

## 2020년 2회 정보처리기사 실기 시험 100% 합격전략집

# 필수 암기 119선

- 1장 프로그래밍 언어 활용
- 2장 요구사항 확인
- 3장 데이터 입·출력 구현
- 4장 통합 구현
- 5장 서버 프로그램 구현
- 6장 화면 설계
- 7장 애플리케이션 테스트 관리
- 8장 SQL 응용
- 9장 소프트웨어 개발 보안 구축
- 10장 응용 SW 기초 기술 활용
- 11장 제품 소프트웨어 패키징



# 2020년 2회 정보처리기사 실기 대비용 필수 암기 119선

## 1장 | 프로그래밍 언어 활용

### [암기 001] 서식 문자열

서식 문자열	의미
%d	정수형 10진수를 입 · 출력하기 위해 지정함
%u	부호없는 정수형 10진수를 입 · 출력하기 위해 지정함
%o	정수형 8진수를 입 · 출력하기 위해 지정함
%x	정수형 16진수를 입 · 출력하기 위해 지정함
%c	문자를 입 · 출력하기 위해 지정함
%s	문자열을 입 · 출력하기 위해 지정함
%f	소수점을 포함하는 실수를 입 · 출력하기 위해 지정함
%e	지수형 실수를 입 · 출력하기 위해 지정함
%ld	long형 10진수를 입 · 출력하기 위해 지정함
%lo	long형 8진수를 입 · 출력하기 위해 지정함
%lx	long형 16진수를 입 · 출력하기 위해 지정함
%p	주소를 16진수로 입 · 출력하기 위해 지정함

### [암기 002] 주요 제어문자

문자	의미	기능
\n	new line	커서를 다음 줄 앞으로 이동함
\b	backspace	커서를 왼쪽으로 한 칸 이동함
\t	tab	커서를 일정 간격 띄움
\r	carriage return	커서를 현재 줄의 처음으로 이동함
\0	null	널 문자를 출력함

\'	single quote	작은따옴표를 출력함
\"	double quote	큰따옴표를 출력함
\a	alert	스피커로 벨 소리를 출력함
\\	backslash	역 슬래시를 출력함
\f	form feed	한 페이지를 넘김

### [암기 003] 연산자 우선순위

다음 표의 한 줄에 가로로 나열된 연산자는 우선순위가 같기 때문에 결합규칙에 따라 ←는 오른쪽에 있는 연산자부터, →는 왼쪽에 있는 연산자부터 차례로 계산된다.

대분류	중분류	연산자	결합 규칙	우선 순위
단항 연산자	단항 연산자	!(논리 not) ~ (비트 not) ++ (증가) -- (감소) sizeof (기타)	←	높음 ↑
	산술 연산자	* / % (나머지) + -		
	시프트 연산자	<< >>		
	관계 연산자	< <= >= > == (같다) != (같지 않다)	→	
	비트 연산자	& (비트 and) ^ (비트 xor)   (비트 or)		
삼항 연산자	조건 연산자	? :	→	낮음 ↓
	대입 연산자	= += -= *= /= %= <<= >>= 등	←	
	순서 연산자	(, 콤마)	→	

## [암기 004] 절차적 프로그래밍 언어의 종류

언어	특징
C	<ul style="list-style-type: none"> <li>1972년 미국 벨 연구소의 데니스 리치에 의해 개발되었음</li> <li>시스템 소프트웨어를 개발하기 편리하여 시스템 프로그래밍 언어로 널리 사용됨</li> <li>컴파일러 방식의 언어이며, 이식성이 좋아 컴퓨터 기종에 관계없이 프로그램을 작성할 수 있음</li> </ul>
ALGOL	<ul style="list-style-type: none"> <li>수치 계산이나 논리 연산을 위한 과학 기술 계산용 언어</li> <li>PASCAL과 C언어의 모체가 되었음</li> </ul>
COBOL	<ul style="list-style-type: none"> <li>사무 처리용 언어</li> <li>영어 문장 형식으로 구성되어 있어 이해와 사용이 쉬움</li> <li>4개의 DIMENSION으로 구성되어 있음</li> </ul>
FORTRAN	<ul style="list-style-type: none"> <li>과학 기술 계산용 언어</li> <li>수학과 공학 분야의 공식이나 수식과 같은 형태로 프로그래밍 할 수 있음</li> </ul>

## [암기 005] 객체지향 프로그래밍 언어의 종류

언어	특징
Java	<ul style="list-style-type: none"> <li>분산 네트워크 환경에 적용이 가능하며, 멀티스레드 기능을 제공하므로 여러 작업을 동시에 처리할 수 있음</li> <li>운영체제 및 하드웨어에 독립적이며, 이식성이 강함</li> <li>캡슐화가 가능하고 재사용성 높음</li> </ul>
C++	<ul style="list-style-type: none"> <li>C언어에 객체지향 개념을 적용한 언어</li> <li>모든 문제를 객체로 모델링하여 표현함</li> </ul>
Smalltalk	<ul style="list-style-type: none"> <li>1세대 객체지향 프로그래밍 언어 중 하나로 순수한 객체지향 프로그래밍 언어</li> <li>최초로 GUI를 제공했음</li> </ul>

## [암기 006] 객체지향 프로그래밍 언어의 구성 요소

객체 (Object)	<ul style="list-style-type: none"> <li>데이터(속성)와 이를 처리하기 위한 연산(메소드)을 결합시킨 실체</li> <li>데이터 구조와 그 위에서 수행되는 연산들을 가지고 있는 소프트웨어 모듈임</li> <li>속성(Attribute) : 한 클래스 내에 속한 객체들이 가지고 있는 데이터 값들을 단위별로 정의하는 것으로서 성질, 분류, 식별, 수량 또는 현재 상태 등을 표현함</li> <li>메소드(Method) : 객체가 메시지를 받아 실행해야 할 때 구체적인 연산을 정의하는 것으로, 객체의 상태를 참조하거나 변경하는 수단이 됨</li> </ul>
클래스 (Class)	<ul style="list-style-type: none"> <li>두 개 이상의 유사한 객체들을 묶어서 하나의 공통된 특성을 표현하는 요소임. 즉 공통된 특성과 행위를 갖는 객체의 집합이라고 할 수 있음</li> <li>객체의 유형 또는 타입(Object Type)을 의미함</li> <li>데이터를 추상화하는 단위임</li> </ul>
메시지 (Message)	<ul style="list-style-type: none"> <li>객체들 간에 상호작용을 하는데 사용되는 수단으로 객체의 메소드(동작, 연산)를 일으키는 외부의 요구 사항</li> <li>메시지를 받은 객체는 대응하는 연산을 수행하여 예상된 결과를 반환하게 됨</li> </ul>

## [암기 007] 객체지향 프로그래밍 언어의 특징

캡슐화 (Encapsulation)	<ul style="list-style-type: none"> <li>데이터(속성)와 데이터를 처리하는 함수를 하나로 묶는 것</li> <li>캡슐화된 객체의 세부 내용이 외부에 은폐(정보 은닉)되어, 변경이 발생할 때 오류의 파급 효과가 적음</li> <li>캡슐화된 객체들은 재사용이 용이함</li> </ul>
정보 은닉 (Information Hiding)	<p>캡슐화에서 가장 중요한 개념으로, 다른 객체에게 자신의 정보를 숨기고 자신의 연산만을 통하여 접근을 허용하는 것</p>
추상화 (Abstraction)	<ul style="list-style-type: none"> <li>불필요한 부분을 생략하고 객체의 속성 중 가장 중요한 것에만 중점을 두어 개략화하는 것, 즉 모델화하는 것</li> <li>데이터의 공통된 성질을 추출하여 슈퍼클래스를 선정하는 개념임</li> </ul>

상속성 (Inheritance)	<ul style="list-style-type: none"> <li>이미 정의된 상위 클래스(부모 클래스)의 모든 속성과 연산을 하위 클래스가 물려받는 것</li> <li>상속성을 이용하면 하위 클래스는 상위 클래스의 모든 속성과 연산을 자신의 클래스 내에서 다시 정의하지 않고서도 즉시 사용할 수 있음</li> </ul>
다형성 (Polymorphism)	<ul style="list-style-type: none"> <li>메시지에 의해 객체(클래스)가 연산을 수행하게 될 때 하나의 메시지에 대해 각 객체(클래스)가 가지고 있는 고유한 방법(특성)으로 응답할 수 있는 능력을 의미</li> <li>객체(클래스)들은 동일한 메소드명을 사용하며 같은 의미의 응답을 함</li> </ul>



## [암기 008] 스크립트 언어

- HTML 문서 안에 직접 프로그래밍 언어를 삽입하여 사용하는 것으로, 기계어로 컴파일 되지 않고 별도의 번역기가 소스를 분석하여 동작하게 하는 언어이다.
- 스크립트 언어의 분류

서버용 스크립트 언어	<ul style="list-style-type: none"> <li>서버에서 해석되어 실행된 후 결과만 클라이언트로 보내는 스크립트 언어</li> <li>종류 : ASP, JSP, PHP, 파이썬(Python), 펄(Perl), 루비(Ruby) 등</li> </ul>
클라이언트용 스크립트 언어	<ul style="list-style-type: none"> <li>클라이언트의 웹 브라우저에서 해석되어 실행되는 스크립트 언어</li> <li>종류 : 자바 스크립트(Java Script), VB 스크립트(Visual Basic Script) 등</li> </ul>

## [암기 009] 스크립트 언어의 종류

자바 스크립트 (Java Script)	<ul style="list-style-type: none"> <li>넷스케이프(Netscape)의 브랜던 아이크(Brendan Eich)가 개발한 언어</li> <li>웹 페이지의 동작을 제어하며, 변수 선언이 필요 없음</li> </ul>
ASP(Active Server Page)	<ul style="list-style-type: none"> <li>서버 측에서 동적으로 수행되는 페이지를 만들기 위한 언어</li> <li>마이크로 소프트사에서 제작하였으며, Windows 계열에서만 수행 가능함</li> </ul>
JSP(Java Server Page)	<ul style="list-style-type: none"> <li>Java로 만들어진 서버용 스크립트 언어</li> <li>다양한 운영체제에서 사용이 가능함</li> </ul>
PHP(Professional Hypertext Preprocessor)	<ul style="list-style-type: none"> <li>서버용 스크립트 언어로, Linux, Unix, Windows 운영체제에서 사용이 가능함</li> <li>C, Java 등과 문법이 유사하여 배우기 쉽고 웹 페이지 제작에 많이 사용됨</li> </ul>
파이썬 (Python)	<ul style="list-style-type: none"> <li>객체지향 기능을 지원하는 대화형 인터프리터 언어</li> <li>플랫폼에 독립적이고 문법이 간단하여 배우기 쉬움</li> </ul>

## [암기 010] 선언형 언어

- 프로그램이 수행해야 할 문제를 기술하는 언어로, 목표를 명시하고 알고리즘은 명시하지 않는다.
- 선언형 언어의 분류

함수형 언어	<ul style="list-style-type: none"> <li>수학적 함수를 조합하여 문제를 해결하는 언어로, 알려진 값을 함수에 적용하는 것을 기반으로 함</li> <li>대표적인 언어 : LISP</li> </ul>
논리형 언어	<ul style="list-style-type: none"> <li>기호 논리학에 기반을 둔 언어로, 논리 문장을 이용하여 프로그램을 표현하고 계산을 수행함</li> <li>대표적인 언어 : PROLOG</li> </ul>

## [암기 011] 선언형 언어의 종류

HTML	<ul style="list-style-type: none"> <li>인터넷의 표준 문서인 하이퍼텍스트 문서를 만들기 위해 사용하는 언어</li> <li>특별한 데이터 타입이 없는 단순한 텍스트이므로 호환성이 좋고 사용이 편리함</li> </ul>
LISP	인공지능 분야에 사용되는 언어로, 기본 자료 구조가 연결 리스트 구조이며 재귀(Recursion) 호출을 많이 사용함
PROLOG	논리학을 기초로 한 고급 언어로, 인공 지능 분야에서의 논리적인 추론이나 리스트 처리 등에 주로 사용됨
XML	<ul style="list-style-type: none"> <li>기존 HTML의 단점을 보완하여 웹에서 구조화된 폭넓고 다양한 문서들을 상호 교환할 수 있도록 설계된 언어</li> <li>사용자가 HTML에 새로운 태그(Tag)를 정의할 수 있음</li> </ul>
Haskell	함수형 프로그래밍 언어로, 부작용(Side Effect)이 없고 코드가 간결하여 에러 발생 가능성이 낮음

## 2장 | 요구사항 확인

### [암기 012] 요구사항 유형

기능 요구사항	시스템이 갖춰야할 필수적인 기능에 대한 요구사항
비기능 요구사항	필수 기능 외의 품질이나 제약사항에 관한 요구사항
사용자 요구사항	사용자 관점에서 본 시스템이 제공해야 할 요구사항
시스템 요구사항	개발자 관점에서 본 시스템 전체가 사용자와 다른 시스템에 제공해야 할 요구사항

### [암기 013] 요구사항 개발 프로세스

요구사항 도출 (Requirement Elicitation)	<ul style="list-style-type: none"> <li>시스템, 사용자, 그리고 시스템 개발에 관련된 사람들이 서로 의견을 교환하여 요구사항이 어디에 있는지, 어떻게 수집할 것인지를 식별하고 이해하는 과정</li> <li>주요 기법 : 인터뷰, 설문, 브레인스토밍, 워크샵, 프로토타이핑, 유스케이스</li> </ul>
요구사항 분석 (Requirement Analysis)	<ul style="list-style-type: none"> <li>개발 대상에 대한 사용자의 요구사항 중 명확하지 않거나 모호하여 이해되지 않는 부분을 발견하고 이를 걸러내기 위한 과정</li> <li>비용과 일정에 대한 제약 설정, 타당성 조사</li> </ul>
요구사항 명세 (Requirement Specification)	요구사항을 체계적으로 분석한 후 승인될 수 있도록 문서화하는 것
요구사항 확인 (Requirement Validation)	개발 자원을 요구사항에 할당하기 전에 요구사항 명세서가 정확하고 완전하게 작성되었는지를 검토하는 활동

### [암기 014] 요구사항 분석 기법

- 요구사항 분류(Requirement Classification) : 요구사항을 명확히 확인할 수 있도록 요구사항을 분류함
- 개념 모델링(Conceptual Modeling)
  - 요구사항을 보다 쉽게 이해할 수 있도록 현실 세계의 상황을 단순화하여 개념적으로 표현한 것을 모델이라고 하며, 이러한 모델을 만드는 과정을 모델링이라고 한다.
  - 개념 모델 종류 : 유스케이스 다이어그램, 데이터 흐름 모델, 상태 모델, 목표기반 모델, 사용자 인터랙션, 객체 모델, 데이터 모델 등
- 요구사항 할당(Requirement Allocation) : 요구사항을 만족시키기 위한 구성 요소를 식별하는 것
- 요구사항 협상(Requirement Negotiation) : 요구사항이 서로 충돌될 경우 이를 적절히 해결하는 과정
- 정형 분석(Formal Analysis) : 구문(Syntax)과 의미(Semantics)를 갖는 정형화된 언어를 이용해 요구사항을 수학적 기호로 표현한 후 이를 분석하는 과정



## [암기 015] 자료 흐름도(DFD; Data Flow Diagram)

- 요구사항 분석에서 자료의 흐름 및 변환 과정과 기능을 도형 중심으로 기술하는 방법으로 자료 흐름 그래프, 버블 차트라고도 한다.
- 자료 흐름도 구성 요소 표기법

프로세스 (Process)	<ul style="list-style-type: none"> <li>자료를 변환시키는 시스템의 한 부분(처리 과정)을 나타내며 처리, 기능, 변환, 버블이라고함</li> <li>원이나 둥근 사각형으로 표시하고 그 안에 프로세스 이름을 기입함</li> </ul>
자료 흐름 (Data Flow)	<ul style="list-style-type: none"> <li>자료의 이동(흐름)이나 연관관계를 나타냄</li> <li>화살표 위에 자료의 이름을 기입함</li> </ul>
자료 저장소 (Data Store)	<ul style="list-style-type: none"> <li>시스템에서의 자료 저장소(파일, 데이터베이스)를 나타냄</li> <li>도형 안에 자료 저장소 이름을 기입함</li> </ul>
단말 (Terminator)	<ul style="list-style-type: none"> <li>시스템과 교신하는 외부 개체로, 입력 데이터가 만들어지고 출력 데이터를 받음(정보의 생산자와 소비자)</li> <li>도형 안에 이름을 기입함</li> </ul>

## [암기 016] 요구사항 확인 기법

- 요구사항 검토(Requirement Reviews) : 문서화된 요구사항을 훑어보면서 확인하는 것으로 가장 일반적인 요구사항 검증 방법
- 프로토타이핑(Prototyping)
  - 초기 도출된 요구사항을 토대로 프로토타입(Prototype)을 만든 후 대상 시스템의 개발이 진행되는 동안 도출되는 요구사항을 반영하면서 지속적으로 프로토타입을 재작성하는 과정
  - 프로토타입 : 상품이나 서비스가 출시되기 전에 개발 대상 시스템 또는 그 일부분을 개략적으로 만든 원형
- 모델 검증(Model Verification) : 요구사항 분석 단계에서 개발된 모델이 요구사항을 충족시키는지 검증하는 것
- 인수 테스트(Acceptance Test) : 사용자가 실제로 사용될 환경에서 요구사항들이 모두 충족되는지 사용자 입장에서 확인하는 과정

## [암기 017] UML의 구성 요소 – 사물(Things)

- 모델을 구성하는 가장 중요한 기본 요소로, 다이어그램 안에서 관계가 형성될 수 있는 대상들이다.
- 구조(Structural) 사물 : 시스템의 개념적, 물리적 요소를 표현
- 행동(Behavioral) 사물 : 시간과 공간에 따른 요소들의 행위를 표현
- 그룹(Grouping) 사물 : 요소들을 그룹으로 묶어서 표현
- 주해(Annotation) 사물 : 부가적인 설명이나 제약조건 등을 표현

## [암기 018] UML의 구성 요소 – 관계(Relationships)

- 사물과 사물 사이의 연관성을 표현하는 것이다.
- 연관(Association) 관계 : 2개 이상의 사물이 서로 관련되어 있음을 표현함
- 집합(Aggregation) 관계 : 하나의 사물이 다른 사물에 포함되어 있는 관계를 표현함
- 포함(Composition) 관계 : 집합 관계의 특수한 형태로, 포함하는 사물의 변화가 포함되는 사물에게 영향을 미치는 관계를 표현함
- 일반화(Generalization) 관계 : 하나의 사물이 다른 사물에 비해 더 일반적인지 구체적인지를 표현함
- 의존(Dependency) 관계 : 연관 관계와 같이 사물 사이에 서로 연관은 있으나 필요에 의해 서로에게 영향을 주는 짧은 시간 동안만 연관을 유지하는 관계를 표현함
- 실체화(Realization) 관계 : 사물이 할 수 있거나 해야 하는 기능(행위, 인터페이스)으로 서로를 그룹화 할 수 있는 관계를 표현함

## [암기 019] 클래스(Class) 다이어그램

- 클래스와 클래스가 가지는 속성, 클래스 사이의 관계를 표현한다.
- UML을 이용한 정적 모델링의 대표적인 것이 클래스 다이어그램이다.

## • 클래스 다이어그램의 구성 요소

클래스(Class)	<ul style="list-style-type: none"> <li>• 각각의 객체들이 갖는 속성과 오퍼레이션(동작)을 표현</li> <li>• 일반적으로 3개의 구획(Compartment)으로 나눠 클래스의 이름, 속성, 오퍼레이션을 표기함</li> <li>• 속성(Attribute) : 클래스의 상태나 정보를 표현</li> <li>• 오퍼레이션(Operation, 연산) : 클래스가 수행할 수 있는 동작으로, 함수(Method)라고도 함</li> </ul>
제약조건	속성에 입력될 값에 대한 제약조건이나 오퍼레이션 수행 전후에 지정해야 할 조건이 있다면 이를 적음
관계(Relationships)	클래스와 클래스 사이의 연관성을 표현

## [암기 020] 시퀀스 다이어그램

- 시스템이나 객체들이 메시지를 주고받으며 시간의 흐름에 따라 상호 작용하는 과정을 그림으로 표현한 것이다.
- 시퀀스 다이어그램의 구성 요소 : 액터, 객체, 라이프라인, 활성 상자, 메시지, 객체 소멸, 프레임 등

라이프라인(Lifeline)	객체가 메모리에 존재하는 기간으로, 객체 아래쪽에 점선을 그어 표현
활성 상자(Activation Box)	객체가 메시지를 주고받으며 구동되고 있음을 라이프라인 상에 겹쳐 직사각형 형태로 표현

## [암기 021] 커뮤니케이션(Communication) 다이어그램

- 시퀀스 다이어그램과 같이 동작에 참여하는 객체들이 주고받는 메시지를 표현하는데, 메시지뿐만 아니라 객체들 간의 연관까지 표현한다.
- 커뮤니케이션 다이어그램의 구성 요소 : 액터, 객체, 링크, 메시지 등

링크(Link)	<ul style="list-style-type: none"> <li>• 객체들 간의 관계를 표현하는 데 사용</li> <li>• 액터와 객체, 객체와 객체 간에 실선을 그어 표현</li> </ul>
----------	---

## [암기 022] 상태(State) 다이어그램

- 하나의 객체가 자신이 속한 클래스의 상태 변화 혹은 다른 객체와의 상호 작용에 따라 상태가 어떻게 변화하는지를 표현한다.
- 상태 다이어그램의 구성 요소 : 상태, 이벤트, 상태 전환 등

상태(State)	<ul style="list-style-type: none"> <li>• 객체의 상태를 등근 사간형 안에 기술</li> <li>• 시작 상태 : 상태의 시작을 속이 채워진 원(●)으로 표현</li> <li>• 종료 상태 : 상태의 종료를 속이 채워진 원을 둘러싼 원(◎)으로 표현</li> </ul>
이벤트(Event)	조건, 외부 신호, 시간의 흐름 등 상태에 변화를 주는 현상



## [암기 023] 유스케이스 다이어그램

- 개발될 시스템과 관련된 외부 요소들, 즉 사용자와 다른 외부 시스템들이 개발될 시스템을 이용해 수행할 수 있는 기능을 사용자의 관점(View)에서 표현한 것이다.
- 유스케이스 다이어그램의 구성 요소 : 시스템 범위, 액터, 유스케이스, 관계

시스템 범위(System Scope)	시스템 내부에서 수행되는 기능들을 외부 시스템과 구분하기 위해 시스템 내부의 유스케이스들을 사각형으로 묶어 시스템의 범위를 표현
액터(Actor)	시스템과 상호작용을 하는 모든 외부 요소로, 사람이나 외부 시스템을 의미
유스케이스(Use Case)	사용자가 보는 관점에서 시스템이 액터에게 제공하는 서비스 또는 기능을 표현한 것





## [암기 024] 활동 다이어그램

- 자료 흐름도와 유사한 것으로, 사용자의 관점(View)에서 시스템이 수행하는 기능을 처리 흐름에 따라 순서대로 표현한 것이다.
- 활동 다이어그램의 구성 요소 : 액션, 액티비티, 노드, 스윔레인 등

액션(Action)/ 액티비티(Activity)	<ul style="list-style-type: none"> <li>• 액션(Action)은 더 이상 분해할 수 없는 단일 작업</li> <li>• 액티비티(Activity)는 몇 개의 액션으로 분리될 수 있는 작업</li> </ul>
노드	<ul style="list-style-type: none"> <li>• 시작 노드 : 액션이나 액티비티가 시작됨을 의미</li> <li>• 종료 노드 : 액티비티 안의 모든 흐름이 종료됨을 의미</li> <li>• 조건(판단) 노드 : 조건에 따라 제어의 흐름이 분리됨을 표현</li> <li>• 병합 노드 : 여러 경로의 흐름이 하나로 합쳐짐을 표현</li> <li>• 포크(Fork) 노드 : 액티비티의 흐름이 분리되어 수행됨을 표현</li> <li>• 조인(Join) 노드 : 분리되어 수행되던 액티비티의 흐름이 다시 합쳐짐을 표현</li> </ul>
스윔레인 (Swim Lane)	액티비티 수행을 담당하는 주체를 구분

## [암기 026] 데이터 모델의 종류

- 개념적 데이터 모델 : 현실 세계에 대한 인간의 이해를 돕기 위해 현실 세계에 대한 인식을 추상적 개념으로 표현하는 과정
- 논리적 데이터 모델 : 개념적 모델링 과정에서 얻은 개념적 구조를 컴퓨터가 이해하고 처리할 수 있는 컴퓨터 세계의 환경에 맞도록 변환하는 과정
- 물리적 데이터 모델 : 실제 컴퓨터에 데이터가 저장되는 방법을 정의하는 물리 데이터베이스 설계 과정

## [암기 027] 데이터 모델에 표시할 요소

- 구조(Structure) : 논리적으로 표현된 개체 타입들 간의 관계로서 데이터 구조 및 정적 성질을 표현함
- 연산(Operation) : 데이터베이스에 저장된 실제 데이터를 처리하는 작업에 대한 명세로서 데이터베이스를 조작하는 기본 도구
- 제약 조건(Constraint) : 데이터베이스에 저장될 수 있는 실제 데이터의 논리적인 제약 조건

## 3장 | 데이터 입 · 출력 구현

### [암기 025] 데이터 모델

- 현실 세계의 정보들을 컴퓨터에 표현하기 위해서 단순화, 추상화하여 체계적으로 표현한 개념적 모형이다.
- 데이터 모델의 구성 요소
  - 개체(Entity) : 데이터베이스에 표현하려는 것으로, 사람이 생각하는 개념이나 정보 단위 같은 현실 세계의 대상체
  - 속성(Attribute) : 데이터의 가장 작은 논리적 단위로서 파일 구조상의 데이터 항목 또는 데이터 필드에 해당함
  - 관계(Relationship) : 개체 간의 관계 또는 속성 간의 논리적인 연결을 의미함

### [암기 028] 이상(Anomaly)

- 테이블에서 일부 속성들의 중복으로 인해 데이터의 중복(Redundancy)이 발생하고, 이 중복으로 인해 테이블 조작 시 문제가 발생하는 현상을 의미한다.
- 이상의 종류
  - 삽입 이상(Insertion Anomaly) : 테이블에 데이터를 삽입할 때 의도와는 상관없이 원하지 않은 값들로 인해 삽입할 수 없게 되는 현상
  - 삭제 이상(Deletion Anomaly) : 테이블에서 한 튜플을 삭제할 때 의도와는 상관없는 값들도 함께 삭제되는, 즉 연쇄 삭제가 발생하는 현상
  - 갱신 이상(Update Anomaly) : 테이블에서 튜플에 있는 속성 값을 갱신할 때 일부 튜플의 정보만 갱신되어 정보에 불일치성(Inconsistency)이 생기는 현상



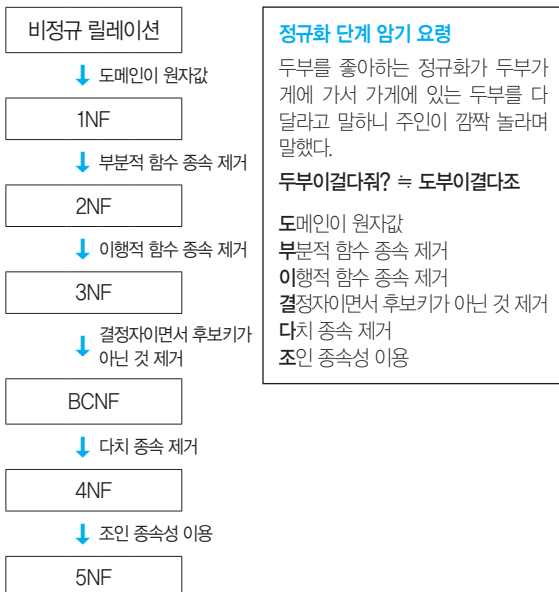


## [암기 029] 함수적 종속(Functional Dependency)

- 어떤 테이블 R에서 X와 Y를 각각 R의 속성 집합의 부분 집합이라 할 때 속성 X의 값 각각에 대해 시간에 관계없이 항상 속성 Y의 값이 오직 하나만 연관되어 있을 때 Y는 X에 함수적 종속 또는 X가 Y를 함수적으로 결정한다고 하고,  $X \rightarrow Y$ 로 표기한다.
- $X \rightarrow Y$ 의 관계를 갖는 속성 X와 Y에서 X를 결정자(Determinant)라 하고, Y를 종속자(Dependent)라고 한다.
- 완전 함수적 종속 : 어떤 테이블 R에서 속성 Y가 다른 속성 집합 X 전체에 대해 함수적 종속이면서 속성 집합 X의 어떠한 진부분 집합 Z(즉,  $Z \subset X$ )에도 함수적 종속이 아닐 때 속성 Y는 속성 집합 X에 완전 함수적 종속이라고 함
- 부분 함수적 종속 : 어떤 테이블 R에서 속성 Y가 다른 속성 집합 X 전체에 대해 함수적 종속이면서 속성 집합 X의 임의의 진부분 집합에 대해 함수적 종속일 때, 속성 Y는 속성 집합 X에 부분 함수적 종속이라고 함

## [암기 030] 정규화(Normalization)

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



※ 이행적 함수적 종속 :  $A \rightarrow B$ 이고  $B \rightarrow C$ 일 때  $A \rightarrow C$ 를 만족하는 관계를 이행적 함수적 종속이라고 함

## [암기 031] 반정규화(Denormalization)

- 시스템의 성능 향상, 개발 및 운영의 편의성 등을 위해 정규화된 데이터 모델을 통합, 중복, 분리하는 과정으로, 의도적으로 정규화 원칙을 위배하는 행위이다.
- 반정규화 방법

테이블 통합	두 개의 테이블이 조인(Join)되는 경우가 많아 하나의 테이블로 합쳐 사용하는 것이 성능 향상에 도움이 될 경우 수행함
테이블 분할	<ul style="list-style-type: none"> <li>수평 분할(Horizontal Partitioning) : 레코드(Record)를 기준으로 테이블을 분할하는 것으로, 레코드별로 사용 빈도의 차이가 큰 경우 사용 빈도에 따라 테이블을 분할함</li> <li>수직 분할(Vertical Partitioning) : 하나의 테이블에 속성이 너무 많을 경우 속성을 기준으로 테이블을 분할하는 것</li> </ul>
중복 테이블 추가	<ul style="list-style-type: none"> <li>여러 테이블에서 데이터를 추출해서 사용해야 하거나 다른 서버에 저장된 테이블을 이용해야 하는 경우 중복 테이블을 추가하여 작업의 효율성을 향상시킬 수 있음</li> <li>추가 방법 : 집계 테이블의 추가, 진행 테이블의 추가, 특정 부분만을 포함하는 테이블의 추가</li> </ul>
중복 속성 추가	조인해서 데이터를 처리할 때 데이터를 조회하는 경로를 단축하기 위해 자주 사용하는 속성을 하나 더 추가하는 것

## [암기 032] 인덱스(Index)

- 데이터 레코드를 빠르게 접근하기 위해 <키 값, 포인터> 쌍으로 구성되는 데이터 구조이다.
- 인덱스 키의 순서에 따라 데이터가 정렬되어 저장되는 방식인 클러스터드 인덱스(Clustered Index)와 인덱스의 키 값만 정렬되어 있을 뿐 실제 데이터는 정렬되지 않는 방식인 넌클러스터드 인덱스(Non-Clustered Index)가 있다.

### [암기 033] 뷰(View)

- 사용자에게 접근이 허용된 자료만을 제한적으로 보여 주기 위해 하나 이상의 기본 테이블로부터 유도된, 이름을 가지는 가상 테이블이다.
- 저장장치 내에 물리적으로 존재하지 않지만, 사용자에게는 있는 것처럼 간주된다.
- CREATE문으로 정의하고, DROP문으로 제거한다.
- 데이터의 논리적 독립성을 제공한다.

### [암기 034] 트랜잭션(Transaction)

- 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 모두 수행되어야 할 일련의 연산들을 의미한다.
- 특징

Atomicity (원자성)	트랜잭션의 연산은 데이터베이스에 모두 반영 되도록 완료(Commit)되든지 아니면 전혀 반영 되지 않도록 복구(Rollback)되어야 함
Consistency (일관성)	트랜잭션이 그 실행을 성공적으로 완료하면 언제나 일관성 있는 데이터베이스 상태로 변환함
Isolation (독립성)	둘 이상의 트랜잭션이 동시에 병행 실행되는 경우 어느 하나의 트랜잭션 실행중에 다른 트랜잭션의 연산이 끼어들 수 없음
Durability (지속성)	성공적으로 완료된 트랜잭션의 결과는 시스템이 고장나더라도 영구적으로 반영되어야 함

### [암기 035] 클러스터(Cluster)

- 데이터 저장 시 데이터 액세스 효율을 향상시키기 위해 동일한 성격의 데이터를 동일한 데이터 블록에 저장하는 물리적 저장 방법이다.
- 클러스터링키로 지정된 컬럼 값의 순서대로 저장되고, 여러 개의 테이블이 하나의 클러스터에 저장된다.
- 클러스터링 된 테이블은 데이터 조회 속도는 향상시키지만 데이터 입력, 수정, 삭제에 대한 성능은 저하시킨다.
- 처리 범위가 넓은 경우에는 단일 테이블 클러스터링을, 조인이 많이 발생하는 경우에는 다중 테이블 클러스터링을 사용한다.

### [암기 036] 파티션(Partition)

- 대용량의 테이블이나 인덱스를 작은 논리적 단위로 나누는 것을 말한다.
- 파티셔닝 방식에 따른 파티션의 종류

범위 분할 (Range Partitioning)	지정한 열의 값을 기준으로 분할
해시 분할 (Hash Partitioning)	해시 함수를 적용한 결과 값에 따라 데이터를 분할
조합 분할 (Composite Partitioning)	범위 분할로 분할한 다음 해시 함수를 적용하여 다시 분할

- 인덱스 파티션 : 파티션된 테이블의 데이터를 관리하기 위해 인덱스를 나누는 것

## 4장 | 통합 구현

### [암기 037] 연계 메커니즘

- 연계 메커니즘은 데이터의 생성 및 전송을 담당하는 송신 체계와 데이터 수신 및 운영 DB 반영을 담당하는 수신 체계로 구성된다.
- 송신 시스템은 운영 DB로부터 인터페이스 테이블이나 파일(xml, text, csv 등) 형식으로 연계 데이터를 생성하여 송신한다.
- 수신 시스템은 송신 시스템으로부터 전송된 데이터를 받아 수신 시스템에 맞는 데이터로 변환한 후 운영 DB에 반영한다.
- 송·수신 시스템 사이에는 데이터의 송·수신과 송·수신 시스템 현황을 모니터링하는 중계 시스템을 설치할 수 있다.
- 송·수신 시스템과 중계 시스템은 제각기 역할이 중복되지 않도록 아키텍처를 설계한 후 인터페이스 테스트와 통합 테스트를 통해 기능을 검증한다.
- 연계 메커니즘의 연계 방식에는 직접 연계 방식과 간접 연계 방식이 있다.

## [암기 038] 연계 메커니즘의 직접 연계 방식 종류

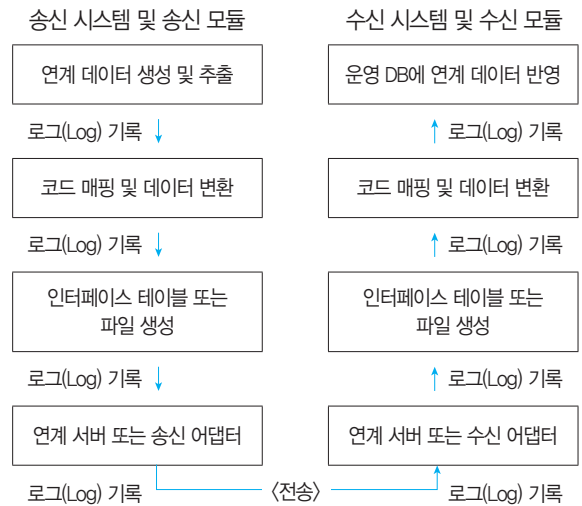
DB Link	DB에서 제공하는 DB Link 객체를 이용하는 방식
API (Application Programming Interface)/ Open API	데이터를 송신 시스템의 DB에서 읽어와 제공하는 애플리케이션 프로그래밍 인터페이스 ※ API(Application Programming Interface) : 운영 체제나 프로그래밍 언어 등에 있는 라이브러리를 응용프로그램 개발 시 이용할 수 있도록 규칙 등에 대해 정의해 놓은 인터페이스 ※ Open API : API의 기능을 누구나 무료로 사용하여 프로그램을 개발하거나 Open API에 새로운 API를 추가할 수 있도록 공개된 API
DB Connection	수신 시스템의 WAS(웹 애플리케이션 서버)에서 송신 시스템의 DB로 연결해주는 방식
JDBC(Java DataBase Connectivity)	Java에서 DB에 접근하여 데이터를 삽입, 삭제, 수정, 조회할 수 있도록 Java와 DB를 연결해주는 방식



## [암기 039] 연계 메커니즘의 간접 연계 방식 종류

연계 솔루션	EAI 서버와 송·수신 시스템에 설치되는 클라이언트(Client)를 이용하는 방식
ESB (Enterprise Service Bus)	애플리케이션 간 연계, 데이터 변환, 웹 서비스 지원 등 표준 기반의 인터페이스를 제공하는 방식
Socket	서버는 통신을 위한 소켓(Socket)을 생성하여 포트를 할당하고 클라이언트의 통신 요청 시 클라이언트와 연결하여 통신하는 네트워크 기술
Web Service	웹 서비스(Web Service)에서 WSDL과 UDDI, SOAP 프로토콜을 이용하여 연계하는 방식

## [암기 040] 연계 메커니즘의 구성



## [암기 041] 연계 모듈의 구현 환경

EAI (Enterprise Application Integration)	<ul style="list-style-type: none"> <li>기업 내 각종 애플리케이션 및 플랫폼 간의 정보 전달, 연계, 통합 등 상호 연동이 가능하게 해주는 솔루션</li> <li>EAI의 구축 유형                         <ul style="list-style-type: none"> <li>Point-to-Point : 가장 기본적인 애플리케이션 통합 방식으로, 애플리케이션을 1:1로 연결</li> <li>Hub &amp; Spoke : 단일 접점인 허브 시스템을 통해 데이터를 전송하는 중앙 집중형 방식</li> <li>Message Bus(ESB 방식) : 애플리케이션 사이에 미들웨어를 두어 처리하는 방식</li> <li>Bus Hybrid : Hub &amp; Spoke와 Message Bus의 혼합 방식</li> </ul> </li> </ul>
ESB (Enterprise Service Bus)	<ul style="list-style-type: none"> <li>애플리케이션 간 연계, 데이터 변환, 웹 서비스 지원 등 표준 기반의 인터페이스를 제공하는 솔루션</li> <li>ESB는 애플리케이션 통합 측면에서 EAI와 유사하지만 애플리케이션 보다는 서비스 중심의 통합을 지향함</li> </ul>

웹 서비스  
(Web  
Service)

- 네트워크의 정보를 표준화된 서비스 형태로 만들어 공유하는 기술로, 서비스 지향 아키텍처(SOA) 개념을 실현하는 대표적인 기술
- 웹 서비스의 구성
  - SOAP(Simple Object Access Protocol) : HTTP, HTTPS, SMTP 등을 활용하여 XML 기반의 메시지를 네트워크 상에서 교환하는 프로토콜
  - UDDI(Universal Description, Discovery and Integration) : WSDL을 등록하여 서비스와 서비스 제공자를 검색하고 접근하는데 사용됨
  - WSDL(Web Services Description Language) : 웹 서비스명, 서비스 제공 위치, 프로토콜 등 웹 서비스에 대한 상세 정보를 XML 형식으로 구현



[암기 042] XML(eXtensible Markup Language)

- 웹브라우저 간 HTML 문법이 호환되지 않는 문제와 SGML의 복잡함을 해결하기 위하여 개발된 다목적 마크업 언어이다.
- 유니코드를 기반으로 다국어를 지원한다.
- 사용자가 직접 문서의 태그를 정의할 수 있으며, 다른 사용자가 정의한 태그를 사용할 수 있다.
- XML의 구성
  - XML의 첫 문단

```
<?xml version="버전" encoding="언어셋" standalone="yes/no"?>
```

- XML 요소(Element)의 구성

```
<요소이름 속성1="속성값1" 속성2="속성값2"... > 내용 </요소이름>
```

[암기 043] JSON(JavaScript Object Notation)

- 속성-값 쌍(Attribute-Value Pairs)으로 이루어진 데이터 객체를 전달하기 위해 사람이 읽을 수 있는 텍스트를 사용하는 개방형 표준 포맷이다.
- 비동기 처리에 사용되는 AJAX에서 XML을 대체하여 사용되고 있다.

5장 | 서버 프로그램 구현

[암기 044] 개발 환경 구축 - 하드웨어 환경

- 개발 환경 구축은 응용 소프트웨어 개발을 위해 개발 프로젝트를 이해하고 소프트웨어 및 하드웨어 장비를 구축하는 것을 의미한다.
- 하드웨어 환경은 사용자와의 인터페이스 역할을 하는 클라이언트(Client) 그리고 클라이언트와 통신하여 서비스를 제공하는 서버(Server)로 구성된다.
- 서버는 사용 목적에 따라 웹 서버, 웹 애플리케이션 서버, 데이터베이스 서버, 파일 서버 등으로 나뉜다.

웹 서버 (Web Server)	클라이언트로부터 직접 요청을 받아 처리하는 서버로, 저장량의 정적 파일들을 제공
웹 애플리케이션 서버(WAS; Web Application Server)	사용자에게 동적 서비스를 제공하기 위해 웹 서버로부터 요청을 받아 데이터 가공 작업을 수행하거나, 웹 서버와 데이터베이스 서버 또는 웹 서버와 파일 서버 사이에서 인터페이스 역할을 수행하는 서버
데이터베이스 서버(DB Server)	데이터베이스와 이를 관리하는 DBMS를 운영하는 서버
파일 서버 (File Server)	데이터베이스에 저장하기에는 비효율적이거나, 서비스 제공을 목적으로 유지하는 파일들을 저장하는 서버

## [암기 045] 모듈(Module)

- 모듈화를 통해 분리된 시스템의 각 기능들로, 서브루틴, 서브시스템, 소프트웨어 내의 프로그램, 작업 단위 등과 같은 의미로 사용된다.
  - 모듈화 : 소프트웨어의 성능을 향상시키거나 시스템의 수정 및 재사용, 유지 관리 등이 용이하도록 시스템의 기능들을 모듈 단위로 분해하는 것으로 모듈 간 결합도의 최소화, 응집도의 최대화가 목표임
- 모듈의 기능적 독립성은 소프트웨어를 구성하는 각 모듈의 기능이 서로 독립됨을 의미하는 것으로, 모듈이 하나의 기능만을 수행하고 다른 모듈과의 과도한 상호작용을 배제함으로써 이루어진다.
- 독립성이 높은 모듈일수록 모듈을 수정하더라도 다른 모듈들에게는 거의 영향을 미치지 않으며, 오류가 발생해도 쉽게 발견하고 해결할 수 있다.
- 모듈의 독립성은 결합도(Coupling)와 응집도(Cohesion)에 의해 측정되며, 독립성을 높이려면 모듈의 결합도는 약하게, 응집도는 강하게, 모듈의 크기는 작게 만들어야 한다.

## [암기 046] 결합도(Coupling)

- 모듈 간에 상호 의존하는 정도 또는 두 모듈 사이의 연관 관계를 의미한다.
- 다양한 결합으로 모듈을 구성할 수 있으나 결합도가 약할수록 품질이 높고, 강할수록 품질이 낮다.
- 결합도가 강하면 시스템 구현 및 유지보수 작업이 어렵다.

자료 결합도 (Data Coupling)	<ul style="list-style-type: none"> <li>• 모듈 간의 인터페이스가 자료 요소로만 구성될 때의 결합도</li> <li>• 어떤 모듈이 다른 모듈을 호출하면서 매개 변수나 인수로 데이터를 넘겨주고, 호출 받은 모듈은 받은 데이터에 대한 처리 결과를 다시 돌려주는 방식</li> </ul>
스탬프(검인) 결합도 (Stamp Coupling)	모듈 간의 인터페이스로 배열이나 레코드 등의 자료 구조가 전달될 때의 결합도
제어 결합도 (Control Coupling)	어떤 모듈이 다른 모듈 내부의 논리적인 흐름을 제어하기 위해 제어 신호를 이용하여 통신하거나 제어 요소(Function Code, Switch, Tag, Flag)를 전달하는 결합도

외부 결합도 (External Coupling)	어떤 모듈에서 선언한 데이터(변수)를 외부의 다른 모듈에서 참조할 때의 결합도
공통(공유) 결합도 (Common Coupling)	공유되는 공통 데이터 영역을 여러 모듈이 사용할 때의 결합도
내용 결합도 (Content Coupling)	한 모듈이 다른 모듈의 내부 기능 및 그 내부 자료를 직접 참조하거나 수정할 때의 결합도

## [암기 047] 응집도(Cohesion)

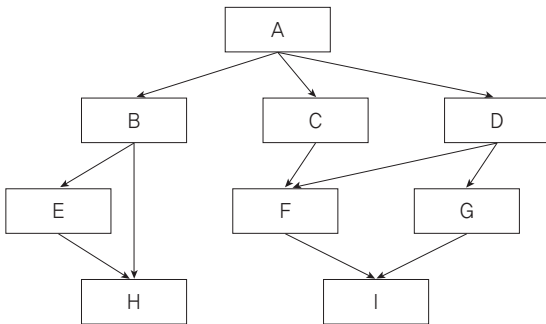
- 정보 은닉 개념을 확장한 것으로, 명령어나 호출문 등 모듈의 내부 요소들의 서로 관련되어 있는 정도, 즉 모듈이 독립적인 기능으로 정의되어 있는 정도를 의미한다.
- 다양한 기준으로 모듈을 구성할 수 있으나 응집도가 강할수록 품질이 높고, 약할수록 품질이 낮다.

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

## [암기 048] 팬인(Fan-In) / 팬아웃(Fan-Out)

- 팬인은 어떤 모듈을 제어(호출)하는 모듈의 수를 나타낸다.
- 팬아웃은 어떤 모듈에 의해 제어(호출)되는 모듈의 수를 나타낸다.
- 팬인과 팬아웃을 분석하여 시스템의 복잡도를 알 수 있다.
- 팬인이 높다는 것은 재사용 측면에서 설계가 잘 되어있다고 볼 수 있으나, 단일 장애점이 발생할 수 있으므로 중점적인 관리 및 테스트가 필요하다.

**예제** 다음의 시스템 구조도에서 각 모듈의 팬인(Fan-In)과 팬아웃(Fan-Out)을 구하시오.



### 해설

- 팬인(Fan-In) : A는 0, B · C · D · E · G는 1, F · H · I는 2
- 팬아웃(Fan-Out) : H · I는 0, C · E · F · G는 1, B · D는 2, A는 3

## [암기 049] 공통 모듈의 명세 기법

정확성 (Correctness)	시스템 구현 시 해당 기능이 필요하다는 것을 알 수 있도록 정확히 작성
명확성 (Clarity)	해당 기능을 이해할 때 중의적으로 해석되지 않도록 명확하게 작성
완전성 (Completeness)	시스템 구현을 위해 필요한 모든 것을 기술
일관성 (Consistency)	공통 기능들 간 상호 충돌이 발생하지 않도록 작성
추적성 (Traceability)	기능에 대한 요구사항의 출처, 관련 시스템 등의 관계를 파악할 수 있도록 작성

## [암기 050] DBMS 접속(Connection) 기술

JDBC(Java DataBase Connectivity)	Java 언어로 다양한 종류의 데이터베이스에 접속하고 SQL문을 수행할 때 사용되는 표준 API
ODBC(Open DataBase Connectivity)	데이터베이스에 접근하기 위한 표준 개방형 API로, 개발 언어에 관계없이 사용할 수 있음
MyBatis	<ul style="list-style-type: none"> <li>• JDBC 코드를 단순화하여 사용할 수 있는 SQL Mapping 기반 오픈 소스 접속 프레임워크</li> <li>• SQL을 거의 그대로 사용할 수 있어 SQL 친화적인 국내 환경에 적합하여 많이 사용됨</li> </ul>

## [암기 051] 서버 개발 프레임워크

프레임워크(Framework)는 본래 ‘뼈대’, ‘골조’를 의미하는 용어로, 소프트웨어에서는 특정 기능을 수행하기 위해 필요한 클래스나 인터페이스 등을 모아둔 집합체를 가리킨다.

프레임워크	특징
spring	JAVA를 기반으로 만들어진 프레임워크로, 전자정부 표준 프레임워크의 기반 기술로 사용되고 있음
Node.js	JavaScript를 기반으로 만들어진 프레임워크로, 비동기 입 · 출력 처리와 이벤트 위주의 높은 처리 성능을 갖고 있어 실시간으로 입 · 출력이 빈번한 애플리케이션에 적합함
Django	Python을 기반으로 만들어진 프레임워크로, 컴포넌트의 재사용과 플러그인화를 강조하여 신속한 개발이 가능하도록 지원함
Codeigniter	PHP를 기반으로 만들어진 프레임워크로, 인터페이스가 간편하며 서버 자원을 적게 사용함
Ruby on Rails	Ruby를 기반으로 만들어진 프레임워크로, 테스트를 위한 웹 서버를 지원하며 데이터베이스 작업을 단순화, 자동화시켜 개발 코드의 길이가 짧아 신속한 개발이 가능함



## [암기 052] 프레임워크의 특징

모듈화 (Modularity)	프레임워크는 캡슐화를 통해 모듈화를 강화하고 설계 및 구현의 변경에 따른 영향을 최소화함으로써 소프트웨어의 품질을 향상시킴
재사용성 (Reusability)	프레임워크는 재사용 가능한 모듈들을 제공함으로써 개발자의 생산성을 향상시킴
확장성 (Extensibility)	프레임워크는 다형성(Polymorphism)을 통한 인터페이스 확장이 가능하여 다양한 형태와 기능을 가진 애플리케이션 개발이 가능함
제어의 역흐름 (Inversion of Control)	개발자가 관리하고 통제해야 하는 객체들의 제어 권한을 프레임워크에 넘김으로써 생산성을 향상시킴

## [암기 053] 서버 개발 과정

DTO (Data Transfer Object)/ VO(Value Object) 구현	<ul style="list-style-type: none"> <li>데이터 교환을 위해 사용할 객체를 만드는 과정</li> <li>변수 및 객체를 송·수신할 데이터의 자료형(Data Type)에 알맞게 생성함</li> <li>알고리즘 등의 로직은 구현하지 않고, 변수와 데이터를 저장하고 반환하는 메소드만 구현함</li> </ul>
SQL 구현	<ul style="list-style-type: none"> <li>데이터의 삽입, 변경, 삭제 등의 작업을 수행할 SQL문을 생성하는 과정</li> <li>SQL문은 소스 코드 내에 직접 입력하거나, 별도의 XML 파일로 저장하여 관리함</li> <li>XML 파일로 SQL문을 관리하는 경우 중복되는 SQL문을 최소화할 수 있고, 유지보수가 간편해짐</li> </ul>
DAO (Data Access Object) 구현	데이터베이스에 접근하고, SQL을 활용하여 데이터를 실제로 조작하는 코드를 구현하는 과정
Service 구현	사용자의 요청에 응답하기 위한 로직을 구현하는 과정
Controller 구현	사용자의 요청에 적절한 서비스를 호출하여, 그 결과를 사용자에게 반환하는 코드를 구현하는 과정

## [암기 054] 배치 프로그램(Batch Program)

- 사용자와의 상호 작용 없이 여러 작업들을 미리 정해진 일련의 순서에 따라 일괄적으로 처리하는 것을 의미한다.
- 주요 배치 스케줄러

스프링 배치 (Spring Batch)	<ul style="list-style-type: none"> <li>Spring Source사와 Accenture사가 2007년 공동 개발한 오픈소스 프레임워크</li> <li>로그 관리, 추적, 트랜잭션 관리, 작업 처리 통계, 작업 재시작 등의 다양한 기능을 제공함</li> <li>주요 구성 요소 : Job, Job Launcher, Step, Job Repository</li> </ul>
Quartz	<ul style="list-style-type: none"> <li>스프링 프레임워크로 개발되는 응용 프로그램들의 일괄 처리를 위한 다양한 기능을 제공하는 오픈 소스 라이브러리</li> <li>주요 구성 요소 : Scheduler, Job, JobDetail, Trigger</li> </ul>
Cron	리눅스의 스케줄러 도구로 crontab 명령어를 통해 작업을 예약할 수 있음

## 6장 | 화면 설계

### [암기 055] 사용자 인터페이스(UI; User Interface)

- 사용자와 시스템 간의 상호작용이 원활하게 이뤄지도록 도와주는 장치나 소프트웨어를 의미한다.
- 사용자 인터페이스의 구분

CLI(Command Line Interface)	명령과 출력이 텍스트 형태로 이뤄지는 인터페이스
GUI(Graphical User Interface)	아이콘이나 메뉴를 마우스로 선택하여 작업을 수행하는 그래픽 환경의 인터페이스
NUI(Natural User Interface)	사용자의 말이나 행동으로 기기를 조작하는 인터페이스



## [암기 056] 사용자 인터페이스의 기본 원칙

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

유스케이스 (Use Case)	<ul style="list-style-type: none"> <li>• 사용자 측면에서의 요구사항으로, 사용자가 원하는 목표를 달성하기 위해 수행할 내용을 기술함</li> <li>• 사용자의 요구사항을 빠르게 파악함으로써 프로젝트의 초기에 시스템의 기능적인 요구를 결정하고 그 결과를 문서화할 수 있음</li> </ul>
---------------------	--

## [암기 057] UI 설계 도구

와이어프레임 (Wireframe)	<ul style="list-style-type: none"> <li>• 기획 단계의 초기에 제작하는 것으로, 페이지에 대한 개략적인 레이아웃이나 UI 요소 등에 대한 뼈대를 설계하는 단계</li> <li>• 와이어프레임을 제작할 때는 각 페이지의 영역 구분, 콘텐츠, 텍스트 배치 등을 화면 단위로 설계함</li> </ul>
목업 (Mockup)	<ul style="list-style-type: none"> <li>• 디자인, 사용 방법 설명, 평가 등을 위해 와이어프레임보다 좀 더 실제 화면과 유사하게 만든 정적인 형태의 모형</li> <li>• 시각적으로만 구성 요소를 배치하는 것으로 실제로 구현되지는 않음</li> </ul>
스토리보드 (Story Board)	<ul style="list-style-type: none"> <li>• 와이어프레임에 콘텐츠에 대한 설명, 페이지 간 이동 흐름 등을 추가한 문서</li> <li>• 스토리보드의 상단이나 우측에는 제목, 작성자 등을 입력하고, 좌측에는 UI 화면, 우측에는 디스크립션(Description)을 기입함</li> </ul>
프로토타입 (Prototype)	<ul style="list-style-type: none"> <li>• 와이어프레임이나 스토리보드 등에 인터랙션을 적용함으로써 실제 구현된 것처럼 테스트가 가능한 동적인 형태의 모형</li> <li>• 프로토타입은 사용성 테스트나 작업자 간 서비스 이해를 위해 작성하는 샘플임</li> <li>• 페이퍼 프로토타입 : 아날로그적인 방법으로, 스케치, 그림, 글 등을 이용하여 손으로 직접 작성하는 방법</li> <li>• 디지털 프로토타입 : 파워포인트, 아크로벳, 비지오, 옴니그래프 등과 같은 프로그램을 사용하여 작성하는 방법</li> </ul>

## 7장 | 애플리케이션 테스트 관리

### [암기058] 애플리케이션 테스트의 기본 원리

- 완벽한 테스트 불가능 : 애플리케이션 테스트는 소프트웨어의 잠재적인 결함을 줄일 수 있지만 소프트웨어에 결함이 없다고 증명할 수는 없음. 즉 완벽한 소프트웨어 테스트는 불가능함
- 결함 집중(Defect Clustering) : 애플리케이션의 결함은 대부분 개발자의 특성이나 애플리케이션의 기능적 특징 때문에 특정 모듈에 집중되어 있으며, 애플리케이션의 20%에 해당하는 코드에서 전체 결함의 80%가 발견된다고 하여 파레토 법칙(Pareto Principle)을 적용하기도 함
- 살충제 패러독스(Pesticide Paradox) : 애플리케이션 테스트에서는 동일한 테스트 케이스로 동일한 테스트를 반복하면 더 이상 결함이 발견되지 않는 '살충제 패러독스(Pesticide Paradox) 현상이 발생하며, 살충제 패러독스를 방지하기 위해서 테스트 케이스를 지속적으로 보완 및 개선해야 함
- 테스트는 정황(Context) 의존 : 애플리케이션 테스트는 소프트웨어 특징, 테스트 환경, 테스터 역량 등 정황(Context)에 따라 테스트 결과가 달라질 수 있으므로, 정황에 따라 테스트를 다르게 수행해야 함
- 오류-부재의 궤변(Absence of Errors Fallacy) : 소프트웨어의 결함을 모두 제거해도 사용자의 요구사항을 만족시키지 못하면 해당 소프트웨어는 품질이 높다고 말할 수 없음을 의미함

## [암기059] 애플리케이션 테스트의 분류

### • 프로그램 실행 여부에 따른 테스트

정적 테스트	프로그램을 실행하지 않고 명세서나 소스 코드를 대상으로 분석하는 테스트
동적 테스트	프로그램을 실행하여 오류를 찾는 테스트로, 소프트웨어 개발의 모든 단계에서 테스트를 수행할 수 있음

### • 테스트 기반(Test Bases)에 따른 테스트

명세 기반 테스트	사용자의 요구사항에 대한 명세를 빠짐없이 테스트 케이스로 만들어 구현하고 있는지 확인하는 테스트
구조 기반 테스트	소프트웨어 내부의 논리 흐름에 따라 테스트 케이스를 작성하고 확인하는 테스트
경험 기반 테스트	유사 소프트웨어나 기술 등에 대한 테스트의 경험을 기반으로 수행하는 테스트

### • 시각에 따른 테스트

검증(Verification) 테스트	개발자의 시각에서 제품의 생산 과정을 테스트하는 것으로, 제품이 명세서대로 완성됐는지를 테스트함
확인(Validation) 테스트	사용자의 시각에서 생산된 제품의 결과를 테스트하는 것으로, 사용자가 요구한대로 제품이 완성됐는지, 제품이 정상적으로 동작하는지를 테스트함

## [암기060] 목적에 따른 애플리케이션 테스트의 분류

회복(Recovery) 테스트	시스템에 여러 가지 결함을 주어 실패하도록 한 후 올바르게 복구되는지를 확인하는 테스트
안전(Security) 테스트	시스템에 설치된 시스템 보호 도구가 불법적인 침입으로부터 시스템을 보호할 수 있는지를 확인하는 테스트
강도(Stress) 테스트	시스템에 과도한 정보량이나 빈도 등을 부과하여 과부하 시에도 소프트웨어가 정상적으로 실행되는지를 확인하는 테스트

성능(Performance) 테스트	소프트웨어의 실시간 성능이나 전체적인 효율성을 진단하는 테스트로, 소프트웨어의 응답 시간, 처리량 등을 테스트
구조(Structure) 테스트	소프트웨어 내부의 논리적인 경로, 소스 코드의 복잡도 등을 평가하는 테스트
회귀(Regression) 테스트	소프트웨어의 변경 또는 수정된 코드에 새로운 결함이 없음을 확인하는 테스트
병행(Parallel) 테스트	변경된 소프트웨어와 기존 소프트웨어에 동일한 데이터를 입력하여 결과를 비교하는 테스트

## [암기061] 화이트박스 테스트(White Box Test)

- 모듈의 원시 코드를 오픈시킨 상태에서 원시 코드의 논리적인 모든 경로를 테스트하여 테스트 케이스를 설계하는 방법이다.
- 원시 코드(Source Code)의 모든 문장을 한 번 이상 실행함으로써 수행된다.
- 모듈 안의 작동을 직접 관찰할 수 있다.
- 화이트박스 테스트의 종류

기초 경로 검사	<ul style="list-style-type: none"> <li>• 테스트 케이스 설계자가 절차적 설계의 논리적 복잡성을 측정할 수 있게 해주는 테스트 기법</li> <li>• 테스트 측정 결과는 실행 경로의 기초를 정의하는 데 지침으로 사용됨</li> </ul>
제어 구조 검사	<ul style="list-style-type: none"> <li>• 조건 검사(Condition Testing) : 프로그램 모듈 내에 있는 논리적 조건을 테스트하는 테스트 케이스 설계 기법</li> <li>• 루프 검사(Loop Testing) : 프로그램의 반복(Loop) 구조에 초점을 맞춰 실시하는 테스트 케이스 설계 기법</li> <li>• 데이터 흐름 검사(Data Flow Testing) : 프로그램에서 변수의 정의와 변수 사용의 위치에 초점을 맞춰 실시하는 테스트 케이스 설계 기법</li> </ul>

## [암기062] 블랙박스 테스트(Black Box Test)

- 소프트웨어가 수행할 특정 기능을 알기 위해서 각 기능이 완전히 작동되는 것을 입증하는 테스트로, 기능 테스트라고도 한다.
- 사용자의 요구사항 명세를 보면서 테스트하는 것으로, 주로 구현된 기능을 테스트한다.
- 소프트웨어 인터페이스에서 실시되는 테스트이다.
- 블랙박스 테스트의 종류

동치 분할 검사	입력 자료에 초점을 맞춰 테스트 케이스를 만들고 검사하는 기법
경계값 분석	입력 자료에만 치중한 동치 분할 기법을 보완하기 위한 기법으로, 입력 조건의 경계값을 테스트 케이스로 선정하여 검사함
원인 - 효과 그래프 검사	입력 데이터 간의 관계와 출력에 영향을 미치는 상황을 체계적으로 분석한 다음 효용성이 높은 테스트 케이스를 선정하여 검사하는 기법
오류 예측 검사	과거의 경험이나 확인자의 감각으로 테스트하는 기법
비교 검사	여러 버전의 프로그램에 동일한 테스트 자료를 제공하여 동일한 결과가 출력되는지 테스트하는 기법

## [암기063] 인수 테스트(Acceptance Test)

- 개발한 소프트웨어가 사용자의 요구사항을 충족하는지에 중점을 두고 테스트하는 것이다.
- 알파 테스트 : 개발자의 장소에서 사용자가 개발자 앞에서 행하는 테스트 기법으로, 테스트는 통제된 환경에서 행해지며, 오류와 사용상의 문제점을 사용자와 개발자가 함께 확인하면서 기록함
- 베타 테스트 : 선정된 최종 사용자가 여러 명의 사용자 앞에서 행하는 테스트 기법으로, 개발자에 의해 제어되지 않은 상태에서 테스트가 행해지며, 발견된 오류와 사용상의 문제점을 기록하고 개발자에게 주기적으로 보고함

## [암기064] 통합 테스트 - 점진적 통합 방식

- 모듈 단위로 단계적으로 통합하면서 테스트하는 방법이다.
- 오류 수정이 용이하고, 인터페이스와 연관된 오류를 완전히 테스트할 가능성이 높다.
- 하향식 통합 테스트(Top Down Integration Test) : 프로그램의 상위 모듈에서 하위 모듈 방향으로 통합하면서 테스트하는 기법
- 상향식 통합 테스트(Bottom Up Integration Test) : 프로그램의 하위 모듈에서 상위 모듈 방향으로 통합하면서 테스트하는 기법
- 혼합식 통합 테스트 : 하위 수준에서는 상향식 통합, 상위 수준에서는 하향식 통합을 사용하여 최적의 테스트를 지원하는 방식으로, 샌드위치(Sandwich)식 통합 테스트 방법이라고도 함



## [암기065] 테스트 시나리오(Test Scenario)

- 테스트 케이스를 적용하는 순서에 따라 여러 개의 테스트 케이스들을 묶은 집합으로, 테스트 케이스들을 적용하는 구체적인 절차를 명세한 문서이다.
- 테스트 시나리오에는 테스트 순서에 대한 구체적인 절차, 사전 조건, 입력 데이터 등이 설정되어 있다.
- 테스트 시나리오는 시스템별, 모듈별, 항목별 등과 같이 여러 개의 시나리오로 분리하여 작성해야 한다.
- 각각의 테스트 항목은 식별자 번호, 순서 번호, 테스트 데이터, 테스트 케이스, 예상 결과, 확인 등을 포함해서 작성해야 한다.
- 테스트 시나리오는 유스케이스(Use Case) 간 업무 흐름이 정상적인지를 테스트할 수 있도록 작성해야 한다.

## [암기066] 테스트 오라클(Test Oracle)

- 테스트 결과가 올바른지 판단하기 위해 사전에 정의된 참 값을 대입하여 비교하는 기법 및 활동이다.
- 참(True) 오라클 : 모든 테스트 케이스의 입력 값에 대해 기대하는 결과를 제공하는 오라클로, 발생한 모든 오류를 검출할 수 있음
- 샘플링(Sampling) 오라클 : 특정한 몇몇 테스트 케이스의 입력 값들에 대해서만 기대하는 결과를 제공하는 오라클
- 추정(Heuristic) 오라클 : 샘플링 오라클을 개선한 오라클로, 특정 테스트 케이스의 입력값에 대해 기대하는 결과를 제공하고, 나머지 입력 값들에 대해서는 추정으로 처리하는 오라클
- 일관성 검사(Consistent) 오라클 : 애플리케이션의 변경이 있을 때, 테스트 케이스의 수행 전과 후의 결과 값이 동일한지를 확인하는 오라클

## [암기067] 테스트 자동화 도구 유형

- 사람이 반복적으로 수행하던 테스트 절차를 테스트 자동화 도구를 사용함으로써 휴먼 에러(Human Error)를 줄이고 테스트의 정확성을 유지하면서 테스트의 품질을 향상시킬 수 있다.
- ※ 휴먼 에러(Human Error) : 사람의 판단 실수나 조작 실수 등으로 인해 발생하는 에러
- 정적 분석 도구(Static Analysis Tools) : 프로그램을 실행하지 않고 분석하는 도구로, 소스 코드에 대한 코딩 표준, 코딩 스타일, 코드 복잡도 및 남은 결함 등을 발견하기 위해 사용됨
- 테스트 실행 도구(Test Execution Tools) : 스크립트 언어를 사용하여 테스트를 실행하는 방법으로, 테스트 데이터와 테스트 수행 방법 등이 포함된 스크립트를 작성한 후 실행함
- 성능 테스트 도구(Performance Test Tools) : 애플리케이션의 처리량, 응답 시간, 경과 시간, 자원 사용률 등을 인위적으로 적용한 가상의 사용자를 만들어 테스트를 수행함으로써 성능의 목표 달성 여부를 확인함

- 테스트 통제 도구(Test Control Tools) : 테스트 계획 및 관리, 테스트 수행, 결함 관리 등을 수행하는 도구로, 종류에는 형상 관리 도구, 결함 추적/관리 도구 등이 있음
- 테스트 하네스 도구(Test Harness Tools) : 테스트가 실행될 환경을 시뮬레이션 하여 컴포넌트 및 모듈이 정상적으로 테스트 되도록 하는 도구



## [암기068] 결함 관리(Fault)

- 결함은 오류 발생, 작동 실패 등과 같이 소프트웨어가 개발자가 설계한 것과 다르게 동작하거나 다른 결과가 발생하는 것을 의미한다.
- 결함 관리 프로세스 처리 순서 : 결함 관리 계획 → 결함 기록 → 결함 검토 → 결함 수정 → 결함 재확인 → 결함 상태 추적 및 모니터링 활동 → 최종 결함 분석 및 보고서 작성
- 결함 관리 측정 지표

결함 분포	모듈 또는 컴포넌트의 특정 속성에 해당하는 결함 수 측정
결함 추세	테스트 진행 시간에 따른 결함 수의 추이 분석
결함 에이징	특정 결함 상태로 지속되는 시간 측정

- 결함 추적 순서 : 결함 등록(Open) → 결함 검토(Reviewed) → 결함 할당(Assigned) → 결함 수정(Resolved) → 결함 종료(Closed) → 결함 해제(Clarified)
- ※ Fixed(고정) : 개발자가 필요한 변경 작업을 수행하여 결함 수정 작업을 완료한 상태

## [암기069] 애플리케이션 성능 측정 지표

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



## [암기070] 소스 코드 최적화

- 나쁜 코드(Bad Code)를 배제하고, 클린 코드(Clean Code)로 작성하는 것이다.
- 클린 코드(Clean Code) : 누구나 쉽게 이해하고 수정 및 추가할 수 있는 단순, 명료한 코드
- 나쁜 코드(Bad Code) : 코드의 로직이 서로 얽혀 있는 스파게티 코드 등 프로그램의 로직(Logic)이 복잡하고 이해하기 어려운 코드
- 나쁜 코드로 작성된 애플리케이션의 코드를 클린 코드로 수정하면 애플리케이션의 성능이 개선된다.
- 클린 코드 작성 원칙 : 가독성, 단순성, 의존성 배제, 중복성 최소화, 추상화
- 소스 코드 최적화 유형
  - 클래스 분할 배치 : 하나의 클래스는 하나의 역할만 수행하도록 응집도를 높이고, 크기를 작게 작성함
  - Loosely Coupled(느슨한 결합) : 인터페이스 클래스를 이용하여 추상화된 자료 구조와 메소드를 구현함으로써 클래스 간의 의존성을 최소화함
  - 코딩 형식 준수, 좋은 이름 사용, 적절한 주석문 사용

## 8장 | SQL 응용

### [암기071] DDL – CREATE TABLE

#### CREATE TABLE

- 테이블을 정의하는 명령문이다.
- 표기 형식

```
CREATE TABLE 테이블명
(속성명 데이터_타입 [DEFAULT 기본값] [NOT NULL], ...
[, PRIMARY KEY(기본키_속성명, ...)]
[, UNIQUE(대체키_속성명, ...)]
[, FOREIGN KEY(외래키_속성명, ...)
[REFERENCES 참조테이블(기본키_속성명, ...)
[ON DELETE 옵션]
[ON UPDATE 옵션]
[, CONSTRAINT 제약조건명] [CHECK (조건식)];
```

#### 다른 테이블을 이용한 테이블 정의

- 기존 테이블의 정보를 이용해 새로운 테이블을 정의할 수 있다.
- 표기 형식

```
CREATE TABLE 신규테이블명 AS SELECT 속성명[, 속성명, ...] FROM 기존테이블명;
```

### [암기072] DDL – CREATE VIEW

- 뷰(View)를 정의하는 명령문이다.
- 표기 형식

```
CREATE VIEW 뷰명[(속성명[, 속성명, ...])]
AS SELECT문;
```

### [암기073] DDL – ALTER TABLE

- 테이블에 대한 정의를 변경하는 명령문이다.
- 표기 형식

```
ALTER TABLE 테이블명 ADD 속성명 데이터_타입 [DEFAULT '기본값'];
ALTER TABLE 테이블명 ALTER | MODIFY 속성명 [SET DEFAULT '기본값'];
ALTER TABLE 테이블명 DROP COLUMN 속성명 [CASCADE];
```

- ADD : 새로운 속성(열)을 추가할 때 사용
- ALTER | MODIFY : 특정 속성의 정의를 변경할 때 사용
- DROP COLUMN : 특정 속성을 삭제할 때 사용

### [암기074] DDL – DROP TABLE

- 기본 테이블을 제거하는 명령문이다.
- 표기 형식

```
DROP TABLE 테이블명 [CASCADE | RESTRICT];
```

- CASCADE : 제거할 요소를 참조하는 다른 모든 개체를 함께 제거함. 즉 주 테이블의 데이터 제거 시 각 외래키와 관계를 맺고 있는 모든 데이터를 제거하는 참조 무결성 제약 조건을 설정하기 위해 사용됨
- RESTRICT : 다른 개체가 제거할 요소를 참조중일 때는 제거를 취소함

### [암기075] DCL – GRANT / REVOKE

- 데이터베이스 관리자가 데이터베이스 사용자에게 권한을 부여하거나 취소하기 위한 명령어이다.
- GRANT : 권한 부여를 위한 명령어
- REVOKE : 권한 취소를 위한 명령어

#### • 사용자등급 지정 및 해제

- GRANT 사용자등급 TO 사용자\_ID\_리스트 [IDENTIFIED BY 암호];
- REVOKE 사용자등급 FROM 사용자\_ID\_리스트;

#### • 테이블 및 속성에 대한 권한 부여 및 취소

- GRANT 권한\_리스트 ON 개체 TO 사용자 [WITH GRANT OPTION];
- REVOKE [GRANT OPTION FOR] 권한\_리스트 ON 개체 FROM 사용자 [CASCADE];

- 권한 종류 : ALL, SELECT, INSERT, DELETE, UPDATE, ALTER 등
- WITH GRANT OPTION : 부여받은 권한을 다른 사용자에게 다시 부여할 수 있는 권한을 부여함
- GRANT OPTION FOR : 다른 사용자에게 권한을 부여할 수 있는 권한을 취소함
- CASCADE : 권한 취소 시 권한을 부여받았던 사용자가 다른 사용자에게 부여한 권한도 연쇄적으로 취소함

### [암기076] COMMIT / ROLLBACK / SAVEPOINT

COMMIT	트랜잭션이 성공적으로 끝나면 데이터베이스가 새로운 일관성(Consistency) 상태를 가지기 위해 변경된 모든 내용을 데이터베이스에 반영하여야 하는데, 이때 사용하는 명령어
ROLLBACK	아직 COMMIT되지 않은 변경된 모든 내용들을 취소하고 데이터베이스를 이전 상태로 되돌리는 명령어
SAVEPOINT	트랜잭션 내에 ROLLBACK 할 위치인 저장점을 지정하는 명령어로, 저장점을 지정할 때는 이름을 부여하며, ROLLBACK 시 지정된 저장점까지의 트랜잭션 처리 내용이 취소됨



## [암기077] DML – 삽입, 삭제, 갱신문

- 삽입문(INSERT INTO ~) : 기본 테이블에 새로운 튜플을 삽입할 때 사용

```
INSERT INTO 테이블명(속성명1, 속성명2, ...)
VALUES (데이터1, 데이터2, ...);
```

- 삭제문(DELETE FROM ~) : 기본 테이블에 있는 튜플들 중에서 특정 튜플을 삭제할 때 사용

```
DELETE
FROM 테이블명
[WHERE 조건];
```

- 갱신문(UPDATE ~ SET ~) : 기본 테이블에 있는 튜플들 중에서 특정 튜플의 내용을 변경할 때 사용

```
UPDATE 테이블명
SET 속성명 = 데이터[, 속성명=데이터]
[WHERE 조건];
```

## [암기078] DML – SELECT

```
SELECT [PREDICATE] [테이블명].[속성명] [AS 별칭], [테이블명].[속성명, ...] [, 그룹함수(속성명) [AS 별칭]]
FROM 테이블명[, 테이블명, ...]
[WHERE 조건]
[GROUP BY 속성명, 속성명, ...]
[HAVING 조건]
[ORDER BY 속성명 [ASC | DESC]];
```

- SELECT절
  - PREDICATE : 불러올 튜플 수를 제한할 명령어를 기술함
    - ▶ DISTINCT : 중복된 튜플이 있으면 그 중 첫 번째 한 개만 검색함
  - 속성명 : 검색하여 불러올 속성(열) 및 수식들을 지정함
  - AS : 속성 및 연산의 이름을 다른 제목으로 표시하기 위해 사용함

- FROM절 : 질의에 의해 검색될 데이터들을 포함하는 테이블명을 기술함
- WHERE절 : 검색할 조건 기술함
- GROUP BY절 : 특정 속성을 기준으로 그룹화하여 검색할 때 그룹화 할 속성을 지정함
- HAVING절 : GROUP BY와 함께 사용되며, 그룹에 대한 조건을 지정함
- ORDER BY절 : 특정 속성을 기준으로 정렬하여 검색할 때 사용함
  - 속성명 : 정렬의 기준이 되는 속성명을 기술함
  - [ASC | DESC] : 정렬 방식으로 'ASC'는 오름차순, 'DESC'는 내림차순임. 생략하면 오름차순으로 지정됨

### 참관만요 ! SELEC문의 실행 작동 순서

FROM → WHERE → GROUP BY → HAVING → SELECT → DISTINCT → ORDER BY

## [암기079] 하위 질의

조건절에 주어진 질의를 먼저 수행하여 그 검색 결과를 조건절의 피연산자로 사용한다.

### 〈사원〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
임꺽정	인터넷	01/09/69	서교동	80
황진이	편집	07/21/75	합정동	100
김선달	편집	10/22/73	망원동	90
성춘향	기획	02/20/64	대흥동	100
장길산	편집	03/11/67	상암동	120
일지매	기획	04/29/78	연남동	110
강건달	인터넷	12/11/80		90

### 〈여가활동〉

이름	취미	경력
김선달	당구	10
성춘향	나이트댄스	5
일지매	태권	15
임꺽정	씨름	8

**예제** '취미'가 '나이트댄스'인 사원의 '이름'과 '주소'를 검색하시오.

```
SELECT 이름, 주소
FROM 사원
WHERE 이름 = (SELECT 이름 FROM 여가활동 WHERE 취미 = '나이트댄스');
```



- 먼저 "SELECT 이름 FROM 여가활동 WHERE 취미 = '나이트댄스'"를 수행하여 <여가활동> 테이블에서 '성춘향'을 찾는다. 그런 다음 하위 질의에 해당하는 피연산자의 자리에 '성춘향'을 대입하면 질의문은 "SELECT 이름, 주소 FROM 사원 WHERE 이름 = '성춘향'"과 같다.
- WHERE 절의 조건에 IN 연산자를 이용하여 다음과 같이 조건을 지정해도 된다.
- WHERE 이름 IN (SELECT 이름 FROM 여가활동 WHERE 취미 = '나이트댄스');

#### <결과>

이름	주소
성춘향	대흥동

### [암기080] DML - INNER JOIN

가장 일반적인 JOIN의 형태로, 관계가 설정된 두 테이블에서 조인된 필드가 일치하는 행만을 표시한다.

```
SELECT [테이블명1.]속성명, [테이블명2.]속성명, ...
FROM 테이블명1, 테이블명2, ...
WHERE 테이블명1.속성명 = 테이블명2.속성명;
```

#### <학생>

학번	이름	학과코드	선배	성적
15	고길동	com		83
16	이순신	han		96
17	김선달	com	15	95
19	아무개	han	16	75
37	박치민		17	55

#### <학과>

학과코드	학과명
com	컴퓨터
han	국어
eng	영어

**예제** <학생> 테이블과 <학과> 테이블에서 학과코드 값이 같은 튜플을 JOIN하여 학번, 이름, 학과코드, 학과명을 출력하는 SQL문을 작성하시오.

```
SELECT 학번, 이름, 학생.학과코드, 학과명
FROM 학생, 학과
WHERE 학생.학과코드 = 학과.학과코드;
```

#### <결과>

학번	이름	학과코드	학과명
15	고길동	com	컴퓨터
16	이순신	han	국어
17	김선달	com	컴퓨터
19	아무개	han	국어

### [암기081] DML - OUTER JOIN

- 릴레이션에서 JOIN 조건에 만족하지 않는 튜플도 결과로 출력하기 위한 JOIN 방법으로, LEFT OUTER JOIN, RIGHT OUTER JOIN 등이 있다.
- LEFT OUTER JOIN : INNER JOIN의 결과를 구한 후, 우측 항 릴레이션의 어떤 튜플과도 맞지 않는 좌측 항의 릴레이션에 있는 튜플들에 NULL 값을 붙여서 INNER JOIN의 결과에 추가함

- 표기 형식

```
SELECT [테이블명1.]속성명, [테이블명2.]속성명, ...
FROM 테이블명1 LEFT OUTER JOIN 테이블명2
ON 테이블명1.속성명 = 테이블명2.속성명;
```

- RIGHT OUTER JOIN : INNER JOIN의 결과를 구한 후, 좌측 항 릴레이션의 어떤 튜플과도 맞지 않는 우측 항의 릴레이션에 있는 튜플들에 NULL 값을 붙여서 INNER JOIN의 결과에 추가함

- 표기 형식

```
SELECT [테이블명1.]속성명, [테이블명2.]속성명, ...
FROM 테이블명1 RIGHT OUTER JOIN 테이블명2
ON 테이블명1.속성명 = 테이블명2.속성명;
```

#### <학생>

학번	이름	학과코드	선배	성적
15	고길동	com		83
16	이순신	han		96
17	김선달	com	15	95
19	아무개	han	16	75
37	박치민		17	55

#### <학과>

학과코드	학과명
com	컴퓨터
han	국어
eng	영어

**예제** <학생> 테이블과 <학과> 테이블에서 학과코드 값이 같은 튜플을 JOIN하여 학번, 이름, 학과코드, 학과명을 출력하는 SQL문을 작성하시오. 이때, 학과코드가 입력되지 않은 학생도 출력하시오.

```
SELECT 학번, 이름, 학생.학과코드, 학과명
FROM 학생 LEFT OUTER JOIN 학과
ON 학생.학과코드 = 학과.학과코드;
```

**해설** INNER JOIN을 하면 학과코드가 입력되지 않은 박치민은 출력되지 않는다. 그러므로 JOIN 구문을 기준으로 왼쪽 테이블, 즉 <학생>의 자료는 모두 출력되는 LEFT JOIN을 사용한 것이다. 다음과 같이 JOIN 구문을 기준으로 테이블의 위치를 교환하여 RIGHT JOIN을 사용해도 결과는 같다.

<결과>

학번	이름	학과코드	학과명
15	고길동	com	컴퓨터
16	이순신	han	국어
17	김선달	com	컴퓨터
19	아무개	han	국어
37	박치민		

## [암기082] 절차형 SQL

- **프로시저(Procedure)** : 절차형 SQL을 활용하여 특정 기능을 수행하는 일종의 트랜잭션 언어로, 호출을 통해 실행되어 미리 저장해 놓은 SQL 작업을 수행함
- **트리거(Trigger)** : 데이터베이스 시스템에서 데이터의 삽입(Insert), 갱신(Update), 삭제(Delete) 등의 이벤트(Event)가 발생할 때마다 관련 작업이 자동으로 수행되는 절차형 SQL
- **사용자 정의 함수** : 프로시저와 유사하게 SQL을 사용하여 일련의 작업을 연속적으로 처리하며, 종료 시 처리 결과를 단일 값으로 반환하는 절차형 SQL
- **커서(Cursor)** : 쿼리문의 처리 결과가 저장되어 있는 메모리 공간을 가리키는 포인터(Pointer)로, 내부에서 자동으로 생성되어 사용되는 묵시적 커서와, 사용자가 직접 정의해서 사용하는 명시적 커서가 있음

## 9장 | 소프트웨어 개발 보안 구축

### [암기083] 하향식 비용 산정 기법

- 과거의 유사한 경험을 바탕으로 전문 지식이 많은 개발자들이 참여한 회의를 통해 비용을 산정하는 비과학적인 방법이다.

#### • 종류

전문가 감정 기법	조직 내에 있는 경험이 많은 두 명 이상의 전문가에게 비용 산정을 의뢰하는 기법
델파이 기법	전문가 감정 기법의 주관적인 편견을 보완하기 위해 많은 전문가의 의견을 종합하여 산정하는 기법

### [암기084] 상향식 비용 산정 기법

- 프로젝트의 세부적인 작업 단위별로 비용을 산정한 후 집계하여 전체 비용을 산정하는 방법이다.

#### • 종류

<p>• 소프트웨어 각 기능의 원시 코드 라인 수의 비관치, 낙관치, 기대치를 측정하여 예측치를 구하고 이를 이용하여 비용을 산정하는 기법</p> <p>• 산정 공식:</p> <ul style="list-style-type: none"> <li>- 노력(인월) = 개발 기간 × 투입 인원</li> <li>= LOC / 1인당 월평균 생산 코드 라인 수</li> <li>- 개발 비용 = 노력(인월) × 단위 비용 (1인당 월평균 인건비)</li> <li>- 개발 기간 = 노력(인월) / 투입 인원</li> <li>- 생산성 = LOC / 노력(인월)</li> </ul> <p><b>예</b> LOC 기법에 의하여 예측된 총 라인 수가 40,000라인, 개발에 참여할 프로그래머가 10명, 프로그래머들의 평균 생산성이 월간 400라인일 때 개발에 소요되는 기간은?</p> <ul style="list-style-type: none"> <li>• 노력(인월) = LOC / 1인당 월평균 생산 코드 라인 수 = 40000 / 400 = 100명</li> <li>• 개발 기간 = 노력(인월) / 투입 인원 = 100 / 10 = 10개월</li> </ul>	<p>LOC(원시 코드 라인 수, source Line Of Code) 기법</p>
<p>개발 단계별 인월수(Effort Per Task) 기법</p>	<p>LOC 기법을 보완하기 위한 기법으로, 각 기능을 구현시키는 데 필요한 노력을 생명 주기의 각 단계별로 산정</p>

## [암기085] 서비스 거부 공격의 유형

Ping of Death (죽음의 핑)	Ping 명령을 전송할 때 패킷의 크기를 인터넷 프로토콜 허용 범위(65,536 바이트) 이상으로 전송하여 공격 대상의 네트워크를 마비시키는 서비스 거부 공격 방법
Smurfing(스머핑)	IP나 ICMP의 특성을 악용하여 엄청난 양의 데이터를 한 사이트에 집중적으로 보냄으로써 네트워크 또는 시스템의 상태를 불안으로 만드는 공격 방법
SYN Flooding	TCP(Transmission Control Protocol)는 신뢰성 있는 전송을 위해 3-way-handshake를 거친 후에 데이터를 전송하게 되는데, SYN Flooding은 공격자가 가상의 클라이언트로 위장하여 3-way-handshake 과정을 의도적으로 중단시킴으로써 공격 대상 지인 서버가 대기 상태에 놓여 정상적인 서비스를 수행하지 못하게 하는 공격 방법
TearDrop	데이터의 송·수신 과정에서 패킷의 크기가 커 여러 개로 분할되어 전송될 때 분할 순서를 알 수 있도록 Fragment Offset 값을 함께 전송하는데, TearDrop은 이 Offset 값을 변경시켜 수신 측에서 패킷을 재조립할 때 오류로 인한 과부하를 발생시킴으로써 시스템이 다운되도록 하는 공격 방법
Land	패킷을 전송할 때 송신 IP 주소와 수신 IP 주소를 모두 공격 대상의 IP 주소로 하여 공격 대상에게 전송하는 것으로, 이 패킷을 받은 공격 대상은 송신 IP 주소가 자신이므로 자신에게 응답을 수행하게 되는데, 이러한 패킷이 계속해서 전송될 경우 자신에 대해 무한히 응답하게 하는 공격
DDoS(Distributed Denial of Service, 분산 서비스 거부) 공격	여러 곳에 분산된 공격 지점에서 한 곳의 서버에 대해 분산 서비스 공격을 수행하는 것으로, 네트워크에서 취약점이 있는 호스트들을 탐색한 후 이들 호스트들에 분산 서비스 공격용 툴을 설치하여 에이전트(Agent)로 만든 후 DDoS 공격에 이용함

## [암기086] 네트워크 침해 공격 관련 용어

스미싱 (Smishing)	각종 행사 안내, 경품 안내 등의 문자 메시지(SMS)를 이용해 사용자의 개인 신용 정보를 빼내는 수법
스피어 피싱 (Spear Phishing)	사회 공학의 한 기법으로, 특정 대상을 선정한 후 그 대상에게 일반적인 이메일로 위장한 메일을 지속적으로 발송하여, 발송 메일의 본문 링크나 첨부된 파일을 클릭하도록 유도해 사용자의 개인 정보를 탈취 함
APT(Advanced Persistent Threats, 지능형 지속 위협)	다양한 IT 기술과 방식들을 이용해 조직적으로 특정 기업이나 조직 네트워크에 침투해 활동 거점을 마련한 뒤 때를 기다리면서 보안을 무력화시키고 정보를 수집한 다음 외부로 빼돌리는 형태의 공격
무작위 대입 공격 (Brute Force Attack)	암호화된 문서의 암호키를 찾아내기 위해 적용 가능한 모든 값을 대입하여 공격하는 방식
큐싱(Qshing)	QR코드(Quick Response Code)를 통해 악성 앱의 다운로드를 유도하거나 악성 프로그램을 설치하도록 하는 금융사기 기법의 하나로, QR코드와 개인정보 및 금융정보를 낚는(Fishing)는 의미의 합성 신조어
SQL 삽입(Injection) 공격	전문 스캐너 프로그램 혹은 봇넷 등을 이용해 웹사이트를 무차별적으로 공격하는 과정에서 취약한 사이트가 발견되면 데이터베이스 등의 데이터를 조작하는 일련의 공격 방식
크로스사이트 스크립팅(XSS)	웹페이지에 악의적인 스크립트를 삽입하여 방문자들의 정보를 탈취하거나, 비정상적인 기능 수행을 유발하는 등 스크립트의 취약점을 악용한 해킹 기법



## [암기087] 정보 보안 침해 공격 관련 용어

좀비(Zombie) PC	악성코드에 감염되어 다른 프로그램이나 컴퓨터를 조종하도록 만들어진 컴퓨터로, C&C(Command & Control) 서버의 제어를 받아 주로 DDoS 공격 등에 이용됨
C&C 서버	해커가 원격지에서 감염된 좀비 PC에 명령을 내리고 악성코드를 제어하기 위한 용도로 사용하는 서버를 말함
봇넷(Botnet)	악성 프로그램에 감염되어 악의적인 의도로 사용될 수 있는 다수의 컴퓨터들이 네트워크로 연결된 형태를 말함
웜(Worm)	네트워크를 통해 연속적으로 자신을 복제하여 시스템의 부하를 높임으로써 결국 시스템을 다운시키는 바이러스의 일종으로, 분산 서비스 거부 공격, 버퍼 오버플로 공격, 슬래머 등이 웜 공격의 한 형태임
제로 데이 공격 (Zero Day Attack)	보안 취약점이 발견되었을 때 발견된 취약점의 존재 자체가 널리 공표되기도 전에 해당 취약점을 통하여 이루어지는 보안 공격으로, 공격의 신속성을 의미함
키로거 공격 (Key Logger Attack)	컴퓨터 사용자의 키보드 움직임을 탐지해 ID, 패스워드, 계좌번호, 카드번호 등과 같은 개인의 중요한 정보를 몰래 빼가는 해킹 공격
랜섬웨어 (Ransomware)	인터넷 사용자의 컴퓨터에 잠입해 내부 문서나 파일 등을 암호화해 사용자가 열지 못하게 하는 프로그램으로, 암호 해독용 프로그램의 전달을 조건으로 사용자에게 돈을 요구하기도 함
백도어 (Back Door, Trap Door)	<ul style="list-style-type: none"> <li>시스템 설계자가 서비스 기술자나 유지 보수 프로그램 작성자(Programmer)의 액세스 편의를 위해 시스템 보안을 제거하여 만들어놓은 비밀 통로로, 컴퓨터 범죄에 악용되기도 함</li> <li>백도어 탐지 방법 : 무결성 검사, 로그 분석, SetUID 파일 검사</li> </ul>
트로이 목마 (Trojan Horse)	정상적인 기능을 하는 프로그램으로 위장하여 프로그램 내에 숨어 있다가 해당 프로그램이 동작할 때 활성화되어 부작용을 일으키는 것으로, 자기 복제 능력은 없음

## [암기088] 보안 요소

기밀성	시스템 내의 정보와 자원은 인가된 사용자에게만 접근이 허용됨
무결성	시스템 내의 정보는 오직 인가된 사용자만 수정할 수 있음
가용성	인가받은 사용자는 언제라도 사용할 수 있음
인증	시스템 내의 정보와 자원을 사용하려는 사용자가 합법적인 사용자인지를 확인하는 모든 행위를 말함
부인 방지	데이터를 송·수신한 자가 송·수신 사실을 부인할 수 없도록 송·수신 증거를 제공함.

## [암기089] 암호 알고리즘

개인키 암호화 (Private Key Encryption) 기법	<ul style="list-style-type: none"> <li>동일한 키로 데이터를 암호화하고 복호화함</li> <li>개인키 암호화 기법은 대칭 암호 기법 또는 단일키 암호화 기법이라고도 함</li> </ul>
공개키 암호화 (Public Key Encryption) 기법	<ul style="list-style-type: none"> <li>데이터를 암호화할 때 사용하는 공개키 (Public Key)는 데이터베이스 사용자에게 공개하고, 복호화할 때의 비밀키(Secret Key)는 관리자가 비밀리에 관리함</li> <li>공개키 암호화 기법은 비대칭 암호 기법이라고도 함</li> </ul>
해시(Hash)	<ul style="list-style-type: none"> <li>임의의 길이의 입력 데이터나 메시지를 고정된 길이의 값이나 키로 변환하는 것을 의미함</li> <li>해시 함수의 종류 : SHA 시리즈, MD5, SNEFRU 등                             <ul style="list-style-type: none"> <li>SHA 시리즈 : 1993년에 미국 NSA가 제작하고 미국 국립표준기술연구소(NIST)에서 표준으로 채택한 암호화 알고리즘으로 가장 많이 사용됨</li> <li>MD5 : 1991년 R. Rivest가 MD4를 개선한 암호화 알고리즘으로 각각의 512 비트 짜리 입력 메시지 블록에 대해 차례로 동작함</li> <li>SNEFRU : 1990년에 R.C. Merkle에 의해 제안된 128, 254비트 암호화 알고리즘</li> </ul> </li> </ul>

## [암기090] 주요 암호화 알고리즘

SEED	<ul style="list-style-type: none"> <li>1999년 한국인터넷진흥원(KISA)에서 개발한 블록 암호화 알고리즘</li> <li>블록 크기는 128비트이며, 키 길이에 따라 128, 256으로 분류</li> </ul>
ARIA(Academy, Research Institute, Agency)	<ul style="list-style-type: none"> <li>2004년 국가정보원과 산학연협회가 개발한 블록 암호화 알고리즘</li> <li>블록 크기는 128비트이며, 키 길이에 따라 128, 192, 256으로 분류</li> </ul>
DES(Data Encryption Standard)	<ul style="list-style-type: none"> <li>1975년 미국 NBS에서 발표한 개인키 암호화 알고리즘</li> <li>블록 크기는 64비트이며, 키 길이는 56비트</li> </ul>
AES(Advanced Encryption Standard)	<ul style="list-style-type: none"> <li>2001년 미국 표준 기술 연구소(NIST)에서 발표한 개인키 암호화 알고리즘</li> <li>블록 크기는 128비트이며, 키 길이에 따라 128, 192, 256으로 분류</li> </ul>
RSA(Rivest Shamir Adleman)	<ul style="list-style-type: none"> <li>1978년 MIT의 라이베스트(Rivest), 샤미르(Shamir), 애들먼(Adelman)에 의해 제안된 공개키 암호화 알고리즘</li> <li>소인수 분해 문제를 이용한 공개키 암호화 기법에 널리 사용되는 암호화 알고리즘</li> </ul>

## 10장 | 응용 SW 기초 기술 활용

### [암기091] 운영체제(OS; Operating System)

- 목적 : 처리 능력 향상, 신뢰도 향상, 사용 가능도 향상, 반환 시간 단축
- 성능 평가 기준

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

### [암기092] UNIX 시스템의 구성

#### 커널(Kernel)

- UNIX의 가장 핵심적인 부분이다.
- 프로세스, 기억장치, 파일, 입·출력 관리, 프로세스 간 통신, 데이터 전송 및 변환 등 여러 가지 기능을 수행한다.

#### 셸(Shell)

- 사용자의 명령어를 인식하여 프로그램을 호출하고, 명령을 수행하는 명령어 해석기이다.
- 시스템과 사용자 간의 인터페이스를 담당한다.
- 종류 : Bourne Shell, C Shell, Korn Shell 등



### [암기093] 페이지 교체 알고리즘

- OPT(OPTimal replacement, 최적 교체) : 앞으로 가장 오랫동안 사용하지 않을 페이지를 교체하는 기법
- FIFO(First In First Out) : 각 페이지가 주기억장치에 적재될 때마다 그때의 시간을 기억시켜 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법
- LRU(Least Recently Used) : 최근에 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법
- LFU(Least Frequently Used) : 사용 빈도가 가장 적은 페이지를 교체하는 기법
- NUR(Not Used Recently) : 최근에 사용하지 않은 페이지를 교체하는 기법으로, 참조 비트(Reference Bit)와 변형 비트(Modified Bit)가 사용됨
- SCR(Second Chance Replacement, 2차 기회 교체) : 가장 오랫동안 주기억장치에 있던 페이지 중 자주 사용되는 페이지의 교체를 방지하기 위한 것으로, FIFO 기법의 단점을 보완하는 기법



## [암기094] 비선점(Non-preemptive) 스케줄링

- 이미 할당된 CPU를 다른 프로세스가 강제로 빼앗아 사용할 수 없는 스케줄링 기법이다.
- FCFS(First Come First Service, 선입 선출) = FIFO(First In First Out) : 준비상태 큐에 도착한 순서에 따라 차례로 CPU를 할당하는 기법으로, 가장 간단한 알고리즘임
- SJF(Shortest Job First, 단기 작업 우선) : 준비상태 큐에서 기다리고 있는 프로세스들 중에서 실행 시간이 가장 짧은 프로세스에게 먼저 CPU를 할당하는 기법
- HRN(Highest Response-ratio Next)
  - 실행 시간이 긴 프로세스에 불리한 SJF 기법을 보완하기 위한 것으로, 대기 시간과 서비스(실행) 시간을 이용하는 기법
  - 우선순위 계산식 : (대기 시간 + 서비스 시간) / 서비스 시간
- 기한부(Deadline) : 프로세스에게 일정한 시간을 주어 그 시간 안에 프로세스를 완료하도록 하는 기법
- 우선순위(Priority) : 준비상태 큐에서 기다리는 각 프로세스마다 우선순위를 부여하여 그 중 가장 높은 프로세스에게 먼저 CPU를 할당하는 기법

### 잠깐만요! 에이징(Aging) 기법

시스템에서 특정 프로세스의 우선순위가 낮아 무한정 기다리게 되는 경우, 한 번 양보하거나 기다린 시간에 비례하여 일정 시간이 지나면 우선순위를 한 단계씩 높여 가까운 시간 안에 자원을 할당받도록 하는 기법입니다.

## [암기095] 선점(Preemptive) 스케줄링

- 하나의 프로세스가 CPU를 할당받아 실행하고 있을 때 우선순위가 높은 다른 프로세스가 CPU를 강제로 빼앗아 사용할 수 있는 스케줄링 기법이다.
- 선점 우선순위 : 준비상태 큐의 프로세스들 중에서 우선순위가 가장 높은 프로세스에게 먼저 CPU를 할당하는 기법
- SRT(Shortest Remaining Time) : 비선점 스케줄링인 SJF 기법을 선점 형태로 변형한 기법으로, 선점 SJF 기법이라고도 함

- 라운드 로빈(RR; Round Robin) : 시분할 시스템(Time Sharing System)을 위해 고안된 방식으로, FCFS 기법과 같이 준비상태 큐에 먼저 들어온 프로세스가 먼저 CPU를 할당받지만 각 프로세스는 시간 할당량(Time Slice, Quantum) 동안만 실행한 후 실행이 완료되지 않으면 다음 프로세스에게 CPU를 넘겨주고 준비상태 큐의 가장 뒤로 배치됨
- 다단계 큐(MQ; Multi-level Queue) : 프로세스를 특정 그룹으로 분류할 수 있을 경우 그룹에 따라 각기 다른 준비상태 큐를 사용하는 기법
- 다단계 피드백 큐(MFQ; Multi-level Feedback Queue) : 특정 그룹의 준비상태 큐에 들어간 프로세스가 다른 준비상태 큐로 이동할 수 없는 다단계 큐 기법을 준비상태 큐 사이를 이동할 수 있도록 개선한 기법



## [암기096] 교착상태(Dead Lock)

- 상호 배제에 의해 나타나는 문제점으로, 둘 이상의 프로세스들이 자원을 점유한 상태에서 서로 다른 프로세스가 점유하고 있는 자원을 요구하며 무한정 기다리는 현상이다.
- 교착상태 발생의 필요 충분 조건
  - 상호 배제(Mutual Exclusion) : 한 번에 한 개의 프로세스만이 공유 자원을 사용할 수 있어야 함
  - 점유와 대기(Hold and Wait) : 최소한 하나의 자원을 점유하고 있으면서 다른 프로세스에 할당되어 사용되고 있는 자원을 추가로 점유하기 위해 대기하는 프로세스가 있어야 함
  - 비선점(Non-preemption) : 다른 프로세스에 할당된 자원은 사용이 끝날 때까지 강제로 빼앗을 수 없어야 함
  - 환형 대기(Circular Wait) : 공유 자원과 공유 자원을 사용하기 위해 대기하는 프로세스들이 원형으로 구성되어 있어 자신에게 할당된 자원을 점유하면서 앞이나 뒤에 있는 프로세스의 자원을 요구해야 함



## [암기097] 교착상태 해결 방법

예방 기법 (Prevention)	<ul style="list-style-type: none"> <li>교착상태가 발생하지 않도록 사전에 시스템을 제어하는 방법으로, 교착상태 발생의 네 가지 조건 중에서 어느 하나를 제거(부정)함으로써 수행됨</li> <li>종류 : 상호 배제(Mutual Exclusion) 부정, 점유 및 대기(Hold and Wait) 부정, 비선점(Non-preemption) 부정, 환형 대기(Circular Wait) 부정</li> </ul>
회피 기법 (Avoidance)	교착상태가 발생할 가능성을 배제하지 않고 교착상태가 발생하면 적절히 피해나가는 방법으로, 주로 은행원 알고리즘(Banker's Algorithm)이 사용됨
발견 (Detection) 기법	시스템에 교착 상태가 발생했는지 점검하여 교착 상태에 있는 프로세스와 자원을 발견하는 것으로, 자원 할당 그래프 등을 사용함
회복 (Recovery) 기법	교착 상태를 일으킨 프로세스를 종료하거나 교착 상태의 프로세스에 할당된 자원을 선점하여 프로세스나 자원을 회복하는 것

## [암기098] DBMS(데이터베이스 관리 시스템)

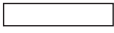







사용자와 데이터베이스 사이에서 사용자의 요구에 따라 정보를 생성해주고, 데이터베이스를 관리해 주는 소프트웨어로, 필수 기능은 다음과 같다.

정의 기능	모든 응용 프로그램들이 요구하는 데이터 구조를 지원하기 위해 데이터베이스에 저장될 데이터의 형(Type)과 구조에 대한 정의, 이용방식, 제약 조건 등을 명시하는 기능
조작 기능	데이터 검색, 갱신, 삽입, 삭제 등을 체계적으로 처리하기 위해 사용자와 데이터베이스 사이의 인터페이스 수단을 제공하는 기능
제어 기능	데이터베이스를 접근하는 갱신, 삽입, 삭제 작업이 정확하게 수행되어 데이터의 무결성이 유지되도록 제어하는 기능

## [암기099] 고급 데이터베이스

- 데이터 웨어하우스(Data Warehouse) : 급증하는 다량의 데이터를 효과적으로 분석하여 정보화하고 이를 여러 계층의 사용자들이 효율적으로 사용할 수 있도록 한 데이터베이스
- 데이터 마트(Data Mart) : 전사적으로 구축된 데이터 웨어하우스로부터 특정 주제나 부서 중심으로 구축된 소규모 단일 주제의 데이터 웨어하우스를 말함
- 데이터 마이닝(Data Mining) : 데이터 웨어하우스에 저장된 데이터 집합에서 사용자의 요구에 따라 유용하고 가능성 있는 정보를 발견하기 위한 기법
- OLAP(Online Analytical Processing) : 다차원으로 이루어진 데이터로부터 통계적인 요약 정보를 분석하여 의사결정에 활용하는 방식
- OLTP(Online Transaction Processing) : 온라인 업무 처리 형태의 하나로 네트워크상의 여러 이용자가 실시간으로 데이터베이스의 데이터를 갱신하거나 검색하는 등의 단위 작업을 처리하는 방식

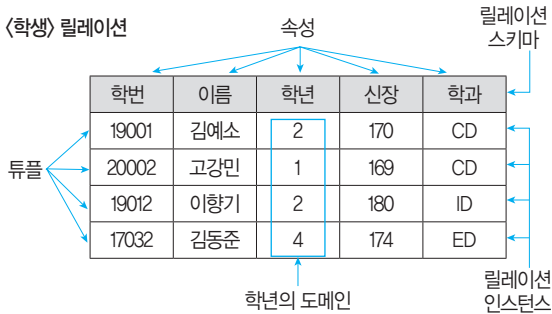
## [암기100] ER(Entity Relationship) 도형

도형	의미
	개체 타입(Entity Type)
	관계 타입(Relationship Type)
	속성
	기본키(Primary Key) 속성
	개체 타입과 속성을 연결
	다중값 속성
	복합 속성
	개체 타입 간의 연관성



## [암기101] 관계 데이터베이스의 Relation 구조

릴레이션은 데이터들을 표(Table)의 형태로 표현한 것으로, 구조를 나타내는 릴레이션 스키마와 실제 값들인 릴레이션 인스턴스로 구성된다.



### 튜플(Tuple)

- 릴레이션을 구성하는 각각의 행이다.
- 튜플의 수 = 카디널리티(Cardinality) = 기수 = 대응수

### 속성(Attribute)

- 릴레이션을 구성하는 각각의 열이다.
- 속성의 수 = 디그리(Degree) = 차수

### 도메인(Domain)

- 하나의 애트리뷰트가 취할 수 있는 같은 타입의 원자(Atomic)값들의 집합이다.
- 실제 애트리뷰트 값이 나타날 때 그 값의 합법 여부를 시스템이 검사하는 데에도 이용된다.

### 릴레이션 인스턴스(Relation Instance)

데이터 개체를 구성하고 있는 속성들에 데이터 타입이 정의되어 구체적인 데이터 값을 갖고 있는 것을 말한다.

## [암기102] 키(Key)의 개념 및 종류

- 키(Key)는 데이터베이스에서 조건에 만족하는 튜플을 찾거나 순서대로 정렬할 때 기준이 되는 속성이다.
- 슈퍼키(Super Key) : 한 릴레이션 내에 있는 속성들의 집합으로 구성된 키로, 릴레이션을 구성하는 모든 튜플에 대해 유일성(Uunique)은 만족하지만, 최소성(Minimality)은 만족하지 못함

- 후보키(Candidate Key) : 릴레이션을 구성하는 속성들 중에서 튜플을 유일하게 식별하기 위해 사용되는 속성들의 부분집합으로, 유일성과 최소성을 모두 만족함
- 기본키(Primary Key) : 후보키 중에서 특별히 선정된 키로 중복된 값과 NULL 값을 가질 수 없음
- 대체키(Alternate Key) : 후보키 중에서 선정된 기본키를 제외한 나머지 후보키를 의미함
- 외래키(Foreign Key) : 다른 릴레이션의 기본키를 참조하는 속성 또는 속성들의 집합을 의미하며, 릴레이션 간의 관계를 표현할 때 사용함

## [암기103] 무결성(Integrity)

- 데이터베이스에 저장된 데이터 값과 그것이 표현하는 현실 세계의 실제값이 일치하는 정확성을 의미한다.
- 개체 무결성(Entity Integrity, 실체 무결성) : 기본 테이블의 기본키를 구성하는 어떤 속성도 Null 값이나 중복값을 가질 수 없다는 규정
- 도메인 무결성(Domain Integrity, 영역 무결성) : 주어진 속성 값이 정의된 도메인에 속한 값이어야 한다는 규정
- 참조 무결성(Referential Integrity) : 외래키 값은 Null이거나 참조 릴레이션의 기본키 값과 동일해야 함. 즉 릴레이션은 참조할 수 없는 외래키 값을 가질 수 없다는 규정
- 사용자 정의 무결성(User-Defined Integrity) : 속성 값들이 사용자가 정의한 제약조건에 만족해야 한다는 규정
- NULL 무결성 : 릴레이션의 특정 속성 값이 NULL이 될 수 없도록 하는 규정
- 고유(Unique) 무결성 : 릴레이션의 특정 속성에 대해 각 튜플이 갖는 속성값들이 서로 달라야 한다는 규정
- 키(Key) 무결성 : 하나의 릴레이션에는 적어도 하나의 키가 존재해야 한다는 규정
- 관계(Relationship) 무결성 : 릴레이션에 어느 한 튜플의 삽입 가능 여부 또는 한 릴레이션과 다른 릴레이션의 튜플들 사이의 관계에 대한 적절성 여부를 지정한 규정

## [암기104] IPv6(Internet Protocol version 6)

- 현재 사용하고 있는 IP 주소 체계인 IPv4의 주소 부족 문제를 해결하기 위해 개발되었다.
- 16비트씩 8부분, 총 128비트로 구성되어 있다.
- 각 부분을 16진수로 표현하고, 콜론(:)으로 구분한다.
- IPv4에 비해 자료 전송 속도가 빠르고, IPv4와 호환성이 뛰어나다.
- 인증성, 기밀성, 데이터 무결성의 지원으로 보안 문제를 해결할 수 있다.
- IPv6의 주소 체계
  - 유니캐스트(Unicast) : 단일 송신자와 단일 수신자 간의 통신(1:1 통신에 사용)
  - 멀티캐스트(Multicast) : 단일 송신자와 다중 수신자 간의 통신(1:N 통신에 사용)
  - 애니캐스트(Anycast) : 단일 송신자와 가장 가까이 있는 단일 수신자 간의 통신(1:1 통신에 사용)

## [암기105] TCP/IP / 도메인 네임

- TCP/IP : 인터넷에 연결된 서로 다른 기종의 컴퓨터들이 데이터를 주고받을 수 있도록 하는 표준 프로토콜
- 도메인 네임 : 숫자로 된 IP 주소를 사람이 이해하기 쉬운 문자 형태로 표현한 것으로, 문자로 된 도메인 네임을 컴퓨터가 이해할 수 있는 IP 주소로 변환하는 역할을 하는 시스템을 DNS(Domain Name System)라고 함

## [암기106] OSI(Open System Interconnection) 참조 모델

- OSI 7계층 : 하위 계층(물리 계층 → 데이터 링크 계층 → 네트워크 계층), 상위 계층(전송 계층 → 세션 계층 → 표현 계층 → 응용 계층)
- 물리 계층(Physical Layer) : 전송에 필요한 두 장치 간의 실제 접속과 절단 등 기계적, 전기적, 기능적, 절차적 특성에 대한 규칙을 정의함

### • 데이터 링크 계층(Data Link Layer)

- 두 개의 인접한 개방 시스템들 간에 신뢰성 있고 효율적인 정보 전송을 할 수 있도록 함
- 흐름 제어, 프레임 동기화, 오류 제어, 순서 제어

### • 네트워크 계층(Network Layer, 망 계층)

- 개방 시스템들 간의 네트워크 연결을 관리하는 기능과 데이터의 교환 및 중계 기능을 함
- 경로 설정(Routing), 트래픽 제어, 패킷 정보 전송

### • 전송 계층(Transport Layer)

- 종단 시스템(End-to-End) 간의 전송 연결 설정, 데이터 전송, 연결 해제 기능을 함
- 주소 설정, 다중화(데이터의 분할과 재조립), 오류 제어, 흐름 제어

### • 세션 계층(Session Layer)

- 송·수신 측 간의 관련성을 유지하고 대화 제어를 담당함
- 대화(회화) 구성 및 동기 제어, 데이터 교환 관리 기능

### • 표현 계층(Presentation Layer)

- 응용 계층으로부터 받은 데이터를 세션 계층에 맞게, 세션 계층에서 받은 데이터는 응용 계층에 맞게 변환하는 기능
- 코드 변환, 데이터 암호화, 데이터 압축, 구문 검색, 정보 형식(포맷) 변환, 문맥 관리 기능

- 응용 계층(Application Layer) : 사용자(응용 프로그램)가 OSI 환경에 접근할 수 있도록 응용 프로세스 간의 정보 교환, 전자 사서함, 파일 전송, 가상 터미널 등의 서비스를 제공함

## [암기107] 네트워크 관련 장비

### • 허브(Hub)

- 한 사무실이나 가까운 거리의 컴퓨터들을 연결하는 장치로, 각 회선을 통합적으로 관리하며, 신호 증폭 기능을 하는 리피터의 역할도 포함함
- 더미 허브(Dummy Hub) : 네트워크에 흐르는 모든 데이터를 단순히 연결하는 기능만을 제공함
- 스위칭 허브(Switching Hub) : 네트워크상에 흐르는 데이터의 유무 및 흐름을 제어하여 각각의 노드가 허브의 최대 대역폭을 사용할 수 있는 지능형 허브임

- 리피터(Repeater) : 물리 계층의 장비로, 전송되는 신호가 왜곡되거나 약해질 경우 원래의 신호 형태로 재생함
- 브리지(Bridge) : 데이터 링크 계층의 장비로, LAN과 LAN을 연결하거나 LAN 안에서의 컴퓨터 그룹을 연결함
- 라우터(Router) : 네트워크 계층의 장비로, LAN과 LAN의 연결 및 경로 선택, 서로 다른 LAN이나 LAN과 WAN을 연결함
- 게이트웨이(Gateway) : 전 계층(1~7계층)의 프로토콜 구조가 전혀 다른 네트워크의 연결을 수행함
- 스위치(Switch) : 브리지와 같이 LAN과 LAN을 연결하여 훨씬 더 큰 LAN을 만드는 장치

### [암기108] 프로토콜(Protocol)

- 서로 다른 기기들 간의 데이터 교환을 원활하게 수행할 수 있도록 표준화시켜 놓은 통신 규약이다.
- 프로토콜의 기본 요소

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

### [암기109] TCP/IP의 구조

OSI	TCP/IP	기능
응용 계층 표현 계층 세션 계층	응용 계층	<ul style="list-style-type: none"> <li>• 응용 프로그램 간의 데이터 송·수신을 제공</li> <li>• TELNET, FTP, SMTP, SNMP, DNS, HTTP 등</li> </ul>
전송 계층	전송 계층	<ul style="list-style-type: none"> <li>• 호스트들 간의 신뢰성 있는 통신을 제공</li> <li>• TCP, UDP</li> </ul>

네트워크 계층	인터넷 계층	<ul style="list-style-type: none"> <li>• 데이터 전송을 위한 주소 지정, 경로 설정을 제공</li> <li>• IP, ICMP, IGMP, ARP, RARP</li> </ul>
데이터 링크 계층 물리 계층	네트워크 액세스 계층	<ul style="list-style-type: none"> <li>• 실제 데이터(프레임)를 송·수신하는 역할</li> <li>• Ethernet, IEEE 802, HDLC, X.25, RS-232C, ARQ 등</li> </ul>

### [암기110] TCP/IP의 응용 계층 프로토콜

- FTP(File Transfer Protocol) : 컴퓨터와 컴퓨터 또는 컴퓨터와 인터넷 사이에서 파일을 주고받을 수 있도록 하는 원격 파일 전송 프로토콜
- SMTP(Simple Mail Transfer Protocol) : 전자 우편을 교환하는 서비스
- TELNET : 멀리 떨어져 있는 컴퓨터에 접속하여 자신의 컴퓨터처럼 사용할 수 있도록 해주는 서비스
- SNMP(Simple Network Management Protocol) : TCP/IP의 네트워크 관리 프로토콜로, 라우터나 허브 등 네트워크 기기의 네트워크 정보를 네트워크 관리 시스템에 보내는 데 사용되는 표준 통신 규약
- DNS(Domain Name System) : 도메인 이름을 IP 주소로 매핑(Mapping)하는 시스템
- HTTP(HyperText Transfer Protocol) : 월드 와이드 웹(WWW)에서 HTML 문서를 송수신 하기 위한 표준 프로토콜

### [암기111] TCP/IP의 전송 계층 프로토콜

- TCP(Transmission Control Protocol)
  - 양방향 연결(Full Duplex Connection)형 서비스를 제공한다.
  - 가상 회선 연결(Virtual Circuit Connection) 형태의 서비스를 제공한다.
  - 스트림 위주의 전달(패킷 단위)을 한다.
  - 신뢰성 있는 경로를 확립하고 메시지 전송을 감독한다.

- UDP(User Datagram Protocol)
  - 데이터 전송 전에 연결을 설정하지 않는 비연결형 서비스를 제공한다.
  - TCP에 비해 상대적으로 단순한 헤더 구조를 가지므로, 오버헤드가 적다.
  - 고속의 안정성 있는 전송 매체를 사용하여 빠른 속도를 필요로 하는 경우, 동시에 여러 사용자에게 데이터를 전달할 경우, 정기적으로 반복해서 전송할 경우에 사용한다.
  - 실시간 전송에 유리하며, 신뢰성보다는 속도가 중요시되는 네트워크에서 사용된다.
- RTCP(Real-Time Control Protocol)
  - RTP(Real-time Transport Protocol) 패킷의 전송 품질을 제어하기 위한 제어 프로토콜이다.
  - 세션(Session)에 참여한 각 참여자들에게 주기적으로 제어 정보를 전송한다.
  - 하위 프로토콜은 데이터 패킷과 제어 패킷의 다중화(Multiplexing)를 제공한다.

## [암기112] TCP/IP의 인터넷 계층 프로토콜

- IP(Internet Protocol) : 전송할 데이터에 주소 지정 및 경로 설정 등의 기능을 하며, 비연결형인 데이터그램 방식을 사용하므로 신뢰성이 보장되지 않음
- ICMP(Internet Control Message Protocol) : IP와 조합하여 통신중에 발생하는 오류의 처리와 전송 경로 변경 등을 위한 제어 메시지를 관리하는 역할을 하며, 헤더는 8Byte로 구성됨
- IGMP(Internet Group Management Protocol) : 멀티캐스트를 지원하는 호스트나 라우터 사이에서 멀티캐스트 그룹 유지를 위해 사용됨
- ARP(Address Resolution Protocol) : 호스트의 IP 주소를 호스트와 연결된 네트워크 접속 장치의 물리적 주소(MAC Address)로 바꿈
- RARP(Reverse Address Resolution Protocol) : ARP와 반대로 물리적 주소를 IP 주소로 변환하는 기능을 함

## [암기113] 라우팅 프로토콜

- RIP(Routing Information Protocol) : 현재 가장 널리 사용되는 라우팅 프로토콜로, 소규모 동종의 네트워크 내에서 효율적인 방법이며, 최대 홉수를 15로 제한함
- IGRP(Interior Gateway Routing Protocol) : RIP의 단점을 보완하기 위해 만들어 개발된 것으로, 네트워크 상태를 고려하여 라우팅하며, 중규모 네트워크에 적합함
- OSPF(Open Shortest Path First Protocol) : 대규모 네트워크에서 많이 사용되는 라우팅 프로토콜로, 라우팅 정보에 변화가 생길 경우 변화된 정보만 네트워크 내의 모든 라우터에 알려지며, RIP에 비해 홉수에 제한이 없음
- BGP(Border Gateway Protocol) : 자율 시스템(AS) 간의 라우팅 프로토콜로, EGP의 단점을 보완하기 위해 개발되었음

## 11장 | 제품 소프트웨어 패키징

### [암기114] 릴리즈 노트(Release Note)

- 개발 과정에서 정리된 릴리즈 정보를 소프트웨어의 최종 사용자인 고객과 공유하기 위한 문서이다.
- 릴리즈 노트 작성 순서
  - ① 모듈 식별 : 모듈별 빌드 수행 후 릴리즈 노트에 작성될 내용 확인
  - ② 릴리즈 정보 확인 : 릴리즈 노트 및 소프트웨어 이름, 릴리즈 버전 및 날짜, 노트 날짜 및 버전 등 확인
  - ③ 릴리즈 노트 개요 작성 : 소프트웨어 및 변경사항 전체에 대한 간략한 내용 작성
  - ④ 영향도 체크 : 버그나 이슈 관련 내용 또는 해당 릴리즈 버전에서의 기능 변화가 다른 소프트웨어나 기능을 사용하는데 미칠 수 있는 영향 기술
  - ⑤ 정식 릴리즈 노트 작성 : Header(머릿말), 개요, 영향도 체크 항목을 포함하여 정식 릴리즈 노트에 작성될 기본 사항 작성
  - ⑥ 추가 개선 항목 식별 : 추가 버전 릴리즈 노트 작성시 필요한 경우 추가 릴리즈 노트 작성
- ※ Header(머릿말) : 릴리즈 노트 이름, 소프트웨어 이름, 릴리즈 버전, 릴리즈 날짜, 릴리즈 노트 날짜, 릴리즈 노트 버전 등을 표시함



## [암기115] 디지털 저작권 관리(DRM)

- 저작권자가 배포한 디지털 콘텐츠가 저작권자가 의도한 용도로만 사용되도록 디지털 콘텐츠의 생성, 유통, 이용까지의 전 과정에 걸쳐 사용되는 디지털 콘텐츠 관리 및 보호 기술이다.
- 디지털 저작권 관리의 기술 요소
  - 암호화(Encryption) : 콘텐츠 및 라이선스를 암호화하고 전자 서명을 할 수 있는 기술
  - 키 관리(Key Management) : 콘텐츠를 암호화한 키에 대한 저장 및 분배 기술
  - 암호화 파일 생성(Packager) : 콘텐츠를 암호화된 콘텐츠로 생성하기 위한 기술
  - 식별 기술(Identification) : 콘텐츠에 대한 식별 체계 표현 기술
  - 저작권 표현(Right Expression) : 라이선스의 내용 표현 기술
  - 정책 관리(Policy Management) : 라이선스 발급 및 사용에 대한 정책 표현 및 관리 기술
  - 크랙 방지(Tamper Resistance) : 크랙에 의한 콘텐츠 사용 방지 기술
  - 인증(Authentication) : 라이선스 발급 및 사용의 기준이 되는 사용자 인증 기술

## [암기116] 소프트웨어 패키징의 형상 관리(SCM)

- 형상 관리(SCM; Software Configuration Management)는 소프트웨어의 개발 과정에서 소프트웨어의 변경 사항을 관리하기 위해 개발된 일련의 활동이다.
- 형상 관리 기능
  - 형상 식별 : 형상 관리 대상에 이름과 관리 번호를 부여하고, 계층(Tree) 구조로 구분하여 수정 및 추적이 용이하도록 하는 작업
  - 버전 제어 : 소프트웨어 업그레이드나 유지 보수 과정에서 생성된 다른 버전의 형상 항목을 관리하고, 이를 위해 특정 절차와 도구(Tool)를 결합시키는 작업

- 형상 통제(변경 관리) : 식별된 형상 항목에 대한 변경 요구를 검토하여 현재의 기준선(Base Line)이 잘 반영될 수 있도록 조정하는 작업
- 형상 감사 : 기준선의 무결성을 평가하기 위해 확인, 검증, 검열 과정을 통해 공식적으로 승인하는 작업
- 형상 기록(상태 보고) : 형상의 식별, 통제, 감사 작업의 결과를 기록·관리하고 보고서를 작성하는 작업

## [암기117] 소프트웨어 버전 관리 도구

- 공유 폴더 방식 : 버전 관리 자료가 로컬 컴퓨터의 공유 폴더에 저장되어 관리되는 방식
- 클라이언트/서버 방식 : 버전 관리 자료가 중앙 시스템(서버)에 저장되어 관리되는 방식
- 분산 저장소 방식 : 버전 관리 자료가 하나의 원격 저장소와 분산된 개발자 PC의 로컬 저장소에 함께 저장되어 관리되는 방식
- Subversion(SVN) : CVS(Concurrent Version System)를 개선한 것으로, 클라이언트/서버 구조이며, 아파치 소프트웨어 재단에서 2000년에 발표하였음
- Git : 리누스 토발즈(Linus Torvalds)가 2005년 리눅스 커널 개발에 사용할 관리 도구로 개발한 이후 주니오 하마노(Junio Hamano)에 의해 유지 보수되고 있음

## [암기118] Git 명령어

- add : 작업 내역을 지역 저장소에 저장하기 위해 스테이징 영역(Staging Area)에 추가
- commit : 작업 내역을 지역 저장소에 저장
- branch : 새로운 브랜치 생성
- checkout : 지정한 브랜치로 이동



- merge : 지정한 브랜치의 변경 내역을 현재 HEAD 포인터가 가리키는 브랜치에 반영함으로써 두 브랜치를 병합
- init : 지역 저장소 생성
- remote add : 원격 저장소에 연결
- push : 로컬 저장소의 변경 내역을 원격 저장소에 반영
- fetch : 원격 저장소의 변경 이력만을 지역 저장소로 가져와 반영
- clone : 원격 저장소의 전체 내용을 지역 저장소로 복제
- fork : 지정한 원격 저장소의 내용을 자신의 원격 저장소로 복제

## [암기119] 빌드 자동화 도구

### Jenkins

- Java 기반의 오픈 소스 형태로, 가장 많이 사용되는 빌드 자동화 도구이다.
- 서블릿 컨테이너에서 실행되는 서버 기반 도구이다.
- SVN, Git 등 대부분의 형상 관리 도구와 연동이 가능하다.
- 여러 대의 컴퓨터를 이용한 분산 빌드나 테스트가 가능하다.

### Gradle

- Groovy를 기반으로 한 오픈 소스 형태의 자동화 도구로, 안드로이드 앱 개발 환경에서 사용된다.
- 안드로이드 뿐만 아니라 플러그인을 설정하면, Java, C/C++, Python 등의 언어도 빌드가 가능하다.
- Groovy를 사용해서 만든 DSL(Domain Specific Language)을 스크립트 언어로 사용한다.
- Gradle은 실행할 처리 명령들을 모아 태스크(Task)로 만든 후 태스크 단위로 실행한다.

