

2020 개정판

정보처리기사 핵심 요약집

-

김광섭 강사

Part 1.
소프트웨어 설계

| Chapter 01. 요구사항 확인

01. 현행 시스템 분석

* 핵심 정리

1) 현행 시스템 파악

- 응용소프트웨어 엔지니어링의 현행 시스템 파악 절차 및 세부 시스템의 구성요소를 도출
- 구성요소
 - 현행 시스템 아키텍처 구성도
 - 소프트웨어 구성도
 - 하드웨어 구성도
 - 네트워크 구성도

2) 개발기술 환경 정의

- 기술개발 환경에 대한 정의 및 기술 요소별 특징 및 고려 사항을 인지
- 기술 환경 요소
 - 운영체제
 - DBMS : 사용자, 다른 애플리케이션, 데이터베이스와 상호 작용하여 데이터를 저장하고 분석하기 위한 컴퓨터 소프트웨어
 - 미들웨어
 - 오픈 소스 소프트웨어

02. 요구사항 확인하기

* 핵심정리

1) 요구사항 정의

- 요구사항에 대한 개념을 정확히 파악하고 이를 기반으로 고객의 요구사항을 체계적으로 정립
- 요구사항 개발 프로세스
 - 요구사항 도출(Elicitation)
 - 분석(Analysis)
 - 명세(Specification)
 - 확인(Validation)

2) 요구사항의 시스템화 타당성 분석

- 요구사항이 시스템으로 구축되었을 때 타당성이 있는지 사전에 파악하는 방안
- 요구사항의 기술적 타당성 검토요소
 - 성능 및 용량산정의 적정성
 - 시스템 간 상호 운용성
 - IT 시장 성숙도 및 트렌드 부합성
 - 기술적 위험 분석

03. 분석모델 확인하기

* 핵심정리

1) 분석모델 검증

- 분석모델링 검증은 도출된 요구사항에 대하여 분석모델을 확인하고 이에 대한 추가적인 필요 의견을 제시하는 단계
- 분석 모델 검증
 - 유스케이스 모델 검증
 - 개념 수준의 분석 클래스 검증
 - 분석 클래스 검증

2) 분석모델의 시스템화 타당성 분석

- 유스케이스 모델의 개별 유스케이스에 대한 분석모델을 작성한 이후, 해당 분석모델로 시스템을 개발하는 경우에 어떠한 영향을 미치는지 성능 및 용량 등 필요한 자원 적정성, 상호 운용성, 시장 성숙도, 기술적 위험 분석 측면에서 타당성 조사
- 분석 모델의 시스템으로서의 구축 타당성 여부를 최종적으로 검토하는 단계

| Chapter 02. 화면설계

01. UI 요구사항 확인하기

* 용어 정리

1) 소프트웨어 아키텍처

소프트웨어 개발을 쉽게 하도록 기본 틀을 만드는 것이며 복잡한 개발을 체계적으로 하기 위한 밑그림

2) UI(User Interface)

사용자와 컴퓨터 상호 간의 소통을 원활히 하게 도와주는 연계 작업

3) UI 스토리보드

UX구현에 수반되는 사용자와 컴퓨터, 인터페이스간 상호작용을 시각화한 문서

* 핵심 정리

1) 소프트웨어 아키텍처

- 소프트웨어 개발을 쉽게 하도록 기본 틀을 만드는 것이며 복잡한 개발을 체계적으로 하기 위한 밑그림
- ISO/IEC 9126 모델 소프트웨어 품질 특성
 - 기능성
 - 신뢰성
 - 사용성
 - 효율성
 - 유지 보수성

2) UI 표준과 지침

- UI(User Interface)는 사용자와 컴퓨터 상호 간의 소통을 원활히 하게 도와주는 연계 작업을 뜻함
- UI 설계 원칙
 - 직관성, 유효성, 학습성, 유연성
- UI 설계 지침
 - 사용자 중심, 일관성, 단순성, 결과 예측 가능, 가시성,
 - 표준화, 접근성, 명확성, 오류 발생 해결

3) 스토리 보드

- UX구현에 수반되는 사용자와 컴퓨터, 인터페이스간 상호작용을 시각화하여, 디자이너와 개발자의 의사소통을 돕는 도구
- 스토리보드 작성 순서
 - 1단계 메뉴 구성도 만들기
 - 2단계 스타일 확정
 - 3단계 설계하기

02. UI 설계하기

* 용어 정리

1) UI 설계서

UI 설계서는 표지, 개정 이력, UI 요구사항 정의, 시스템 구조, 프로세스 정의, 화면 설계 등으로 구성

2) 프로토타입(Prototype)

새로운 소프트웨어의 설계 또는 성능, 구현 및 운용 가능성을 평가하거나 요구 사항을 좀 더 잘 이해하기 위해 전체적인 기능을 간략한 형태로 구현한 시제품

3) 감성공학(Human Sensibility Ergonomics)

개인의 경험을 통해 얻어지는 복합적인 감성을 과학적 측면으로 측정하고 분석하여 제품설계에 최대한 반영하는 공학기술

* 핵심정리

1) UI 흐름 설계

UI 설계서는 표지, 개정 이력, UI 요구사항 정의, 시스템 구조, 사이트맵, 프로세스 정의, 화면 설계 등으로 구성

유용한 시스템은 사용자 모형과 개발자 모형 간의 실행 차와 평가 차를 최소화

2) UI 상세 설계

UI 시나리오는 사용자가 경험하게 될 이야기를 미리 그려 봄으로써 서비스 전체의 윤곽을 표현한 것으로 웹(앱)을 사용하는 모든 단계마다 일어나는 사용자와 서비스의 상호작용을 기술

UI 프로토타입은 확정된 요구사항을 기반으로 UI전략을 실체화하는 과정이며, UI 디자인 작성 이전에 미리 화면을 설계하는 단계

3) 감성공학

개인의 경험을 통해 얻어지는 복합적인 감성을 과학적 측면으로 측정하고 분석하여 제품설계에 최대한 반영하는 공학기술

감성공학기술에는 크게 생체측정, 오감센서 및 감성처리기술, 감성 디자인 기술, 마이크로 가공 기술, 인간에 대한 적합성을 판단하고 새로운 감성을 창출하기위한 기술

| Chapter 03. 애플리케이션 설계

01. 공통 모듈 설계하기

* 용어 정리

1) 공통 모듈

시스템을 구축 시 여러 서브 시스템에서 공통으로 사용되는 모듈을 모아 놓은 소프트웨어 묶음으로서 소프트웨어의 중복 구현을 줄여주고 재사용성을 높이기 위한 소프트웨어

2) 응집도

인터페이스의 요청을 처리함에 있어서 공통 모듈 내의 클래스들 간에 얼마나 유기적으로 협업하여 처리하는가에 관한 정도

3) 결합도

어떤 모듈이 다른 모듈에 의존하는 정도

* 핵심 정리

1) 공통 모듈이란?

시스템을 구축 시 여러 하위 시스템에서 공통으로 사용되는 모듈을 모아 놓은 소프트웨어 묶음으로서 소프트웨어 중복 구현을 줄여주고 재사용성을 높이기 위한 소프트웨어

▪ 공통 모듈의 종류

- 공통 모듈 클래스
- 공통 모듈 라이브러리
- 공통 모듈 컴포넌트
- 공통 모듈 프레임워크

2) 공통 모듈 도출

하향식 기법 : 개발 프레임워크에서 공통적인 모듈을 도출

상향식 기법 : 기능 목록을 기준으로 여러 부서에서 공통으로 사용되는 모듈 도출

3) 공통 모듈 설계

요구사항 분석 : 유스케이스 다이어그램

정적 분석 : 클래스 다이어그램

동적 분석 : 시퀀스 다이어그램

4) 공통 모듈 평가

- 응집도(Cohesion)
 - 공통 모듈 내부의 클래스 간의 유기적인 관계를 나타냄
 - 높을수록 품질이 좋음
- 결합도(Coupling)
 - 공통 모듈 간의 연관관계를 나타냄
 - 낮을수록 품질이 좋음

02. 객체지향 설계하기

* 용어 정리

1) 객체지향 방법

실 세계의 개체(Entity)를 속성과 메서드가 결합된 형태의 객체(Object)로 보고 구현대상을 객체와 객체들 간의 관계로 모델링하는 방법

2) 디자인 패턴

많은 개발자들이 경험으로 체득한 설계 지식을 검증하고 이를 추상화하여 일반화한 템플릿이며 디자인 패턴을 사용하면 효율성과 재사용성을 높일 수 있다.

* 핵심 정리

1) 객체지향(OOP)

실 세계의 개체(Entity)를 속성과 메서드가 결합된 형태의 객체(Object)로 보고 구현대상을 객체와 객체들 간의 관계로 모델링하는 방법

2) 객체지향의 특징

추상화, 상속, 다형성, 캡슐화, 정보은닉

3) 클래스 간 관계

- 클래스는 한 개 이상의 다른 클래스와 관계를 가진다.
- 클래스들 간 관계의 종류
 - 연관, 일반화-특수화
 - 집합연관, 복합연관

| Chapter 04. 인터페이스 설계

01. 인터페이스 요구사항 확인하기

* 용어 정리

1) 시스템 인터페이스

서로 독립적인 내외부 시스템이 연동을 통해 상호 작용하기 위한 접속 방법이나 규칙을 의미

2) 요구 사항 검증(Verification)

요구 사항 명세서에 사용자의 요구가 올바르게 기술되었는지에 대해 검토하고 베이스라인(baseline, 기준선)으로 설정하는 활동

* 핵심 정리

1) 인터페이스 요구 사항 분석

시스템 인터페이스 요구 사항은 인터페이스 이름, 연계 대상 시스템, 연계 범위 및 내용, 연계 방식, 송신 데이터, 인터페이스 주기, 기타 고려 사항을 명시한 것으로 내·외부 인터페이스 대상 시스템 및 기관과 시스템 연동 방안을 사전에 협의

2) 인터페이스 요구 사항 검증

인터페이스 요구사항 검토 방법

- 동료 검토(Peer Review)
- 워크 스루(Walk Through)
- 인스펙션(Inspection)
- CASE 도구 활용

02. 인터페이스 대상 식별하기

* 용어 정리

1) 시스템 아키텍처

시스템 전체(하드웨어와 소프트웨어를 포괄한 것)에 대한 논리적인 기능 체계와 그것을 실현하기 위한 구성 방식. 시스템의 전체적인 최적화를 목표로 한다.

* 핵심 정리

1) 인터페이스 시스템 식별

시스템 아키텍처는 시스템 전체(하드웨어와 소프트웨어를 포괄한 것)에 대한 논리적인 기능 체계와 그것을 실현하기 위한 구성 방식. 시스템의 전체적인 최적화를 목표로 한다.

2) 인터페이스 시스템 구성

- 송신 시스템
- 수신 시스템
- 중계 서버

3) 송수신 데이터 식별

- 송수신 전문에서 전문 공통부는 인터페이스 표준 항목을 포함하고, 전문 개별부는 송수신 시스템에서 업무 처리에 필요한 데이터를 포함하며, 전문 종료부는 전송 데이터의 끝을 표시하는 문자를 포함한다.
- 인터페이스 설계 데이터베이스 산출물
 - 개체 정의서
 - 테이블 정의서
 - 코드 정의서

03. 인터페이스 상세 설계하기

* 핵심 정리

1) 인터페이스 방법 명세화

개발하고자하는 응용소프트웨어의 내부와 외부 인터페이스를 위한 송수신 방법과 필요한 데이터, 오류시 처리방안을 명세화

2) 인터페이스 설계서 작성

인터페이스 설계서는 인터페이스 목록과 인터페이스 정의서로 구성

PART 2.
소프트웨어 개발

| Chapter 05. 데이터 입출력 구현

01. 논리 데이터저장소 확인

* 용어 정리

1) 데이터 모델링

기업의 정보 구조를 실체(Entity)와 관계(Relation)를 중심으로 명확하고 체계적으로 표현하여 문서화하는 기법

2) 논리 데이터 모델링

개념 모델로부터 업무 영역의 업무 데이터 및 규칙을 구체적으로 표현한 모델

3) 정규화(Normalization)

이상 현상을 제거하고 중복성을 최소화하면서 정보의 일관성을 보장하기 위해 데이터베이스를 설계해 가는 과정

* 핵심 정리

1) 자료구조

컴퓨터에서 자료를 효율적으로 표현하고 저장하고 처리할 수 있도록 만들어진 논리적인 틀

자료구조의 분류

- 선형구조 : 리스트, 스택, 큐, 데크
- 비선형구조 : 트리, 그래프

2) 논리 데이터모델 개요

논리 데이터 모델링은 정확한 업무 분석을 통한 자료의 흐름을 분석하여 현재 사용 중인 양식, 문서, 장표를 중심으로 자료항목을 추출하고 추출된 엔터티와 속성들의 관계(Relation)를 구조적으로 정의하는 단계

정규화 : 릴레이션을 관련 있는 속성들로만 구성되도록 릴레이션(테이블)을 분해하는 과정

- 제1정규화 : 중복속성제거
- 제2정규화 : 부분종속성제거
- 제3정규화 : 이행종속성제거

3) 논리 데이터모델 검증

- 논리 데이터저장소 확인절차
 - 엔터티 및 속성 확인
 - 관계 확인
 - 데이터 흐름 확인
 - 데이터 접근 권한 확인
 - 데이터 백업정책 및 분산구조 확인

02. 물리 데이터저장소 설계

* 용어 정리

- **반정규화**
: 정규화에 충실하여 모델링을 수행하면 종속성, 활용성은 향상되나 수행속도가 저하되는 경우가 발생하여 이를 극복하기 위해 성능에 중점을 두어 정규화 하는 방법
- **인덱스**
: 어떤 종류의 검색 연산을 최적화하기 위해 데이터베이스상에 Row들의 정보를 구성하는 데이터 구조로 인덱스를 이용하면 전체 데이터를 검색하지 않고 데이터베이스에서 원하는 정보를 빠르게 검색 가능
- **파티션**
: 성능, 가용성, 유지보수 등을 목적으로 테이블 또는 인덱스를 파티션 단위로 나누어 저장하는 기법

* 핵심 정리

1) 물리 데이터모델 설계

논리 데이터 모델로부터 물리 데이터 모델로 변환하는 것은 단위 엔터티를 테이블로, 속성을 컬럼으로 UID를 기본키로, 관계를 외래키로 변환 후 컬럼 유형과 길이 정의

데이터 처리 범위와 빈도 수를 분석하여 반정규화 고려

반정규화는 중복테이블을 추가하거나 테이블 조합, 분할, 제거 하는 방법 등을 통해 설계

2) 물리 데이터저장소 구성

물리 데이터 모델링 완료 후 모델링 결과에 따라 디스크라는 물리 데이터 저장소에 다양한 오브젝트 구성

테이블, 인덱스, 뷰, 클러스터, 파티션 등

오브젝트는 디스크 구성 설계를 통해 구성

- 테이블 제약 조건 : 참조 무결성 관리
- 인덱스 : 빠른 검색 속도를 위해 설계
- 뷰 : 가상 테이블을 설계하여 사용성 높임
- 클러스터 : 하나 혹은 그 이상의 테이블 설계
- 파티션 : 대용량 DB에서 성능 저하를 막고 관리를 용이하게 하기 위해 설계

3) ORM 프레임워크

객체 관계 매핑(ORM, Object-relationship mapping)은 관계형 데이터베이스와 객체지향 프로그래밍 언어 간의 호환되지 않는 데이터를 변환하는 프로그래밍 기법

ORM 절차

- 클래스를 테이블로 변환
- 속성(Attribute)은 컬럼(Column)으로 변환
- 클래스간 관계는 관계형 테이블 간의 관계로 변환

4) 트랜잭션 인터페이스

물리 데이터 저장소에서 트랜잭션 인터페이스는 전체적인 데이터베이스 트랜잭션의 골격 및 인터페이스를 정의하는 활동

데이터베이스 트랜잭션 특징

- 원자성(Atomicity)
- 일관성(Consistency)
- 독립성(Isolation)
- 영속성(Durability)

03. 데이터 조작 프로시저 작성

* 용어 정리

1) 데이터 정의어(DDL : Data Definition Language)

데이터를 저장하는 테이블 등의 구조를 생성하고 변경하기 위해 사용하는 명령어들
CREATE, DROP, RENAME, ALTER, TRUNCATE 등

2) 데이터 조작어(DML : Data Manipulation Language)

데이터베이스에 있는 데이터를 변경하거나 검색하기 위하여 사용되는 명령어들
INSERT, UPDATE, DELETE, SELECT 등

3) 데이터 제어어(DCL : Data Control Language)

사용자별로 데이터베이스에 접근할 수 있는 권한을 부여하거나 회수하는 명령어들
ROLE, GRANT, REVOKE 등

* 핵심 정리

1) 데이터 조작 프로시저 개발

SQL은 관계형 데이터베이스관리시스템(RDBMS)의 데이터를 관리하기 위해 설계된 특수 목적의 프로그래밍 언어로 데이터정의어, 데이터조작어, 데이터제어어를 가진다.

PL/SQL은 Oracle에서 개발한 데이터 조작 언어이며, 프로그래밍 언어의 특성을 수용한 SQL의 확장 기능이 우수함

PL/SQL로 작성할 수 있는 저장형 객체로는 Trigger, Stored Function, Stored Procedure, Stored Package 등이 있다.

2) 데이터 조작 프로시저 테스트

Oracle DBMS는 모든 데이터 조작 프로시저에 대한 테스트 환경으로 SQL*Plus 도구를 제공함.

04. 데이터 조작 프로시저 최적화

* 용어 정리

1) 튜닝(Tuning, 성능 개선)

시스템이 제공하는 한정된 자원을 이용해서 시스템의 성능을 최적화 시키는 작업

2) 옵티마이저(Optimizer)

개발자가 작성한 SQL을 가장 빠르고 효율적으로

수행할 최적의 처리 경로를 생성해 주는 데이터베이스 핵심 프로세스로 가장 빠르게 실행되는 방법을 비용(Cost)으로 계산하여 선택하는 CBO(Cost Based Optimizer, 비용 기반)와 미리 정해져 있는 규칙에 따라 가장 빠른 실행 방법을 결정하는 RBO(Rule Based Optimizer, 룰 기반)모드가 있음

3) 파서(Parser)

SQL 문의 구문 구조와 의미를 분석하는 것으로 SQL 문이 올바르게 작성되었는지 검사하는 구문 구조 분석(Syntax analysis)과 데이터베이스 객체나 참조된

객체들이 올바른지 검사하는 의미 분석(Semantic analysis)이 있음

* 핵심 정리

1) 데이터 조작 프로시저 성능 개선

모니터링 결과 문제시 되는 내용 확인시 SQL 성능개선 순서

- 문제 있는 SQL 식별
- 옵티마이저 통계 확인
- 실행 계획 검토
- SQL문을 재구성하거나 인덱스 재구성

2) 소스코드 인스펙션

데이터베이스 성능 향상을 위해 프로시저 코드를 보면서 성능 문제점을 개선해 나가는 활동

SQL 코드 인스펙션 대상

- 사용하지 않는 변수
- 사용되지 않는 서브쿼리
- Null 값과 비교하는 프로시저 소스
- 과거의 데이터타입 사용

| Chapter 06. 통합 구현

01. 모듈 구현

* 용어 정리

1) 단위모듈

통합 구현에서 단위 모듈 구현은 비즈니스 컴포넌트, 내외부 인터페이스 모듈, 데이터베이스 접근 모듈 등 통합 구현이 필요한 단위 컴포넌트를 뜻함

2) 단위모듈 테스트

컴퓨터 프로그래밍에서 소스 코드의 특정 모듈이 의도된 대로 정확히 작동하는지 검증하는 절차

* 핵심 정리

1) 단위모듈 구현

통합 구현에서 단위 모듈 구현은 비즈니스 컴포넌트, 내외부 인터페이스 모듈, 데이터베이스 접근 모듈 등 통합 구현이 필요한 단위 컴포넌트 구현

단위모듈 구현시 고려사항

- 응집도는 높이고, 결합도는 낮춤
- 공통모듈을 먼저 구현
- 항상 예외처리 로직을 고려하여 구현

2) 단위모듈 테스트

컴퓨터 프로그래밍에서 소스 코드의 특정 모듈이 의도된 대로 정확히 작동하는지 검증하는 절차

단위모듈 테스트 방법

- 화이트박스 테스트
- 메소드 기반 테스트
- 화면 기반 테스트
- 스텝과 드라이버 활용 테스트

02. 통합 구현 관리

* 용어 정리

형상관리(Configuration Management)

소프트웨어 개발 과정에서 소프트웨어의 변경 사항을 관리하기 위해 변경을 체계적으로 관리, 추적 및 제어하기 위한 일련의 활동

* 핵심 정리

1) IDE 도구

통합 개발 환경(IDE)은 코드의 작성 및 편집, 디버깅, 컴파일, 배포 등 프로그램 개발에 관련된 모든 작업을 하나의 프로그램 안에서 처리하는 환경을 제공하는 소프트웨어

IDE 도구의 기능

- 개발 환경 지원

- 컴파일 및 디버깅 기능 제공
- 외부 연계모듈과 통합 기능 제공

2) 협업 도구

협업 도구는 기능에 따라 문서공유, 소스공유, 아이디어 공유, 디자인 공유, 마인트 맵핑, 프로젝트 관리, 일정관리 등이 있음

협업도구 기능

- 개발자간 커뮤니케이션
- 일정 및 이슈 공유
- 개발자간 집단지성 활용

3) 형상관리 도구

형상관리는 소프트웨어 개발 과정에서 소프트웨어의 변경 사항을 관리하기 위해 변경을 체계적으로 관리, 추적 및 제어하기 위한 일련의 활동

형상관리절차는 형상 식별, 변경 제어, 형상 감사, 형상 기록 수행

소프트웨어 형상관리 도구는 CVS, SVN, Git

| Chapter 07. 제품소프트웨어 패키징

01. 제품소프트웨어 패키징

* 용어 정리

1) 모듈(Module)

소프트웨어 설계에서 기능단위로 분해하고 추상화 되어 재사용 및 공유 가능한 수준으로 만들어진 단위

2) 모듈화

소프트웨어의 성능을 향상시키거나 시스템의 디버깅, 시험, 통합 및 수정을 용이하도록 하는 소프트웨어 설계 기법

3) 소프트웨어 빌드(Software Build)

소스 코드 파일을 컴퓨터에서 실행할 수 있는 제품
소프트웨어의 단위로 변환하는 과정

* 핵심 정리

1) 애플리케이션 패키징

애플리케이션 패키징은 개발이 완료된 소프트웨어를 고객에게 전달하기 위한 형태로 패키징하며, 설치와 사용에 필요한 매뉴얼 작성과 소프트웨어 패치개발과 업그레이드를 위한 버전관리도 포함됨
사용자 관점의 애플리케이션 패키징 고려사항과 애플리케이션 패키징 순서 이해
애플리케이션의 변경 정보가 포함된 릴리즈 노트 개념과 작성항목 이해

2) 애플리케이션 배포 도구

- 애플리케이션 패키징 도구는 애플리케이션을 배포하기 위한 패키징 시디지털 콘텐츠의 지적 재산권을 보호하고 관리하는 기능을 제공하며, 안전한 유통과 배포를 보장하는 도구이자 솔루션
- 패키징 도구 활용시 고려사항 이해
- 저작권 보호 측면의 패키징 도구인 DRM(Digital Rights Management) 개념과 특징
- 저작권 관리의 흐름과 구성 요소 이해

3) 애플리케이션 모니터링 도구

애플리케이션 모니터링 도구는 제품 소프트웨어를 사용자 환경에 설치 후 기능 및 성능 운영 현황을 파악하는 도구

애플리케이션 변경관리, 성능관리, 정적분석, 동적분석 도구 등이 있음

02. 제품소프트웨어 매뉴얼 작성

* 핵심 정리

1) 제품소프트웨어 매뉴얼 작성

제품소프트웨어 매뉴얼은 설치 매뉴얼과 사용자 매뉴얼이 있음

설치 및 사용자 매뉴얼은 개발자의 기준이 아닌 사용자의 기준으로 작성

설치 및 사용자 매뉴얼의 작성 단계와 작성 항목에 대한 이해

2) 국제 표준 제품 품질 특성

제품 품질 표준과 프로세스 품질 표준으로 나뉨

소프트웨어 제품 품질 관련 국제 표준

- ISO/IEC 9126, ISO/IEC 14598, ISO/IEC 12119, ISO/IEC 25000

소프트웨어 프로세스 품질 관련 국제 표준

- ISO/IEC 9000, ISO/IEC 12207, ISO/IEC 15504, ISO/IEC 15288, CMMi

03. 제품소프트웨어 버전관리

* 용어 정리

1) 제품 소프트웨어 버전 관리

SW 개발과 관련하여 코드와 라이브러리, 관련 문서 등 시간의 변화에 따른 변경을 관리하는 전체 활동을 의미

* 핵심 정리

1) 소프트웨어 버전 관리 도구

- 형상관리지침을 활용하여 제품 소프트웨어의 신규 개발, 변경, 개선과 관련된 수정 내역을 관리하는 도구
- 소프트웨어 버전관리 도구 유형
 - 공유 폴더 방식 : RCS, SCCS
 - 클라이언트/서버 방식 : CVS, SVN

- 분산 저장소 방식 : Git, Bitkeeper

2) 빌드 자동화 도구

제품 소프트웨어 실행 파일 생성을 자동화하기 위해서 저장소에 있는 소스를 자동으로 읽어서 빌드를 하여 실행 파일을 만드는 도구

온라인 빌드 자동화 도구, 젠킨스(Jenkins) 이해

안드로이드 환경에 적합한 도구, 그라들(Gradle) 이해

| Chapter 08. 애플리케이션 테스트 관리

01. 애플리케이션 테스트케이스 설계

* 용어 정리

1) 테스트 케이스

테스트 설계 산출물로 특정한 프로그램의 일부분 또는 경로에 따라 수행하거나, 특정한 요구사항을 준수 하는지 확인하기 위해 설계된 입력 값, 실행 조건, 기대 결과로 구성된 테스트 항목의 명세서

2) 테스트 시나리오

테스트 수행을 위한 여러 테스트 케이스의 집합으로, 테스트 케이스의 동작 순서를 기술한 문서이며 테스트를 위한 절차를 명세한 문서

* 핵심 정리

1) 애플리케이션 테스트 케이스 작성

- 테스트에 대한 기본 개념 및 테스트의 기법을 파악하고 테스트 케이스 작성 기법
- 소프트웨어 테스트 기법 분류에서는 소프트웨어 내부 구조 참조 여부에 따른 분류, 테스트 설계의 근원에 따른 분류
- 테스트 케이스 작성 방법

2) 애플리케이션 테스트 시나리오 작성

- 테스트 시나리오 개념 및 작성시 유의사항
- 테스트 환경구축의 개념 및 유의사항
- 소프트웨어 테스트 관련 국제표준(ISO/IEC 29119)에 의한 테스트 지식 체계

02. 애플리케이션 통합 테스트

* 용어 정리

1) 에러, 오류

결함(Defect)의 원인이며, 사람에 의하여 생성된 실수가 대부분임

2) 결함, 결점, 버그

에러가 원인이 되어 제품에 포함된 결함

제품이 일으키게 되는 실패(Failure) 또는 문제의 원인

3) 실패, 문제

제품의 결함이 있는 부분이 실행될 때 발생하는 현상

* 핵심 정리

1) 애플리케이션 통합 테스트 수행

- 통합 테스트에 대한 개념 이해, 통합 테스트 수행 방법
- 테스트 자동화 도구와 개념과 유형

2) 애플리케이션 테스트 결과 분석

- 소프트웨어 결함관련 용어, 테스트 완료 조건
- 테스트 결함 관리 개념과 결함관리 도구

03. 애플리케이션 성능 개선

* 핵심 정리

1) 알고리즘

알고리즘은 수학과 컴퓨터 과학, 언어학 또는 관련 분야에서 어떠한 문제를 해결하기 위해 정해진 일련의 절차나 방법을 공식화한 형태로 표현한 것

정렬 알고리즘 : 선택 정렬, 삽입 정렬, 버블 정렬

그래프 탐색 : 깊이우선탐색(DFS), 너비우선탐색(BFS)

탐색 알고리즘 : 선형 탐색, 이진 탐색

2) 애플리케이션 성능 개선

- 소스코드 최적화는 읽기 쉽고 변경 및 추가가 쉬운 클린 코드를 작성하는 것으로, 소스 코드 품질을 위해 기본적으로 지킬 원칙과 기준을 정의
- 클린코드의 작성원칙과 소스코드 최적화 기법
- 소스코드 품질분석 도구의 이해
- 정적분석 : Pmd, cppcheck, SonarQube, checkstyle
- 동적분석 : Avalanche, valgrind

| Chapter 09. 인터페이스 구현

01. 인터페이스 설계 확인

* 용어 정리

1) EAI(Enterprise Application Integration)

기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션들 간의 정보 전달, 연계, 통합을 가능하게 해 주는 솔루션, 연계 서버

2) ESB(Enterprise Service Bus)

애플리케이션보다는 서비스 중심으로 통합을 지향하는 아키텍처 또는 기술

* 핵심 정리

1) 인터페이스 기능 확인

인터페이스 기능 확인 방법

- 인터페이스 설계서를 통한 인터페이스 기능 확인
- 정적동적 모형을 통한 인터페이스 기능 확인
- 데이터 명세 정의를 통한 인터페이스 기능 확인

내부, 외부 모듈 연계 방법 : EAI, ESB

2) 데이터 표준 확인

인터페이스 데이터 표준은 인터페이스를 위해 인터페이스가 되어야 할 범위의 데이터들의 형식과 표준을 정의하는 것

02. 인터페이스 기능 구현

* 용어 정리

1) 스니핑(Sniffing)

‘쿵쿵 냄새를 맡다’ 라는 뜻으로 네트워크 주변을 지나다니는 패킷을 엿보면서 계정(ID)과 패스워드를 알아내기 위한 행위

2) 스푸핑(Spoofing)

‘속여먹다’ 라는 뜻을 지닌 ‘spoof’에서 나온 말로 해커가 악용하고자 하는 호스트의 IP 주소나 e-메일 주소를 바꾸어서 이를 통해 해킹을 하는 것

* 핵심 정리

1) 인터페이스 보안

인터페이스 보안 취약점에 대한 시큐어 코딩 항목

인터페이스 보안 기능 적용

- 네트워크 구간 보안 기능 적용
- 애플리케이션에 보안 기능 적용
- 데이터베이스에 보안 기능 적용

데이터베이스 암호화 알고리즘

- 대칭키, 해시, 비대칭키

2) 소프트웨어 연계 테스트

내외부 연계 모듈 구현에서 소프트웨어 연계 테스트는 송신 시스템과 수신 시스템 간에 연계 테스트를 의미함

소프트웨어 연계 단위 및 통합 테스트

소프트웨어 연계 단위 및 통합 테스트 케이스 작성

03. 인터페이스 구현 검증

* 핵심 정리

1) 인터페이스 구현 검증

- 인터페이스 구현 검증 도구, 감시도구

도구	설명
xUnit	java(Junit), C++(Cppunit), .Net(Nunit) 등 다양한 언어를 지원하는 단위 테스트 프레임워크
STAF	서비스 호출, 컴포넌트 재사용 등 다양한 환경을 지원하는 테스트 프레임워크
FitNesse	웹 기반 테스트 케이스 설계/실행/결과 확인 등을 지원하는 테스트 프레임워크
NTAF	Naver 테스트 자동화 프레임워크이며, STAF와 FitNesse를 통합
Selenium	다양한 브라우저 지원 및 개발언어를 지원하는 웹 애플리케이션 테스트 프레임워크
watir	Ruby 기반 웹 애플리케이션 테스트 프레임워크

- 인터페이스 구현 검증에 필요한 설계 산출물

2) 인터페이스 오류 처리 확인 및 보고서 작성

- 인터페이스 오류 처리 방법
 - 사용자 화면에서 오류를 발생
 - 인터페이스 오류 로그 생성
 - 인터페이스 관련 테이블에 오류 사항 기록

Part 3.
데이터베이스 구축

| Chapter 10.SQL 응용

01. 절차형 SQL 작성

* 핵심 정리

1) 사용자 정의함수

절차형 SQL을 활용하여 일련의 연산 처리 결과를 단일값으로 반환할 수 있는 SQL 사용자 정의함수의 호출을 통해 실행되며, 반환되는 단일값을 조회 또는 삽입, 수정 작업에 이용

2) 트리거

- 특정 테이블에 삽입, 수정, 삭제 등의 데이터 변경 이벤트가 발생하면 DBMS에서 자동적으로 실행 되도록 구현된 프로그램
- 이벤트는 전체 트랜잭션 대상과 각 행에 의해 발생하는 경우 모두를 포함할 수 있으며 테이블과 뷰(View), DB 작업을 대상으로 정의
- 데이터 제어어(DCL, Data Control Language) 사용 불가

3) 이벤트

- 특정 시간에 특정한 쿼리, 프로시저, 함수 등을 실행시키는 기능

02. 응용 SQL 작성

* 용어 정리

1) 데이터 조작어(DML : Data Manipulation Language)

데이터베이스에 있는 데이터를 변경하거나 검색하기 위하여 사용되는 명령어들
INSERT, UPDATE, DELETE, SELECT 등

2) 데이터 제어어(DCL : Data Control Language)

사용자별로 데이터베이스에 접근할 수 있는 권한을 부여하거나 회수하는 명령어
GRANT, REVOKE, COMMIT, ROLLBACK 등

3) 데이터 분석 함수

데이터 튜플 간의 상호 연관 및 계산 분석을 위한 함수
집계 함수(AGGREGATE FUNCTION), 그룹 함수(GROUP FUNCTION), 윈도우 함수(WINDOW FUNCTION)

* 핵심 정리

1) 집계성 SQL 작성

집계성 SQL 작성은 단일 행을 기반으로 산출하지 않고 복수 행을 그룹별로 모아 놓고 그룹당 단일 계산 결과를 반환

GROUP BY 구문을 활용하여 복수 행을 그룹핑
SELECT, HAVING, ORDER BY 등의 구문에 활용

데이터 분석 함수 종류

- 집계 함수 : COUNT, SUM, AVG, MAX, MIN 등
- 그룹 함수 : ROLLUP, CUBE, GROUPING SETS
- 윈도우 함수 : 순위함수, 그룹 내 비율 함수, 행순서 함수

2) 오류 처리

프로그램 코드 상의 구문 오류 또는 프로그램 실행시 상황에 따라 발생하는 오류를 처리하는 과정

오류 처리 방법

- 오류 복구
- 오류 회피
- 오류 전환

| Chapter 11. SQL 활용

01. 기본 SQL 작성

* 용어 정리

1) 데이터 정의어(DDL : Data Definition Language)

데이터를 저장하는 테이블 등의 구조를 생성하고 변경하기 위해 사용하는 명령어
CREATE, DROP, RENAME, ALTER, TRUNCATE 등

2) 트랜잭션(Transaction)

일련의 연산 집합으로 데이터베이스의 상태를 변환시키기 위하여 논리적 기능을 수행하는 하나의 작업 단위

트랜잭션의 특징

- 원자성(Atomicity)
- 일관성(Consistency)
- 고립성(Isolation)
- 영속성(Durability)

* 핵심 정리

1) 관계형 데이터 모델

- 2차원 테이블을 구성하여 테이블 내에 있는 속성들 간의 관계를 설정하거나 테이블 간의 관계를 정의하는 DB구조
- 관계형 데이터 모델은 데이터를 개체테이블과 관계테이블로 표현
- E-R 모델은 데이터를 개체, 관계, 속성으로 묘사

2) 테이블

- 데이터를 저장하는 객체(Object)로서 관계형 데이터베이스 기본 단위
- 관계형 데이터베이스에서는 모든 데이터를 열과 행의 2차원 구조로 나타내며 세로 방향을 컬럼(Column), 가로 방향을 행(Row)이라고 하고 컬럼과 행이 겹치는 하나의 공간을 필드(Field)라고 함

3) 데이터 사전

- 데이터 사전(Data Dictionary)에는 데이터베이스의 데이터를 제외한 모든 정보가 있음
- 오라클에서 데이터 사전 검색
- MySQL에서 데이터 사전 검색

02. 고급 SQL 작성

* 핵심 정리

1) 뷰(View)

하나 이상의 물리 테이블로부터 생성 가능한 가상 테이블

- 뷰의 장점
 - 논리적 독립성 제공, 사용자 데이터 관리 용이,
 - 데이터 보안 용이
- 뷰의 단점
 - 뷰 자체 인덱스 불가, 뷰 정의 변경 불가,
 - 데이터 변경 제약 존재

2) 인덱스(Index)

인덱스는 데이터를 빠르게 찾을 수 있는 수단으로서, 테이블에 대한 조회 속도를 높여 주는 자료구조

3) 다중 테이블 검색

관계형 데이터베이스에서 데이터를 통합하는 기법으로 다중 테이블에 대한 검색이 사용됨

- 다중 테이블 검색 방법
 - 조인 : 두 개의 테이블을 결합하여 데이터를 추출하는 기법
 - 서브쿼리 : SQL문 안에 포함된 SQL문 형태의 사용 기법
 - 집합연산 : 테이블을 집합 개념으로 조작하는 기법

| Chapter 12. 논리 데이터베이스 설계

01. 관계데이터베이스 모델

* 용어 정리

1) 관계 데이터 모델

실 세계 데이터를 행과 열과 구성된 표(테이블, 릴레이션) 형태로 저장하고 한 테이블의 필드 값을 이용하여 다른 테이블에 관련된 데이터를 찾는 식으로 검색하는 데이터 모델

2) 시스템 카탈로그

데이터베이스의 객체(사용자, 릴레이션, 뷰, 인덱스, 권한 등)와 구조들에 관한 모든 데이터를 포함하는 시스템 데이터베이스

* 핵심 정리

1) 관계 데이터 모델

- 실 세계 데이터를 행과 열과 구성된 표(테이블, 릴레이션) 형태로 저장하고 한 테이블의 필드 값을 이용하여 다른 테이블에 관련된 데이터를 찾는 식으로 검색하는 데이터 모델
- 관계 데이터 모델 기본용어
 - 릴레이션(Relation), 속성(Attribute), 튜플(Tuple), 도메인(Domain), 차수(Degree), 카디널리티(Cardinality)
- 릴레이션 특성과 무결성 제약조건의 개념

2) 관계데이터언어

- 관계 데이터 모델의 연산이며, 원하는 데이터를 얻기 위해 릴레이션에 필요한 처리 요구를 수행하는 것으로 관계 대수와 관계 해석이 있음
- 관계대수 개념과 연산자의 종류
 - 일반 집합 연산자, 순수 관계 연산자
- 관계해석의 개념

3) 시스템 카탈로그와 뷰

- 시스템 카탈로그는 데이터베이스의 객체(사용자, 릴레이션, 뷰, 인덱스, 권한 등)와 구조들에 관한 모든 데이터를 포함하는 시스템 데이터베이스
- 뷰는 가상 테이블을 구성하는 데이터베이스 오브젝트로 별도의 저장 공간은 없지만 뷰를 통해 SELECT, DELETE, UPDATE를 할 수 있음

02. 데이터모델링 및 설계

* 용어 정리

1) 데이터 모델

데이터 모델은 현실 세계의 정보들을 컴퓨터에 표현하기 위해서 단순화, 추상화하여 체계적으로 표현한 개념적 모형

2) 이상(anomaly) 현상

불필요한 데이터 중복으로 인해 릴레이션에 대한 데이터 삽입수정삭제 연산을 수행할 때 발생할 수 있는 부작용




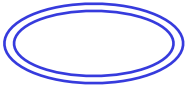


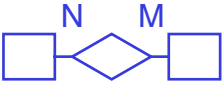

* 핵심 정리

1) 데이터 모델 개념

- 데이터 모델은 현실 세계의 정보들을 컴퓨터에 표현하기 위해서 단순화, 추상화하여 체계적으로 표현한 개념적 모형
- 데이터 모델 종류
 - 개념적 데이터 모델, 논리적 데이터 모델, 물리적 데이터 모델
- 데이터 모델 구성 요소
 - 개체, 속성, 관계
- 데이터 모델에 표시할 요소
 - 구조, 연산, 제약조건

2) 개체-관계(E-R)모델

- E-R 모델은 개체 타입(Entity Type)과 이들 간의 관계 타입(Relationship Type)을 이용하여 현실 세계를 개념적으로 표현
- E-R 다이어그램 기호

기호	기호이름	의미
	사각형	개체(Entity) 타입
	마름모	관계(Relationship) 타입
	타원	속성(Attribute)
	이중 타원	다중값 속성 (복합 속성)
	밑줄 타원	기본키 속성
	복수 타원	복합 속성 (ex. 성명과 성과 이름으로 구성)
	관계	1:1, 1:N, N:M 등의 개체 간 관계에 대한 대응수를 선 위에 기술함
	선 링크	개체 타입과 속성을 연결

3) 논리적 데이터모델링

- E-R 다이어그램으로 표현된 개념적 구조를 데이터베이스에 저장할 형태로 표현한 논리적 구조로 정의하는 과정
- 논리적 모델은 H/W나 S/W(특히 DBMS)에 독립적
- 관계 데이터 모델, 계층 데이터 모델, 네트워크 데이터 모델 등이 있음

4) 데이터베이스 정규화

- 중복성을 최소화하고 정보의 일관성 보장을 위해 릴레이션을 관련 있는 속성들로만 구성되도록 릴레이션을 분해하는 과정



5) 논리 데이터모델 품질검증

- 좋은 데이터 모델의 조건
 - 완전성, 중복배제, 비즈니스 룰, 데이터 재사용, 안정성 및 활용성, 간결성, 의사소통, 통합성
- 데이터 모델 품질 검증 기준
 - 정확성, 완전성, 준거성, 최신성, 일관성, 활용성

| Chapter 13. 물리 데이터베이스 설계

01. 물리요소 조사 분석

* 용어 정리

1) 분산 데이터베이스

논리적으로 같은 시스템에 속하지만, 컴퓨터 네트워크를 통해 물리적으로 분산되어 있는 데이터베이스

2) 데이터베이스 이중화

시스템 오류로 인한 데이터베이스 서비스 중단이나 물리적 손상 발생 시 이를 복구하기 위해 동일한 데이터베이스를 복제하여 관리

* 핵심 정리

1) 스토리지

- 스토리지는 단일 디스크로 처리할 수 없는 대용량의 데이터를 저장하기 위해 서버와 저장장치를 연결하는 기술
- 스토리지 종류
 - DAS(Direct Attached Storage)
 - NAS(Network Attached Storage)
 - SAN(Storage Area Network)

2) 분산 데이터베이스

- 논리적으로 같은 시스템에 속하지만, 컴퓨터 네트워크를 통해 물리적으로 분산되어 있는 데이터베이스
- 분산 데이터베이스의 4가지 투명성
 - 위치투명성
 - 중복투명성
 - 병행투명성
 - 장애투명성
- 분산 데이터베이스 설계 방법
 - 테이블 위치 분산, 분할, 할당

3) 데이터베이스 이중화 구성

- 시스템 오류로 인한 데이터베이스 서비스 중단이나 물리적 손상 발생 시 이를 복구하기 위해 동일한 데이터베이스를 복제하여 관리
- 데이터 이중화의 분류
 - Eager 기법
 - Lazy 기법
- 데이터 이중화의 구성 방법
 - Active-Active
 - Active-Standby

4) 데이터베이스 암호화

- 데이터베이스 암호화(Encryption) 알고리즘

구분	내용
대칭 키 (비밀키) 암호 알고리즘	DES, ARIA 128/192/256, SEED
해시 알고리즘	SHA 256/384/512, HAS - 160
비대칭 키 (공개키) 암호 알고리즘	RSA, ECDSA

- 데이터베이스 암호화 기법
 - API 방식, Plug-in 방식, Hybrid 방식

5) 접근통제

- 데이터베이스에 대한 사용자들의 접근을 통제함으로써 데이터를 보호하는 방법
- 접근통제의 3요소
 - 접근통제 정책, 접근통제 메커니즘, 접근통제 보안모델

02. 데이터베이스 물리 속성 설계

* 용어 정리

1) 파티셔닝(Partitioning)

데이터베이스에서 파티션은 대용량의 테이블이나 인덱스를 작은 논리적 단위인 파티션으로 나누는 것

- 파티션의 종류
 - 범위 분할(Range Partitioning)
 - 해시 분할(Hash Partitioning)

- 조합 분할(Composite Partitioning)

2) 클러스터(Cluster)

데이터 저장시 데이터 액세스 효율을 향상시키기 위해 동일한 성격의 데이터를 동일한 데이터 블록에 저장하는 물리적 저장 방법

3) 데이터 지역화(Data Locality)

데이터베이스의 저장 데이터를 효율적으로 이용할 수 있도록 저장하는 방법

* 핵심 정리

1) 클러스터링

- 데이터 저장시 데이터 액세스 효율을 향상시키기 위해 동일한 성격의 데이터를 동일한 데이터 블록에 저장하는 물리적 저장 방법
- 클러스터 대상 테이블
 - 분포도가 넓은 테이블
 - 대량의 범위를 자주 조회하는 테이블
 - 입력, 수정, 삭제가 자주 발생하지 않는 테이블
 - 자주 조인되어 사용되는 테이블
 - ORDER BY, GROUP BY, UNION이 빈번한 테이블

2) 데이터베이스 백업

- 데이터베이스 백업은 전산 장비의 장애에 대비하여 데이터베이스에 저장된 데이터를 보호하고 복구하기 위한 작업
- 로그 파일
 - 데이터베이스의 처리 내용이나 이용 상황 등 상태 변화를 시간의 흐름에 따라 모두 기록한 파일
- 데이터베이스 복구 알고리즘
 - NO-UNDO/REDO, UNDO/NO-REDO
 - UNDO/REDO, NO-UNDO/NO-REDO
- 백업 종류 : 물리 백업, 논리 백업

3) 데이터베이스 용량 설계

- 데이터베이스 용량을 설계할 때는 테이블에 저장할 데이터양과 인덱스, 클러스터 등이 차지하는 공간 등을 예측하여 반영함
- 데이터베이스의 용량을 정확히 산정하여 디스크의 저장공간을 효과적으로 사용하고 확장성 및 가용성을 높임
- 테이블스페이스 설계 시 고려사항
 - 테이블스페이스는 업무별로 구분하여 지정
 - 대용량 테이블은 하나의 테이블스페이스에 독립적으로 저장
 - 테이블과 인덱스는 분리하여 저장
 - LOB(Large Object)타입의 데이터는 독립적인 공간으로 지정

4) 데이터 지역화(locality)

- 데이터베이스의 저장 데이터를 효율적으로 이용할 수 있도록 저장하는 방법

- 물리적 데이터베이스 설계, 보조 기억 장치의 역할, 디스크 상의 파일의 배치를 지역화 관점에서 검토

03. 물리 데이터베이스 모델링

* 용어 정리

1) 데이터베이스 무결성 데이터의 정확성을 보장하기 위해 부정확한 자료가 데이터베이스 내에 저장되는 것을 방지하기 위한 제약 조건

- 무결성의 종류
 - 개체 무결성
 - 도메인 무결성
 - 참조 무결성
 - 사용자 정의 무결성

* 핵심 정리

1) 칼럼 속성

- 속성은 개체의 구성 항목이며 특성을 기술하며, 파일 구조상의 데이터 항목 또는 데이터 필드에 해당됨
- 속성의 특성에 따른 분류
 - 기본 속성, 설계 속성, 파생 속성
- 개체 구성 방식에 따른 분류
 - 기본키 속성, 외래키 속성, 일반 속성

2) 키 종류

- 키는 데이터베이스에서 조건에 만족하는 튜플을 찾거나 순서대로 정렬할 때 튜플들을 서로 구분할 수 있는 기준이 되는 애트리뷰트
- 키의 종류
 - 슈퍼키, 후보키, 기본키, 대체키

3) 반정규화

- 시스템의 성능 향상, 개발 및 운영의 편의성 등을 위해 정규화된 데이터 모델을 통합, 중복, 분리하는 과정으로 의도적으로 정규화 원칙을 위배하는 행위
- 반정규화의 방법
 - 테이블 통합
 - 테이블 분할
 - 중복 테이블 추가
 - 중복 속성 추가

04. 물리 데이터 모델 품질 검토

* 용어 정리

1) CRUD 매트릭스 분석

데이터 프로세스를 행으로 하고 엔티티 목록을 열로 하여 CRUD Matrix를 작성한 후 CRUD Matrix에서 사용되지 않는 프로세스와 엔티티 여부 확인하는 분석 기법

- CRUD 매트릭스 점검사항
 - 모든 엔티티 타입에 CRUD가 한 번 이상 표기
 - 모든 엔티티 타입에 C가 한 번 이상 존재
 - 모든 엔티티 타입에 R이 한 번 이상 존재
 - 모든 단위 프로세스가 하나 이상의 엔티티 타입에 표기

* 핵심 정리

1) 물리 데이터 모델 품질 기준

- 물리 데이터 모델 품질 검토의 목적은 데이터베이스 성능 향상과 오류 예방
- 물리 데이터 모델 품질 기준
 - 정확성, 완전성, 준거성
 - 최신성, 일관성, 활용성

2) 물리 E-R 다이어그램

- 논리 데이터 모델 물리 데이터 모델 변환
 - 단위 엔티티(Entity)를 테이블로 변환
 - 속성을 칼럼(Column)으로 변환
 - UID를 기본키(Primary Key)로 변환
 - 관계를 외래키(Foreign Key)로 변환
 - 관리 목적의 테이블/칼럼 추가
 - 칼럼 유형과 길이 정의
 - 데이터 표준 적용

3) SQL 성능 튜닝

- SQL 성능 튜닝의 정의
 - 튜닝 대상이 되는 SQL을 이해하고, SQL의 정보를 분석하여 성능을 개선하는 활동
- SQL 성능 튜닝의 순서
 - 문제 있는 SQL 식별
 - 옵티마이저 통계 확인
 - 실행 계획 검토
 - SQL문 재구성
 - 인덱스 재구성
 - 실행 계획 유지 관리

| Chapter 14. 데이터 전환

01. 데이터 전환 기술

* 용어 정리

1) ETL(Extraction, Transformation, Loading)

소스시스템(Source System)으로부터 필요한 데이터를 추출(Extraction)하여 새로 개발할 정보시스템에서 운영 가능하도록 변환(Transformation) 작업을 거쳐 타겟 시스템(Target System)으로 전송 및 로딩>Loading)하는 일련의 과정을 뜻함

* 핵심 정리

1) 초기데이터 구축

- 기존 운영 시스템의 이해를 바탕으로 구축 범위를 명확하게 하여 구축
- 초기 데이터 구축 절차
 - 구축 전략 수립
 - 초기 데이터 구축 대상 파악
 - 초기 데이터 구축 범위
 - 초기 데이터 구축 시 세부 고려사항

2) 파일 처리 기술

- 자료 구조이론에서 파일 처리 기술은 많은 양의 자료를 각종 매체에 저장하는 기법
- 파일 처리 기술 종류
 - 순차파일
 - 색인 순차파일
 - 가상 기억 접근 방식(VSAM) 파일
 - 직접 파일

02. 데이터 전환 수행

* 핵심 정리

1) 데이터 전환 수행 계획

- 데이터 전환 절차는 전환 계획 및 요건정의, 전환 설계, 전환 개발, 전환 테스트 및 검증으로 구분할 수 있음.
- 데이터 전환 수행 계획 작성을 위해서는 데이터 전환 범위, 전환 일정, 전환 절차를 명확히 해야 함.
- 데이터 전환 계획서의 주요 항목
 - 데이터 전환 개요, 데이터 전환 대상 및 범위,
 - 데이터 전환 환경구성, 데이터 전환 조직 및 역할,
 - 데이터 전환 일정, 데이터 전환 방안,
 - 데이터 정비 방안, 비상계획

2) 체크리스트

- 전환 프로그램의 에러, 시간의 제약, 업무 프로세스의 변경, 빈번한 데이터 요건 변경, 하드웨어 장애 등의 위험 요소에 최대한 대응하기 위해 체크리스트 작성
- 체크리스트 포함 내용
 - 데이터 전환 수행자가 수행할 작업의 상세 항목
 - 작업 내용

- 작업 담당자
- 예정 시작종료 시각

3) 데이터 검증

- 원천 시스템의 데이터를 목적 시스템의 데이터로 전환하는 과정이 정상적으로 수행되었는지 여부를 확인하는 과정
- 검증 방법에 따른 분류
 - 로그 검증, 기본 항목 검증, 응용 프로그램 검증
 - 응용 데이터 검증, 값 검증
- 데이터 단계에 따른 분류
 - 추출, 전환, DB 적재, DB 적재 후, 전환 완료 후

03. 데이터 정제

* 핵심 정리

1) 데이터 정제

- 데이터 정제 항목을 정제 시점에 따라 전환 테스트 전, 전환 테스트 중, 최종 전환 3단계로 구분하여 데이터 정제 작업을 수행

2) 데이터 품질 분석

- 데이터 품질 관리는 기관이나 조직 내외부의 정보시스템 및 DB 사용자의 기대를 만족시키기 위해 지속적으로 수행하는 데이터 관리 및 개선활동
- 원천 데이터와 전환된 목적 데이터베이스의 품질 분석
- 원천 데이터와 전환 데이터의 정합성 검증 항목

3) 오류 데이터 측정

- 데이터 중 정상 데이터와 오류 데이터를 정량적으로 측정
- 오류 목록의 내용을 확인하고 오류 해결 방안을 참조하여 원천 데이터의 정제를 요청할 것인지, 아니면 전환 프로그램을 수정할 것인지 데이터 정제 여부를 결정

Part 4

프로그래밍 언어 활용

| Chapter 15. 서버프로그램 구현

01. 개발환경 구축

* 용어 정리

1) 구현 도구

프로그램을 개발할 때 가장 많이 사용되는 도구로서 코드의 작성 및 편집, 디버깅 등과 같은 다양한 작업이 가능하며 Eclipse, NetBeans, IntelliJ 등 다양한 소프트웨어들이 사용되고 있음

2) 클라이언트(Client)

시스템에서 제공하는 서버 서비스를 활용하기 위해 거래를 발생시키는 하드웨어

3) 형상관리

소프트웨어의 개발 과정에서 소프트웨어의 변경사항을 관리하기 위해 개발된 일련의 활동

* 핵심 정리

1) 개발환경 구축

- 응용 소프트웨어 개발을 위해 개발 프로젝트를 이해하고 소프트웨어 및 하드웨어 장비를 구축하는 것
- 개발 하드웨어 환경
 - 클라이언트
 - 서버(웹 서버, WAS 서버, DB 서버, 파일 서버 등)
- 개발 소프트웨어 환경
 - 시스템 소프트웨어
 - 개발 소프트웨어
 - 요구사항 관리 도구, 설계/모델링 도구, 구현 도구,
 - 빌드 도구, 테스트 도구, 형상 관리 도구

2) 서버 개발 프레임워크

- 서버 프로그램 개발 시 다양한 네트워크 설정, 요청 및 응답처리, 아키텍처 모델 구현 등을 손쉽게 처리할 수 있도록 클래스나 인터페이스를 제공하는 소프트웨어
- 서버 개발 프레임워크의 종류
 - Spring, Node.js, Django
 - Codeigiter, Ruby on Rails

02. 공통 모듈 구현

* 핵심 정리

1) 재사용

- 목표 시스템의 개발 시간 및 비용 절감을 위하여 검증된 기능을 파악하고 재구성하여 시스템에 응용하기 위한 최적화 작업
- 재사용 범위에 따른 분류
 - 함수와 객체 재사용
 - 컴포넌트 재사용
 - 애플리케이션 재사용

2) 모듈화

- 소프트웨어 개발 작업을 실제로 개발할 수 있는 작은 단위로 나누는 것
- 모듈화 측정 척도
 - 응집도(Cohesion)
 - 결합도(Coupling)
- 모듈 간의 좋은 관계
 - 응집도는 높게, 결합도는 낮게

3) 응집도(Cohesion)

- 인터페이스의 요청을 처리함에 있어서 공통 모듈 내의 클래스들 간에 얼마나 유기적으로 협업하여 처리하는가에 관한 정도
- 응집도 유형
 - Functional Cohesion
 - Sequential Cohesion
 - Communication Cohesion
 - Procedural Cohesion
 - Temporal Cohesion
 - Logical Cohesion
 - Coincidental Cohesion

4) 결합도(Coupling)

- 어떤 모듈이 다른 모듈에 의존하는 정도
- 결합도 유형
 - Content Coupling
 - Common Coupling
 - External Coupling
 - Control Coupling
 - Stamp Coupling
 - Data Coupling

03. 서버 프로그램 구현

* 용어 정리

1) API(Application Programming Interface)

응용 프로그램 개발 시 운영체제나 프로그래밍 언어 등에 있는 라이브러리를 이용할 수 있도록 규칙 등을 정의해 놓은 인터페이스

* 핵심 정리

1) 보안 취약성 식별

- 소프트웨어 개발 보안은 소프트웨어 개발 과정에서 발생할 수 있는 보안 취약점을 최소화하고, 서버 보안 위협에 대응할 수 있는 안전한 소프트웨어를 개발하기 위한 일련의 보안 활동
- 소프트웨어 개발 보안 점검 항목
 - 입력 데이터 검증 및 표현
 - 보안 기능
 - 시간 및 상태

- 에러 처리
- 코드 오류
- 캡슐화
- API 오용

2) API

- API는 응용 프로그램 개발 시 운영체제나 프로그래밍 언어 등에 있는 라이브러리를 이용할 수 있도록 규칙 등을 정해 놓은 인터페이스
- API의 종류
 - Windows API | 단일 유닉스 규격(SUS) | Java API | 웹 API

04. 배치 프로그램 구현

* 용어 정리

1) 배치 프로그램(Batch Program)

사용자와의 상호 작용 없이 일련의 작업들을 작업 단위로 묶어 정기적으로 반복 수행하거나 정해진 규칙에 따라 일괄 처리하는 것

* 핵심 정리

1) 배치 프로그램의 필수 요소

요소	설명
대용량 데이터	· 대용량의 데이터를 처리할 수 있어야 함
자동화	· 심각한 오류 상황 외에는 사용자의 개입 없이 동작해야 함
견고함	· 유효하지 않은 데이터의 경우도 처리해서 비정상적인 동작 중단이 발생하지 않아야 함
안정성	· 어떤 문제가 생겼는지, 언제 발생했는지 등을 추적할 수 있어야 함
성능	· 주어진 시간 내에 처리를 완료할 수 있어야 하고, 동시에 동작하고 있는 다른 애플리케이션을 방해하지 말아야 함

| Chapter 16. 프로그래밍 언어 활용

01. 기본문법 활용

* 핵심 정리

1) 변수

어떤 값을 주기억 장치에 기억하기 위해서 사용하는 공간

2) 데이터 타입

변수가 가질 수 있는 속성값의 길이 및 성질

- C/C++의 데이터 타입

종류	데이터 타입	크기	기억 범위
문자	char	1Byte	-128 ~ 127
부호없는 문자형	unsigned char	1Byte	0 ~ 255
정수	short	2Byte	-32,768 ~ 32,767
	int	4Byte	-2,147,483,648 ~ 2,147,483,648
	long	4Byte	-2,147,483,648 ~ 2,147,483,648
	long long	8Byte	-9,223,137,203,685,775,808 ~ 9,223,137,203,685,775,807
실수	float	4Byte	$1.2 \times 10^{-38} \sim 3.4 \times 10^{38}$
	double	8Byte	$2.2 \times 10^{-308} \sim 1.8 \times 10^{308}$
	long double	8Byte	$2.2 \times 10^{-308} \sim 1.8 \times 10^{308}$

3) 연산자

- 데이터 처리를 위해 연산을 표현하는 기호로 +, - 등과 같은 연산자를 포함
- 연산자의 종류는 산술 연산자, 관계 연산자, 비트 연산자, 시프트 연산자, 논리 연산자 등이 있음
- 연산 표기법의 종류
 - 전위(Prefix) 표기법 $+a*bc$
 - 중위(Infix) 표기법 $a+(b*c)$
 - 후위(Postfix) 표기법 $abc*+$

02. 언어특성 활용

* 핵심 정리

1) 절차적 프로그래밍 언어

- 일련의 처리 절차를 수행하는 프로시저를 구현하며, 정해진 문법에 따라 순서대로 기술하는 언어
 - 장점 : 실행 속도가 빠르며, 모듈구성이 용이한 구조적 프로그래밍
 - 단점 : 프로그램 분석이 어렵고, 유지 보수나 코드 수정이 어려움

- 종류 : C, ALGOL, COBOL, FORTRAN

2) 객체지향 프로그래밍 언어

- 프로시저보다는 명령과 데이터로 구성된 객체를 중심으로 하는 프로그래밍 기법
 - 장점 : 재사용성, 소프트웨어 개발 및 유지보수가 용이
 - 단점 : 구현 시 처리 시간의 지연
- 특징 : 캡슐화, 정보은닉, 추상화, 상속성, 다형성
- 종류 : C#, JAVA, C++, Smalltalk

3) 스크립트 언어

- HTML 문서 안에 직접 프로그래밍 언어를 삽입하여 사용되며, 기계어로 컴파일 되지 않고 별도의 번역기가 소스를 분석하여 동작하는 언어
 - 장점 : 컴파일 없이 바로 실행하며 소스 코드를 빠르게 수정 가능
 - 단점 : 코드를 읽고 해석해야 하므로 실행속도가 느림
- 클라이언트용 스크립트 언어 : JavaScript
- 서버용 스크립트 언어 : ASP, JSP, PHP, Python

4) 선언형 언어

- 명령형 언어가 문제를 해결하기 위한 방법을 기술한다면 선언형 언어는 프로그램이 수행해야 할 문제를 기술하는 언어
- 종류 : LISP, PROLOG, Haskell, SQL, HTML, XML

03. 라이브러리 활용

* 핵심 정리

1) 라이브러리

- 프로그램을 효율적으로 개발할 수 있도록 자주 사용하는 함수나 데이터들을 미리 만들어 모아 놓은 집합체
- 일반적으로 도움말, 설치 파일, 샘플 코드 등을 제공
- 라이브러리 종류 : 표준 라이브러리, 외부 라이브러리
- C언어의 대표적인 표준 라이브러리
 - stdio.h, string.h, math.h, stdlib.h, time.h 등
- JAVA언어의 대표적인 표준 라이브러리
 - java.lang, java.util, java.io, java.net 등

2) 데이터 입출력

- 키보드로 입력 받아 화면으로 출력할 때 사용하는 함수 또는 클래스와 메소드
- C언어의 데이터 표준 입출력 함수
 - scanf(), getchar(), gets(),
 - printf(), putchar(), puts()
- Java언어의 데이터 표준 입출력
 - 입력 관련 클래스 : Scanner
 - 서식 지정 출력 : System.out.printf()

3) 예외 처리

- 프로그램의 정상적인 실행을 방해하는 조건이나 상태를 뜻하는 예외가 발생했을 때 해당 문제에 대한 처리 루틴을 수행하도록 하는 것
- Java에서의 예외 처리
 - try~catch 구문을 이용한 예외 처리

```
try{
    System.out.print("오류 테스트 중");
}catch(Exception e){
    System.out.print("오류: " + e.message());
}finally{
    System.out.print("프로그램 실행이 완료되었습니다");
}
```

4) 프로토타입

- 프로그래밍 언어에서 프로토타입이란 함수 원형(Prototype)이라는 의미로 컴파일러에게 사용될 함수에 대한 정보를 미리 알리는 것
- C언어에서는 함수가 호출되기 전에 함수가 미리 정의된 경우에는 프로토타입을 정의하지 않아도 됨
- 프로토타입에 정의된 반환 형식은 함수 정의에 지정된 반환 형식과 반드시 일치하여야 함

| Chapter 17. 응용 SW 기초 기술 활용

01. 운영체제 기초 활용

* 핵심 정리

1) 운영체제

- 운영체제 종류
 - Windows, UNIX, LINUX, MacOS, MS-DOS
- 운영체제 역할
 - 사용자와 시스템 간의 편리한 인터페이스 제공
 - 시스템의 각종 하드웨어와 네트워크를 관리하고 제어
 - 자원을 효율적으로 관리하기 위해 자원의 스케줄링 기능 제공
 - 데이터를 관리하고, 데이터 및 자원의 공유 기능 제공
 - 입출력에 대한 보조 기능을 제공
 - 시스템의 오류를 검사하고 복구

- 윈도우 특징

특징	설명
GUI (Graphic User Interface)	키보드로 명령어를 직접 입력하지 않고, 마우스로 아이콘이나 메뉴를 선택하여 모든 작업을 수행
선점형 멀티태스킹	동시에 여러 개의 프로그램을 실행하는 멀티태스킹을 하면서 운영체제가 각 작업의 CPU 이용 시간을 제어하여 응용 프로그램 시행 중 문제가 발생하면 해당 프로그램을 강제 종료시키고 모든 시스템 자원을 반환하는 방식
PnP (Plug and Play)	컴퓨터 시스템에 프린터나 사운드 카드 등의 하드웨어를 설치했을 때 해당 하드웨어를 사용하는데 필요한 시스템 환경을 운영체제가 자동으로 구성해 주는 기능
OLE (Object Linking Embedding)	다른 응용 프로그램에서 작성된 문자나 그림 등의 개체를 현재 작성 중인 문서에 자유롭게 연결(Linking)하거나 삽입(Embedding)하여 편집할 수 있게 하는 기능

2) 메모리 관리

- 기억장치의 계층 구조 : 레지스터, 캐시 기억장치, 주기억장치, 보조기억장치
- 기억장치 관리 전략 : Fetch(반입), Placement(배치), Replacement(교체)
- 주기억장치 할당 기법 : 연속 할당, 분산 할당 기법
- 가상기억장치의 구현 기법
 - 페이징 기법, 세그먼테이션 기법
- 페이지 교체 알고리즘 종류
 - FIFO, LRU, NUR, OPT, LFU, SCR 등
- 가상기억장치의 기타 관리
 - Locality, Working Set, Thrashing

3) 프로세스 스케줄링

- 프로세스 : 프로세서(처리기, CPU)에 의해 처리되는 사용자 프로그램
- 스레드 : 프로세스 내에서의 작업 단위로서 시스템의 여러 자원을 할당 받아 실행하는 프로그램의 단위
- 스케줄링 : 프로세스가 생성되어 실행될 때 필요한 시스템의 여러 자원을 해당 프로세스에게 할당하는 작업
- 비선점 스케줄링의 종류
 - FIFO(FCFS), SJF, HRN
- 선점 스케줄링의 종류
 - RR, SRT, MLQ, MFQ

4)환경변수

- 시스템 소프트웨어의 동작에 영향을 미치는 동적인 값들의 모임
- 윈도우 환경변수
 - %APPDATA%, %COMSPEC%, %PATH%, %USERNAME%
 - %PROGRAMFILES%, %SYSTEMDRIVE%
- 유닉스 환경변수
 - \$DISPLAY, \$HOME, \$LANG, \$PATH

- \$PWD, \$TERM, \$USER

5) Shell Script

CLI(Command Line Interface)는 키보드로 명령어를 직접 입력하여 작업을 수행하는 사용자 인터페이

UNIX의 기본 명령어

명령어	기능
cat	파일 내용을 화면에 표시
chdir	디렉토리의 위치 변경
chmod	파일의 사용 허가(권한) 지정
chown	소유자 변경
cp	파일 복사
getpid	자신의 프로세스 아이디를 얻음
ls	현재 디렉토리 내의 파일 목록을 표시
rm	파일 삭제

02. 네트워크 기초 활용

* 용어 정리

1) OSI 7 Layer

국제 표준화 기구인 ISO(International Standard Organization)에서 다른 시스템간 통신을 위해 네트워크 구조를 제시한 기본 모델로 7계층 구조임

2) 프로토콜(Protocol)

서로 다른 기기들 간의 데이터 교환을 원활하게 수행할 수 있도록 표준화시켜 놓은 통신 규약

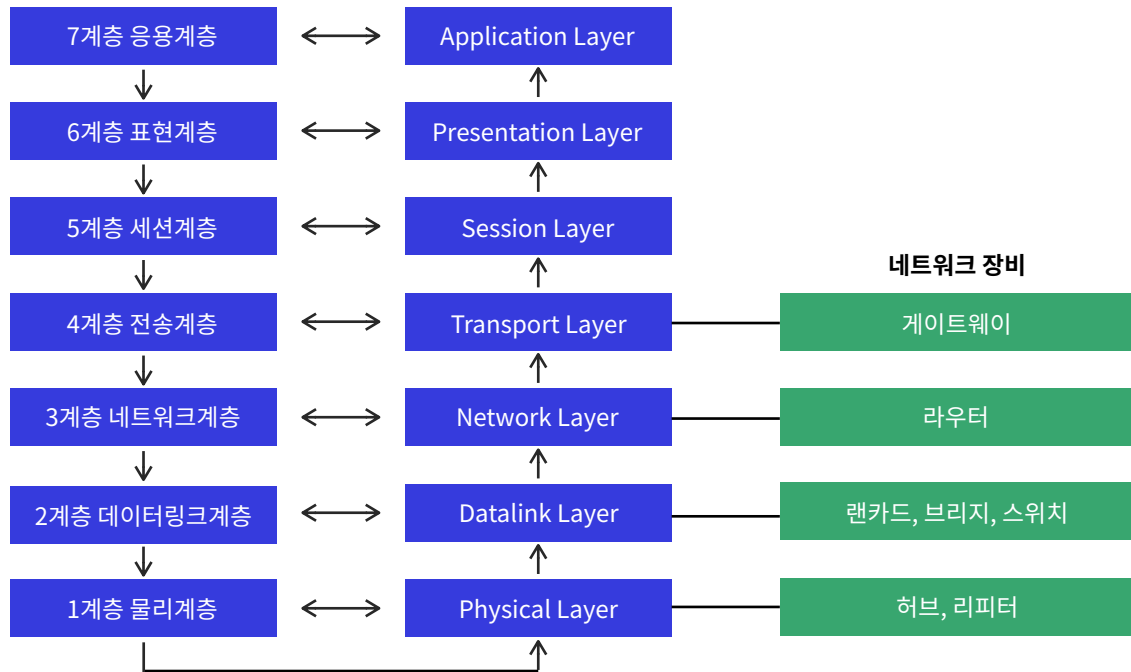
* 핵심 정리

1) 인터넷 구성의 개념

- 인터넷은 TCP/IP 프로토콜을 기반으로 전 세계 수많은 컴퓨터와 네트워크들이 연결된 광범위한 컴퓨터 통신망
- 네트워크 장비의 종류
 - 게이트웨이(Gateway)
 - 라우터(Router)
 - 리피터(Repeater)
 - 허브(Hub)
 - 랜 카드(NIC)
 - 브리지(Bridge)
 - 스위치(Switch)

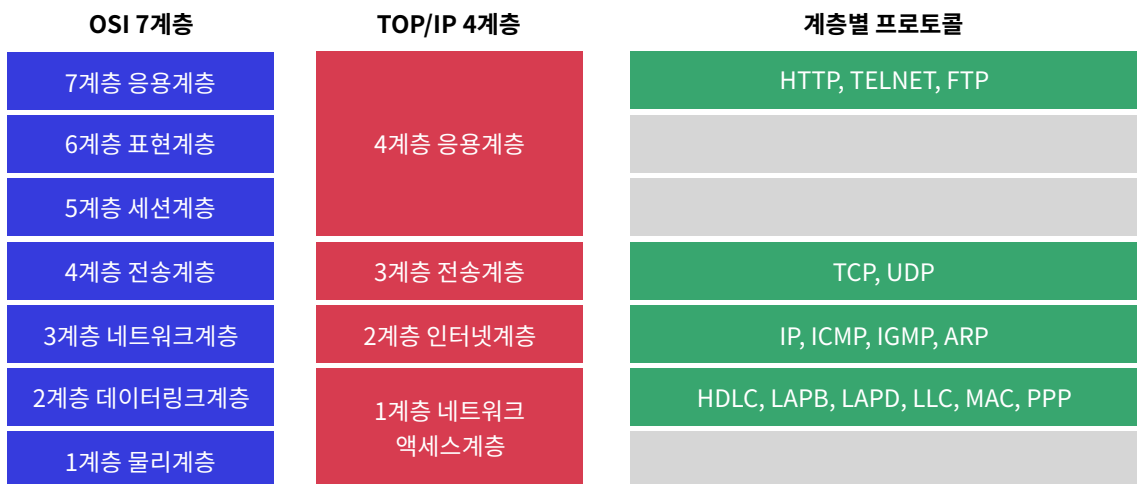
2) 네트워크 7 계층

- OSI 7계층은 국제 표준화 기구인 ISO에서 다른 시스템간 통신을 위해 네트워크 구조를 제시한 기본 모델



3) 네트워크 프로토콜

- 프로토콜은 서로 다른 기기들 간의 데이터 교환을 원활하게 수행할 수 있도록 표준화 시켜 놓은 통신 규약
- TCP/IP 4계층 : 네트워크 액세스, 인터넷, 전송, 응용 계층
- 프로토콜의 기본 요소 : 구문, 의미, 시간
- 라우팅 프로토콜 : RIP, IGRP, OSPF, BGP



4) IP

- IP는 OSI 7계층의 네트워크 계층에서 호스트의 주소지정과 패킷 분할 및 조립 기능을 담당

■ IPv4 와 IPv6 주소 비교

구분	IPv4	IPv6
주소길이	32비트	128비트
주소개수	$2^{32} = \text{약 } 43\text{억}$	$2^{128} = 43\text{억} \times 43\text{억} \times 43\text{억} \times 43\text{억}$
등분	8비트씩 4등분 예) 210.117.63.1	16비트씩 8등분 1111:12fa:a1ca:ffff:1111:ffff:0101
구분기호	.	:
진법	10진법(0~9) 2진수 8비트 10진수 표현범위 0~255	16진법 (0~9, A~F)
특징	<ul style="list-style-type: none"> - IPv4 가용 IP주소 고갈 상태 - A~E등급으로 구분 - 일반사용 A~C Class - A Class : 1회선당 2^{24} 호스트 - D Class : 멀티캐스트 방송용 - E Class : 실험용 	<ul style="list-style-type: none"> - 멀티미디어 기능강화 - IP 보안성 강화 - 3가지 주소 체계 - 유니캐스트 : 1대 1통신 - 멀티캐스트 : 1대 다 통신 - 애니캐스트 : 1대 1통신 (가까이 있는) - NAT(Network Address Translator)로 IPv4와 호환

5) TCP/UDP

- TCP는 OSI 7계층의 전송 계층에서 논리적인 1:1 가상 회선을 지원하고 CRC 체크와 재전송 기능을 통해 신뢰성 있는 연결형 서비스를 제공
- TPC 사용 서비스
 - FTP, Telnet, Http, SMTP, POP, IMAP
- UDP는 비연결성이고 신뢰성이 없는 데이터 전송
- UPT 사용 서비스
 - SNMP, DNS, TFTP, NFS, NETBIOS, 인터넷 게임

03. 기본 개발환경 구축

* 핵심 정리

1) 웹서버

- 웹서버는 웹 브라우저 클라이언트로부터 HTTP request 를 받아 HTML과 같은 정적인 contents 를 제공하는 프로그램과 해당 애플리케이션 서버가 설치된 컴퓨터
- 웹 서버 종류
 - Apache, Nginx, IIS, GWS
- WAS서버는 DB 조회나 다양한 로직 처리를 요구하는 동적인 contents 를 제공하기 위한 Application Server
- WAS 서버 종류
 - Tomcat, Undertow, JEUS, Weblogic, Websphere

2) DB서버

- 사용자, 다른 애플리케이션, 데이터베이스와 상호 작용하여 데이터를 저장하고 분석하기 위한 컴퓨터 소프트웨어
- DB서버 종류
 - Oracle, DB2, Microsoft SQL Server
 - MySQL, MongoDB 등
- DB서버 고려사항
 - 가용성, 성능, 기술지원, 상호호환성, 구축비용

3) 패키지

- 패키지 방식 개발은 여러 성공사례의 노하우를 기반으로 만들어진
- 개발된 제품을 이용하여 시스템을 구축하는 방식
- 패키지 방식 개발의 장점
 - 국제 및 산업계 표준으로 정착된 비즈니스 프로세스 적용
 - 품질이 검증된 안정적인 시스템 구축 가능
 - 개발 기간의 단축으로 비용절감 효과
- 패키지 방식 개발의 단점
 - 사용자 요구사항에 대한 대처가 쉽지 않음
 - 사용자(고객)의 프로세스 개선의 저항발생

Part 5

정보시스템 구축 관리

| Chapter 18. 소프트웨어개발 방법론 활용

01. 소프트웨어개발 방법론 선정

* 핵심 정리

1) 소프트웨어 생명주기 모델

- 소프트웨어 생명 주기 모델은 소프트웨어를 어떻게 개발할 것인가에 대한 추상적 표현으로 순차적 또는 병렬적 단계로 나누어 표현하는 형태
- 소프트웨어 생명 주기 모델의 종류
 - 폭포수 모델(Waterfall Model)
 - 프로토타입 모델(Prototype Model)
 - 나선형 모델(Spiral Model)
 - 애자일 모델(Agile Model)

2) 소프트웨어 개발 방법론

- 소프트웨어 개발, 유지보수 등에 필요한 여러 가지 일들의 수행 방법과 이러한 일들을 효율적으로 수행하려는 과정에서 필요한 각종 기법 및 도구를 체계적으로 정리하여 표준화한 것
- 소프트웨어 개발 방법론의 종류
 - 구조적 방법론
 - 정보공학 방법론
 - 객체지향 방법론
 - 컴포넌트 기반 방법론
 - 애자일 방법론
 - 제품 계열 방법론

3) 요구공학 방법론

- 요구사항의 지속적인 중요성 증대와 체계적인 관리의 필요성 대두되어 시스템 요구사항을 정의하고, 문서화하고, 관리하는 프로세스
- 요구공학 프로세스
 - 요구사항 도출(Elicitation)
 - 요구사항 분석(Analysis)
 - 요구사항 명세(Specification)
 - 요구사항 확인/검증(Validation/Verification)

4) 비용산정 모델

- 소프트웨어의 개발 규모를 소요되는 인원, 자원, 기간 등으로 확인하여 실행 가능한 계획을 수립하기 위해 필요한 비용을 예측하는 과학적이고 합리적인 활동
- 하향식 산정기법 : 전문가 감정 기법, 델파이 기법
- 상향식 산정기법 : LOC 기법, Effort Per Task 기법
- 수학적 산정기법
 - COCOMO 모형
 - Putnam 모형
 - FP 모형

02. 소프트웨어개발 방법론 테일러링

* 핵심 정리

1) 소프트웨어 개발 표준

- 소프트웨어 개발 단계의 품질 관리에 사용되는 국제 표준
- ISO/IEC 12207
 - 국제 표준화 기구에서 제공하는 소프트웨어 생명주기 프로세스 표준
 - 기본 생명 주기, 지원 생명 주기, 조직 생명 주기 프로세스
- CMMI
 - 소프트웨어 개발 조직의 업무 능력 및 조직의 성숙도 평가 모델
 - 성숙도는 5단계 : 초기, 관리, 정의, 정량적 관리, 최적화
- SPICE
 - 소프트웨어 프로세스를 평가 및 개선하는 국제 표준
 - 수행 능력 6단계 : 불완전, 수행, 관리, 확립, 예측, 최적화

2) 테일러링 기준

- 소프트웨어 개발 방법론 테일러링은 프로젝트 상황 및 특성에 맞도록 개발 방법론의 절차, 사용기법 등을 수정 및 보완하는 작업
- 소프트웨어 개발 방법론 테일러링 고려사항
 - 내부적 기준 : 목표환경, 요구사항, 프로젝트 규모, 보유기술
 - 외부적 기준 : 법적 제약사항, 표준 품질기준
- 소프트웨어 개발 방법론 테일러링 기법
 - 프로젝트 규모와 복잡도, 프로젝트 구성원,
 - 팀내 방법론 지원, 자동화에 따른 테일러링

3) 소프트웨어 개발 프레임워크

- 소프트웨어 개발에 공통적으로 사용되는 구성 요소와 아키텍처를 일반화하여 손쉽게 구현할 수 있도록 여러 가지 기능들을 제공해 주는 반제품 형태의 소프트웨어 시스템
- 소프트웨어 개발 프레임워크 종류
 - 스프링 프레임워크(Spring Framework)
 - 전자정부 표준 프레임워크
 - 스트럿츠 프레임워크(STRUTS Framework)
 - 닷넷 프레임워크(.NET Framework)
 - 앵귤러 JS(Angular JS)
 - 장고 프레임워크(Django Framework)

| Chapter 19. IT프로젝트 정보시스템 구축관리

01. 네트워크 구축 관리

* 용어 정리

1)백본 스위치(Backbone Switch)

여러 네트워크들을 연결할 때 중추적 역할을 하는 네트워크 백본(Backbone)이라 하고 백본에서 스위칭 역할을 하는 장비

* 핵심 정리

1) IT 신기술 및 네트워크 장비 트렌드 정보

- 네트워크 관련 신기술
- 애드 혹 네트워크(Ad-hoc Network)
- 클라우드 컴퓨팅(Cloud Computing)
- IoT(Internet of Things, 사물 인터넷)
- NDN(Named Data Networking)
- NFC(Near Field Communication, 근거리 무선 통신)
- 피코넷(PICONET)
- USN(Ubiquitous Sensor Network, 유비쿼터스 센서 네트워크) 등

2)네트워크 장비(라우터, 백본 스위치 등)

- 네트워크 토폴로지(Topology) 종류
 - 성형, 버스형, 링형, 트리형, 망형
- 네트워크 분류
 - LAN, MAN, WAN, VAN
- 스위치(Switch)의 분류
 - L2, L3, L4, L7 스위치
- Hierarchical 3 Layer 모델
 - 액세스 계층, 디스트리뷰션 계층, 코어계층

02. SW구축관리

* 핵심 정리

1) IT 신기술 및 SW 개발 트렌드 정보

- 소프트웨어 관련 신기술
 - 인공지능(AI ; Artificial Intelligence)
 - 증강현실(AR ; Augmented Reality)
 - 블록체인(Blockchain)
 - 딥 러닝(Deep Learning)
 - 전문가 시스템(Expert System)
 - 그레이웨이(Grayware)
 - 매시업(Mashup)
 - 시맨틱 웹(Semantic Web)
 - 서비스 지향 아키텍처(SOA ; Service Oriented Architecture)

2) SW개발보안 정책

- SW개발 생명주기의 각 단계에서 요구되는 보안활동을 수행해
- 안전한 소프트웨어를 개발하는 것이 목적임
- SW개발 보안 관련 기관
 - 행정안전부, 한국인터넷진흥원, 발주기관,
 - 사업자, 감리법인
- SW개발 역할별 보안활동
 - 프로젝트 관리자, 요구사항 분석가, 아키텍트,
 - 설계자, 구현개발자, 테스트분석가, 보안감사자
- SW개발보안 관련 법령 및 규정
 - 개인정보 보호법
 - 정보통신망 이용촉진 및 정보보호 등에 관한 법률 등

03. HW구축관리

* 용어 정리

1) Secure-OS

기존의 운영체제에 내재된 보안 취약점을 해소하기

위해 보안 기능을 갖춘 커널을 추가하여 외부의 침입으로부터 시스템 자원을 보호하는 운영체제

2) 고가용성(High Availability)

긴 시간 동안 안정적인 서비스 운영을 위해 장애 발생 시 즉시 다른 시스템으로 대체 가능한 환경을 구축하는 메커니즘

* 핵심 정리

1) IT 신기술 및 서버장비 트렌드 정보

- 하드웨어 관련 신기술
 - 3D 프린팅(Three Dimension Printing)
 - 엠디스크(M-DISC, Millennial DISC)
 - 멤리스트(Memristor)
 - 네트워크 가상화(Network Virtualization)
 - 앤 스크린(N-Screen)
 - RAID(Redundant Array of Inexpensive Disk)
 - 서버 가상화(Server Virtualization)

2) 서버장비 운영(Secure-OS, NAS, DAS, SAN, 고가용성 등)

- 서버 장비 운영 요소
 - 서버 장비 뿐만 아니라 스토리지, 운영 체제, 고가용성 장비, 보안 솔루션 등 정보 시스템 운영에 필요한 모든 것들을 포함
- 저장 장치(스토리지 시스템)
 - DAS, NAS, SAN
- 고가용성(High Availability)
 - 안정적인 서비스 운영을 위해 장애 발생시 즉시 다른 시스템으로 대체 가능한 환경을 구축하는 메커니즘

04. DB구축관리

* 핵심 정리

1) IT 신기술 및 데이터베이스 기술 트렌드 정보

- 데이터베이스 관련 신기술
 - 빅데이터(Big Data)
 - 브로드 데이터(Broad Data)
 - 디지털 아카이빙(Digital Archiving)
 - 메타 데이터(Meta Data)
 - 데이터 다이어트(Data Diet)
 - 하둡(Hadoop)
 - 타조(Tajo)

2) 데이터베이스 관리기능

- 데이터베이스 회복(Recovery)
 - 트랜잭션들을 수행하는 도중에 장애가 발생하여 데이터베이스가 손상되었을 때 손상되기 이전의 정상 상태로 복구하는 작업
- 병행제어(Concurrency Control)
 - 다중 프로그램의 이점을 활용하여 동시에 여러 개의 트랜잭션을 병행 수행할 때, 동시에 실행되는 트랜잭션들이 데이터베이스의 일관성을 파괴하지 않도록 트랜잭션 간의 상호작용을 제어하는 것

3) 데이터베이스 표준화

- 데이터 표준화는 시스템을 구성하는 데이터 요소의
- 명칭, 정의, 형식, 규칙에 대한 원칙을 수립하고 적용하는 것
- 데이터베이스 표준화 구성요소
 - 데이터 표준, 데이터 관리 조직, 데이터 표준화 절차

| Chapter 20. 소프트웨어 개발 보안 구축

01. SW 개발 보안 설계

* 핵심 정리

1) Secure SDLC(Software Development Life Cycle)

- Secure SDLC는 보안상 안전한 소프트웨어를 개발하기 위해 소프트웨어 개발 생명 주기에 보안 강화를 위한 프로세스를 포함한 것
- 요구사항 분석, 설계, 구현, 테스트, 유지보수 등 SDLC 전체 단계에 걸쳐 수행되어야 할 보안 활동 제시
- 보안 필수 요소
 - 기밀성(Confidentiality)
 - 무결성(Integrity)
 - 가용성(Availability)

2) 입력데이터 검증 및 표현

- 폼 양식의 입력란을 통해 입력되는 데이터로 인해 발생하는 문제들을 예방하기 위해 구현 단계에서 검증해야 하는 보안 점검 항목들
- 보안 약점 종류
 - SQL 삽입
 - 크로스사이트스크립트(XSS)
 - 위험한 형식 파일 업로드
 - 경로 조작 및 자원 삽입

3) 보안기능(인증, 접근제어, 기밀성, 권한 관리 등)

- 소프트웨어 개발의 구현 단계에서 코딩하는 기능인 인증, 접근제어, 기밀성, 암호화 등을 올바르게 구현하기 위한 보안 점검 항목들
- 보안 약점 종류
 - 적절한 인증 없이 중요기능 허용
 - 중요한 자원에 대한 잘못된 권한 설정
 - 부적절한 인가
 - 하드코딩된 비밀번호
 - 취약한 암호화 알고리즘 사용
 - 중요정보 평문저장 및 전송

4) 에러처리

- 소프트웨어 실행 중 발생할 수 있는 오류(Error)들을 사전에 정의하여 오류로 인해 발생할 수 있는 문제들을 예방하기 위한 보안 점검 항목들
- 보안 약점의 종류
 - 오류 메시지를 통한 정보노출
 - 오류 상황에 대한 대응 부재
 - 부적절한 예외처리

5) 세션통제

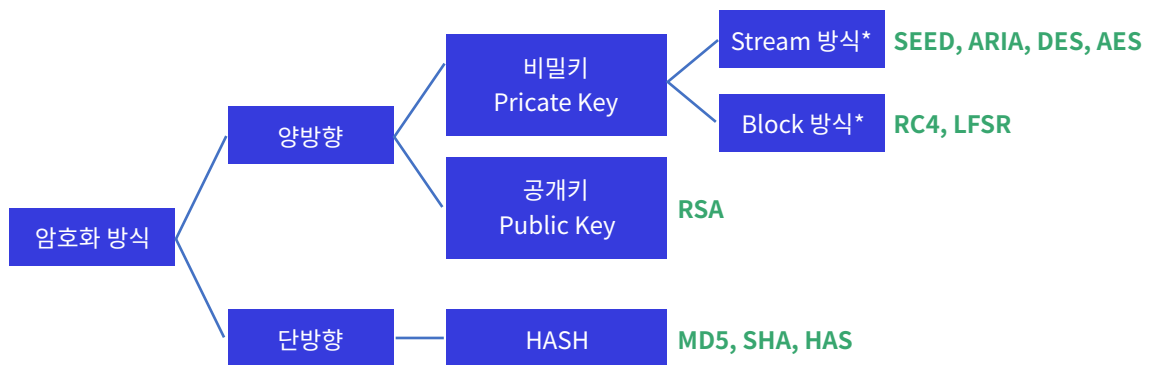
- 세션은 서버와 클라이언트의 연결을 의미하고 세션통제는 세션의 연결과 연결로 인해 발생하는 정보를 관리하는 것
- 보안 약점의 종류
 - 불충분한 세션 관리
 - 잘못된 세션에 의한 정보 노출
 - 세션 설계 시 고려사항과 세션ID의 관리방법

02. SW 개발 보안 구현

* 핵심 정리

1) 암호 알고리즘

- 주민번호, 패스워드, 은행계좌 등 중요 정보를 보호하기 위해 평문을 암호문으로 만드는 절차 또는 방법
- 해시(HASH)를 사용하는 단방향 암호화 방식과 개인키와 공개키로 분류되는 양방향 암호화 방식으로 구분됨



2) 코드오류

- 코드 오류는 소프트웨어 구현 단계에서 개발자들이 코딩 중 실수하기 쉬운 형(Type) 변환, 자원 반환 등의 오류를 예방하기 위한 보안 점검 항목들임
- 보안 약점의 종류
 - 널(Null) 포인터 역참조
 - 부적절한 자원 해제
 - 해제된 자원 사용
 - 초기화되지 않은 변수 사용

3) 캡슐화

- 캡슐화의 보안점검은 정보 은닉이 필요한 중요한 데이터와 기능을 불충분하게 캡슐화하거나 잘못 사용함으로써 발생할 수 있는 문제 예방
- 캡슐화 보안 약점의 종류
 - 잘못된 세션에 의한 정보 노출
 - 제거 되지 않고 남은 디버그 코드
 - 시스템 데이터 정보 노출

- Public 메소드로부터 반환된 Private 배열
- Private 배열에 Public 데이터 할당

4) API 오용

- 소프트웨어 구현 단계에서 API를 잘못 사용하거나 보안에 취약한 API를 사용하지 않도록 하기 위한 보안 검증 항목들
- API 오용의 보안 약점
 - DNS lookup에 의존한 보안 결정
 - 취약한 API 사용

| Chapter 21. 시스템 보안 구축

01. 시스템 보안 설계

* 핵심 정리

1) 서비스 공격 유형

- 서비스 거부 공격의 종류
 - Ping Flood, Ping of Death, Smurfing,
 - SYN Flooding, TearDrop, Land, DDoS
- 네트워크 침해 공격
 - Smishing, APT(지능형 지속 위협),
 - Spear Phishing, Qshing
- 정보 보안 침해 공격
 - Botnet, Worm, Zero Day Attack,
 - Ransomware, Trojan Horse,
 - Key Logger Attack

2) 서버 인증

- 보안 서버는 인터넷을 통해 개인정보를 암호화하여 송수신 할 수 있는 기능을 갖춘 서버
- 인증은 다중 사용자 컴퓨터 시스템이나 네트워크 시스템에서 로그인을 요청한 사용자의 정보를 확인하고 접근 권한을 검증하는 보안 절차
- 인증의 종류
 - 지식 기반 인증(Something You Know)
 - 소유 기반 인증(Something You Have)
 - 생체 기반 인증(Something You Are)
 - 행위 기반 인증(Something You Do)

3) 서버 접근통제

- 접근통제는 사용자 및 장비의 접근 필요성에 따라 정보자산에 대한 접근 권한을 부여함으로써 비인가자의 무단 접근을 제한
- 접근통제의 3요소
 - 접근통제 정책, 접근통제 메커니즘, 접근통제 보안모델

- 접근통제 정책 종류
 - 임의적 접근통제
 - 강제적 접근통제
 - 역할기반 접근통제
 - 접근통제 행렬
 - 접근 제어(AC)

4) 보안 아키텍처

- 정보 시스템의 무결성(Integrity), 가용성(Availability), 기밀성(Confidentiality)을 확보하기 위해 보안 요소 및 보안 체계를 식별하고 이들 간의 관계를 정의한 구조
- 보안 아키텍처를 통해 관리적, 물리적, 기술적 보안 개념의 수립, 보안 관리 능력의 향상, 일관된 보안 수준의 유지를 기대할 수 있음
- 보안 아키텍처는 보안 수준에 변화가 생겨도 기본 보안 아키텍처의 수정 없이 지원할 수 있어야 함
- 보안 아키텍처는 보안 요구사항의 변화나 추가를 수용할 수 있어야 함

5) 보안 프레임워크

- 프레임워크는 ‘뼈대’, ‘골조’를 의미하는 용어이며, 보안 프레임워크는 안전한 정보시스템 환경을 유지하고 보안 수준을 향상시키기 위한 체계
- ISO 27001은 정보보안 관리를 위한 국제 표준으로, 일종의 보안 인증이자 가장 대표적인 보안 프레임워크임

02. 시스템 보안 구현

* 핵심 정리

1) 로그 분석

- 로그는 시스템 사용에 대한 모든 내역을 기록해 놓은 것으로 로그를 분석하여 시스템에 대한 침입 흔적이나 취약점 확인 가능
- 리눅스의 주요 로그 파일

로그	파일명	데몬	내용
커널 로그	/dev/console	kernel	커널*에 관련된 내용을 관리자에게 알리기 위해 파일로 저장하지 않고 지정된 장치에 표시
부팅 로그	/var/log/boot.log	boot	부팅 시 나타나는 메시지들을 기록
크론 로그	/var/log/cron	crond	작업 스케줄러인 crond의 작업 내역을 기록
시스템 로그	/var/log/messages	syslogd	커널에서 실시간으로 보내오는 메시지들을 기록
보안 로그	/var/log/secure	xinetd	시스템의 접속에 대한 로그를 기록
FTP 로그	/var/log/xferlog	ftpd	FTP로 접속하는 사용자에 대한 로그를 기록
메일	/var/log/maillog	Sendmail Popper	송수신 메일에 대한 로그를 기록

2) 보안 솔루션

- 접근 통제, 침입 차단 및 탐지 등을 수행하여 외부로부터의 불법적인 침입을 막는 기술 및 시스템
- 주요 보안 솔루션

- 방화벽(Firewall)
- 침입 탐지 시스템(IDS)
- 침입 방지 시스템(IPS)
- 데이터 유출 방지(DLP)
- 웹 방화벽(Web Firewall)
- VPN(Virtual Private Network)
- NAC(Network Access Control)
- ESM(Enterprise Security Management)

3) 취약점 분석

■ 사이버 위협으로부터 정보시스템의 취약점을 분석 및 평가한 후 개선하는 일련의 과정

- 수행 절차 및 방법
 - 취약점 분석평가 계획 수립
 - 취약점 분석 평가 대상 선별
 - 취약점 분석 수행
 - 취약점 평가 수행