP	- 인터넷에서 파일을 전송하는 기본 프로토콜 - Data 전달 시 : 20 포트, 제어정보 전달 시 : 21 포트
	SMTP	- 이메일 전송에 사용되는 네트워크 프로토콜 - 25 포트 사용
UDP 프로토콜	DNS	- 호스트의 도메인 이름을 네트워크 주소로 바꿔주는 프로토콜 - 53 포트 사용
	SNMP	- 네트워크에 있는 장비들을 관리하기 위한 프로토콜
	DHCP	- IP 자동 할당과 분배 기능

3. TCP/IP 헤더

(1) IP(Internet Protocol)

① IP의 특징

- 호스트간의 통신만을 담당
- 비신뢰성(unreliability)과 비연결성(connectionlessness)

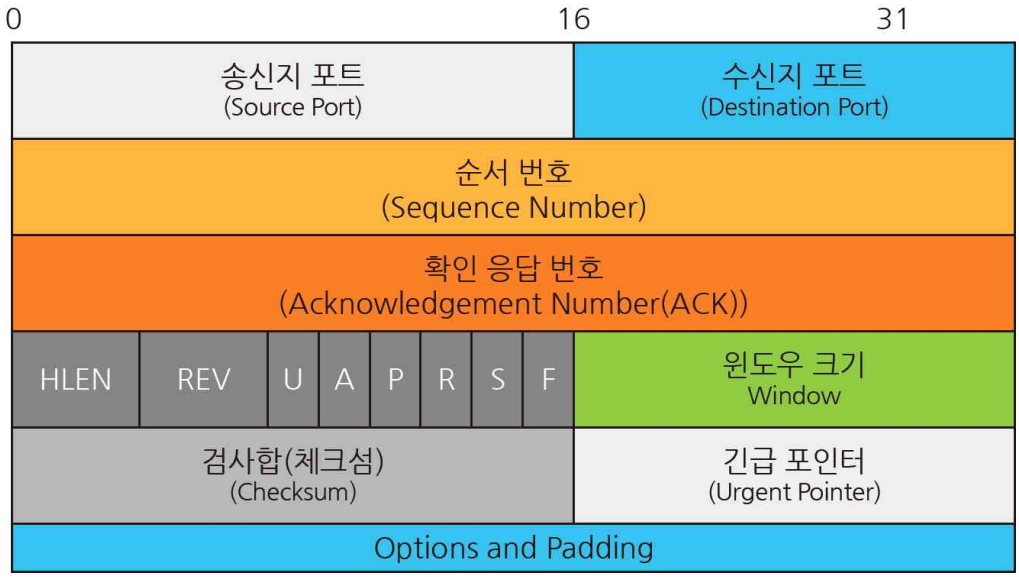
② IP 헤더

(2) TCP(Transmission Control Protocol)

① TCP의 특징

- 연결형 서비스를 지원하는 전송 계층 프로토콜
- 양 종단간 신뢰성 있는 데이터 전달과 흐름제어를 한다.

② TCP 헤더



Section 8. 라우팅 프로토콜

1. 라우팅 프로토콜

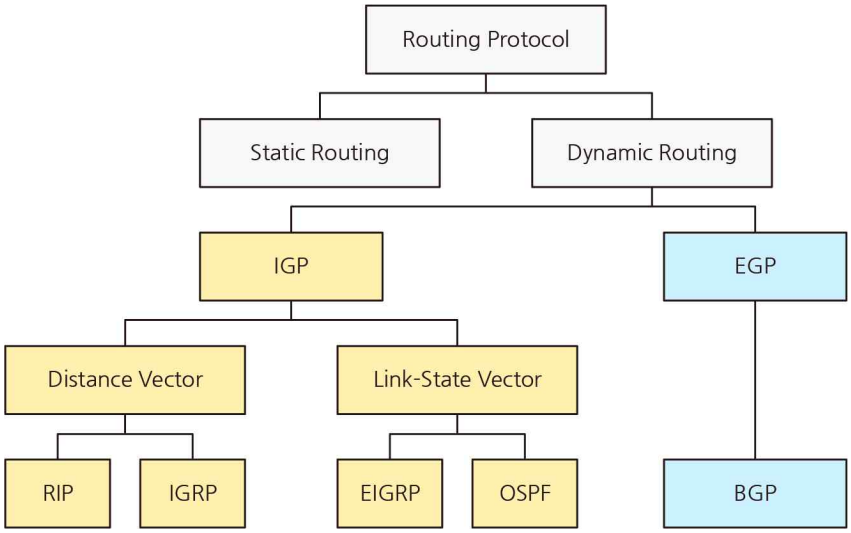
(1) 라우터(Router)

- Path Determination(경로설정)과 Switching(스위칭)을 하는 장비
- 데이터 패킷이 목적지까지 이동할 때 최적의 경로를 판단하는 장비

(2) 라우팅 프로토콜

- 패킷이 목적지까지 가는 방법을 결정해주는 프로토콜
- RIP, OSPF, IGRP, BGP 등이 있다.

2. 라우팅 프로토콜의 종류



3. 주요 라우팅 프로토콜

(1) RIP(Routing Information Protocol)

- 벨만 포드 거리벡터 알고리즘을 사용한 HOP 수 기반 라우팅 프로토콜
- 최대 15홉을 지원하며, 소규모망에 적합
- 30초마다 라우팅 테이블을 이웃 라우터들과 공유
- 네트워크 속도나 안정성을 고려하지 않고, HOP 수만을 고려하여 설계

(2) OSPF(Open Shortest Path First)

- 다익스트라 알고리즘기반 방식
- 최적 경로 선택을 위해 홉수, 대역폭, 지연시간 등을 고려
- 링크상태 변화시에만 라우팅정보전송

(3) BGP(Border Gateway Protocol)

- RIP나 OSPF 등의 라우팅 방식에 비해 규모가 큰 망을 지원할 수 있는 Path Vector기반 라우팅 프로토콜

16 정보보안

Section 1. SW개발 보안 설계

1. 정보보안

(1) 정보보안 개념

- 기업의 정보 및 정보 시스템에 대해서 허가되지 않은 접근, 변경, 삭제 등으로부터 보호하는 것

(2) 정보보안 요소

- 기밀성(Confidentiality)
  - 인가된 사용자만 정보 자산에 접근할 수 있도록 한다.
- 무결성(Integrity)
  - 적절한 권한을 가진 사용자가 인가된 방법으로만 정보를 변경할 수 있도록 접근 통제한다.
- 가용성(Availability)
  - 원하는 시점에 언제든 정보 자산에 접근이 가능하도록 한다.
- 인증(Authentication)
  - 접속한 사용자가 허가받은 사용자인지 확인하는 것
- 부인방지(Non-repudiation)
  - 정보를 보낸 사람이 나중에 정보를 보냈다는 것을 발뺌(부인)하지 못하도록 하는 것

(3) 인증제도

① ISMS(정보보호 관리체계 인증)

- 정보통신망의 안전성 확보를 위하여 수립하는 기술적, 물리적, 관리적 보호조치 등 종합적인 정보보호 관리체계에 대한 인증제도

② PIMS(개인정보보호 관리체계 인증)

- 기관 및 기업이 개인정보보호 관리체계를 갖추고 체계적 · 지속적으로 보호 업무를 수행하는지에 대해 객관적으로 심사하여 기준 만족 시 인증 부여

③ ISMS-P(정보보호 및 개인정보보호 관리체계 인증)

- 정보보호 및 개인정보보호를 위한 일련의 조치와 활동이 인증기준에 적합함을 인터넷진흥원 또는 인증기관이 증명하는 제도

2. Secure SDLC(Software Development Life Cycle)

⑴ Secure SDLC 의 개념

- 보안상 안전한 소프트웨어를 개발하기 위해 SDLC(Software Development Life Cycle)에 보안 강화를 위한 프로세스를 포함한 것

⑵ Secure SDLC 방법론

① CLASP(Comprehensive, Lightweight Application Security Process)

- SDLC의 초기 단계에서 보안을 강화하기 위해 개발된 방법론
- 현재 운용중인 시스템에 적용하기에 적합함

② MS-SDL

- MS사에서 안전한 소프트웨어 개발을 위해 기존의 SDLC를 개선한 방법론

③ Seven Touchpoints

- 소프트웨어 보안의 모범사례를 SDLC에 통합한 방법론

3. 시큐어 코딩(Secure Coding)

⑴ OWASP(The Open Web Application Security Project)

- 오픈소스 웹 애플리케이션 보안 프로젝트

⑵ 시큐어 코딩 가이드

① 입력 데이터 검증 및 표현

- 프로그램 입력값에 대한 검증 누락 또는 부적절한 검증, 데이터형식을 잘못 지정하여 발생하는 보안 약점

② 보안기능

- 보안 기능을 부적절하게 구현하는 경우 발생할 수 있는 보안 약점

③ 시간 및 상태

- 동시 수행을 지원하는 병렬 시스템이나 하나 이상의 프로세스가 동작하는 환경에서 시간 및 상태를 부적절하게 관리하여 발생할 수 있는 보안 약점

④ 에러 처리

- 에러를 처리하지 않거나 불충분하게 처리하여 에러 정보에 중요 정보가 포함될 때 발생할 수 있는 보안 약점

⑤ 코드 오류

- 개발자가 범할 수 있는 코딩 오류로 인해 유발되는 보안 약점

⑥ 캡슐화

- 중요한 데이터 또는 기능성을 불충분하게 캡슐화하거나 잘못 사용해 발생하는 보안 약점

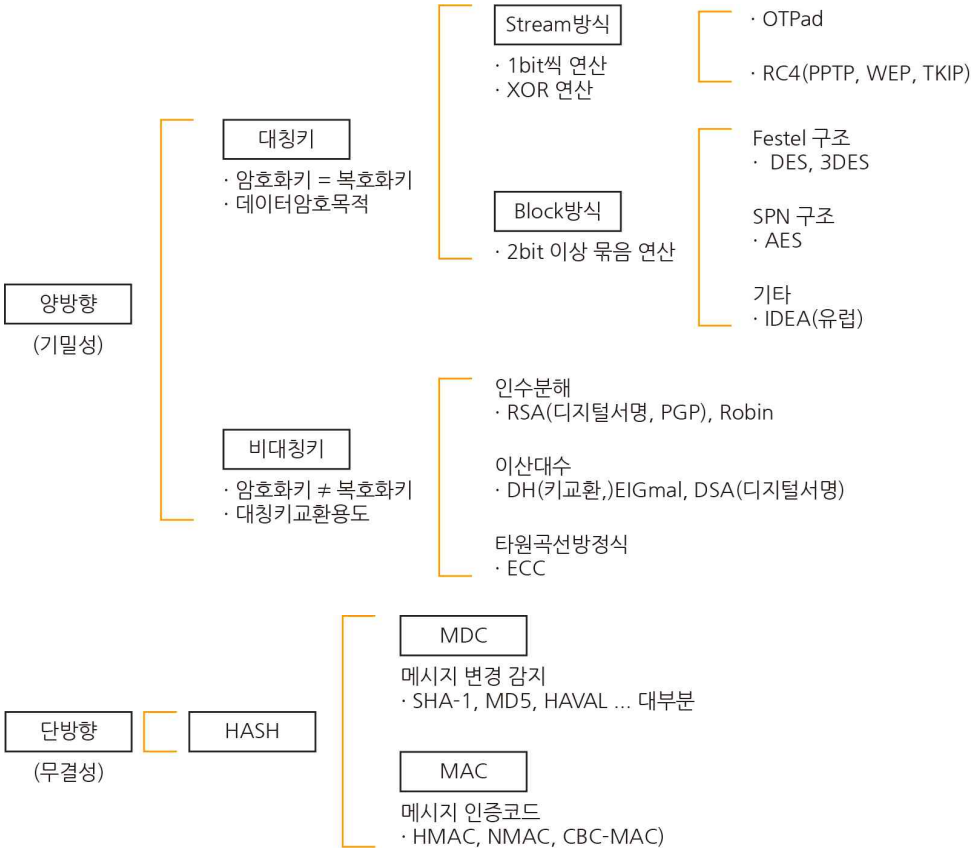
⑦ API 오용

- 의도된 사용에 반하는 방법으로 API를 사용하거나 보안에 취약한 API를 사용하여 발생할 수 있는 보안 약점

Section 2. SW개발 보안 구현

1. 암호 알고리즘

(1) 암호 방식에 따른 분류



⑵ 대칭키 암호(Symmetric key)

① 대칭키 암호 개념

- 암호화할 때의 키와 복호화할 때의 키가 동일한 암호 시스템

② 블록암호 알고리즘

알고리즘	설명
DES	- 64bit 블록, 56bit 암호화 키 사용 - Feistel 암호 방식을 사용한다.
3-DES	- 암호화 키 2개를 사용하여 암호화→복호화→암호화 순으로 암호화
AES	- 128bit 평문을 128/192/256bit로 암호화 - SPN 암호 방식을 사용한다.
SEED	- 순수 국내기술로 개발한 128비트 및 256비트 대칭 키 블록의 암호 알고리즘
ARIA	- 국가 보안 기술 연구소(NSRI) 필두로 학계, 국가 정보원 등의 암호 기술 전문가들이 개발한 국가 암호화 알고리즘 - AES 알고리즘과 똑같이 128/192/256비트 암호화키를 지원한다.
IDEA	- 1990년 스위스에서 만들어진 PES를 개량하여 만들어진 블록 암호 알고리즘 - 키길이가 128bit, 블로길이가 64bit - Feistel 방식과 SPN의 중간형태 구조

③ 스트림암호 알고리즘

알고리즘	설명
LFSR	- LFSR은 현재 상태에서 선형 연산을 통해 다음 상태를 생성하는 레지스터 - 스트림 암호의 난수를 생성하는 용도로 많이 사용한다. - 블록암호에 비해 경량 및 고속 동작이 용이하다.
RC4	- 로널드 라이베스트가 만들었다. - 각 단계에서 키스트림 한 바이트를 생성한다. - LFSR의 구조를 가지지 않으며 옥텟 단위를 기반으로 한다. - 비트 단위의 암호화 보다 실행속도가 빠르다.
A5	- 시프트 레지스터를 기반으로 사용 - GSM 휴대폰 체계에 사용

⑶ 비대칭키 암호

① 비대칭키 암호 개념

- 암호화와 복호화에 이용하는 키가 다른 방식
- 공개 키 암호 방식이라고도 한다.

② 비대칭키 알고리즘

구분		설명
소인수 분해 기반	RSA	- 대표적인 공개키 암호 알고리즘
	Robin	- 1979년 Robin이 개발, RSA보다 빠르다.
이산대수 기반	Diffie-Hellman	- 키관리 센터 없이 공개키 전달 가능
	DSA	- 미국의 전자서명 표준
	ELGamal	- 같은 평문에서 다른 암호문의 생성이 가능
타원 곡선	ECC	- 타원 곡선상의 이산대수를 이용

⑷ 단방향 암호화

① 단방향 암호화 개념

- Hash를 이용하여 암호화하는 과정
- 평문을 암호화 할 순 있지만, 복호화는 불가능하다.

② 해시 함수 특성

특성	설명
역상 저항성	- 해시 값이 주어졌을 때, 그 해시 값을 생성하는 입력값을 알아내기가 불가능하다는 특성
제 2역상 저항성	- 어떤 입력 값과 동일한 해시 값(결과 값)을 가지는 다른 입력 값을 찾을 수 없어야 한다는 특성
충돌 저항성	- 해시 값(결과 값)이 같은 두 개를 찾을 수 없다는 특성

③ 해시 함수 종류

종류	설명
MD5	- 128 비트 암호학 해시 함수이다. - 1991년에 MD4를 대체하기 위해 고안되었다. - 1996년에 암호화 결함이 발견되었고 2008년에 결함을 이용해 SSL 인증서를 변조하는 것이 가능하다고 발표되었다.
SHA	- 미국 국가안보국(NSA)가 설계 했으며 미국 국가 표준으로 지정되어있다. - SHA-0, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512

④ 암호학적 해시 함수의 결점

- 무차별 대입 공격(Brute-force attack)
- Rainbow table 공격

⑤ 암호학적 해시 함수의 보완

- 키 스트레칭(Key Stretching)
  - 해시 암호화를 여러 번 반복하여서 암호학적 문제가 발생하는 점을 줄일 수 있다.
  - 무차별 대입 공격 을 방지하는 효과가 있다.
- 솔팅(Salting)
  - 데이터 앞/뒤에 임의의 값을 넣어 해시값을 만든다.
  - Rainbow Table 공격을 방지하는 효과가 있다.

Section 3. 인증과 접근통제

1. 인증과 인가

⑴ 인증(Authentication)

① 인증의 개념

- 로그인을 요청한 사용자의 정보를 확인하고 접근 권한을 검증하는 보안 절차

② 인증 유형

- 지식 기반 인증
- 소유 기반 인증
- 생체기반 인증
- 행위기반 인증
- 위치기반 인증

⑵ 인가(Authorization)

- 로그인 후, 인증된 사용자에게 권한을 부여한다.
- 권한에 따라 사용 가능한 기능이 제한된다.

⑶ 인증방식

① 계정 정보를 요청 헤더에 넣는 방식

② Cookie/Session 방식

③ 토큰 기반 인증 방식(JSON Web Token, jwt)

④ SSO(Single Sign-On)

2. 접근 통제

⑴ 접근 통제 개념

- 정당한 사용자에게는 권한을 부여하고 그 외의 다른 사용자는 거부하는 것

⑵ 접근 통제 과정

① 식별(Identification)

- 사용자 ID를 확인하는 과정

② 인증(Authentication)

- 패스워드가 정확한지 확인

③ 인가(Authorization)

- 읽고, 쓰고, 실행시키는 권한을 부여

⑶ 접근 통제 정책

정책	MAC	DAC	RBAC
권한부여	시스템	데이터 소유자	중앙관리자
접근결정	보안등급(Label)	신분(Identity)	역할(Role)
정책변경	고정적(변경 어려움)	변경 용이	변경 용이
장점	안정적, 중앙 집중적	구현 용이, 유연함	관리 용이

⑷ 접근 통제 모델

① 벨-라파둘라 모델(BLP, Bell-LaPadula Confidentiality Model)

- 미 국방부 지원 모델로 기밀성을 강조한 모델이다.
- 정보가 높은 레벨에서 낮은 레벨로 퍼지는 것을 방지한다.
- No Read Up, No Write Down

② 비바 모델(Biba Integrity Model)

- 무결성을 위한 상업용 모델이다.
- 무결성의 3가지 목표 중 비인가자의 데이터 수정 방지 가능
- No Read Down, No Write Up

③ 클락-윌슨 모델(Clark-Wilson Integrity Model)

- 무결성 중심의 상업용 모델이다.

④ 만리장성 모델(Chinese Wall Model, Breswer-Nash Model)

- 충돌을 야기하는 어떠한 정보의 흐름도 차단해야 한다는 모델로 이익 충돌 회피를 위한 모델
- 금융 서비스 제공 회사가 이해 충돌의 발생을 막기 위한 내부 규칙

Section 4. 시스템 보안 구현

1. 취약점 분석

(1) 보안 취약점

- 정보시스템에 불법적인 사용자의 접근을 허용할 수 있는 위협

(2) 보안 취약점 점검 분류

① 관리적 관점

② 기술적 관점

③ 물리적 관점

2. 보안관제

(1) 보안관제 개념

- 24시간 정보자산을 지키기 위해 모니터링하고, 외부의 공격자가 전달하는 패킷을 관측한다.
- 실제 침해사고 시 CERT(Computer Emergency Response Team)팀이 대응함

(2) 통합로그 분석 장비

- ESM(Enterprise Security Management)
- SIM(Security Information and event Management)
- SOAR

3. 보안 운영체제(Secure-OS), 신뢰성 운영체제(Trusted OS)

(1) 보안 운영체제 개념

- 컴퓨터 운영체제 상에 내재된 보안상의 결함으로 인하여 발생할 수 있는 각종 해킹으로부터 시스템을 보호하기 위하여 기존의 운영체제 내에 보안 기능을 추가한 운영체제

(2) 보안 운영체제 목적

① 안정성

- 중단 없는 안정적인 서비스를 지원함

② 신뢰성

- 중요 정보의 안전한 보호를 통한 신뢰성 확보

③ 보안성

- 주요 서버에 대한 침입차단 및 통합 보안관리

4. 보안 솔루션

(1) 방화벽(firewall)

- 기업이나 조직 내부의 네트워크와 인터넷 간에 전송되는 정보를 선별하여, 수용/거부/수정하는 기능을 가진 침입차단 시스템

(2) 웹방화벽(Web Firewall)

- SQL 삽입공격, Cross-Site Scripting(XSS) 등의 웹기반 공격을 방어할 목적으로 만들어진 웹서버 특화 방화벽

(3) 침입탐지시스템(IDS; Intrusion Detection System)

- 컴퓨터 시스템의 비정상적인 사용, 오용, 남용 등을 "실시간"으로 탐지하는 시스템
- 침입탐지 방식에 따른 분류
  - 오용탐지 : 미리 입력해 둔 공격 패턴이 감지되면 이를 알려준다.
  - 이상탐지 : 평균적인 시스템의 상태를 기준으로 비정상적인 행위나 자원의 사용이 감지되면 알려준다.
- 침입탐지 대상에 따른 분류
  - 네트워크 기반 IDS(NIDS) : 네트워크 패킷을 분석하여 침입을 탐지한다.
  - 호스트 기반 IDS(HIDS) : 로그 분석과 프로세스 모니터링을 통한 침입을 탐지한다.

(4) 침입방지시스템(IPS; Intrusion Prevention System)

- 방화벽과 침입탐지시스템을 결합한 것
- 탐지 후 방화벽 가동

(5) 데이터유출방지(DLP; Data Leakage/Loss Prevention)

- 내부 정보의 외부 유출을 방지하기 위한 보안솔루션

(6) VPN(Virtual Private Network, 가상 사설 통신망)

- 인터넷 등 통신 사업자의 공중 네트워크에 암호화 기술을 이용하여 사용자가 마치 자신의 전용 회선을 사용하는 것처럼 해주는 보안솔루션

(7) NAC(Network Access Control)

- 네트워크에 접속하는 내부PC의 MAC주소(고유랜카드주소)를 IP관리 시스템에 등록한 후 일관된 보안관리 기능을 제공하는 보안솔루션
- 내부PC의 소프트웨어 사용현황을 관리하여 불법적인 소프트웨어 설치를 방지

(8) ESM(Enterprise Security Management)

- 다양한 장비에서 발생하는 로그 및 보안 이벤트(방화벽, IDS, IPS, 웹방화벽, VPN 등)를 통합관리 하는 보안 솔루션
- 종합적인 보완관리체계 수립

5. 방화벽(firewall)

⑴ 구현방식에 따른 유형

유형	설명
패킷 필터링 (Packet Filtering)	- 네트워크 계층과 전송 계층에서 동작한다. - IP주소, Port주소 등의 데이터를 바탕으로 방화벽 정책을 세워 패킷을 필터링 한다. - 다른 방화벽에 비해 속도가 빠르다.
애플리케이션 게이트웨이 (Application Gateway)	- 응용계층에서 동작한다. - 로그에서 다양한 정보를 얻어 여러 기능을 추가할 수 있다.
회선 게이트웨이 (Circuit Gateway)	- 응용계층과 세션 계층 사이에서 동작한다.
상태 기반 패킷 검사 (Stateful Packet Inspection)	- OSI 의 모든 계층에서 패킷을 분석하여 차단하는 기능 - 방화벽 중 가장 강력하다.
혼합형 타입 (Hybrid Type)	- 서비스 종류에 따라 복합적으로 구성한다.

⑵ 방화벽 시스템 구축 유형

① 스크리닝 라우터(Screening Router)

- IP, TCP, UDP의 헤더 부분에 포함된 내용만 분석하여 동작한다.
- 내부 네트워크와 외부 네트워크 사이의 패킷을 허용/거부하는 라우터이다.
- 비용이 적게 들지만, 패킷 내의 데이터는 차단 및 관리 어렵다.

② 베스천 호스트(Bastion Host)

- 내부 네트워크로 진입하기 전에 베스천 호스트를 두어 내부 네트워크를 전체적으로 보호한다.
- 접근 제어를 기본으로 프록시 기능을 사용하며, 인증, 로깅 등의 여러 작업을 수행한다.
- 스크린 라우터보다 안전하고, 로그 생성 관리가 용이하다.

③ 듀얼 홈드 호스트(Dual-Homed Host)

- 2개의 인터페이스를 가진 베스천 호스트로, 하나의 NIC은 내부 네트워크 연결, 다른 NIC는 외부 네트워크와 연결한다.

④ 스크린드 호스트(Screened Host)

- 패킷 필터 라우터와 베스천 호스트로 구성되어 있다.
- 패킷 필터 라우터는 내/외부 패킷을 통과시킬 것인지 결정한다.
- 베스천 호스트는 내/외부 네트워크 시스템에 대한 인증을 담당한다.

⑤ 스크린드 서브넷(Screened Subnet)

- 스크린드 호스트의 보안성 문제점을 해결한 것이다.
- 외부 네트워크와 내부 네트워크 사이에 하나 이상의 경계 네트워크를 두고, 내/외부 네트워크를 분리하기 위한 구조이다.
- 두 개의 스크리닝 라우터와 한 개의 베스천 호스트로 구성되어 있다.

6. 보안 프로토콜

(1) SSH(Secure Shell Protocol)

- 원격 호스트에 접속하기 위해 사용되는 보안 프로토콜
- 22번 포트를 사용한다.

(2) SSL(Secure Socket Layer)

- 웹 브라우저와 웹 서버 간에 데이터를 안전하게 주고 받기 위한 업계 표준 프로토콜
- SSL이 적용된 웹 페이지는 URL이 https 로 시작되며, 443번 포트를 사용한다.(http는 80포트)

(3) TLS(Transport Layer Security)

- 전송계층을 기반으로 개발 되었다.
- 데이터의 정보 보호와 무결성을 제공하기 위해 만들어졌다.

(4) IPSec

① IPSec 개념

- IP계층(네트워크 계층)을 안전하게 보호하기 위한 기법
- 패킷에 대한 보안을 제공한다.

② 동작모드

- 전송 모드(Transport Mode)
  - IP 헤더를 제외한 IP 패킷의 페이로드(Payload)만을 보호한다.
- 터널 모드(Tunneling Mode)
  - IP 패킷 전체를 보호한다.

③ 프로토콜

- AH(Authentication Header)
  - 메세지 인증 코드(MAC)를 이용하며 무결성(Data Integrity)과 인증(Authentication) 기능 제공
- ESP(Encapsulating Security Payload)
  - AH가 가진 무결성과, 인증도 제공하고 추가적으로 대칭키 암호화를 통해 기밀성(Confidentiality) 제공
- IKE(Internet key Exchange)
  - IPSec에서 키 교환에 사용되는 프로토콜

(5) S-HTTP (Secure HTTP)

- 웹상에서 네트워크 트래픽을 암호화하는 주요 방법 중 하나이다.
- 웹상의 파일들이 안전하게 교환될 수 있도록 해주는 HTTP의 확장판이다.

7. 스토리지

(1) 스토리지 개념

- 컴퓨터에 데이터를 저장하는 저장소의 역할을 수행하는 부품

(2) 스토리지 종류

① DAS(Direct Attachted Storage)

② NAS(Networt Attached Storage)

③ SAN(Storage Area Network)

(3) RAID(Redundant Array of Inexpensive Disks)

① RAID 개념

- 복수의 HDD를 하나의 드라이브와 같이 인식하고 표기한다.

② RAID 구성

- 스트라이핑(Striping) : 논리적으로 연속된 데이터들이 물리적으로 여러 개의 디스크에 라운드로빈 방식으로 저장되는 형태
- 미러링(Mirring) : 데이터를 그대로 복제하는 것으로 신뢰성 확보를 위해 사용됨

③ RAID 형태

- RAID-0
  - 빠른 데이터 입출력을 위해 스트라이핑을 사용하는 방식으로, 디스크의 모든 용량을 사용한다.
- RAID-1
  - 두 개 이상의 디스크를 미러링을 통해 하나의 디스크처럼 사용한다.
- RAID-2
  - 오류 정정을 위한 해밍코드를 사용하는 방식
- RAID-3
  - 하나의 디스크를 패리티(Parity) 정보를 위해 사용하고 나머지 디스크에 데이터를 균등하게 분산 저장
- RAID-4
  - RAID-3 과 같이 패리티 정보를 독립된 디스크에 저장한다.
  - 블록(Block)단위로 분산 저장하는 차이가 있다.
- RAID-5
  - 3개 이상의 디스크를 붙여서 하나의 디스크처럼 사용하고 각각의 디스크에 패리티 정보를 가지고 있는 방식
- RAID-6
  - 하나의 패리티를 두 개의 디스크에 분산 저장하는 방식

8. 고가용성(HA, High Availability)

- 서버와 네트워크, 프로그램 등의 정보 시스템이 오랜 기간 동안 지속적으로 정상 운영이 가능한 성질

Section 5. 서비스 공격 유형

1. DoS(Denial of Service) 공격

⑴ DoS 공격의 개념

- 대상 시스템이 정상적인 서비스를 할 수 없도록 가용성을 떨어뜨리는 공격
- 대상 시스템에 과도한 트래픽을 보내거나, 트래픽 발생을 유도함으로써 정상적인 운영이나 서비스가 될 수 없게 하는 공격

⑵ 공격 목표

- 물리적인 파괴 (디스크 및 시스템 파괴)
- 시스템 자원 공격 (CPU , Memory , Disk의 자원고갈)
- 네트워크 자원 공격 (대역폭 고갈)

⑶ DoS 공격 유형

① Smurf Attack

- 여러 호스트들로 하여, 특정 대상에게 다량의 ICMP Echo Request 를 보내게 하는 공격 방법
- IP와 ICMP 의 특성을 이용한다.
- 공격자는 IP 주소를 공격 서버의 IP 주소로 위장하고, ICMP Request 패킷을 브로드캐스트를 통해 다수의 시스템에 전송한다. 이 때 브로드캐스트를 수신한 다수의 시스템은 ICMP Echo Reply 패킷을 공격자가 아닌 공격 대상의 서버로 전송하게 되면서 부하를 발생시킨다.

② Ping Of Death

- 규정 크기 이상의 ICMP 패킷으로 시스템을 마비시키는 공격 방법
- ICMP 패킷을 정상적인 크기보다 크게 만들어 공격 대상에게 전송하면, 사이즈가 크기 때문에 패킷을 나눠서 보내게 된다. 공격대상은 쪼개진 패킷을 조립하는 과정에서 많은 부하가 발생하고, 재조합 버퍼의 오버플로우가 발생하여 정상적인 서비스가 불가능해진다.
- 반복적으로 들어오는 일정 수 이상의 ICMP 패킷을 무시하는 방법으로 대응한다.

③ Land Attack

- 출발지 IP와 목적지 IP가 같은 패킷을 만들어 보내는 공격 방법
- 수신자가 응답을 보낼 때, 목적지 주소가 자기 자신이므로 SYN 신호가 계속 자신의 서버를 돌게 되어 서버의 자원을 고갈 시켜 가용성을 파괴한다.
- 방화벽에서 출발지와 목적지가 같은 패킷은 모두 제거하여 대응한다.

④ Teardrop Attack

- 재조합을 할 수 있는 fragment number를 위조하는 공격 방법
- 데이터를 보낼 때, 데이터를 나누고, 재조립 할 수 있는 fragment number를 부여 하는데, fragment number를 위조하여 재조합이 안 되어 다운되게 하는 공격

⑤ SYN Flooding

- TCP의 연결과정(3Way Handshaking)의 취약점을 이용한 공격 방법
- 공격자가 SYN 신호만 전달하고, ACK 응답을 받지 않아, Backlog queue에 연결 정보를 계속 쌓게 하여 정상적인 서비스 제공이 불가능하게 만드는 공격

⑥ UDP Flooding

- 다량의 UDP 패킷을 전송하여 네트워크 자원을 고갈시키는 공격

⑦ Ping Flooding

- 특정 사이트에 매우 많은 ICMP Echo를 보내면, 이에 대한 응답(Respond)을 하기 위해 시스템 자원을 모두 사용해버려 시스템이 정상적으로 동작하지 못하도록 하는 공격 방법

2. DDoS(Distributed Denial of Service attack) 공격

⑴ DDoS 공격의 개념

- 특정 서버(컴퓨터)나 네트워크 장비를 대상으로 많은 데이터를 발생시켜 장애를 일으키는 대표적인 서비스 거부 공격
- 분산된 다수의 좀비 PC를 이용하여 공격 대상 시스템의 서비스를 마비시키는 공격 형태

⑵ DDoS 공격 구성

① 공격자(Attacker)

- 해커의 시스템

② 명령 제어(C&C:Command and Control)

- 공격자로부터 공격 명령을 전달 받는 시스템

③ 좀비(Zombie) PC

- C&C의 명령을 받고 실제 공격을 수행하는 다수의 PC

④ 공격 대상(Target)

- 좀비 PC의 공격을 받는 대상 시스템

⑶ DDoS 공격 툴의 종류

① Trinoo(트리누)

- Master/Agent 로 구성되어 있으며 Master의 명령으로 Agent가 작업을 수행하는 DDos 공격 도구
- UDP Flooding 공격을 수행한다.
- 목표 시스템에 대량의 UDP 패킷이 전송되어 시스템을 다운 시킨다.

② TFN(Tribal Flood Network)

- Master와 Agent의 통신에는 ICMP ECHO-REPLY 메시지를 사용한다

③ Stacheldraht(슈타첼드라트)

- Trinoo의 네크워크 구조와 TFN의 다양한 공격방법, Communication 상의 encryption 기능을 포함한 공격도구

3. 기타 해킹 기법

- 웜(Worm)
  - 네트워크를 통해 자신을 복제하고 전파할 수 있는 악성 프로그램
- 바이러스(Virus)
  - 파일, 부트, 메모리 영역에서 스스로를 복사하는 악성프로그램으로 파일 속에 숨어 옮겨 다닌다.
- 트로이목마(Trojan)
  - 겉으로 보기에는 전혀 해를 끼치지 않을 것처럼 보이고 자기 복제 능력이 없지만 실제로는 바이러스 등의 위험인자를 포함하고 있는 프로그램
- 혹스(Hoax)
  - 남을 속이거나 장난을 친다는 뜻으로, 말 그대로 가짜 바이러스를 말한다.
- 포트 스캐닝(Port Scanning)
  - 서버에 열려있는 포트를 확인 후 해당 포트의 취약점을 이용한 공격
  - Nmap을 이용해 열린포트, 호스트, 버전 등을 탐지할 수 있다.
- 스니핑 공격(Sniffing Attack)
  - 네트워크로 전송되는 패킷을 훔쳐보는 공격
- Smishing(SMS phishing)
  - 문자메시지를 이용한 피싱
- Qshing
  - QR코드를 통해 악성 링크로 접속을 유도하거나 직접 악성코드를 심는 금융범죄 기법
- 세션 하이재킹(Session Hijacking)
  - 이미 인증을 받아 세션을 생성, 유지하고 있는 연결을 빼앗는 공격
- IP Spoofing
  - 자신의 IP 주소를 속여서 접속하는 공격
- ARP Spoofing
  - ARP 프로토콜의 허점을 이용하여 자신의 MAC(Media Access Control) 주소를 다른 컴퓨터의 MAC인 것처럼 속이는 공격
- DNS Spoofing
  - DNS 서버로 보내는 질문을 가로채서 변조된 결과를 보내주는 것으로 일종의 중간자 공격
- SQL injection
  - 코드 인젝션의 한 기법으로 클라이언트의 입력값을 조작하여 서버의 데이터베이스를 공격할 수 있는 공격
- Rainbow Table
  - 해시함수(MD-5, SHA-1, SHA-2 등)를 사용하여 만들어낼 수 있는 값들을 대량으로 저장한 테이블
- Backdoor
  - 정상적인 인증 절차를 거치지 않고, 응용프로그램 및 시스템에 접근할 수 있도록 만든 프로그램
  - 탐지방법 : 현재 동작 중인 프로세스 및 열린 포트 확인, SetUID 파일 검사, 무결성 검사(tripwire), 로그분석 등
- Password Cracking
  - 시스템의 비밀번호를 각종 툴(프로그램)을 통해 알아내는 공격 기법

- Format String Attack
  - 문자열의 출력 포맷을 애매하게 설정할 때의 취약점을 포착하여, 메모리의 RET 위치에 악성코드 주소를 입력하여 공격하는 기법
- APT(Advanced Persistent Threat)
  - 지속적이고 지능적인 해킹 공격의 통칭
- CSRF(Cross-site requset forgery)
  - 사용자가 자신의 의지와는 무관하게 공격자가 의도한 행위를 특정 웹사이트에 요청하게 하는 해킹 공격
- XSS(Cross-Site Scripting)
  - 악의적인 사용자가 공격하려는 사이트에 스크립트를 넣는 기법
- Nucking
  - 특정 아이피에 대량의 패킷을 보내 인터넷 접속을 끊는 크래킹의 일종
- Buffer Overflow
  - 버그의 일종 또는 이를 이용한 공격 방법
  - 프로그램이 실행될 때 입력받는 값이 버퍼를 가득 채우다 못해 넘쳐흘러 버퍼 이후의 공간을 침범하는 현상
  - 방어기법 : 스택가드(Stackguard), 스택쉴드(Stack Shield), ASLR(Address Space Layout Randomization)
- 부채널 공격(side channel attack)
  - 암호 알고리즘을 대상으로 한 물리적 공격 기법
- Brute Force
  - 조합 가능한 모든 문자열을 하나씩 대입해 공격
- Dictionary Attack
  - 암호화되어 저장된 패스워드를 알아내기 위한 공격 방법 중 하나
- Key Logger Attack
  - 컴퓨터 사용자의 키보드 움직임을 탐지해 ID, 패스워드 등 개인의 중요한 정보를 몰래 빼가는 해킹 공격
- 스파이웨어(Spyware)
  - 스파이와 소프트웨어 합성어로, 광고나 마케팅을 목적으로 배포되는게 대부분이어서 애드웨어(Adware)라고도 한다.
- 랜섬웨어(Ransomware)
  - 컴퓨터 시스템을 감염시켜 접근을 제한하고 일종의 몸값을 요구하는 악성 소프트웨어의 한 종류이다.
- 제로데이 공격(Zero-Day Attack)
  - 컴퓨터 소프트웨어의 취약점(exploit)을 공격하는 기술적 위협으로, 해당 취약점에 대한 패치가 나오지 않은 시점에서 이루어지는 공격
- 사회공학(social engineering)
  - 기술적인 방법이 아닌 사람들 간의 기본적인 신뢰를 기반으로 사람을 속여 비밀정보를 획득하는 기법
- Evil Twin Attack
  - 와이파이(WiFi) 무선 네트워크에서 공격자가 가짜 AP(Access Point)를 구축하고 강한 신호를 보내어 사용자가 가짜 AP에 접속하게 함으로써 사용자 정보를 중간에서 가로채는 기법
- Bluebug
  - 블루투스 장비간 취약한 연결관리를 이용한 공격
  - 한번 연결되면 이후에는 다시 연결해주지 않아도 자동으로 연결되는 인증 취약점 이용

- BlueSnarf
  - 블루투스 취약점을 이용하여 장비의 임의의 파일에 접근하는 공격
  - 인증없이 정보를 교환하는 OPP 기능을 사용하여 파일에 접근
- BluePrinting
  - 블루투스 공격장치의 검색 활동
- Switch Jamming
  - 위조된 매체 접근 제어(MAC) 주소를 지속적으로 네트워크로 흘려보내, 스위치 MAC 주소 테이블의 저장 기능을 혼란시켜 더미 허브(Dummy Hub)처럼 작동하게 하는 공격
- Honeypot
  - 비정상적인 접근의 탐지를 위해 의도적으로 설치해 둔 시스템
  - 침입자를 속여 실제 공격을 당하는 것처럼 보여줌으로써 크래커를 추적 및 공격기법의 정보를 수집하는 역할을 한다.
  - 쉽게 공격자에게 노출되어야 하며 쉽게 공격이 가능한 것처럼 취약해 보여야 한다.

17 신기술 용어

Section 1. S/W 개발 동향

1. 중앙 집중식 인프라, 클라우드 서비스

(1) 클라우드 서비스 개념

- 인터넷 기반의 컴퓨팅

(2) 서비스 제공 형태

- 퍼블릭 클라우드(Public Cloud, 공공 클라우드, 개방형 클라우드)
  - 인터넷에 접속 가능한 모든 사용자를 위한 클라우드 서비스 모델
- 프라이빗 클라우드(Private Cloud, 사설 클라우드, 폐쇄 클라우드)
  - 제한된 네트워크 상에서 특정 기업이나 특정 사용자만을 대상으로 하는 클라우드
- 하이브리드 클라우드(Hybrid Cloud)
  - 퍼블릭 클라우드와 프라이빗 클라우드를 병행해 사용하는 방식

(3) 서비스 유형

종류	설명
IaaS (Infrastructure as a Service)	클라우드로 IT 인프라 자원을 제공하는 서비스
PaaS (Platform as a Service)	사용자가 소프트웨어를 개발할 수 있는 클라우드 컴퓨팅 플랫폼
SaaS (Software as a Service)	사용자가 필요로 하는 소프트웨어를 인터넷상에서 이용하는 클라우드 서비스
BaaS (Blockchain as a Service)	블록체인 기술 응용 서비스 개발과 관리를 클라우드 기반으로 더욱 편리하게 지원하는 서비스

(4) 도커(Docker)

- 컨테이너 응용프로그램의 배포를 자동화하는 오픈소스 엔진

(5) 하이퍼바이저(컴퓨팅 가상화 솔루션)

- 하나의 호스트 컴퓨터상에서 동시에 다수의 운영체제를 구동시킬 수 있는 하드웨어와 운영체제 사이의 소프트웨어 가상화 플랫폼

2. 소프트웨어 정의 기술(Software-Defined Everything, SDx)

(1) 소프트웨어 정의 기술 개념

- 다양한 소프트웨어 정의(Software-Defined) 관련 기술을 하나로 통칭하여 부르는 용어

(2) 종류

종류	설명
SDN (Software-Defined Networking)	- 소프트웨어 정의 네트워킹 - 네트워킹 리소스를 가상화된 시스템으로 추상화하는 IT 인프라에 대한 하나의 접근 방식 - 비용 절감, 우수한 확장성 및 유연성, 관리 간소화
SDS (Software-Defined Storage)	- 소프트웨어 정의 스토리지 - 하드웨어에서 스토리지 소프트웨어를 분리하는 스토리지 아키텍처 - 자동화, 표준 인터페이스, 가상화된 데이터 경로, 확장성, 투명성
SDC (Software-Defined Computing)	- 소프트웨어 정의 컴퓨팅
SDDC (Software-Defined Data Center)	- 소프트웨어 정의 데이터 센터 - 데이터센터의 모든 인프라인 네트워크, 스토리지, 컴퓨터, 보안 등이 가상화 되어 서비스로서 제공되는 차세대의 핵심적인 데이터센터 솔루션 - 인력 개입 없이 소프트웨어 조작만으로 자동 제어 관리한다.

3. 양자컴퓨터(quantum computer)

- 양자역학에서 양자얽힘, 중첩, 텔레포테이션 등의 효과를 이용해 계산하는 컴퓨터
- 기존 컴퓨터가 0과 1만 구분할 수 있는 반면, 양자 컴퓨터는 0과 1을 동시에 공존 시킬 수 있다.
- 현존 최고의 슈퍼 컴퓨터가 수백 년이 걸려도 풀기 힘든 문제도 단 몇 초 이내의 어마어마한 속도로 빠르게 풀 수 있을 것으로 전망된다.
- 양자 컴퓨터에서 자료의 양은 큐비트로 측정된다.

4. 블록체인(Blockchain)

- 분산 컴퓨팅 기술 기반의 데이터 위변조 방지 기술
- '블록'은 개인과 개인의 거래(P2P)의 데이터가 기록되는 장부
- '블록'들을 형성된 후 시간의 흐름에 따라 순차적으로 연결된 체인의 구조를 가진다.
- 공공 거래장부 또는 분산 거래장부라고도 한다.
- 비트코인 시스템을 개발하면서 발생하는 문제를 블록체인을 개발, 적용함으로써 해결함

5. 인공지능(Artificial Intelligence)

⑴ 인공지능 개념

- 인간이 지닌 지적 능력의 일부 또는 전체, 혹은 그렇게 생각되는 능력을 인공적으로 구현한 것

⑵ 인공지능 분야의 기술

- 기계 학습(Machine Learning)
  - 규칙을 일일이 프로그래밍하지 않아도 자동으로 데이터에서 규칙을 학습하는 알고리즘을 연구하는 분야
- 인공 신경망
  - 인간의 뉴런 구조를 본떠 만든 기계 학습 모델
- 딥 러닝(Deap Learning)
  - 머신러닝 알고리즘 중에 인공 신경망을 기반으로 한 방법들을 통칭
  - 인공 신경망이라고도 하며, 텐서플로와 파이토치가 대표적인 라이브러리이다.

⑶ 인공지능 키워드

- 사이킷런(scikit-learn)
  - 2007년 구글 썸머 코드에서 처음 구현되었으며, 가장 널리 사용되는 머신러닝 패키지 중 하나
- 텐서플로(TensorFlow)
  - 구글이 만든 딥러닝 라이브러리
  - CPU와 GPU를 사용해 인공신경망 모델을 효율적으로 훈련하며 모델 구축과 서비스에 필요한 다양한 도구를 제공
- 파이토치(Pytorch)
  - Facebook에서 개발하여 2016년 공개한 파이썬 기반의 오픈소스 머신러닝 라이브러리
- 케라스(Keras)
  - 2015년에 공개된 파이썬 기반의 오픈소스 신경망 라이브러리

6. 신속한 애플리케이션 개발

- No Code
  - 코드를 사용하지 않고 애플리케이션을 개발하는 것
  - 코드를 전혀 모르는 사람들을 위해 뭔가를 쉽게 만들게 해주는 도구들
- Low-Code
  - 필요한 부품을 간단한 명령으로 조합하여 시스템을 만드는 개발 방법
  - 디자이너, 제품담당자, 창업자, 엔지니어들이 (약간의 기술지식을 가지고) 빠르고 편하게 생산성을 향상

7. 클라이언트 측 웹 프레임워크

(1) React

- 페이스북에서 개발

(2) Vue.js

- Evan You에 의해 개발

(3) AngularJS

- 구글에서 개발

(4) Ajax(Asynchronous JavaScript and XML)

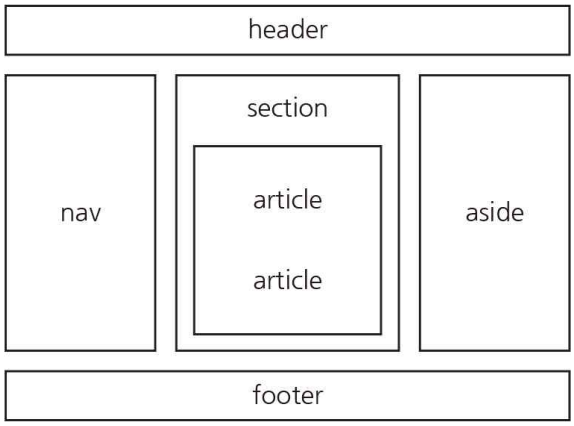
- 비동기적인 웹 애플리케이션의 제작을 위한 웹 개발 기법

8. 시맨틱 웹(semantic web)

(1) 시맨틱 웹 개념

- 컴퓨터가 이해할 수 있는 방식으로 인터넷 웹사이트를 제작함으로써 인터넷에 존재하는 다양한 정보를 컴퓨터가 쉽게 이해하고 해석할 수 있도록 하는 것

(2) 시맨틱 웹을 위한 HTML 태그(구조)



종류	설명
header	- 헤더
nav	- 네비게이션
aside	- 사이드에 위치하며 부차적인 내용을 담는 태그
section	- 본문의 여러 내용(article)을 포함하는 공간
article	- 본문의 주 내용이 들어가는 공간
footer	- 푸터

9. 온톨로지(Ontology)

(1) 온톨로지 개념

- 사람들이 세상에 대해 느끼고 생각하며 합의한 바를 컴퓨터에서 다룰 수 있는 형태로 표현한 모델

(2) 구성

- 클래스(Class)
- 인스턴스(Instance)
- 속성(Property)
- 관계(Relation)

10. 매시업(Mashup)

- 웹으로 제공하고 있는 정보와 서비스를 융합하여 새로운 소프트웨어나 서비스, 데이터베이스 등을 만드는 것
- 기존의 자원을 활용하여 만들기 때문에 서비스 구축을 위한 비용이 적게 든다.
- 구글 지도에 부동산 매물 정보를 결합한 서비스인 구글의 하우징 맵스 등으로 활용된다.

11. 디지털 트윈(Digital Twin)

- 물리적 자산, 시스템 또는 프로세스를 소프트웨어로 표현하는 것

12. 메타버스(Metaverse)

- 가상을 의미하는 meta와 세계를 의미하는 universe의 합성어

Section 2. 네트워크 / 데이터베이스 신기술 용어

1. 네트워크 신기술 용어

- IoT(Internet of Things), 사물 인터넷
  - 실세계와 가상세계의 다양한 사물들을 연결하여 진보된 서비스를 제공하기 위한 서비스 기반 시설
  - 센싱 기술, 유무선 통신 및 네트워크 인프라 기술, 사물 인터넷 인터페이스 기술, 사물 인터넷을 통한 서비스 기술 등이 있다.
- M2M(Machine to Machine), 사물 통신
  - 기계와 기계 사이의 통신
  - 기계, 센서, 컴퓨터 등 다양한 장치들이 유무선 통신 기술을 이용해 서로 정보를 교환한다.
- BLE(Bluetooth Low Energy), 저전력 블루투스
  - 2.4 GHz 주파수 대역 기반의 저전력 저용량 데이터 송수신이 가능한 블루투스 기술
  - 대부분의 시간은 슬립 모드(sleep mode)로 있어 전력 소모가 매우 적다.
  - 시계나 장난감, 비컨(beacon), 그리고 착용 컴퓨터(웨어러블 기기) 등에 많이 사용된다.
- NFC(Near Field Communication), 근접 무선 통신
  - 13.56 MHz 주파수를 이용한 자기 유도결합 기반의 근거리 자기장 통신 기술
  - 수 cm 이내의 거리에서 개인 간 통신(P2P: Peer-To-Peer)을 지원하는 전파 식별(RFID) 기술의 일종
- RFID(Radio Frequency IDentification), 전파 식별
  - 전파 신호를 통해 비접촉식으로 사물에 부착된 얇은 평면 형태의 태그를 식별하여 정보를 처리하는 시스템
- ZigBee, 지그비
  - 저속, 저비용, 저전력의 무선망을 위한 기술
- 지능형 초연결망
  - 네트워크 전체에 소프트웨어 정의 기술(SDE)을 적용하는 차세대 국가망
- Ad-hoc Network, 애드 혹 네트워크
  - 노드(node)들에 의해 자율적으로 구성되는 기반 구조가 없는 네트워크
  - 응용 분야로는 긴급 구조, 긴급 회의, 전쟁터에서의 군사 네트워크 등이 있다.

- Mesh Network
  - 기존 무선 랜의 한계 극복을 위해 등장
  - 대규모 디바이스의 네트워크 생성에 최적화되어 차세대 이동통신, 홈네트워킹, 공공 안전 등의 특수목적을 위한 새로운 방식의 네트워크 기술
- Mobile Computing, 이동 컴퓨팅
  - 휴대형 기기로 이동하면서 자유로이 네트워크에 접속하여 업무를 처리할 수 있는 환경
- 지능형 전력망
  - 전력 에너지의 효율성을 증대하고 전력 품질을 고도화하고, 전력시스템의 안정성을 제공하기 위하여 기존의 전력 시스템에 IT 기술을 부가하는 새로운 개념의 IT 융합 기술
- Smart Grid, 스마트 그리드
  - 전기 및 정보통신기술을 활용하여 전력망을 지능화, 고도화함으로써 고품질의 전력서비스를 제공하고 에너지 이용효율을 극대화하는 전력망
- Wi-Sun
  - 스마트 그리드 서비스를 제공하기 위한 와이파이 기반의 저전력 장거리 통신기술
- NDN(Named Data Networking), 근접 무선 통신
  - 인터넷에서 콘텐츠 자체의 정보와 라우터 기능만을 이용하여 목적지로 데이터를 전송하는 기술
- Piconet
  - 여러 개의 독립된 통신 장치가 블루투스 기술이나 초광대역 무선(UWB) 통신기술을 사용하여 통신망을 형성하는 무선 네트워크 기술
  - 무선랜(WLAN)과 달리 전송을 위한 기반 구조가 미리 설정되지 않고 상황에 따라 기기들 간에 조정 프로토콜에 의하여 네트워크를 형성한다.
- UWB(Ultra-WideBand technology), 초 광대역 기술
  - 기존의 스펙트럼에 비해 매우 넓은 대역에 걸쳐 낮은 전력으로 대용량의 정보를 전송하는 무선통신 기술
  - 대역폭이 500 MHz 이상이거나 비대역폭(FBW: Fractional Band Width)이 20% 이상인 신호를 이용한 근거리 무선 기술
- SON(Self-Organizing Network), 자동 구성 네트워크
  - 주변 상황에 자동적으로 적응하여 스스로 망을 구성하는 네트워크
- GIS(Geographical Information System), 지리 정보 시스템
  - 지도에 관한 속성 정보를 컴퓨터를 이용해서 해석하는 시스템
- USN(Ubiquitous Sensor Network), 유비쿼터스 센서 네트워크
  - 각종 센서에서 감지한 정보를 무선으로 수집할 수 있도록 구성한 네트워크
- WPAN(Wireless Personal Area Network), 무선 사설망
  - 무선을 이용하는 개인영역네트워크
- WDM(Wavelength Division Multiplexer), 파장 분할 다중화기
  - 하나의 광섬유 채널을 빛의 파장에 의해서 분할하여 복수의 통신로로 사용할 수 있게 하는 장치
- VPN(Virtual Private Network), 가상 사설 통신망
  - 인터넷과 같은 공중망에 사설망을 구축하여 마치 전용망을 사용하는 효과를 가지는 보안 솔루션

- MQTT(Message Queuing Telemetry Transport)
  - TCP/IP 기반 네트워크에서 동작하는 발행-구독 기반의 메시징 프로토콜
  - 사물통신, 사물인터넷과 같이 대역폭이 제한된 통신환경에 최적화하여 개발된 푸시기술 기반의 경량 메시지 전송 프로토콜
- N-Screen
  - 동일한 콘텐츠를 N 개의 이종 단말기에서 자유롭게 이용할 수 있는 서비스
  - 하나의 콘텐츠를 TV나 PC, 태블릿 PC, 스마트 폰 등 다양한 기기에서도 끊김 없이 이용할 수 있게 해주는 서비스
- Vlan(Virtual Local Area Network)
  - 물리적 배치와 상관없이 논리적으로 LAN을 구성할 수 있는 기술
  - 네트워크 리소스 보안을 높이고, 비용을 절감할 수 있다.

2. 데이터베이스 신기술 용어

- 빅데이터(Big Data)
  - 디지털 환경에서 발생하는 대량의 모든 데이터
  - 기존 데이터베이스 관리 도구의 능력을 넘어서 데이터에서 가치를 추출하고 결과를 분석하는 기술
  - 빅데이터의 특징 : 규모(Volume), 다양성(Variety), 속도(Velocity), 정확성(Veracity)과 가치(Value)
- 정형 데이터(Structured data)
  - 미리 정해 놓은 형식과 구조에 따라 저장되도록 구성된 데이터
- 비정형 데이터(Unstructured data)
  - 정의된 구조가 없이 정형화되지 않은 데이터
- 메타 데이터(Meta Data)
  - 데이터에 대한 데이터
  - 데이터에 관한 구조화된 데이터로, 다른 데이터를 설명해 주는 데이터
- 데이터웨어하우스(Data Warehouse)
  - 기간시스템의 데이터베이스에 축적된 데이터를 공통의 형식으로 변환해서 관리하는 데이터베이스
  - 정보에 입각한 의사 결정을 내릴 수 있도록 분석 가능한 정보의 중앙 리포지토리
- 데이터 마트(Data Mart)
  - 데이터의 한 부분으로서 특정 사용자가 관심을 갖는 데이터들을 담은 비교적 작은 규모의 데이터 웨어하우스
- 데이터 마이닝(Data Mining)
  - 대규모로 저장된 데이터 안에서 체계적이고 자동적으로 통계적 규칙이나 패턴을 찾아내는 것
  - 수많은 데이터에서 가치있는 유용한 정보를 찾아내는 것
- 텍스트 마이닝(Text Mining)
  - 자연어로 구성된 비정형 텍스트 데이터에서 패턴 또는 관계를 추출하여 의미 있는 정보를 찾아내는 것
- 하둡(High-availability distributed object-oriented platform, Hadoop)
  - 다수의 범용 컴퓨터를 연결하여 하나의 시스템처럼 작동하도록 묶어 대용량의 다양한 데이터들을 분산 처리하는 공개 소스 프레임워크

- Sqoop(SQL to Hadoop)
  - 관계형 데이터베이스(RDB)와 분산 파일 시스템(HDFS) 사이의 양방향 데이터 전송을 위해 설계된 툴
- 맵리듀스(MapReduce)
  - 분산 컴퓨팅에서 대용량 데이터를 병렬 처리하기 위해 개발된 소프트웨어 프레임워크
  - 구글이 수집한 문서와 로그 등 방대한 데이터들을 분석하기 위해 2004년에 발표한 소프트웨어 프레임워크
  - 방대한 입력 데이터를 분할하여 여러 머신들이 분산 처리하는 맵(Map) 함수 단계와 이를 다시 하나의 결과로 합치는 리듀스(Reduce) 함수 단계로 나뉜다.
- 타조(Tajo)
  - 아파치 하둡(Apache Hadoop) 기반의 분산 데이터 웨어하우스 프로젝트
- R
  - 양이 많은 정보(데이터)를 통계적 방법으로 분석할 때 사용된다.
- OLAP(On-Line Analytical Processing)
  - 이용자가 직접 데이터베이스를 검색, 분석해서 문제점이나 해결책을 찾는 분석형 애플리케이션 개념이다.
- NoSQL(Not only SQL)
  - 기존 관계형 DBMS가 갖고 있는 특성뿐만 아니라, 다른 특성들을 부가적으로 지원