


---

## 위캔코딩 스터디방 합격자 '정처기 2 회 실기 합' 님의 정보처리기사 용어 필기 노트

---

 출처 : 오픈채팅 위캔코딩 스터디방

★ 외부 유출시에 출처 포함해주세요 ★ 오타 및 오류 말씀주세요

### ● 트랜잭션: 인가받지 않은 사용자로부터 데이터를 보장하기위해 DBMS 가 가져야하는 특성

#### ● 트랜잭션의 특성(ACID)

- 1) **원자성(Atomicity)**: 트랜잭션을 구성하는 연산 전체가 모두 정상적으로 실행되거나 모두 취소되어야 하는 성질
- 2) **일관성(Consistency)**: 시스템이 가지고 있는 고정요소는 트랜잭션 수행 전과 트랜잭션 수행 완료 후의 상태가 같아야 하는 성질
- 3) **격리성=고립성(Isolation)**: 동시에 실행되는 트랜잭션들이 서로 영향을 미치지 않아야 한다는 성질
- 4) **영속성=지속성(Durability)**: 성공이 완료된 트랜잭션의 결과는 영속적으로 데이터베이스에 저장되어야 하는 성질

### ● 트랜잭션 제어언어 3 가지: 트랜잭션의 결과를 허용하거나 취소하는 목적으로 사용되는 언어

- 1) **커밋(COMMIT)**: 트랜잭션을 메모리에 영구적으로 저장하는 명령어
- 2) **롤백(ROLLBACK)**: 트랜잭션 내역을 저장 무효화시키는 명령어
- 3) **체크포인트(CHECKPOINT)**: ROLLBACK 을 위한 시점을 지정하는 명령어

### ● 병행제어 기법의 종류

- **로킹**: 일관성과 무결성을 유지하기 위해 트랜잭션의 순차적인 진행을 보장하는 직렬화 기법  
DB, 파일, 레코드 등은 로킹 단위 가능 / 로킹 단위 작아지면 DB 공유도 증가&로킹 오버헤드 증가 / -  
한번에 로킹할 수 있는 객체의 크기를 로킹 단위라고 함
- **낙관적 검증**: 검증하지 않고 일단 트랜잭션 수행한 뒤 트랜잭션 종료 시 검증 수행하여 DB 반영  
타임 스탬프 순서: 타임스탬프를 부여하여 부여된 시간에 따라 수행
- **다중 버전 동시성 제어**: 트랜잭션의 타임스탬프와 접근하려는 데이터의 타임스탬프 비교하여  
직렬가능성이 보장되는 적절한 버전을 선택하여 접근하도록 하는 기법

## ● 회복 기법 요소

- **REDO**: 장애 발생 전 DB 로 복구하는 기법으로 디스크에 저장된 로그를 분석하여 시작(Start)과 완료(Commit)기록이 있는 트랜잭션의 작업들을 재작업하는 기법
- **UNDO**: 장애 시 디스크에 저장된 로그를 분석하여 시작(Start)은 있지만 완료(Commit) 기록이 없는 트랜잭션들이 작업한 변경 내용을 모두 취소하는 기법

## ● 회복 기법 종류

### - 로그 기반 회복 기법

1. 지연 갱신 회복 기법: 트랜잭션이 완료되기 전까지 DB 에 기록하지 않는 기법
2. 즉각 갱신 회복 기법: 트랜잭션 수행 중 갱신 결과를 바로 DB 에 반영하는 기법

- **체크 포인트 회복 기법(Checkpoint Recovery)**: 장애 발생 시 검사점 이후에 처리된 트랜잭션에 대해서만 장애 발생 이전의 상태로 복원시키는 회복 기법

- **그림자 페이징 회복 기법(Shadow Paging Recovery)**: DB 트랜잭션 수행 시 복제본을 생성하여 DB 장애 시 이를 이용해 복구하는 기법

## ● DDL 대상

- 1) **도메인**: 하나의 속성이 가질 수 있는 원자값들의 집합
- 2) **스키마**: DB 의 구조, 제약조건 등의 정보를 담고 있는 기본적인 구조
  - > **외부 스키마**: 사용자나 개발자의 관점에서 필요로 하는 DB 의 논리적 구조
  - > **개념 스키마**: DB 의 전체적인 논리적 구조
  - > **내부 스키마**: 물리적 저장 장치의 관점에서 보는 DB 구조
- 3) **테이블**: 데이터 저장 공간
- 4) **뷰**: 하나 이상의 물리 테이블에서 유도되는 가상의 테이블
- 5) **인덱스**: 검색을 빠르게 하기 위한 데이터 구조

## ● DCL 관련

- 1) **GRANT**: 관리자가 사용자에게 DB 에 대한 권한을 부여하는 명령어
- 2) **REVOKE**: 관리자가 사용자에게 부여했던 권한을 회수하기 위한 명령어

## ● 절차형 SQL

1) **프로시저(Procedure)**: 일련의 쿼리들을 마치 하나의 함수처럼 실행하기 위한 쿼리의 집합  
사용자 정의 함수(User-Defined Function): 일련의 SQL 처리를 수행하고 수행결과를 단일값으로 반환할 수 있는 절차형 SQL

2) **트리거(Trigger)**: DB 시스템에서 삽입, 갱신, 삭제 등의 이벤트가 발생할 때마다 관련 작업이 자동으로 수행되는 절차형 SQL

3) **옵티마이저**: SQL 을 가장 빠르고 효율적으로 수행할 최적의 처리 경로를 생성해주는 DBMS 내부의 핵심엔진

- **규칙기반 옵티마이저**: 통계 정보가 없는 상태에서 사전 등록된 규칙에 따라 질의실행 계획을 선택하는 옵티마이저

- **비용기반 옵티마이저**: 통계 정보로부터 모든 접근 경로를 고려한 질의실행 계획을 선택하는 옵티마이저

## ● 식별자 표기법

- **카멜 표기법**: 첫 단어 시작만 소문자로 표시하고, 각 단어의 첫 글자는 대문자로 지정하는 표기법

- **파스칼 표기법**: 각 단어의 첫 글자는 대문자로 지정하는 표기법

- **스네이크 표기법**: 여러 단어가 이어지면 단어 사이에 언더바를 넣는 표기법

- **헝가리안 표기법**: 두어에 자료형을 붙이는 표기법(int->n/char->c/문자열->sz)

● **소프트웨어 생명주기(Software Development Life Cycle)**: 요구분석부터 유지보수까지 전 공정을 체계화한 절차이다.

요구사항분석 ⇄ 설계 ⇄ 구현 ⇄ 테스트 ⇄ 유지보수

- **폭포수모델**: 각 단계를 확실히 마무리 지은 후에 다음 단계로 넘어감

- **프로토타입 모델**: 프로토타입 먼저 구현

- **나선형 모델**: 점진적으로 완벽한 시스템으로(위험최소화)

- **반복적 모델**: 반복적 개발로 점증 완성

## ● 소프트웨어 개발방법론

- **구조적 방법론**: 전체 시스템을 기능에 따라 나누어 개발하고, 이를 통합하는 분할과 정복 접근 방식의 방법론

- **컴포넌트 기반 방법론**: 소프트웨어를 구성하는 컴포넌트를 조립해서 하나의 새로운 응용프로그램을 작성하는 방법론

- **정보공학 방법론**: 정보 시스템 개발에 필요한 관리 절차와 작업 기법을 체계화한 방법론

- **애자일 방법론**: 절차보다는 사람이 중심이 되어 변화에 유연하고 신속하게 적응하며, 효율적으로 시스템을 개발할 수 있는 방법론
- **XP**: 의사소통 개선과 즉각적 피드백으로 소프트웨어 품질을 높이기 위한 방법론
  - > Pair Programming, Collective Ownership, Continuous Integration, Metaphor, 40-Hour Work, On Site Customer
  - > 테스트 기반 개발(Test Driven Development): 테스트를 먼저 수행하고 이 테스트를 통과할 수 있도록 실제 프로그램의 코드를 작성한다는 원리
  - > 리팩토링(Refactoring): 프로그램의 기능을 바꾸지 않으면서 중복제거, 단순화 등을 위해 시스템을 재구성한다는 원리
- **스크럼(SCRUM)**: 매일 정해진 시간, 장소에서 짧은 시간의 개발을 하는 팀을 위한 프로젝트 관리 중심 방법론
- **린(LEAN)**: 도요타의 린 시스템 품질 기법을 낭비 요소를 제거하여 품질을 향상시킨 방법론

## ● 객체지향 설계 원칙(SOLID)

1. 단일 책임의 원칙(SRP): 하나의 클래스 하나의 목적
2. 개방 폐쇄 원칙(OCF): 구성요소가 확장엔 열려있고, 변경엔 닫혀있어야함
3. 리스코프 치환의 원칙(LSP): 서브타입은 자신의 기반 타입으로 교체할 수 있어야함.
4. 인터페이스 분리의 원칙(ISP): 사용 X 인터페이스 구현 X
5. 의존성 역전의 원칙(DIP): 실제 사용관계는 바뀌지 않음, 관계를 느슨하게

## ● 객체지향 분석 방법론

- 1) **객체 모델링=정보모델링(Object, Information)**: 객체간의 관계를 정의하여 E-R 다이어그램을 만드는 과정까지의 모델링 -> 객체 다이어그램 활용해 표현
- 2) **동적 모델링(Dynamic)**: 시간의 흐름에 따라 객체들 사이에 제어흐름, 동작 순서등의 동적인 행위를 표현한 모델링 -> 상태 다이어그램 활용
- 3) **기능 모델링(Functional)**: 프로세스들을 자료 흐름을 중심으로 처리 과정을 표현하는 모델링 -> 자료흐름도(DFD)를 활용하여 표현

## ● 델파이 기법: 전문가의 경험적 지식을 통한 문제해결 및 미래 예측을 위한 기법

## ● 비용산정 모형

1. **LOC**: 낙관치, 중간치 비관치 측정->예측치
2. **Man Month**: 한 사람이 1 개월동안 할 수 있는 일의 양
  - >  $\text{Man Month} = \text{LOC} / \text{개발자의 월간생산성}$ ,  $\text{프로젝트 기간} = \text{Man Month} / \text{인력}$

-> Man Month: 한 사람이 1 개월 동안 할 수 있는 일의 양을 기준으로 프로젝트 비용을 산정하는 방식

3. **COCOMO**: organic semi detached, embedded

4. **푸트남**: 요구 인력의 분포 가정

5. **기능점수(FP)**: 요구 기능을 증가시키는 인자별로 가중치 부여후 점수 합산

## ● 일정 관리 모델

1. **주공정법(CPM)**: 가장 긴 일수

2. **PERT**: 비관치, 중간치, 낙관치의 3 점 추정 방식

● **소프트웨어 아키텍처**: 여러 가지 소프트웨어 구성요소와 그 구성요소가 가진 특성 중에 외부에 드러나는 특성, 그리고 구성요소 간의 관계를 표현하는 시스템의 구조와 구조체이다.

● **소프트웨어 아키텍처 패턴**: 소프트웨어를 설계할 때 참조할 수 있는 전형적인 해결방식

● **소프트웨어 아키텍처 프레임워크**: 시스템에서 아키텍처가 표현해야하는 내용 및 이들간의 관계를 제공하는 아키텍처 기술 표준

-> **계층화 패턴**: 시스템을 계층으로 구분하여 구성하는 패턴

-> **파이프-필터 패턴**: 서브 시스템이 입력 받아 처리하고 넘겨줌(데이터 스트림 생성 처리)

-> **브로커 패턴**: 컴포넌트 간의 통신을 조정하는 역할

-> **MVC 패턴**: 대화형 애플리케이션을 모델, 뷰, 컨트롤러 3 개의 서브 시스템으로 구조화하는 패턴, 이면에서 실행되는 비즈니스 로직을 서로 영향없이 쉽게 고칠 수 있는 패턴

● **SAAM**: 변경 용이성과 기능성에 집중, 경험이 없는 조직에서도 활용 가능한 비용평가 모델

## ● 4+1 뷰

1) **유스케이스뷰**: 유스케이스 또는 아키텍처를 도출하고 설계하며 다른 뷰를 검증하는 데 사용되는 뷰(사용자, 설계자, 개발자, 테스트 관점)

2) **논리 뷰**: 시스템의 기능적인 요구사항이 어떻게 제공되는지 설명해주는 뷰(설계자, 개발자 관점)

3) **프로세스 뷰**: 시스템의 비기능적인 속성으로서 자원의 효율적인 사용, 병행 실행, 비동기, 이벤트 처리 등을 표현한 뷰(개발자, 시스템 통합자 관점)

4) **구현 뷰**: 개발 환경 안에서 정적인 소프트웨어 모듈의 구성을 보여주는 뷰(컴포넌트 구조)

5) **배포 뷰**: 컴포넌트가 물리적인 아키텍처에 어떻게 배치되는가를 매핑해서 보여주는 뷰“

● **디자인 패턴**: 소프트웨어 공학의 소프트웨어 설계에서 공통으로 발생하는 문제에 대해 자주 쓰이는 설계 방법을 정리한 패턴

## 1. 목적에 따른 유형

- > 생성(Creational): 객체 인스턴스 생성
- > 구조(Structural): 클래스나 객체의 조합
- > 행위(Behavioral): 클래스나 객체들이 상호작용하는 방법

## 2. 범위에 따른 유형

- > 클래스(컴파일타임), 객체(런타임)

## ● 디자인 패턴 종류

### 1. 생성패턴

- 1) Builder: 객체를 생성하는 방법(과정)과 객체를 구현(표현)하는 방법을 분리
- 2) Prototype: 일반적인 원형을 만들어놓고, 필요한 부분만 수정하여 사용
- 3) Factory Method: 상위 클래스에서 인터페이스만 정의, 실제 생성은 서브 클래스가 담당
- 4) Abstract Factory: 구체적인 클래스에 의존하지 않고, 인터페이스 제공
- 5) Singleton: 전역변수 사용 X, 객체를 하나만 생성

### 2. 구조패턴

- 1) Bridge: 기능의 클래스 계층과 구현의 클래스 계층을 연결
- 2) Adaptor: 기존에 생성된 클래스를 재사용할 수 있도록 중간에서 맞춰주는 역할

### 3. 행위패턴

- 1) Template Method: 전체 일의 수행구조는 바꾸지 않으면서 특정단계 수행내역은 바꾸는 패턴=상위클래스는 추상메서드(골격), 하위클래스는 세부처리
- 2) Observer: 한 객체의 상태가 바뀌면 자동갱신
- 3) Strategy: 행위(알고리즘)를 클래스로 캡슐화해 동적으로 자유롭게 바꿀 수 있게 해줌
- 4) State: 상태에 따라 다르게 처리할 수 있도록 행위 내용 변경(코드 수정 최소화)
- 5) Command: 명령이 들어오면 그에 맞는 서브클래스가 선택되어 실행됨

● **요구공학**: 사용자의 요구가 반영된 시스템을 개발하기 위하여 사용자의 요구사항에 대한 ”도출, 분석, 명세, 확인 및 검증하는 구조화된 활동이다.

-> 기능적 요구사항: 시스템이 제공하는 기능, 서비스에 대한 요구사항

-> 비기능적 요구사항: 시스템이 수행하는 기능 이외의 사항, 시스템 구축에 대한 제약사항에 관한 요구사항

● **요구사항 개발 단계**(도출->분석->명세->확인)

● 요구사항을 분석하고 정의하는 단계에서 작성되는 최종산출물: **요구사항 명세서**

● **인스펙션**: 저작자 외에 다른 전문가 또는 팀이 검사하여 문제를 식별하고 해결을 찾아냄

● **형상통제 위원회:** 형상관리에 대한 주요 방침을 정하고 산출물을 검토하며, 단계별 의사결정을 수행하는 조직이다.

● **브레인스토밍:** 말을 꺼내기 쉬운 분위기

- **롤플레이팅:** 현실에 일어나는 장면을 설정하고 연기

- **인터뷰:** 이해관계자와 직접 대화를 통해 정보를 구하는 공식적, 비공식적 정보 수집 방법

- **동료 검토:** 2~3 명의 이해관계자들이 설명을 들으며 결함을 발견

- **워크스루:** 검토 자료를 회의 전에 배포해서 사전 검토한 후 개선, 학습 기회 제공(가장 비형식적)

- **인스펙션:** 저작자 외의 전문가 또는 팀이 검사하여 오류를 찾아냄”

● **비정형 명세기법(자연어 기반) / 정형 명세기법(수학적인 원리와 표기법)**

● **UI/UX**

- **UI:** 사용자 인터페이스, 사용자와 시스템 사이에서 의사소통할 수 있도록 고안된 물리적, 가상의 매개체

- **UX:** 사용자가 직/간접적으로 경험하면서 느끼고 생각하는 총체적 경험“

● **UI 유형**

→ **CLI(텍스트 기반):** 명령어를 텍스트로 입력하여 조작

→ **GUI(그래픽 기반):** 그래픽 환경을 기반으로 한 마우스나 전자펜 이용

→ **NUI(직관적사용자반응기반):** 키보드나 마우스 없이 신체부위 이용

→ **OUI(유기적 상호작용):** 모든 사물이 입출력장치 가능”

● **UI 설계원칙**

→ **직관성(Intuitiveness):** 누구나 쉽게 이해하고, 쉽게 사용할 수 있어야함

→ **유효성(Efficiency):** 정확하고 완벽하게 사용자의 목표가 달성될 수 있도록 제작

→ **학습성(Learnability):** 초보와 숙련자 모두가 쉽게 배우고 사용할 수 있게 제작

→ **유연성(Flexibility):** 사용자의 요구사항을 최대한 수용하고 실수를 방지할 수 있도록 제작”

● **UI 설계지침중 가시성:** 주요 기능을 메인 화면에 노출하여 쉬운 조작이 가능해야한다.

● **UI 품질 요구사항(ISO/IEC 9126 기신사효유이)**

: 기능성, 신뢰성, 사용성, 효율성, 유지보수성, 이식성

\*사용성

- > 이해성: 소프트웨어의 논리적인 개념과 적용가능성을 분간하는 데 필요한 사용자의 노력정도
- > 학습성: 소프트웨어 애플리케이션을 익히는데 필요한 사용자의 노력정도
- > 운용성: 소프트웨어 활용과 운용통제에 필요한 사용자의 노력

#### ● UI 품질 요구사항(ISO/IEC 9126 기반)

- > 적절성: 주어진 작업과 사용자의 목표에 필요 적절한 기능들을 제공
- > 정밀성: 요구되는 정확도로 결과를 산출

#### ● UI 화면설계 구분

- > **와이어프레임**: 이해관계자들과 화면구성을 협의하거나 서비스의 간략한 흐름을 공유하기 위해 화면 단위의 레이아웃을 설계하는 작업
- > **스토리보드**: 정책, 프로세스, 콘텐츠 구성, 와이어프레임(UI, UX), 기능 정의, DB 연동 등 서비스 구축을 위한 모든 정보가 담겨있는 설계산출물
- > **프로토타입**: 정적인 화면으로 설계된 와이어프레임 또는 스토리보드에 동적 효과를 적용하여 실제 구현된 것처럼 시뮬레이션 할 수 있는 모형

● **미들웨어**: 운영체제와 소프트웨어 애플리케이션 사이 위치, 원만한 통신이 이루어지도록 제어(Weblogic, Websphere, Jboss, Tomcat...)

● **목업**: 실제 제품이 나오기 전 만드는 모형(비용 절감 목적)

● **UML**: 객체지향 소프트웨어 개발과정에서 산출물을 명세화, 시각화, 문서화 할 때 사용되는 모델링 기술과 방법론을 통합하여 만든 표준화된 범용모델링 언어

#### ● UML 구성요소

사물, 다이어그램, 관계

\*관계: 사물의 의미를 확장하고 명확히 하는 요소, 사물과 사물을 연결하여 표현하는 요소

#### ● UML 관계

- 의존관계(Dependency): 하나의 클래스가 또 다른 클래스 사용
- 일반화 관계(Generalization): 개념화, 하나의 사물이 다른 사물에 비해 더 일반적인지 구체적인지를 표현



- 포함관계=복합관계(Composition): 포함하는 사물의 변화가 포함되는 사물에게 영향
- 집합관계(Aggregation): 추상화, 하나의 사물이 다른 사물에 포함
- +연관관계(Association), +실체화관계(Realization)

## ● UML 스테레오타입

<<include>>: 다른 유스케이스를 포함하는 관계

<<extend>>: 확장 관계

## ● \*용어정리

- 클래스: 공통의 속성, 연산(메서드)관계, 의미를 공유하는 객체들의 집합
- 인스턴스: 해당 클래스의 구조로 컴퓨터 저장 공간에서 할당된 실체
- 인터페이스: 기능을 모아놓은 클래스로 추상메서드와 상수만을 포함하는 추상클래스
- 속성: 특성에 해당하는 인스턴스가 보유할 수 있는 값의 범위를 기술하는 구성요소

## ● 구조적 다이어그램

- 클래스 다이어그램(연의일실폐집), 객체 다이어그램
- 컴포넌트 다이어그램: 컴포넌트와 그들 사이의 의존관계를 나타내는 다이어그램
- 배치 다이어그램, 복합체 구조 다이어그램
- 패키지 다이어그램: 서로 다른 패키지들 사이에 의존관계 표현

## ● 행위적 다이어그램=동적 다이어그램

- 유스케이스 다이어그램
- 시퀀스 다이어그램: 객체 간 상호작용을 메시지 흐름과 객체 사이 메시지를 보내는 시간을 표현하는 - 동적 다이어그램으로 객체, "생명선", 실행 메시지로 구성되어있다.
- 커뮤니케이션 다이어그램: 객체들이 메시지를 주고받으며, 시간의 흐름에 따라 상호작용 하는 과정을 표현
- 상태 다이어그램: 하나의 객체가 자신이 속한 클래스의 상태변화 또는 상호작용에 상태가 어떻게 변화하는지 표현
- 활동 다이어그램: 객체의 처리 로직이나 조건에 따른 처리의 흐름으로 순서대로 표현
- +타이밍 다이어그램

## ● 3C 분석: 고객 자사, 경쟁사 비교 분석

- 사용성테스트: 사용자가 직접 제품을 사용하면서 미리 작성된 시나리오에 맞추어 과제를 수행한 후, 질문에 답하도록 하는 테스트
- SWOT: 기업의 내부환경과 외부환경을 분석하여 강점, 약점, 기회, 위협 요인을 규정하고, 이를 토대로 경영전략을 수립하는 방법
- 시나리오 플래닝: 불확실성이 높은 상황 변화를 사전에 예측하고 다양한 시나리오를 설계하는 방법
- 워크숍: 소집단 정도의 인원으로 연구회 및 세미나
- 프로토타이핑 도구: UX 핀, 액슈어, 네이버 프로토나우

## ● 데이터 모델 표시요소

- 연산: DB에 저장된 실제 데이터를 처리하는 작업에 대한 명세
- 구조: DB에 논리적으로 표현될 대상으로서의 개체 타입과 개체 타입들 간의 관계
- 제약조건

## ● 데이터 모델 절차

요구사항분석

- > 개념적 설계: 추상적, 개념적 표현 주요산출물=E-R 다이어그램
- > 논리적 설계: 스키마 설계, 트랜잭션 인터페이스 설계, 정규화과정 수행
- > 물리적 설계: DBMS의 특성과 성능 고려 결과로 나오는 명세서로 테이블 정의서

● **정규화**: 데이터의 중복성을 제거하며, 이상현상을 방지

● **반정규화=비정규화**: 정규화된 엔터티, 속성, 관계에 대해 성능 향상과 개발 운영의 단순화를 위해 중복, 통합 분리 등을 수행하는 데이터 모델링 기법

ex) 1:1 관계 1:M 관계를 통합해 조인횟수를 줄여 성능 향상=테이블 통합(반정규화 기법)

● 튜플(행)->튜플수=카디널리티 / 속성(열)->속성수=차수

릴레이션=테이블 / 스키마: 데이터베이스의 구조, 제약조건등의 정보를 담고 있는 기본적인구조

인스턴스: 정의된 스키마에 따라 생성된 테이블에 실제 저장된 데이터의 집합

● **관계 대수 연산자**(대절해비=관계대수는 절차적 관계해석은 비절차적)

● **일반 집합 연산자**->합교차카(un-x)

● **순수 관계 연산자**

- 1) 선택(select): 릴레이션 R에서 조건을 만족하는 튜플 반환(행)

- 2) 프로젝트(project): 릴레이션 R에서 주어진 속성들의 값으로만 구성된 튜플 반환(열)
- 3) 조인(join): 공통 속성을 이용해 R과 S의 튜플들을 연결해 만들어진 튜플 반환
- 4) 디비전(division): B 조건에 맞는 A 튜플 반환

## ● 논리 데이터 모델링 속성(개속관)

“개체: 실체 / 속성: 구체적 항목 / 관계: 개체간의 대응관계“

● **E-R 모델:** 데이터와 그들간의 관계를 사람이 이해할 수 있는 형태로 표현하기 위해 가장 널리 사용되는 모델

\* E-R 다이어그램은 참고..

● **이상현상:** 데이터의 중복성으로 인해 릴레이션을 조작할 때 발생하는 비합리적 현상

- > 삽입 이상: 정보저장시 해당 정보의 불필요한 세부 정보를 입력해야 하는 경우
- > 삭제 이상: 정보삭제시 원치 않는 다른 정보가 같이 삭제되는 경우
- > 갱신 이상: 중복 데이터 중에서 특정 부분만 수정되어 중복된 값이 모순을 일으키는 경우

● **ERD:** 업무 분석 결과로 도출된 개체와 개체 간의 관계를 도식화한 다이어그램이다.

ER 모델 요소: 개체(사각형), 속성(타원형)

## ● 정규화 단계

- 1) 1차 정규화: 원자값으로 구성(값 여러개)
- 2) 2차 정규화: 부분함수 종속 제거(완전 함수적 종속관계)
- 3) 3차 정규화: 결정자 후보키가 아닌 함수 종속 제거
- 4) 보이스코드 정규화: 결정자가 후보키가 아닌 함수 종속 제거
- 5) 4차 정규화: 다치(다중값) 속성 제거
- 6) 5차 정규화: 조인 종속성 제거

## ● 함수 종속

- 1) 부분함수 종속(partial):  $X \twoheadrightarrow Y(\{\text{학번}, \text{과목번호}\} \twoheadrightarrow \text{성적})$
- 2) 완전함수 종속(full):  $X \rightarrow Y(\text{학번} \rightarrow \text{학년})$
- 3) 이행함수 종속(transitive):  $X \rightarrow Y, Y \rightarrow Z$

## ● 데이터베이스 무결성

참조 무결성: 외래 키가 참조하는 다른 개체의 기본 키에 해당하는 값이 기본 키 값이나 NULL 이어야 하는 제약 조건

## ● 데이터베이스 특성(실시간 접근성, 지속적인 변화, 동시 공유, 내용 참조)

## ● 데이터베이스(통합된, 저장된, 운영, 공유 데이터임)

● 빅데이터: 시스템, 서비스, 조직 등에서 주어진 비용, 시간 내에 처리 가능한 데이터 범위를 넘어서는 크기의 비정형 데이터

## ● 키

- 기본 키: 각 튜플들을 고유하게 식별하는 컬럼
- 대체 키: 후보 키 중에서 기본 키로 선택되지 않은 키
- 후보 키: 유일성 최소성 만족하는 키
- 슈퍼 키: 유일성은 만족하지만, 최소성은 만족하지 못하는 키
- 외래 키: 한 릴레이션의 컬럼이 다른 릴레이션의 기본 키로 이용되는 키

## ● 물리 데이터 모델링: 논리 모델을 적용하고자 하는 기술에 맞도록 상세화해 가는 과정

## ● 관계형 DB 에서 인덱스: 검색 연산의 최적화를 위해 DB 내 열에 대한 정보를 구성한 데이터 구조

● 파티셔닝: 레인지(연속적인 숫자나 날짜), 해시(해시함수), 리스트(명시적 제어 ㄱㄴ), 컴포지트(두개 이상 결합), 라운드로빈

● DBMS: 데이터 관리의 복잡성을 해결하는 동시에 데이터 추가, 변경, 검색, 삭제 및 백업 등의 기능을 지원하는 소프트웨어이다.(특징: 데이터 “무결성, 일관성, 회복성, 보안성, 효율성”)

## ● DBMS 유형(키컬도그)

- 키-값 DBMS: 키 기반 Get/Put/Delete 제공
- 컬럼 기반 데이터 저장 DBMS: Key 안에 (column/value)로 조합된 여러개 필드를 가짐
- 문서 저장 DBMS: 값의 데이터 타입이 문서라는 타입을 사용
- 그래프 DBMS: 시멘틱 웹과 온톨로지 분야에서 활용되는 그래프로 데이터 표현

● 빅데이터 특성: 3V(Volume, Variety, Velocity): 처리할 수 없는 범위(PB) 비정형 데이터

● **데이터베이스**: 다수의 인원, 시스템 또는 프로그램이 사용할 목적으로 통합하여 관리되는 데이터의 집합(계층형 DB 관리시스템: 상하 종속관계 / 관계형 DB 관리시스템: 보편적)

● **시멘틱웹**: 온톨로지의 의미적 상호 운용성을 이용해서 서비스 검색, 조합, 중재 기능을 자동화하는 웹

● **온톨로지**: 실세계에 존재하는 모든 개념들과 개념들의 속성, 그리고 개념들간의 관계 정보를 컴퓨터가 이해할 수 있도록 서술해놓은 지식베이스이다.

● **데이터마이닝**: 대규모로 저장된 데이터 안에서 체계적이고 자동적으로 통계적 규칙이나 패턴을 찾아내는 기술

\* 데이터마이닝 주요기법

1) 분류규칙: 과거 데이터로부터 분류모형

2) 연관규칙: 항목들 간의 종속관계

3) 연속규칙: 연관 규칙에 시간 관련 정보가 포함된 기법

● **텍스트마이닝**: 대량의 텍스트 데이터로부터 패턴 또는 관계를 추출하여 의미 있는 정보를 찾아내는 기법

● **웹 마이닝**: 웹으로부터 얻어지는 방대한 양의 정보로부터 유용한 정보를 찾아내기 위하여 분석“

● **HDFS**: 대용량 데이터의 집합을 처리하는 응용프로그램에 적합하도록 설계된 분산 파일시스템

● **맵 리듀스**: 구글에서 대용량 데이터 처리를 분산 병렬 컴퓨팅에서 처리하기 위한 목적으로 제작해 2004년에 발표한 소프트웨어 프레임워크(맵:키-값을 읽어 필터링하거나 변환, 리듀스: 맵을 통해 출력된 값으로 새로운 키 기준으로 그룹화 후 집계연산 결과 출력)

● **NoSQL**: 데이터 저장에 고정된 테이블 스키마가 필요하지 않고 조인 연산을 사용할 수 없으며 수평적으로 확장이 가능한 DBMS 이다.(Key-value, Column Family Data Store, Document Store, Graph Store)

● **NoSQL 특성**(Basically available=언제든지 접근, Soft-State=내부 X, 외부에서 전송된 정보를 통해 결정, Eventually Consistency=일정 시간이 지나면 데이터의 일관성이 유지)”

● **DB 링크**: DB 에서 제공하는 DB 링크 객체 이용

● **DB 연결을 위한 커넥션 풀**: DB 와 연결된 커넥션을 미리 만들어서 풀 속에 저장해두고 필요할 때 커넥션을 풀에서 가져와쓰고 다시 풀에 반환하는 기법

● **JDBC**: 데이터베이스에 접속할 수 있는 자바 API 를 사용하는 연계 방식  
현재 페이지에서 다른 부분으로 가거나 다른 페이지로 이동: [하이퍼링크](#)

● **EAI**: 기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션 간의 정보를 전달, 연계, 통합이 가능하도록 해주는 솔루션(포터메하 point to point(1:1), hub&spoke(성형), message bus(버스형), hybrid(하이브리드형)), 어댑터(데이터 입출력 도구), 브로커(포맷과 코드 변환) 사용, 메시지 큐(비동기 메시지 송수신) 사용“

● **ESB**: 기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션 간을 하나의 시스템으로 관리 운영할 수 있도록 “서비스 중심의 통합“을 지향하는 연계방식”

● **HTTP(GET, POST, PUT 사용), Hypertext, HTML**

● **IPC**: 운영체제에서 프로세스 간 서로 데이터를 주고받기 위한 통신기술

● **API**: 응용프로그램에서 사용할 수 있도록 운영체제나 프로그래밍 언어가 제공하는 기능을 제어할 수 있게 규격을 정해놓은 인터페이스

● **SOAP**: HTTP, HTTPS, SMTP 등을 사용하여 XML 기반의 메시지를 네트워크 상태에서 교환하는 프로토콜이다.

● **WSDL**: 웹 서비스명, 제공 위치, 메시지 포맷, 프로토콜 정보 등 웹 서비스에 대한 상세 정보가 기술된 XML 형식으로 구현되어 있는 언어

● **UDDI**: 웹 서비스에 대한 정보인 WSDL 을 등록하고 검색하기 위한 저장소로 공개적으로 접근, 검색이 가능한 레지스트리이자 표준이다.

● **세션**: 네트워크를 경유하는 프로세스 간 통신의 접속점이고, IP Address 와 Port 넘버가 합쳐진 형태로 네트워크상에서 서버프로그램과 클라이언트 프로그램이 통신할 수 있는 교환기술

● **REST**: 웹과 같은 분산 하이퍼미디어 환경에서 자원의 존재/상태 정보를 표준화된 HTTP 메서드로 주고받는 소프트웨어 아키텍처(리소스, 메서드, 메시지)->자원, 처리, 메시지 그림

● **JSON**: 속성-값 쌍 또는 키값쌍으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 개방형 표준 포맷(자료형중 객체 O)

● **AJAX**: HTML 만으로 어려운 다양한 작업을 웹 페이지에서 구현한다.

(주요 기술: XMLHttpRequest(메서드가 데이터를 전송), DOM(트리 구조의 형태로), XSLT(W3C 에서 재정한 표준 Xpath)

● **XML**: 송수신 시스템 간 데이터 연계의 편의성을 위해 전송되는 데이터의 구조를 동일한 형태로 정의하고 인간과 기계가 모두 이해할 수 있는 텍스트 형태로 정의, HTML 의 단점을 보완한 인터넷 언어로 특수한 목적을 갖는 마크업 언어

● **xUnit**: 자바, C++, .Net 등 다양한 언어를 지원하는 단위테스트 프레임워크(java->JUnit)

● **STAF**: 서비스 호출, 컴포넌트 재사용 등 다양한 환경 지원하는 테스트 프레임 워크

● **FitNess**: 웹기반 테스트 케이스 설계/실행/결과 확인 등을 지원

● **NTAF**: FitNess + STAF

● **Selenium**: 다양한 브라우저 지원 및 개발언어를 지원, 테스트 스크립트 언어 학습할 필요없음, 웹 애플리케이션 테스트 프레임워크

● **watir**: 루비 기반 웹 애플리케이션 테스트 프레임워크

● **스카우터(SCOUTER)**: 애플리케이션에 대한 모니터링 및 DB Agent 를 통해 오픈 소스 DB 모니터링 가능

● **제니퍼(Jennifer)**: 애플리케이션 개발부터 테스트, 오픈, 운영, 안정화까지 전 생애주기 단계 동안 성능을 모니터링하고 분석

● **IPSec**: 네트워크계층에서 무결성과 인증을 보장하는 인증헤터와 기밀성을 보장하는 암호화를 이용하여 양 종단 간의 보안 서비스를 제공하는 터널링 프로토콜(인증(AH), 암호화(ESP, 송신처 인증과 기밀성), 키 관리(IKE)프로토콜로 구성)

● **SSL/TLS**: 443 포트 이용, 전송계층과 응용계층 사이 데이터 암호화(HandShake Protocol)

S-HTTP: 웹 상에서 네트워크 트래픽을 암호화하는 주요방법

● 시큐어 코딩 가이드 -> 블록문 내에서만 재귀함수 호출되도록 설정

공개키와 비밀키를 사용하는 알고리즘: 비대칭 키 알고리즘

DB 암호화 방식(1. API 방식(애플리케이션 레벨), 2. Plug-in 방식(확장성 프로시저) 3. TDE 방식(내장된 암호화 기능 4. Hybrid 방식))

## ● 해시 알고리즘(SHA-256, HAS-160)

● 형상통제: 형상 항목의 형상 관리를 위해 형상통제위원회를 운영한다.

Linux 위에서 구동하며 자바 및 코틀린으로 개발 -> 안드로이드

● 형상관리 절차(식통감기 -> 형상 식별(관리번호)-형상 통제(형상통제위원회)-형상 감사(무결성 평가)-형상 기록(수행결과)), 베이스라인: 각 단계별 산출물을 검토, 평가, 조정, 처리등의 변화를 통제하는 시점의 기준

## ● 개발도구의 분류

빌구테형 -> 빌드도구(의존성 관리)-구현도구(디버깅, 수정)-테스트도구(검증)-형상관리도구(버전관리)“

## ● 서버 종류(웹 서버, 웹 애플리케이션 서버, 데이터베이스 서버, 파일 서버)

- 아파치=웹 서버/ 톰캣=웹 애플리케이션 서버

## ● 형상관리 도구

1. 공유폴더방식(RCS, SCCS)-공유 폴더에 복사하는 방식
2. 클라이언트/서버 방식(CVS(가장 오래됨), SVN)-중앙집중형 서버 저장소, 클라이언트-서버 방식
3. 분산 저장소 방식(Git)-로컬 저장소와 원격 저장소로 분리되어 분산 저장하는 방식

## ● 응집도와 결합도

1. 응집도(우논시절통(교)순기)-> 낮은데서 높아짐: 하나의 모듈은 하나의 기능을 수행

1) Coincidental Cohesion(우연적 응집도): 모듈 내부의 구성요소가 각 “연관이 없을 경우”

2) Logical Cohesion(논리적 응집도): “유사한 성격, 특정 형태”로 분류되는 처리 요소들이 한 모듈에서 처리되는 경우

3) Temporal Cohesion(시간적 응집도): “특정 시간”에 처리 되어야 하는 활동들

4) Procedural Cohesion(절차적 응집도): 모듈이 다수의 관련 기능을 갖고, 모듈 안의 구성요소들이 그 기능을 ”순차적“으로 수행할 경우



5) **Communication Cohesion(교환적=통신적 응집도)**: “동일한 입력과 출력”을 사용해 다른 기능을 수행하는 활동들의 모임

6) **Sequential Cohesion(순차적 응집도)**: “모듈 내 한 활동으로부터 나온 출력값”을 “다른 활동”이 사용할 경우

7) **Functional Cohesion(기능적 응집도)**: 모듈 내부의 “모든 기능이 단일한 목적”을 위해 수행되는 경우

## 2. 결합도(내공외제스자)-> 높은데서 낮아짐: 모듈 간의 상호의존도, 관련성

1) **Content Coupling(내용 결합도)**: ”다른 모듈 내부에 있는 변수“나 다른 모듈에서 사용하는 경우

2) **Common Coupling(공통 결합도)**: 파라미터가 아닌 모듈 밖에 선언되어 있는 ”전역 변수“를 참조하고 갱신하는 식으로 상호작용하는 경우

3) **External Coupling(외부 결합도)**: 두 개의 모듈이 외부에서 도입된 ”데이터 포맷, 통신 프로토콜, 또는 디바이스 인터페이스“를 공유할 경우

4) **Control Coupling(제어 결합도)**: 어떻게 ”처리를 해야 한다는 제어 요소“가 전달되는 경우

5) **Stamp Coupling(스탬프 결합도)**: 모듈 간의 인터페이스로 ”배열이나 객체, 구조“ 등이 전달되는 경우

6) **Data Coupling(자료 결합도)**: 모듈 간의 인터페이스로 전달되는 ”파라미터“를 통해서만 모듈 간의 상호 작용이 일어나는 경우

● **배치프로그램**: 일련의 작업들을 작업 단위로 묶어 정기적으로 반복수행하거나 일괄처리

● Cron 표현식(퀴츠 크론)(초분시일 월요일)[\*: 모든수, ?: 미사용, ,:특정 기간 -:기간 설정, /:반복간격, L: 마지막 기간, W: 가장 가까운 평일, #: 몇 번째 주, 요일 설정]  
시스템을 분해하고 추상화->모듈화

## ● 정보보안 3대 요소

- 1) 기밀성: 인가되지 않은 개인 혹은 시스템 접근에 따른 정보 공개 및 노출을 차단
- 2) 무결성: 인가된 사용자에 대해서만 자원 수정이 가능하며 전송중인 정보는 수정되지 않음
- 3) 가용성: 인가된 사용자는 가지고 있는 권한 범위 내에서 언제든지 자원 접근이 가능

## ● DoS: 시스템을 악의적으로 공격해 해당 시스템의 자원을 부족하게 해 사용하지 못하게 하는 공격

-> **SYN 플러딩**: TCP 프로토콜의 구조적 문제를 이용한 공격, SYN 패킷만 보냄

-> **UDP 플러딩**: 대량의 UDP 패킷을 만들어 임의의 포트 번호로 전송

-> **스머프, 스머핑**: 출발지 주소를 공격 대상의 IP로 설정하여 네트워크 전체에게 ICMP Echo 패킷을 직접 브로드캐스팅해 마비시킴

-> **죽음의 핑(PoD)**: ICMP 패킷을 정상적인 크기보다 아주 크게 만들어 전송

-> **랜드 어택**: 출발지 IP와 목적지 IP를 같은 패킷 주소로 만들어 보내, 수신자가 자기 자신에게 응답

-> **티어 드롭:** IP 패킷의 재조합 과정에서 잘못된 Fragment Offset 정보로 인해 수신시스템이 단편화된 패킷의 재조합 과정에서 문제를 발생하도록 하는 DoS 공격(IP 헤더가 조작된 일련의 IP 패킷 조각들을 전송)

-> **퐁크:** 같은 시퀀스 번호 계속 보냄

-> **보잉크:** 일정한 간격으로 시퀀스 번호에 빈 공간 생성

### ● DDoS: 여러 대의 공격자를 분산 배치해 동시에 동작하게 함으로써 특정 사이트 공격

-> HAMAD (Handler, Agent, Master, Attacker, Daemon) [Slow Read Attack: TCP 윈도우 크기 낮게, Hulk DoS: 공격자가 웹 페이지 주소를 지속적으로 변경하면서 다량으로 GET 요청 발생]

● **DRDoS:** 서버역할을 할 수 있는 단말 장비까지 공격기기로 이용하여 공격 근원지 파악이 어렵다.

● **HTTP GET 플러딩:** 과도한 GET 메시지(애플리케이션 공격)

● **Slowloris:** HTTP GET 메시지를 사용하여 공격

● **RUDY 공격:** (Content-Length: 9999999 설정 이후 1 바이트 씩 전송하여 지속적인 연결 유지를 통해 가용 자원을 소진시키는 공격)

### ● 네트워크 공격

- **스니핑(Sniffing):** 공격대상에게 직접 공격 하지 않고 데이터만 몰래 들여다보는 수동적 공격

- **네트워크 스캐너(Scanner), 스니퍼(Sniffer):** 공격자가 HW, SW 구성의 취약점을 탐색하는 공격 도구

- **IP 스푸핑(Spoofing):** 침입자가 인증된 컴퓨팅 시스템인 것처럼 속여 정보를 빼내기 위해 본인의 패킷 헤더를 인증된 호스트의 IP 어드레스로 위조해 타겟에 전송

- **ARP 스푸핑(Spoofing):** 공격자가 특정 호스트의 MAC 주소를 자신의 MAC 주소로 위조한 ARP Reply 를 만들어 희생자에게 지속적으로 전송

- **ICMP Redirect 공격:** 스니핑 시스템을 네트워크에 존재하는 또 다른 라우터라고 알림으로써 패킷의 흐름을 바꾸는 공격 기법(3 계층)

- **트로이 목마(Trojan Horses):** 악성 루틴이 숨어있는 프로그램, 겉보기에는 정상적인 프로그램처럼 보이지만 실행하면 악성 코드를 실행하는 프로그램

### ● 패스워드 크래킹/공격>Password Cracking)

1. **사전 공격(Dictionary):** ID 와 패스워드가 될 가능성이 있는 단어를 파일로 만들어놓음

2. **무차별 대입 공격(Brute Force):** 패스워드로 사용될 수 있는 영문자, 숫자, 특수문자등을 무작위로 패스워드 자리에 대입해 패스워드를 알아내는 공격 기법

3. **패스워드 하이브리드 공격:** 사전 + 무차별 대입

4. **레인보우 테이블 공격:** 패스워드 별로 해시 값을 미리 생성해 테이블에 모아놓음

● **버퍼 오버플로우 공격:** 메모리에 할당된 버퍼 크기를 초과하는 양의 데이터를 입력해 공격

● **3A(인증("authentication"), 권한 부여(Authorization), 계정 관리(Accounting))**

-> 스택실드(RET), 스택가드(카나리), ASLR(난수)

● **주요 시스템 보안 공격 기법**

- **포맷 스트링 공격:** 포맷 스트링을 인자로 하는 함수의 취약점
- **레이스 컨디션 공격:** 실행되는 프로세스가 임시파일을 만드는 경우 악의적인 프로그램을 통해 그 프로세스의 실행중에 끼어들어 임시파일을 심볼링 링크하여 악의적 공격

● **접근 통제 보호 모델**

1. 벨-라파둘라 모델: 기밀성
2. 비바 모델: 무결성

● **인증 관련 기술**

- SSO: 한 번의 인증 과정으로 여러 컴퓨터 상의 자원을 이용할 수 있도록 해주는 인증 기술
- 커버로스: 1980년대 중반 MIT의 Athena 프로젝트의 일환으로 개발

● **보안 관련 용어**

- **스피어피싱:** 사회 공학의 한 기법, 메일을 이용한 공격 기법
- **스미싱:** 문자를 이용한 공격 기법
- **큐싱:** 큐알 코드를 이용한 공격 기법
- **APT 공격:** 특정 타깃을 목표로 해 다양한 수단을 통한 지속적이고 지능적인 맞춤형 공격 기법
- **랜섬웨어(Ransomware):** 감염된 시스템의 파일들을 암호화해 인질처럼 잡고 몸값을 요구하는 악성 소프트웨어
- **이블 트윈 공격:** 무선 WiFi 피싱 기법, 합법적인 WiFi 제공자처럼 행세해 정보를 탈취하는 무선

● **네트워크 기법**

- **공급망 공격(Supply Chain Attack):** 소프트웨어 개발사의 네트워크에 침투하여 소스 코드의 수정을 통해 악의적인 코드를 삽입해 사용자 PC에 소프트웨어를 설치 또는 업데이트시 자동적으로 감염되도록 하는 공격

- **봇넷:** 악성 프로그램에 감염되어 악의적인 의도로 사용될 수 있는 다수의 컴퓨터들이 네트워크로 연결된 형태
- **사회공학:** 사람들의 심리와 행동 양식을 교묘하게 이용
- **제로데이 공격:** 보안 취약점이 공표되기 전 해당 취약점의 악용이 이루어짐
- **웜:** 스스로복제 / 악성 봇: 스스로 실행되지 못하고 해커의 명령에 의해 원격에서 제어 또는 실행가능한 프로그램
- **사이버 킬체인:** APT 공격 방어 분석 모델
- **트러스트존:** 독립 보안구역을 두어 중요한 정보를 보호하는 하드웨어 기반 보안기술
- **타이포스쿼팅:** 유사한 유명 도메인을 미리 등록하는 일
- **서버 접근 통제 유형->**Identification(식별=누구라고 밝히는 행위)
- **DAC:** 시스템에 대한 접근을 사용자/그룹의 신분 기반으로 제한(임의적)
- **MAC:** 시스템에 대한 접근을 시스템 정보의 허용등급을 기준으로 제한(강제적)
- **RBAC:** 시스템에 대한 접근을 조직 내 맡은 역할을 기반으로 제한(역할기반)

● **대칭 키 암호화 알고리즘:** 암호화 복호화에 같은 암호 키를 씀.

- **DES:** 1975 년 미국 연방 표준국(NIST)에서 발표, IBM 에서 개발
- **SEED:** 1999 년 한국인터넷진흥원(KISA) 개발
- **AES:** 2001 년 미국 표준 기술 연구소(NIST)에서 발표, 128 192 256bit 로 나눠짐 DES 의 상위호환
- **ARIA:** 2004 년 국가정보원과 산학연구협회가 개발
- **IDEA:** DES 를 대체하기 위해 스위스 연방기술 기관(Xuejia Lai 와 James Messey)에서 개발, 128 비트의 키를 사용해 64 비트 암호문을 만든다.
- **LFSR:** 선형 되먹임 시프트 레지스터, 선형함수로 계산
- **Skipjack:** 미 국가안보국에서 개발한 Clipper 칩에 내장된 블록 알고리즘

● **비대칭키(=공개키) 암호화 알고리즘**

- **디피-헬만(Diffie-Hellman):** 최초의 공개키 알고리즘
- **RSA:** 1977 년 3 명의 MIT 수학 교수가 고안한 큰 인수의 곱을 소인수 분해하는 수학적 알고리즘 이용
- **ECC(Elliptic Curve Cryptography):** 유한체 위에서 정의된 타원곡선 군에서의 이산대수의 문제에 기초한 공개키 암호화 알고리즘
- **ElGamal:** RSA 와 유사, Elgamal 이 1984 년 제안함 이산대수의 계산이 어려운문제를 기본원리

● **해시 암호화 알고리즘**

- **MD5:** 1991 년 MD4 를 개선한 알고리즘, 프로그램이나 파일의 무결성 검사에 사용
- **SHA-1:** 1993 년 NSA 에서 미 정부 표준으로 지정, DSA 에서 사용, 해시 값 생성

- **SHA-256/384/512**: AES의 키 길이인 128 192 256 비트에 대응하도록 길이를 늘인 해시 알고리즘
- **HAS-160**: KISA에서 국내 표준 서명 알고리즘(KCDSA)을 위해 개발된 해시함수(MD5+SHA-1)
- **HAVAL**: 메시지를 1024bits 블록으로 나누고 메시지 다이제스트를 출력

## ● 애플리케이션 공격

- HTTP GET 플러딩: 과도한 Get 메시지를 이용하여 웹 서버 과부하를 유발

## ● 비즈니스 연속성 계획 관련 주요 용어

- **BIA(Business Impact Analysis)**: 장애나 재해로 인한 운영상의 주요 손실을 볼 것을 가정하여 비즈니스 영향 분석
- **RTO(Recovery Time Objective)**: 업무중단 시점부터 업무가 복구되어 다시 가동될 때까지의 시간 (핫 사이트: 동일 정보기술 / 웜 사이트: 부분적 정보기술 / 콜드 사이트: 확보 X)
- **RPO(Recovery Point Objective)**: 업무중단 시점부터 데이터가 복구되어 다시 정상 가동될 때 데이터의 손실 허용 시점
- **DRP(Disaster Recovery Plan)**: 재난으로 장기간에 걸쳐 시설의 운영이 불가능한 경우를 대비한 재난 복구 계획
- **DRS(Disaster Recovery System)**: 재해 복구 센터

- **OTP**: 로그인 할 때마다 그 세션에서만 사용할 수 있는 일회성 패스워드

## ● 사설망=VPN / 가상 네트워크=토르 네트워크

- 캡슐화 취약점(세션), 코드 오류 취약점(널 포인터 역참조)

- **XSS(Cross Site Scripting)**: 사용자가 해당 웹페이지를 열람함으로써 웹페이지에 포함된 부적절한 스크립트가 실행되는 공격(Stored, Reflected, DOM)

- **사이트 간 요청 위조(CSRF)**: 사용자가 자신의 의지와는 무관하게 공격자가 의도한 행위를 특정 웹사이트에 요청하게 하는 공격

- **SQL Injection**: 악의적인 SQL 구문 삽입(Form, Union, Stored Procedure, Mass(대량), Error-Based, Blind(참과 거짓을 통해 의도하지 않은 SQL 실행))

- 콘텐츠 유출 방지 보안 솔루션(데이터 유출방지(DLP), 디지털 저작권 관리(DRM))

● 웹 방화벽(WAF): 웹 애플리케이션 보안

● 네트워크 접근 제어(Network Access Control, NAC)

● 인증 기술 유형(지소생특)

-> 지식기반 인증(패스워드), 소지기반 인증(공인인증서, OTP), 생체기반 인증(지문), 특징(=행위)기반 인증(서명, 발걸음 몸짓)

● 필기노트 (기출 같은 거 푸시면서 정리하신 듯)

웹브라우저 간 HTML 문법이 호환되지 않는 문제와 SGML의 복잡함을 해결하기 위하여 개발된 다목적 마크업 언어이다. 보통 SOAP이나 XML 기반의 웹 서비스 프로토콜이 사용되며, 웹 서버의 응답을 처리하기 위해 클라이언트 쪽에서는 자바스크립트를 쓴다. -> XML

속성-값 쌍(attribute-value pairs)으로 이루어진 데이터 오브젝트를 전달하기 위해 사용하는 개방형 표준 포맷이다. -> JSON

같은 테스트 케이스를 가지고, 테스트를 계속해서 반복하는 경우 어느 시점부터는 더 이상 결함을 발견하지 못한다. -> 살충제 패러독스

비상사태 또는 업무중단 시점으로부터 업무가 복구되어 다시 정상가동 될 때까지의 시간 -> RTO

비동기식 자바스크립트 XML을 의미하는 용어로, 클라이언트와 웹서버 간에 XML 데이터를 내부적으로 통신하는 대화식 웹 애플리케이션의 제작을 위해 사용된다. -> AJAX

작업 중 문제가 발생했을 때 트랜잭션의 처리과정에서 발생한 변경사항을 취소하고 이전의 데이터로 돌리는 명령어 -> ROLLBACK

애플리케이션을 실행하지 않고, 소스 코드에 대한 코딩 표준, 코딩 스타일, 코드 복잡도 및 남은 결함을 발견하기 위하여 사용하는 테스트 자동화 도구 유형 -> 정적분석

HTTP 등의 프로토콜을 이용하여 XML 기반의 메시지를 교환하는 프로토콜 -> REST(Representational State Transfer)

전세계 오픈된 정보를 하나로 묶는 방식 -> Linked Open Data

결과의 변경없이 코드의 구조를 재조정하는 것으로 가독성을 높이고, 유지보수를 쉽게하기 위한 목적 -> **리팩토링**

TCP/IP 에서 신뢰성없는 IP 를 대신하여 송신측으로 네트워크의 IP 상태 및 에러 메시지를 전달해주는 프로토콜 -> **ICMP**

객체가 생성될 때 자동으로 호출되는 메서드 -> **생성자**

네트워크 중간에서 패킷정보를 엿보다 -> **스니핑**

IP 패킷에서 외부의 공인 IP 주소와 포트 주소에 해당하는 내부 IP 주소를 재기록하여 라우터를 통해 네트워크 트래픽을 주고받는 기술 -> **NAT**

인가된 사용자는 원할 때 정보에 접근이 가능해야 함 -> **가용성**

웹 서비스명, 제공 위치, 메세지 포맷, 프로토콜 정보 등 웹 서비스에 대한 상세 정보가 기술된 XML 형식으로 구성된 언어 -> **WSDL**

인터페이스간 시스템 -> **통합테스트**

## **데이터 모델 구성요소**

개체 데이터 모델에서 실제 데이터를 처리하는 작업 -> **연산**

논리 데이터 모델에서 어떻게 나타낼 것인지 -> **구조**

데이터 무결성 유지를 위한 데이터베이스의 보편적 방법 -> **제약조건**

내용: 내부 동작 의존 / 공통: 전역 의존 / 외부: 외부 프로토콜 함께 의존 / 제어: 제어권 의존 / 스텝프: 배열 의존 / 자료: 파라미터 정도로만 의존

세션 관리 취약점을 이용한 공격 기법으로, '세션을 가로채다' 라는 의미 -> **세션 하이재킹**

네트워크 장치를 필요로하지 않고 네트워크 토폴로지가 동적으로 변화되는 네트워크 -> **애드혹**

모두 반영되거나 아니면 전혀 반영되지 않아야 하는 특성 -> **원자성**

1NF 2NF 3NF BCNF 4NF 5NF 제 12345 정규화 보이스코드 정규화

## **패킷교환 방식**

목적지 호스트와 미리 연결한 후, 통신하는 연결형 교환 방식 -> **가상회선 방식**

헤더에 붙어서 개별적으로 전달하는 비연결형 교환 방식 -> **데이터그램 방식**

접근한 데이터에 대한 연산을 모두 마칠때까지 상호배제하는 기법 -> **로킹**

데이터 링크 상의 프로토콜인 arp 이용 -> **arp spoofing**

데이터를 목적지까지 가장 안전하고 빠르게 전달하는 기능 -> **네트워크 계층**

## 테스트케이스 구성요소

- > 테스트 조건, 테스트 데이터, 예상 결과
- > 사용자 초기 화면/사용자 아이디비번/로그인성공

## 객체지향 추상화

- > 클래스들 사이의 전체 또는 부분 같은 관계를 나타내는 것이고=Aggregation(집합관계)
- > 한 클래스가 다른 클래스를 포함하는 상위 개념일 때 IS-A 관계라하며, 일반화 관계로 모델링한다=Generalization(일반화관계)

입력 자료 간의 관계와 출력에 영향을 미치는 상황을 체계적으로 분석 후 효용성이 높은 테스트 케이스를 선정해서 테스트하는 기법 -> **블랙박스 테스트**

```
int * arr[3] = {&a, &b, &c};
```

-> \*\*arr = arr 의 주소 안의 a 주소 안의 값

java에서는 true, false 를 비트연산자로 연산할때는 0,1 로 연산하고 true, false 를 반환

$7 == b \wedge c \neq a$  -> 관계 먼저 비트 나중

부모(상위) 클래스에 알려지지 않은 구체 클래스를 생성하는 패턴이며, 자식(하위) 클래스가 어떤 객체를 생성할지를 결정하도록 하는 패턴이기도 하다. 부모(상위) 클래스 코드에 구체 클래스 이름을 감추기 위한 방법으로 사용한다. -> **Factory Method**

이 다이어그램은 문제 해결을 위한 도메인 구조를 나타내어 보이지 않는 도메인 안의 개념과 같은 추상적인 개념을 기술하기 위해 나타난 것이다. 또한 소프트웨어의 설계 혹은 완성된 소프트웨어의 구현 설명을 목적으로 사용할 수 있다. 이 다이어그램은 속성(attribute)과 메서드(method)를 포함한다. -

## >클래스 다이어그램

파일 구조 -> 순차, 인덱스, 해싱 접근

```
int(*p)[3] = NULL; //열 기준 맞춤 배열포인터
```

arr[2][3] -> p = arr 시 p 와 arr 은 같아짐

하위 모듈부터 시작하여 상위 모듈로 테스트를 진행하는 방식 =**상향통합방식**

이미 존재하는 하위 모듈과 존재하지 않은 상위 모듈에 대한 인터페이스 역할을 한다 =**테스트 드라이버**



두 개 이상의 하드디스크를 병렬로 연결해 하나의 디스크처럼 이용하는 기술, 스트라이프 방식으로 구현하여 I/O 속도가 빠르다 -> RAID 0

로그 기반 회복 기법 -> redo / undo

키보드나 마우스와 같은 장치 없이 말이나 행동 그리고 감정과 같은 인간의 자연스러운 표현으로 컴퓨터나 장치를 제어할 수 있는 환경 -> NUI

정부에서 정한 인증기관 및 심사기관에서 기업이 주요 정보자산을 보호하기 위해 수립·관리·운영하는 정보보호 관리체계가 인증 기준에 적합한지를 심사하여 인증을 부여하는 제도 -> ISMS

인터넷을 통해 디바이스 간에 사설 네트워크 연결을 생성하며, 퍼블릭 네트워크를 통해 데이터를 안전하게 익명으로 전송하는 데 사용된다. 또한 사용자 IP 주소를 마스킹하고 데이터를 암호화하여 수신 권한이 없는 사람이 읽을 수 없도록 한다. -> VPN

하드웨어나 소프트웨어의 개발 단계에서 상용화하기 전에 실시하는 제품 검사 작업. 제품의 결함 여부, 제품으로서의 가치 등을 평가하기 위해 실시한다. 선발된 잠재 고객으로 하여금 일정 기간 무료로 사용하게 한 후에 나타난 여러 가지 오류를 수정, 보완한다. 공식적인 제품으로 발매하기 이전에 최종적으로 실시하는 검사 작업이다. -> 베타테스트

새로운 제품 개발 과정에서 이루어지는 첫 번째 테스트. 즉, 시제품이 운영되는 동안의 신제품 연구와 개발 과정 단계에서 초기 작동의 결과를 평가하는 수단이며 개발 회사 내부에서 이루어지는 테스트로서 단위 테스트, 구성 테스트, 시스템 테스트 등을 포함한다. -> 알파테스트

같은 도메인=IGP / 다른 도메인 사이 라우팅시 빠른 수행보다 보안과 제어 목적=EGP  
규모 큰거=OSPF / EGP 중 AS 구성=BGP