실전성 1위

만족도 1위

가성비 1위

2주만에 개념 완성 / 비전공자를 위한 초단기 실전서

2022 윤파고의 정보처리 기사 (개정판)

단기 합격률 압도적 1위

분석하라 윤파고

- IT 유튜브 채널 운영
- 현직 정보 보안 회사 기획팀
- 2021 정보처리기사 교재 출판
- 2020 컴퓨터 활용 1급 교재 출판

Contents

1. 필독사항 - 0p

2. 공부 방법 - 3p

Ⅰ. 가장 이상적인 2주 완성 공부 법 - 3p
Ⅱ. 조금은 급하게 10일 완성 공부 법 - 5p
Ⅲ. 남은 시간이 없을 때 일주일 완성 공부 법 - 7p

3. 이론 정리 - 10p

Ⅰ. 소프트웨어 설계 (대단원1/5) - 12p
Ⅱ. 소프트웨어 개발 (대단원2/5) - 26p
Ⅲ. 데이터베이스 구축 (대단원3/5) - 40p
Ⅳ. 프로그래밍 언어 활용 (대단원4/5) - 58p
Ⅴ. 정보시스템 구축관리 (대단원5/5) - 72p

4. 마무리하며 - 86p

5. 질의응답 - 87p

[1. 필독사항] - 윤파고의1주일 완성 정처기 2022.ver(개정판)

책을 들어가기 앞서 당신이 왜 이 책을 잘 선택한 것인지 그리고 이 책이 정확히 어떠한 효과를 가지고 있는지, 어떠한 방식으로 이 책을 사용해야 하는지, 왜 이 책처럼 공부를 해야 하는 건지 먼저 설명하도록 하겠다. 어찌 보면 뒤에 있을 이론과 기출 문제보다 앞의 이 필독사항과 공부 방법 부분이 여러분들에게 제일 중요한 부분이 될 수도 있다고 생각한다. 그러니 **반드시 '1. 필독사항'과 '2. 공부 방법'을 읽고 이해한 후 학습을 진행하길 바라며 혹여나 이해하지 못했다면 두 번, 세 번이고 읽어서 반드시 해당 내용을 이해하고 넘어가길 바란다.**

현재 시중에는 수많은 정보처리기사 참고서와 강의들이 있다. 책의 두께를 보면 정말 이게 전공서적인지 사전인지 알 수 없을 정도의 두께의 책을 3~4만원 사이의 돈을 받고 팔고 있다. 물론 그 책으로 합격하는 수많은 수험생들이 나오고 있지만 나는 이러한 참고서로 필기시험을 공부해서 합격하는 이들이 정말 많은 돈 낭비 시간 낭비를 한다고 생각한다. 참고서를 사서 첫 장부터 필기 공부를 시작하는 당신은, 아니 당신뿐만 아니라 컴퓨터 공학 전공자인 나조차 현재 참고서로 나와 있는 책을 다 보기에는 너무 오랜 시간이 걸린다. 생각해보라 3~4만원을 주고 산, 전공서적보다 더 무거운 책을 도서관까지 낑낑대며 들고 가 필기 공부에만 최소 한 달을 잡고 남는 여가 시간을 모두 올인 해야 겨우 붙을까 말까 한 어려운 시험이 되는 것이다. 그러나 정보처리기사는 그 중에서도, 특히 기사 필기시험은 절대 그 정도로 어려운 시험이 아니다. 나는 이에 대해서 확실하게 말할 수 있다. 처음부터 끝까지 정도의 방향으로 참고서를 공부한다면 그건 정말 멍청한 짓이다. 3~4만 원 정도의 돈이 아깝냐고? 아니다! 기사급의 자격증을 취득하기 위해서 충분히 쓸 수 있는 돈이라고 생각한다. 책의 무게? 그래 종종 전자책이 아니라 책장을 넘기며 공부하는 것에 행복함을 느끼는 사람도 있으니 그 또한 그럴 수 있다고 생각한다. **다만 내가 제일 화가 나는 것은 '전혀 필요 없는 쓸데없는 내용들까지 NCS(국가직무능력표준)에 있으니까 모조리 박아 넣고선 원래 자격증 공부는 정석대로 해야 하는 거야~'**라고 약을 팔며 수험생들의, 그리고 절박한 취준생들의 시간을 뺐어가는 일이다. 돈 3,4만원이야 충분히 낼 수 있다. 그러나 여러분들의 시간은??? 돈 한 두 푼보다 더 중요한 것이 바로 여러분들의 시간이다. **나는 장담할 수 있다. 내가 써준 공부 방법, 내가 시키는 루트로 여러분들이 기사 공부를 한다면 정말 머리가 안 좋고 멍청한 사람이 기존에 정도의 길로 시중의 유명하고 두꺼운 참고서로 공부한다고 가정할 시, 자격증을 따는데 100시간이 걸렸다면 내 공부 방법과 교재를 사용한다면 30~40시간, 정도의 1/3 정도의 시간을 쓰면 충분히 합격할 수 있다.** 물론 비전공자도 말이다. 생각해보라 당신의 시간을 시급 8천원으로만 잡아도 50시간이면 40만원이다(물론 당신의 시간이 더한 가치가 있다면 이보다 훨씬 큰 금액이 될 것이다). 이 책을 선택하고 이 책으로 공부하게 될 여러분들은 내가 장담하건데 정도의 길로 공부하는 사람들보다 **훨씬 합리적이고 간편하며 경제적**으로 기사 자격증을 취득할 수 있을 것이다.

내가 왜 이렇게 자신 있게 말할 수 있냐고? 수험서를 찍어내는 유명한 이X적, 시X공 같은 거대한 회사들이 말해주지 않는 비밀을 나는 내가 직접 시험공부를 하면서 깨달았기 때문이다. **기본적으로 한국의 자격증 시험은 기출 문제를 만들어 놓고 그 안에서 냈던 문제를 지속적으로 돌려**

쓰는 형태의 문제은행제 방식이다. 그렇기 때문에 사실 이런 전공서적 같은 책의 처음부터 끝까지 다 보는 것은 정말 엄청난 시간 낭비다. NCS의 모든 내용? 절대로 다 알 필요도 없고 실제 현장으로 가면 기사 자격은 최소한의 상식을 보여주는 정도지 내가 60점으로 기사를 땄는지 100점으로 기사를 땄는지는 전혀 중요하지 않다. 60점을 넘으면 모두가 평등한 기사 자격을 가진 사람들이다. 고로 단어에 대한 이론부터 구체적으로 이게 뭔지 저게 뭔지 정리해놓은 사전만한 크기의 책을 다 볼 필요는 전혀 없다는 말이다. (물론 책을 더 깊게 볼수록 현업에서 도움이 전혀 안 되는 것은 아니지만 우리는 기사 합격만을 최우선 목표로 한다)

누군가는 내가 하는 이러한 주장에 대해서 이렇게 말하고는 한다. "이론과 원리를 제대로 파악하고 이를 문제에서 적용해야만 제대로 된 공부가 되고 제대로 된 자격증 취득이 아닌가?" 혹은 "그렇게 하면 현업에서 전혀 못 써먹지 않나요?"라고 말이다. 하나는 알고 둘은 모르는 소리다. 자격증의 원래 취지가 뭔가? 자격증이 있으면 본인의 능력이 뛰어나다는 것을 증명 하는 건가??? 정말 멍청한 소리다. 다시 한 번 말하지만 현장에서 자격증은 본인이 이 자격, 이 분야의 업무에 대해서 어느 정도 내가 관심이 있고, 어느 정도의 기초 상식이 있다는 것을 보여주는 그 이상 그 이하도 아니다. **한 마디로 취업을 위해서 필요하거나 혹은 이미 취업한 사람이 본인의 스펙과 전문성을 조금 더 단단하게 만들기 위해서 필요한 것이란 말이다.** 사람들이 자격증을 취득을 하다보면 본인이 쏟아 넣고 갈아 넣은 시간이 아까워서 본인이 자격증을 따려던 이유를 많이 잊어버린다. 우리는 취업하기 위해서 자격증이 필요한 것이다. 우리는 승진, 혹은 최소한의 조건을 맞추기 위해서 자격증이 필요한 것이다. 그렇다면 우리는 **효율적으로 공부를 해야만 한다. 효율적으로 시간을 써야만 한다.** 당신의 시간은 무한정하지 않다!

다른 학습서에는 어떠한 형태든 NCS에 나와 있는 모든 내용을 책 한권에 때려 박아 넣고 '이거 싹 다 보면 합격이야!' 이러한 형태의 학습서를 만들어 놓았다. 왜냐하면 이들은 수험생의 입장을 고민할 필요가 전혀 없기 때문이다. 이런 학습이 비효율 적이라는 것을 이들이 과연 몰랐을까? 나는 절대 아니라고 생각한다. 이들은 이미 알고 있었다. 이렇게 공부하면 비효율 적이라는 것을, 다만 그들은 또 알고 있었다. 이렇게 다 때려 넣어야만 돈이 된다는 것을... 그렇기 때문에 필기는 필기대로 공부하고 실기는 실기대로 공부하며 돈은 두 배, 시간은 N 배 라는 기적의 비효율 학습이 우리의 선배들에게는 너무나 당연한 것이 되었고... 우리 또한 그렇게 공부를 해야만 했다. 다양한 국가 자격증 시험에 합격하고 뒤돌아보니 훨씬 효율적으로 쓸 수 있었던 시간들이 너무 아까웠다. 그리고 학습자의 입장을 배려하면서 써진 책이 정말 단 한 권도 없었다. 실기와 필기 둘 다 같은 개념을 베이스로 하고 있지만 필기 책에서는 필기 기출이랑만 엮어주고, 실기 책에서는 실기 기출이랑만 엮어준다. 같은 개념을 공부하지만 필기 공부를 하면서 실기까지 고려해주는 책은 없다고 봐도 무방하다.

한 권의 책 안에서 필기와 실기를 완벽하게 모두 담아내는 것은 분명 힘들 것이라 판단된다. 그러나 적어도 필기를 공부하면서 실기도 어느 정도의 비율을 충분히 같이 공부 할 수 있다. 다만 아무도 이에 대해서 고민하거나 알려주지 않았을 뿐이다.(시험장에서 필기시험 2수, 실기시험 3수 째라며 분통을 터트리던 학생이 아직도 기억난다. 이 공부 방법을 미리 알았더라면...) 이 책

은 여러분들이 필기 공부 시간을 현저히 줄이면서 안전하게 필기시험에 합격하게 해 줄 뿐 아니라 나아가 실기 시험 학습에 대해서 학습을 훨씬 쉽게 할 수 있는 베이스를 깔아 줄 것이다. 이에 대해서는 이미 주변의 동생들을 대상으로 임상실험을 완료하였다.(임상 실험이라고 거창하게 말했지만 그래봤자 나포함 5명이며 전원 합격하였다.) 나를 믿고 따라준 동생들에게 그리고 나를 믿고 책을 구매해준 이들에게 부끄럽지 않도록 책을 구성하였으니 믿고 따라와 주길 바란다. 또한 본인들이 합격했다고 거기서 끝내지 말고 꼭 좋은 후기와 함께 주변 지인들에게도 이 책을 추천해서 다른 이들의 시간이 낭비되는 것을 막아주기 바란다.

그리고 일단 이 책을 샀으면 **나를 믿어라.**

다른 사람들이 시키는 거? 학원갈 필요? 다른 참고서? 맹세코 할 필요 없다. 겨우 내 주변 사람들로 한 테스트지만 어떻게 합격률이 100%였겠는가? 어떠한 국가 자격증 시험이든 단기 합격에 있어서는 대한민국 1타 강사라고 감히 자부할 수 있다. 참고서에 들어가 있는 쓸데없는 개념 쓸데없는 단원, 도표 차트 다 버렸다. 정말 꼭 봐야하는 것들만 추려서 정리했다. 또한 필기 공부와 동시에 실기 공부에서도 중요한 부분을 체크해 놓았다. 이 책만 제대로 봐도 실기 공부에서 훨씬 수월함을 느낄 것이다. 내 이름을 걸고 맹세한다. **당신이 지불한 것은 겨우 2만 7천원이지만 이 책으로 줄이는 당신의 시간은 최소 50만원 그 이상일 것이다.**

[2. 공부 방법]

서론이 너무 길었다. 지금부터는 여러분들이 합격하기 위한 공부 방법에 대해서 구체적으로 작성하도록 하겠다. 본인이 남은 시간에 따라 아래의 3가지 공부 방법 중 선택해서 공부하면 될 것이며 제일 추천하는 공부 방법은 **Ⅰ. 가장 이상적인 2주일 완성 공부 법(필기 100% + 실기60% 동시 대비)** 이다. 책을 구성하며 가장 신경 썼던 부분은 최대한 불필요한 학습파트와 시간을 줄이는 것이었으며, 나아가 필기를 공부하면서 실기와도 연관시켜 실기 공부에서도 훨씬 효율적으로 공부를 할 수 있게 구성하였다.(실제로 베타테스터로 해당 책으로 공부한 친한 동생은 필기 합격 3일 공부하고 필기 합격, 추가로 3일 공부하고 실기 합격 하였다. 물론 이런 재능 충 얘들이 흔한 것은 절대 아니지만 필기는 여러분들이 비전공자라도 1~2주일 합격 충분히 가능하다

링크 : https://threeidiotscoding.tistory.com/56)

아래의 공부방법은 해당 책의 목적에 맞게 '필기' 단기 합격에 초점을 맞춘 공부 방법이다. 비전공자고 실기를 고려한 장기 계획을 세우고 싶다면 아래의 블로그 링크를 참조하도록 하자 > 링크 : https://blog.naver.com/dbswjdgkssla/222590511516

Ⅰ. 가장 이상적인 2주일 완성 공부 법(필기 100% + 실기60% 동시 대비)

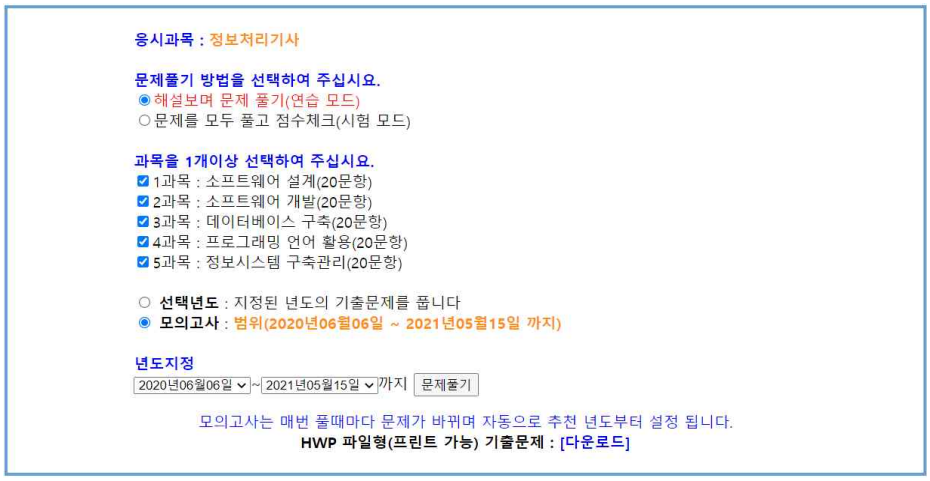
: 이론 공부 30시간(3시간 10일) + 기출 공부 16시간(4시간 4일) + 윤파고의 프로그래밍 특강 자료 4시간(비전공자는 필수) = 총 50시간

기사급의 자격증을 따려면 이론공부가 1도 필요 없다고 한다면 거짓말이다. 필기까지는 몰라도 실기까지 한 번에 따겠다는 생각이 있다면 반드시 이론공부는 병행 되어야만 한다. 객관식 형태의 필기시험은 사실상 이론 공부가 크게 병행 되지 않아도 충분히 합격할 수 있다. 다만 주관식 형태의 실기 시험까지 고려해야 불필요한 시간을 최소한으로 줄일 수 있다. 그렇게 때문에 아래와 같은 순서의 공부 순서를 따르면 된다. 최대한 아무것도 모르는 초심자를 기준으로 작성하였으니 이론에 어느 정도 자신이 있고 전공자라면 **Ⅱ. 조금은 급하게 10일 완성 공부 법(필기 100% + 실기 40% 동시 대비)** 으로 공부 하는 것을 추천한다.

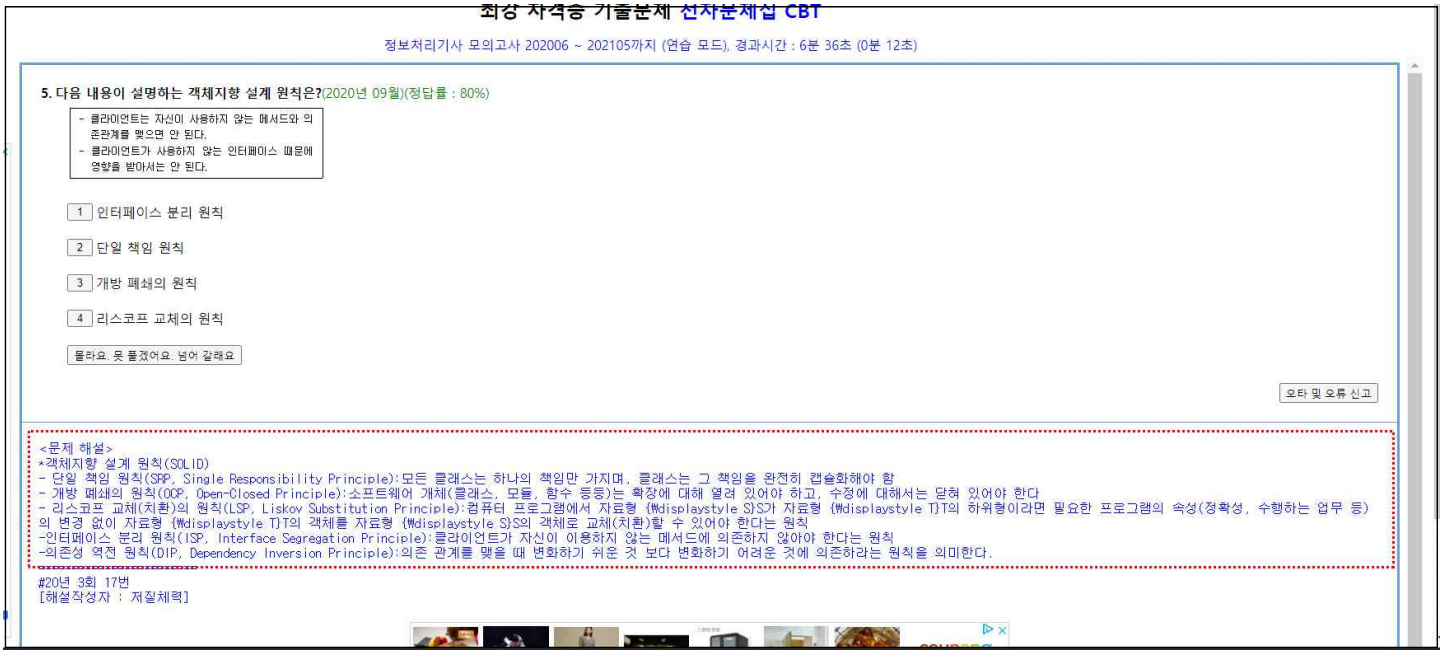
ㄱ. [3. 이론 정리]의 이론을 정독하면서 학습하며 하루에 한 파트씩 학습한다.

하루 3시간씩 10일을 목표로 잡고 공부하며 총 5개의 단원으로 구성된 파트를 이틀에 한 파트씩 정리하며 학습한다. 제일 이상적인 것은 2일 중 첫 날은 해당 범위를 모두 학습하고 다음 날은 해당 범위를 복습하는 것이 가장 이상적이나 초심자의 경우 시간이 조금 부족 할 수 있다. 그렇다 하더라도 모두 엑기스만 정리 된 내용들이니 2일이면 한 단원은 충분히 정리 할 수 있다. 학습을 하면서 중요한 것은 [필기 빈출],[실기 빈출]**이라는 표기가 된 것들은 더 중요하게 눈 여겨 봐야 한다**는 것이다. (시간이 여유롭다면 반드시 해당 표시가 있는 것들은 이해하고, 이해가 안 된다면 외우기라도 해서 넘어가는 것이 좋다 어차피 실기에서도 외우고 공부해야 할 내용이기 때문이다.)

ㄴ. 10일간의 이론 공부가 끝나면 이미 필기에 대한 준비는 40%는 끝난 것이다. 남은 60%는 CBT를 활용 한다. CBT사이트로 들어간 후 (https://www.comcbt.com/cbt/index2.php?hack_number=29) 정보처리 기사를 선택하고 아래의 사진과 같이 세팅한 후 문제를 푼다.



문제 풀기 방법은 '해설 보며 문제 풀기(연습 모드)'를 선택하고 과목은 '모두 선택', '모의고사' 형태로 선택한 후 본인이 풀 년도를 지정해서 풀면 된다. 추천하는 것은 3회분 정도로 1년 치의 년도를 지정하고 학습하는 것이 좋으며 가장 최신의 년도부터 1년씩 뒤로 가면서 학습하는 것을 추천한다. 학습 후 틀리든 맞았든 문제 해설을 보고 이해를 하고 넘어가야 하며 이는 시험이 아니기에 모르는 문제는 절대 찍지 말고 "몰라요 잘 못 풀겠어요."를 선택해서 틀린 문제로 넣어줘야 한다. 선택한 영역 안에서 마지막까지 문제를 다 풀고 나면 본인이 틀린 문제에 대해서 정리해서 한 번에 보고 확인 할 수 있게 해주는데 여기서 본인이 어떤 부분이 부족했고 몰랐는지 체크해야한다.(이 선별 작업이 제일 중요하다 문제 해설을 잘 참고 하도록 하자)



해당 사항을 3일에 나눠서 4시간씩 매일 기출을 풀고 정리하는 시간을 가져야하며 마지막 4일째는 본인이 공부했던 모든 회차를 범위에 넣고 문제를 풀어 본다.

해당 단계(ㄴ)에 들어섰다면 이제 개념에 너무 집착하지 말고 전반적으로 나오는 문제와 어떤 것들이 정답인지에 대한 암기에 집중해야만 한다.(이는 기출을 다 풀고 나면 자연스럽게 체화된다.) 우리는 문제를 맞추어 점수를 올려야 하는 것이지 IT공부를 잘하고 싶은 것이 아니다. 기출을 보고 해설을 보는 이유는 문제를 맞추기 위해서다. 수단과 목적을 혼동해서는 안 된다.

ㄷ. 4과목 프로그래밍 파트는 필기시험에서 예외적으로 조금은 전공자의 지식이 필요한 영역이다. 물론 기출과 해설을 꼼꼼히 읽었다면 필기시험에서는 충분히 풀 수 있는 정도의 난이도지만 비전공자들을 위해 필기를 풀 때 어떻게 접근해야 하는지 설명하는 영상과 자료를 2022년 버전부터 추가하였다. 비전공자가 아니라 전공자라도 프로그래밍 파트가 약하다면 꼭 시청하고 프로그래밍 파트 문제는 어떻게 접근해야 하는지, 버려야할 문제와 버리면 안되는 문제는 무엇인지, 어떤 개념을 중점으로 학습해야 하는지 정리했으니 참고 하길 바라며, 강의 영상보다는 PDF 자료가 정리가 잘 되어있으니 강의는 참고 자료로 쓰고, PDF자료는 꼭 꼼꼼하게 학습하길 바란다.

[링크]

1강(20년 6월, 8월) : https://www.youtube.com/watch?v=3DHHwx2Q0xQ

2강(20년 9월, 21년 3월) : https://www.youtube.com/watch?v=KFtT8fA2oul

3강은 1~2강의 설명과 너무 겹쳐서 영상 X

영상은 참고만 하고 제공된 PDF 파일 위주로 학습할 것

ㄹ. 준비는 완벽하다 해당 내용까지 학습을 정상적으로 완료했으면 필기시험은 이제 너무 쉬운 난이도로 느껴질 것이다. 조금 불안하다면 본인이 마지막으로 시험을 풀고 나서 틀렸던 문제모음을 출력해서 시험장에 가져가서 보면 된다. 시험을 치고 나서 합격했다고 생각하면 이제 바로 실기 시험 준비를 하면 된다. 해당 책은 필기를 위한 책이므로 실기 시험 준비는 “4. 마무리하며”에서 조금 더 언급하도록 하겠다.

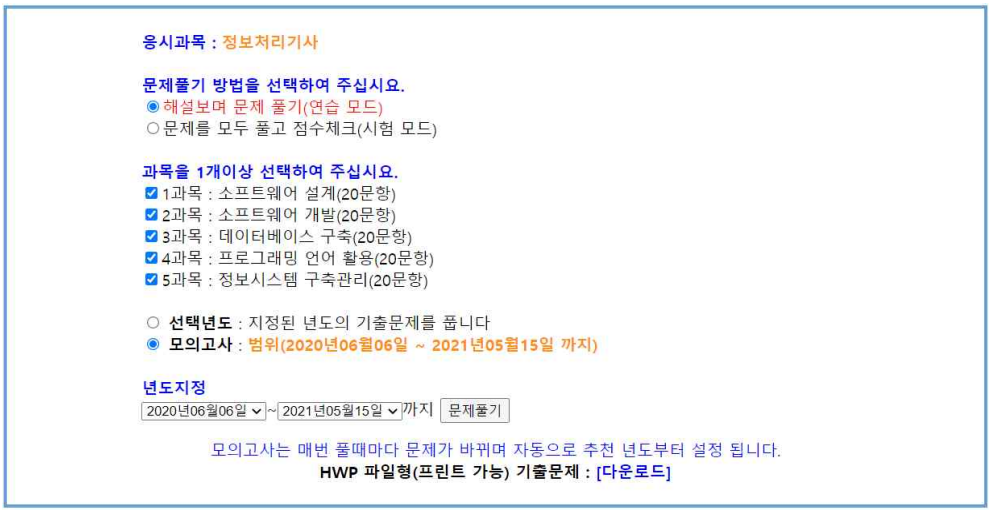
Ⅱ. 조금은 급하게 10일 완성 공부 법(필기 100% + 실기 40% 동시 대비)
: 이론 공부 15시간(3시간 5일) + 기출 공부 16시간(4시간 4일)
+ 윤파고의 프로그래밍 특강 4시간(비전공자는 필수) = 총 35시간

해당 방법은 어느 정도는 이론 공부를 해 본 전공자나 시간이 10일 정도 밖에 남지 않아 촉박하더라도 최대한 효율적인 공부를 하기 위한 사람에게 적합한 공부 방법이다. 하지만 실기에 대해서도 함께 준비하는 것이 가장 공부 효율이 좋으니 웬만하면 자만하지 말고 첫 번째 공부 방법을 선택하는 것을 추천한다.

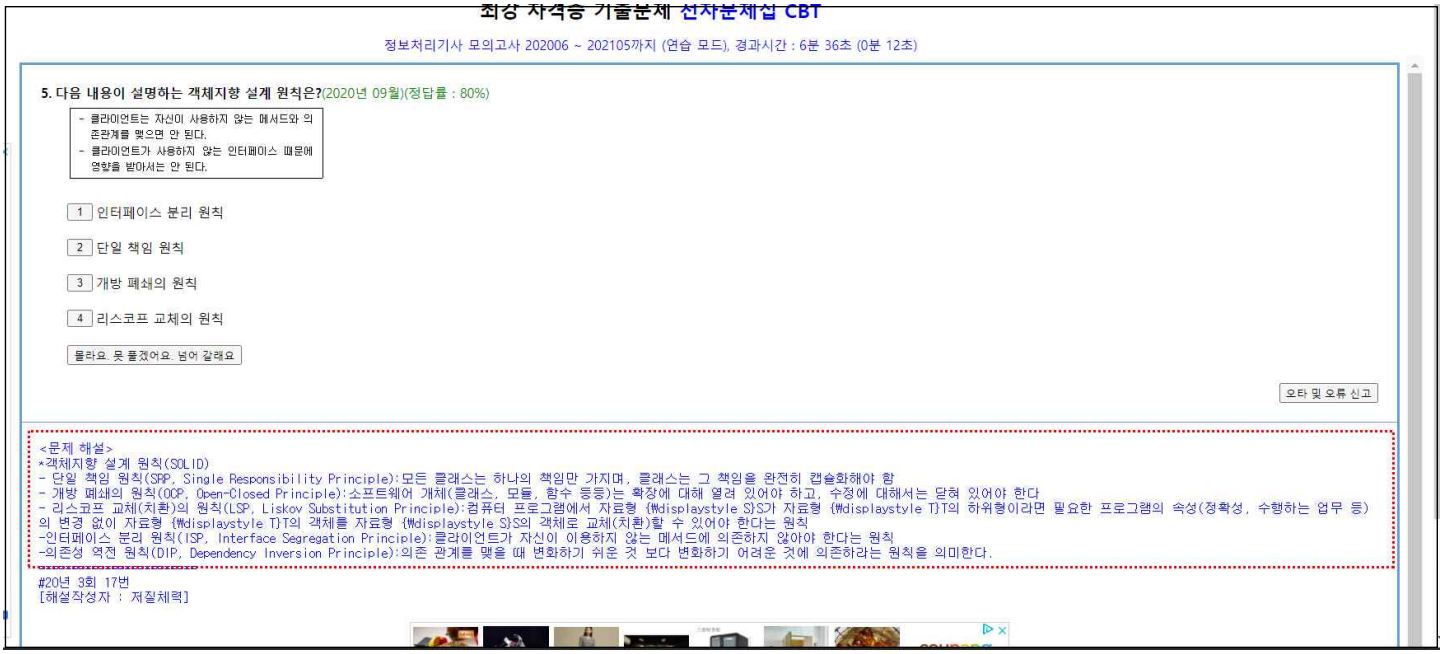
ㄱ. [3. 이론 정리]의 이론을 정독하며 학습하며 하루에 한 파트씩 학습한다.

하루 3시간씩 5일을 목표로 잡고 공부하며 총 5개의 단원으로 구성된 파트를 하루에 한 파트씩 정리하며 학습한다. 제일 이상적인 것은 4일간 1~5단원을 모두 학습하고 5일 차에 1~5강 전체 범위를 복습하는 것이 가장 이상적이다. 학습을 하면서 중요한 것은 [필기 빈출],[실기 빈출]이라는 표기가 된 것들은 더 중요하게 눈 여겨 봐야 한다는 것이다. (다른 것들보다 더 눈 여겨 보고 이해가 안가면 외워서라도 넘어가길 바란다. 어차피 실기에서도 외우고 공부해야 할 내용이기 때문이다.)

ㄴ. 5일간의 이론 공부가 끝나면 이미 필기에 대한 준비는 30%는 끝난 것이다. 남은 70%는 CBT를 활용 한다. CBT사이트로 들어간 후 (https://www.comcbt.com/cbt/index2.php?hack_number=29) 정보처리 기사를 선택하고 아래의 사진과 같이 세팅한 후 문제를 푼다.



문제 풀기 방법은 '해설 보며 문제 풀기(연습 모드)'를 선택하고 과목은 '모두 선택', '모의고사' 형태로 선택한 후 본인이 풀 년도를 지정해서 풀면 된다. 추천하는 것은 3회분 정도로 1년 치의 년도를 지정하고 학습하는 것이 좋으며 가장 최신의 년도부터 1년씩 뒤로 가면서 학습하는 것을 추천한다. 학습 후 틀리든 맞았든 문제 해설을 보고 이해를 하고 넘어가야 하며 이는 시험이 아니기에 모르는 문제는 절대 찍지 말고 "몰라요 잘 못 풀겠어요."를 선택해서 틀린 문제로 넣어줘야 한다. 선택한 영역 안에서 마지막까지 문제를 다 풀고 나면 본인이 틀린 문제에 대해서 정리해서 한 번에 보고 확인 할 수 있게 해주는데 여기서 본인이 어떤 부분이 부족했고 몰랐는지 체크해야한다.(이 선별 작업이 제일 중요하다 문제 해설을 잘 참고 하도록 하자)



해당 사항을 3일에 나눠서 4시간씩 매일 기출을 풀고 정리하는 시간을 가져야하며 마지막 4일째는 본인이 공부했던 모든 회차를 범위에 넣고 문제를 풀어 본다.

해당 단계(ㄴ)에 들어섰다면 이제 개념에 너무 집착하지 말고 전반적으로 나오는 문제와 어떤 것들이 정답인지에 대한 암기에 집중해야만 한다.(이는 기출을 다 풀고 나면 자연스럽게 체화된다.) 우리는 문제를 맞추어 점수를 올려야 하는 것이지 IT공부를 잘하고 싶은 것이 아니다. 기출을 보고 해설을 보는 이유는 문제를 맞추기 위해서다. 수단과 목적을 혼동해서는 안 된다.

ㄷ. 4과목 프로그래밍 파트는 필기시험에서 예외적으로 조금은 전공자의 지식이 필요한 영역이다. 물론 기출과 해설을 꼼꼼히 읽었다면 필기시험에서는 충분히 풀 수 있는 정도의 난이도지만 비전공자들을 위해 필기를 풀 때 어떻게 접근해야 하는지 설명하는 영상과 자료를 2022년 버전부터 추가하였다. 비전공자가 아니라 전공자라도 프로그래밍 파트가 약하다면 꼭 시청하고 프로그래밍 파트 문제는 어떻게 접근해야 하는지, 버려야할 문제와 버리면 안되는 문제는 무엇인지, 어떤 개념을 중점으로 학습해야 하는지 정리했으니 참고 하길 바라며, 강의 영상보다는 PDF 자료가 정리가 잘 되어있으니 강의는 참고 자료로 쓰고, PDF자료는 꼭 꼼꼼하게 학습하길 바란다.

[링크]

1강(20년 6월, 8월) : https://www.youtube.com/watch?v=3DHHwx2Q0xQ

2강(20년 9월, 21년 3월) : https://www.youtube.com/watch?v=KFtT8fA2oul

3강은 1~2강의 설명과 너무 겹쳐서 영상 X

영상은 참고만 하고 제공된 PDF 파일 위주로 학습할 것

ㄹ. 준비는 완벽하다 해당 내용까지 학습을 정상적으로 완료했으면 필기시험은 이제 너무 쉬운 난이도로 느껴질 것이다. 조금 불안하다면 본인이 마지막으로 시험을 풀고 나서 틀렸던 문제모음을 출력해서 시험장에 가져가서 보면 된다. 시험을 치고 나서 합격했다고 생각하면 이제 바로 실기 시험 준비를 하면 된다. 해당 책은 필기를 위한 책이므로 실기 시험 준비는 “4. 마무리하며”에서 조금 더 언급하도록 하겠다.

Ⅲ. 남은 시간이 없을 때 일주일 완성 공부 법(필기 100%)

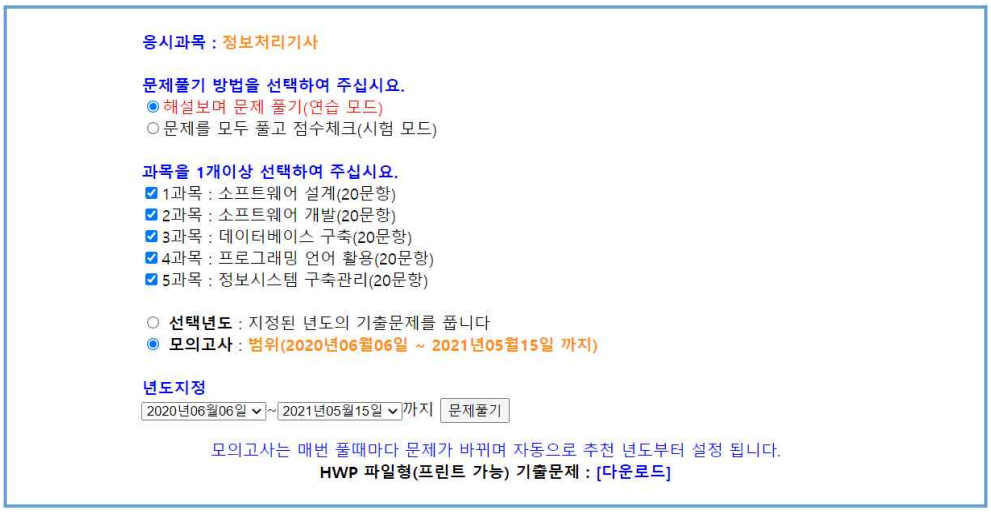
: 기출 공부 21시간(3시간 7일)

+ 윤파고의 프로그래밍 특강 4시간(비전공자는 필수) = 총 25시간

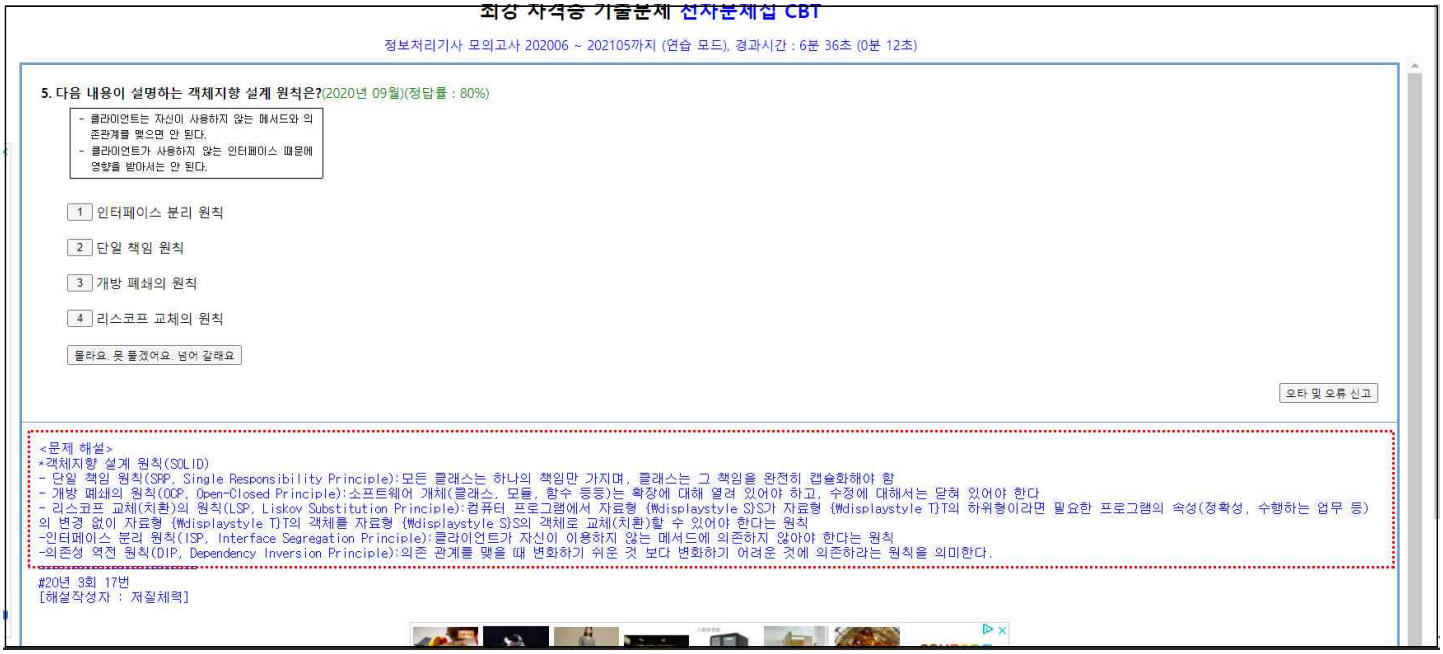
먼저 세 번째 공부 방법을 선택한 당신들... 나는 당신들이 매우 걱정스럽다. 시험공부를 효율적으로 해야 하는 것은 맞다. 나 역시 극한의 효율충이기 때문이다. 그런데 세 번째 공부 방법을 선택한 당신들은 사실상 이론 공부도 제대로 안한 비전공자거나 잘 알지도 못하면서 시험 날짜는 임박했고 시험은 합격하고 싶은, 대충만 아는 전공자인 사람들이다. 한마디로 도둑놈 심보라는 것이다. 비록 이번 시험은 내가 쓴 책으로 어찌어찌 요행으로 넘길지는 몰라도 명심해라, 아무리 효율적인 공부라도 최소한으로 들어가야 하는 시간이라는 게 있다. 7일이더라도 이 방법으로 공부하면 필기시험 합격률을 80% 위로 잡아도 될 것이다. 하지만 앞서 말했다시피 해당 방법으로 공부하면 실기는 별도로 공부해야 한다. 앞으로는 꼭 어떠한 공부를 하더라도 어느 정도의 여유 기간은 잡으면서 공부하기를 바란다.

ㄱ. 여유롭게 이론 정리할 시간이 없다. 최대한 빠르게 기출 문제 풀이로 들어가야 한다. 해당 세 번째 공부 방법은 이론은 거의 포기하고 기출문제를 우선적으로 암기하고 해설을 분석하여 어찌됐든 필기시험은 합격하게 해주는 전략이다. 구매한 이론 정리는 일단 시험을 합격하고 나서 실기 공부를 할 때든 보도록 하고 해당 공부 방법으로 공부한다.(이 이론서는 요약본이니 혹시라도 시간이 조금이라도 있다면 5시간 정도 밤새서 단번에 주욱 읽을 수 있으면 읽는 것이 좋다. 이해를 못하더라도 전반적인 흐름은 파악할 수 있을 것이다.)

ㄴ. CBT사이트로 들어간 후 (https://www.comcbt.com/cbt/index2.php?hack_number=29) 정보처리 기사를 선택하고 아래의 사진과 같이 세팅한 후 문제를 푼다.



문제 풀기 방법은 '해설 보며 문제 풀기(연습 모드)'를 선택하고 과목은 '모두 선택', '모의고사' 형태로 선택한 후 본인이 풀 년도를 지정해서 풀면 된다. 추천하는 것은 3회분 정도로 1년 치의 년도를 지정하고 학습하는 것이 좋으며 가장 최신의 년도부터 1년씩 뒤로 가면서 학습하는 것을 추천한다. 학습 후 틀리든 맞았든 문제 해설을 보고 이해를 하고 넘어가야 하며 이는 시험이 아니기에 모르는 문제는 절대 찍지 말고 "몰라요 잘 못 풀겠어요."를 선택해서 틀린 문제로 넣어줘야 한다. 선택한 영역 안에서 마지막까지 문제를 다 풀고 나면 본인이 틀린 문제에 대해서 정리해서 한 번에 보고 확인 할 수 있게 해주는데 여기서 본인이 어떤 부분이 부족했고 몰랐는지 체크해야한다. 어차피 여러분들은 이론을 안 봤기 때문에 이게 무슨 개소리야 싶은 게 대부분이겠지만 보다보면 결국 문제가 나오는 형태나 정답 형식이 매우 유사한 게 보일 것이다. 그리고 실제 시험에서는 정말 유사한 문제도 자주 나온다. 그러니 이론을 안 본 당신. 기출이라도 달달달 외워서 머리에 박아 넣어야 한다. 해설을 통째로 외워라 달달달 외우라는 게 아니라 해설을 보면서 '아 대충 이러한 개념들 때문에 얘가 답이구나' 라는 맥락을 외워야 한다. 정석인 개념 공부 후 기출 풀이가 아니라 당신들은 기출 풀이를 하면서 동시에 개념을 학습하는 것이다.



해당 사항을 7일에 나눠서 3시간씩 매일 기출을 풀고 정리하는 시간을 가져야하며 마지막 7일째는 본인이 공부했던 모든 회차를 범위에 넣고 문제를 풀어 본다.

아무리 개념 공부 안했어도 기출을 일주일간 잡고 있었다면 정말 웬만하면 필기는 다들 합격한다. 믿어라. 주변에 공부 지지리 안하는 동생 놈들 내가 이 방법으로 5일 컷 가능하다고 했는데 안 믿다가 내가 합격하는 거 보고 나서 한 명 그대로 따라 해서 3일 컷 했다. 내가 쓰는 책 내가 만드는 영상 모두 내 경험으로 실제로 겪고 연구해서 만드는 거다. 제발 의심 하지 말고 일단 공부를 시작해라.

ㄷ. 이 정도 했으면 불완전 하지만 충분히 합격 할만하다. 많이 불안하겠지만 어쩌겠는가 나태했던 본인을 탓해라. 시험장 가기 전 본인이 마지막으로 시험을 풀고 나서 틀렸던 문제모음을 출력해서 시험장에 가져가서 보면 된다. 시험을 치고 나서 합격했다고 생각하면 이제 바로 실기 시험 준비를 하면 된다. 특히나 세 번째 공부 방법을 선택한 당신은 개념을 부분부분 봤기 때문에 실기 공부에 훨씬 더 많은 시간을 쏟아야 할 것이다. 하지만 일단 급한 불은 껐으니 실기 준비는 꼭 천천히 오래 하길 바란다.

본인이 비전공자고 시간이 아직 많이 남아있는 상태(필기 시험까지 최소 2개월 이상 남은 상태)라면, 필기와 실기를 모두 고려한 공부 방법도 있다. 위의 공부 방법은 오직 실기만 고려한 공부 방법이니 필기와 실기 모두 고려해서 두 번 공부하고 싶지 않다면 아래의 링크를 참조하도록 한다.

링크 > https://blog.naver.com/dbswjdgkssla/222590511516

[3. 이론 정리]

<이론 정리 학습 시 참고사항>

이론 정리의 목차는 아래와 같다.

Ⅰ. 소프트웨어 설계 (대단원1/5)
Ⅱ. 소프트웨어 개발 (대단원2/5)
Ⅲ. 데이터베이스 구축 (대단원3/5)
Ⅳ. 프로그래밍 언어 활용 (대단원4/5)
Ⅴ. 정보시스템 구축관리 (대단원5/5)

실제 NCS를 기반으로 똑같은 항목으로 만들었으며, 총 5개의 대단원으로 이루어져 있다.

이론 정리의 규칙은 아래와 같다.

세부 단원 분류(로마 숫자)

: 5개의 대단원 안에서도 세부 단원이 나뉘며 해당 단원의 세부 단원은 "Ⅰ-Ⅰ요구사항 확인" 이런 식으로 나타낸다.

큰 주제 분류(일반 숫자)

: 주로 단어 정리와 번역 -> 단어의 정의(필요시) -> 해당 단어의 설명 순서로 나타나며 "1. 소프트웨어 생명주기" 이런 식으로 나타낸다.

세부 분류(ㄱ,ㄴ,ㄷ....)

: 큰 주제 안에서 종류나 내용이 나뉠 경우 분류는 ㄱ,ㄴ,ㄷ... 순으로 분류한다.

세부 분류의 분류(a,b,c....)

: 세부 분류 안에서 종류나 내용이 나뉠 경우 분류는 a,b,c... 순으로 분류한다.

[필기 빈출],[실기 빈출] 으로 표시 된 문구들은 이해를 하려고 노력해야 하고 중요한 것이니 최대한 꼼꼼하게 보는 것을 추천한다. 이해가 안 간다면 암기라도 하도록 하자!

또한 '큰 주제 분류(일반 숫자)' 자체에 [필기 빈출],[실기 빈출] 표시를 걸어 놓은 것은 해당 주제 자체가 전반적으로 중요한 내용이라는 것이니 해당 주제는 꼭 다른 파트들보다 더 신경 써서 학습해주길 바란다. 그리고 단원이나 큰 주제 분류에 빈출 표시를 해놨는데 안에 있는 세부 분류에도 빈출 표기를 중복으로 해놓은 것들은 그만큼 정말 많이 나오고 중요하게 다루고 있기 때문에 중복 표기 한 것이니 꼭 이해하고 숙지해 주길 바란다.

* 블로그나 참고자료 링크가 깨졌거나 안 들어가지는 경우 그냥 관련 주제로 구글링해서 보도록 하자. 문서에 포함된 링크들은 자료 찾는 시간을 최대한 줄이라고 내가 먼저 구글링 해보고 정리 잘 해놓은 글들을 찾아서 링크를 달아 놓은 것들인데, 블로그 주인 사정에 따라 글을 내리거나 옮기는 경우도 많다. 어차피 현업가면 여러분들 스스로 자료를 찾아서 보는 습관을 들여놔야 한다. 그러므로 자료를 찾아야 한다면 미리미리 본인이 검색해보는 연습을 하도록 하자. 구글링이 여러분의 평생 밥줄을 책임진다는 말이 절대 빈 말이 아니다! 현업에서는 구글링 잘 하는 것도 그 사람의 중요한 실력과 평가 요소 중 하나다.

*구글링 : 구글로 자신이 필요한 정보를 검색하고 찾아서 정리해 보는 것을 지칭하는 용어

이제 본격적으로 정보 처리 기사 공부를 시작하도록 하자!

Ⅰ. 소프트웨어 설계 (대단원1/5)

<Ⅰ-1 요구사항 확인>

1. 소프트웨어 생명 주기

소프트웨어 생명 주기(Life Cycle)

: 소프트웨어 개발 방법론의 바탕이 되며 소프트웨어를 개발하기 위해 정의하고 운용 유지보수 등의 과정을 각 단계별로 나눈 것

- 소프트웨어 개발 단계와 각 단계별 주요 활동 및 활동의 결과를 산출물로써 표현
- 소프트웨어 생명 주기를 표현하는 형태를 소프트웨어 생명 주기 모형, 소프트웨어 프로세스 모형, 소프트웨어 공학 패러다임이라고 함
- 특정 모형을 선택하여 사용하거나 개별적인 모형을 사용할 수 있음

폭포수 모형 [필기 빈출],[실기 빈출]

: 폭포수가 거슬러 올라갈 수 없듯이 이전 단계를 확실히 마무리하고 다음 단계로 진행하는 개발 방법론

- 소프트웨어 공학에서 가장 오래되고 폭넓게 사용된 생명 주기 모형
- 한 단계가 끝나야 다음 단계로 넘어갈 수 있는 선형 순차적 모형
- 매뉴얼을 작성해야 함
- 단계를 끝내고 다음 단계로 가기 위해서는 결과물이 명확히 나와야함

프로토타입 모형

: 요구사항을 정확히 파악하기 위해 실제 개발될 소프트웨어에 대한 시제품을 만들어 최종 결과물을 예측하는 모형

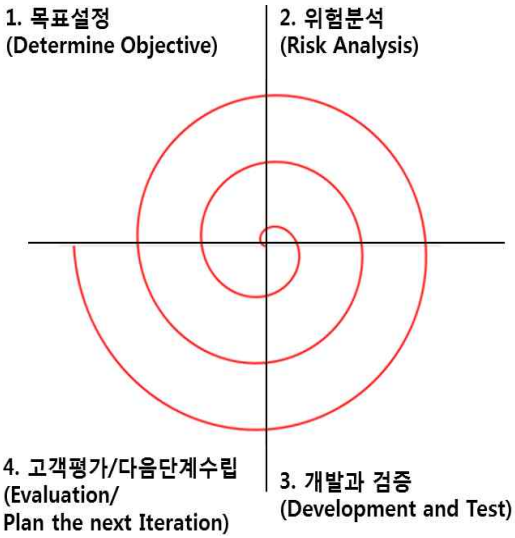
- 폭포수 모델의 단점을 보완하기 위해 만들어진 모형
- 사용자와 시스템 사이 인터페이스에 중점을 두어 개발

나선형 모형 [필기 빈출],[실기 빈출]

: 폭포수 모형과 프로토타입 모형의 장점에 위험 분석 기능을 추가한 모형

- 나선을 따라 돌 듯 여러 번의 개발 과정을 거쳐 점진적으로 완벽한 최종 소프트웨어를 개발
- 소프트웨어를 개발하면서 발생할 수 있는 위험을 관리하고 최소화하는 것이 목적
- 누락되거나 추가된 요구사항을 첨가할 수 있음
- 정밀하고 유지보수 과정이 필요 없음
- 대규모 프로젝트에서 많이 쓰임

타당성에 대해 검토 -> 계획 -> 요구 분석 -> 설계 -> 구현 -> 시험 과정을 거침



<나선형 모델의 프로세스>

2. 애자일 모형 [필기 빈출],[실기 빈출]

: 고객의 요구사항 변화에 유연하게 대응할 수 있도록 일정한 주기를 반복하면서 진행하는 모형

- 좋은 것을 빠르고 낭비 없게 만들기 위해 고객과의 소통에 초점을 맞춘 모든 방법론을 통칭
- 스프린트 또는 이터레이션이라고 불리는 짧은 개발 주기를 반복
- 반복되는 주기마다 결과물에 대해 평가와 요구 수용
- 요구사항에 우선순위를 부여하여 개발 진행
- 애자일 모형을 기반으로 하는 모형에는 스크럼, XP, 칸반, Lean, 크리스탈, ASD, FDD, DSDM 등이 있음

ㄱ. 스크럼 [필기 빈출],[실기 빈출]

: 럭비용어에서 파생된 언어로 여럿이 똘똘 뭉쳐 힘을 쓰는 것처럼 팀끼리 협업하면서 개발하는 것을 말함

- 팀이 중심이 되어 개발의 효율성을 높임
- 팀원 스스로가 팀을 구성하고 개발 작업에 대한 모든 것을 스스로 해결할 수 있어야 함

스크럼의 구성 요소

a. 제품 책임자

: 개발될 제품에 대한 이해도가 높고 요구사항을 책임지고 의사 결정을 할 사람

- 개발 의뢰자나 사용자가 담당
- 이해관계자들의 의견을 종합하여 제품에 대한 요구사항을 작성
- 백로그를 작성

(백로그란 프로젝트의 네비게이터이다. 주로 해야 할 일 목록, 소요 시간 등을 포함하여 프로젝트를 완료 하는 데에 필요한 모든 것들을 보여줌)

- 팀원들은 백로그에 스토리는 추가할 수 있지만 우선순위를 지정하는 것은 제품 책임자임
- 테스트를 수행하면서 주기적으로 요구사항의 우선순위 갱신

b. 스크럼 마스터

: 팀이 잘 수행할 수 있도록 객관적인 시각에서 조언을 해주는 가이드 역할

- 일일 스크럼 회의를 주관하여 진행 사항을 점검하고 개발과정에서 발생된 장애 요소를 공론화하여 처리함

c. 개발팀

: 개발자 외에도 디자이너, 테스터 등 제품 개발을 위해 참여하는 모든 사람을 지칭

- 보통 최대 인원은 7~8명이 적당함

d. 제품 백로그(Product Backlog)

: 개발에 필요한 요구사항을 우선순위에 따라 나열한 목록

- 새롭게 도출되는 요구사항으로 인해 지속적 업데이트
- 작성된 사용자 스토리를 기반으로 릴리즈 계획을 수립

e. 스프린트 계획 회의(Sprint Planning Meeting)

: 이번 스프린트에서 수행할 작업을 대상으로 단기 일정 수립

- 처리할 요구사항을 개발자들이 나눠서 작업할 수 있도록 태스크라는 작업 단위로 나눠 개발자 별로 수행할 작업 목록인 스프린트 백로그(Sprint Backlog) 작성

f. 스프린트(Sprint)

: 실제 개발 작업을 진행하는 과정

- 스프린트 백로그에 작성된 태스크를 대상으로 작업 시간이나 양을 추정한 후 개발 담당자에게 할당
- 태스크를 할당할 때는 개발자가 원하는 태스크를 직접 선별하여 담당할 수 있도록 하는 것이 좋음
- 할당된 태스크는 할 일, 진행 중, 완료의 상태를 가짐

g. 일일 스크럼 회의(Daily Scrum Meeting)

: 모든 팀원이 매일 약속된 시간에 짧은 시간동안 진행 상황을 점검하는 회의

- 스크럼 마스터는 발견된 장애 요소를 해결할 수 있도록 도와줌
- 남은 작업 시간은 소멸 차트에 표시

h. 스프린트 검토 회의(Sprint Review)

: 부분 또는 완성 제품이 요구사항에 잘 부합되는지 사용자가 포함된 참석자 앞에서 테스트를 수행

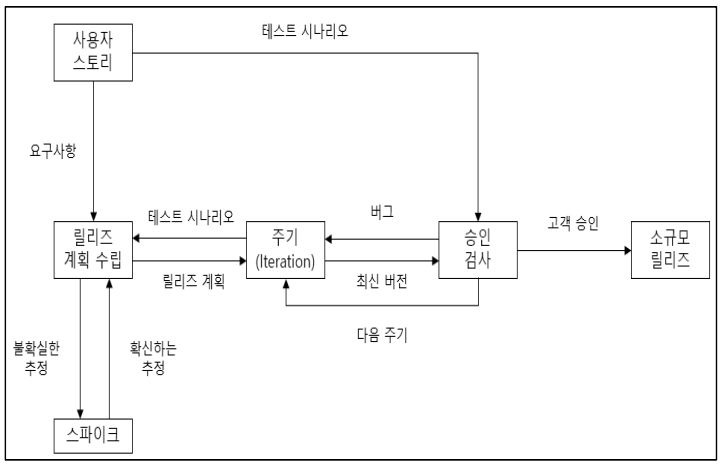
i. 스프린트 회고(Sprint Retrospective)

- 스프린트가 끝나고 정해놓은 규칙을 잘 준수했는지, 개선할 점은 없는지 등을 점검하고 수행

ㄴ. XP(eXtreme Programming)의 정의 [필기 빈출],[실기 빈출]

: 수시로 발생하는 고객의 요구사항에 유연하게 대응하기 위해 고객의 참여와 개발 과정의 반복을 극대화하여 개발 생산성을 향상시키는 기법

- 짧고 반복적인 개발주기, 단순한 설계, 고객의 적극적인 참여를 통해 빠르게 개발하는 것이 목적
- 릴리즈의 기간을 짧게 반복하면서 요구사항 반영에 대한 가시성을 높임
- XP의 5가지 핵심 가치 : 의사소통, 단순성, 용기, 존중, 피드백



a. 사용자 스토리

: 고객의 요구사항을 간단한 시나리오로 표현한 것

b. 릴리즈 계획 수립(release)

: 몇 개의 스토리가 적용되어 부분적으로 기능이 완료된 제품을 제공하는 것에 대한 계획 수립

(release의 사전적 의미는 “풀어 주다”, “석방하다”, “놓아 주다” 로서 여기서의 의미는 “배포” 정도로 생각하면 됨)

c. 스파이크

: 요구사항의 신뢰성을 높이고 기술 문제에 대한 위험을 감소시키기 위해 별도로 만드는 프로그램

d. 이터레이션(Iteration)

: 하나의 릴리즈를 더 세분화 한 단위

e. 승인 검사

: 하나의 이터레이션 안에서 계획된 릴리즈 단위의 부분 완료 제품이 구현되면 수행하는 테스트

- 사용자 스토리 작성 시 함께 기재한 테스트 사항에 대해 고객이 직접 수행

f. 소규모 릴리즈

: 고객의 반응을 기능별로 확인하고 고객의 요구사항에 유연하게 대응하는 것

- 진행된 이터레이션이 모두 완료되면 고객에 의한 최종 테스트 수행 후 최종 결과물을 고객에게 전달함

XP의 주요 실천 방법

a. Pair Programming

: 다른 사람과 함께 프로그래밍 수행

b. Test-Driven Development

: 실제 코드 작성 전 테스트 케이스를 먼저 작성하여 무엇을 해야할지 파악 후 개발

c. Whole Team

: 개발에 참여하는 모든 구성원은 각기 역할이 있어 책임을 다해야 함

d. Continuous integration

: 모듈 단위로 나눠 개발한 코드는 하나의 작업이 마무리되면 지속적으로 통합

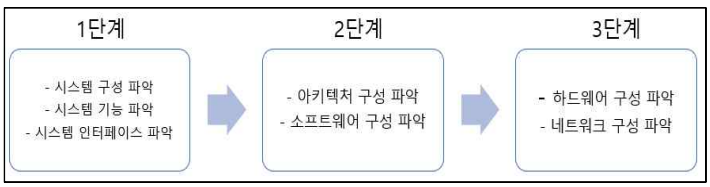
e. Design Improvement / Refactoring

: 프로그램 기능의 변경 없이 시스템을 재구성

f. Small Release

: 릴리즈 기간을 짧게 하여 고객의 요구 변화에 신속하게 대응

현행 시스템 파악 절차



ㄱ. 1단계

- 시스템 구성 파악 : 조직의 업무를 담당하는 기간 업무와 이를 지원하는 업무로 구분하여 나타낸 구성을 파악
- 시스템 기능 파악 : 현재 제공하는 기능들을 주요, 하부, 세부 기능으로 구분하여 계층형으로 표시
- 시스템 인터페이스 파악 : 주고받는 데이터의 종류, 형식, 프로토콜, 연계 유형, 주기 등을 명시

ㄴ. 2단계

- 아키텍쳐 구성 파악 : 어떠한 기술 요소들이 사용되는지 최상위 수준에서 계층별로 표현한 구성 파악
- 소프트웨어 구성 파악 : 업무 처리를 위해 설치되어 있는 소프트웨어의 제품명, 용도, 라이선스 적용 방식 등을 명시

ㄷ. 3단계

- 하드웨어 구성 파악 : 단위 업무 시스템들이 운용되는 서버의 주요 사양과 수량 및 이중화 적용 여부 명시
- 네트워크 구성 파악 : 서버의 위치, 서버 간의 네트워크 연결 방식을 네트워크 구성도로 작성

3. 개발 기술 환경 파악

개발 기술 환경

: 개발하고자 하는 소프트웨어와 관련된 O/S, DBMS, Middle Ware 등을 선정할 때 고려해야 할 사항을 기술하고 오픈 소스 사용 시 주의해야 할 내용을 제시

운영체제(Operating System) [필기 빈출],[실기 빈출]

: 컴퓨터 시스템 자원을 효율적으로 관리하여 사용자가 컴퓨터를 편리하게 사용할 수 있도록 환경을 제공하는 소프트웨어

- 요구사항 식별 시 고려사항 : 가용성, 성능, 기술 지원, 주변 기기, 구축비용

(Window, Unix, Linux 등이 있음)

데이터베이스 관리 시스템(DBMS) [필기 빈출],[실기 빈출]

: 사용자와 데이터베이스 사이에서 사용자의 요구에 따라 정보를 생성해주고 관리해주는 소프트웨어

- 요구사항 식별 시 고려사항 : 가용성, 성능, 기술 지원, 상호 호환성, 구축비용

웹 애플리케이션 서버(WAS) [필기 빈출],[실기 빈출]

: 사용자의 요구에 따라 변하는 동적인 컨텐츠를 처리하기 위해 사용되는 미들웨어

- 요구사항 식별 시 고려사항 : 가용성, 성능, 기술 지원, 구축 비용

(미들웨어 : 컴퓨터 제작 회사가 사용자의 요구대로 만들어 제공하는 프로그램으로, 운영 체제와 응용 소프트웨어의 중간에서 조정과 중개의 역할을 수행하는 소프트웨어)

오픈 소스(Open Source)

: 누구나 제한 없이 사용할 수 있도록 소스 코드를 공개한 것

- 라이선스를 만족하는 소프트웨어
- 요구사항 식별 시 고려사항 : 라이선스의 종류, 사용자의 수, 기술의 지속 가능성

4. 요구사항 정의

요구사항

: 소프트웨어가 어떤 문제를 해결하기 위해 제공하는 서비스에 대한 설명과 정상적으로 운영되는데 필요한 제약조건 등

- 요구사항이 제대로 정의되어야 이를 토대로 이후 과정의 목표와 계획의 수립이 가능

요구사항의 유형 [필기 빈출]

ㄱ. 기술하는 내용에 따른 분류

- 기능 요구사항 : 시스템이 무엇을 하고 어떤 기능을 하는지

에 대한 사항

- 비 기능 요구사항 : 품질이나 제약사항에 대한 사항

ㄴ. 기술 관점과 대상의 범위에 따른 분류

- 사용자 요구사항 : 사용자의 관점에서 본 시스템이 제공해야 할 사항
- 시스템 요구사항 : 개발자의 관점에서 본 시스템 전체가 사용자와 다른 시스템에 제공해야 할 사항

요구사항 개발 프로세스

: 도출 -> 분석 -> 명세 -> 확인

ㄱ. 요구사항 도출 [필기 빈출],[실기 빈출]

: 요구사항이 어디에 있고 어떻게 수집할지 식별하고 이해하는 과정으로 인터뷰, 설문, 브레인스토밍, 워크샵, 프로토타이핑, 유스케이스 등이 있음

- 브레인스토밍 : 3인 이상이 자유롭게 아이디어를 산출해 내는 방법
- 워크샵 : 소집단 정도의 인원으로 특정 문제나 과제에 대한 새로운 지식, 기술, 아이디어, 방법들을 서로 교환하고 검토하는 연구회 및 세미나
- 프로토타이핑 : 초기 도출된 요구사항을 토대로 프로토타입을 만든 후 개발이 진행되는 동안 도출되는 요구사항을 반영하면서 지속적으로 프로토타입을 재작성하는 과정
- 유스케이스 : 사용자 측면에서의 요구사항으로 사용자가 원하는 목표를 달성하기 위해 사용자 요구사항을 기능 단위로 정리한 시나리오

ㄴ. 요구사항 분석

: 개발 대상에 대한 사용자의 요구사항 중 명확하지 않거나 모호하여 이해되지 않는 부분을 발견하고 걸러내는 과정

ㄷ. 요구사항 명세

: 요구사항을 분석한 후 승인될 수 있도록 문서화하는 과정

- 소프트웨어 요구사항 명세서 : 소프트웨어가 반드시 제공해야 하는 기능, 특징, 제약조건을 명시

ㄹ. 요구사항 확인

: 개발 자원을 요구사항에 할당하기 전에 요구사항 명세서가 정확하게 작성되었는지 검토하는 과정

5. 요구사항의 분석 기법

요구사항 분류

: 요구사항을 명확히 확인할 수 있도록 정해진 기준으로 분류

- 기능/비기능 요구사항 분류
- 제품/과정으로 분류
- 우선순위로 분류

개념 모델링

: 요구사항의 현실 세계의 상황을 단순화하여 개념적으로 표현하는 과정

- 실세계 문제에 대한 모델링은 요구사항 분석의 핵심
- 개체와 개체 간의 관계 및 종속성을 반영
- 개념 모델은 다양하게 표현
- 주로 UML(Unified Modeling Language) 을 사용

(UML이란 : 프로그램 설계를 표현하기 위해 사용하는 주로 그림으로 된 표기법을 의미함)

요구사항 할당

- 요구사항을 만족시키기 위한 구성 요소를 식별

요구사항 협상

- 요구사항이 서로 충돌될 경우 해결하는 과정
- 적절한 기준점을 찾아 합의하는 게 좋음
- 서로 충돌하는 경우 우선순위를 부여하면 해결에 도움이 됨

정형 분석

- 구문과 의미를 갖는 정형화된 언어를 이용하여 요구사항을 수학적 기호로 표현 한 후 이를 분석하는 과정

6. 요구사항의 확인 기법

요구사항 검토

: 문서화된 요구사항을 훑어보면서 확인하는 과정

- 시스템 정의서, 시스템 사양서, 소프트웨어 요구사항 명세서 등을 완성한 시점에 이루어짐

프로토타이핑

: 초기 도출된 요구사항을 토대로 프로토타입을 만든 후 개발이 진행되는 동안 도출되는 요구사항을 반영하면서 지속적으로 프로토타입을 재작성하는 과정

ㄱ. 장점

- 빠르고 반복되는 제작을 할 수 있어 발전된 결과물을 얻을 수 있음
- 최종 시스템을 완성하기 전에 추가 또는 변경된 요구사항에 대한 피드백 가능

ㄴ. 단점

- 사용자의 관심이 핵심에서 벗어나 프로토타입 제작에만 집중될 수 있음
- 지속적이고 반복적인 개선에 대한 비용이 부담될 수 있음

모델 검증

: 요구사항 분석 단계에서 개발된 모델이 요구사항을 충족하는지 검증하는 과정

인수 테스트

- 사용자가 실제로 사용될 환경에서 요구사항이 모두 충족되는지 사용자 입장에서 확인하는 과정

7. UML(Unified Modeling Language)

UML이란? [필기 빈출],[실기 빈출]

: 시스템 개발 과정에서 시스템 개발자와 고객 또는 개발자 상호 간의 의사소통이 원활하게 이루어지도록 표준화한 객체 지향 모델링 언어

- 객체지향 방법론의 장점을 통합하였으며 OMG(Object Management Group)에서 표준으로 지정
- 구성 요소 : 사물, 관계, 다이어그램

ㄱ. 사물

: 모델을 구성하는 가장 중요한 기본 요소

- 다이어그램 안에서 관계가 형성될 수 있는 대상

a. 구조 사물

- 시스템의 개념적, 물리적 요소 표현
- 클래스, 유스케이스, 컴포넌트, 노드 등

b. 행동 사물

- 시간과 공간에 따른 요소들의 행위 표현
- 상호작용, 상태 머신 등

c. 그룹 사물

- 요소들의 그룹으로 묶어서 표현
- 패키지(그룹화)

d. 주해 사물

- 부가적인 설명이나 제약조건 등 표현
- 부가설명

ㄴ. 관계 [필기 빈출],[실기 빈출]

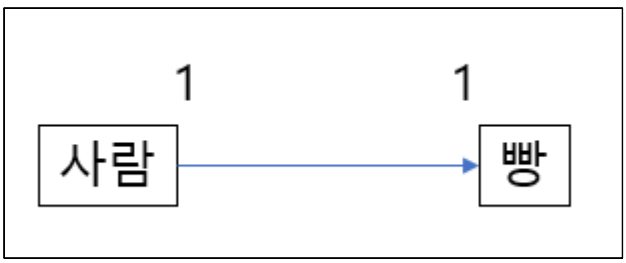
: 사물과 사물 사이의 연관성 표현

a. 연관 관계

- 2개 이상의 사물이 서로 관련되어 있음을 표현
- 실선과 화살표로 연결하여 표현하지만 양방향 관계의 경우 화살표 없이 실선으로 연결하여 표현

다중도	의미
1	1개의 객체가 연관됨
n	n개의 객체가 연관됨
0..1	연관된 객체가 없거나 1개만 존재
0..* / *	연관된 객체가 없거나 다수일 수 있음
1..*	연관된 객체가 적어도 1개 이상
n..*	연관된 객체가 적어도 n개 이상
n..m	연관된 객체가 n~m개

ex)



의미

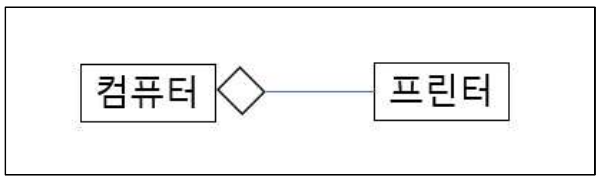
- 사람은 빵을 소유하는 관계. 사람은 자기가 소유하고 있는 빵이 뭔지 알고, 빵은 어느 사람이 자신을 소유하고 있는지 모른다(화살표 방향이 한방향)
- '사람'쪽에 표기된 다중도가 '1'이므로, 빵은 한 사람에 의해서만 소유될 수 있다.
- '빵'쪽에 표기된 다중도가 '1'이므로 사람은 하나의 빵만을 소유할 수 있다

b. 집합 관계 [필기 빈출],[실기 빈출]

: 하나의 사물이 다른 사물에 포함되어 있는 관계

- 부분(포함되는 쪽)에서 전체(포함하는 쪽)로 속이 빈 마름모를 연결하여 표현

ex)



의미

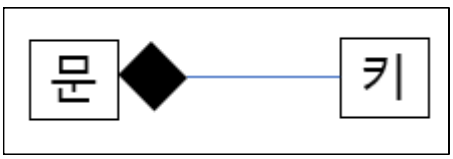
프린터는 컴퓨터에 연결해서 사용할 수 있으며, 다른 컴퓨터에 연결해서 사용할 수도 있다

c. 포함 관계 [필기 빈출],[실기 빈출]

:집합 관계의 특수한 형태로 포함하는 사물의 변화가 포함되는 사물에게 영향을 미치는 관계

- 부분(포함되는 쪽)에서 전체(포함하는 쪽)로 속이 채워진 마름모를 연결하여 표현

ex)



의미

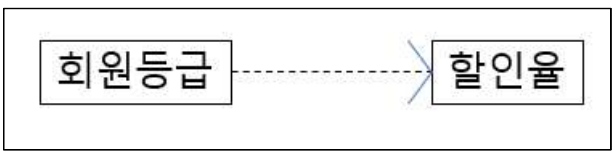
문을 열 수 있는 키는 하나이며, 해당 키로 다른 문을 열 수 없다. 문이 없어지면 키도 더 이상 필요하지 않다

d. 의존 관계

: 사물 사이에 연관은 있으나 필요에 의해서 서로에게 영향을 주는 짧은 시간 동안만 연관을 유지하는 관계

- 영향을 주는 사물이 영향을 받는 사물 쪽으로 점선 화살표 연결

ex)



의미

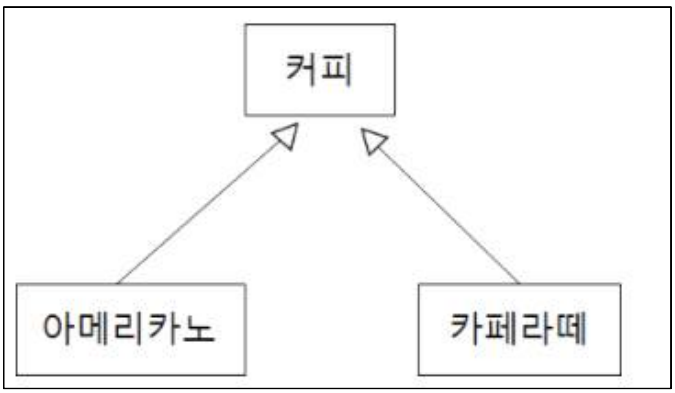
회원등급이 높으면 할인율을 적용, 회원등급이 낮으면 할인율을 미적용 한다

e. 일반화 관계

: 하나의 사물이 다른 사물에 비해 일반적인지 구체적인지 표현

- 구체적인 사물에서 일반적인 사물 쪽으로 속이 빈 화살표를 연결

ex)



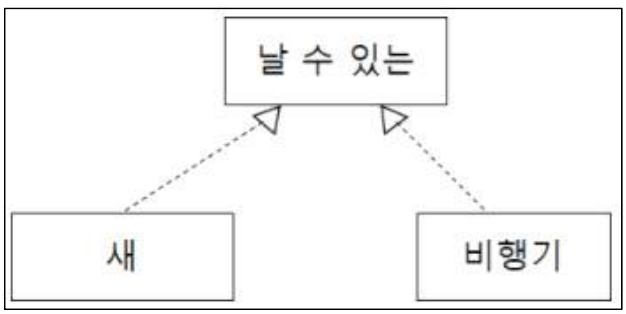
의미

아메리카노와 카페라떼는 커피이다(커피의 종류로는 아메리카노와 카페라떼가 있다)

f. 실체화 관계 [필기 빈출],[실기 빈출]

: 사물이 할 수 있거나 해야 하는 기능으로 서로를 그룹화 할 수 있는 관계

- 사물에서 기능 쪽으로 속이 빈 점선 화살표 연결



의미

비행기와 새는 날 수 있다. 그러므로 비행기와 새는 날 수 있다.(난다는 행위 자체로 그룹화 할 수 있다, 일반화 관계와는 다르게 사물이 할 수 있는 기능에 초점)

8. 다이어그램

다이어그램 [필기 빈출],[실기 빈출]

: 사물과의 관계를 도형으로 표현

- 정적 모델링에서는 주로 구조적 다이어그램을 사용하고 동적 모델링에서는 주로 행위 다이어그램 사용

ㄱ. 구조적 다이어그램

a. 클래스 다이어그램 : 클래스, 클래스가 가지는 속성, 클래스 사이 관계 표현

b. 객체 다이어그램 : 인스턴스를 특정 시점의 객체와 객체 사이의 관계로 표현

c. 컴포넌트 다이어그램 : 구현 단계에서 사용되며 컴포넌트 간의 관계나 인터페이스를 표현

d. 배치 다이어그램 : 구현단계에서 사용되며 결과물, 프로세스, 컴포넌트 등 물리적 요소들의 위치 표현

e. 복합체 구조 다이어그램 : 복잡한 구조를 가지는 클래스 혹은 컴포넌트의 내부 구조 표현

f. 패키지 다이어그램 : 유스케이스, 클래스 등의 모델 요소들을 그룹화한 패키지들의 관계 표현

(초심자들을 위한 Tip

- 클래스 : 객체지향에서 클래스란 객체를 정의하는 틀 또는 설계도와 같은 의미로 사용
- 객체 : 컴퓨터 과학에서 객체 또는 오브젝트(Object)는 클래스에서 정의한 것을 토대로 메모리(실제 저장 공간)에 할당된 것으로 프로그램에서 사용되는 데이터 또는 식별자에 의해 참조되는 공간을 의미하며 물리적으로 존재하거나 추상적으로 생각할 수 있는 것 중 자신의 속성을 가지고 있고 다른 것과 식별 가능한 것들을 말합니다.
- 컴포넌트 : 소프트웨어 시스템에서 독립적인 업무 또는 독립적인 기능을 수행하는 '모듈'로서 시스템을 유지보수 하는데 있어 교체 가능한 부품을 말함

컴포넌트 관련해서 읽어보면 좋은 글, 꼭 읽을 필요는 없다 어차피 뒤에 가면 컴포넌트 또 나옴!)

https://velog.io/@tjnoh/%EC%BB%B4%ED%8F%AC%EB%84%8C%ED%8A%B8)

ㄴ. 행위 다이어그램 [실기 빈출]

- 유스케이스 다이어그램 : 사용자의 요구를 분석하여 기능 모델링 작업에 사용됨
- 시퀀스 다이어그램 : 상호 작용하는 시스템이나 객체들이 주고받는 메시지 표현
- 커뮤니케이션 다이어그램 : 객체들이 주고받는 메시지를 표현할 뿐 아니라 객체들 간의 연관까지 표현
- 상태 다이어그램 : 하나의 객체가 자신이 속한 클래스의 상태 변화 혹은 다른 객체와의 상호 작용에 따라 어떻게 변화하는지 표현
- 활동 다이어그램 : 객체의 처리 로직이나 조건에 따른 처리의 흐름을 순서에 따라 표현
- 상호작용 개요 다이어그램 : 상호작용 다이어그램 간의 제어 흐름 표현
- 타이밍 다이어그램 : 객체 상태 변화와 시간 제약을 명시적으로 표현

<Ⅰ-Ⅱ 화면 설계>

9. 사용자 인터페이스(User Interface)[실기 빈출]

사용자 인터페이스

: 사용자와 시스템 간의 상호작용을 원활하게 도와주는 장치나 소프트웨어(흔히들 현업에 가면 현업자분들이 UI, UI거리는데 사용자 사용 환경, 디자인 등을 말한다고 생각하면 됨, 아래의 그림을 보면 더욱 직관적으로 이해 가능)

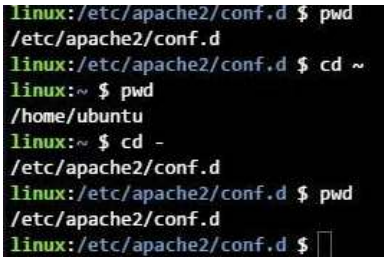
사용자 인터페이스의 3가지 분야

- 정보 제공과 전달을 위한 물리적 제어에 관한 분야
- 콘텐츠의 상세적인 표현과 전체적인 구성에 관한 분야
- 모든 사용자가 편리하고 간편하게 사용하도록 하는 기능에 관한 분야

사용자 인터페이스의 구분 [실기 빈출](3가지 다)

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

ex)

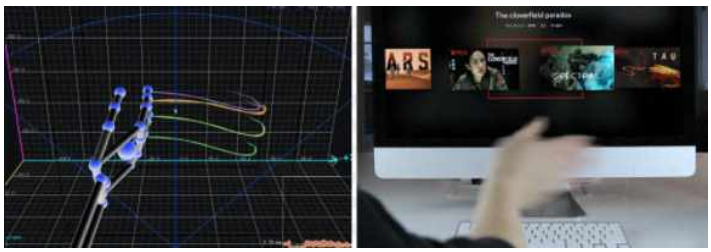


(리눅스 같은 검은화면에 코딩)

ㄴ. GUI(Graphic User Interface) : 아이콘이나 메뉴를 마우스로 선택하여 작업을 수행하는 인터페이스

ex)

ㄷ. NUI(Natural User Interface) : 말이나 행동으로 조작하는 인터페이스



(손짓으로 넘기는 모션 인식이나 언어로 조작하는 삼성의 빅스비 등을 생각하면 됩니다)

사용자 인터페이스의 기본 원칙 [필기 빈출],[실기 빈출]

ㄱ. 직관성 : 사용자가 큰 노력 없이 쉽게 이해하고 사용할 수 있게 제작해야 함

ㄴ. 유효성 : 사용자의 목표가 쉽게 달성될 수 있도록 제작 되어야 함

ㄷ. 학습성 : 초보자가 쉽게 배우고 사용할 수 있게 제작되어야 함

ㄹ. 유연성 : 사용자의 인터랙션을 최대한 포용하고 실수를 방지할 수 있게 제작 되어야 함

사용자 인터페이스의 설계 지침

- 사용자 중심, 일관성, 단순성, 결과 예측 가능, 가시성, 표준화, 접근성, 명확성, 오류 발생 해결

10. UI표준 및 지침

UI 표준 및 지침의 개요

- UI 표준과 지침을 토대로 기술의 중립성(표준), 보편적 표현 보장성(접근성), 기능의 호환성이 고려되었는지 확인

한국형 웹 콘텐츠 접근성 지침(KWCAG)

(KWCAG-Korean Web Content Accessibility Guidelines)

: 장애인과 비장애인이 동등하게 접근할 수 있는 웹 콘텐츠 제작의 방법 제시

- 웹 콘텐츠 접근성 지침 준수를 위한 고려사항임

전자정보 웹 표준 준수 지침

: 정부기관의 홈페이지 구축시 반영해야 할 최소한의 규약

- 전자정부 웹 표준 준수 지침 사항
- 내용의 문법 준수
- 내용과 표현의 분리
- 동작의 기술 중립성 보장
- 플러그인의 호환성
- 콘텐츠의 보편적 표현
- 운영체제에 독립적인 콘텐츠 제공
- 부가 기능의 호환성 확보
- 다양한 프로그램 제공

(어느 기업이 정부 기관 홈페이지 제작을 낙찰 받아 정부기관 홈페이지 구축 시 위의 사항을 준수해야함 현업에서는 중요하지만 시험에서는 크게 중요하게 다뤄지지 않음)

11. UI설계 도구

UI 설계 도구

: 사용자의 요구사항에 맞게 UI를 설계할 때 사용하는 도구

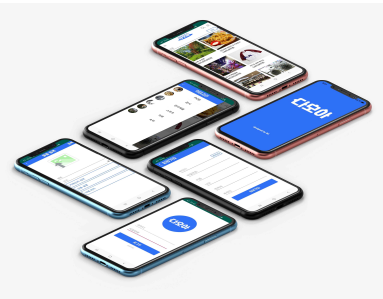
ㄱ. 와이어프레임 [실기 빈출]

: 기획 초기 단계에서 제작하는 것으로 페이지에 대한 대략적인 레이아웃이나 UI 요소 등에 대한 뼈대를 설계

- 와이어프레임 툴 : 파워포인트, 키노트, 스케치, 일러스트, 포토샵 등(손으로 그린 UI도 와이어프레임에 포함)

ㄴ. 목업

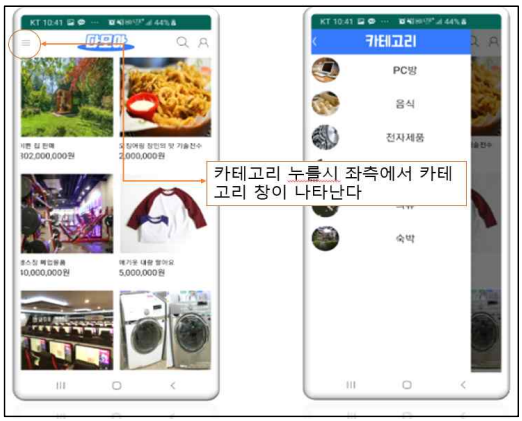
- 와이어프레임보다 좀 더 실제 화면과 유사하게 만드는 정적인 형태의 모형(아래 사진 참고)
- 목업 툴 : 파워 목업, 발사믹 목업 등



현업에 간다면 결과물 디자인을 할 때 정말 많이 쓰며 우스갯소리로 '디자인의 완성은 목업'이라는 말이 있음 그 정도로 목업 작업을 하면 결과물이 좀 있어 보임 (주로 포토샵 무료 배포 파일을 많이 씀, 궁금하거나 디자인쪽에 관심이 많은 사람이라면 구글에 '포토샵 무료 목업 파일' 이라고 치면 많이 나옴)

ㄷ. 스토리보드

- 와이어프레임에 콘텐츠에 대한 설명이나 페이지 간 이동 흐름 등을 추가한 문서
- 디자이너와 개발자가 최종적으로 참고하는 작업 지침서
- 서비스 구축을 위한 모든 정보가 담겨 있어야 함
- 스토리보드 툴 : 파워포인트, 키노트, 스케치, Axure 등

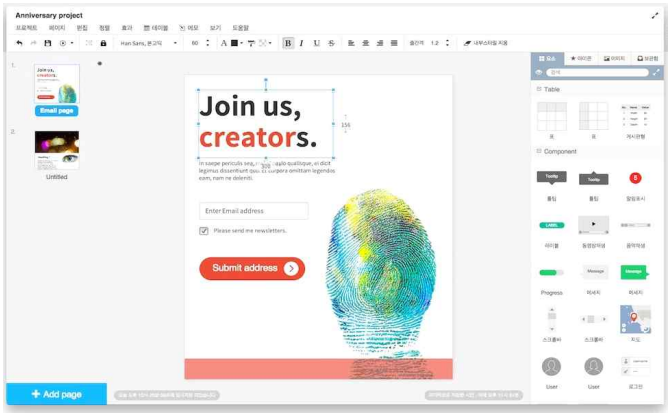


UI에서 어떤 버튼을 누를시 뭐가 나온다는 식의 순서도와 담겨야 하는 기능 등 모든 것을 포함한 최종 지침서 (종종 디자이너와 개발자가 구현여부를 가지고 싸우곤 함)

프로토타입

: 와이어프레임이나 스토리보드 등에 인터랙션을 적용해 실제 구현된 것처럼 테스트가 가능한 동적인 형태의 모형

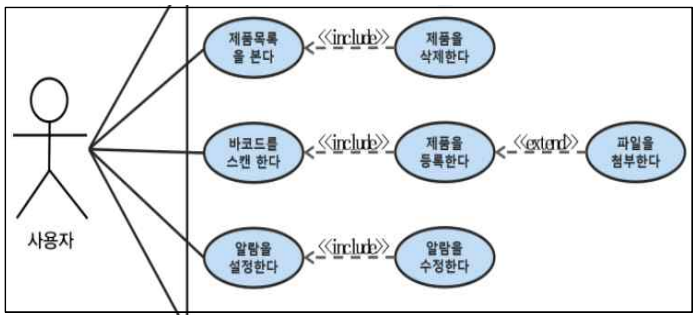
- 작성 방법에 따라 페이퍼/디지털 프로토타입으로 나눔
- 프로토타입 툴 : HTML/CSS, Axure, Flinto, 네이버 포로토나우, 카카오 오븐 등



출처 : https://ovenapp.io/

유스케이스 [필기 빈출],[실기 빈출]

: 사용자 측면에서의 요구사항으로 사용자가 원하는 목표를 달성하기 위해 수행할 내용 기술



사용자의 관점에서 어떠한 내용이 필요한지에 대한 순서도라고 생각하면 됨

12. UI요구사항 확인

UI 요구사항 확인의 개요

: 새로 개발할 시스템에 적용할 UI 관련 요구사항을 조사해서 작성하는 단계

ㄱ. 목표 정의

: 사용자들을 대상으로 인터뷰를 하고 사용자들의 의견이 수렴된 비즈니스 요구사항을 정의

- 인터뷰 진행 시 유의 사항

a. 사업적, 기술적 요구사항을 명확히 이해
b. 가능한 개별적인 진행
c. 가능한 한 시간을 넘기지 않는 것이 좋음
d. 사용자 리서치 시작 전에 해야 함

ㄴ. 활동 사항 정의

: 조사한 요구사항을 토대로 앞으로 해야 할 활동 사항을 정의

- 기술의 발전 가능성을 파악하고 UI 디자인의 방향 제시

ㄷ. UI 요구사항 작성

: 여러 경로로 수집된 사용자의 요구사항을 검토하고 분석하여 UI 개발 목적에 맞게 작성해야 함

- 실 사용자 중심으로 작성

- 여러 사람의 인터뷰를 통해 다양한 의견을 수렴하여 작성

ㄹ. 요구사항 요소 확인

: 파악된 요구사항 요소의 종류와 각각의 표현 방식 등을 검토

- 요구사항 요소 : 데이터 요구, 기능 요구, 제품/서비스의 품질, 제약 사항

ㅁ. 정황 시나리오 작성

: 사용자의 요구사항을 도출하기 위해 작성

- 사용자가 목표를 달성하기 위해 수행하는 방법을 순차적으로 묘사

- 개발하는 서비스의 모습을 상상하는 첫 번째 단계로 사용자 관점에서 시나리오를 작성해야 함

ㅂ. 요구사항 작성

: 정황 시나리오를 토대로 작성

<I-III 애플리케이션 설계>

13. 소프트웨어 아키텍처

소프트웨어 아키텍처의 설계

소프트웨어 아키텍처

: 소프트웨어의 골격이 되는 기본 구조

- 소프트웨어를 구성하는 요소들 간의 관계를 표현하는 시스템의 구조 또는 구조체

- 좋은 품질을 유지하면서 사용자의 비 기능적 요구사항으로 나타난 제약을 반영하고, 기능적 요구사항을 구현하는 방법을 찾는 해결 과정

- 애플리케이션의 분할 방법과 분할된 모듈에 할당될 기능, 모듈 간의 인터페이스 등을 결정

모듈화

: 소프트웨어의 성능을 향상하거나 시스템의 수정 및 재사용, 유지 관리 등이 용이하도록 시스템의 기능들을 모듈 단위로 나누는 것

- 모듈의 크기와 개수는 반비례관계, 개수와 통합 비용은 비례 관계

추상화

: 문제의 전체를 설계 후 세분화하여 구체화하는 과정

- 완전한 시스템을 구축하기 전에 그 시스템과 유사한 모델을 만들어 여러 가지 요인들을 테스트할 수 있음

- 최소 비용으로 실제 상황에 대처할 수 있고 시스템의 구조 및 구성을 대략적으로 파악할 수 있음

추상화 유형

- 과정 추상화 : 전반적인 흐름만 파악

- 데이터 추상화 : 데이터의 세부사항은 정의하지 않고 구조를 대표할 수 있는 표현으로 대체

- 제어 추상화 : 이벤트의 발생의 세부사항은 정의하지 않고 구조를 대표할 수 있는 표현으로 대체

단계적 분해

: 문제를 상위 중요 개념으로부터 하위의 개념으로 구체화하는 분할 기법

- 추상화의 반복으로 세분화

정보 은닉

: 한 모듈 내부에 포함된 정보들을 감추어 다른 모듈이 접근하거나 변경하지 못하도록 하는 기법

- 다른 모듈과 커뮤니케이션을 할 때는 필요한 정보만 인터페이스를 통해 주고받음

- 모듈을 독립적으로 수행하기 때문에 다른 모듈에 영향을 주지 않아 수정, 시험, 유지보수가 용이함

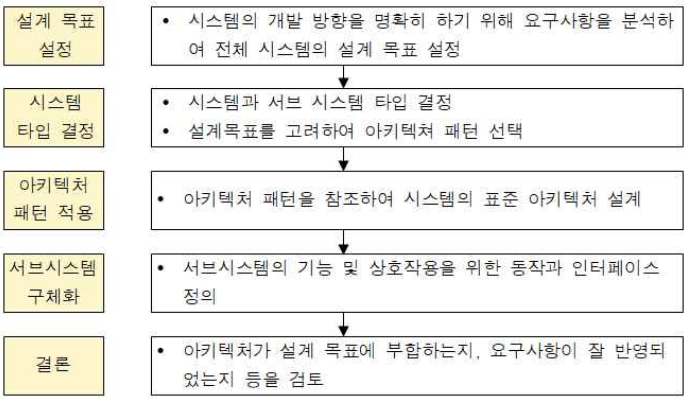
(객체 지향 프로그래밍에서의 캡슐화를 생각하면 됨)

소프트웨어 아키텍처의 품질 속성

: 소프트웨어 아키텍처가 이해 관계자들이 요구하는 수준의 품질을 유지하고 보장할 수 있게 설계되었는지를 확인하기 위

해 품질 요소들을 구체화시켜 놓은 것

소프트웨어 아키텍처의 설계 과정



14. 아키텍처 패턴

아키텍처 패턴 [필기 빈출],[실기 빈출]

: 아키텍처를 설계할 때 참조할 수 있는 전형적인 해결 방식 또는 예제

- 시스템의 구조를 구성하기 위한 기본적인 틀을 제공
- 서브시스템과 그 역할이 정의되어 있어 서브시스템 사이 관계와 규칙, 지침 등이 포함되어 있음

아키텍처 패턴의 장점

- 개발 시간을 단축시킴
- 고품질의 소프트웨어 생산 가능
- 검증된 구조로 작업을 하여 안정적인 개발 가능
- 시스템 구조를 이해하기 쉬워 개발에 참여하지 않아도 유지보수가 쉬움

레이어 패턴

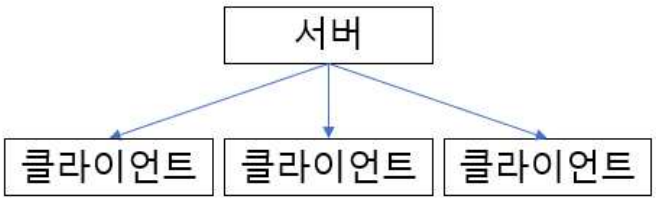
: 시스템을 계층으로 구분하여 구성

- 각각의 서브 시스템이 계층 구조를 이룸
- 상위 계층은 하위 계층에 대한 서비스 제공자가 되고 하위 계층은 상위 계층의 클라이언트가 됨
- 마주 보는 두 계층 사이에만 상호 작용이 이루어짐
- 특정 계층만을 교체해 시스템을 개선하는 것이 가능

클라이언트-서버 패턴

: 하나의 서버와 다수의 클라이언트로 구성

- 사용자는 클라이언트를 통해 서버에 요청하고 응답을 받아 사용자에게 제공
- 서버는 클라이언트의 요청에 대비해 항상 대기 상태 유지
- 요청을 위하여 동기화되는 경우를 제외하고는 서로 독립적임



파이프-필터 패턴 [실기 빈출]

: 데이터 스트림(연결지향통신에서, 전송된 정보를 수집하거나 정보를 전송할 때 사용되는 디지털 방식으로 암호화 된 일관된 신호의 흐름) 절차의 각 단계를 필터 컴포넌트로 캡슐화하여 파이프를 통해 데이터를 전송

- 필터 컴포넌트는 재사용성이 좋고 추가가 쉬워 확장이 용이
- 필터 컴포넌트를 재배치하여 다양한 파이프라인 구축 가능
- 데이터 변환, 버퍼링, 동기화 등에 사용

모델-뷰-컨트롤러 패턴(MVC패턴) [필기 빈출],[실기 빈출]

: 시스템을 아래의 3개의 부분으로 구조화

- 모델(Model) : 서브시스템의 핵심 기능과 데이터 보관
- 뷰(View) : 사용자에게 보여주는 화면단
- 컨트롤러(Controller) : 사용자로부터 받은 입력 처리(관제탑)
- 각 부분은 별도로 분리되어 있어 서로 영향을 받지 않고 독립적인 개발 작업 수행
- 여러 개의 뷰를 만들 수 있어 한 개의 모델에 여러 개의 뷰를 필요로 하는 대화형 애플리케이션에 적합

초심자를 위한 Tip

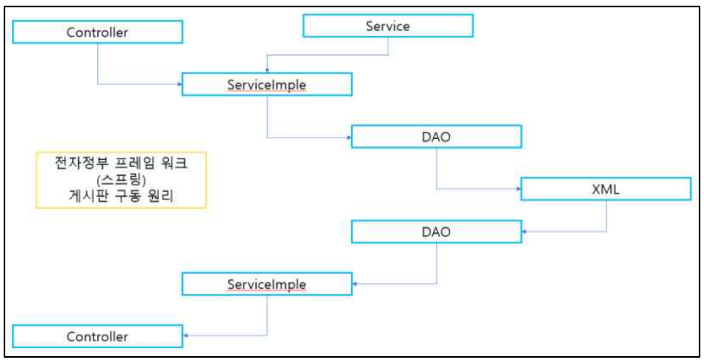
MVC패턴은 현업에서도 정말 많이 사용하는 패턴이다.

특히나 국가에서 많이 사용하는 스프링 기반의 '전자정부프레임워크'라는 프레임워크에서 많이 애용하는 방법으로, 대부분의 한국 신입 프로그래머들이 국가 SI기업에서 프로젝트를 맡게 된다면 MVC패턴을 경험할 확률이 높을 것이다. 아래의 그림은 전자정부 프레임워크에서의 MVC패턴에 대해서 정리해 놓은 것이니 조금 더 궁금한 사람들은 블로그의 글을 읽어 보면 됨.

(출처 : threeidiotscoding.tistory.com/47?category=1125492)

참고로 MVC패턴은 꼭 지켜야 한다는 것이기 보다는 SI특성상 사람들이 자주 나가고 들어오고 하면서 담당자가 자주 바뀌는데 그럴 경우 시스템의 유지보수를 원활하게 하기 위한 방법 중 하나이다. 지키지 않고 그냥 더티 코딩 하는 사람들도 정말 많다(코딩 패턴이 구조화&정형화 되어 있으면 다음 사람이

코드를 볼 때 훨씬 편하고 빠르게 볼 수 있음)



기타 패턴

ㄱ. 마스터-슬레이브 패턴 : 마스터 컴포넌트에서 슬레이브 컴포넌트로 작업을 분할한 후 슬레이브 컴포넌트에서 처리된 결과물을 다시 돌려받는 방식

ㄴ. 브로커 패턴 : 사용자가 원하는 서비스와 특성을 브로커 컴포넌트에 요청하면 브로커 컴포넌트가 요청에 맞는 컴포넌트를 연결

ㄷ. 피어-투-피어 패턴 : 피어를 하나의 컴포넌트로 간주하여 각 피어는 클라이언트 또는 서버가 될 수도 있음

ㄹ. 이벤트-버스 패턴 : 소스가 특정 채널에 이벤트 메시지를 발행하면 해당 채널을 구독한 리스너들이 메시지를 받아 이벤트를 처리하는 방식

ㅁ. 블랙보드 패턴 : 모든 컴포넌트들이 공유 데이터 저장소와 블랙보드 컴포넌트에 접근이 가능하여 검색을 통해 블랙보드에서 원하는 데이터를 찾을 수 있음

ㅂ. 인터프리터 패턴 : 프로그램 코드의 각 라인을 수행하는 방법을 지정하고 기호마다 클래스를 갖도록 구성

15. 객체지향 [필기 빈출],[실기 빈출]

객체지향

: 현실 세계의 개체를 기계의 부품처럼 하나의 객체(Object)로 만들어 소프트웨어를 개발할 때 객체를 조립하여 작성할 수 있는 기법

- 구조적 기법의 문제점을 해결하기 위해 사용
- 소프트웨어의 재사용 및 확장이 용이하여 고품질의 소프트웨어를 빠르게 개발할 수 있어 유지보수가 쉬움
- 복잡한 구조를 단계적이고 계층적이게 표현
- 멀티미디어 데이터 및 병렬 처리 지원
- 구성 요소 : 객체, 클래스, 캡슐화, 상속, 다형성

객체

: 물리적으로 존재하거나 추상적으로 생각할 수 있는 것 중 자신의 속성을 가지고 있고 다른 것과 식별 가능한 것

초심자를 위함 Tip

객체에 대해서는 너무 깊게 생각하지 말고 아 그렇구나 하고 넘어갈 것 처음보거나 이론으로만 봐서는 솔직히 잘 와 닿지 않음. 다만 코딩을 하거나 계속 공부를 하다보면 '아?..대충 이런 걸 객체라고 하는구나.' 하고 느낌이 올 때가 있음. 물론 사람마다 느낌이 오는 속도가 다르기 때문에 느낌이 올 때까지 기본적인 정의만 숙지하고 계속 신경 쓰면서 보면 됨 심지어 구글 검색으로 블로그 찾아보면 알겠지만 죄다 정의가 조금씩 다름

데이터

: 객체가 가지고 있는 정보

함수

: 하나의 특정한 목적의 작업을 수행하기 위해 독립적으로 설계된 코드의 집합

객체의 특성

- 객체는 독립적으로 식별 가능한 이름을 가지고 있음
- 객체의 상태는 시간에 따라 변함
- 객체 간의 상호 연관성에 의해 관계 형성
- 객체가 반응할 수 있는 메시지의 집합을 행위라고 하며 객체는 행위의 특징을 나타낼 수 있음
- 객체는 일정한 기억 장소를 가지고 있음

클래스

: 객체 지향 프로그래밍에서 특정 객체를 생성하기 위해 변수와 메소드를 정의하는 일종의 틀이다

- 인스턴스 : 클래스에 속한 각각의 객체
- 인스턴스화 : 클래스로부터 새로운 객체를 생성하는 것
- 슈퍼 클래스 : 특정 클래스의 부모(상위) 클래스
- 서브 클래스 : 특정 클래스의 자식(하위) 클래스

캡슐화

: 객체의 속성과 행위를 하나로 묶고, 실제 구현 내용 일부를 외부에 감추어 은닉하는 방법

- 인터페이스를 제외한 세부 내용이 은폐되어 외부에서 접근이 제한적이기 때문에 외부에서 변경하기 어려움(핵심은 보안)

상속

: 이미 정의된 상위 클래스의 모든 속성과 연산을 하위 클래스가 물려받는 것

- 하위 클래스는 부모 클래스의 모든 속성과 연산을 다시 정의하지 않고 사용할 수 있음
- 하위 클래스는 상속받은 속성 외에 새로운 속성과 연산을 첨가하여 사용할 수 있음
- 객체와 클래스의 재사용을 높이는 중요한 개념
- 다중 상속 : 한 개의 클래스가 두 개 이상의 상위 클래스로

부터 상속받는 것

다형성

: 하나의 객체가 여러 가지 타입을 가질 수 있는 것

- 객체지향의 오버로딩의 개념(함수 다중 정의)

<Ⅰ-Ⅳ 인터페이스 설계>

16. 시스템 인터페이스 요구사항 분석

시스템 인터페이스 [실기 빈출]

: 시스템 인터페이스는 독립적으로 떨어져 있는 시스템들끼리 서로 연동하여 상호작용하기 위한 접속 방법이나 규칙을 말함

- 시스템 인터페이스 요구사항은 개발을 목표로 하는 시스템과 외부 시스템을 연동하는데 필요한 시스템 인터페이스에 대한 요구사항을 기술한 것

- 시스템 요구사항 명세서 포함 요소 : 인터페이스 이름, 연계 대상 시스템, 연계 범위 및 내용, 연계 방식, 송신 데이터, 인터페이스 주기, 기타 고려사항 등

시스템 인터페이스 요구사항 분석

: 요구사항 명세서에서 요구사항을 기능적 요구사항과 비 기능적 요구사항으로 분류하고 조직화하여 요구사항 명세를 구체화하고 이를 이해관계자에게 전달하는 일련의 과정

ㄱ. 기능적 요구사항

: 시스템이 무엇을 하고 어떤 기능을 하는 가

ㄴ. 비 기능적 요구사항 : 시스템이나 프로젝트 개발 과정 등에서 지켜야 할 제약 사항

- 요구사항의 분해가 필요한 경우 세분화 할 수 있음

시스템 인터페이스 요구사항 분석 절차

17. 인터페이스 요구사항 검증

요구사항 검증

: 인터페이스의 설계 및 구현 전 사용자들의 요구사항이 요구사항 명세서에 정확하고 완전하게 기술되었는지 검토하고 개발 범위의 기준인 베이스라인을 설정하는 것

인터페이스 요구사항 검토 계획 수립(다음 페이지에 표 있음)

검토 기준 및 방법	• 프로젝트 규모, 참여 인력 검토 기간 등을 고려하여 검토 기준 및 방법을 정함
참여자	• 프로젝트 규모에 따라 이해관계자들을 파악하여 요구사항 검토 참여자를 선정
체크리스트	• 완전성, 일관성, 명확성 등의 항목을 점검할 수 있는 체크리스트 작성
관련 자료	• 인터페이스 요구사항 검토에 필요한 자료들을 준비
일정	• 인터페이스 요구사항 검토 일정을 정함

인터페이스 요구사항 검토 및 오류 수정

: 체크리스트의 항목에 따라 인터페이스 요구사항 명세서 검토

- 요구사항 검토 시 오류가 발견되면 이를 수정할 수 있도록 오류 목록과 시정 조치서 작성

- 시정 조치서를 작성할 경우 조치가 완료되었는지를 확인하여 조치가 완료되면 인터페이스 요구사항 검토 작업을 완료

인터페이스 요구사항 베이스라인 설정

- 검증된 인터페이스 요구사항은 주요 의사 결정자에게 공식적으로 승인을 받음

- 소프트웨어 설계 및 구현을 위해 요구사항 명세서의 베이스라인 설정

요구사항 검증 방법 [필기 빈출],[실기 빈출]

요구사항 검토

: 요구사항 명세서의 결함 여부를 검토 담당자들이 수작업으로 분석하는 방법

ㄱ. 동료검토 : 명세서 작성자가 직접 설명하는걸 동료들이 들으면서 결함을 발견하는 방법

ㄴ. 워크스루 : 검토 회의 전 미리 명세서를 배포하여 사전 검토 후 짧은 회의를 통해 결함을 발견하는 방법

ㄷ. 인스펙션 : 명세서 작성자를 제외한 다른 검토 전문가들이 명세서를 확인하면서 결함을 발견하는 방법

ㄹ. 프로토타이핑 : 요구사항을 파악하기 위해 실제 개발될 소프트웨어에 대한 견본품을 만들어서 최종 결과물을 예측

ㅁ. 테스트 설계 : 테스트 케이스를 생성하여 이후에 요구사항이 현실적으로 테스트 가능한지 검토

ㅂ. CASE(Computer Aided Software Engineering)도구 활용
: 일관성 분석을 통해 요구사항 변경사항의 추적, 분석, 관리하고 표준 준수 여부를 확인

인터페이스 요구사항 검증의 주요 항목

- 완전성 : 모든 요구사항이 누락되지 않고 반영 되었는가
- 일관성 : 요구사항이 모순되거나 충돌되는 점 없이 일관성을 유지 하는가
- 명확성 : 모든 참여자가 요구사항을 명확하게 이해할 수 있는가
- 기능성 : 요구사항이 '어떻게'보다 '무엇을'에 중점을 두고 있는가
- 검증 가능성 : 요구사항이 사용자의 요구를 모두 만족하고 개발된 소프트웨어가 사용자의 요구 내용과 일치하는지를 검증할 수 있는가
- 추적 가능성 : 요구사항 명세서와 설계서를 추적할 수 있는가
- 변경 용이성 : 요구사항 명세서의 변경이 쉽도록 작성 되었는가

18. 인터페이스 시스템 식별

개발 시스템 식별

: 인터페이스 관련 자료들을 기반으로 개발하고자 하는 시스템의 상세 식별 정보를 정의하고 목록을 작성함

- 시스템 아키텍쳐 : 시스템 내부에서 하위 시스템이 어떻게 상호작용하는지 파악할 수 있도록 구성이나 동작원리를 나타냄
- 유스케이스 : 사용자의 요구사항을 기능 단위로 표현

내·외부 시스템 식별

- 인터페이스 관련 자료들을 기반으로 개발할 시스템과 연계할 시스템들의 상세 식별 정보를 정의하고 목록을 작성

내·외부 시스템 환경 및 관리 주체 식별

- 연계할 시스템 접속에 필요한 IP, URL, Port 정보 등 시스템의 실제 운용 환경 및 하드웨어를 실제적으로 관리하는 담당자를 확인

내•외부 시스템 네트워크 연결 정보 식별

- 내•외부 시스템을 연계하는데 필요한 네트워크 연결 정보 확인

인터페이스 식별

- 인터페이스 요구사항 명세서와 인터페이스 요구사항 목록을 기반으로 개발할 시스템과 연계할 시스템 사이의 인터페이스를 식별하고 목록을 작성

인터페이스 시스템 식별

- 인터페이스별로 인터페이스에 참여하는 시스템들을 송신 시스템과 수신 시스템으로 구분하여 작성

19. 송수신 데이터 식별

식별 대상 데이터

: 송수신 사이에 교환되는 데이터

- 규격화된 표준 형식에 따라 전송

인터페이스 표준 항목

- 송수신 시스템을 연계하는데 표준적으로 필요한 데이터

ㄱ. 시스템 공통부 : 시스템 연동 시 필요한 공통 정보

ㄴ. 거래 공통부 : 시스템이 연동된 후 송수신되는 데이터를 처리할 때 필요한 정보

송수신 데이터 항목 : 송수신 시스템이 업무를 수행하는데 사용하는 데이터

공통 코드 : 시스템들에서 공통적으로 사용하는 코드

정보 흐름 식별

: 개발할 시스템과 내•외부 시스템 사이에서 전송되는 정보들의 방향성 식별

송수신 데이터 식별

- 개발할 시스템과 연계할 시스템 사이의 정보 흐름과 데이터베이스 산출물을 기반으로 식별
- 인터페이스 표준 항목과 송수신 데이터 항목 식별
- 코드성 데이터 항목 식별

20. 인터페이스 방법 명세화

인터페이스 방법 명세화의 개념
: 내•외부 시스템이 연계하여 작동할 때 인터페이스별 송수신 방법, 송수신 데이터, 오류 식별 및 처리 방안에 대한 내용을 문서화해놓은 것

시스템 연계 기술 [실기 빈출]
:개발할 시스템과 내•외부 시스템을 연계될 때 사용하는 기술

ㄱ. DB Link : DB에서 제공하는 DB Link 객체 이용

ㄴ. API/Open API : 송신 시스템의 DB에서 데이터를 읽어 와 제공하는 Application Programming Interface 프로그램

ㄷ. 연계 솔루션 : EAI(Enterprise Application Integration) 서버와 송수신 시스템에 설치되는 클라이언트 이용

ㄹ. EAI : 기업 응용 프로그램 통합, 기업용 응용 프로그램의 구조적 통합 방안(전사적 응용 프로그램 통합이라고도 함)

ㅁ. Socket : 서버에서 소켓을 생성하여 클라이언트의 통신 요청 시 클라이언트와 연결하여 통신하는 네트워크 기술

ㅂ. Web Service : 웹 서비스에서 WSDL, UDDI, SOAP 프로토콜을 이용하여 연계하는 서비스

초심자를 위한 Tip [실기 빈출]
WSDL(Web Service Description Language)
: 웹 서비스 기술언어 또는 기술된 정의 파일의 총칭으로 XML로 기술된다. 웹 서비스의 구체적 내용이 기술되어 있어 서비스 제공 장소, 서비스 메시지 포맷, 프로토콜 등이 기술된다.

UDDI(Universal Description, Discovery Integration)
: 웹 서비스 관련 정보의 공개와 탐색을 위한 표준이다. 서비스 제공자는 서비스 소비자에게 이미 알려진 온라인 저장소에 그들이 제공하는 서비스 목록들을 저장하게 되고, 서비스 소비자들은 그 저장소에 접근함으로써 원하는 서비스들의 목록을 찾을 수 있게 된다.

SOAP (Simple Object Access Protocol)
: 일반적으로 널리 알려진 HTTP, HTTPS, SMTP 등을 통해 XML 기반의 메시지를 컴퓨터 네트워크 상에서 교환하는 프로토콜이다. SOAP은 웹 서비스에서 기본적인 메시지를 전달하는 기반이 된다

인터페이스 통신 유형
: 데이터를 송수신 하는 형태

ㄱ. 단방향 : 시스템에서 거래 요청 후 응답 없음

ㄴ. 동기 : 시스템에서 거래 요청 후 응답이 올 때까지 대기

ㄷ. 비동기 : 시스템에서 거래를 요청 후 응답이 올 때까지 다른 작업을 하면서 대기

인터페이스 처리 유형
: 송수신 데이터를 어떤 형태로 처리할 것인지에 대한 방식
- 실시간 방식, 지연 처리 방식, 배치 방식(대량 데이터 처리)

인터페이스 발생 주기
: 송수신 데이터가 전송되어 인터페이스가 사용되는 주기

송수신 방법 명세화
: 각각의 인터페이스에 대해 연계 방식, 통신 유형, 처리 유형, 발생 주기 등 송수신 방법을 정의하고 명세

송수신 데이터 명세화
: 인터페이스 시 필요한 송수신 데이터에 대한 명세 작성

오류 식별 및 처리 방안 명세화
: 인터페이스 시 발생할 수 있는 오류를 식별하고 오류 처리 방안에 대한 명세 작성

21. 시스템 인터페이스 설계서 작성

시스템 인터페이스 설계서
: 시스템의 인터페이스 현황을 확인하기 위해 시스템이 갖는 인터페이스 목록과 상세 데이터 명세를 정의한 문서

시스템 인터페이스 목록 작성
: 업무 시스템과 내•외부 시스템 간 데이터를 주고받는 경우에 사용하는 인터페이스에 대해 기술

시스템 인터페이스 정의서 작성
: 인터페이스별로 시스템 간의 연계를 위해 필요한 데이터 항목 및 구현 요건 등을 기술

22. 미들웨어 솔루션 명세

미들웨어의 개념 및 종류
: 운영체제와 응용 프로그램 사이에서 운영체제가 제공하는 서비스 이외에 추가적인 서비스를 제공하는 소프트웨어
- 표준화된 인터페이스를 제공하여 시스템 간의 데이터 교환에 일관성을 보장

미들웨어 솔루션 식별

: 개발 및 운용 환경에 사용될 미들웨어 솔루션을 확인하고 목록을 작성

미들웨어 솔루션 명세서 작성

: 미들웨어 솔루션 목록의 미들웨어 솔루션별로 관련 정보들을 상세하게 기술

Ⅱ. 소프트웨어 개발 (대단원2/5)

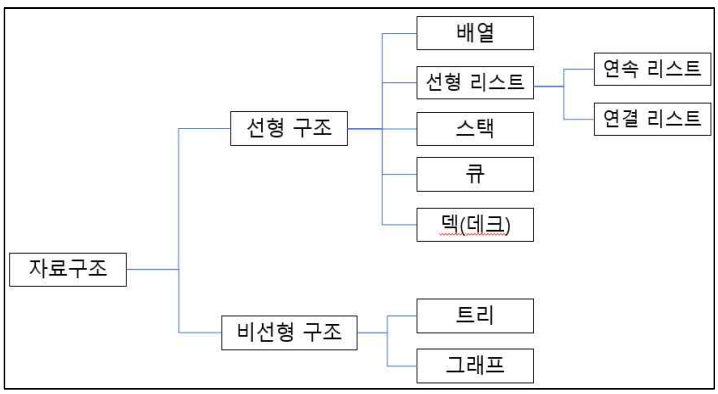
<Ⅱ-Ⅰ 데이터 입출력 구현>

23. 자료 구조 [필기 빈출],[실기 빈출]

'23. 자료 구조'에 해당하는 내용은 실기의 프로그래밍 파트를 이해하는데 기반이 되므로 급박하게 공부하는 것이 아니라면 하나하나 이해하고 암기하면서 가는 것이 추후 시간 절약에 많은 도움이 될 것임 한마디로 이 파트는 거를 타선이 없음

자료 구조

: 프로그램에서 사용하기 위한 자료를 기억장치의 공간 내에 저장하는 방법과 자료 간의 관계, 처리 방법 등을 저장 공간의 효율성 및 실행 간의 신속성을 높이기 위한 연구 분석하는 것



배열

: 동일한 자료형의 데이터들이 같은 크기로 나열되어 순서를 갖고 있는 집합 -> ex) {1, 2, 3, 4, 5}

- 첨자(index)를 이용하여 데이터에 접근
- 첨자의 개수에 따라 n차원 배열이라 부름
- 기억장소의 추가가 어렵고, 데이터 삭제 시 저장되어 있던 기억 장소는 빈 공간으로 남아있어 메모리 낭비가 발생
- 반복적인 데이터 처리 작업에 적합한 구조

a[0]	a[1]	a[2]	a[3]	...	a[n-1]

[크기가 n인 1차원 배열a]

a[0][0]	a[0][1]	a[0][2]	a[0][3]	...	a[0][n-1]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	...	a[1][n-1]
a[2][0]	a[2][1]	a[2][2]	a[2][3]	...	a[2][n-1]
...	...	...	...	...	...
a[n-1][0]	a[n-1][1]	a[n-1][2]	a[n-1][3]	...	a[n-1]

[크기가 nxn인 2차원 배열a]

리스트

: 자료를 나열한 목록(일반적으로 아래와 같은 표 형태)

이름	주소	출석번호	이메일
kim	seoul	99	abc@naver.com
you	tokyo	77	test@naver,com
lee	busan	33	as@daum.net

선형 리스트(Linear List)

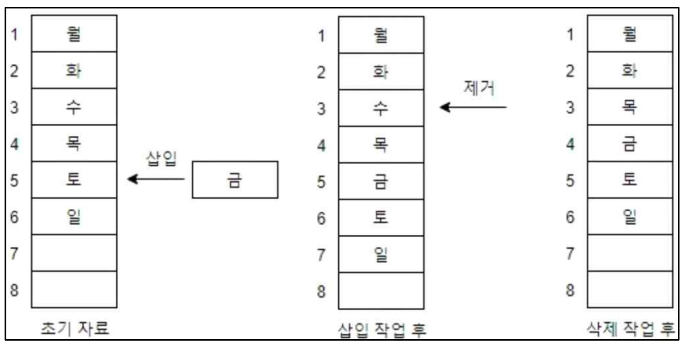
: 일정한 순서에 의해 나열된 자료 구조

- 배열을 이용하는 연속 리스트와 포인터를 이용하는 연결 리스트로 구분

ㄱ.연속 리스트(Contiguous List)

: 배열과 같이 연속되는 기억장소에 저장되는 자료 구조

- 중간에 데이터를 삽입하기 위해 연속된 빈 공간이 있어야 하며 삽입, 삭제 시 자료의 이동 필요



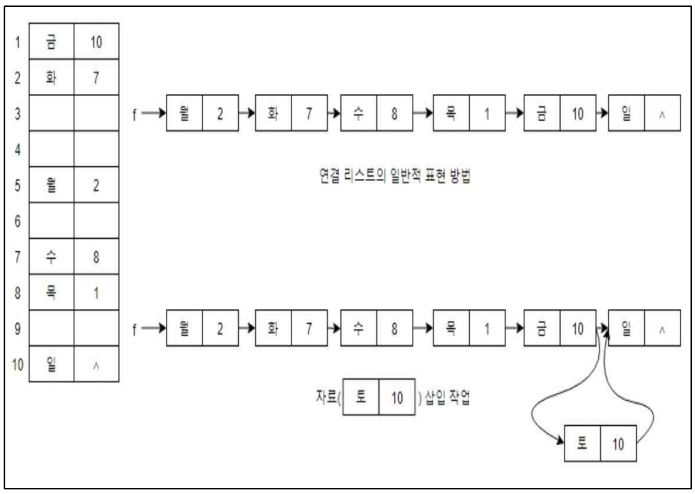
[연속 리스트]

ㄴ. 연결 리스트(Linked List)

: 자료 항목의 순서에 따라 노드의 포인터 부분을 이용하여 서로 연결시킨 자료 구조

- 노드의 삽입·삭제 작업이 용이
- 기억 공간이 연속적으로 놓여 있지 않아도 저장 가능

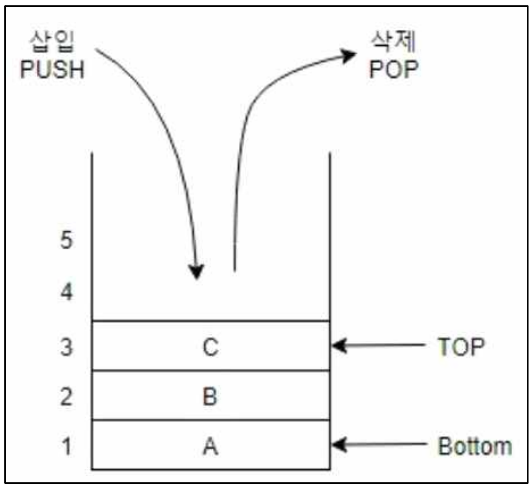
- 링크(포인터) 부분이 필요하기 때문에 순차 리스트에 비해 기억 공간의 이용 효율이 좋지 않다.
- 포인터를 찾는 시간이 필요하기 때문에 접근 속도가 느리다.



[연결 리스트]

스택(Stack)

- 리스트의 한쪽 끝으로만 자료의 삽입, 삭제 작업이 이루어지는 자료 구조
- 가장 나중에 삽입된 자료가 가장 먼저 삭제되는 후입선출(LIFO; Last In First Out) 방식으로 자료를 처리
- 꽉 채워져 있는 상태에서 삽입되면 오버플로(Overflow), 비어 있는 상태에서 삭제하면 언더플로(Underflow)가 발생

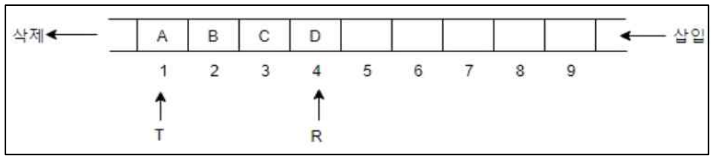


- TOP : 가장 마지막으로 삽입된 자료가 기억된 위치를 가리키는 요소
- Bottom : 스택의 가장 밑바닥

큐(Queue)

: 리스트의 한쪽에서는 삽입 작업이 이루어지고 다른 한쪽에서는 삭제 작업이 이루어지도록 구성한 자료 구조

- 가장 먼저 삽입된 자료가 가장 먼저 삭제되는 선입선출(FIFO; First In First Out) 방식으로 처리
- 시작과 끝을 표시하는 두 개의 포인터가 있다.

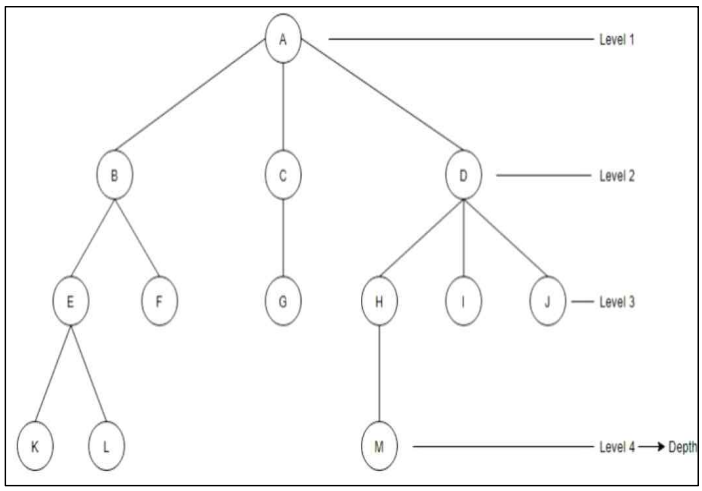


트리(Tree)

: 정점과 선분을 이용하여 사이클을 이루지 않도록 구성한 그래프(Graph)의 특수한 형태

ㄱ. 노드 : 하나의 기억 공간

ㄴ. 링크(Link) : 노드와 노드를 연결하는 선



ㄷ. 근 노드(Root Node) : 트리의 맨 위에 있는 노드 / A

ㄹ. 디그리(Degree, 차수) : 각 노드에서 뻗어 나온 가지의 수 / A=3, B=2, C=1, D=3

ㅁ. 단말 노드(Terminal Node) = 잎 노드(Leaf Node) : 자식이 하나도 없는 노드 / K, L, F, G, M, I, G

ㅂ. 자식 노드(Son Node) : 어떤 노드에 연결된 다음 레벨의 노드들 / D의 자식 노드 : H, I, J

ㅅ. 부모 노드(Parent Node) : 어떤 노드에 연결된 이전 레벨의 노드들 / E, F의 부모 노드 : B

ㅇ. 형제 노드(Brother Node, Sibling) : 동일한 부모를 갖는 노드들 / H의 형제 노드 : I, J

ㅈ. 트리의 디그리 : 노드들의 디그리 중에서 가장 많은 수 / 노드 A나 D가 3개의 디그리를 가지므로 위 트리의 디그리는 3

24. 데이터저장소 / DB / DBMS[필기 빈출],[실기 빈출]

'24. 데이터저장소/DB/DBMS'에 해당하는 내용은 DB 파트를 이해하는데 기반이 되므로 급박하게 공부하는 것이 아니라면 하나하나 이해하고 암기하면서 가는 것이 추후 시간 절약에 많은 도움이 될 것임 한마디로 이 파트는 거를 타선이 없음

데이터저장소

: 소프트웨어 개발 과정에서 다루어야 할 데이터들을 논리적인 구조로 조직화하거나 물리적인 공간에 구현한 것

- 논리 데이터저장소는 데이터 및 데이터 간의 연관성, 제약조건을 식별하여 논리적인 구조로 조직화한 것
- 물리 데이터저장소는 논리 데이터저장소에 저장된 데이터와

구조들을 하드웨어적인 저장장치에 저장한 것

데이터베이스(DB)

: 특정 조직의 업무를 수행하는데 필요한 데이터들의 모임

ㄱ. 통합된 데이터 : 자료의 중복을 최소화

ㄴ. 저장된 데이터 : 컴퓨터가 접근할 수 있는 저장 매체에 저장

ㄷ. 운영 데이터 : 조직의 고유한 업무를 수행하는데 필요

ㄹ. 공용 데이터 : 여러 시스템이 공동으로 소유하고 유지함

DBMS(DataBase Management System)

: 사용자와 데이터베이스 사이에서 사용자의 요구에 따라 정보를 생성해주고 데이터베이스를 관리하는 소프트웨어

- 기존의 파일 시스템이 가지는 데이터의 종속성과 중복성 문제를 해결하기 위해 제안된 시스템

DBMS의 기능

ㄱ. 정의 기능

: 데이터베이스에 저장될 데이터의 타입과 구조에 대해 명시하는 기능

ㄴ. 조작 기능

: 데이터를 검색, 갱신, 삽입, 삭제 등 처리하기 위해 사용자와 데이터베이스 간 인터페이스 수단을 제공하는 기능

ㄷ. 제어 기능

a. 데이터의 무결성이 유지되도록 제어

b. 사용자에게 허가된 데이터만 접근하도록 보안을 유지하고 권한을 검사

c. 여러 사용자가 동시에 접근하여 데이터를 처리할 때 정확성을 유지하도록 병행 제어

DBMS의 장단점

ㄱ. 장점

a. 데이터 독립성, 일관성, 무결성 유지

b. 보안 유지

c. 데이터 실시간 처리, 통합 관리, 표준화 가능

ㄴ. 단점

a. 전문가 부족

b. 전산화 비용 증가

c. 파일의 백업과 회복이 어려움

d. 시스템이 복잡함

25. 데이터 입출력

데이터 입출력

: 소프트웨어의 기능을 구현하기 위해 데이터베이스에 데이터를 입력, 출력하는 작업

SQL(Structured Query Language 구조화 질의어)

- 국제 표준 데이터베이스 언어

- 데이터 정의어, 조작어, 제어어로 구분됨

(대단원 3 데이터 베이스 구축에서 상세하게 다룸)

데이터 접속(Data Mapping)

: 프로그래밍 코드와 데이터베이스의 데이터를 연결하는 것

트랜잭션 [실기 빈출]

: 하나의 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 수행돼야 할 일련의 연산

ㄱ. TCL(Transaction Control Language) : 트랜잭션을 제어하기 위해 사용되는 명령어

ㄴ. COMMIT : 트랜잭션 처리가 정상적으로 종료되어 트랜잭션이 수행한 변경 내용을 데이터베이스에 반영

ㄷ. ROLLBACK : 트랜잭션 처리가 비정상적으로 종료되어 데이터베이스의 일관성이 깨졌을 때 트랜잭션이 행한 모든 변경 작업을 취소하고 이전 상태로 되돌림

ㄹ. SAVEPOINT(CHECKPOINT) : 트랜잭션 내에 ROLLBACK 할 위치인 저장점을 지정

절차형 SQL

: 프로그래밍 언어와 같이 연속적인 실행이나 분기, 반복 등의 제어가 가능한 SQL

- 단일 SQL문장으로 처리가 어려운 연속적인 작업을 처리하는데 적합

- BEGIN ~ END 형식의 블록 구조로 되어 있어 기능별 모듈화가 가능

<II-II 통합 구현>

26. 단위 모듈 구현

단위 모듈

: 소프트웨어 구현에 필요한 여러 동작 중 한 가지 동작을 수행하는 기능을 모듈로 구현한 것

- 사용자 또는 다른 모듈로부터 값을 전달받아 시작되는 작은 프로그램

단위 모듈 구현 순서

단위 기능 명세서 작성 -> 입출력 기능 구현 -> 알고리즘 구현

단위 기능 명세서 작성

: 단위 기능을 명세화한 문서

- 복잡한 시스템을 단순하게 구현하기 위한 추상화 작업이 필요
- 대형 시스템을 분해하여 단위 기능별로 구분하고 각 기능들로 계층적으로 구성하는 구조화 과정을 거침

입출력 기능 구현

: 단위 기능 명세서에서 정의한 데이터 형식에 따라 입출력 기능을 위한 알고리즘 및 데이터 구현

- 모듈 간 연동 또는 통신을 위한 데이터 구현

IPC(Inter Process Communication)

: 모듈 간 통신을 구현하기 위해 사용되는 프로그래밍 인터페이스 집합

a. 공유 메모리 : 다수의 프로세스가 공유 가능한 메모리를 구성하여 통신 수행

b. 소켓 : 네트워크 소켓을 이용하여 네트워크를 경유하는 통신 수행

c. 세마포어 : 공유 자원에 대한 접근 제어를 통해 통신 수행

d. 파이프 : 선입선출의 형태로 구성된 메모리를 여러 프로세스가 공유하여 통신 수행

e. 메시지 큐잉 : 메시지가 발생하면 이를 전달하는 형태로 통신 수행

알고리즘 구현

- 입출력 데이터를 바탕으로 단위 기능별 요구 사항들을 구현 가능 언어를 이용하여 모듈로 구현

27. 단위 모듈 테스트

단위 모듈 테스트 [필기 빈출],[실기 빈출]

: 모듈이 정해진 기능을 정확히 수행하는지 검증

- 단위 테스트라고도 하며 화이트박스 테스트와 블랙박스 테스트 기법 사용
- 시스템 수준의 오류는 발견할 수 없음

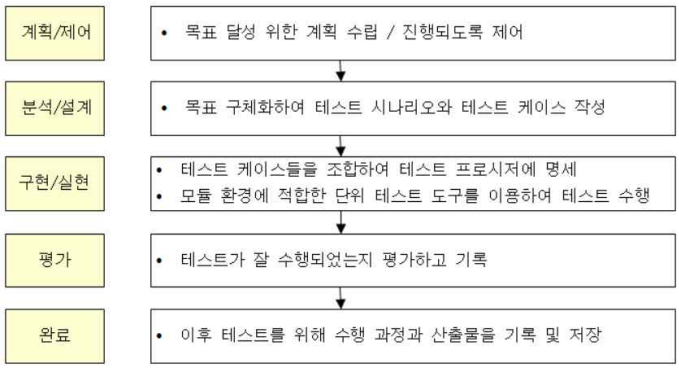
테스트 케이스

: 구현된 소프트웨어가 요구사항을 정확히 준수했는지 확인하기 위한 테스트 항목에 대한 명세서로 명세 기반 테스트의 설계 산출물에 해당

- 입력 데이터, 테스트 조건, 예상 결과 등을 모아 테스트 케이스를 만듬
- 테스트 케이스의 구성 요소 : 식별자, 테스트 항목, 입력 / 출력 명세, 환경 설정, 특수 절차 요구, 의존성 기술

테스트 프로세스

: 테스트를 위해 수행하는 작업이 테스트의 목적과 조건을 달성할 수 있도록 도와주는 과정



28. 개발 지원 도구

통합 개발 환경(IDE : Integrated Development Environment)

: 개발에 필요한 편집기, 컴파일러 디버거 등의 다양한 툴을 하나의 인터페이스로 통합하여 제공

- Eclipse, Visual Studio, Xcode, Android Studio, IDEA 등

빌드 도구

: 소스 코드 파일들을 컴퓨터에서 실행할 수 있는 제품 소프트웨어로 변환하는 과정 또는 결과물

- 소스 코드를 소프트웨어로 변환하는 과정에 필요한 전처리, 컴파일 등의 작업을 수행

ㄱ. Ant : 자바 프로젝트의 공식적인 빌드 도구

ㄴ. Maven : Ant의 대안으로 의존성을 설정하여 라이브러리 관리

ㄷ. Gradle : 안드로이드 스튜디오의 공식 빌드 도구

협업 도구

: 개발에 참여하는 사람들이 서로 다른 작업 환경에서 프로젝트를 수행할 수 있도록 도와주는 도구

- 협업 소프트웨어, 그룹웨어라고도 함

협업 도구의 종류

ㄱ. 프로젝트 및 일정 관리 : 구글 캘린더, 분더리스트, 트렐, 지라, 플로우 등

ㄴ. 정보 공유 및 커뮤니케이션 : 슬랙, 잔디, 태스크 월드 등

ㄷ. 디자인 : 스케치, 제플린 등

ㄹ. 아이디어 공유 : 에버노트 등

ㅁ. API 문서화 : 스웨거 등

ㅂ. Git 웹 호스팅 서비스 : 깃허브 등

<II-III 제품 소프트웨어 패키징>

29. 소프트웨어 패키징

소프트웨어 패키징

: 실행 파일을 묶어 배포용 설치 파일을 만드는 것

- 사용자 중심으로 진행
- 모듈화 하여 일반 배포 형태로 패키징

패키징시 고려사항

- 사용자의 운영체제, CPU, 메모리 등에 필요한 최소 환경 정의
- UI는 시각적인 자료와 함께 매뉴얼과 일치시켜 패키징
- 소프트웨어는 하드웨어와 함께 관리될 수 있도록 Managed Service 형태로 제공

패키징 작업 순서

기능 식별 -> 모듈화 -> 빌드 -> 사용자 환경 분석 -> 패키징 및 적용 시험 -> 패키징 변경 및 개선 -> 배포

30. 릴리즈 노트 작성 [실기 빈출]

릴리즈 노트(배포 노트 정도로 생각하면 됨)

: 개발 과정에서 정의된 릴리즈 정보를 고객에게 공유하기 위한 문서

- 테스트 진행 방법에 대한 결과와 소프트웨어 사양에 대한 개발팀의 정확한 준수 여부 파악
- 소프트웨어의 버전 관리 및 릴리즈 정보를 체계적으로 관리
- 소프트웨어 초기 배포, 출시 후 개선 사항을 적용한 추가 배포 시 제공

릴리즈 노트 초기 버전 작성 시 고려사항

- 정확하고 완전한 정보를 기반으로 개발팀에서 직접 현재 시제로 작성
- 신규 코드, 빌드 등의 이력이 정확하게 관리되어 변경 또는 개선된 항목에 대한 이력 정보들도 작성

릴리즈 노트 추가 버전 작성 시 고려사항

- 테스트 과정에서 베타 버전이 출시되거나 긴급 버그 수정, 업그레이드, 사용자 요청 등의 특수한 상황의 경우 작성
- 긴급 버그 수정 시 수정하는 경우 릴리즈 버전을 출시하고 그 번호를 포함한 모든 내용을 수정된 내용을 담음
- 요구사항에 의해 추가 혹은 수정된 경우 자체 기능 향상과는 다른 별도의 릴리즈 버전으로 출시하고 작성

31. 디지털 저작권 관리 [필기 빈출][실기 빈출]

저작권

: 창작자가 가지는 배타적 독점적 권리로 타인의 침해를 받지 않을 고유한 권한

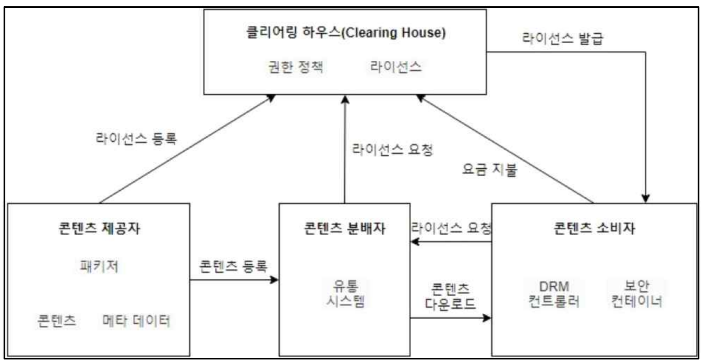
- 컴퓨터 프로그램처럼 복제하기 쉬운 저작물에 대해 저작권을 보호하는 방법을 저작권 보호 기술이라 함

디지털 저작권 관리(Digital Right Management)

: 저작권자가 배포한 디지털 콘텐츠가 저작권자가 의도한 용도로만 사용되도록 생성, 유통, 이용까지 전 과정에 걸쳐 사용되는 디지털 콘텐츠 관리 및 보호 기술

- 크기가 작은 경우 사용자가 콘텐츠를 요청하는 시점에 실시간 패키징 수행
- 크기가 큰 경우 미리 패키징을 수행 후 배포
- 종량제 방식을 적용한 소프트웨어의 경우 서비스의 실제 사용량을 측정하여 이용한 만큼 이용 부과

디지털 저작권 관리의 흐름도



ㄱ. 클리어링 하우스 : 저작권에 대한 사용 권한, 라이선스 발급, 사용량에 따른 결제 관리 등 수행

ㄴ. 콘텐츠 제공자 : 콘텐츠를 제공하는 저작권자

ㄷ. 패키저 : 콘텐츠를 메타 데이터와 함께 배포 가능한 형태로 묶어 암호화 하는 프로그램

ㄹ. 콘텐츠 분배자 : 암호화된 콘텐츠를 유통

ㅁ. 콘텐츠 소비자 : 콘텐츠를 구매해서 사용

ㅂ. DRM 컨트롤러 : 배포된 콘텐츠의 이용 권한을 통제하는 프로그램

ㅅ. 보안 컨테이너 : 콘텐츠 원본을 안전하게 유통하기 위한 전자적 보안 장치

디지털 저작권 관리의 기술 요소

- 암호화, 키 관리, 암호화 파일 생성, 식별 기술, 저작권 표현, 정책 관리, 크랙 방지, 인증

32. 소프트웨어 설치 매뉴얼 작성

소프트웨어 설치 매뉴얼

: 개발 초기에서부터 적용된 기준이나 사용자가 소프트웨어를 설치하는 과정에 필요한 내용을 기록한 문서

- 설치 시작부터 완료까지의 과정을 순서대로 설명

서문

- 문서 이력, 설치 매뉴얼의 주석, 설치 도구의 구성, 설치 환경 체크 항목 기술
- 설치 매뉴얼의 주석 : 주의 사항과 참고 사항 기술
- 설치 환경 체크 항목 : 사용자 환경, 응용 프로그램, 업그레이드 버전, 백업 폴더 확인

기본 사항

- 소프트웨어 개요, 설치 관련 파일, 설치 아이콘, 프로그램 삭제, 관련 추가 정보 설명

설치 매뉴얼 작성

- 사용자가 설치 과정을 이해하기 쉽게 설치 화면을 누락 없이 캡처하여 순서대로 설명
- 설치 화면 및 UI, 설치 이상 메시지, 설치 완료 및 결과, 설치 시 점검 사항, Network 환경 및 보안, 고객 지원 방법, FAQ, 준수 정보 &제한 보증에 대해 기술

33. 소프트웨어 사용자 매뉴얼 작성

소프트웨어 사용자 매뉴얼

: 사용자가 소프트웨어를 사용하는 과정에서 필요한 내용을 기록한 문서

서문

- 문서 이력, 사용자 매뉴얼의 주석, 기록 보관 내용 기술
- 기록 보관 내용 : 소프트웨어를 사용하면서 필요한 기술 지원이나 추가 정보를 얻기 위한 소프트웨어 등록 정보 기술

기본 사항

- 소프트웨어 개요, 사용 환경, 관리, 모델 버전별 특징, 기능 및 인터페이스의 특징, 구동 환경 설명

사용자 매뉴얼 작성

- 사용자가 사용방법을 이해하기 쉽도록 작성
- 사용자 화면 및 UI, 주요 기능 분류, 응용 프로그램 및 설정, 장치 연동, Network 환경, Profile 안내, 고객 지원 방법, 준수 정보 및 제한 보증에 대해 기술

34. 소프트웨어 버전 등록 [실기 빈출]

소프트웨어 패키징 형상 관리

: 형상관리는 소프트웨어의 변경 사항을 관리하기 위한 활동

형상 관리의 중요성

- 지속적으로 변경사항을 체계적으로 관리 및 추적할 수 있음
- 발견된 버그나 수정 사항을 추적
- 무절제한 변경 방지

형상 관리 기능

ㄱ. 형상 식별 : 대상에 이름과 관리 번호를 부여하고 계층 구조로 구분하여 수정 및 추적이 용이하도록 하는 작업

ㄴ. 버전 제어 : 소프트웨어 유지 보수 과정에서 생성된 다른 버전의 형상 항목을 관리하고 특정 절차와 도구를 결합하는 작업

ㄷ. 형상 통제 : 식별된 형상 항목에 대한 변경 요구를 검토하여 현재의 기준선이 잘 반영될 수 있도록 하는 작업

ㄹ. 형상 감사 : 기준선의 무결성을 평가하기 위해 확인, 검증, 검열 과정을 통해 공식적으로 승인하는 작업

ㅁ. 형상 기록 : 형상의 식별, 통제, 감사 작업의 결과를 기록, 관리하고 보고서를 작성하는 작업

소프트웨어 버전 등록 관련 주요 용어

ㄱ. 저장소(Repository) : 형상에 대한 정보들이 저장되어 있는 곳

ㄴ. 가져오기 : 아무것도 없는 저장소에 처음으로 파일 복사

ㄷ. 체크아웃 : 저장소에서 소스 파일, 버전 관리를 위한 파일을 받아옴

ㄹ. 체크인 : 체크아웃으로 받아온 파일을 수정 후 저장소에 새로운 버전으로 갱신

ㅁ. 커밋 : 체크인 수행 시 이전에 갱신된 내용이 있는 경우 충돌을 알리고 diff 도구를 이용해 수정한 후 갱신

ㅂ. 동기화 : 저장소에 있는 최신 버전을 동기화

35. 소프트웨어 버전 관리 도구

[필기 빈출],[실기 빈출]

공유 폴더 방식

: 버전 관리 자료가 로컬 컴퓨터의 공유 폴더에 저장되어 관리

클라이언트/서버 방식

: 버전 관리 자료가 서버에 저장되어 관리

- 서버의 자료를 자신의 PC로 복사하여 작업 후 변경 내용을 서버에 반영
- 모든 버전 관리는 서버에서 수행

분산 저장소 방식

: 버전 관리 자료가 하나의 원격 저장소와 분산된 PC의 로컬 저장소에 함께 저장되어 관리

- 원격 저장소의 자료를 자신의 로컬 저장소로 복사하여 작업 후 변경 내용을 로컬 저장소에서 버전 관리 후 이를 원격 저장소에 반영

Subversion(SVN)

- 아파치 소프트웨어 재단에서 2000년에 발표
- 클라이언트/서버 방식
- 모든 작업은 trunk 디렉토리에서 추가 작업은 branches 디렉토리 안에 별도의 디렉토리를 만들어 작업 후 trunk 디렉토리와 병합
- 커밋 시 커밋의 버전인 리버전이 1씩 증가
- 서버는 주로 유닉스에서 사용

Subversion(SVN)명령어

ㄱ. add : 새로운 파일이나 디렉토리를 관리 대상으로 지정

ㄴ. commit : add한 소스파일을 서버의 소스파일에 적용

ㄷ. update : 서버의 최신 commit 이력을 클라이언트 소스에 적용

ㄹ. checkout : 서버에서 버전 관리 정보와 소스 파일을 받아옴

ㅁ. import : 아무것도 없는 서버의 저장소에 맨 처음 소스 파일을 저장

ㅂ. export : 버전 관리 정보 빼고 소스 파일만 서버에서 받아옴

ㅅ. info : 지정된 파일에 대한 정보를 표시

ㅇ. diff : 지정된 파일이나 경로에 대해 이전 리버전과의 차이를 표시

ㅈ. merge : 다른 디렉토리에서 작업된 버전 관리 내역을 기본 개발 작업과 병행

초심자를 위한 Tip

svn을 쓰는 기업도 분명 있다 (주로 중소기업이 많이 씀) 그러나 요즘은 git이 워낙 대중화되고 유명하다보니 svn보다는 git을 더 많이 사용하는 추세(채용 공고를 보면 대부분이 git 경험자이다) 또한 svn은 자체 서버가 죽으면 아무도 작업을 못하는 불상사가 은근히 자주 발생하는데 git은 그럴 일이 없다. 조금 더 자세하게 알고 싶다면 아래의 블로그를 참고하자
https://dzzienki.tistory.com/46

Git

- 리누스 토르발스즈가 2005년에 개발
- 분산 저장소 방식
- 버전 관리가 지역 저장소에서 진행되어 버전 관리가 신속하게 처리되고, 원격 저장소나 네트워크에 문제가 있어도 작업 가능
- 브랜치를 이용하여 기본 버전 관리 틀에 영향을 주지않으면서 다양항 형태의 테스팅 가능
- 파일의 변화를 스냅샷으로 저장하고 이전 스냅샷의 포인터를 가져 버전의 흐름 파악 가능

Git 명령어

ㄱ. add : 작업 내역을 스테이징 영역에 추가하여 버전 관리 대상으로 지정

ㄴ. commit : 작업 내역을 지역 저장소에 저장

ㄷ. branch : 새로운 브런치 생성 / 삭제

ㄹ. checkout : 지정한 브런치로 이동

ㅁ. merge : 두 브랜치 병합

ㅂ. init : 지역 저장소 생성

ㅅ. remote add : 원격 저장소에 연결

ㅇ. push : 로컬 저장소의 변경 내용을 원격 저장소에 반영

ㅈ. fetch : 원격 저장소의 변경 이력만 지역 저장소에 반영

ㅊ. clone : 원격 저장소의 전체 내용을 지역 저장소로 복제

ㅋ. fork : 지정한 원격 저장소의 내용을 자신의 원격 저장소로 복제

36. 빌드 자동화 도구

빌드 자동화 도구

: 소스 코드를 컴파일한 후 여러 개의 모듈로 묶어 실행 파일로 만드는 과정을 포함하여 테스트 및 배포를 자동화하는 도구

Jenkins(젠킨스)

: 젠킨스(Jenkins)는 소프트웨어 개발 시 지속적 통합서비스를 제공하는 툴이다. 다수의 개발자들이 하나의 프로그램을 개발할 때 버전 충돌을 방지하기 위해 각자 작업한 내용을 공유 영역에 있는 Git등의 저장소에 빈번히 업로드함으로써 지속적 통합이 가능하도록 해 준다

- 서블릿 컨테이너에서 실행되는 서버 기반 도구
- 형상 관리 도구와 연동 가능

- Web GUI 제공으로 사용이 쉬움
- 여러 대의 컴퓨터를 이용한 분산 빌드나 테스트 가능

Gradle(그레이들)

: Groovy를 기반으로 한 오픈 소스 형태의 자동화 도구

(Groovy는 아파치의 자바 가상 머신에서 작동하는 동적 타이핑 프로그래밍 언어이다)

- 안드로이드 앱 개발 환경에 사용
- Java, C/C++, Python 등의 언어도 빌드 가능
- Groovy를 사용해서 만든 DSL(Domain-specific language)을 스크립트 언어로 사용
- 실행할 처리 명령들을 모아 태스크로 만든 후 태스크 단위로 실행
- 이전의 태스크를 재사용하거나 다른 시스템의 태스크를 공유하여 빌드의 속도를 향상시킬 수 있음

<II-IV 애플리케이션 테스트 관리>

37. 애플리케이션 테스트

애플리케이션 테스트

: 애플리케이션에 잠재된 결함을 찾아내는 과정

- 확인(Validation) : 개발된 소프트웨어가 요구사항을 만족시키는지 사용자의 입장에서 확인
- 검증(Verification) : 기능을 제대로 수행하고 명세서에 맞게 만들었는지 개발자의 입장에서 점검
- 테스트 전 개발한 소프트웨어의 유형을 분류하고 특성을 정리해서 중점적으로 테스트할 사항을 정리

애플리케이션 테스트의 필요성

- 미리 오류를 발견하고 새로운 오류의 유입 예방
- 사용자의 요구사항에 만족하는지 테스트해 제품의 신뢰도 향상

애플리케이션 테스트의 기본 원리

- 잠재적인 결함을 줄일 수 있지만 소프트웨어 자체 결함이 없다곤 할 수 없음
- 결함은 특정 모듈에 집중되어 있어 애플리케이션의 20%에 해당하는 코드에서 80%의 결함이 발견된다고 하여 파레토 법칙을 적용하기도 함
- 살충제 패러독스 현상을 방지하기 위해 테스트 케이스를 지속적으로 보완 및 개선
- 테스트를 정황에 따라 다르게 진행
- 결함을 모두 제거해도 사용자의 요구사항을 만족할 수 없으면 안 됨
- 작은 부분에서 시작해서 점점 확대하며 진행

38. 애플리케이션 테스트 분류

프로그램 실행 여부 [필기 빈출],[실기 빈출]

ㄱ. 정적 테스트

: 프로그램을 실행하지 않고 소스코드나 명세서를 분석하여 테스트

- 개발 초기에 결함을 발견할 수 있어 비용이 절감
- 워크 스루, 인스펙션, 코드 검사 등

ㄴ. 동적 테스트

: 프로그램을 실행하여 테스트

- 개발의 모든 단계에서 진행
- 블랙박스 테스트, 화이트 박스 테스트

테스트 기반 [실기 빈출]

ㄱ. 명세 기반 테스트

: 사용자의 요구사항을 테스트 케이스로 만들어 구현하고 있는지 확인하여 테스트

- 동등 분할, 경계 값 분석

ㄴ. 구조 기반 테스트

: 소프트웨어 내부 논리 흐름에 따라 테스트 케이스를 만들어 테스트

- 구문 기반, 결정 기반, 조건 기반, 결정 기반 등

ㄷ. 경험 기반 테스트

: 테스터의 경험을 기반으로 테스트

- 요구사항에 대한 명세가 부족하거나 시간의 제약이 있는 경우
- 에러 추정, 체크 리스트, 탐색적 테스팅

시각

ㄱ. 확인 테스트

: 사용자의 시각에서 결과를 테스트

- 요구사항을 만족하면서 정상적으로 동작이 되는지 테스트

ㄴ. 검증 테스트

: 개발자의 시각에서 과정을 테스트

- 명세서에 맞게 완성되었는지 테스트

목적에 따른 테스트 [실기 빈출]

ㄱ. 회복 테스트 : 결함을 주고 잘 복구되는지 테스트

ㄴ. 안전 테스트 : 시스템 보호 도구가 불법적인 침입으로부터 보호할 수 있는지 테스트

ㄷ. 강도 테스트 : 과부하 시 정상적으로 실행되는지 테스트

ㄹ. 성능 테스트 : 응답 시간, 처리량 등을 테스트

ㅁ. 구조 테스트 : 내부의 논리적인 경로, 소스 코드 복잡도 등을 평가

ㅂ. 회귀 테스트 : 변경 혹은 수정에 따른 새로운 결함이 없는

지를 테스트

ㅅ. 병행 테스트 : 기존의 소프트웨어와 변경된 소프트웨어에 동일한 데이터를 입력하여 결과를 비교하는 테스트

39. 동적 테스트 [필기 빈출],[실기 빈출]

화이트박스 테스트

: 모듈의 원시 코드를 오픈하여 논리적인 모든 경로를 한번 이상 실행하면서 테스트하여 테스트 케이스를 설계

- 테스트 과정의 초기에 진행
- 설계된 절차에 초점을 둔 구조적 테스트
- 모듈 안의 동작을 직접 관찰

화이트박스 테스트의 종류

ㄱ. 기초 경로 검사

: 테스트 케이스 설계자가 절차적 설계의 논리적 복잡성을 측정할 수 있게 해주는 테스트 기법

- 테스트 측정 결과를 통해 실행 경로의 기초를 정의

ㄴ.제어 구조 검사

a. 조건 검사 : 프로그램 내의 논리적 조건을 테스트

b. 루프 검사 : 프로그램 내의 반복 구조에 초점을 맞춰 테스트

c. 데이터 흐름 검사 : 프로그램 내의 변수의 정의와 사용의 위치에 초점을 맞춰 테스트

화이트박스 테스트 검증 기준

ㄱ. 문장 검증 기준 : 모든 구문이 한 번 이상 수행되도록 설계

ㄴ. 분기 검증 기준 : 모든 조건문이 한 번 이상 수행되도록 설계

ㄷ. 조건 검증 기준 : 모든 조건문에 대해 참/거짓인 경우가 한번 이상 수행되도록 설계

ㄹ. 분기/조건 기준 : 모든 조건문과 조건문에 포함된 개별 조건식의 결과가 참/거짓인 경우가 한번 이상 수행되도록 설계

블랙박스 테스트

: 소프트웨어가 수행할 특정 기능을 알기 위해 기능이 완전히 작동되는 것을 입증하는 기능 테스트

- 테스트 과정의 후반부에 진행
- 사용자의 요구사항 명세를 보면서 구현된 기능을 테스트
- 소프트웨어 인터페이스에서 실시

블랙박스 테스트의 종류

ㄱ. 동치(동등) 분할 검사 : 입력 자료에 초점을 맞춰 테스트 케이스를 만들고 검사

ㄴ. 경계값 분석 : 입력 조건의 경계값을 테스트 케이스로 선정하여 검사

ㄷ. 원인-효과 그래프 검사 : 입력 데이터 간의 관계과 출력의 영향을 미치는 상황을 분석 후 효용성이 높은 테스트 케이스를 선정하여 검사

ㄹ. 오류 예측 검사 : 과거 경험이나 확인자의 감각으로 테스트

ㅁ. 비교 검사 : 여러 프로그램에 동일한 테스트 자료를 제공하여 동일한 출력이 나오는지 확인하는 검사

40. 개발 단계에 따른 어플리케이션 테스트 [실기 빈출]

단위 테스트

: 코딩 직후 모듈이나 컴포넌트에 초점을 맞춰 하는 테스트

- 인터페이스, 외부적 I/O, 자료 구조 등을 검사
- 사용자의 요구사항을 기반으로 한 기능성 테스트를 최우선으로 수행

ㄱ. 구조 기반 테스트 : 화이트 박스 테스트를 시행하여 제어 흐름이나 조건 결정을 목적으로 함

ㄴ. 명세 기반 테스트 : 블랙 박스 테스트를 시행하여 동등 분할이나 경계값 분석을 목적으로 함

통합 테스트

: 단위 테스트가 완료된 모듈들을 결합하여 하나의 시스템으로 완성하는 과정에서 테스트

- 모듈 간 또는 통합된 컴포넌트 간 상호 작용 오류 검사

ㄱ.비점진적 통합 방식

a. 모든 모듈이 미리 결합되어 있는 프로그램 전체를 테스트

b. 빅뱅 통합 테스트 방식

c. 오류 발견 및 장애 위치 파악이 어려움

ㄴ.점진적 통합 방식

a. 모듈 단위로 단계적으로 통합하면서 테스트

b. 하향식 / 상향식 / 혼합식 테스트 방식

c. 오류 수정이 용이하고 인터페이스 관련 오류를 완전히 테스트할 수 있음

ㄷ. 하향식 통합 테스트

a. 상위 모듈에서 하위 모듈 방향으로 통합하면서 테스트

b. 깊이 우선 통합법이나 넓이 우선 통합법 사용

c. 상위 모듈에선 테스트 케이스 사용이 어려움

d. 스텁 : 상위 모듈은 있지만 하위 모듈이 없는 경우 하위 모듈 대체

ㄹ. 상향식 통합 테스트

a. 하위 모듈에서 상위 모듈 방향으로 통합하면서 테스트

b. 하나의 주요 제어 모듈과 종속 모듈의 그룹인 클러스터가 필요

c. 드라이버 : 상위 모듈 없이 하위 모듈이 있는 경우 하위 모듈 구동

ㅁ. 혼합식 통합 테스트

- 하위 수준에서는 상향식 통합 상위 수준에서는 하향식 통합을 사용하여 최적의 테스트를 지원
- 샌드위치 통합 테스트

ㅂ. 회귀 테스트

- 이미 테스트된 프로그램의 테스트를 반복
- 통합 테스트로 변경된 모듈이나 컴포넌트에 새로운 오류가 있는지 확인

시스템 테스트

: 개발된 소프트웨어가 원하는 환경에서 수행되는지 테스트

- 실제 환경과 유사하게 만든 테스트 환경에서 진행
- 기능적 요구사항 : 명세서 기반의 블랙박스 테스트
- 비 기능적 요구사항 : 구조적 요소에 대한 화이트박스 테스트

인수 테스트

: 개발한 소프트웨어가 사용자의 요구사항을 충족하는지에 중점을 두고 테스트

- 사용자가 직접 테스트

ㄱ. 사용자 인수 테스트 : 사용자가 시스템 사용의 적절성 여부 확인

ㄴ. 운영상의 인수 테스트 : 시스템 관리자가 시스템 인수 시 수행

ㄷ. 계약 인수 테스트 : 계약상의 조건을 준수하는지 확인

ㄹ. 규정 인수 테스트 : 규정에 맞게 개발되었는지 확인

ㅁ. 알파 테스트 : 개발된 환경에서 사용자가 개발자 앞에서 수행

ㅂ. 베타 테스트 : 사용자의 환경에서 사용자가 직접 테스트 수행

41. 애플리케이션 테스트 프로세스

애플리케이션 테스트 프로세스

: 개발된 소프트웨어가 제대로 만들어 졌는지 테스트하는 절차

- 테스트를 마치면 테스트 계획서, 케이스, 시나리오, 결과서가 산출
- 에러는 빨리 발견될수록 좋음

ㄱ. 테스트 계획 : 프로젝트 계획서 및 요구 명세서를 기반으로 테스트 목표를 정의하고 테스트 대상 및 범위 결정

ㄴ. 테스트 분석 및 디자인 : 테스트의 목적과 원칙을 검토하고 사용자의 요구사항 분석

ㄷ. 테스트 케이스 및 시나리오 작성 : 테스트 케이스를 작성, 검토 및 확인 후 시나리오 작성

ㄹ. 테스트 수행 : 테스트 환경 구축 후 테스트 수행

ㅁ. 테스트 결과 평가 및 리포팅 : 테스트 결과를 분석하여 테스트 결과 작성

ㅂ. 결함 추적 및 관리 : 테스트 수행 후 결함이 어디에서 발생했고 어떤 결함인지 추적하고 관리

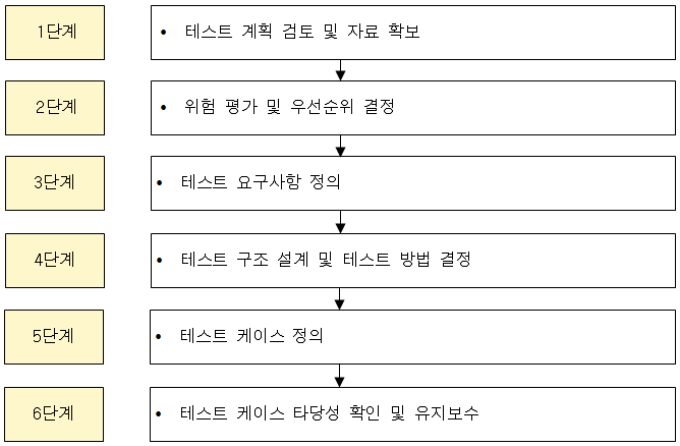
42. 테스트 케이스 / 시나리오 / 오라클

[실기 빈출]

테스트 케이스

: 사용자의 요구사항이 준수되었는지 확인하기 위해 테스트 항목에 대한 명세서

- 명세 기반 테스트의 설계 산출물
- 테스트 케이스 작성 순서



테스트 시나리오

- 테스트 케이스를 적용하는 구체적인 절차를 명세한 문서

테스트 오라클

: 테스트 결과가 올바른지 판단하기 위해 정의된 참 값을 대입하여 비교

테스트 오라클의 특징

ㄱ. 제한된 검증 : 모든 테스트 케이스에는 적용 불가

ㄴ. 수학적 기법 : 수학적 기법을 통해 테스트 오라클 값을 구할 수 있음

ㄷ. 자동화 기능 : 테스트 대상에 대한 실행, 결과 비교 등을 자동화할 수 있음

테스트 오라클의 종류

ㄱ. 참 오라클 : 모든 테스트 케이스의 입력 값에 대해 기대하는 결과를 제공

ㄴ. 샘플링 오라클 : 특정 테스트 케이스의 입력 값에 대해 기대하는 결과를 제공

ㄷ. 추정 오라클 : 특정 테스트 케이스의 입력 값에 대해 기대하는 결과를 제공하고 나머지 값에 대해서는 추정으로 처리

ㄹ. 일관성 검사 오라클 : 변경 시 테스트 케이스 수행 전과 후의 결과 값이 동일한지 확인

43. 테스트 자동화 도구

테스트 자동화

: 반복적인 테스트 절차를 스크립트 형태로 구현하는 자동화 도구를 적용하여 쉽고 효율적으로 테스트 수행

테스트 자동화 도구의 장단점

ㄱ. 장점

a. 반복적인 작업을 자동화해 인력 및 시간 절감

b. 향상된 테스트 품질 보장

c. 사용자의 요구사항 등을 일관성 있게 검증

d. 테스트 결과에 대한 객관적인 평가 기준 제공

e. 테스트 결과를 다양한 표시 형태로 제공

f. UI가 없는 서비스도 정밀 테스트 가능

ㄴ. 단점

a. 사용방법에 대한 교육 및 학습 필요

b. 자동화 도구를 프로세스 단계별로 적용하기 위한 시간, 비용, 노력이 필요

테스트 자동화 수행 시 고려사항

- 모든 과정이 아닌 그때그때 맞는 적절한 도구를 선택
- 자동화 도구를 고려하여 프로젝트 일정 계획
- 프로젝트 초기에 테스트 엔지니어 투입 시기 계획

테스트 자동화 도구의 유형

ㄱ. 정적 분석 도구 : 프로그램을 실행하지 않고 소스코드를 통해 결함을 발견

ㄴ. 테스트 실행 도구 : 스크립트 언어를 사용하여 테스트를 실행

ㄷ. 성능 테스트 도구 : 가상의 사용자를 만들어 테스트를 수행

ㄹ. 테스트 통제 도구 : 테스트 계획 및 관리, 수행, 결함 관리 등을 수행

ㅁ. 테스트 하네스 도구

: 테스트가 실행될 환경을 시뮬레이션하여 컴포넌트 및 모듈이 정상적으로 테스트되도록 함

- 구성요소 : 테스트 드라이버, 테스트 스텁, 테스트 슈트, 테스트 케이스, 테스트 스크립트, 목 오브젝트

테스트 수행 단계별 테스트 자동화 도구

ㄱ. 테스트 계획 단계 : 요구사항 관리 도구

ㄴ. 테스트 분석 및 설계 단계 : 테스트 케이스 생성 도구

ㄷ. 테스트 수행 단계 : 테스트 자동화 / 정적 분석 / 동적 분석 / 성능 테스트 / 모니터링 도구

ㄹ. 테스트 관리 단계 : 커버리지 분석 / 형상 관리 / 결함 추적 및 관리 도구

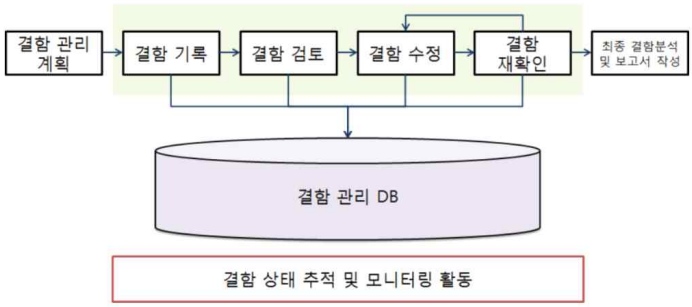
44. 결함 관리

결함

: 소프트웨어가 개발자가 설계한 것과 다르게 동작하거나 다른 결과가 발생하는 것

결함 관리 프로세스

: 애플리케이션 테스트에서 발견된 결함을 처리



결함 상태 추적

: 테스트에서 발견된 결함은 지속적으로 상태 변화를 추적하고 관리해야 함

- 결함 분포 : 특정 속성에 해당하는 결함 수 측정
- 결함 추세 : 시간에 따른 결함 수의 추이 분석
- 결함 에이징 : 결함 상태로 지속되는 시간 측정

결함 추적 순서

: 결함이 발견되고 해결될 때까지의 과정

등록 -> 검토 -> 할당 -> 수정 -> 조치 보류 -> 해제

결함 분류

ㄱ. 시스템 결함 : 주로 애플리케이션이나 데이터베이스 처리에서 발생된 결함

ㄴ. 기능 결함 : 애플리케이션의 기획, 설계, 업무 시나리오 등의 단계에서 유입된 결함

ㄷ. GUI 결함 : 화면 설계에서 발생된 결함

ㄹ. 문서 결함 : 기획자, 사용자, 개발자 간 의사소통 및 기록이 원활하지 않아 발생된 결함

결함 심각도

: 결함이 전체 시스템에 미치는 치명도를 High, Medium, Low로 나눈 것

결함 우선순위

: 발견된 결함 처리에 대한 신속성을 나타내는 척도

- 결함의 중요도와 심각도에 따라 설정되고 수정 여부 결정
- Critical, High, Medium, Low 또는 즉시 해결, 주의 요망, 대기, 개선 권고 등으로 분류

결함 관리 도구

- Mantis : 소프트웨어 설계 시 단위 별 작업 내용을 기록할 수 있어 결함 및 이슈 관리, 추적 도구
- Trac : 결함 추적 및 통합 관리 도구
- Redmine : 프로젝트 관리 및 결함 추적 도구
- Bugzilla : 결함을 지속적으로 관리하고 심각도와 우선순위를 지정할 수 있는 도구

45. 애플리케이션 성능 분석

애플리케이션 성능

: 사용자가 요구한 기능을 최소한의 자원을 사용하여 최대한 많은 기능을 신속하게 처리하는 정도

- 측정 지표 : 처리량, 응답 시간, 경과 시간, 자원 사용률

성능 테스트 도구

: 애플리케이션의 성능을 테스트하기 위해 부하나 스트레스를 가해 성능 측정 지표를 점검하는 도구

- JMeter : 다양한 프로토콜을 지원하는 부하 테스트 도구
- LoadUI : 사용자의 편리성이 강화된 부하 테스트 도구
- OpenSTA : HTTP, HTTPS 프로토콜에 대한 부하 테스트 및 생산품 모니터링 도구

시스템 모니터링 도구

: 애플리케이션 실행 중 시스템 자원의 사용량을 확인하고 분석하는 도구

- 성능 저하의 원인 / 시스템 부하량 / 사용자 분석과 같은 시스템을 안정적으로 운영할 수 있는 기능 제공
- Scouter, Zabbix

애플리케이션 성능 저하 원인 분석

: 애플리케이션을 DB에 연결하기 위해 커넥션 객체를 생성하거나 쿼리를 실행하는 애플리케이션 로직에서 자주 발생

45. 애플리케이션 성능 개선

소스코드 최적화

: 나쁜 코드를 배제하고 클린 코드로 작성

- 클린 코드 작성 원칙 : 가독성, 단순성, 의존성 배제, 중복성 최소화, 추상화

소스 코드 최적화 유형

: 클래스 분할 배치, 느슨한 결합, 코딩 형식 준수, 좋은 이름 사용, 적절한 주석문 사용

<II-V 인터페이스 구현>

46. 모듈 간 공통 기능 및 데이터 인터페이스 확인

모듈 간 공통 기능 및 데이터 인터페이스의 개요

- 공통 기능 : 모듈에 공통적으로 제공되는 기능
- 데이터 인터페이스 : 모듈 간 교환되는 데이터가 저장될 파라미터
- 인터페이스 설계서에서 정의한 모듈의 기능을 기반으로 확인

인터페이스 설계서

: 교환 데이터 및 관련 업무, 송수신 시스템 등에 대한 내용을 정리한 문서

일반적인 인터페이스 설계서

: 인터페이스 목록, 상세 데이터 명세, 기능의 세부 정보를 정의한 문서

ㄱ. 시스템 인터페이스 설계서 : 시스템 인터페이스 목록과 상세 데이터 명세를 정의

ㄴ. 상세 기능별 인터페이스 명세서 : 기능의 세부 인터페이스 정보 정의

ㄷ. 정적/도형 모형을 통한 인터페이스 설계서 : 시스템의 구성요소를 다이어그램으로 표현하여 만든 문서

47. 모듈 연계를 위한 인터페이스 기능 식별

모듈 연계

: 모듈 간 데이터 교환을 위해 관계를 설정

EAI(Enterprise Application Integration) [실기 빈출]

: 기업 내 정보 전달, 연계, 통합 등 상호 연동이 가능하게 해주는 솔루션

ㄱ. Point-to-Point

- 애플리케이션끼리 1:1로 연결
- 변경 및 재사용이 어려움

ㄴ. Hub &Spoke

- 단일 접점인 허브 시스템을 통해 데이터를 전송하는 중앙 집중형 방식
- 확장 및 유지보수 용이
- 허브 장애 시 전체 시스템에 영향

ㄷ. Message Bus(ESB 방식)

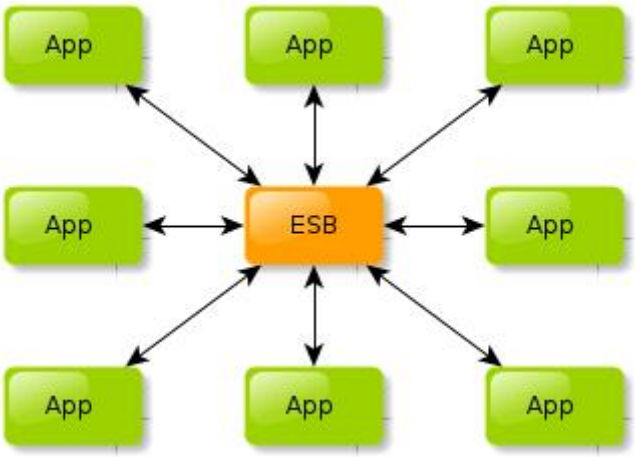
- 애플리케이션 사이 미들웨어를 두어 처리
- 확장성이 뛰어나고 대용량 처리 가능

ㅁ. Hybrid

- Hub &Spoke와 Message Bus의 혼합 방식
- 그룹 내에선 Hub &Spoke 방식을 그룹 간에는 Message Bus 방식 이용
- 데이터 병목 현상 최소화

ESB(Enterprise Service Bus) [실기 빈출]

: 애플리케이션 간 표준 기반 인터페이스를 제공하는 솔루션



- 애플리케이션보다는 서비스 중심의 통합을 지향
- 애플리케이션과의 결합도를 약하게 유지
- 관리 및 보안 유지가 쉽고 높은 수준의 품질 지원

모듈 간 연계 기능 식별

: 모듈 간 공통 기능 및 데이터 인터페이스를 기반으로 모듈과 연계된 기능을 시나리오 형태로 구체화하여 식별

- 인터페이스 기능을 식별하는 데 사용

모듈 간 인터페이스 기능 식별

: 식별된 모듈 간 기능을 검토하여 인터페이스 동작에 필요한 기능을 식별

- 해당 업무에 대한 시나리오를 통해 내부 모듈과 관련된 인터페이스 기능 식별
- 외부 및 인터페이스 모듈 간 동작하는 기능을 통해 인터페이스 기능 식별

48. 모듈 간 인터페이스 데이터 표준 확인

인터페이스 데이터 표준

: 모듈 간 인터페이스에 사용되는 데이터의 형식을 표준화

데이터 인터페이스 확인

: 데이터 표준을 위해 식별된 데이터 인터페이스에서 입출력 값의 의미와 데이터의 특성 등을 구체적으로 확인

인터페이스 기능 확인

: 데이터 표준을 위해 식별된 인터페이스 기능을 기반으로 인터페이스 기능 구현을 위해 필요한 데이터 항목 확인

인터페이스 데이터 표준 확인

: 데이터 인터페이스에서 확인된 데이터 표준과 인터페이스 기능을 통해 확인된 데이터 항목을 검토하여 최종적으로 데이터 표준 확인

49. 인터페이스 기능 구현 정의

인터페이스 기능 구현

: 인터페이스를 실제로 구현하기 위해 인터페이스 기능에 대한 구현 방법을 기능별로 기술

인터페이스 기능 구현 정의 순서

ㄱ. 컴포넌트 명세서 확인

ㄴ. 인터페이스 명세서 확인

ㄷ. 일관된 인터페이스 기능 구현 정의

ㄹ. 정의된 인터페이스 기능 구현을 정형화

모듈 세부 설계서

: 모듈의 구성 요소와 세부적인 동작 등을 정의한 설계서

ㄱ. 컴포넌트 명세서 : 컴포넌트의 개요 및 내부 클래스의 동작, 인터페이스를 통해 외부와 통신하는 명세 등을 정의

ㄴ. 인터페이스 명세서 : 컴포넌트 명세서의 항목 중 인터페이스 클래스의 세부 조건 및 기능 등을 정의

모듈 세부 설계서 확인

: 모듈의 컴포넌트 명세서와 인터페이스 명세서를 기반으로 인터페이스에 필요한 기능 확인

인터페이스 기능 구현 정의

: 인터페이스의 기능, 데이터 표준, 모듈 세부 설계서를 기반으로 일관성 있고 정형화된 인터페이스 기능 구현에 대해 정의

50. 인터페이스 구현

인터페이스 구현

: 송수신 시스템 간의 데이터 교환 및 처리를 실현해주는 작업

- 정의된 인터페이스 기능 구현을 기반으로 인터페이스 구현 방법을 분석하고 분석한 인터페이스 구현 정의를 기반으로 구현

데이터 통신을 이용한 인터페이스 구현

: 애플리케이션 영역에서 인터페이스 형식에 맞춘 데이터 포맷을 인터페이스 대상으로 전송하고 이를 수신하는 측에서 파싱 하여 해석하는 방식

- JSON, XML 형식 사용

인터페이스 엔티티를 이용한 인터페이스 구현

: 인터페이스가 필요한 시스템 사이에 별도의 인터페이스 엔티티를 두어 상호 연계하는 방식

- 인터페이스 테이블 활용

51. 인터페이스 예외 처리

인터페이스 예외 처리

: 구현된 인터페이스가 동작하는 과정에서 기능상 예외 상황이 발생했을 때 처리하는 절차

데이터 통신을 이용한 인터페이스 예외 처리

: 인터페이스 객체를 이용해 구현한 인터페이스 동작이 실패할 경우를 대비

- 송수신 시 발생할 수 있는 예외 케이스를 정의하고 예외 처리 방법을 기술

- 시스템 환경, 송수신 데이터, 프로그램 자체 원인 등의 원인으로 예외 상황 발생

인터페이스 엔티티를 이용한 인터페이스 예외 처리

: 엔티티에 인터페이스의 실패 상황과 원인을 기록

- 조치를 취할 수 있도록 사용자와 관리자에게 알려주는 방식으로 예외 처리

52. 인터페이스 보안

인터페이스 보안

: 충분한 보안을 갖추지 않으면 시스템 전체에 악영향을 주는 취약점이 될 수 있음

인터페이스 보안 취약점 분석

: 인터페이스 기능이 수행되는 각 구간들의 구현 현황을 확인 후 어떤 취약점이 있는지 확인

- 송수신 영역의 구현 기술 및 특징을 구체적으로 확인

- 확인된 인터페이스 기능을 기반으로 영역별로 발생할 수 있는 취약점을 시나리오 형태로 작성

인터페이스 보안 기능 적용

: 분석한 인터페이스 기능과 취약점을 기반으로 보안 기능 적용

ㄱ. 네트워크 영역 : 송수신간 스니핑 등을 이용한 데이터 탈취 및 변조 위험을 방지하기 위해 네트워크 트래픽에 대한 암호화 설정

ㄴ. 애플리케이션 영역 : 소프트웨어 개발 보안 가이드를 참조하여 코드 상의 취약점을 보완

ㄷ. 데이터베이스 영역 : 접근 권한과 데이터베이스 동작 객체의 취약점에 보안 기능 적용

53. 연계 테스트

연계 테스트

: 구축된 연계 시스템과 구성 요소가 정상적으로 동작하는지 확인

연계 테스트 케이스 작성

: 연계 시스템 간의 데이터 및 프로세스 흐름을 분석하여 필요한 테스트 항목을 도출

- 송수신 연계 응용 프로그램의 단위 테스트 케이스와 연계 테스트 케이스를 각각 작성

연계 테스트 환경 구축

: 테스트의 환경을 송수신 기관과의 협의를 통해 결정하고 구축

연계 테스트 수행

: 연계 응용 프로그램을 실행하여 연계 테스트 케이스의 시험 항목 및 처리 절차 등을 실제로 진행

연계 테스트 수행 결과 검증

: 연계 테스트 케이스의 시험 항목 및 처리 절차를 수행한 결과가 예상 결과와 동일한지 확인

- 연계 서버에서 적용하는 모니터링 현황 확인

- 시스템에서 기록하는 로그 확인

- 테이블 또는 파일을 열어 데이터를 확인

54. 인터페이스 구현 검증

인터페이스 구현 검증

: 인터페이스가 정상적으로 잘 작동하는지 확인하는 것

인터페이스 구현 검증 도구 [실기 빈출]

ㄱ. xUnit : Java, C++, .Net 등 다양한 언어를 지원

ㄴ. STAF : 서비스 호출 및 컴포넌트 재사용 등 다양한 환경을 지원

ㄷ. FitNesse : 웹 기반 테스트케이스 설계, 진행, 결과 확인 등을 지원

ㄹ. NTAF : FitNess의 협업 기능과 STAF의 재사용 및 확장성을 통합한 NHN의 프레임워크

ㅁ. Selenium : 다양한 브라우저 및 개발 언어 지원

ㅂ. watir : Ruby를 사용

인터페이스 구현 감시 도구

- APM(Application Performance Management)을 사용하여 감시 가능 : 애플리케이션 성능 관리 서비스
- 애플리케이션 성능 관리 도구를 통해 데이터베이스와 웹 애플리케이션의 다양한 정보를 조회하고 분석할 수 있음
- 스카우터(Scouter), 제니퍼(Jennifer) 등

인터페이스 구현 검증 도구 및 감시 도구 선택

- 인터페이스 명세서의 세부 기능을 참조하여 검증 도구와 감시 도구의 요건을 분석
- 분석 후 시장 및 솔루션 조사를 통해 적절한 도구 선택

인터페이스 구현 검증 확인

- 외부 시스템과 연계 모듈 동작 상태 확인
- 예상되는 결과값과 실제 검증 값이 동일한지 비교

인터페이스 구현 감시 확인

- 외부 시스템과 연결 모듈이 서비스를 제공하는 동안 정상적으로 동작하는지 확인

55. 인터페이스 오류 확인 및 처리 보고서 작성

인터페이스 오류 확인 및 처리 보고서

: 인터페이스 오류 발생 시 오류사항을 확인하고 오류 처리 보고서를 작성하여 관리 조직에 보고

인터페이스 오류 발생 즉시 확인

- 화면에 오류 메시지를 표시하고 자동으로 SMS나 이메일을 발생하는 것으로 즉시 오류 발생 확인

주기적인 인터페이스 오류 발생 확인

- 시스템 로그나 인터페이스 오류 관련 테이블 등을 통해 주기적으로 오류 발생 여부 확인

인터페이스 오류 처리 보고서 작성

: 인터페이스 작동 시 발생하는 오류의 발생 및 종료 시점, 원인 및 증상, 처리사항 등을 정리한 문서

- 보고시기 최초 발생 시, 오류 처리 경과 시, 완료 시로 나누어 작성

III. 데이터베이스 구축 (대단원3/5)

<III-I 논리 데이터베이스 설계>

초심자를 위한 Tip [필기 빈출],[실기 빈출]

3단원 DB파트를 대해야 하는 자세

데이터베이스 파트는 정말 프로그래머에게 있어서 떼려야 뗄 수 없는 과목이다. 사실 DB(데이터베이스)에 관해서는 이론으로 보는 것 보다는 한 번 실제로 만지고 다뤄 보는 것이 가장 빠르고 직관적으로 배울 수 있는 방법이다. 다만 우리는 지금 정보처리기사(산업기사)를 준비하는 중이므로 지금 이 필기자료 정리 모음에서는 필기 합격을 위해서의 조언을 할 것이다.

필기 합격을 위해서는 지금 여기 나오는 개념에 대해서 이해하고 기출문제를 풀어야 한다.(다시 말하지만 사실상 필기시험에서 가장 중요한 것은 기출문제를 푸는 것이다) 다만 시간이 정말 부족하고 빠르게 '필기합격'만을 목표로 하는 사람이라면 이해보다는 암기 위주로 개념을 한 번 훑고 기출 문제를 풀면 된다. 하지만 실기까지 깔끔하게 준비 하고 싶고, 시간을 두 배로 쓰고 싶은 것이 아니라면 최대한 이해를 하며 개념을 적립하고 기출을 풀 때 이해를 기반으로 푸는 것을 추천한다. 왜 이렇게 강조하고 겁을 주느냐? 후에 실기에서는 DB파트 문제를 풀려면 이해가 기반이 되어 있어야 훨씬 수월하고 DB가 뭔지 확실히 이해 할 수 있어서이다(그리고 이 파트를 제대로 공부해 자신의 것으로 만들어 놓는다면 현업에서 어마어마하게 도움이 될 것이다)

56. 데이터베이스 설계

데이터베이스 설계

: 사용자의 요구를 분석하여 그것에 맞게 설계하고 특정 DBMS로 데이터베이스를 구현하여 사용자들이 사용하는 것

(DBMS : database management system/ 데이터 베이스 관리 시스템)

데이터베이스 설계 순서

: 요구 조건 분석 → 개념적 설계 → 논리적 설계 → 물리적 설계 → 구현

ㄱ. 개념적 설계 : 개념 스키마, E-R 모델, 트랜잭션 모델링

ㄴ. 논리적 설계 : 논리 스키마 설계 / 트랜잭션 인터페이스 설계 / 관계형 DB - Table, 계층형 DB - Tree, 망형 DB - Graph

ㄷ. 물리적 설계 : 컴퓨터에 저장

데이터베이스 설계 시 고려사항

- 무결성, 일관성, 회복, 보안, 효율성, 데이터베이스 확장

(무결성 : 컴퓨팅 분야에서 완전한 수명 주기를 거치며 데이터의 정확성과 일관성을 유지하고 보증하는 것)

56. 데이터 모델의 개념

데이터 모델

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

데이터 모델의 구성 요소 [필기 빈출],[실기 빈출]

- 개체(Entity), 속성(Attribute), 관계(Relation)

데이터 모델의 종류

- 개념적 데이터 모델, 논리적 데이터 모델, 물리적 데이터 모델

데이터 모델에 표시할 요소

ㄱ. 구조(Structure) : 개체 타입들 간의 관계, 데이터 구조 및 정적 성질 표현

ㄴ. 연산(Operation) : 저장된 데이터를 처리하는 작업에 대한 명세, DB를 조작하는 기본 도구

ㄷ. 제약 조건(Constraint) : 데이터의 논리적인 제약 조건

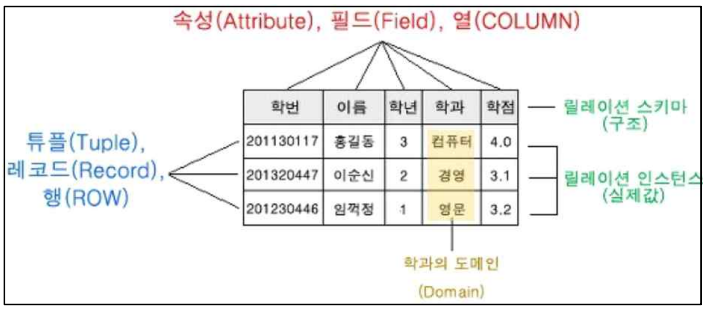
57. 데이터 모델의 구성 요소

개체 [필기 빈출],[실기 빈출]

: 데이터베이스의 표현하려는 정보

- 유형, 무형의 정보로서 서로 연관된 몇 개의 속성으로 이루어짐
- 유일한 식별자에 의해 식별이 가능
- 개체(튜플)의 수를 카디널리티라고 함
- 개체 인스턴스 : 개체를 구성하고 있는 속성들이 값을 가져 하나의 개체를 나타내는 것

릴레이션 스키마와 인스턴스 [필기 빈출],[실기 빈출]



ㄱ. 릴레이션

: 주로 테이블(Table)과 같은 의미로 사용되며, 데이터의 집합을 의미한다. 튜플(Tuple)과 어트리뷰트(Attribute)로 구성됨

ㄴ. 스키마

: 관계 데이터베이스의 릴레이션이 어떻게 구성되는지 어떤 정보를 담고 있는지에 대한 기본적인 구조를 정의

- 테이블에서 스키마는 첫 행인 헤더에 나타나며 데이터의 특징인 속성, 자료 타입 등의 정보를 담고 있음

ㄷ. 인스턴스

: 정의된 스키마에 따라 테이블에 실제로 저장되는 데이터의 집합

ㄹ. 튜플(=레코드(Record)=행(Row))

: 릴레이션을 구성하는 각각의 행

- 튜플의 수 = 카디널리티 = 기수

ㅁ. 도메인(Domain)

: 하나의 속성(Attribute)이 취할 수 있는 같은 타입의 원자 값들의 집합

ㅂ. 속성(Attribute)

: 릴레이션을 구성하는 각각의 열(DB의 가장 작은 논리 단위)

- 데이터 베이스를 구성하는 가장 작은 논리적 단위
- 파일 구조상 데이터 항목 또는 데이터 필드에 해당
- 개체의 특성을 기술함
- 속성의 수 = 디그리(Degree) = 차수

속성의 분류

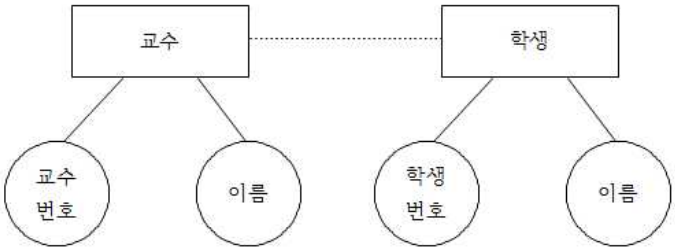
ㄱ. 기본키 속성(Primary Key Attribute) : 개체를 식별할 수 있는 속성 ex) 학번

ㄴ. 외래키 속성(Foreign Key Attribute) : 다른 개체와의 관계에서 포함된 속성

ㄷ. 일반 속성 : 개체에 포함되어 있지만 기본키, 외래키가 아닌 속성

관계 [필기 빈출],[실기 빈출]

: 개체 간의 논리적인 연결



[개체 간 관계는 점선, 속성 간 관계는 실선으로 표현]

관계의 형태

- 1:1, 1:N, N:M 3가지 관계가 있음

식별 / 비식별 관계

- 식별 관계 : A,B 개체 간의 관계에서 A 개체의 기본키가 B 개체의 외래키면서 동시에 기본인 것
- 비식별 관계 : A,B 개체 간의 관계에서 A 개체의 기본키가 B 개체의 외래키이지만 기본키는 아닌 것
- 한 개체의 기본키를 다른 개체가 기본키로 사용하면 식별, 아니면 비식별

57. 식별자

식별자

: 하나의 개체 내에서 각각의 인스턴스를 유일하게 구분 지을 수 있는 것

식별자의 분류

학번	이름	학년	학과	학점
201130117	홍길동	3	컴퓨터	4.0
201320447	이순신	2	경영	3.1
201230446	임꺽정	1	영문	3.2

ㄱ. 대표성 여부 : 개체를 유일하게 식별할 수 있음

a. 주 식별자

- 개체를 대표하는 유일한 식별자 ex) 학번
- 주 식별자의 특징 : 유일성, 최소성, 불변성, 존재성

b.보조 식별자 : 주 식별자를 대신하여 개체를 식별할 수 있는 것 ex) 이름 or 학과 or 학년 등 유일하지 않은 것

ㄴ. 스스로 생성 여부

a. 내부 식별자 : 개체 내에서 스스로 만들어지는 식별자

b. 외부 식별자 : 다른 개체와의 관계에서 만들어지는 식별자

ㄷ. 단일 속성 여부

a. 단일 식별자 : 주 식별자가 한 가지 속성으로 구성된 식별자

b. 복합 식별자 : 주 식별자가 두 개 이상의 속성으로 구성된 식별자

ㄹ. 대체 여부

a. 원조 식별자(=본질 식별자) : 업무에 의해 만들어지는 가공되지 않은 원래의 식별자

b. 대리 식별자 : 주 식별자의 속성이 두 개 이상인 경우 속성들을 하나의 속성으로 묶어 사용하는 식별자

58. E-R(개체-관계) 모델 [필기 빈출],[실기 빈출]

E-R 모델 (Entitiy, Attribute, Relationship)

: E-R 모델은 개념적 데이터 모델의 가장 대표적인 모델

(개념적 데이터 모델 : 개념 스키마, E-R모델, 트랜잭션 모델링)

-피터첸에 의해 제안되어 기본적인 구성 요소 성립

- 데이터를 개체, 관계, 속성으로 묘사

E-R 다이어그램

: E-R 모델의 기본 아이디어를 쉽게 기호를 사용하여 시각적으로 표현한 것

- 표기법에는 피터 첸 표기법, 정보 공학 표기법 등이 있음

ㄱ. 피터 첸 표기법

a. 사각형 : 개체 타입

b. 마름모 : 관계 타입

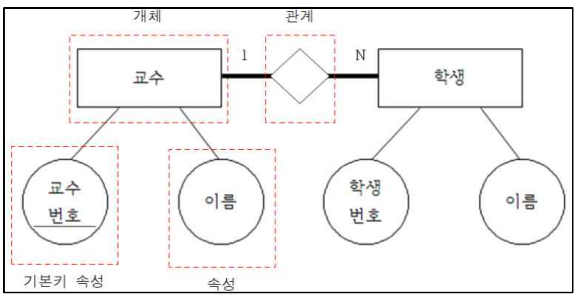
c. 타원 : 속성 타입

d. 이중 타원 : 복합 속성

e. 밑줄 타원 : 기본키 속성

f. 복수 타원 : 복합 속성

g. 관계 : 1:1, 1:N, N:M 같은 관계에 대한 대응 수



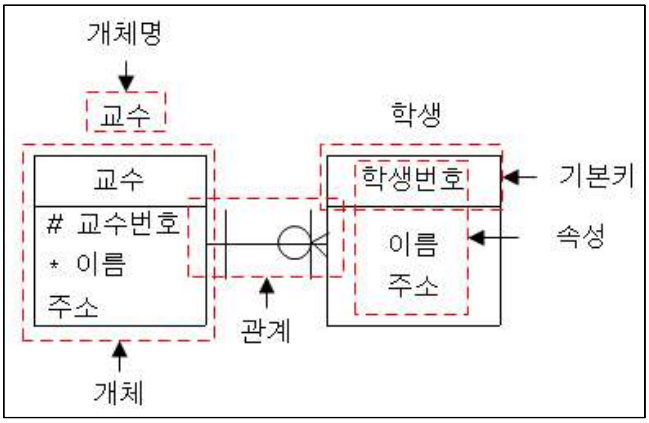
ㄴ. 정보 공학 표기법

- 관계 표시 기호

a. | : 1

b. O : 0

c. <: N



초심자를 위한 Tip

정보 공학 표기법은 실무가서 DB설계에서도 정말 많이 쓰임 꼭 숙지하도록 할 것

59. 관계형 데이터베이스의 구조 [필기 빈출],[실기 빈출]

관계형 데이터베이스

: 가장 많이 사용되는 데이터베이스로서 흔히 행(Row)과 열(Column)로 표현되는 테이블간의 관계를 나타낼 때 사용되며, 이렇게 표현된 데이터베이스는 SQL을 통하여 관리 및 접근한다.

(57번의 데이터 모델의 구성 요소 참조)

RDB 종류

- Oracle, MySQL(Oracle) / MS-SQL(Microsoft) / DB2, Infomix(IBM) / Maria DB(Sun Microsystems) / Derby(Apache) / SQLite(Opensource)

초심자를 위한 Tip

관계형 데이터베이스 말고 DB의 종류에 NoSQL이란 것도 있다. 현재로서는 NoSQL보다는 RDB가 더 많이 쓰인다고 할 수 있으나 NoSQL은 대용량 데이터를 다루거나 데이터 분산 처리에 용이하여 Cloud Computing에 많이 사용한다.

그 외에도 계층형 데이터베이스나 네트워크형 데이터베이스도 존재하나 현 시점에는 RDB와 NoSQL을 가장 많이 사용한다고 생각하면 되겠다. 필기에서보다는 실기에서 많이 다룬다.

참고자료

-> https://honeyteacs.tistory.com/19

60. 관계형 데이터베이스의 제약 조건

[필기 빈출],[실기 빈출]

Key의 개념

: key는 데이터베이스에서 조건에 맞는 튜플을 찾거나 정렬할 때 튜플을 서로 구분할 수 있는 기준이 되는 속성

Key의 종류

ㄱ. 후보키

: 기본키로 사용할 수 있는 속성

- 유일성과 최소성의 성질을 만족

ㄴ. 기본키

: 후보 키 중에서 선정된 Main Key로 중복된 값을 가질 수 없음

- 후보 키의 부분집합
- NULL 값을 가질 수 없음(=개체 무결성)

(NULL 값 : 정보의 부재를 나타내기 위해 사용하는 값. 0이 아님 NULL 값은 말 그대로 NULL이라는 값!)

ㄷ. 대체키

: 후보 키가 둘 이상일 때 기본키를 제외한 나머지 후보키

ㄹ. 슈퍼키

한 가지 속성일 땐 Key가 될 수 없지만 여러 속성이 뭉쳐서 Key의 속성을 가짐

-유일성의 성질을 만족

ㅁ. 외래키

: 다른 릴레이션의 기본 키를 참조한 것

- 외래 키의 값은 참조한 릴레이션의 기본키 값과 동일해야 함(=참조 무결성)

참고자료 ->https://limkydev.tistory.com/108

61. 정규화 [필기 빈출],[실기 빈출]

정규화

: 관계형 데이터베이스에서 정확성을 더욱 유지하기 위해 스키마를 쪼개는 과정

- 데이터베이스의 논리적 설계 단계에서 수행

정규화의 목적

- 데이터 구조의 안정성 및 무결성을 유지
- 이상의 발생을 방지 및 자료 저장 공간의 최소화

이상(Anomaly)

- 사용자의 의도와는 상관없이 데이터가 삽입, 삭제, 갱신되는 현상

정규화 과정

: 1NF -> 2NF -> 3NF -> BCNF -> 4NF -> 5NF

(밑의 요약정리만 보면 이해가 안 갈수도 있으니 참고해주는 블로그의 정리 글을 꼭 보고 이해하고 넘어갈 것)

ㄱ. 1NF(제1 정규형)

: 릴레이션에 속한 모든 값들이 원자 값으로만 구성

ㄴ. 2NF(제2 정규형)

: 기본키가 아닌 모든 속성이 기본키에 대하여 완전 함수적 종속을 만족

a. 완전 함수적 종속 : 기본키에 의해서 속성이 결정

b. 부분 함수적 종속 : 기본키의 일부에 의해 속성이 결정

- 아래와 같이 학번과 과목 코드가 기본키인 릴레이션이 있을 때 과목 점수는 기본키(학번, 과목 코드)를 가지고 알 수 있음 이를 '완전 함수적 종속' 이라고 함, 이름은 기본키의 일부를 가지고 알 수 있음 이를 '부분 함수적 종속' 이라고 함

학번	이름	과목코드	과목점수
2015xxxx	박xx	A01	95
2017xxxx	김xx	A02	80

ㄷ. 3NF(제3 정규형)

: 기본키가 아닌 모든 속성이 기본키에 대해 이행적 종속을 만족하지 않음

- 이행적 종속 : A → B, B → C 일 때 A → C를 만족하는 관계

- BCNF(Boyce-Codd정규형) : 결정자가 모두 후보키

ㄹ. 4NF(제4 정규형)

: 릴레이션에 다치 종속이 성립하는 경우 모든 속성이 함수적 종속 관계를 만족

ㅁ. 5NF(제5 정규형)

: 모든 조인 종속이 후보키를 통해서만 성립

정규화 과정

(정규화는 실기에서도 자주 다루니 꼭 이해를 할 것)

참고 블로그

https://yaboong.github.io/database/2018/03/09/database-normalization-1/

62. 반정규화 [필기 빈출],[실기 빈출]

반정규화

: 정규화된 데이터를 다시 통합, 중복, 분리하는 과정으로 의도적으로 정규화 원칙을 위해

- 과도한 정규화로 성능이 떨어졌을 때 실행

반정규화의 종류

ㄱ. 테이블 통합

: 하나의 테이블로 합쳐 사용하는 것이 성능 향상에 도움이 될 경우 수행

- Not NULL, Default, Check 등의 제약조건을 설계하기 어려움

ㄴ. 테이블 분할

: 테이블을 수평 또는 수직으로 분할

(수직분할은 컬럼을 기준으로 수평분할은 Row를 기준으로 분리)

ㄷ. 중복 테이블 추가

: 여러 테이블에서 데이터를 추출해서 사용해야 하거나 다른 서버에 저장된 테이블을 이용해야 하는 경우 수행

ㄹ. 중복 속성 추가

: 조인해서 데이터를 처리할 때 데이터를 조회하는 경로를 단축하기 위해 자주 사용하는 속성을 하나 더 추가

63. 시스템 카탈로그 [실기 빈출]

시스템 카탈로그

: 시스템 그 자체에 관련이 있는 다양한 객체에 관한 정보를 포함하는 시스템 데이터베이스

- 데이터 사전(Data Dictionary)이라고도 함

시스템 카탈로그 저장 정보

: 시스템 카탈로그에 저장되는 정보를 메타 데이터라고 함

- 메타 데이터 : 데이터에 대한 설명. 메타 데이터가 모이면 데이터 사전이 됨

메타 데이터의 유형

ㄱ. 데이터 베이스 객체 정보 : Table, Index, View 등의 구조 및 통계 정보

ㄴ. 사용자 정보

ㄷ. 테이블의 무결성 제약 조건 정보

ㄹ. 함수, 프로시저, 트리거 등에 대한 정보

시스템 카탈로그의 특징

- 시스템 테이블로 구성되어 있어 SQL문으로 검색해 볼 수 있음
- DML(INSERT, DELETE, UPDATE) 문으로 갱신이 불가능

<Ⅲ-Ⅱ물리 데이터베이스 설계>

64. 사전 조사 분석

물리 데이터베이스 설계

: 논리적 구조로 표현된 논리적 데이터베이스를 디스크 등의 물리적 저장장치에 저장할 수 있는 물리적 구조의 데이터로 변화하는 과정

데이터베이스 설계 순서

요구 조건 분석 → 개념적 설계 → 논리적 설계 → 물리적 설계 → 구현

물리적 설계 단계에서 수행해야 할 것

ㄱ. 저장 레코드의 양식 설계

ㄴ. 레코드 집중의 분석 및 설계

ㄷ. 접근 경로 설계

- 물리적 데이터베이스 구조는 여러 가지 타입의 저장 레코드 집합이라는 면에서 단순한 파일과 다름

물리적 설계 옵션

ㄱ. 반응시간 : 트랜잭션 수행을 요구한 시점부터 처리 결과를 얻을 때까지의 경과 시간

ㄴ. 공간 활용도 : 데이터베이스 파일과 액세스 경로 구조에 의해 사용되는 저장 공간의 양

ㄷ. 트랜잭션 처리량 : 단위시간 동안 데이터베이스 시스템에 의해 처리될 수 있는 트랜잭션의 평균 개수

데이터 명명 규칙 파악

- 물리 데이터 모델에 적용해야하는 규칙
- 데이터 표준화 및 논리 데이터베이스 설계의 결과물들을 통해 파악
- 물리-논리 데이터베이스 설계 적용되는 명명규칙은 일관성 유지
- 명명 규칙을 파악하기 위해선 도메인과 데이터 사전에 대한 지식 필요

시스템 자원 파악

: 데이터베이스 설치에 영향을 미칠 수 있는 물리적인 요소

하드웨어 자원, 운영체제 및 DBMS의 버전, DMBS 파라미터 정보 등으로 구분

데이터베이스 관리 요소 파악

: 데이터베이스 운영과 관련된 관리요소를 파악

- 데이터베이스 관리 요소를 파악한 후 이를 기반으로 시스템 조사 분석서를 작성
- 시스템 조사 분석서를 기반으로 다음과 같은 요소들을 파악

ㄱ. 데이터베이스 구조 : 데이터베이스 구조에 따라 문제 발생 시 대응 방법이 다름

ㄴ. 이중화 구성 : 문제 발생에 대비하여 동일한 데이터베이스를 복제하여 관리

ㄷ. 분산 데이터베이스 : 물리적인 피해에 데이터 유실을 최소화할 수 있고 장애로 인한 데이터 유실 복구에 효과적

ㄹ. 접근&제어 통제 : 접근 가능한 사용자의 권한 남용으로 인한 정보 유출 및 변조가 빈번하게 발생

ㅁ. DB암호화 : 데이터 암호화, 암호 키에 대한 인증 등을 통해 데이터 유출 시 데이터의 복호화를 어렵게 함

65. 데이터베이스 저장 공간 설계

테이블 [필기 빈출],[실기 빈출]

: 데이터베이스의 가장 기본적인 객체로써 행(Row)과 열(Column)으로 구성됨

- 논리 설계 단계의 개체(Entity)에 대응하는 객체
- 종류에는 일반 테이블, 클러스터 인덱스 테이블, 파티셔닝 테이블, 외부 테이블, 임시 테이블 등이 있음

일반 테이블

- 현재 사용되는 대부분의 DBMS(DataBase Management System)에서 표준 테이블로 사용되는 테이블

군번	이름	병과	계급
2014xxxx	윤xx	공병	병장
2018xxxx	김xx	보병	상병
2013xxxx	박xx	포병	일병
2012xxxx	이xx	운전병	이병

ex) 군인 테이블

클러스터드 인덱스 테이블

: 기본키나 인덱스키의 순서에 따라 데이터가 저장되는 테이블

- 일반적인 인덱스를 사용하는 테이블에 비해 접근 경로가 단축

군번	이름	병과	계급
2012xxxx	윤xx	공병	병장
2013xxxx	김xx	보병	상병
2014xxxx	박xx	포병	일병
2018xxxx	이xx	운전병	이병

ex) 기본키인 군번을 기준으로 정렬

파티셔닝 테이블

: 대용량의 테이블을 작은 논리적인 단위인 파티션으로 나눈 테이블

- 대용량의 데이터를 효과적으로 관리할 수 있지만 파티션 키를 잘못 구성하면 성능 저하 등 역효과를 초래할 수 있음

- 방식에 따라 범위 분할, 해시 분할, 조합 분할 등으로 나뉨

ㄱ. 범위 분할 : 지정한 열의 값을 기준으로 분할

ㄴ. 해시 분할 : 해시 함수를 적용한 결과 값에 따라 데이터를 분할

ㄷ. 조합 분할 : 범위 분할로 분할한 다음 해시 함수를 적용하여 다시 분할

외부 테이블

: 데이터베이스에서 일반 테이블처럼 이용할 수 있는 외부 파일로 데이터베이스 내에 객체로 존재

- 데이터 쉐어하우스, ETL(Extraction Transformation Loading, 추출 변환 적재) 등의 작업에 사용

임시 테이블

: 트랜잭션이나 세션별로 데이터를 저장하고 처리할 수 있는 테이블

- 절차적인 처리를 위해 임시로 사용

- 저장된 데이터는 트랜잭션이 종료되면 삭제

컬럼 [필기 빈출],[실기 빈출]

: 테이블의 열을 구성하는 요소로 데이터타입과 길이 등으로 정의

- 데이터타입은 데이터의 일관성 유지를 위해 사용되는 가장 기본적인 것

- 도메인을 정의한 경우 도메인에 따라 데이터의 타입과 길이가 정의

- 두 칼럼을 비교하는 연산에서 두 칼럼의 데이터 타입이나 길이가 다르면 DBMS 내부적으로 데이터 타입 변화 후 비교 연산 수행

- 참조 관계인 칼럼들은 데이터 타입과 길이가 일치해야 함

테이블스페이스

: 테이블이 저장되는 논리적인 영역으로 하나의 테이블스페이스에 하나 또는 그 이상의 테이블을 저장할 수 있음

- 테이블을 저장하면 논리적으론 테이블스페이스에 물리적으론 연관된 데이터 파일에 저장됨

- 테이블스페이스를 테이블, 테이블스페이스, 데이터 파일로 나눠 관리하면 논리적 구성이 물리적 구성에 종속되지 않아 투명성이 보장됨

투명성 : 사실의 존재 여부를 염두에 두지 않아도 되는 성질

- 데이터베이스에 저장되는 내용에 따라 테이블, 인덱스, 임시 등의 용도로 구분하여 설계

66. 트랜잭션 및 CRUD 분석

트랜잭션 [필기 빈출],[실기 빈출]

: 데이터베이스의 상태를 변환시키는 하나의 논리적인 기능을 수행하기 위한 작업 단위 또는 한꺼번에 수행되어야 할 일련의 연산들을 의미

- 데이터베이스 시스템에서 병행 제어 및 회복 작업 시 처리되는 작업의 논리적인 단위

- 사용자가 시스템에 대한 서비스 요구 시 시스템이 응답하기 위한 상태 변환 과정의 작업 단위

트랜잭션의 특성 [필기 빈출],[실기 빈출]

ㄱ. 원자성(Atomicity)

a. 데이터베이스에 반영되도록 완전히 완료 아니면 전혀 반영되지 않도록 복구되어야 함

b. 어느 하나라도 오류가 발생하면 트랜잭션 전부가 취소되어야 함

ㄴ. 일관성(Consistency)

a. 문법을 일괄적으로 맞춰야 함

b. 트랜잭션을 성공적으로 완료하면 언제나 일관성 있는 데이터베이스 상태로 변환

c. 시스템이 가지고 있는 고정 요소는 트랜잭션 수행 전과 후가 상태가 같아야 함

ㄷ. 독립성(Isolation)

a. 하나의 트랜잭션 연산중에는 다른 트랜잭션이 관여하면 안 됨

b. 수행 중인 트랜잭션은 완전히 완료될 때까지 다른 트랜잭션에서 수행 결과를 참조할 수 없음

ㄹ. 지속성(Durability)

a. 성공적으로 완료된 트랜잭션의 결과는 영구적으로 유지, 반영되어야 함

CRUD 분석 [필기 빈출],[실기 빈출]

- 데이터베이스의 테이블에 변화를 주는 트랜잭션 연산 중 생성(Create), 읽기(Read), 갱신(Update), 삭제(Delete)의 연산에 대해 CRUD 매트릭스를 작성하여 분석하는 것

- 테이블에 발생되는 트랜잭션의 주기별 발생 횟수를 파악하고 연관된 테이블들을 분석하면 테이블에 저장되는 데이터의 양을 유추할 수 있음

CRUD 매트릭스

: 2차원 표로 행에는 프로세스 열에는 테이블을 행과 열이 만나는 위치에는 프로세스가 테이블에 발생시키는 변화를 표시하는 업무 프로세스와 데이터 간 상관 분석표

- CRUD 매트릭스를 통해 프로세스의 트랜잭션이 테이블에 수행하는 작업을 검증

- CRUD 매트릭스의 각 셀에는 C, R, U, D가 들어가고 복수의 작업 시에는 우선순위는 C >D >U >R를 적용

- CRUD 매트릭스가 완성되면 C, R, U, D 중 어느 것도 적히지 않는 행이나 열, C나 R이 없는 행을 확인하여 불필요하거나 누락된 테이블 또는 프로세스를 찾음

프로세스 / 테이블	회원	상품	주문
상품 등록		C	
주문 요청	R	R	C
주문 변경			R
주문 취소			R, D

[온라인 쇼핑몰 CRUD 매트릭스]

트랜잭션 분석

: CRUD 매트릭스를 기반으로 테이블에 발생하는 트랜잭션 양을 분석하고 테이블에 저장되는 데이터의 양을 유추하고 이를 근거로 DB용량을 산정, DB 구조를 최적화하는 것

트랜잭션 분석서

- 단위 프로세스와 CRUD 매트릭스를 이용하여 작성

- 구성 요소에는 단위 프로세스, CRUD 연산, 테이블 명, 칼럼명, 테이블 참조 횟수, 트랜잭션 수, 발생 주기 등

(한마디로 CRUD 매트릭스로 트랜잭션이 얼마나 발생하는지 확인하고 이를 근거로 DB용량을 산정하는 것)

67. 인덱스 설계 [필기 빈출],[실기 빈출]

인덱스

: 데이터 레코드를 빠르게 접근하기 위한 키값, 포인터 쌍으로 구성되는 데이터 구조

(쉽게 예를 들자면 책의 목차 혹은 중요한 지점에 붙여놓는 포스트잇을 생각하면 됨)

- 데이터가 저장된 물리적 구조와 밀접한 관계가 있음

- 파일의 레코드에 대한 액세스를 빠르게 수행할 수 있음

- 인덱스가 없으면 특정한 값을 찾기 위해 모든 데이터 페이지를 확인하는 TABLE SCAN이 발생

TABLE SCAN

: 데이터가 나올 때까지 모든 레코드를 순차적으로 읽는 것

- 레코드의 삽입과 삭제가 수시로 일어나는 경우에는 인덱스의 개수를 최소로 하는 것이 효율적

- 클러스터드 인덱스 : 인덱스 키의 순서에 따라 데이터가 정렬되어 저장되는 방식

- 넌클러스터드 인덱스 : 인덱스의 키값만 정렬되어 있을 뿐 실제 데이터는 정렬되지 않는 방식

트리 기반 인덱스

: 인덱스를 저장하는 블록들이 트리 구조를 이루고 있는 것으로 상용 DBMS에서는 트리 구조 기반의 B+ 트리 인덱스를 주로 활용

ㄱ. B 트리 인덱스

: 일반적으로 사용하는 인덱스 방식

- 루트 노드에서 하위 노드로 키값의 크기를 비교하면서 데이터를 검색

- 모든 리프 노드의 레벨은 같음

ㄴ. B+ 트리 인덱스

: 단말 노드가 아닌 노드로 구성된 인덱스

- 세트와 단말 노드로만 구성된 순차 세트로 구분

- 인덱스 세트에 있는 노드들은 단말 노드에 있는 키 값을 찾아갈 수 있는 경로로만 제공

- 순차 세트에 있는 단말 노드가 해당 데이터 레코드의 주소를 가리킴

- 인덱스 세트에 있는 모든 키 값이 단말 노드에 다시 나타나므로 단말 노드만을 이용한 순차 처리 가능

비트맵 인덱스

: 인덱스 칼럼의 데이터를 Bit 값인 0 또는 1로 변환하여 인덱스 키로 사용하는 방법

- 키 값을 포함하는 로우(Row)의 주소를 제공

- 데이터가 Bit로 구성되어 있어 효율적인 논리 연산이 가능하고 저장공간이 작음

함수 기반 인덱스

: 칼럼의 값 대신 칼럼의 특정 함수나 수식을 적용하여 산출된 값을 사용

- B+ 트리 인덱스 또는 비트맵 인덱스를 생성하여 사용

- 데이터를 입력하거나 수정할 때 함수를 적용하기 때문에 부하가 발생할 수 있음

- 사용자 정의 함수를 사용했을 경우 시스템 함수보다 부하가 더 크다

- 대소문자, 띄어쓰기 등에 상관없이 조회할 때 유용하게 사용

비트맵 조인 인덱스

: 다수의 조인된 객체로 구성된 인덱스

도메인 인덱스

: 개발자가 필요한 인덱스를 직접 만들어 사용하는 것으로 확장형 인덱스라고도 함

인덱스 설계 순서

: 인덱스의 대상 테이블이나 칼럼 등을 선정 → 인덱스의 효율성을 검토하여 인덱스 최적화 수행 → 인덱스 정의서 작성

인덱스 테이블 선정 기준

- MULTI BLOCK READ 수에 따라 판단

MULTI BLOCK READ

: 테이블 액세스 시 메모리에 한 번에 읽어 들일 수 있는 블록의 수

- 랜덤 액세스가 빈번한 테이블
- 특정 범위나 특정 순서로 데이터 조회가 필요한 테이블
- 다른 테이블과 순차적 조인이 발생되는 테이블

인덱스 설계 시 고려사항

- 새로 추가되는 인덱스는 기존 액세스 경로에 영향을 미칠 수 있음
- 인덱스를 지나치게 만들면 오버헤드 발생
- 넓은 범위를 인덱스로 처리하면 많은 오버헤드 발생
- 인덱스를 만들면 추가적인 저장 공간 필요
- 인덱스와 테이블 데이터의 저장 공간이 분리되도록 설계

68. 뷰 설계 [필기 빈출],[실기 빈출]

뷰(View)

: 사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 하나 이상의 기본 테이블로부터 유도된 이름을 가지는 가상 테이블

- 물리적으로 존재하지는 않지만 사용자에게는 있는 것처럼 간주됨
- 데이터 보정 작업, 처리 과정 시험 등 임시적인 작업을 위한 용도로 활용
- 조인문의 최소화로 사용자 편의성을 최대화함

뷰의 특징

: 기본 테이블과 같은 형태의 구조를 사용하고 조작도 기본 테이블과 거의 같음

- 가상테이블이기 때문에 물리적으로 구현되어 있지 않음
- 데이터의 논리적 독립성을 제공할 수 있음
- 필요한 데이터만 뷰로 정의해서 처리할 수 있기 때문에 관리가 용이하고 명령문이 간단해짐
- 뷰를 통해서만 데이터에 접근하게 되면 뷰에 나타나지 않는 데이터를 안전하게 보호하는 효율적인 기법으로 사용할 수 있음
- 뷰가 정의된 기본 테이블이나 뷰를 삭제 시 그 테이블이나 뷰를 기초로 정의된 다른 뷰도 자동으로 삭제

뷰의 장단점

ㄱ. 장점

- 논리적 데이터 독립성 제공
- 동일 데이터에 대해 동시에 여러 사용자의 상이한 요구를 지원
- 사용자의 데이터 관리가 용이
- 접근 제어를 통한 자동 보안 제공

ㄴ. 단점

- 독립적인 인덱스를 가질 수 없음
- 뷰의 정의 변경 불가
- 뷰로 구성된 내용에 대해 INSERT, DELETE, UPDATE 연산에 제약이 따름

뷰 설계 순서

- 대상 테이블 선정 → 대상 칼럼 선정 → 정의서 작성

뷰 설계 시 고려사항

- 테이블 구조가 단순화될 수 있도록 반복적으로 조인을 설정하여 사용하거나 동일한 조건절을 사용하는 테이블을 뷰로 생성
- 동일한 테이블이라도 업무에 따라 테이블을 이용하는 부분이 달라질 수 있으므로 사용할 데이터를 다양한 관점에서 제시
- 데이터의 보안을 유지하며 설계

69. 클러스터의 설계

클러스터

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

- 클러스터링키로 지정된 칼럼 값의 순서대로 저장되고 여러 개의 테이블이 하나의 클러스터에 저장

클러스터의 특징

: 데이터 조회 속도는 향상시키지만 데이터 입력 수정 삭제에 대한 성능은 저하시킴

- 데이터의 분포도가 넓을수록 유리
- 대용량을 처리하는 트랜잭션은 전체 테이블을 스캔하는 일이 자주 발생하므로 클러스터링을 지양
- 파티셔닝 된 테이블에는 적용할 수 없음

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

70. 파티션 설계 [실기 빈출]

파티션

: 대용량의 테이블이나 인덱스를 작은 논리적 단위인 파티션으로 나누는 것

- 대용량 DB의 경우 테이블들을 작은 단위로 나눠 분산시키면 성능 저하를 방지하고 데이터 관리가 용이함
- 데이터 처리는 테이블 단위, 데이터 저장은 파티션 별로 수행

파티션의 장단점

ㄱ. 장점

- 데이터 접근 시 액세스 범위를 줄여 쿼리 성능 향상
- 데이터가 분산되어 저장되므로 디스크 성능 향상
- 파티션별로 백업 및 복구를 수행하므로 속도 향상
- 시스템 장애 시 데이터 손상 정도 최소화
- 데이터 가용성 향상
- 파티션 단위로 입출력 분산

ㄴ. 단점

- 하나의 테이블을 세분화하여 관리하기 때문에 세심한 관리가 요구됨
- 테이블 간 조인에 대한 비용이 증가
- 용량이 작은 테이블에 파티셔닝을 수행하면 성능이 저하됨

파티션의 종류

ㄱ. 범위 분할 : 지정한 열의 값을 기준으로 분할

ㄴ. 해시 분할 : 해시 함수를 적용한 결과 값에 따라 데이터를 분할

ㄷ. 조합 분할 : 범위 분할로 분할한 다음 해시 함수를 적용하여 다시 분할

파티션 키 선정 시 고려사항

: 파티션 키는 테이블 접근 유형에 따라 파티셔닝이 이루어지도록 선정

- 데이터 관리의 용이성을 위해 이력성 데이터는 파티션 생성 주기와 소멸 주기를 일치시켜야 함
- 매일 생성되는 날짜 칼럼, 백업의 기준이 되는 날짜 칼럼, 파티션 간 이동이 없는 칼럼, I/O 병목을 줄일 수 있는 데이터 분포가 양호한 칼럼 등을 파티션 키로 선정

인덱스 파티션

: 파티션 된 테이블의 데이터를 관리하기 위해 인덱스를 나눈 것

ㄱ. 파티션 된 테이블의 종속 여부에 따른 구분

- Local Partitioned Index : 테이블 파티션과 인덱스 파티션이 1:1 대응되도록 파티셔닝 함
- Global Partitioned Index : 테이블 파티션과 인덱스 파티션이 독립적으로 구성되도록 파티셔닝 함
- Local Partitioned Index가 Global 보다 관리하기 용이함

ㄴ. 인덱스 파티션 키 칼럼의 위치에 따른 구분

- Prefixed Partitioned Index : 인덱스 파티션 키와 인덱스 첫 번째 칼럼이 같음
- Non-Prefixed Partitioned Index : 인덱스 파티션키와 인덱스 첫 번째 칼럼이 다름

71. 데이터베이스 용량 설계

데이터베이스 용량 설계

: 데이터가 테이블에 저장될 공간을 정의

- 테이블에 저장할 데이터양, 인덱스, 클러스터 등이 차지하는 공간 등을 예측하여 반영

데이터베이스 용량 설계의 목적

: 디스크의 저장 공간을 효과적으로 사용하고 확장성 및 가용성을 높임

- 디스크의 입출력 부하를 분산시키고 채널의 병목현상 최소화
- 디스크에 대한 입출력 경합이 최소화되도록 설계함으로써 데이터 접근성이 향상
- 데이터베이스에 생성되는 오브젝트의 익스텐트 발생을 최소화하여 성능을 향상
- 테이블과 인덱스에 적합한 저장 옵션을 지정

데이터 접근성을 향상시키는 설계 방법

ㄱ. 테이블의 테이블스페이스와 인덱스의 테이블스페이스를 분리하여 구성

ㄴ. 테이블스페이스와 임시 테이블스페이스를 분리하여 구성

ㄷ. 테이블을 마스터 테이블과 트랜잭션 테이블로 분류

데이터베이스 용량 분석 절차

- 데이터 예상 건수, 로우 길이, 보존 기간, 증가율 등 기초 자료를 수집하여 용량 분석
- 분석된 자료를 바탕으로 DBMS에 이용될 테이블, 인덱스 등 오브젝트별 용량을 산정
- 테이블과 인덱스의 테이블스페이스 용량을 산정
- 데이터베이스에 저장될 모든 데이터 용량과 데이터베이스 설치 및 관리를 위한 시스템 용량을 합해 디스크 용량 산정

72. 분산 데이터베이스 설계

[필기 빈출],[실기 빈출]

분산 데이터베이스

: 논리적인 하나의 시스템이지만 물리적으로는 네트워크로 연결된 여러 개의 컴퓨터 사이트에 분산되어 있는 데이터베이스

- 데이터베이스를 네트워크를 이용해 나눠놓은 것

분산 데이터베이스의 구성 요소

- 분산 처리기 : 지리적으로 분산되어 있는 컴퓨터 시스템
- 분산 데이터베이스 : 지리적으로 분산되어 있는 데이터베이스
- 통신 네트워크 : 분산 처리기들을 네트워크로 연결하여 하나의 시스템처럼 동작할 수 있도록 하는 통신 네트워크

분산 데이터베이스 설계 시 고려사항

- 작업 부하의 노드별 분산 정책
- 지역의 자치성 보장 정책
- 데이터의 일관성 정책
- 사이트나 회선의 고장으로부터의 회복 기능
- 통신 네트워크를 통한 원격 접속 기능

분산 데이터베이스의 목표

ㄱ. 위치 투명성 : 접근하려는 데이터베이스의 실제 위치를 알 필요 없이 논리적인 명칭만으로 접근 가능

ㄴ. 중복 투명성 : 동일 데이터가 여러 곳에 중복되어 있어도 사용자는 하나의 데이터만 존재하는 것처럼 사용 가능

ㄷ. 병행 투명성 : 분산 데이터베이스와 관련된 다수의 트랜잭션들이 동시에 실현되더라도 그 트랜잭션의 결과는 영향을 받지 않음

ㄹ. 장애 투명성 : 트랜잭션, DBMS, 네트워크, 컴퓨터 등 장애에도 트랜잭션을 정확하게 처리

ㅁ. 투명성 : 사실 존재 여부를 염두에 두지 않아도 되는 성질

분산 데이터베이스의 장점

ㄱ. 장점

- 지역 자치성이 높음
- 자료의 공유성 향상
- 분산 제어 가능
- 시스템 성능 향상
- 중앙 컴퓨터의 장애가 전체 시스템에 영향을 끼치지 않음

a. 효용성과 융통성이 높음

b. 신뢰성 및 가용성이 좋음

c. 점진적 시스템 용량 확장 용이

ㄴ. 단점

- DBMS가 수행할 기능이 복잡
- DB 설계가 어려움
- 소프트웨어 개발 비용 증가
- 처리 비용 증가
- 잠재적 오류 증가

분산 데이터베이스 설계

- 애플리케이션이나 사용자가 분산되어 저장된 데이터에 접근하는 것이 목적
- 전역 관계망을 논리적 측면에서 소규모 단위로 분할, 분할된 결과를 복수의 노드에 할당
- 분산 설계 방법에는 테이블 위치 분산, 분할, 할당이 있음

테이블 위치 분산

- 데이터베이스의 테이블을 각기 다른 서버에 분산시켜 배치
- 테이블의 구조를 변경시키지 않고 다른 데이터베이스의 테이블과 중복되지 않게 배치

분할

- 테이블의 데이터를 분할하여 분산

분할 규칙

ㄱ. 완전성 : 전체 데이터를 대상으로 분할해야 함

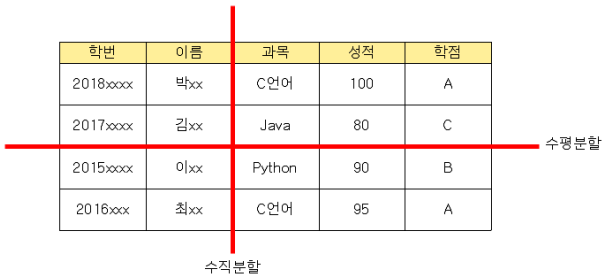
ㄴ. 재구성 : 분할된 데이터는 관계 연산을 활용하여 본래의 데이터로 재구성이 가능해야 함

ㄷ. 상호 중첩 배제 : 분할된 데이터는 서로 다른 분할의 항목에 속하지 않아야 함

분할 방법

ㄱ. 수평 분할 : 행 단위로 분할

ㄴ. 수직 분할 : 속성 단위로 분할



할당

: 동일한 분할을 여러 개의 서버에 생성하는 분산 방법

ㄱ. 비중복 할당 방식

: 최적의 노드를 선택해서 분산 데이터베이스의 단일 노드에서만 분할이 존재

ㄴ. 중복 할당 방식

: 동일한 테이블을 다른 서버에 복제하는 방식

73. 데이터베이스 이중화/서버 클러스터링

데이터베이스 이중화

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

- 하나 이상의 데이터베이스가 항상 같은 상태를 유지
- 데이터베이스가 문제가 생기면 즉시 해결 가능
- 사용자가 작업을 수행하면 이중화 시스템에 연결된 다른 데이터베이스도 동일하게 적용
- 애플리케이션을 여러 개의 데이터베이스로 분산시켜 처리해 데이터베이스의 부하를 감소

초심자를 위한 Tip

데이터베이스 이중화는 실제 현장에서 매우 중요한 개념이다. 여러분들이 사용하는 대부분의 유명한 서비스들은 거의 99%가 데이터베이스 이중화를 해 놓았다고 생각하면 된다. 그 이유는 위에 나온 것처럼 여러 가지 이유(DB과부하 or 해킹 등등)로 기존 사용하던 A라는 데이터베이스에 문제가 생기면 A라는 DB가 죽더라도 그와 똑같은 기능을 할 수 있도록 만들어놓은 B라는 DB가 바로 A의 역할을 받아서 대신해 주기 때문이다. 종종 예비용 DB까지 죽는 경우도 있는데 그럴 경우 서비스 장애가 생기게 되고 각종뉴스에 대서특필 당하게 되는 재밌는 경험을 할 수 있을 것이다(네이버 실검의 주인공이 당신이 될 수 있다!...~~아 아재 네이버 실검 없자?~~..)

데이터베이스 이중화의 분류

- 내용 전달 방식에 따른 분류

ㄱ. Eager 기법 : 트랜잭션 수행 중 데이터 변경 발생 시 이중화된 모든 데이터베이스에 즉시 전달하여 변경 내용이 즉시 적용

ㄴ. Lazy 기법 : 트랜잭션 수행이 종료되면 변경 사실을 새로운 트랜잭션에 작성하여 각 데이터베이스에 전달하여 데이터베이스마다 새로운 트랜잭션이 수행되는 것으로 간주

데이터베이스 이중화 구성 방법

ㄱ. 활동-대기 방법

- 한 DB가 서비스 시 다른 DB는 대기
- 활성 DB에 장애 발생 시 대기 중이었던 DB가 모든 서비스를 대신 수행
- 구성 방법 및 관리가 쉬워 많은 기업에서 사용

ㄴ. 활동-활동 방법

- 두 개의 DB가 서로 다른 서비스를 제공
- 한쪽 DB에 장애 발생 시 다른 DB가 서비스를 제공
- 두 DB가 모두 처리를 해 처리율이 높지만 구성 방법 및 설정이 복잡

클러스터링

- 두 대 이상의 서버를 하나의 서버처럼 운영하는 기술
- 서버 이중화 및 공유 스토로지를 사용하여 서버의 고가용성을 제공

스토리지(Storage) : 데이터를 저장하는 저장소

고가용성 : 시스템을 오랜 시간 동안 계속해서 정상적으로 운영이 가능한 성질

- 병렬 처리 클러스터링 : 처리율을 높이기 위해 하나의 작업을 여러 개의 서버에서 분산하여 처리
- 고가용성 클러스터링 : 하나의 서버에 장애가 발생 시 다른 서버가 받아서 처리하여 서비스 중단을 방지하는 방식

74. 데이터베이스 보안 / 암호화

[필기 빈출],[실기 빈출]

데이터베이스 보안

: 데이터베이스의 일부분 또는 전체에 권한이 없는 사용자가 액세스 하는 것을 금지하기 위해 사용되는 기술

암호화 / 복호화

: 암호화는 데이터를 보낼 때 송신자가 지정한 수신자 외는 그 내용을 알 수 없도록 평문을 암호문으로 변환하는 것, 복호화는 암호문을 원래의 평문으로 바꾸는 것

개인키 / 공개키 암호 방식

ㄱ. 암호화 방식의 키와 복호화 방식의 키가 같을 때

- 개인키 / 비밀키 / 대칭키 암호 방식
- 종류 : 전위 기법, 대수 기법, 합성 기법(DES)

ㄴ. 암호화 방식의 키와 복호화 방식의 키가 다를 때

- 공개키 / 비대칭키
- RSA 기법

75. 데이터베이스 보안 - 접근통제

[필기 빈출],[실기 빈출]

접근통제

: 데이터가 저장된 객체와 사용하려는 주체 사이의 정보 흐름을 제한

- 데이터에 대해 통제를 함으로써 자원의 불법적인 접근 및 파괴를 예방

ㄱ. 임의 접근통제(Discretionary Access Control)

: 데이터에 접근하는 사용자의 신원에 따라 접근 권한 부여

- 통제 권한이 주체에 있어 접근통제 권한을 주체가 지정하고 제어
- 객체를 생성한 사용자가 객체에 대한 모든 권한을 부여받고 다른 사용자에게 허가 가능
- SQL 명령어로는 GRANT, REVOKE가 있음

ㄴ. 강제 접근통제(Mandatory Access Control)

: 주체와 객체의 등급을 비교하여 접근 권한을 부여

- 제 3자가 접근통제 권한 지정

- 객체별로 보안 등급을 부여할 수 있으며 사용자별로 인가 등급을 부여할 수 있음

- 주체의 보안 등급이 자신의 보안등급보다 높으면 읽기, 수정, 등록이 모두 불가하고 동등하면 읽기, 수정, 등록이 모두 가능하며 낮으면 읽기만 가능

접근통제 정책

: 어떤 주체(Who)가 언제(When) 어디서(Where) 어떤 객체(What)에게 어떤 행위(How)에 대한 허용 여부를 정의하는 것

ㄱ. 신분 기반 정책

: 주체나 그룹의 신분에 근거하여 객체의 접근을 제한

- IBP(Individual-Based Policy) : 최소 권한 정책, 단일 주체에게 하나의 객체에 대한 허가를 부여

- GBP(Group-Based Policy) : 복수 주체에 대한 허가를 부여

ㄴ. 규칙 기반 정책

: 주체가 갖는 권한에 근거하여 객체의 접근을 제한

- MLP(Multi-Level Policy) : 사용자 및 객체별로 지정된 기밀 분류에 따른 정책

- CBP(Compartment-Based Policy) : 집단별로 지정된 기밀 허가에 따른 정책

ㄷ. 역할 기반 정책

: 주체의 신분이 아니라 주체가 맡은 역할에 근거하여 객체의 접근을 제한

접근통제 매커니즘

: 정의된 접근통제 정책을 구현하는 기술적인 방법

- 접근통제 목록 : 객체를 기준으로 특정 객체에 대해 어떤 주체가 어떤 행위를 할 수 있는지 기록한 목록

- 능력 리스트 : 주체를 기준으로 주체에게 허가된 자원 및 권한을 기록한 목록

- 보안 등급 : 주체나 객체에 부여된 보안 속성의 집합

- 패스워드 : 주체가 자신임을 증명할 때 사용하는 인증 방법

- 암호화

접근통제 보안 모델

: 보안 정책을 구현하기 위한 정형화된 모델

ㄱ. 기밀성 모델

: 군사적인 목적으로 개발된 최초의 수학적 모델로 기밀성 보장이 최우선

ㄴ. 무결성 모델

: 기밀성 모델에서 발생하는 불법적인 정보 변경을 방지하기 위해 무결성을 기반으로 개발된 모델

- 데이터 일관성 유지에 중점을 두어 개발

ㄷ. 접근통제 모델

: 접근통제 매커니즘을 보안 모델로 발전시킨 것

접근통제 모델 : 임의적인 접근 통제를 관리하기 위한 모델로 행은 주체 열은 객체를 의미

- R : 읽기 권한

- W : 쓰기 권한

- ALL : 모든 권한

접근통제 조건

: 접근통제 매커니즘의 취약점을 보안하기 위해 접근 통제 정책에 부가하여 적용할 수 있는 조건

ㄱ. 값 종속 통제 : 객체에 저장된 값에 따라 다르게 접근통제를 허용해야 하는 경우

ㄴ. 다중 사용자 통제 : 지정된 객체에 다수의 사용자가 동시에 접근을 요구하는 경우

ㄷ. 콘텍스트 기반 통제 : 특정 시간, 네트워크 주소, 접근 경로, 인증 경로 등에 근거하여 접근을 제어하는 방식

감사 추적

- 사용자나 애플리케이션의 데이터베이스에 접근하여 수행한 모든 활동을 기록하는 기능

- 오류가 발생한 데이터베이스를 복구하거나 부적절한 데이터 조작을 파악하기 위해 사용

76. 데이터베이스 백업

데이터베이스 백업

: 전산 장비의 장애에 대비하여 데이터베이스에 저장된 데이터를 보호하고 복구하기 위한 작업

데이터베이스 장애 유형

ㄱ. 사용자 실수 : 사용자의 실수로 인한 오류

ㄴ. 미디어 장애 : 하드웨어 장애나 데이터가 파손된 경우

ㄷ. 구문 장애 : 프로그램 오류나 사용 공간의 부족으로 발생하는 장애

ㄹ. 사용자 프로세스 장애 : 프로그램이 비정상적으로 종료되거나 네트워크 이상으로 세션이 종료되어 발생하는 장애

ㅁ. 인스턴스 장애 : 하드웨어 장애, 정전, 시스템 파일 파손 등 비정상적 요인으로 메모리나 데이터베이스 서버의 프로세스가 중단된 경우

로그 파일

: 데이터베이스의 처리 내용이나 이용 상황 등 상태 변화를 시간의 흐름에 따라 기록한 파일

- 로그 파일을 기반으로 과거 상태로 복귀(UNDO)시키거나 현재 상태로 재생(REDO)시켜 데이터베이스 상태를 일관성 있게 유지

데이터베이스 복구 알고리즘

- 동기적/비동기적 갱신에 따라 분류
- NO-UNDO/REDO : 비동기적으로 갱신한 경우
- UNDO/NO-REDO : 동기적으로 갱신한 경우
- UNDO/REDO : 동기/비동기적으로 갱신한 경우
- NO-UNDO/NO-REDO : 동기적으로 저장 매체에 기록하지만 데이터베이스와는 다른 영역에 기록한 경우

백업 종류

ㄱ. 물리 백업 : 데이터베이스 파일을 백업하는 방법

ㄴ. 논리 백업 : 데이터베이스 내의 논리적 객체들을 백업하는 방법

77. 스토리지(Storage)

[필기 빈출],[실기 빈출]

스토리지

: 단일 디스크로 처리할 수 없는 대용량의 데이터를 저장하기 위해 서버와 저장장치를 연결하는 기술

DAS(Direct Attached Storage)

: 서버와 저장장치를 전용 케이블로 직접 연결하는 방식

- 서버에서 저장장치를 관리
- 저장장치를 직접 연결하므로 속도가 빠르고 설치 및 운영이 쉬움
- 다른 서버에서 스토리지에 접근하여 사용 불가

NAS(Network Attached Storage)

: 서버와 저장장치를 네트워크를 통해 연결하는 방식

- 별도의 파일 관리 기능이 있는 NAS Storage가 내장된 저장장치를 직접 관리
- DAS에 비해 확장성 및 유연성이 좋음
- 서버들이 자유롭게 스토리지에 접근하여 파일 공유

SAN(Storage Area Network)

: DAS의 빠른 처리와 NAS의 파일 공유 장점을 혼합한 방식

- 서버와 저장장치를 연결하는 전용 네트워크를 별도로 구성
- 파이버 채널(FC) 스위치를 이용하여 네트워크를 구성
- 파이버 채널 : 장치 간 데이터 전송 속도를 기가바이트로 높이기 위한 네트워크 기술
- 서버나 저장장치를 광케이블로 연결하므로 처리 속도가 빠름
- 서버들이 저장장치 및 파일을 자유롭게 공유

78. 논리 데이터 모델의 물리 데이터 모델 변환

[필기 빈출],[실기 빈출]

테이블

: 데이터를 저장하는 데이터베이스의 가장 기본적인 오브젝트

엔티티를 테이블로 변환

: 논리 데이터 모델에서 정의된 엔티티를 물리 데이터 모델의 테이블로 변환

- 테이블과 엔티티 명칭은 동일하게 하는 것을 권고
- 테이블은 소스코드 가독성을 위해 영문명을 사용
- 표준화된 용어 사용 지향
- 변환 규칙

논리적 설계		물리적 설계
엔티티	=>	테이블
속성		칼럼
주 식별자		기본키
외부 식별자		외래키
관계		관계

슈퍼타입/서브타입을 테이블로 변환

- 슈퍼타입과 서브타입은 논리 데이터 모델에서 이용되는 형태이므로 물리 데이터 모델을 설계할 때는 테이블로 변환

세부 내용(정리 잘 되어있으니 이해하고 넘어갈 것)

-> https://kyungi.tistory.com/73

<III-III SQL 응용>

79. SQL의 개념 [필기 빈출],[실기 빈출]

SQL(Structured Query Language)이란?

: 구조화된 질의 언어, 국제 표준 데이터베이스 언어이며 많은 관계형 데이터베이스(RDB)를 지원하는 언어로 채택

SQL의 분류

ㄱ. DDL(Data Define Language, 데이터 정의어)

: 스키마, 도메인, 테이블, 뷰, 인덱스를 정의, 변경, 삭제할 때 사용하는 언어

a. CREATE : 스키마, 도메인, 테이블, 뷰, 인덱스를 정의
b. ALTER : 테이블에 대한 정의를 변경

c. DROP : 스키마, 도메인, 테이블, 뷰, 인덱스를 삭제

d. TRUNCATE : 테이블을 초기화

ㄴ. DML(Data Manipulation Language, 데이터 조작어)

: 사용자가 저장된 데이터를 실질적으로 처리하는 데 사용

a. SELECT : 테이블에서 조건에 맞는 튜플 검색

b. INSERT : 테이블에 새로운 튜플 삽입

c. DELETE : 테이블에서 조건에 맞는 튜플 삭제

d. UPDATE : 테이블에서 조건에 맞는 튜플의 내용 변경

ㄷ. DCL(Data Control Language, 데이터 제어어)

: 데이터의 보안, 무결성, 회복, 병행 수행 제어 등을 정의하는 데 사용하는 언어

a. COMMIT : 명령에 의해 수행된 결과를 실제 물리적 디스크로 저장하고 데이터베이스 조작 작업이 정상적으로 완료되었음을 알려줌

b. ROLLBACK : 데이터베이스 조작 작업이 비정상적으로 종료되었을 때 원래의 상태로 복구

c. GRANT : 데이터베이스 사용자에게 사용 권한을 부여

d. REVOKE : 특정 DB사용자에게 권한 박탈, 회수

초심자를 위한 Tip

위의 것들이 이해가 안갈시 참고할 수 있는 무료 유튜브 강의

-> https://www.youtube.com/watch?v=MJsOoA8yM7A

사실 SQL을 배우는 가장 좋은 방법은 실습을 하면서 직접 명령어를 날리고 눈으로 확인 하는 것이 가장 빠르고 직감적으로 배울 수 있는 방법이다. 실제로 실기에서도 SQL은 계속 나올 것이고 현업에서도 정말 많이 쓴다만 사실상 필기를 준비할 때는 개념을 외우고 기출 문제를 풀어도 충분히 합격은 할 수 있을 것이다. 다만 실기를 준비할 때와 현업에서는 암기를 하는 것으로는 한계가 있으므로, 꼭 이해를 하도록 하자!

<III-IV SQL 활용>

80. 프로시저와 트리거 [실기 빈출]

프로시저

: 절차형 SQL을 활용하여 특정 기능을 수행하는 일종의 트랜잭션 언어이며 호출을 통해 실행되어 미리 저장해 놓은 SQL 작업을 수행, 여러 프로그램에서 호출하여 사용 가능하고, 시스템의 일일 마감 작업, 일괄 작업 등에 주로 사용됨

프로시저 구성

- DECLARE : 프로시저의 명칭, 변수, 인수, 데이터 타입을 정의하는 선언부 (필수)
- BEGIN / END : 프로시저의 시작과 종료를 의미 (필수)
- CONTROL : 조건문 또는 반복문이 삽입, 순차적 처리
- SQL : DML, DCL이 삽입돼, 조회, 추가, 수정, 삭제 작업을 수행
- EXCEPTION : 구문 실행 중 예외 발생 시 처리 방법 정의
- TRANSACTION : 작업들을 DB에 적용할지 취소할지 결정하는 처리부

프로시저 생성과 실행, 제거

표기 형식

: CREATE PROCEDURE 프로시저명(파라미터)
BEGIN
BODY;
END;

- CREATE와 PROCEDURE 사이에 OR REPLACE 예약어 사용 시 동일한 프로시저 이름이 존재할 경우, 기존의 프로시저를 대체함
- 파라미터는 IN, OUT, INOUT, 매개변수명, 자료형이 올 수 있음. 값을 전달하거나 반환할 때 IN과 OUT, INOUT을 사용함
- 프로시저의 BODY는 BEGIN과 END 사이의 코드 작성 부분이며 하나 이상의 SQL 문이 있어야 한다.
- 프로시저 실행을 위해 EXECUTE, CALL, EXEC로 사용할 수 있음 ex) EXEC 프로시저명;
- 제거를 위해서는 DROP PROCEDURE를 사용함

트리거

: DB 시스템에서 삽입, 갱신, 삭제 등 이벤트가 발생할 때마다 자동 수행되는 절차형 SQL로 무결성 유지, 로그 메시지 출력 등의 목적으로 사용함

-구문에는 DCL을 사용할 수 없고 DCL이 포함된 프로시저나 함수 호출도 불가능

트리거 구성

- DECLARE : 프로시저와 같음
- EVENT : 실행되는 조건 명시
- BEGIN / END : 프로시저와 같음
- CONTROL : 프로시저와 같음

- SQL : 프로시저와 같으나 DCL은 삽입될 수 없음
- EXCEPTION : 프로시저와 같음

트리거 생성

표기 형식

: CREATE TRIGGER 트리거명 (실행 시기)(옵션) ON 테이블명
REFERENCING (NEW or OLD) AS 테이블명
FOR EACH ROW
BEGIN
BODY;
END;

- OR REPLACE는 프로시저와 동일하고, 실행 시기는 이벤트가 발생하기 전에 실행하는지 후에 실행하는지에 따라 AFTER와 BEFORE를 사용, 그 뒤에 실행되게 할 작업의 종류를 지정하는 옵션에는 INSERT, DELETE, UPDATE가 있음
- 추가되거나 수정에 참여할 테이블은 NEW, 수정되거나 삭제 전 대상이 되는 테이블은 OLD로 사용한다.
- FOR EACH ROW 는 각 튜플마다 트리거를 적용한다는 의미이며 BODY 부분은 프로시저와 같다.
- 제거시 DROP TRIGGER 명령어 사용

공통점과 차이점

- 데이터베이스에 저장하며 절차형 SQL인 것과 구성 시 DECLARE, BEGIN, END를 필수로 가져야 하는 점은 같음

그러나 목적이 다르고 트리거는 데이터 제어어를 사용할 수 없다. EVENT로 자동 수행하는 부분 등, 상세한 부분의 차이를 알아두어야 한다.

81. 사용자 정의 함수

사용자 정의 함수

: 프로시저와 유사하게 SQL을 사용하여 일련의 작업을 연속적으로 처리하여 종료 시 처리 결과를 단일 값으로 반환하는 절차형 SQL

- 데이터베이스에 저장되어 DML문의 호출에 의해 실행
- 예약어 RETURN을 통해 값이 반환되기 때문에 출력 파라미터가 없음
- 테이블 조작은 할 수 없고 SELECT를 통해 검색만 할 수 있음
- 프로시저를 호출하여 사용할 수 없음
- 프로시저의 구성과 유사함

표기 형식

: CREATE [OR REPLACE] FUNCTION 사용자 정의 함수명(파라미터)
[지역변수 선언]
BEGIN
    사용자 정의 함수 BODY;
    RETURN 반환;
END;

82. DBMS 접속 기술

DBMS 접속 기술

: 사용자가 데이터를 접속하기 위해 응용 시스템을 이용하여 DBMS에 접근하는 것

- 응용 시스템은 사용자로부터 매개 변수를 전달받아 SQL을 실행하고 DBMS로부터 전달받은 결과를 사용자에게 전달
- 웹 응용프로그램은 웹 응용 시스템을 통해 DBMS에 접근
- 웹 응용 시스템은 웹 서버와 웹 애플리케이션 서버(WAS)로 구성
    -> 사용자 ↔ 웹 서버 ↔ WAS ↔ DBMS

DBMS 접속 기술

: DBMS에 접근하기 위해 사용하는 API 또는 프레임워크를 의미

초심자를 위한 Tip

API(Application Programming Interface) : 응용 프로그램 개발 시 운영 체제나 DBMS 등을 이용할 수 있도록 규칙 등에 대해 정의해 놓은 인터페이스

프레임워크 : 소프트웨어에서는 특정 기능을 수행하기 위해 필요한 클래스나 인터페이스 등을 모아둔 집합체

ㄱ. JDBC(Java DataBase Connectivity)

- 썬 마이크로시스템에서 출시(java언어)
- Java SE에 포함되어 있고 JDBC 클래스는 java.sql, javax.sql에 포함
- 접속하려는 DBMS에 대한 드라이버 필요

ㄴ. ODBC(Open DataBase Connectivity)

- 개발 언어와 상관없음
- 마이크로소프트에서 출시
- MS-Access, DBase, DB2, Excel, Text 등 다양한 데이터베이스에 접근 가능

ㄷ. MyBatis

- JDBC 코드를 단순화하여 사용할 수 있는 SQL Mapping 기반 오픈소스 접속 프레임워크
- SQL 문장을 분리하여 XML 파일을 만들고 Mapping을 통해 SQL을 실행
- SQL을 거의 그대로 사용할 수 있어 국내 환경에 적합

동적 SQL

: 개발 언어에 삽입되는 SQL 코드를 문자열 변수에 넣어 처리하는 것

- 조건에 따라 SQL 구문을 동적으로 변경하여 처리 가능
- NVL 함수를 사용할 필요가 없음
- 응용 프로그램 수행 시 SQL이 변형될 수 있어 프리컴파일할 때 구문 분석, 접근 권한 확인 등을 할 수 없음

83. SQL 테스트

SQL 테스트

: SQL이 작성 의도에 맞게 원하는 기능을 수행하는지 검증하는 과정

- 단문 SQL은 코드를 직접 실행한 후 결과를 확인

- 절차형 SQL은 테스트 전에 생성을 통해 구문 오류나 참조 오류의 존재 여부 확인

- 정상적으로 생성된 절차형 SQL은 디버깅을 통해 로직을 검증하고 결과를 통해 최종 확인

단문 SQL 테스트

- DDL, DML, DCL이 포함되어 있는 SQL과 TCL을 직접 실행하여 테스트

- DESCRIBE 명령어를 이용하면 DDL로 작성된 테이블이나 뷰의 속성, 자료형, 옵션들을 확인할 수 있음

- DCL로 설정된 사용자 권한은 사용자 권한 정보가 저장된 테이블을 SELECT문으로 조회하거나 SHOW 명령어로 확인할 수 있음

ㄱ. Oracle : SELECT * FROM DBA_ROLE_PRIVES WHERE GRANTEE = 사용자;

ㄴ. MySQL : SHOW GRANTS FOR 사용자@호스트;

절차형 SQL 테스트

: 프로시저, 사용자 정의 함수, 트리거 등의 절차형 SQL은 디버깅을 통해 기능의 적합성 여부 검증, 실행을 통해 결과를 확인하는 테스트를 수행

- SHOW 명령어를 통해 오류 내용을 확인

ex) SHOW ERRORS;

- 데이터베이스에 변화를 줄 수 있는 SQL문은 주석 처리 후 출력문을 이용하여 결과 확인

ㄱ. Oracle : DBMS_OUTPUT.ENABLE; / DBMS_OUTPUT.PUT_LINE(데이터);

ㄴ. MySQL : SELECT 데이터;

84. ORM

ORM(Object-Relational Mapping)

: 객체지향 프로그래밍의 객체와 관계형 데이터베이스의 데이터를 연결하는 기술

- 객체지향 프로그래밍에서 사용할 수 있는 가상의 객체지향 데이터베이스를 만들어 프로그래밍 코드와 데이터 연결

- 프로그래밍 코드 또는 데이터베이스와 독립적이므로 재사용 및 유지보수가 용이

- SQL 코드를 직접 사용하지 않기 때문에 직관적이고 간단하게 데이터 조작 가능

ORM 프레임워크

: ORM을 구현하기 위한 구조와 구현을 위해 필요한 여러 기능들을 제공하는 소프트웨어

- JAVA : JPA, Hibernate, EclipseLink, DataNucleus, Ebean 등
- C++ : ODB, QxOrm 등
- Python : Django, SQLAlchemy, Storm 등
- iOS : DatabaseObjects, Core Data 등
- .NET : NHibernate, DatabaseObjects, Dapper 등
- PHP : Doctrine, Propel, RedBean 등

ORM의 한계

- 프레임워크가 자동으로 작성하기 때문에 의도대로 작성되었는지 확인할 필요가 있음

- 객체지향적 사용을 고려, 설계한 데이터베이스가 아닌 경우 프로젝트가 크고 복잡할수록 ORM 기술을 적용하기 어려움

- 기존의 기업들은 ORM 고려하지 않은 데이터베이스를 사용하고 있어 ORM에 적합하게 변환하려면 많은 시간과 노력이 필요

85. 쿼리 성능 최적화

쿼리 성능 최적화

: 데이터 입출력 애플리케이션의 성능 향상을 위해 SQL 코드를 최적화

- 최적화 전 APM을 사용하여 최적화할 쿼리 선정

- APM(Application Performance Management/Monitoring)

: 애플리케이션의 성능 관리를 위해 다양한 모니터링 기능을 제공하는 도구

- RBO(Rule Based Optimizer) : 규칙 기반 옵티마이저

- CBO(Cost Based Optimizer) : 비용 기반 옵티마이저

초심자를 위한 Tip [필기 빈출],[실기 빈출]

옵티마이저(Optimizer)란?

SQL 개발자가 SQL을 작성하여 실행할 때, 옵티마이저는 SQL을 어떻게 실행할 것인지를 계획하게 된다

SQL 실행 계획(Execution Plan)을 수립하고 SQL을 실행할시

옵티마이저는 SQL의 실행 계획을 수립하고 SQL을 실행하는 데이터베이스 관리 시스템의 소프트웨어이다

동일한 결과가 나오는 SQL도 어떻게 실행하느냐에 따라서 성능이 달라지기에, SQL 성능에 옵티마이저는 아주 중요한 역할을 한다

실행 계획

: DBMS의 옵티마이저가 수립한 SQL 코드의 실행 절차와 방법을 의미

- EXPLAIN 명령어를 통해 확인

- 그래픽이나 텍스트로 표현

- 요구사항을 처리하기 위한 연산 순서가 적혀있고 연산에는 조인, 테이블 검색, 필터, 정렬 등이 있음

쿼리 성능 최적화

: 실행 계획에 표시된 연산 순서, 조인 방식, 테이블 조회 방법 등을 참고하여 SQL문이 더 빠르고 효율적으로 작동하도록 코드와 인덱스를 재구성

ㄱ. SQL 코드 재구성

a. WHERE 절을 추가하여 일부 레코드만 조회

b. WHERE 절에 연산자 사용 자제

c. 특정 데이터 확인 시 IN보다 EXISTS 사용

d. 힌트를 활용하여 실행 계획의 액세스 경로 및 조인 순서 변경

ㄴ. 인덱스 재구성

a. SQL 코드에서 조회되는 속성과 조건들을 고려하여 인덱스 구성

b. 인덱스를 추가하거나 기존 인덱스의 열 순서 변경

c. 단일 인덱스로 쓰기나 수정 없이 읽기로만 사용되는 경우 IOT(Index-Organized Table)로 구성

<III-V 데이터 전환>

86. 데이터 전환

데이터 전환

: 데이터 전환이란 운영 중인 기본 정보 시스템에 축적되어 있는 데이터를 추출하여 새로 개발할 정보 시스템에서 운영 가능하도록 변화한 후, 적재하는 일련의 과정을 말한다

- 데이터 전환을 ETL(Extraction, Transformation, Load), 즉 추출, 변환, 적재 과정이라고 함

- 데이터 전환을 데이터 이행(Data Migration) 또는 데이터 이관이라고 함

데이터 전환 계획서

- 데이터 전환 계획서는 데이터 전환이 필요한 대상을 분석하여 데이터 전환 작업에 필요한 모든 계획을 기록하는 문서

데이터 전환 대상 및 범위

- 데이터 전환 대상 정보, 해당 업무에 사용되는 Table 수, 데이터 크기를 기술

데이터 전환 환경 구성

- 데이터 전환 환경 구성 항목에는 원천 시스템과 목적 시스템의 구성도, 전환 단계별 DISK사용량을 기술 한다

- 원천 시스템 구성도 : 원천 시스템의 서버, 스토리지, 네트워크 등을 포함한 구성도 작성

- 목적 시스템 구성도 : 목적 시스템의 서버, 스토리지, 네트워크 등을 포함한 구성도 작성

- 전환 단계별 DISK 용량 산정 : 전환 검증, 시험 단계, 본 전환 단계별로 요구되는 File공간과 DB공간을 산정하여 기술

데이터 전환 조직 및 역할 작성

- 데이터 전환 조직 및 역할에는 데이터 전환을 수행하고 결과를 검증할 작업자와 작업자별 역할을 최대한 상세히 기술

데이터 전환 방안

- 데이터 전환 방안 항목에는 데이터 전환 규칙, 데이터 전환 절차, 데이터 전환 방법, 데이터 전환 설계, 전환 프로그램 개발 및 테스트 계획, 데이터 전환 계획, 데이터 검증 방안이 있다

데이터 전환 규칙

- 데이터 전환 규칙 항목에는 데이터 전환 과정에서 공통적으로 적용해야 할 규칙들을 기술

데이터 전환 절차

- 데이터 전환 절차 항목에는 전환 준비, 전환 설계/개발, 전환 테스트, 실 데이터 전환, 최종 전환 및 검증의 데이터를 전환 절차를 체계적이고 상세하게 기술

데이터 전환 방법

- 데이터 전환 방법 항목에는 단위 업무별로 데이터 전환 방법을 기술하되, 데이터 전환 시 업무별로 요구되는 전제 조건도 함께 기술

데이터 전환 설계

- 데이터 전환 설계 항목에는 업무별로 전환 대상과 전환 제외 대상을 기술하고 원천 시스템 테이블과 목적 시스템 테이블의 매핑 정의서 작성

전환 프로그램 개발 및 테스트 계획

- 전환 프로그램 개발 및 테스트 계획 항목에는 전환 프로그램 개발 계획과 전환 프로그램 테스트 계획을 수립한 후 관련 내용을 기술

데이터 전환 계획

- 데이터 전환 계획 항목에는 데이터 전환 시간을 단축하기 위해 선 전환, 본 전환, 후 전환으로 분리하여 계획을 수립한 후 관련 내용을 기술

- 데이터 전환 시간을 단축하기 위해 일자별 거래 내역, 일자별 근태 내역과 같은 대량의 데이터 테이블은 사전에 전환

데이터 검증 방안

- 데이터 검증 방안 항목에는 데이터 전환 이후 전환 데이터의 정합성을 검증하고 전환 과정에서 발생할 수 있는 문제에 대응할 수 있도록 단계별 데이터 전환 검증 방안을 수립한 후 관련 내용을 기술

데이터 검증

: 데이터 검증이란 시스템의 데이터를 목적 시스템의 데이터로 전환하는 과정이 정상적으로 수행되었는지 여부를 확인하는 과정

- 검증 방법과 검증 단계에 따라 분류할 수 있다.

검증 방법에 따른 분류

- 로그 검증, 기본 항목 검증, 응용 프로그램 검증, 응용 데이터 검증, 값 검증

검증 단계에 따른 분류

- 추출, 전환, DB적재, DB적재 후, 전환 완료 후

Ⅳ. 프로그래밍 언어 활용 (대단원4/5)

<Ⅳ-Ⅰ서버 프로그램 구현>

87. 개발 환경 구축 [필기 빈출],[실기 빈출]

개발 환경 구축

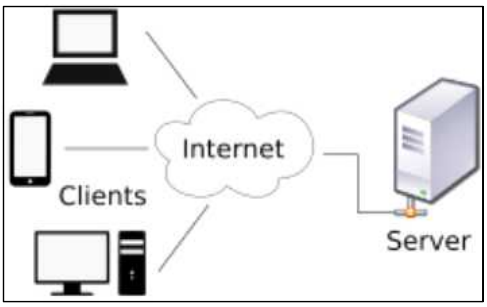
: 응용 소프트웨어 개발을 위해 개발 프로젝트를 이해하고 S/W 및 H/W 장비를 구축하는 것

(S/W : 소프트웨어 / H/W : 하드웨어)

- 응용 소프트웨어가 운영될 환경과 유사한 구조로 구축
- 개발 프로젝트의 분석 단계의 산출물을 바탕으로 개발에 필요한 S/W와 H/W를 선정
- S/W와 H/W의 성능, 편의성, 라이선스 등 비즈니스 환경에 적합한 제품들을 최종적으로 결정하여 구축

H/W 환경

- 사용자의 인터페이스 역할을 하는 클라이언트와, 클라이언트와 통신하여 서비스하는 서버로 구성



웹 서버

: 클라이언트로부터 직접 요청을 받아 처리하는 서버

- 저용량의 정적 파일(HTML, CSS 등)을 제공
- ex) Apache HTTP Server, Microsoft Internet Information Service 등

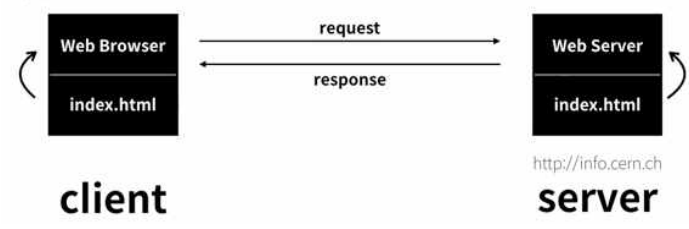
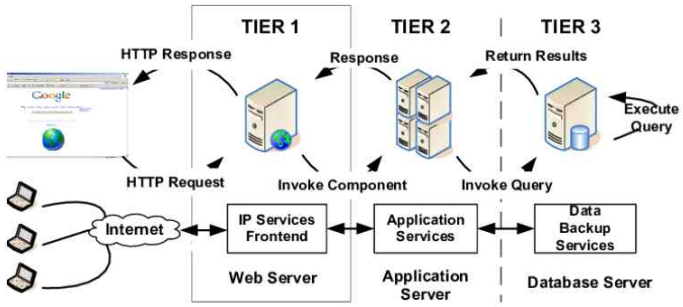


사진 출처 : 생활코딩

웹 애플리케이션 서버(Web Application Server)

- 사용자에게 동적 서비스를 제공하기 위해 웹 서버로부터 요청을 받아 데이터 가공 작업을 수행
- 웹 서버와 데이터베이스 서버 또는 웹 서버와 파일 서버 사이에서 인터페이스 역할 수행

ex) Apache Tomcat, IBM WebSphere 등



초심자를 위한 Tip

: 웹 서버와 WAS서버의 차이는 동적인 처리를 하느냐 아니냐의 차이이다. 웹 서버는 주로 정적 컨텐츠(html, png css등)를 처리하는 경우 사용하고 정적 컨텐츠가 아니라서 웹서버에서는 처리가 힘든 경우 WAS서버로 요청을 보낸다. WAS서버는 정적 컨텐츠가 아닌 동적 콘텐츠(DB조회, 로직처리 요구 등)를 처리하는 경우 사용된다. WAS서버는 동적처리 정적처리 모두 가능하지만 정적처리를 WAS가 하게 될 시 부하가 많이 걸려서 좋지 않다.

추천 글

->https://jeong-pro.tistory.com/84

데이터베이스 서버

: 데이터베이스와 이를 관리하는 DBMS를 운영하는 서버

ex) MySQL Server, Oracle Server 등

파일 서버

: 데이터베이스에 저장하기에는 비효율적이거나 서비스 제공 목적으로 유지하는 파일들을 저장하는 서버

ex) AWS, S3 등

S/W 환경

- 클라이언트 서버 운영을 위한 시스템 S/W와 개발에 사용되는 개발 S/W로 구성

ㄱ. 시스템 S/W

- O/S, 웹 서버, WAS 운용을 위한 서버 프로그램
- DBMS

ㄴ. 개발 S/W

- 요구사항 관리 도구 : 요구사항 수집과 분석, 추적 등을 편리하게 도와주는 S/W
- 설계 모델링 도구 : UML을 지원하며 개발의 전 과정에서 설계 및 모델링을 도와주는 S/W
- 구현 도구 : 개발 언어를 통해 애플리케이션의 실제 구현을 지원하는 S/W
- 빌드 도구 : 구현 도구를 통해 작성된 소스의 빌드 및 배포, 라이브러리 관리를 지원하는 S/W

- 테스트 도구 : 모듈들이 요구사항에 적합하게 구현되었는지 테스트하는 S/W
- 형상 관리 도구 : 버전 관리를 하여 품질 향상을 지원하는 S/W

87. 서버 개발 [필기 빈출],[실기 빈출]

서버 개발

: 웹 애플리케이션의 로직을 구현할 서버 프로그램을 제작하여 WAS에 탑재하는 것

- 서버 프로그램을 개발할 수 있도록 지원하는 프레임워크가 있음

서버 개발 프레임워크

: 서버 프로그램 개발 시 다양한 설정을 손쉽게 할 수 있도록 클래스나 인터페이스를 제공하는 S/W

- 뷰-모델-컨트롤러(MVC) 패턴을 기반으로 개발됨

ㄱ. Spring

- Java 기반 프레임워크
- 전자정부 표준 프레임워크의 기반 기술로 사용

ㄴ. Node.js

- Javascript 기반 프레임워크
- 실시간으로 입출력이 빈번한 애플리케이션에 적합

ㄷ. Django

- Python 기반 프레임워크
- 컴포넌트 재사용 및 플러그인화를 기반으로 신속한 개발을 지원

ㄹ. Codeigniter

- PHP 기반 프레임워크
- 인터페이스가 간편하며 서버 자원을 적게 사용

ㅁ. Ruby on Rails

- Ruby 기반 프레임워크
- 테스트를 위한 웹 서버 지원
- 데이터베이스 작업을 단순화, 자동화시켜 신속한 개발 가능
- 프레임워크의 특성 : 모듈화, 재사용성, 확장성, 제어의 역 흐름

서버 프로그램 구현

- 응용 S/W와 동일하게 모듈 및 공통 모듈 개발 후 통합하는 방식으로 구현
- 모듈 개발 시 다른 모듈과의 과도한 상호작용을 배제함으로써 특정 모듈의 수정이 다른 모듈에게 영향을 미치지 않아야 함
- 모듈의 독립성은 결합도와 응집도에 의해 측정
- 공통 모듈은 여러 프로그램에서 재사용할 수 있는 모듈

결합도 [필기 빈출],[실기 빈출]

: 어떤 모듈이 다른 모듈에 의존하는 정도를 나타내는 것으로 독립적인 모듈이 되기 위해서는 낮을수록 좋음

결합도의 종류 [필기 빈출],[실기 빈출]

자료 결합도 < 스탬프 결합도 < 제어 결합도 < 외부 결합도 < 공통 결합도 < 내용 결합도

(낮을수록 좋음)

ㄱ. 자료 결합도(Data Coupling) : 모듈간의 인터페이스 전달되는 파라미터를 통해서만 모듈간의 상호 작용이 일어나는 경우.

ㄴ. 스탬프 결합도(Stamp Coupling) : 모듈간의 인터페이스로 배열이나 오브젝트, 스트럭쳐등이 전달되는 경우.

제어 결합도(Control Coupling) : 단순히 처리를 해야할 대상인 값만 전달되는게 아니라 어떻게 처리를 해야 한다는 제어 요소(DCD, Flag등)이 전달되는 경우.

ㄷ. 외부 결합도(External Coupling) : 어떤 모듈에서 반환한 값을 다른 모듈에서 참조해서 사용하는 경우

ㄹ.공통 결합도(Common Coupling) : 파라미터가 아닌 모듈 밖에 선언되어 있는 전역 변수를 참조하고 전역변수를 갱신하는 식으로 상호작용하는 경우

ㅁ. 내용 결합도(Content Coupling) : 다른 모듈 내부에 있는 변수나 기능을 다른 모듈에서 사용 하는 경우

응집도 [필기 빈출],[실기 빈출]

: 하나의 모듈이 하나의 기능을 수행하는 요소들 간의 연관성 척도, 독립적인 모듈이 되기 위해서는 응집도가 높을수록 좋음

응집도 순서 [필기 빈출],[실기 빈출]

우연적 응집도 < 논리적 응집도 < 시간적 응집도 < 절차적 응집도 < 교환적 응집도 < 순차적 응집도 < 기능적 응집도

ㄱ. 기능적 응집도(Functional Cohesion) : 모듈 내 모든 요소들이 단일 기능을 수행

ㄴ. 순차적 응집도(Sequential Cohesion) : 모듈 내의 한 요소의 출력 자료가 다음 요소의 입력 자료로 사용

ㄷ. 교환적 응집도(Communication Cohesion) : 모듈 내의 요소들이 동일한 입출력 자료로 서로 다른 기능을 수행

ㄹ. 절차적 응집도(Procedural Cohesion) : 모듈 수행 요소들이 반드시 특정 순서대로 수행

ㅁ. 시간적 응집도(Temporal Cohesion) : 특정 시간에 실행되는 기능들을 모아 작성된 모듈

ㅂ. 논리적 응집도(Logical Cohesion) : 논리적으로 유사한 기

능을 수행 하지만 서로의 관계는 밀접하지 않음

ㅅ. 우연적 응집도(Coincidental Cohesion) : 모듈 내 요소들이 뚜렷한 관계가 없이 존재, 어떠한 의미 있는 연관관계도 지니지 않은 기능 요소로 구성되고 서로 다른 상위 모듈에 의해 호출되어 처리상의 연관성이 없는 서로 다른 기능을 수행하는 경우

88. 보안 및 API [필기 빈출],[실기 빈출]

S/W 개발 보안

: 개발 과정에서 발생할 수 있는 보안 취약점을 최소화하여 보안 위협으로부터 안전한 S/W를 개발하기 위한 보안 활동

- 데이터의 기밀성, 무결성, 가용성을 유지해야 함
- S/W 개발 보안 가이드를 참고하여 점검해야 할 보안 항목을 점검해야 함

S/W 개발 보안 점검 항목

ㄱ. 세션 통제

- 서버와 클라이언트의 연결 간 발생하는 정보를 관리
- 불충분한 세션 관리 또는 잘못된 세션에 의한 정보 노출

ㄴ. 입력 데이터 검증 및 표현

- 입력 데이터에 대한 유효성 검증 체계를 갖추고 실패 시 이를 처리할 수 있도록 코딩하는 것
- SQL 삽입, 경로 조작 및 자원 삽입, 크로스 사이트 스크립팅

ㄷ. 보안 기능

- 인증, 접근제어, 기밀성, 암호화 등의 기능
- 적절한 인증 없는 중요 기능 허용, 부적절한 인가

ㄹ. 시간 및 상태

- 병렬 처리 시스템이나 다수의 프로세스가 동작하는 환경에서 시간과 실행 상태를 관리하여 시스템이 원활히 동작되도록 하는 것
- 검사 시점과 사용 시점 경쟁 조건, 무한 루프, 재귀 함수

ㅁ. 에러 처리

- S/W 실행 중 발생할 수 있는 오류들을 사전에 정의하여 에러로 인해 발생할 수 있는 문제들을 예방
- 오류 메시지를 통한 정보 노출, 오류 상황 대응 부재

ㅂ. 코드 오류

- 형 변환, 자원의 반환 등을 고려하여 코딩하는 것
- 널 포인터 역참조, 부적절한 자원 해제

ㅅ. 캡슐화

- 데이터와 데이터를 처리하는 함수를 하나의 객체로 묶어 코딩하는 것
- 잘못된 세션에 의한 정보 노출, 제거되지 않고 남은 디버그 코드

ㅇ. API 오용

- API를 잘못사용하거나 보안의 취약한 API를 사용하지 않도록 고려하여 코딩하는 것
- DNS lookup에 의존한 보안 결정

API(Application Programming Interface)

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

89. 배치 프로그램 [실기 빈출]

배치 프로그램

: 사용자와의 상호 작용 없이 여러 작업들을 미리 정해진 작업을 일괄적으로 처리하는 것

- 배치 프로그램의 필수 요소 : 대용량 데이터, 자동화, 견고성, 안정성, 신뢰성, 성능
- 정기 배치 : 정해진 기간에 정기적으로 수행
- 이벤트성 배치 : 설정한 특정 조건이 충족될 때 수행
- On-Demand 배치 : 사용자 요청 시 수행

배치 스케줄러

: 일괄 처리 작업이 설정된 주기에 맞춰 자동으로 수행되도록 지원해주는 도구

- 잡 스케줄러라고도 함

ㄱ. 스프링 배치

- Spring Source 사와 Accenture 사가 공동 개발한 오픈소스 프레임워크
- 스프링 프레임워크의 특성을 그대로 가져와 스프링의 기능을 모두 사용할 수 있음
- 데이터베이스나 파일의 데이터를 교환하는데 필요한 컴포넌트를 제공
- 로그 관리, 추적, 트랜잭션 관리, 작업 처리 통계, 작업 재시작 등의 다양한 기능 제공
- 구성요소 : Job, Job Launcher, Step, Job Repository

ㄴ. Quatz

- 스프링 프레임워크로 개발되는 응용 프로그램들의 일괄 처리를 위한 다양한 기능을 제공하는 오픈소스 라이브러리
- 수행할 작업과 수행 시간을 관리하는 요소들을 분리하여 일괄 처리 작업에 유연성을 제공
- 구성요소 : Scheduler, Job, JobDetail, Trigger

90. 패키지 소프트웨어

패키지 소프트웨어

: 기업에서 일반적으로 사용하는 여러 기능을 통합하여 제공하는 S/W

- 기업에서는 패키지 소프트웨어를 구입하여 기업 환경에 적합하게 커스터마이징하여 사용

패키지 소프트웨어의 특징

: 요구사항을 분석하여 업무 특성에 맞게 전용으로 개발되는 S/W와 비교하여 안정성, 라이선스, 생산성 등의 차이가 있음

- 전문 업체에 의해 품질이 검증되었고 국제 산업계 표준을 준수하고 있어 안정적인 이용 가능
- S/W에 대한 라이선스가 판매자에게 있어 시스템 구축 후 기능 추가 및 코드 재사용 등에 제약이 있음
- 개발 조직을 갖추어야 할 필요성이 없어 비용을 절감할 수 있음
- 이미 개발된 S/W를 사용하기 때문에 프로젝트 기간 단축

<IV-II 프로그래밍 언어 활용>

91. 프로그래밍 언어 [필기 빈출],[실기 빈출]

개발도구

- 구성 : 빌드 도구, 구현 도구, 테스트 도구, 형상관리 도구
- 빌드 도구 : Gradle, Maven, Ant
- 테스트 도구 : xUnit, PMD, Findbugs, Cppcheck, Sonar

형상관리

: 소프트웨어 개발을 위한 전체 과정에서 발생하는 모든 항목의 변경 사항을 관리하기 위한 활동

프레임워크

- 소프트웨어의 구체적인 부분에 해당하는 설계와 구현을 재사용이 가능하게끔 협업화된 형태로 클래스들을 제공하는 틀
- 구성요소 : 개발환경, 실행환경, 운영환경, 관리환경
- 특징 : 모듈화, 재사용성, 확장성, 제어의 역행
- 재사용성 : 인터페이스를 통해 여러 애플리케이션에서 반복적으로 사용되는 일반적 컴포넌트를 정의하여 재사용성을 높임
- 확장성 : 다형성을 통해 애플리케이션이 프레임워크의 인터페이스를 확장
- 제어의 역행 : 프레임워크 코드가 전체 애플리케이션의 처리흐름을 제어

모듈화

: 프로그램 개발 시 생산성과 최적화, 관리에 용이하게 기능단위로 분할하는 기법

- 원리 : 정보은닉, 분할과 정복, 데이터 추상화, 모듈의 독립성
- 유형 : 설계 -> 모듈화, 컴포넌트, 서비스 // 구현 -> 함수, 매크로, 인라인

배치 프로그램

: 사용자와의 상호작용 없이 일련의 작업들을 작업 단위로 묶어 정기적으로 반복 수행하거나 정해진 규칙에 따라 일괄 처리하는 방법

- 유형 : 이벤트 배치, 온디멘드 배치, 정기 배치
- 온디멘드 배치 : 사용자의 명시적 요구가 있을 때마다 실행하는 방법
- 정기 배치 : 정해진 시점(주로 야간)에 실행하는 배치

프로세스 (= 작업, task)

: CPU에 의해 처리되는 사용자 프로그램, 시스템 프로그램, 즉 실행중인 프로그램을 의미

- 상태 : 생성, 준비, 실행, 대기, 완료
- PID(Process identifier) : 프로세스 식별자
- PCB(Process Control Block) : 운영체제가 프로세스 관리를 위해 필요한 자료를 담고 있는 자료 구조로 프로세스 식별자, 프로세스 상태, 프로그램 카운트, 레지스터 저장 영역, 프로세서 스케줄링 정보, 계정 정보, 입출력 상태 정보, 메모리 관리 정보 등을 담고 있음

프로세스 상태 전이

: 하나의 작업이 컴퓨터 시스템에 입력되어 완료되기까지 프로세스의 상태가 준비, 실행 및 대기 상태 등으로 변화는 활동

프로세스 상태 전이 종류

: 디스패치, 타이머 런 아웃, 블록, 웨이크업, Swap-in/out

- Swap-in : 프로세스에게 다시 기억 장치가 할당될 경우
- Swap-out : 프로세스가 기억 장치를 잃은 경우

프로토콜

: 서로 다른 시스템에 있는 두 개체 간의 데이터 교환을 원활히 하기 위한 일련의 통신 규약

IP

: 송신 호스트와 수신 호스트가 패킷 교환 네트워크에서 정보를 주고받는데 사용하는 정보 위주의 규약

- OSI 3계층에서 호스트의 주소 지정과 패킷 분할 및 조립 기능을 담당하는 프로토콜

TCP(Transmission Control Protocol)

: 근거리 통신망이나 인트라넷, 인터넷에 연결된 컴퓨터에서 실행되는 프로그램 간에 일련의 옥텟을 안정적으로, 순서대로, 에러 없이 교환할 수 있게 하는 프로토콜

UDP (User Datagram Protocol)

: 같은 네트워크 내에서 컴퓨터들 간에 메시지들이 교환될 때 제한된 서비스만을 제공하는 통신 프로토콜이다

IPSEC(Internet Protocol Security)

: IP계층에서 무결성과 인증을 보장하는 인증 헤더와 기밀성을 보장하는 암호화를 이용한 IP 보안 프로토콜

- 현재 전 세계에서 사용되는 인터넷 상거래시 요구되는 개인정보와 크레딧카드 정보의 보안 유지에 가장 많이 사용되고 있는 프로토콜

제네릭 프로그래밍

: 재사용 프로그래밍 기법 중 하나의 값이 여러 데이터 타입을 가질 수 있어서 재사용성을 높일 수 있는 특징이 있는 프로그래밍 방식

- 재사용 프로그래밍 기법 : 객체지향, 제네릭, 자동, 메타 프로그래밍

- 메타 프로그래밍 : 런타임에 수행해야 할 작업의 일부를 컴파일 타임 동안 수행하는 프로그램

객체지향 프로그래밍

- 구성 요소 : 객체, 클래스, 메시지

- 원리 : 캡슐화, 추상화, 다형성, 정보은닉, 상속성

윈도우 계열 운영체제 특징

: GUI사용(Graphical User Interface), 선점형 멀티태스킹 방식 사용, 자동 감지 기능(Plug &Play), OLE사용

유닉스 계열 운영체제 특징

: 대화식 운영체제 기능 제공, 다중 작업 기능 제공, 다중 사용자 기능 제공, 이식성 제공, 계층적 트리 구조 파일 시스템 제공

스레싱

: 어떤 프로세스가 계속적으로 페이지 부재가 많이 발생하여 프로세스의 실제 처리 시간보다 페이지 교체 시간이 더 많아지는 현상

워킹세트

: 각 프로세스가 많이 참조하는 페이지들의 집합을 주기억장치 공간에 계속 상주하게 하여 빈번한 페이지 교체 현상을 줄이고자 하는 기법

절차적 프로그래밍 언어

ㄱ. 알골(ALGOL) : 재귀호출이 최초로 가능하게 한 절차형 언어

ㄴ. C언어 : 이식성이 높은 언어로 유닉스 운영체제에서 사용하기 위한 언어, 데니스 리치에 의해 개발

ㄷ. 베이직(BASIC) : 교육용으로 개발되어 문법이 쉬움

ㄹ. 포트란(FORTRAN) : 과학계산에서 필수적인 백터, 행렬 계산 기능 등이 내장되어 있는 과학기술 전문 언어

기억장치 접근 속도, 접근 시간 빠른 순 >느린 순

- 레지스터 >캐시 기억장치 >주기억장치 >보조기억장치

구역성(Locality)

: 데닝 교수에 의해 구역성 개념이 증명됨

ㄱ. 시간 구역성(Temporal Locality)

: 프로세스가 실행되면서 하나의 페이지를 일정시간 동안 집중적으로 액세스하는 현상

- 반복(loop), 스택(stack), 부프로그램(sub routine)

ㄴ. 공간 구역성(Spatial Locality)

: 프로세스 실행 시 일정 위치의 페이지를 집중적으로 액세스하는 현상

- 배열, 순차코드, 순차코드의 실행

WAS(웹애플리케이션서버)의 종류

- Tomcat, Weblogic, Jeus 등

스크립트 언어

: 소스 코드를 컴파일하지 않고도 실행할 수 있는 프로그래밍 언어

- 빠르게 배우고 쉽게 작성 가능

- 다른 언어들에 비해 상대적으로 단순한 구문과 의미를 내포

- 시작에서 끝날 때까지 실행되며, 명확한 시작점이 없음

파이썬 언어

- 다양한 플랫폼에서 쓸 수 있고, 라이브러리(모듈)가 풍부

- 유니코드 문자열을 지원하여 다양한 언어의 문자 처리가 가능

- 들여쓰기를 사용하여 블록을 구분

- 다른 언어로 쓰인 모듈들을 연결하는데 사용

리스프(LISP)

: 수학 표기법을 나타내기 위한 목적으로 생성

- 트리 자료구조, 가비지 콜렉션, 동적 자료형과 인터프리터와 같은 개념 제시

- 함수 호출 시 함수 이름 혹은 연산자가 첫 번째로 위치하여 피연산자가 이어 위치

<IV-III 응용 SW 기초기술 활용>

92. 운영체제의 개념 [필기 빈출],[실기 빈출]

운영체제

: 컴퓨터 시스템의 자원들을 효율적으로 관리하며, 사용자가 컴퓨터를 편리하고 효과적으로 사용할 수 있도록 환경을 제공하는 여러 프로그램의 모임

ex) 우리가 흔히 쓰는 윈도우, 혹은 MacOS같은 것들

운영체제의 목적

: 처리 능력 향상, 사용 가능도 향상, 신뢰도 향상, 반환 시간 단축 등

운영체제의 성능을 평가하는 기준

ㄱ. 처리능력 : 일정 시간 내에 시스템이 처리하는 일의 양

ㄴ. 반환 시간(≒응답 시간) : 시스템에 작업을 의뢰한 시간부터 처리가 완료될 때까지 걸리는 시간

ㄷ. 사용 가능도 : 시스템을 사용할 필요가 있을 때 즉시 사용 가능한 정도

ㄹ. 신뢰도 : 주어진 문제를 정확하게 해결하는 정도

운영체제의 기능

- 프로세서(처리기 / Processor / CPU), 기억장치(주기억, 보조기억), 입 출력 장치, 파일 및 정보 등의 자원 관리
- 자원을 효율적으로 관리하기 위해 스케줄링 기능 제공
- 사용자와 시스템 간 편리한 인터페이스 제공
- 각종 하드웨어와 네트워크를 관리 제어
- 데이터 관리, 데이터 및 자원의 공유 기능 제공
- 시스템의 오류 검사 및 복구
- 자원 보호 기능 제공
- 입 출력에 대한 보조 기능 제공

운영체제의 주요 자원 관리

- 프로세스 관리 : 프로세스 스케줄링 및 동기화 관리
- 기억장치 관리 : 프로세스에게 메모리 할당 및 회수 관리
- 주변장치 관리 : 입 출력장치 스케줄링 및 전반적인 관리
- 파일 관리 : 파일의 생성과 삭제, 변경, 유지 등의 관리

운영체제의 종류

- Windows, UNIX, LINUX, MacOS, MS-DOS 등

93. Window [필기 빈출]

Windows

: 마이크로소프트에서 개발한 운영체제

Windows 시스템의 특징

: 그래픽 사용자 인터페이스(GUI, Graphci User Interface) : 키보드로 명령어를 수행하지 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 실행 중 문제가 발생하면 해당 프로그램을 강제 종료시키고 모든 시스템 자원을 반환
- Pnp(Plug and Play) : 하드웨어를 설치할 때 해당 하드웨어를 사용하는데 필요한 시스템 환경을 운영체제가 자동으로 구성해주는 기능
- OLE(Object Linking and Embedding) : 다른 응용 프로그램에서 작성된 문자나 그림 등의 개체 현재 작성 중인 문서에 자유롭게 연결하거나 삽입하여 편집할 수 있는 기능
- Single User 시스템 : 컴퓨터 한대를 한 사람이 독점 사용

93. UNIX / LINUX / MacOS [실기 빈출]

UNIX

: 1960년대 AT&T 벨(Bell) 연구소, MIT, General Electric이 공동 개발한 운영체제

- 시분할 시스템을 위해 설계된 대화식 운영체제
- 소스코드가 개방형 시스템으로 구성되어 있음
- 대부분 C언어로 작성되어 이식성이 높으며 장치, 프로세스 간 호환성이 높음
- 다중 사용자(Multi-User) 및 다중 작업(Multi-Tasking)을 지원
- 트리 구조의 파일 시스템을 가짐

UNIX 시스템의 구성

UNIX의 구성

ㄱ. 커널(Kernel)

- 컴퓨터가 부팅될 때 주기억장치에 적재된 후 상주하면서 실행됨

- 하드웨어를 보호하고 프로그램과 하드웨어 간의 인터페이스 역할을 담당
- 프로세스 관리, 기억장치 관리, 파일 관리, 입출력 관리, 프로세스 간 통신, 데이터 전송 및 변환 등 여러 가지 기능 수행

ㄴ. 쉘(Shell)
- 명령어를 인식하여 수행하는 명령어 해석기
- 시스템과 사용자 간의 인터페이스 담당
- DOS의 COMMAND.COM과 같은 기능 수행
- 주기억장치에 상주하지 않고 명령어가 포함된 파일 형태로 존재
- 보조기억장치에서 교체 처리 가능
- 파이프라인 기능 지원
- 파이프라인 : 둘 이상의 명령을 함께 묶어 처리한 결과를 다른 명령의 입력으로 전환하는 기능
- 입출력 재지정을 통해 입력과 출력의 방향을 변경할 수 있음
- 공용 Shell이나 사용자가 만들 Shell을 사용할 수 있음

ㄷ. 유틸리티(Utility Program)
- 사용자가 작성한 외부 프로그램을 처리
- DOS에서의 외부 명령어에 해당
- 에디터, 컴파일러, 인터프리터, 디버거 등

LINUX
: 1991년 리누스 토발즈가 UNIX를 기반으로 개발한 운영체제
- 프로그램 소스 코드가 무료로 공개되어 있어 사용자가 원하는 기능을 추할 수 있고 다양한 플랫폼에 설치하여 사용이 가능하여 재배포가 가능
- UNIX와 완벽하게 호환
- 대부분의 특징이 UNIX와 동일

MacOS
: 1980년대 애플사가 UNIX를 기반으로 개발한 운영체제
- 애플사에서 생산하는 제품에서만 사용 가능
- 드라이버 설치 및 install / uninstall 과정이 단순

94. 저장장치 관리의 개요

[필기 빈출],[실기 빈출]

초심자를 위한 Tip

배치 전략은 많이 어렵다. 처음 접하는 사람들은 '아 이런 게 있구나' 정도로만 이해하고 넘어가면 된다. 이거 한 문제 틀린다고 합격 못하지 않는다. 실기에서는 중요하니까 필기 때는 기출로 접하고 넘어가고 실기 때 조지도록 하자.

저장장치 계층 구조

: 주기억장치는 각기 자신의 주소를 가지는 워드 또는 바이트들로 구성되어 주소를 이용하여 접근
- 보조기억장치에 있는 프로그램이나 데이터는 CPU가 직접 액세스 할 수 없음
- 보조기억장치에 있는 데이터는 주기억장치에 적재된 후 CPU에 의해 액세스

기억장치의 관리 전략

: Fetch(반입), 배치(Placement), 재배치(Replacement) 전략

반입(Fetch) 전략

: 보조기억장치에 보관중인 데이터를 언제 주기억장치에 적재할 것인지를 결정하는 전략

ㄱ. 요구 반입 : 실행중인 프로그램이 특정 프로그램이나 데이터 등의 참조를 요구할 때 적재

ㄴ. 예상 반입 : 실행중인 프로그램에 의해 참조될 프로그램이나 데이터를 미리 예상하여 적재

배치(Placement) 전략

: 새로 반입되는 데이터를 주기억장치의 어디에 위치시킬 것인지를 결정하는 전략

ㄱ. 최초 적합(First Fit) : 배치가 가능한 크기의 빈 영역 중에서 첫 번째 분할 영역에 배치

ㄴ. 최적 적합(Best Fit) : 배치가 가능한 크기의 빈 영역 중에서 단편화를 가장 적게 남기는 분할 영역에 배치

ㄷ. 최악 적합(Worst Fit) : 배치가 가능한 크기의 빈 영역중에서 단편화를 가장 많이 남기는 분할 영역에 배치

단편화

ㄱ. 내부 단편화 : 배치 후 남은 공간

ㄴ. 외부 단편화 : 배치를 못해 빈 공간으로 남아있는 공간

재배치(Replacement) 전략

: 주기억장치의 모든 영역이 이미 사용 중인 상태에서 새로운 프로그램이나 데이터가 배치하려고 할 때, 이미 사용 중인 영역에서 어느 영역을 교체할 것인지를 결정하는 전략
- FIFO, OPT, LRU, LFU, NUR, SCR 등

95. 주기억장치 할당 기법

주기억장치 할당

: 프로그램이나 데이터를 실행시키기 위해 주기억장치에 어떻게 할당할 것인지에 대한 내용

ㄱ. 연속 할당 기법

: 프로그램을 주 기억장치에 연속으로 할당하는 기법
- 단일 분할 할당 기법 : 오버레이, 스와핑
- 다중 분할 기법 : 고정 분할 할당 기법, 동적 분할 할당 기법

ㄴ. 분산 할당 기법

: 프로그램을 특정 단위의 조각으로 나누어 주기억장치 내에 분산하여 할당하는 기법

- 페이징 기법, 세그먼테이션 기법

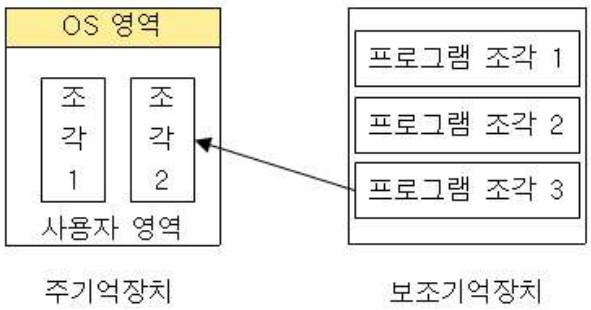
단일 분할 할당 기법

: 주기억장치를 운영체제 영역과 사용자 영역으로 나누어 한 순간에는 오직 한 명의 사용자만이 주기억장치의 사용자 영역을 사용하는 기법

ㄱ. 오버레이 기법

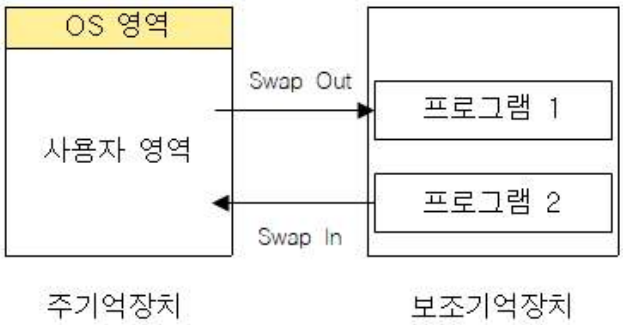
: 주기억장치보다 큰 사용자 프로그램을 실행하기 위한 기법

- 보조기억장치에 저장된 하나의 프로그램을 여러개의 조각으로 분할한 후 필요한 조각을 차례로 주기억장치에 적재하여 프로그램을 실행



ㄴ. 스와핑 기법

: 하나의 프로그램 전체를 주기억장치에 할당하여 사용하다 필요에 따라 다른 프로그램과 교체하는 기법



다중 분할 할당 기법

ㄱ. 고정 분할 할당 기법 : 프로그램에 할당하기 전에 운영체제가 주기억장치의 사용자 영역을 여러 개의 고정된 크기로 분할하고 준비상태 큐에서 준비 중인 프로그램을 각 영역에서 할당하여 수행하는 기법

ㄴ. 가변 분할 할당 기법 : 미리 주기억장치에 분할해 놓는 것이 아닌 프로그램을 주기억장치에 적재하면서 필요한 만큼의 크기로 영역을 분할

96. 가상 기억장치 구현 기법 / 페이지 교체 알고리즘

[필기 빈출],[실기 빈출]

초심자를 위한 Tip

이 파트도 많이 어렵다. 처음 접하는 사람들은 '아 이런 게 있구나' 정도로만 이해하고 넘어가면 된다. 이거 한 문제 틀린다고 합격 못하지 않는다. 실기에서는 중요하니까 필기 때는 기출로 접하고 넘어가고 실기 때 조지도록 하자.

가상 기억장치

: 보조기억장치의 일부를 주기억장치처럼 사용하는 것

- 용량이 작은 주기억장치를 마치 큰 용량을 가진 것처럼 사용할 수 있음

- 프로그램을 여러 작은 블록 단위로 나누어서 가상 기억장치에 보관해 놓고, 프로그램 실행 시 요구되는 블록만 주기억장치에 불연속적으로 할당하여 처리

- 주기억장치의 이용률과 다중 프로그래밍의 효율 상승

- 가상기억장치에 저장된 프로그램을 실행하려면 가상 기억장치의 주소를 주기억장치의 주소로 바꾸는 주소 변환(Mapping) 작업이 필요

- 연속 할당 방식에서 발생할 수 있는 단편화를 해결할 수 있음

페이징 기법

: 가상기억장치의 보관되어 있는 프로그램과 주기억장치의 영역을 동일한 크기로 나눈 후 나눠진 프로그램을 동일하게 나눠진 주기억장치의 영역에 적재시켜 실행하는 기법

- 프로그램을 일정한 크기로 나눈 크기를 페이지라 하고 페이지 크기로 일정하게 나누어진 주기억장치의 단위를 페이지 프레임이라고 함

- 외부 단편화는 발생하지 않으나 내부 단편화는 발생할 수 있음

- Mapping 작업을 위해서 페이지 맵 테이블이 필요함

세그먼트 기법

: 가상기억장치에 보관되어 있는 프로그램을 다양한 크기의 논리적인 단위로 나눈 후 주기억장치에 적재시켜 실행시키는 방법

- 프로그램을 논리적인 크기로 나눈 단위를 세그먼트라고 함

- 내부 단편화는 발생하지 않으나 외부 단편화는 발생할 수 있음

- Mapping 작업을 위해서 세그먼트 맵 테이블이 필요함

- 세그먼트가 주기억장치에 적재될 때 다른 세그먼트에게 할당된 영역을 침범할 수 없으며 이를 위해 기억장치 보호키가 필요

페이지 교체 알고리즘

: 페이지 부재가 발생했을 때 가상 기억장치의 필요한 페이지를 주기억장치에 적재해야 하는데 이럴 경우 주기억장치의 모든 페이지 프레임이 사용 중이면 어떤 페이지 프레임을 선택하여 교체할 것인지 결정하는 기법

종류 : OPT, FIFO, LRU,LFU 등이 있음

페이지 교체 알고리즘 세부 참고 내용(실기때 참고)

->

https://velog.io/@codemcd/%EC%9A%B4%EC%98%81%EC%B2%B4%EC%A0%9COS-16.-%ED%8E%98%EC%9D%B4%EC%A7%80-%EA%B5%90%EC%B2%B4-%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98

97. 가상 기억장치 기타 관리 사항

페이지 크기에 따른 특징

ㄱ. 페이지 크기가 작을 경우

- 단편화와 주기억장치로 이동하는 시간 감소
- 불필요한 내용이 적재될 확률이 낮아 워킹 셋이 효율적으로 유지
- Locality에 더욱 일치하여 기억장치에 효율 상승
- 페이지 맵 테이블 크기가 커지므로 매핑 속도가 늦어짐
- 디스크 접근 횟수가 많아져 전체적인 입출력 시간이 늘어남

ㄴ. 페이지 크기가 클 경우

- 단편화와 주기억장치로 이동하는 시간이 증가
- 프로세스 수행에 불필요한 내용까지 적재될 수 있음
- 페이지 맵 테이블 크기가 작아지므로 매핑 속도가 빨라짐
- 디스크 접근 횟수가 줄어들어 전체적인 입출력 시간이 줄어듦

Locality

: 프로세스가 실행되는 동안 주기억장치를 참조할 때 일부 페이지만 집중적으로 참조하는 성질이 있다는 이론

- 스래싱을 방지하기 위한 워킹 셋 이론의 기반
- 프로세스가 집중적으로 사용하는 페이지를 알아내는 방법

시간 구역성

: 프로세스가 실행되면서 하나의 페이지를 일정 시간 동안 집중적으로 액세스 하는 현상

- 시간 구역성이 이루어지는 기억 장소 : 반복, 스택, 부프로그램, 1씩 증감, 집계에 사용되는 변수

공간 구역성

: 프로세스 실행 시 일정 위치의 페이지를 집중적으로 액세스하는 현상

- 공간 구역성이 이루어지는 기억 장소 : 배열 순회, 순차적 코드의 실행, 프로그래머들이 관련된 변수들을 서로 근처에 선언하여 할당되는 기억 장소

워킹 셋

: 프로세스가 일정 시간 동안 자주 참조하는 페이지들의 집합

- 자주 참조되는 워킹 셋을 주기억장치에 상주시켜 페이지 부재 및 페이지 교체 현상이 줄어들어 프로세스의 기억장치 사용이 안정됨
- 워킹 셋은 시간에 따라 변화

페이지 부재 빈도 방식

- 페이지 부재 빈도는 페이지 부재가 일어나는 횟수
- 페이지 부재율에 따라 주기억장치에 있는 페이지 프레임 수를 조정하여 적정 수준으로 유지하는 방식
- 운영체제는 프로세스 실행 초기에 임의의 페이지 프레임 할당 후 페이지 부재율에 따라 프레임을 할당하거나 회수

프리 페이징

: 처음의 과도한 페이지 부재를 방지하기 위해 필요할 것 같은 모든 페이지를 한꺼번에 페이지 프레임에 적재하는 기법

스래싱

: 프로세스의 처리 시간보다 페이지 교체에 소요되는 시간이 더 많아지는 현상

- 다중 프로그래밍 시스템이나 가상 기억장치를 사용하는 시스템에서 하나의 프로세스 수행 중 자주 페이지 부재가 발생하면서 나타나는 현상
- 전체 프로세스 성능이 저하됨
- 다중 프로그래밍의 정도가 높아짐에 따라 CPU의 이용률은 어느 특정 시점까지는 높아지지만 다중 프로그래밍의 정도가 더욱 커지면 스래싱이 나타나고 CPU의 이용률은 급격히 감소됨

스래싱 현상 방지 방법

- 다중 프로그래밍의 정도를 적정 수준으로 유지
- 페이지 부재 빈도를 조절하여 사용
- 워킹 셋 유지
- 부족한 자원 증설, 일부 프로세스 중단
- CPU 성능에 대한 자료의 지속적 관리 및 분석으로 임계치를 예상하여 운영

98. 프로세스의 개요

프로세스

: 프로세서에 의해 처리되는 사용자 프로그램이나 시스템 프로그램을 의미

- 실행 중인 프로그램을 의미하며 작업 혹은 태스크라고도 함

프로세스의 또 다른 형태

- PCB를 가진 프로그램
- 실기억 장치에 저장된 프로그램
- 디스패치가 가능한 단위
- 프로시저(부 프로그램)가 활동 중인 것
- 비동기적 행위(다수의 프로세스가 서로 독립적으로 실행)를 일으키는 주체
- 지정된 결과를 얻기 위한 일련의 계통적 동작
- 목적 또는 결과에 따라 발생되는 사건들의 과정
- 운영체제가 관리하는 실행 단위

프로세스 상태 전이 관련 용어

- Dispatch : 준비 상태에 대기하고 있는 프로세스 중 하나가 프로세스를 할당받아 실행 상태로 전이되는 과정
- Wake up : 입출력 작업이 완료되어 프로세스가 대기 상태

에서 준비 상태로 전이되는 과정

- Spooling : 입출력 장치의 공유 및 상대적으로 느린 입출력 장치의 처리 속도를 보완하기 위해 입출력할 데이터를 직접 장치에 보내지 않고 나중에 한꺼번에 입출력하기 위해 디스크에 저장하는 과정
- 교통량 제어기 : 프로세스의 상태에 대한 조사와 통보를 담당

스레드(Thread)

- 프로세스 내에서의 작업 단위로서 시스템의 여러 자원을 할당받아 실행하는 프로세스 단위
- 프로세스의 일부 특성을 가지고 있어 경량 프로세스라고도 함
- 동일 프로세스 환경에서 독립적인 다중 수행 가능

스레드의 분류

- 사용자 수준의 스레드 : 사용자가 만든 라이브러리를 사용하여 운용하기 때문에 속도는 빠르지만 구현이 어려움
- 커널 수준의 스레드 : 운영체제의 커널에 의해 스레드를 운영하기 때문에 속도는 느리지만 구현이 쉬움

스레드 사용의 장점

- 하나의 프로세스를 여러 개의 스레드로 생성하여 병행성을 증진할 수 있음
- H/W, O/S의 성능과 응용 프로그램의 처리율을 향상시킬 수 있음
- 응용프로그램의 응답 시간을 단축시킬 수 있음
- 프로세스들 간의 통신이 향상
- 공통적으로 접근 가능한 기억장치를 통해 효율적으로 통신

99. 스케줄링

스케줄링

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

- 프로세스가 생성되어 완료될 때까지 여러 종류의 스케줄링 과정을 거침
- 그 종류에는 장기, 중기, 단기 스케줄링이 있음

문맥 교환

: 하나의 프로세스에서 다른 프로세스로 CPU가 할당되는 과정에서 발생되는 것

- 새로운 프로세스에 CPU를 할당하기 위해 현재 CPU가 할당된 프로세스의 상태 정보 저장
- 새로운 프로세스의 상태 정보를 설정한 후 CPU를 할당하여 실행하도록 하는 작업

스케줄링의 목적

- CPU나 자원을 효율적으로 사용하기 위한 정책
- 공정성 : 모든 프로세스에게 공정하게 할당
- 처리율(처리량) 증가 : 단위 시간당 프로세스를 처리하는 비율 혹은 양을 증가
- CPU 이용률 증가 : CPU의 낭비 시간을 줄이고 CPU가 순수 프로세스를 실행하는 데 사용되는 시간 비율 증가
- 우선순위 제도 : 우선순위가 높은 프로세스를 먼저 실행
- 오버헤드 최소화 : 어떤 처리하기 위해 들어가는 간접적인 처리시간이나 메모리를 최소화
- 응답 시간 최소화 : 작업을 지시하고 반응하기 시작하는 시간을 최소화
- 반환 시간 최소화 : 프로세스를 제출한 시간부터 실행이 완료될 때까지 걸리는 시간 최소화
- 대기 시간 최소화 : 프로세스가 준비상태 큐에서 대기하는 시간 최소화
- 균형 있는 자원의 사용
- 무한 연기 회피

프로세스 스케줄링의 기법

[필기 빈출],[실기 빈출]

초심자를 위한 Tip

이 부분도 많이 어렵다. 처음 접하는 사람들은 '아 이런 게 있구나' 정도로만 이해하고 넘어가면 된다. 이거 한 문제 틀린다고 합격 못하지 않는다. 실기에서는 중요하니까 필기 때는 기출로 접하고 넘어가고 실기 때 조지도록 하자.

ㄱ. 비선점 스케줄링

- 강제 종료 불가능
- 이미 할당된 CPU를 다른 프로세스가 강제로 빼앗아 사용할 수 없음
- 프로세스가 CPU를 할당받으면 완료될 때 까지 사용
- 프로세스 응답 시간 예측이 용이
- 일괄 처리 방식에 적합
- 중요한(짧은) 작업이 중요하지 않은(긴) 작업을 기다리는 경우가 발생할 수 있음

ex) FCFS, SJF, 우선순위, HRN, 기한부 등의 알고리즘

ㄴ. 선점 스케줄링

- 강제 종료 가능
- 우선순위가 높은 다른 프로세스가 CPU를 강제로 빼앗아 사용할 수 있음
- 우선순위가 높은 프로세스를 빠르게 처리할 수 있음
- 빠른 응답 시간을 요구하는 대화식 시분할 시스템에 사용
- 많은 오버헤드를 초래함
- 선점이 가능하도록 일정 시간 배당에 대한 인터럽트용 타이머 클록이 필요
- Round Robin, SRT, 선점 우선순위, 다단계 큐, 다단계 피드백 큐 등의 알고리즘

100. 환경변수

환경변수

: 시스템 소프트웨어의 동작에 영향을 미치는 동적인 값들의 모임

Windows 환경 변수

- Windows에서 환경 변수를 명령어나 스크립트에서 사용하기 위해서는 변수명 앞뒤에 %를 붙여야 함

UNIX / LINUX 환경 변수

- UNIX나 LINUX에서 환경 변수를 사용하기 위해서는 변수명 앞에 $를 붙여야 함

101. 운영체제 기본 명령어[필기 빈출],[실기 빈출]

운영체제 기본 명령어

- CLI(Command Line Interface) : 키보드로 명령어를 입력하여 작업을 수행하는 인터페이스
- GUI(Graphic User Interface) : 마우스로 아이콘이나 메뉴를 선택하여 작업을 수행하는 인터페이스

Windows &UNIX / LINUX 기본 명령어

Windows	UNIX / LINUX	기능
DIR	ls	파일 목록 표시
COPY	cp	파일 복사
TYPE	cat	파일 내용 표시
REN	mv	파일 이름 변경
MOVE		파일 이동
MD	mkdir	디렉토리 생성
CD	chdir	디렉토리 위치 변경
CLS	clear	화면 내용 지움
ATTRIB	chmod	파일 속성 변경
FIND	find	파일 찾기
CHKDSK		디스크 상태 점검
FORMAT		디스크 표면을 섹터로 나누어 초기화
	chown	소유자 변경
	exec	새로운 프로세스 수행
	fork	새로운 프로세스 생성(Tree 구조)
	fsck	파일 시스템 검사·보수
	getpid	자신의 프로세스 아이디를 얻음
	mount	파일 시스템을 마운팅

초심자를 위한 Tip

해당 명령어의 경우 윈도우 환경에서는 윈도우 검색창에 cmd 라고 입력하면 나타나는 검은 화면에서 명령어들을 입력하면 테스트 할 수 있고 유닉스/리눅스 같은 경우에는 현업에 갔을 경우 정말 위에 있는 명령어들을 엄청나게 많이 쓰게 된다.

102. 인터넷

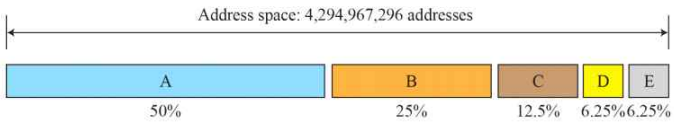
인터넷

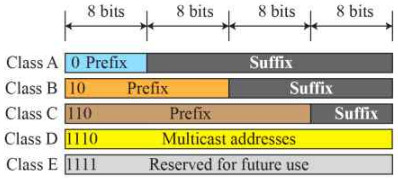
: TCP/IP 프로토콜을 기반으로 전 세계 수많은 컴퓨터와 네트워크들이 연결된 광범위한 컴퓨터 통신망

- 미 국방성의 ARPANET에서 시작
- 유닉스 운영체제를 기반으로 함
- 인터넷에 연결된 컴퓨터는 고유한 IP 주소를 가짐
- 인터넷을 구성하기 위해서는 브리지, 라우터, 게이트웨이가 사용됨
- 백본 : 네트워크를 연결하여 중추적 역할을 하는 네트워크로 인터넷의 주가 되는 기간망을 일컫는 용어

IP주소(Internet Protocol Address) [필기 빈출],[실기 빈출]

- 인터넷에 연결된 컴퓨터를 구분하기 위한 고유한 주소
- 8비트 씩 4부분으로 구성됨(IPv4)





Class	Prefixes	First byte
A	$n = 8$ bits	0 to 127
B	$n = 16$ bits	128 to 191
C	$n = 24$ bits	192 to 223
D	Not applicable	224 to 239
E	Not applicable	240 to 255

서브네팅

- 할당된 네트워크 주소를 다시 여러 개의 작은 네트워크로 나누어 사용
- IPv4의 주소 부족 문제를 해결하기 위한 방법
- 서브넷 마스크 : 네트워크 주소와 호스트 주소를 구분하기 위한 비트

IPv6 [필기 빈출],[실기 빈출]

: 시간이 지나면서 인터넷 사용자가 많아지면서 IPv4의 주소가 부족해지게 되어 IPv4의 주소 부족 문제를 해결하기 위해 개발됨

- 128비트의 긴 주소를 사용하여 주소 부족 문제를 해결할 수 있고, 자료 전송 속도가 빠름
- 인증성, 기밀성, 데이터 무결성의 지원으로 보안 문제 해결 가능

ㄱ. 인증성 : 사용자의 식별과 접근 권한 검증

ㄴ. 기밀성 : 시스템 내의 정보와 자료는 인가된 사용자에게만 접근 허용

ㄷ. 무결성 : 시스템 내의 정보는 인가된 사용자만 수정 가능

- Traffic Class, Flow Label을 이용하여 등급별, 서비스별로 패킷을 구분할 수 있어 품질 보장이 용이
- Traffic Class : IPv6 패킷의 클래스나 우선순위를 나타내는 필드
- Flow Label : 네트워크 상에서 패킷들의 흐름에 대한 특성을 나타내는 필드

도메인 네임

IP 주소 사람이 이해하기 쉬운 문자 형태로 표현한 것

ex)

http://192.168.4.10/(IP주소) -> www.google.com (도메인 네임)

도메인 네임의 구성

EX) : www.naver.com

ㄱ. www -> world wide web의 약어 (인터넷망)

ㄴ. naver -> 해당 페이지의 이름 (네이버, 구글 등등)

ㄷ. com -> 해당 국가 (국가에 따라 바뀜 일본은 jp)

- 도메인 네임을 IP 주소로 변환하는 역할을 하는 시스템을 DNS라고 하며 이런 역할을 하는 서버를 DNS 서버라고 함

: 시험을 위해 쉽게 풀어쓰긴 했지만 조금 더 자세히 알고 싶다면 해당 링크 참조

=> https://ithub.tistory.com/337

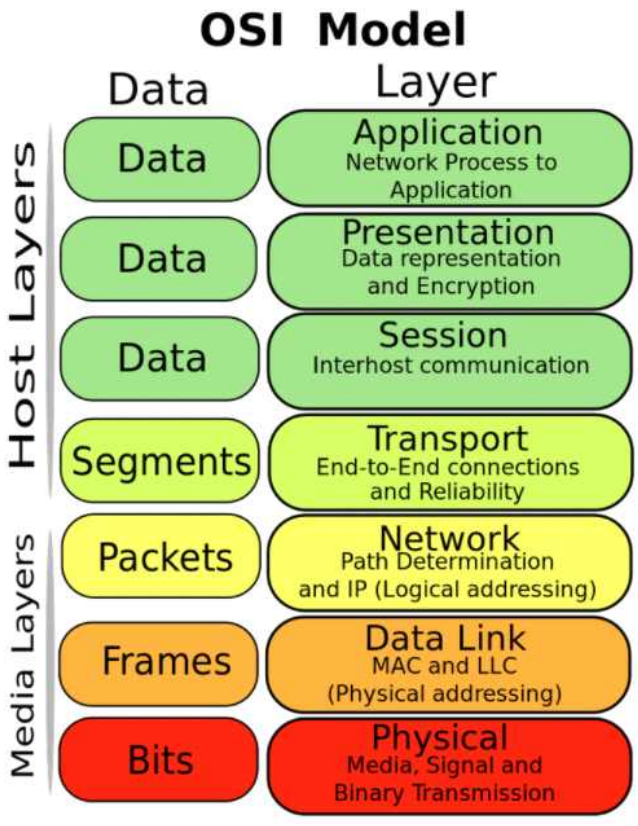
103. OSI 7 Layer [필기 빈출],[실기 빈출]

매우 중요한 파트다 확실하게 이해하고 알아두면 필기 실기 뽕을 뽑는 파트니 반드시 모두 이해하고 부족하면 검색해서라도 이해하고 암기하고 넘어가자

OSI(Open System Interconnetion)7 Layer 참조 모델

: 다른 시스템 간의 원활한 통신을 위해 ISO(국제표준화기구)에서 제안한 통신 규약(프로토콜)으로 컴퓨터 네트워크 프로토콜을 기능별로 나누어 각 계층으로 설명한 것

(아래 그림을 보면 맨 아래의 물리단계가 1단계~맨 위의 애플리케이션 단계 7단계 까지 7계의 계층으로 구분)



네트워크를 계층 구조로 나눈 이유

ㄱ. 데이터의 흐름이 한눈에 보인다.

ㄴ. 하나의 문제를 7개의 작은 문제로 나누기 때문에 문제 해결이 쉽다.

ㄷ. 각 계층마다 사용하는 장비가 다른데, 표준화를 통해 여러 회사의 네트워크 장비를 사용하더라도 이상 없이 작동할 수 있게 된다.

초심자를 위한 Tip

- OSI 7 Layer를 설명할 때 가장 자주 예시로 드는 것이 아마 게임을 할 때를 예로 많이 드는 것 같다. 만약 여러분들이 PC방에서 리그오브레전드 같은 게임에서 랭크게임을 하고 있는데 갑자기 연결이 끊겨버렸다. 이 경우 어느 부분이 문제인지 해결을 해야 할 것이다.(~~어이 당신! 일단 키보드 샷컨부터 멈춰!!!~~) 만약 PC방 안의 모든 PC가 문제가 있다면 라우터의 문제(3계층 네트워크 계층)이거나 혹은 광랜을 제공하는 회사의 회선 문제(1계층 물리 계층)일 수도 있다. 하지만 만약 본인의 PC만 문제가 있고 리그오브레전드 소프트웨어에 문제가 있다면(7계층 어플리케이션 계층) 그 아래의 계층에는 문제가 없다고 판단하고 다른 장비나 소프트웨어를 건드리지 않을 수 있다. 이처럼 네트워크를 계층화 한다면 데이터의 흐름을 한 눈에 확인 할 수 있고 문제 해결을 훨씬 손 쉽게 할 수 있다.

OSI 모델의 계층

ㄱ. 1 물리 계층

: 전송에 필요한 두 장치 간의 실제 접속과 절단 등 기계적, 전기적, 기능적, 절차적 특성에 대한 규칙을 정의

- 물리적 전송 매체와 신호 방식을 정의
- RS-232C, X.21 등의 표준
- 관련 장비 : 리피터, 허브
- 데이터 단위 : 비트

ㄴ. 2 데이터 링크 계층

: 두 개의 인접 개방 시스템들 간의 신뢰성 있고 효율적인 정보 전송을 할 수 있도록 함

- 주요 기능 : 흐름 제어, 프레임 동기화, 오류 제어, 순서 제어
- HDLC, LAPB, LLC, MAC, LAPD, PPP 등의 표준
- 관련 장비 : 브리지
- 데이터 단위 : 프레임

ㄷ. 3 네트워크 계층

: 개방 시스템들 간의 네트워크 연결을 관리하고 데이터의 교환 및 중계 기능

- 네트워크 연결 설정, 유지, 해제
- 경로 설정, 데이터 교환 및 중계, 트래픽 제어, 패킷 정보 전송 수행
- X.25, IP 등의 표준
- 관련 장비 : 라우터
- 데이터 단위 : 패킷

ㄹ. 4 전송 계층

: 논리적 안정과 균일한 데이터 전송 서비스를 제공

- 종단 시스템 간의 투명한 데이터 전송을 가능하게 함
- 종단 시스템 간의 전송 연결 설정, 데이터 전송, 연결 해제 기능
- 주소 설정, 다중화, 오류 제어, 흐름 제어
- TCP, UDP 등의 표준
- 관련 장비 : 게이트웨이
- 데이터 단위 : 세그먼트

ㅁ. 5 세션 계층

: 송수신 측 간의 관련성을 유지하고 대화 제어를 담당

- 동기점 : 송수신 측간의 대화 동기를 위해 전송하는 정보를 일정한 부분에 두어 정보의 수신 상태를 체크하는 포인트
- 데이터 단위 : 메시지

ㅂ. 6 표현 계층

: 데이터를 응용 계층, 세션 계층에 보내기 전에 계층에 맞게 변환

- 서로 다른 데이터 표현 형태를 갖는 시스템 간 상호 접속을 위해 필요한 계층
- 코드 변환, 데이터 암호화, 데이터 압축, 구문 검색, 정보 형식 변환, 문맥 관리 기능
- 데이터 단위 : 메시지

ㅅ. 7 애플리케이션 계층

: 사용자가 OSI 환경에 접속할 수 있도록 서비스 제공

- 프로세스 간의 정보 교환, 전자 사서함, 가상 터미널 등의 서비스 제공

104. 네트워크 관련 장비 [필기 빈출],[실기 빈출]

네트워크 인터페이스 카드(Network Interface Card)

: 컴퓨터를 연결하는 장치로 정보 전송 시 정보가 케이블을 통해 전송될 수 있도록 정보 형태를 변경

- 이더넷 카드 혹은 네트워크 어댑터라고도 함

허브(hub)

: 가까운 거리의 컴퓨터를 연결하는 장치

- 각 회선을 통합적으로 관리하며 신호 증폭 기능을 하는 리피터의 역할도 포함

ㄱ. 더미 허브

- 네트워크에 흐르는 모든 데이터를 단순히 연결만 함
- LAN이 보유한 대역폭을 나누어 제공

ㄴ. 스위칭 허브

- 네트워크상에 흐르는 데이터의 유무 및 흐름을 제어하여 각각의 노드가 허브의 최대 대역폭을 사용할 수 있는 지능형 허브

리피터(Repeater)

: 전송되는 신호가 원래의 형태와 다르게 왜곡되거나 약해질 경우 원래의 신호 형태로 재생하여 다시 전송하는 역할

- 근접한 네트워크 사이에 신호를 전송
- 전송 거리의 연장 또는 배선의 자유도를 높이는 용도

브리지(Bridge)

: LAN과 LAN을 연결하거나 LAN안에서 컴퓨터 그룹을 연결

- 데이터 링크 계층 중 MAC 계층에서 사용되므로 MAC 브리지라고도 함
- 현재는 스위치에 밀려 거의 사용되지 않는다

스위치(Switch)

- LAN과 LAN을 연결하여 훨씬 더 큰 LAN을 만드는 장치
- 포트마다 각기 다른 전송속도를 지원하도록 제어할 수 있음
- 수십 ~ 수백 개의 포트를 지원

라우터(Router)

- LAN과 LAN의 연결 기능에 데이터 전송의 최적 경로를 선택할 수 있는 기능을 추가
- 서로 다른 LAN이나 LAN과 WAN의 연결도 수행
- 접속 가능한 경로에 대한 정보를 Routing Table에 저장하여 보관

게이트웨이(Gateway)

: 전 계층의 프로토콜 구조가 다른 네트워크의 연결을 수행

- LAN에서 다른 네트워크에 데이터를 송수신하는 출입구 역할을 함

105. 프로토콜의 개념 [필기 빈출],[실기 빈출]

프로토콜

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

프로토콜의 기본 요소

- 구문 : 전송하고자 하는 데이터의 형식, 부호화, 신호 레벨 등을 규정
- 의미 : 두 기기 간의 효율적이고 정확한 정보 전송을 위한 협조 사항과 오류 관리를 위한 제어 정보를 규정
- 시간 : 두 기기 간의 통신 속도, 메시지의 순서 제어 등을 규정

프로토콜의 기능

ㄱ. 단편화와 재결합

a. 단편화 : 송신 측에서 전송할 데이터를 전송에 알맞은 작은 크기의 블록으로 자르는 작업

b. 재결합 : 수신 측에서 수신한 단편화된 데이터를 다시 모으는 작업

ㄴ. 캡슐화

- 단편화 된 데이터에 주소, 오류 검출 코드, 프로토콜 제어 정보를 부가하는 것

ㄷ. 흐름 제어

- 수신 측에서 송신 측의 데이터 전송 속도나 전송 량을 제어할 수 있는 기능
- 정지-대기 방식이나 슬라이딩 윈도우 방식을 이용

ㄹ. 오류 제어

- 전송 중에 발생하는 오류를 검출하고 정정하여 데이터나 제어 정보의 파손에 대비하는 기능

ㅁ. 동기화

- 송수신 측이 같은 상태를 유지하도록 타이밍을 맞추는 기능

ㅂ. 순서 제어

- 전송되는 데이터 블록에 전송 순서를 부여하여 연결 위주의 데이터 전송 방식에 사용
- 흐름 제어 및 오류 제어를 용이하게 함

ㅅ. 주소 지정

- 데이터가 목적지까지 정확하게 전송될 수 있도록 목적지 이름, 주소, 경로를 부여하는 기능

ㅇ. 다중화

- 한 개의 통신 회선을 여러 가입자들이 동시에 사용하도록 하는 기능

ㅈ. 경로 제어

- 송수신 측간의 송신 경로 중에서 최적의 패킷 교환 경로를 설정하는 기능

ㅊ. 전송 서비스

- 전송하려는 데이터가 사용하도록 하는 별도의 부가 서비스

106. TCP/IP [필기 빈출],[실기 빈출]

TCP/IP

: 인터넷에 연결된 서로 다른 기종의 컴퓨터들이 데이터를 주고받을 수 있도록 하는 표준 프로토콜

- 1960년대 말 ARPA에서 개발하여 ARPANET에서 사용하기 시작
- UNIX의 기본 프로토콜로 사용되었다가 현재는 인터넷 범용 프로토콜로 사용
- TCP(Transmission Control Protocol)과 IP(Internet Protocol)이 결합

응용 계층의 주요 프로토콜

ㄱ. FTP : 원격 파일 전송 프로토콜

ㄴ. SMTP : 전자 우편 교환 서비스

ㄷ. TELNET : 원격 접속 서비스

- 가상 터미널 기능 수행

ㄹ. SNMP : TCP/IP의 관리 프로토콜

- 네트워크 기기의 네트워크 정보를 네트워크 관리 시스템에 보내는 데 사용되는 프로토콜

ㅁ. DNS : 도메인 네임을 IP주소로 매핑하는 시스템

ㅂ. HTTP : WWW에서 HTML을 송수신하기 위한 표준 프로토콜

전송 계층의 주요 프로토콜 [실기 빈출]

ㄱ. TCP : 양방향 연결형 서비스 제공

- 가상 회선 연결 형태의 서비스 제공
- 순서 제어, 오류 제어, 흐름 제어 기능을 함
- 스트림 위주의 패킷 단위 전달

ㄴ. UDP : 비연결형 서비스 제공

- 실시간 전송에 유리하며, 신뢰성보다는 속도가 중요시되는 네트워크에서 사용

ㄷ. RTCP

- 패킷의 전송 품질을 제어하기 위한 제어 프로토콜
- 세션에 참여한 각 참여자들에게 주기적으로 제어 정보를 전송
- 데이터 전송을 모니터링하고 최소한의 제어와 인증 기능만을 제공
- 패킷은 항상 32비트의 경계로 끝남

인터넷 계층의 주요 프로토콜

ㄱ. IP : 전송할 데이터에 주소를 지정하고 경로를 설정

ㄴ. ICMP : IP와 조합하여 통신 중에 발생하는 오류의 처리와 전송 경로 변경 등을 위한 제어 메시지를 관리

- 헤더는 8Byte로 구성

ㄷ. IGMP : 멀티캐스트를 지원하는 호스트나 라우터 사이에서 멀티캐스트 그룹 유지를 위해 사용

ㄹ. ARP : IP 주소를 MAC Address로 변환 (논리 주소 → 물리 주소)

ㅁ. RARP : ARP의 반대로 MAC Address를 IP 주소로 변환 (물리 주소 → 논리 주소)

네트워크 액세스 계층의 주요 프로토콜 [필기 빈출]

(그러나 이 문제는 필기에서 나오면 걍 버려도 된다. 하나 맞추자고 암기하기에는 너무 리스크가 크다)

ㄱ. IEEE 802 : LAN을 위한 표준 프로토콜

- IEEE 802.3(Ethernet) : CSMA/CD 방식의 LAN
- IEEE 802.4 : 토큰 박스
- IEEE 802.5 : 토큰링
- IEEE 802.11 : 무선 LAN

ㄴ. HDLC : 비트 위주의 데이터 링크 제어 프로토콜

ㄷ. X.25 : 패킷 교환망을 통한 DTE와 DCE 간의 인터페이스를 제공하는 프로토콜

- DTE(신호 단말 장치), DCE(신호 통신 장비)

ㄹ. RS-232C : 공중전화 교환망을 통한 DTE와 DCE 간의 인터페이스를 제공하는 프로토콜

V. 정보시스템 구축관리 (대단원5/5)

<V-I 소프트웨어 개발 방법론 활용>

107. 소프트웨어 개발 방법론

[필기 빈출],[실기 빈출]

소프트웨어 개발 방법론

: 소프트웨어 개발, 유지보수에 필요한 수행 방법과 효율적으로 수행하려는 과정에서 필요한 기법 및 도구를 정리하여 표준화

구조적 방법론

: 정형화된 분석 절차에 따라 사용자 요구사항을 파악하여 문서화하는 처리 중심의 방법론

- Divide and Conquer 원리 적용

순서

- 타당성 검토 -> 계획 -> 요구사항 분석-> 설계-> 구현-> 시험 -> 운용/유지보수

정보공학 방법론

: 정보 시스템 개발을 위해 계획, 분석, 설계, 구축에 정형화된 기법들을 상호 연관성 있게 통합 및 적용하는 자료 중심의 방법론

- 대규모 정보시스템 구축에 적합

순서

-정보 전략 계획수립 -> 업무 영역 분석-> 업무 시스템 설계 -> 업무 시스템 구축

객체지향 방법론

- 기계의 부품을 조립하듯이 객체들을 조립하여 소프트웨어를 구현하는 방법론
- 구조적 기법의 해결책으로 채택

순서

-요구사항 분석-> 설계-> 구현->테스트 및 검증 ->인도

컴포넌트 기반(CBD) 방법론

- 컴포넌트를 조합하여 새로운 애플리케이션을 만드는 방법론
- 컴포넌트의 재사용이 가능하여 시간, 노력, 비용을 절감하고 품질을 높임

순서

-개발 준비-> 분석-> 설계 ->구현 ->테스트 -> 전개-> 인도

애자일 방법론

- 고객의 요구사항 변화에 빠르고 유연하게 대응하도록 일정한 주기를 반복하면서 개발 과정을 진행하는 방법론

순서

사용자 요구사항 -> 계획-> 개발 -> 승인 테스트

제품 계열 방법론

- 특정 제품에 적용하고 싶은 공통된 기능을 정의하여 개발하는 방법론
- 임베디드 소프트웨어 개발에 적합

108. 비용 산정 기법 [필기 빈출],[실기 빈출]

소프트웨어 비용 산정

: 소프트웨어의 개발 규모를 소요되는 인원, 자원, 기간 등으로 확인하여 실행 가능한 계획을 수립하기 위해 필요한 비용을 산정하는 것

소프트웨어 비용 결정 요소

ㄱ. 프로젝트 요소 : 제품 복잡도, 시스템 크기, 요구되는 신뢰도

ㄴ. 자원 요소 : 인적 자원, 하드웨어 자원, 소프트웨어 자원

ㄷ. 생산성 요소 : 개발자 능력, 개발 기간

하향식 산정 기법

: 과거의 유사한 경험을 바탕으로 전문 지식이 많은 개발자들이 참여한 회의를 통해 비용을 산정

ㄱ. 전문가 감정 기법

- 조직 내 경험이 많은 두 명 이상의 전문가에게 비용 산정을 의뢰
- 진행했던 유사한 프로젝트와 진행할 새로운 프로젝트 간 새로운 요소가 있을 수 있고 경험이 없을 수 있음

ㄴ. 델파이 기법

- 많은 전문가의 여러 의견을 종합하여 산정
- 한 명의 조정자와 여러 전문가로 구성되어 객관적임

상향식 산정 기법

: 프로젝트의 세부적인 작업 단위별로 비용을 산정 후 집계하여 전체 비용을 산정

ㄱ. LOC(source Line Of Code) 기법

: 소프트웨어의 각 기능의 원시 코드 라인 수로 예측치를 구하고 비용을 산정하는 기법

- 노력(인월) = 개발 기간 x 투입 인원 = LOC / 1인당 월평균 생산 코드 라인 수
- 개발 비용 = 노력(인월) x 단위 비용(1인당 월평균 인건비)
- 개발 기간 = 노력(인월) / 투입 인원
- 생산성 = LOC / 노력(인월)

ㄴ. 개발 단계별 인월수 기법

: 각 기능을 구현시키는 데 필요한 노력을 생명 주기의 각 단계별로 산정

109. 수학적 산정 기법 [필기 빈출],[실기 빈출]

수학적 산정 기법

- 상향식 선정 기법
- 개발 비용 산정의 자동화를 목표로 함
- 경험적 / 실험적 추정 모형이라고도 함
- 과거 유사한 프로젝트를 기반으로 공식을 유도

COCOMO 모형

: 보헴이 제안하였으며 LOC에 의한 비용 산정 기법

개발 유형

: 소프트웨어의 복잡도 또는 원시 프로그램의 규모에 따라 분류

ㄱ. 조직형(Organinc Mode) : 기관 내부에서 개발된 중·소규모의 소프트웨어로 5만 라인 이하의 소프트웨어를 개발하는 유형

ㄴ. 반분리형(Semi-Detached Mode) : 트랜잭션 처리 시스템이나 운영체제, 데이터베이스 관리 시스템 등 30만 라인 이하의 소프트웨어를 개발하는 유형

ㄷ. 내장형(Embedded Mode) : 최대형 규모의 트랜잭션 처리 시스템이나 운영체제 등 30만 라인 이상의 소프트웨어를 개발하는 유형

모형의 종류

: 비용 산정 단계 및 적용 변수의 구체화 정도로 구분

ㄱ. 기본형(Basic) : 소프트웨어의 크기와 개발 유형만을 이용하여 비용을 산정

ㄴ. 중간형(Intermediate) : 기본형의 공식을 사용하거나 4가지 특성의 15가지 요인에 의해 비용을 산정

a. 제품의 특성 : 요구되는 신뢰도, 데이터베이스 크기, 제품의 복잡도

b. 컴퓨터의 특성 : 수행 시간의 제한, 기억 장소의 제한, 가상 기계의 안정성, 반환 시간

c. 개발 요원의 특성 : 분석가의 능력, 개발 분야의 경험, 가상 기계의 경험, 프로그래머의 능력, 프로그래밍 언어의 경험

d. 프로젝트 특성 : 소프트웨어 도구의 이용, 프로젝트 개발 일정, 최신 프로그래밍 기법의 이용

ㄷ. 발전형(Detailed) : 개발 공정별로 보다 자세하고 정확하게 노력을 산출하여 비용을 산정

Putnam 모형 [필기 빈출]

: 소프트웨어 생명 주기의 전 과정 동안에 사용될 노력의 분포를 가정해주는 모형

- 생명 주기 예측 모형이라고도 함
- 시간에 따라 함수로 표현되는 Rayleigh-Norden 곡선의 노력 분포도를 기초로 함
- 대형 프로젝트 노력 분포 산정에 이용
- SLIM : Rayleigh-Norder 곡선과 Putnam 예측 모델을 기초로 하여 개발된 자동화 측정 도구

기능 점수(FP) 모형

: 소프트웨어 기능을 증대시키는 요인별로 가중치를 부여하고 합산하여 총 기능 점수를 산출하며 총 기능 점수와 영향도를 이용하여 기능 점수를 구한 후 이를 이용해서 비용을 산정

- 알브레히트가 제안

110. 소프트웨어 개발 방법론 결정

소프트웨어 개발 방법론

: 프로젝트 관리와 재사용 현황을 개발 방법론에 반영하고 확정된 소프트웨어 생명 주기와 개발 방법론에 맞춰 개발 단계, 활동, 작업, 절차 등을 정의

프로젝트 관리 유형

ㄱ. 일정 관리 : 작업 순서 / 기간 설정, 일정 개발 / 통제

ㄴ. 비용 관리 : 비용 산정 / 예산 편성 / 통제

ㄷ, 인력 관리 : 프로젝트 팀 편성 / 관리 / 개발, 프로젝트 조직 정의, 자원 산정 / 통제

ㄹ. 위험 관리 : 위험 식별 / 평가 / 대처 / 통제

ㅁ. 품질 관리 : 품질 계획 / 보증 수행 / 통제 수행

111. 소프트웨어 개발 표준

소프트웨어 개발 표준

- 소프트웨어 개발 단계에서 수행하는 품질 관리에 사용되는 국제 표준

ISO/IEC 12207

- ISO에서 만든 표준 소프트웨어 생명 주기 프로세스
- 소프트웨어의 개발, 운영, 유지보수를 관리하기 위한 생명 주기 표준을 제공
- 기본 / 생명 / 조직 생명 주기 프로세스로 구분

CMMI(Capability Maturity Model Integration, 능력 성숙도 통합 모델)

: 소프트웨어 개발 조직의 업무 능력 및 조직의 성숙도를 평가

- 성숙도는 초기, 관리, 정의, 정량적 관리, 최적화로 구분

SPICE(Software Process Improvement and Capability dEtermination, 소프트웨어 처리 개선 및 능력 평가 기준)

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

- ISO/IEC 15504

목적

- 프로세스 개선을 위해 개발 기관이 스스로 평가
- 기관에서 지정한 요구조건의 만족 여부를 개발 조직이 스스로 평가
- 계약 체결을 위해 수탁 기관의 프로세스를 평가

프로세스

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

프로세스 수행 능력 단계

- 불완전 : 구현되지 않거나 목적을 달성하지 못함
- 수행 : 수행되고 목적이 달성됨
- 관리 : 정의된 자원 한도 내에서 작업 산출물을 인도
- 확립 : 소프트웨어 공학 원칙에 기반하여 정의된 프로세스가 수행
- 예측 : 목적 달성을 위해 통제되고 양적인 측정을 통해 일관되게 수행
- 최적화 : 수행을 최적화하고 지속적인 개선을 통해 업무 목적을 만족시킴

112. 소프트웨어 개발 방법론 테일러링

소프트웨어 개발 방법론 테일러링

- 프로젝트 상황 및 특성에 맞도록 정의된 소프트웨어 개발 방법론의 절차, 사용 기법 등을 수정 및 보완하는 작업

소프트웨어 개발 방법론 테일러링 고려사항

- 내부적 요건 : 목표 환경, 요구사항, 프로젝트 규모, 보유 기술
- 외부적 요건 : 법적 제약사항, 표준 품질 기준

소프트웨어 개발 방법론 테일러링 기법

- 프로젝트 규모와 복잡도에 따른 테일러링 기법 : 프로젝트 규모와 업무의 난이도에 따라 복잡도를 구분
- 프로젝트 구성원에 따른 테일러링 기법 : 구성원들의 숙련도와 방법론의 이해 정도를 확인하여 테일러링 수준을 결정
- 팀 내 방법 지원에 따른 테일러링 기법 : 각 팀별로 방법론 담당 인력을 배정하여 팀의 방법론 교육과 프로젝트 전체의 방법론 운영을 위한 의사소통을 담당하도록 인력을 구성
- 자동화에 따른 테일러링 기법 : 중간 단계에서 산출물을 자동화 도구를 사용하여 작업 부하를 줄이면서 산출하도록 지원

113. 소프트웨어 개발 프레임워크

소프트웨어 개발 프레임워크

: 소프트웨어 개발에 공통적으로 사용하는 구성 요소와 아키텍처를 일반화하여 손쉽게 구현할 수 있도록 여러 가지 기능들을 제공해주는 반제품 형태의 소프트웨어 시스템

ㄱ. 스프링 프레임워크 : 자바 플랫폼과 동적인 웹 사이트의 개발을 위한 프레임워크

ㄴ. 전자정부 프레임워크 : 우리나라의 공공부문 정보화 사업 시 효율적인 정보 시스템의 구축을 지원하기 위해 필요한 기능 및 아키텍처를 제공

ㄷ. 닷넷 프레임워크 : Windows 프로그램의 개발 및 실행 환경을 제공하는 프레임워크로 Microsoft에서 통합 인터넷 전략을 위해 개발됨

<V-II IT프로젝트 정보 시스템 구축 관리>

114. 신기술 관련 용어 [필기 빈출],[실기 빈출]

초심자를 위한 TIP

: 신기술 관련 용어 파트는 사실상 IT상식적인 내용이 자주 나온다. 특히 필기는 해당 정리로도 충분하지만 실기에서는 사회에서 이슈성으로 나오는 IT용어들에 대해서도 종종 문제화되어 나온다.(예를들면 블록체인같은 것들) 그러니 해당 파트는 기존에 나왔던 것이라도 확실히 알고, 새로운 유형으로 나오면 그냥 상식에 맡기는 수 밖에 없고, 크게 합격 유무에 영향을 주지는 않는다.

네트워크 관련 신기술 용어

1. IoT(Internet of Things, 사물인터넷) : 정보 통신 기술을 기반으로 실세계와 가상 세계의 다양한 사물과 사람을 인터넷으로 소로 연결하여 진보된 서비스를 제공하기 위한 서비스 기반 기술

2. M2M(Machine to Machine, 사물 통신) : 무선 통신을 이용한 기계와 기계 사이의 통신

3. 모바일 컴퓨팅 : 휴대형 기기로 이동하면서 자유롭게 네트워크에 접속하여 업무를 처리할 수 있는 환경

4. 클라우딩 컴퓨팅 : 각종 컴퓨터 자원을 중앙 컴퓨터에 두고 인터넷 기능을 갖는 단말기로 언제 어디서나 인터넷을 통해 컴퓨터 작업을 수행할 수 있는 환경

5. 모바일 클라우딩 컴퓨팅 : 모바일 컴퓨팅과 클라우딩 컴퓨팅을 혼합하여 클라우드 서비스를 이용하여 모바일 기기로 클라우딩 컴퓨팅 인프라를 구성하여 여러 가지 정보와 자원을 공유하는 ICT 기술

6. 인터클라우드 컴퓨팅 : 각기 다른 클라우드 서비스를 연동하거나 컴퓨팅 자원의 동적 할당이 가능하도록 여러 클라우드 서비스 제공자들이 제공하는 클라우드 서비스나 자원을 연결하는 기술

7. 매시 네트워크 : 차세대 이통통신, 홈네트워킹, 공공 안전 등 특수 목적을 위한 새로운 방식의 네트워크 기술로 대규모 디바이스의 네트워크 생성에 최적화

8. 와이선 : 장거리 무선 통신을 필요로 하는 사물인터넷 서비스를 위한 저전력 장거리 통신 기술

9. NDN(Named Data Networking) : 콘텐츠 자체의 정보와 라우터 기능만으로 데이터 전송을 수행하는 기술

10. NGN(Next Generation Network, 차세대 통신망) : 유선망 기반의 차세대 통신망으로 유선망뿐만 아니라 이동 사용자를 목표로 하며 이동 통신에서 제공하는 완전한 이동성 제공을 목표로 개발

11. SDN(Software Defined Networking) : 네트워크를 컴퓨터처럼 모델링하여 여러 사용자가 각각의 소프트웨어들로 네트워킹을 가상화하여 제어하고 관리하는 네트워크

12. NFC(Near Field Communication) : 고주파를 이용한 근거리 무선 통신 기술

13. UWB(Ultra WideBand) : 짧은 거리에서 많은 양의 디지털 데이터를 낮은 전력으로 전송하기 위한 무선 기술로 무선 디지털 펄스라고도 함

14. 피코넷 : 여러 개의 독립된 통신 장치가 블루투스 기술이나 UWB 통신 기술을 사용하여 통신망을 형성하는 무선 네트워크 기술

15. WBAN(Wireless Body Area Network) : 웨어러블 또는 몸에 심는 형태의 센서나 기기를 무선으로 연결하는 개인 영역 네트워킹 기술

16. GIS(Geographic Information System) : 지리적인 자료를 수집, 저장, 분석, 출력할 수 있는 컴퓨터 응용 시스템으로 위성을 이용해 모든 사물의 위치 정보를 제공해줌

17. USN(Ubiquitous Sensor Network) : 각종 센서로 수집한 정보를 무선으로 수집할 수 있도록 구성하는 네트워크

18. SON(Self Organizing Network) : 주변 상황에 맞추어 스스로 망을 구성하는 네트워크

19. 애드 혹 네트워크 : 재난 현장과 같이 별도의 고정된 유선망을 구축할 수 없는 장소에서 모바일 호스트만을 이용하여 구성한 네트워크

20. 네트워크 슬라이싱 : 여러 글로벌 이동통신 표준화 단체가 선정한 5G의 핵심기술 중 하나로 네트워크에서 하나의 물리적인 코어 네트워크 인프라를 독립된 다수의 가상 네트워크로 분리하여 각각의 네트워크를 통해 다양한 고객 맞춤형 서비스를 제공하는 것을 목적으로 하는 네트워크 기술

21. 저전력 블루투스 기술 : 일반 블루투스와 동일한 주파수 대역을 사용하지만 연결되지 않은 상태에서는 절전 모드를 유지하는 기술

22. 지능형 초연결망 : 스마트 시티, 스마트 스테이션 등 4차 산업혁명 시대를 맞아 새로운 변화에 따라 급격하게 증가하는 데이터 트래픽을 효과적으로 수용하기 위해 시행되는 정부 주관 사업

SW 관련 신기술 용어

1. 인공지능 : 인간의 두뇌와 같이 컴퓨터 스스로 추론, 학습, 판단 등 인간 지능적인 작업을 수행하는 시스템

2. 뉴럴링크 : 테슬라의 CEO 일론 머스크가 사람의 뇌와 컴퓨터와 결합하는 기술을 개발하기 위해 설립한 회사

3. 딥 러닝 : 인간의 두뇌를 모델로 만들어진 인공 신경망을 기반으로 하는 기계 학습 기술

4. 전문가 시스템 : 특정 분야의 전문가가 수행하는 고도의 업무를 지원하기 위한 컴퓨터 응용 프로그램

5. 증강현실 : 실제 촬영한 화면에 가상의 정보를 부가하여 보여주는 기술

6. 블록체인 : P2P 네트워크를 이용하여 온라인 금융 거래 정보를 온라인 네트워크 참여자의 디지털 장비에 분산 저장하는 기술

7. 분산 원장 기술 : 중앙 관리자나 중앙 데이터 저장소가 존재하지 않고 P2P 망내의 참여자들에게 모든 거래 목록이 분산 저장되어 거래가 발생할 때마다 지속적으로 갱신되는 디지털 원장

8. 해시 : 임의의 길이의 입력 데이터나 메시지를 고정된 길이의 값이나 키로 변환

9. 양자 암호키 분배 : 양자 통신을 위해 비밀키를 분배하여 관리하는 기술

10. 프라이버시 강화 기술 : 개인 정보 위험 관리 기술

11. 공통 평가 기준 : ISO 15408 표준으로 채택된 정보 보호 제품 평가 기준

12. 개인정보 영향평가 제도 : 개인정보를 활용하는 새로운 정보 시스템의 도입 및 기존 정보시스템의 중요한 변경 시 시스템의 구축, 운영이 기업의 고객은 물론 국민의 사생활에 미칠 영향에 대해 미리 조사, 분석, 평가하는 제도

13. 그레이웨어 : 소프트웨어를 제공하는 입장에서는 악의적이지 않은 유용한 소프트웨어 일지라도 사용자 입장에서는 유용할 수도 있고 악의적일 수도 있는 악성 코드나 공유 웨어

14. 매시업 : 웹에서 제공하는 정보 및 서비스를 이용하여 새로운 소프트웨어나 서비스 데이터베이스 등을 만드는 기술

15. 리치 인터넷 애플리케이션 : 플래스 애니메이션 기술과 웹 서버 애플리케이션 기술을 통합하여 기존 HTML보다 역동적인 웹페이지를 제공하는 플래시 웹페이지 제작 기술

16. 시맨틱 웹 : 컴퓨터가 사람을 대신하여 정보를 읽고 이해하고 가공하여 새로운 정보를 만들어 낼 수 있도록 이해하기 쉬운 의미를 가진 차세대 지능형 웹

17. 증발품 : 판매 계획 또는 배포 계획은 발표되었으나 실제로 고객에게는 판매되거나 배포하지 않고 있는 소프트웨어

18. 오픈 그리드 서비스 아키텍처 : 애플리케이션 공유를 위한 웹 서비스를 그리드 상에서 제공하기 위해 만든 개방형 표준

19. 서비스 지향 아키텍처 : 기업의 소프트웨어 인프라인 정보시스템을 공유와 재사용이 가능한 서비스 단위나 컴포넌트 중심으로 구축하는 정보기술 아키텍처

20. 서비스형 소프트웨어(SaaS) : 소프트웨어의 여러 기능 중에서 사용자가 필요로 하는 서비스만 이용할 수 있도록 한 소프트웨어

21. 소프트웨어 에스크로 : 소프트웨어 개발자의 지적재산권을 보호하고 사용자는 저렴한 비용으로 소프트웨어를 안정적으로 사용 및 유지보수받을 수 있도록 소스 프로그램과 기술정보 등을 제3의 기관에 보관하는 것

22. 복잡 이벤트 처리 : 실시간으로 발생하는 많은 사건들 중 의미가 있는 것만을 추출할 수 있도록 사건 발생 조건을 정의하는 데이터 처리 방법

23. 디지털 트윈 : 현실속의 사물을 소프트웨어로 가상화한 모델

HW 관련 신기술 용어

1. 고가용성 : 긴 시간동안 안정적인 서비스 운영을 위해 장애 즉시 다른 시스템으로 대체 가능한 환경을 구축하는 메커니즘

2. 3D 프린팅 : 평면에 출력하는 것이 아닌 얇은 두께로 한층 한층 적재시켜 하나의 형태를 만들어내는 기술

3. 4D 프린팅 : 특정 시간이나 환경 조건이 갖추어지면 스스로 형태를 변화시키거나 제조되는 자가 조립 기술이 적용된 제품을 3D 프린팅 하는 기술

4. RAID : 여러 개의 하드디스크로 디스크 배열을 구성하여 파일을 구성하여 파일을 구성하고 있는 데이터 블록들을 서로 다른 디스크들에 분산 저장하는 기술

5. 앤 스크린 : N개의 서로 다른 단말기에서 동일한 콘텐츠를 자유롭게 이용할 수 있는 서비스

6. 컴패니언 스크린 : TV 방송 시청 시 방송 내용을 공유하며 추가적인 기능을 수행할 수 있는 디바이스

7. 씬(Thin) 클라이언트 PC : 하드디스크나 주변 장치 없이 기본적인 메모리만 갖추고 서버와 네트워크로 운용되는 개인용 컴퓨터

8. 멤스 : 초정밀 반도체 기술을 바탕으로 센서나 액추에이터 등 기계 구조를 다양한 기술로 미세 가공하여 전기기계적 동작을 할 수 있도록 한 초미세 장치

9. 패블릿 : 태블릿 기능을 포함한 5인치 이상의 대화면 스마트폰

10. 트러스트존 기술 : ARM에서 개발한 하나의 프로세서 내에 일반 애플리케이션을 처리하는 일반 구역과 보안이 필요한 애플리케이션을 처리하는 보안 구역으로 분할하여 관리하는 하드웨어 기반의 보안 기술

11. 엠디스크 : 한 번의 기록만으로도 자료를 영구 보관할 수 있는 광 저장 장치

12. 멤리스터 : 메모리와 레지스터의 합성어로 전류의 방향과 양 등 기존의 경험을 모두 기억하는 소자

DB 관련 신기술 용어

1. 빅데이터 : 기존의 관리 방법이나 분석 체계로는 처리하기 어려운 막대한 양의 정형 / 비정형 데이터 집합

2. 브로드 데이터 : 다양한 채널에서 소비자와 상호 작용을 통해 생산 되었거나 기업 마케팅에 있어 효율적이고 다양한 데이터, 이전에 사용하지 않거나 몰랐던 새로운 데이터나 기존 데이터에 새로운 가치가 더해진 데이터

3. 메타 데이터 : 일련의 데이터를 정의하고 설명해주는 데이터

4. 디지털 아카이빙 : 디지털 정보 자원을 장기적으로 보존하기 위한 작업

5. 하둡 : 오픈 소스를 기반으로 한 분산 컴퓨팅 플랫폼

6. 타조 : 우리나라가 주도적으로 개발 중인 하둡 기반의 분산 데이터 웨어하우스 프로젝트

7. 데이터 다이어트 : 데이터를 삭제하는 것이 아닌 압축하고 중복된 정보를 배제하고 새로운 기분에 따라 나누어 저장하는 작업

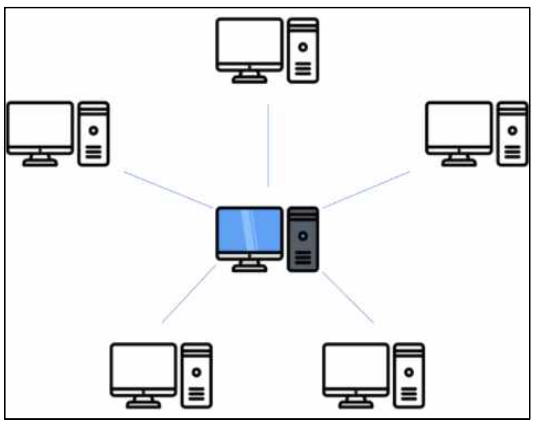
115. 네트워크 구축 [필기 빈출]

네트워크 설치 구조

: 정보를 전달하기 위해 통신 규약에 의해 연결한 통신 설비의 집합

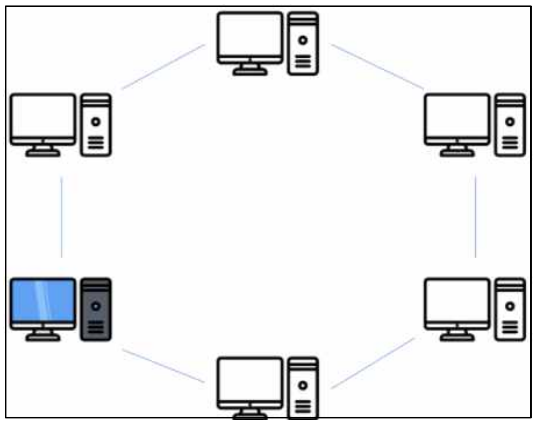
ㄱ. 성형(중앙 집중형)

- 중앙 컴퓨터에 단말 장치들이 연결되는 구조
- Point-to-Point 방식으로 연결
- 중앙 집중식이므로 교환 노드의 수가 가장 적음
- 단말장치가 고장이나도 전체에 영향을 주지 않지만 중앙 장치가 고장이 나면 전체에 영향을 줌



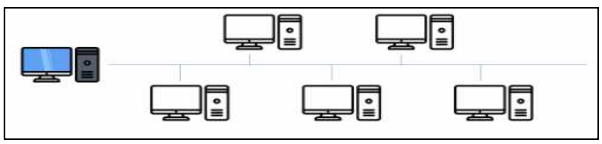
ㄴ. 링형(루프형)

- 컴퓨터와 단말장치들을 서로 이웃하는 것끼리 Point-to-Point 방식으로 연결
- 데이터는 단방향 또는 양방향으로 전송 가능
- 하나의 단말장치라도 고장이나면 전체에 영향을 줌



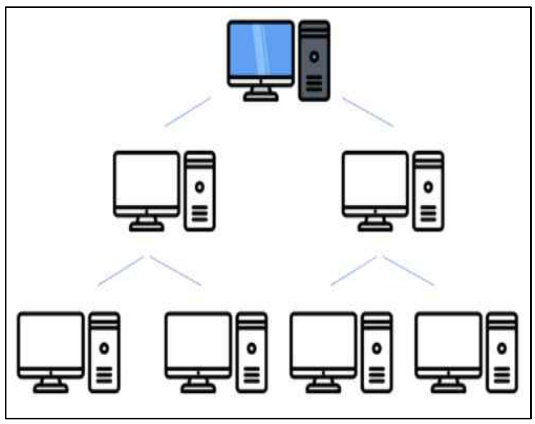
ㄷ. 버스형

- 한 개의 통신 회선에 여러 대의 단말 장치가 연결되어 있는 형태
- 단말장치가 고장이 나도 전체에 영향을 주지 않음

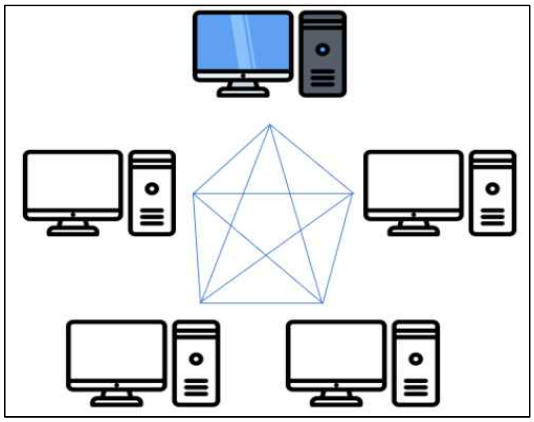


ㄹ. 계층형(분산형)

- 중앙 컴퓨터와 일정 지역의 단말 장치까지는 하나의 회선으로 연결시키고 이웃하는 단말장치는 일정 지역 내 설치된 중간 단말장치로부터 다시 연결
- 분산처리 시스템을 구성하는 방식



ㅁ. 망형(메쉬형)
- 모든 지점의 컴퓨터와 단말장치를 서로 연결시킨 상태
- 많은 양의 통신을 필요로 하는 경우 사용
- 필요한 포트의 수 = 노드 수 - 1
- 필요한 회선의 수 = 노드 수(노드 수 -1) / 2



네트워크 분류

ㄱ. 근거리 통신망(LAN)
- 비교적 가까운 거리에 있는 노드들을 연결하여 구성
- 주로 버스형이나 링형 구조 사용

ㄴ. 광대역 통신망(WAN)
- 국가와 국가, 대륙과 대륙 등 멀리 떨어진 사이트들을 연결하여 구성
- 일정 지역은 LAN으로 연결하고 각 LAN을 연결하는 방식을 사용

116. 스위치 [필기 빈출],[실기 빈출]

스위치

: LAN과 LAN을 연결하여 훨씬 더 큰 LAN을 만드는 장치

스위치의 분류

ㄱ. L2 스위치
- OSI 2계층에 속함
- 일반적으로 부르는 스위치를 말함
- MAC주소를 기반으로 프레임 전송
- 동일 네트워크 간 연결만 가능

ㄴ. L3 스위치
- OSI 3계층에 속함
- L2 스위치에 라우터 기능이 추가
- IP 주소를 기반으로 패킷 전송
- 서로 다른 네트워크 연결 가능

ㄷ. L4 스위치
- OSI 4계층에 속함
- L3 스위치에 로드밸런서(트래픽 분산 장치) 추가
- IP 주소 및 TCP/UDP를 기반으로 사용자들의 요구를 서버의 부하가 적은 곳에 배분하는 로드밸런싱 기능 제공

ㄹ. L7 스위치
- OSI 7계층에 속함
- IP 주소, TCP/UDP 포트 정보에 패킷 내용까지 참조하여 세밀하게 로드밸런싱함

스위칭 방식
- 스위치가 프레임을 전달하는 방식에 따라 나뉨
- Store and Forwarding : 데이터를 모두 받은 후 스위칭
- Cut-through : 데이터의 목적지 주소만 확인 후 바로 스위칭
- Fragment Free : 위의 두 방식의 장점만을 결합한 방식

백본 스위치
- 여러 네트워크들을 연결할 때 중추적 역할을 하는 네트워크인 백본에서 스위칭하는 장비
- 모든 패킷이 지나가는 네트워크 중심에 배치
- 주로 L3 스위치가 백본 스위치 역할을 함

117. 경로 트래픽 제어

경로 제어

: 전송 경로 중 어느 한 경로에 데이터의 양이 집중되는 것을 피하면서 최저의 비용으로 최단 시간에 송신할 수 있는 경로인 최적 패킷 교환 경로를 결정
- 경로 제어표를 참조해서 라우터에 의해 수행
- 경로 제어 요소 : 성능 기준, 경로의 결정 시간 / 장소, 정보 발생지, 경로 정보의 갱신 시간

경로 제어 프로토콜

: 효율적인 경로 제어를 위해 네트워크를 제어하는 프로토콜

ㄱ. IGP(내부 게이트웨이 프로토콜)
- 하나의 자율 시스템(AS) 내의 라우팅에 사용
- RIP : 현재 가장 널리 사용되며 소규모 네트워크에서 효율적인 방법
- OSPF : 대규모 네트워크에서 많이 사용되는 프로토콜

ㄴ. EGP(외부 게이트웨이 프로토콜)
- 자율 시스템 / 게이트웨이 간 라우팅에 사용되는 프로토콜

ㄷ. BGP
- 자율 시스템 간의 라우팅 프로토콜
- EGP의 단점을 보완하기 위해 만들어짐
- 초기 연결 시 라우팅 테이블을 교환하고 이후에는 변화된 정보만을 교환

트래픽 제어

: 네트워크의 보호, 성능 유지, 자원의 효율적인 이용을 위해 전송되는 패킷의 흐름, 양을 조절하는 기능

흐름 제어

: 네트워크 내의 원활한 흐름을 위해 송수신 사이 전송되는 패킷의 양이나 속도를 규제

- 송수신간 처리 속도 또는 버퍼 크기의 차이에 의해 생길 수 있는 버퍼 오버플로우를 방지
- 정지-대기 : 수신 측에서 확인 신호를 받아야 다음 패킷을 전송

슬라이딩 윈도우

- 확인 신호를 이용하여 송신 데이터의 양을 조절
- 수신 측의 확인 신호를 받지 않아도 정해진 패킷 수만큼 연속적으로 전송하는 방식
- 한 번에 여러 개의 패킷을 전송할 수 있음

혼잡 제어

- 네트워크 내의 패킷 수를 조절하여 네트워크의 오버플로우를 방지

교착상태 방지 [필기 빈출],[실기 빈출]

필기에서도 중요하지만 어차피 문제로 접하니 상세개념 패스

- 교환기 내에 패킷들을 축적하는 기억 공간이 꽉 차있을 때 다음 패킷들을 이 기억 공간에 들어가기 위해 무한정 기다리는 현상을 방지

118. 소프트웨어 개발 보안

소프트웨어 개발 보안

: 소프트웨어 개발 과정에서 발생할 수 있는 보안 취약점을 최소화하여 보안 위협으로부터 안전한 소프트웨어를 개발하기 위한 보안 활동

- 데이터의 기밀성, 무결성, 가용성을 유지하는 것이 목표

소프트웨어 개발 보안 관련 기관

- 행정안전부(정책기관), 한국인터넷진흥원(전문기관), 행정기관(발주기관), 사업자(개발기관), 감리법인(보안 약점 진단)

119. 소프트웨어 개발 직무별 보안 활동

소프트웨어 개발 직무별 보안 활동

ㄱ. 프로젝트 관리자 : 보안 전략을 조직 구성원에게 전달하고 모니터링

ㄴ. 요구사항 분석가 : 아키텍트가 고려해야 할 보안 관련 비즈니스 요구사항을 설명

ㄷ. 아키텍트 : 보안 오류가 발생하지 않도록 보안 기술 문제를 충분히 이해하고 시스템에 사용되는 모든 리소스 정의 및 보안 요구사항 적용

ㄹ. 설계자 : 특정 기술에 대해 보안 요구사항의 만족성 여부를 파악하고 소프트웨어에서 발견된 보안 위협에 대해 적절히 대응

ㅁ. 구현 개발자 : 개발 환경에서 프로그램을 구현할 수 있도록 시큐어 코딩 표준을 준수하여 개발

ㅂ. 테스트 분석가 : 개발 요구사항과 구현 결과를 반복적으로 확인

ㅅ. 보안 감시자 : 개발 프로젝트의 전체 단계에서 활동하며 현재 상태와 보안을 보장

120. Secure OS

Secure OS

: 기존 운영체제에 내재된 보안 취약점을 해소하기 위해 보안 기능을 갖춘 커널을 이식한 운영체제

- TCB를 기반으로 참조모니터의 개념을 구현하고 집행
- 참조 모니터와 보안 커널의 특징 : 격리성, 검증 가능성, 완전성
- 보호 대상 : 메모리, 보조 기억장치 및 저장된 데이터, 하드웨어 장치, 자료 구조, 명령어, 각종 보호 메커니즘 등

Secure OS의 보안 기능

- 식별 및 인증, 임의적 접근통제, 강제적 접근 통제, 객체 재사용 보호, 완전한 조정, 신뢰 경로, 감사 및 감사기록 축소

121. 회복/병행제어 [필기 빈출],[실기 빈출]

회복

- 트랜잭션을 수행하는 도중 장애가 발생하여 데이터베이스가 손상되었을 때 복구하는 작업
- 장애의 유형 : 트랜잭션 장애, 시스템 장애, 미디어 장애
- 회복 관리기 : 트랜잭션이 실행이 완료되지 못하면 트랜잭션이 데이터베이스에 생성했던 모든 변화를 취소(Undo)시키고 이전의 원래 상태로 복구하는 역할을 담당

병행 제어

- 동시에 여러 개의 트랜잭션을 수행할 때 데이터베이스의 일관성을 유지할 수 있도록 트랜잭션 간 상호작용을 제어

병행 제어의 목적

- 데이터베이스의 공유 및 시스템의 활용도 최대화
- 데이터베이스 일관성 유지
- 응답 시간 최소화

병행 수행의 문제점

ㄱ. 갱신 분실 : 두 개 이상의 트랜잭션이 같은 자료를 공유하여 갱신할 때 갱신 결과의 일부가 없어짐

ㄴ. 비 완료 의존성 : 하나의 트랜잭션이 실패한 후 회복되기 전에 다른 트랜잭션이 실패한 갱신 결과를 참조

ㄷ. 모순성 : 병행 수행될 때 원치 않는 자료를 이용하여 문제가 발생

ㄹ. 연쇄 복귀 : 트랜잭션 중 하나에 문제가 생겨 ROLLBACK 하는 경우 다른 트랜잭션도 같이 ROLLBACK 됨

122. 데이터 표준화

데이터 표준화

: 시스템을 구성하는 데이터 요소의 명칭, 정의, 형식, 규칙에 대한 원칙을 수립하고 정의

데이터 표준

- 데이터 모델이나 데이터베이스에서 정의할 수 있는 모든 오브젝트를 대상으로 수행
- 표준 단어 : 업무에서 사용하는 일정한 의미를 가진 최소 단위의 단어
- 표준 도메인 : 칼럼을 성질에 따라 그룹핑함
- 표준 코드 : 선택할 수 있는 값을 기준에 맞게 이미 정의된 코드값
- 표준 용어 : 표준 단어 / 도메인 / 코드를 바탕으로 표준 용어 구성

데이터 표준화 절차

- 요구사항 수집-> 데이터 표준 및 표준화 원칙 정의 -> 데이터 표준 검토/확정/공표 -> 데이터 표준 관리

데이터 표준화의 대상

- 데이터 명칭 : 데이터를 유일하게 구별, 의미 전달, 업무적 보편성을 갖는 이름을 가져야 함
- 데이터 정의 : 제3자의 입장에서도 쉽게 이해할 수 있도록 데이터가 의미하는 범위와 자격 요건을 규정
- 데이터 형식 : 데이터를 형식을 일관적으로 정의함으로써 데이터 입력 오류 및 통제 위험 등을 최소화
- 데이터 규칙 : 데이터 값을 사전에 지정해 데이터의 정합성 및 완전성 향상

데이터 표준화의 기대효과

- 데이터의 의미나 위치를 파악하고 의사소통하기 쉽다
- 데이터 유지보수 및 운용에 있어 여러 이점이 있음

<V-III 소프트웨어 개발 보안 구축>

123. Secure SDLC

Secure SDLC(Software Development Life Cycle)

: 보안상 안전한 소프트웨어를 개발하기 위해 SDLC(소프트웨어 개발 생명주기)에 보안 강화를 위한 프로세스를 포함한 것

- 유지보수 단계에서 보안 이슈를 해결하기 위해 소모되는 비용을 최소화하기 위함
- Secure Software 사의 CLASP, Microsoft 사의 SDL 등

요구사항 분석 단계에서의 보안 활동

- 보안 항목에 해당하는 요구사항을 식별하는 작업 수행
- 보안 수준을 보안 요소별로 등급을 구분하여 분류
- 보안 요소 : 기밀성, 무결성, 가용성, 인증, 부인 방지

설계 단계에서의 보안 활동

- 식별된 요구사항을 소프트웨어 설계서에 반영하고 보안 설계서 작성
- 네트워크, 서버, 물리적 보안, 개발 프로그램 등 환경에 대한 보안통제 기준을 수립하여 설계에 반영

구현 단계에서의 보안 활동

- 표준 코딩 정의서 및 소프트웨어 개발 보안 가이드를 준수하여 설계서에 따라 보안 요구 사항 구현
- 단위 테스트 실행
- 시큐어 코딩 : 구현 단계에서 발생할 수 있는 보안 취약점을 최소화하기 위해 보안 요소들을 고려하여 코딩

테스트 단계에서의 보안 활동

- 작성된 보안 설계서를 바탕으로 보안 사항들이 정확히 반영되고 동작되는지 점검

유지보수 단계에서의 보안 활동

- 이전 과정을 모두 수행했음에도 발생할 수 있는 보안 사고들을 식별하고 발생 시 해결하고 보안 패치 실시

124. 세션 통제

세션 통제

: 서버와 클라이언트의 연결인 세션 간의 연결로 인해 발생하는 정보를 관리

- 요구사항 분석 및 설계 단계에서 진단해야 하는 보안 점검 내용

불충분한 세션 관리

- 일정한 규칙이 존재하는 세션ID가 발급되거나 타임아웃이 너무 길게 설정되어 있는 경우 발생
- 세션 하이재킹(세션 정보를 가로채는 공격)을 통해 획득한 세션 ID로 접근할 수 있음

잘못된 세션에 의한 정보 노출

- 다중 스레드 환경에서 멤버 변수에 정보를 저장할 때 발생
- 변수의 범위를 제한하는 방법으로 방지 가능
- 싱글톤 패턴에서 발생하는 레이스컨디션으로 인해 동기화 오류가 발생하거나 멤버 변수의 정보가 노출될 수 있음
- 레이스컨디션 : 두 개 이상의 프로세스가 공용 자원을 획득하기 위해 경쟁하고 있는 상태

세션 설계 시 고려사항

- 로그아웃 요청 시 할당된 세션이 완전히 제거되도록 함
- 이전 세션이 종료되지 않으면 새로운 세션이 생성되지 못하도록 함

125. 입력 데이터 검증 및 표현

입력 데이터 검증 및 표현 [실기 빈출]

: 입력 데이터로 인해 발생하는 문제들을 예방하기 위해 구현 단계에서 검증해야 하는 보안 점검 항목

- 개발 단계에서 유효성 검증 체계를 갖추고 검증되지 않은 데이터가 입력될 시 처리할 수 있도록 구현해야 함
- 일관된 언어셋을 사용하여 코딩

종류	보안 약점
SQL 삽입	입력란에 SQL을 삽입하여 무단으로 DB를 조회, 조작
경로 조작 및 자원 삽입	데이터 입출력 경로를 조작하여 서버 자원을 수정 삭제
크로스사이트 필터링 (XSS)	웹페이지에 악성 스크립트를 삽입하여 방문자의 정보 탈취, 비정상적 기능 수행 유발
운영체제 명령어 삽입	외부 입력값을 통해 시스템 명령어 실행을 유도하여 권한을 탈취하거나 시스템 장애 유발
위험한 형식 파일 업로드	악의적인 명령어가 포함된 스크립트 파일을 업로드하여 시스템에 손상을 입히거나 제어
신뢰되지 않는 URL 주소로 연결	입력값으로 사이트 주소를 받는 경우 이를 조작하여 피싱 사이트로 유도

종류	해결 방법
SQL 삽입	동적 쿼리에 사용되는 입력 데이터에 예약어 및 특수문자가 입력되지 않게 필터링 되도록 설정
경로 조작 및 자원 삽입	경로 순회 공격을 막는 필터 사용
크로스사이트 필터링 (XSS)	HTML 태그 사용을 제한 <,>,& 등의 문자를 다른 문자로 치환
운영체제 명령어 삽입	웹 인터페이스를 통해 시스템 명령어 전달 방지 외부 입력값을 검증없이 내부 명령어로 사용하지 않음
위험한 형식 파일 업로드	업로드 되는 파일의 확장자 제한 파일명의 암호화 웹사이트와 파일 서버의 경로 분리 실행 속성 제거
신뢰되지 않는 URL 주소로 연결	연결되는 외부 사이트의 주소를 화이트 리스트로 관리

126. 보안 기능 [실기 빈출]

보안 기능

: 코딩하는 기능인 인증, 접근제어, 기밀성, 암호화들을 올바르게 구현하기 위해 구현 단계에서의 보안 점검 항목

127. 시간 및 상태

보안 기능

: 코딩하는 기능인 인증, 접근제어, 기밀성, 암호화들을 올바르게 구현하기 위해 구현 단계에서의 보안 점검 항목

시간 및 상태의 개요

- 동시 수행을 지원하는 병렬 시스템이나 다수의 프로세스가 동작하는 환경에서 시간과 실행 상태를 관리하여 원활하게 동작되도록 하기 위한 보안 검증 항목

TOCTOU 경쟁 조건

- 검사 시점과 사용 시점을 고려하지 않고 발생하는 보안 약점

종료되지 않은 반복문 또는 재귀 함수

- 조건이나 논리 구조를 잘못 구성하여 종료할 수 없게 되는 경우 시스템 자원을 끊임없이 사용하여 자원고갈로 인한 서비스 또는 시스템 장애 발생

128. 에러 처리의 개요

에러 처리

- 소프트웨어 실행 중 발생할 수 있는 오류들을 사전에 정의하여 오류로 인해 발생할 수 있는 문제들을 예방하기 위한 보안 점검 항목
- 예외처리 구문을 통해 오류에 대한 사항 정의

오류 메시지를 통한 정보 노출

- 오류 발생으로 실행 환경, 사용자 정보, 디버깅 정보 등 중요 정보를 소프트웨어가 메시지로 외부에 노출하는 보안 약점
- 오류 발생 시 최대한 내부에서 처리하거나 메시지를 최소한의 내용으로 출력하여 정보 노출을 방지해야 함

오류 상황 대응 부재

- 소프트웨어의 오류에 대한 에러 처리를 하지 않았거나 미비로 인해 발생하는 보안 약점

부적절한 예외처리

- 함수의 반환 값 또는 오류들을 세분화하여 처리하지 않고 광범위하게 묶어서 한 번에 처리하거나 누락된 예외가 존재할 때 발생하는 보안 약점

129. 코드 오류

코드 오류

- 소프트웨어 구현 단계에서 코딩 중 실수하기 쉬운 형 변환, 자원 반환 등 오류를 예방하기 위한 보안 점검 항목

널 포인터 역참조

- 널 포인터가 가리키는 메모리에 어떠한 값을 저장할 때 발생하는 보안 약점
- 오류로 인해 반환되는 널 값을 포인터로 참조하는 경우 발생

부적절한 자원 해제

- 자원을 반환하는 코드를 누락하거나 프로그램 오류로 할당된 자원을 반환하지 못했을 때 발생하는 보안 약점
- 유한한 시스템 자원이 계속 점유하고 있으면 자원 부족이 발생

해제된 자원 사용

- 이미 반환된 메모리를 참조하는 경우 발생하는 보안 약점
- 반한 된 메모리를 참조하는 경우 예상하지 못한 값 또는 코드를 수행하게 되어 의도하지 않은 결과가 발생됨

초기화되지 않은 변수 사용

- 변수 선언 후 값이 부여되지 않은 변수를 사용할 때 발생하는 보안 약점

130. 캡슐화 [필기 빈출],[실기 빈출]

캡슐화

: 정보 은닉이 필요한 중요한 데이터와 기능을 불충분하게 캡슐화하거나 잘못 사용함으로써 발생할 수 있는 문제를 예방하기 위한 보안 점검 항목

제거되지 않고 남은 디버그 코드

- 개발 중에 버그 수정이나 결과값을 확인을 위해 남겨둔 코드들로 인해 발생하는 보안 약점

시스템 데이터 정보 노출

- 시스템의 내부 정보를 시스템 메시지 등을 통해 외부로 출력하도록 구현했을 때 발생하는 보안 약점

Public 메소드로부터 반환된 Private 배열

- Private 배열을 Public 메소드에서 반환할 때 발생하는 보안 약점

Private 배열에 Public 데이터 할당

- Private 배열에 Public으로 선언된 데이터 또는 메소드의 파리 미터를 저장할 때 발생하는 보안 약점

131. API오용

API 오용

- 소프트웨어 구현 단계에서 API를 잘못 사용하거나 보안에 취약한 API를 사용하지 않도록 하는 보안 검증 항목

DNS Lookup에 의존한 보안 결정

- 도메인명에 의존하여 보안 결정을 내리는 경우 발생하는 보안 약점
- IP 주소를 직접 입력하여 접근하게 하여 방지 가능

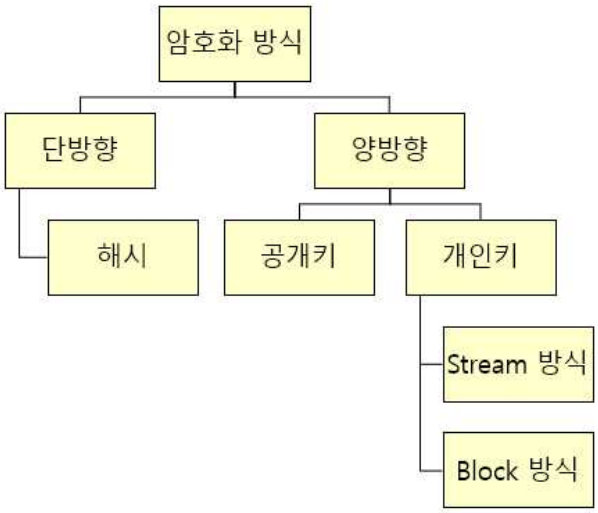
취약한 API 사용

- 보안 문제로 사용이 금지된 API를 사용하거나 잘못된 방식으로 API를 사용했을 때 발생하는 보안 약점

132. 암호 알고리즘 [필기 빈출],[실기 빈출]

암호 알고리즘

: 중요정보를 보호하기 위한 평문을 암호화된 문장으로 만드는 방법



개인키 암호화 기법

- 동일한 키로 데이터를 암호화하고 복호화함
- 대칭 암호 기법, 단일키 암호화 기법이라고도 함
- Stream 기법 : 평문과 동일한 길이의 스트림을 생성하여 비트 단위로 암호화
- Block 기법 : 한 번에 하나의 데이터 블록을 암호화

공개키 암호화 기법

- 데이터를 암호화하는 공개키는 데이터베이스 사용자에게 공개하고 복호화하는 비밀키는 관리자에게만 공개
- 비대칭 암호화 기법이라고도 함
- RSA기법 : 공개키와 비밀키는 메시지를 열고 잠그는 상수를 의미

양방향 암호화 알고리즘 종류

- SEED : 블록 크기는 128비트, 키의 길이에 따라 128, 256로 분류
- ARIA : 블록 크기는 128비트, 키의 길이에 따라 128, 192, 256로 분류
- DES : 블록 크기는 64비트, 키의 길이 56비트
- AES : 블록 크기는 128비트, 키의 길이에 따라 128, 192, 256로 분류

해시(Hash)

- 임의의 길이의 입력 데이터나 메시지를 고정된 길이의 값이나 키로 변환
- SHA 시리즈, MD5, N-NASH, SNEFRU 등

133. 서비스 공격 유형 [필기 빈출],[실기 빈출]

서비스 거부 공격

: 표적이 되는 서버의 자원을 고갈시킬 목적으로 다수의 공격자 또는 시스템에서 대량의 데이터를 한 곳의 서버에 집중적으로 전송함으로써 표적이 되는 서버의 정상적인 기능을 방해

Ping of Death

: Ping 명령 전송 시 패킷의 크기를 인터넷 프로토콜 허용 범위 이상으로 공격하여 공격 대상의 네트워크를 마비시키는 서비스 거부 방법

SMURFING

: IP나 ICMP의 특성을 악용하여 엄청난 양의 데이터를 한 사이트에 집중적으로 보냄으로써 네트워크를 불능 상태로 만드는 공격 방법

SYN Flooding

: 공격자가 가상의 클라이언트로 위장하여 3-way-handshake 과정을 의도적으로 중단시킴으로써 공격 대상지인 서버가 대기 상태에 놓여 정상적인 서비스를 수행하지 못하게 하는 공격 방법

TearDrop

: 데이터의 송수신 단계에서 전송되는 Fragment Offset 값을 변경시켜 패킷을 재조립할 때 오류로 인한 과부하를 발생시킴으로 시스템이 다운되도록 하는 공격 방법

Land

: 패킷 전송 시 송수신 IP 주소를 모두 공격 대상의 IP주소로 하여 공격 대상에게 전송하여 무한히 자신에게 응답을 수행하게 되는 공격 방법

DDos(Distributed Denial of Service, 분산 서비스 거부) 공격

: 여러 곳에 분산된 공격 지점에서 한 곳의 서버에 대해 공격을 수행

- 네트워크에서 취약점이 있는 호스트들을 탐색한 후 호스트들에게 분산 서비스 공격 툴을 설치하여 에이전트로 만든 후 공격에 이용

분산 서비스 공격 툴

ㄱ. Trin00 : 초기 형태의 데몬으로 UDP Flooding 공격 수행

ㄴ. TFN : UDP Flooding, TCP SYN Flood 공격, ICMP 응답 요청, 스머핑 공격 등 수행

ㄷ. TFN2K : TFN의 확장판

ㄹ. Stacheldraht : 이전의 툴들을 유지하면서 암호화된 통신을 수행하며 툴이 자동으로 업데이트되도록 설계

네트워크 침해 공격 관련 용어

ㄱ. 스미싱 : 문자 메시지를 이용해 사용자의 개인 신용 정보

를 빼내는 수법

ㄴ. 스피어 피싱 : 일반적인 이메일로 위장한 메일을 지속적으로 발송하여 메일의 링크나 첨부된 파일을 클릭하게 유도하여 개인 정보를 탈취

ㄷ. APT(지능형 지속 위협) : 조직적으로 특정 기업이나 조직 네트워크에 침투해 활동 거점을 마련한 뒤 때를 기다리면서 보안을 무력화시키고 정보를 수집한 다음 외부로 빼돌리는 형태의 공격

ㄹ. 무작위 대입 공격 : 암호화된 문서의 암호키를 찾기 위해 무작위로 값을 대입하여 공격하는 방식

ㅁ. 큐싱 : QR코드를 통해 악성 앱을 다운받게 하여 개인 정보를 탈취하는 공격 방식

ㅂ. SQL 삽입 공격 : 웹사이트를 무차별적으로 공격하는 과정에서 취약한 사이트 발견 시 데이터를 조작하는 일련의 공격 방식

ㅅ. 크로스 사이트 스크립 : 웹 페이지의 내용을 사용자 브라우저에 표현하기 위해 사용되는 스크립트의 취약점을 악용한 해킹 기법

정보 보안 침해 공격 관련 용어 [필기 빈출],[실기 빈출]

ㄱ. 좀비 PC : 악성코드에 감염되어 다른 프로그램이나 컴퓨터를 조종하도록 만들어진 컴퓨터

ㄴ. C&C 서버 : 해커가 원격지에서 감염된 좀비 PC에 명령을 내리고 악성코드를 제어하기 위한 용도로 사용하는 서버

ㄷ. 봇넷 : 악성 프로그램에 감염된 컴퓨터들이 네트워크로 연결된 형태

ㄹ. 웜 : 네트워크를 통해 연속적으로 자신을 복사하여 시스템의 부하를 높여 시스템을 다운시키는 바이러스의 일종

ㅁ. 제로 데이 공격 : 보안 취약점이 발견됐을 때 공표되기도 전에 해당 취약점을 통해 신속하게 이루어지는 보안 공격

ㅂ. 키로거 공격 : 사용자의 키보드 움직임을 탐지하여 개인 정보를 몰래 빼가는 공격

ㅅ. 랜섬웨어 : 사용자의 컴퓨터에 잡입해 파일을 암호화하여 사용자가 열지 못하게 하는 프로그램

ㅇ. 백도어 : 액세스 편의를 위해 시스템 보안을 제거하여 만들어 놓은 비밀 통로를 통해 범죄에 악용되는 형태

ㅈ. 트로이 목마 : 정상적인 기능을 하는 프로그램인 척 프로그램에 숨어 있다가 해당 프로그램이 동작될 때 활성화되어 부작용을 일으키는 형태

134. 서버 인증

보안 서버

: 인터넷을 통해 개인정보를 암호화하여 송수신할 수 있는 기능을 갖춘 서버

- 서버에 SSL(Secure Socket Layer) 인증서를 설치하여 전송 정보를 암호화하여 송수신
- 서버에 암호화 응용 프로그램을 설치하고 전송 정보를 암호화하여 송수신

인증의 개념

: 다중 사용자 컴퓨터 / 네트워크 시스템에서 로그인을 요청한 사용자의 정보를 확인하고 접근 권한을 검증하는 보안 절차

ㄱ. 지식 기반 인증
- 사용자가 기억하고 있는 정보를 기반으로 인증을 수행
- 고정된 패스워드, 패스 프레이즈, 아이핀

ㄴ. 소유 기반 인증
- 사용자가 소유하고 있는 것을 기반으로 인증을 수행
- 신분증, 메모리 카드, 스마트 카드, OTP

ㄷ. 생체 기반 인증
- 사용자의 고유한 생체 정보를 기반으로 인증을 수행
- 지문, 홍채/망막, 얼굴, 음성, 정맥

ㄹ. 기타 인증 방법
- 행위 기반 인증 : 사용자의 행동 정보를 이용해 인증 수행
- 위치 기반 인증 : 인증을 시도하는 위치나 적절성 확인

135. 보안 아키텍처/프레임워크

보안 아키텍처

: 정보 시스템의 무결성, 가용성, 기밀성을 확보하기 위해 보안 요소 및 보안 체계를 식별하고 이들 간의 관계를 정의한 구조

- ITU-T, X.805의 보안 표준을 기준으로 하여 보안 아키텍처 모델 구성
- 보안 계층 : 인프라 시스템, 응용 프로그램, 데이터, 단말기, 인터페이스
- 보안 영역 : 정보 시스템, 제어 시스템, 클라우드, 무선, 사물인터넷
- 보안 요소 : 인증, 접근 통제, 데이터 처리 보호, 암호화, 감사 추적, 위협 탐지

보안 프레임워크

: 안전한 정보 시스템 환경을 유지하고 보안 수준을 향상시키기 위한 체계

- ISO 27001 : 정보 보안 관리를 위한 국제 표준이며 가장 대표적인 보안 프레임워크

136. 로그 분석

로그

- 시스템 사용에 대한 모든 내역을 기록하여 시스템 침해 사고 발생 시 해킹 흔적이나 공격 기법을 파악할 수 있음

초심자를 위한 Tip

수학에서 지수와 로그 할 때 그 로그와는 다른 로그이다. 비전공자인 당신이 떠올린 $\log 2^3$ 이런 수학과는 전혀 무관하다.

리눅스 로그

- var/log 디렉토리에서 기록하고 관리
- syslogd 데몬은 etc/syslog.conf 파일을 읽어 로그 관련 파일들의 위치를 파악 후 작업 시작
- 커널 로그, 부팅 로그, 크론 로그, 시스템 로그, 보안 로그, FTP 로그, 메일 로그

윈도우 로그

- Windows 시스템에서 이벤트 로그 형식으로 시스템의 로그 확인
- 응용 프로그램, 보안, 시스템, Setup, Forwareded Event에 대한 로그 확인 가능

137. 보안 솔루션

보안 솔루션

: 접근 통제, 침입 차단 등을 수행하여 외부로부터 불법적인 침입을 막는 기술 및 시스템

방화벽

: 기업이나 조직 내부의 네트워크와 인터넷 간에 전송되는 정보를 선별하여 수용, 거부, 수정하는 기능을 가진 침입 차단 시스템

침입 탐지 시스템(IDS)

: 컴퓨터 시스템의 비정상적인 행위를 실시간으로 탐지하는 시스템

- 문제 발생 시 모든 내외부 정보의 흐름을 실시간으로 차단하기 위해 해커 침입 패턴에 대한 추적과 유해 정보 감시가 필요

침입 방지 시스템(IPS)

: 방화벽과 침입 탐지 시스템을 결합

- 비정상적인 트래픽을 능동적으로 차단하고 격리하는 방어 조치를 취하는 보안 솔루션

데이터 유출 방지(DLP)

: 내부 정보의 외우 유출을 방지하는 보안 솔루션

- 내부 PC와 네트워크상의 모든 정보를 검색하고 사용자 행위를 탐지, 통제해 외부로의 유출을 사전에 방지

웹 방화벽

: 일반 방화벽이 탐지하지 못하는 SQL 삽입 공격, XSS 등의 웹 기반 공격을 방어할 목적으로 만들어진 웹 서버에 특화된 방화벽

VPN(가상 사설 통신망)

: 인터넷 등 통신 사업자의 공중 네트워크와 암호화 기술을 이용하여 사용자가 마치 자신의 전용 회선을 사용하는 것처럼 해주는 보안 솔루션

NAC

: 네트워크에 접속하는 내부 PC의 MAC 주소를 IP 관리 시스템에 등록 후 일관된 보안 관리 기능을 제공하는 보안 솔루션

ESM

: 다양한 장비에서 발생하는 로그 및 보안 이벤트를 통합하여 관리하는 보안 솔루션

여기까지 오느라

고생하셨습니다.

이제 당신은 필기시험의 반은 준비가 된 것입니다.

다음 단계인(ㄴ) 기출 문제 풀이 및 분석이 필기시험 준비의 핵심입니다. **앞의 단계를 제대로 할 시간이 없다면 반드시 ㄴ 단계의 기출문제 풀이 및 분석에 더 집중하시길 바랍니다.**

[4. 마무리하며]

ㄴ단계까지 정상적으로 마무리 한 당신! 당신은 이미 필기시험은 합격했다고 봐도 무방하다.

사실상 필기시험만을 합격하기 위해서라면 개념정리 단계(ㄱ)는 없이 기출만 암기하고 분석해서 충분히 합격할 수 있다. 그러나 그렇게 합격하게 된다면 실기시험에서 개념을 바닥부터 다시 쌓아올려야 한다. 그렇기에 최소한의 시간 투자와 효율적인 공부를 위해서 ㄱ단계를 넣은 것이다.

처음 목표는 필기와 실기를 모두 100% 대비 가능한 학습서를 만드는 것이었다. 하지만 회사를 다니며(필자는 정보보안 회사에서 근무 중이다) 틈틈이 작성하다보니 시간이 부족한 것도 있고 실기 시험은 개념정리 책을 내기에는 시험 범위가 너무 광범위하며 시험 문제가 너무 지엽적이다. 어느 정도 시험 문제 운도 따라야 하는 시험이라는 뜻이다. 하지만 내가 집어 준 [실기 빈출] 부분은 대부분의 실기 시험에서 나왔거나 실기에서 자주 나오는 개념을 내포하고 있는 부분들이니 해당 부분들을 제대로 잘 학습해 놓았으면 실기 공부를 할 때 많이 수월하다고 느낄 것이다. (물론 프로그래밍 파트는 제외하고!, 이 파트는 전공자들도 어려워한다, 그래서 2022년 ver 이상부터는 자료를 조금 더 보충하였다.)

필자의 경우 실기 시험 학습을 할 때 '수제비'라는 참고서를 사용 했었다. 솔직히 수제비도 '다른 참고서보다 낫다' 정도이지 좋은 교재라고 까지는 말 못할 것 같다. 하지만 다른 참고서 보다 '두음 암기 법칙'이나 시각적 자료가 잘 되어 있으므로 추천할 법 하다. 그 외의 교재들은 수제비 보다 더 과하게 너무 이것저것 안 나올 법한 내용까지 다 때려 넣은 느낌이 강했다. (수제비 책을 홍보해 주는 것은 절대 아니다. '다른 실기 준비 책 보다는 낫다'는 것이지 '너무 좋다' 라는 것이 아니다!)

책을 판매하는 판매자와 출판사 입장에서야 NCS 내용 다 때려 박아야 나중에 말도 안 될 정도로 지엽적인 문제가 나왔을 경우 수험생들이 "왜 이 파트는 안 넣었어요!!" 라며 따지면 책임을 피할 수 있을 것이고 책의 페이지 수도 현저하게 늘어 날 것이기 때문에 책 판매 가격 면에서도 훨씬 경쟁성이 있을 것이다. 기업의 입장에서는 굳이 나처럼 수험생들의 입장을 생각하는 공부 방법을 만들 이유가 1도 없다. 그렇게 해야만 돈이 되기 때문이다. 어쩔 수 없는 자본주의 시장의 법칙대로 흘러가는 것이다. 그러다보니 이 공부 방법을 모르는, 이제 막 해당 공부를 시작하는 학생들, 직장인들의 공부 방식은 '전공 서적(참고서)을 사서 그냥 이 책 싹 다 외워!'식의 방식으로 흘러가는 것이다. 기업의 입장에서는 굳이 이런 학습자들을 고려할 필요가 없다.

또한 실기 공부를 하다 보면 프로그래밍 파트가 매우 중요해지는데 이는 비전공자라서 어려운 것이 아니라, 그냥 프로그래밍 파트가 재능이 없으면 매우 어려운 파트다...(눈물) 가끔 프로그래밍에 재능이 있는 친구들은 매우 쉽게 점수 거저먹는다며 좋아하는데 전공자고 자시고 그런 사람들은 매우 드물다. 그러니 어렵다고 본인이 바보라고 생각하지 말 것! 실기 책에 나와 있는 프로그래밍 파트를 3회독 정도 하면 충분히 누구나 다 풀 수 있을 정도의 난이도로 나온다. 다만 절대 프로그래밍 파트 좀 커버 치겠다고 강의를 듣거나 실기 책 말고 별도의 코딩 책을 사서 보지 말 것! 겨우 기사 시험 합격하는데 그것은 너무 과투자다. 그렇게 해서 프로그래밍 마스터하고 합격을 하겠다하면 내년쯤에야 합격을 할 수 있을 것이고, 그렇게 해서 고득점으로 합격해봤자 아무도 알아주지도 않는다. 우리의 목표는 항상 '**60점을 넘겨 자격증 취득을 하는 것**'이라는 걸 명심하면서 학습에 임하도록 하자.

마지막으로 내가 이 책을 쓰게 된 이유는 나 역시 시간도 빠듯하고 돈도 없는 '학생'이었기 때문이다. 자라나는 동생들과 후배들을 보며 적어도 필기시험만큼은 내가 다룰 수 있는 바운더리 내에서 현재 나와 있는 참고서 그 누구보다 적은 시간과 효율적인 학습방법으로 경쟁력을 갖추었으면 해서 쓰게 된 책이다.

합격하고 영상에 '좋아요'와 '합격 후기 댓글' 달아주는 것 잊지 말고 해당 학습법에 감동했다면 살포시 '구독'이라는 버튼을 눌러도 된다.

이 책을 봐줘서 고맙고 이 마지막 장을 봐준 여러분들이 자랑스럽다. 지금까지 해온 것처럼 파이팅해서 실기도 합격하길 바라고 실기 관련해서 공부했던 방법은 유튜브에 영상을 따로 만들어 놓았다. 내 방식으로 공부한 실기 합격 후기를 보면 매우 효율적인 방식이라고 자부하니, 실기는 해당 영상 참고해서 공부하면 된다. 필기는 시작일 뿐 실기가 본 게임이다. 필기보다 더 최선을 다해서 실기 공부를 하길 바란다.

2021년 7월 11일 업데이트 한 실기 공부 방법 영상 링크 : https://www.youtube.com/watch?v=CXbpZkKuCFA
내가 코칭한 실기 학습 방법으로 합격한 사람의 후기 링크 : https://blog.naver.com/dbswjdgkssla/222590511516

2022년 4~5월 업데이트 예정 - 실기 과정 1:1 코칭반 모집
-> 추후 링크 자리

[5. Q&A 질의응답]

좋은 질문을 주신 분들이 있어 내용을 **추가로 공유**합니다. 본 질의응답은 책 버전이 업그레이드 될 때마다 내용이 바뀔 수 있습니다.

[유튜브 구독자 / 책 구매자 분들이 주신 질문] Q = 질문 / A = 답변

Q1. 프로그램 짜는 거는 기출 문제 외운다고 되는 게 아닌데 제일 중요한 부분이 책에 언급이 안 되어있는 것 같습니다

A1. 필기에서는 프로그램 파트가 실기에 비해 상대적으로 훨씬 덜 중요합니다. 그렇기 때문에 기출을 봄으로써 충분히 대처가 가능합니다. 또한 필기와 실기 모두 프로그램을 짜는 기출은 나오지 않습니다. 개정된 후 지금까지(20~21년 2회차까지) 필기와 실기 모두 프로그램을 짜라는 기출은 단 한 번도 나온 적 없습니다. 결과 값을 쓰라는 문제이거나 해당 결과 값이 나오려면 어떤 값을 넣어줘야 하냐는 식의 문제가 나왔지요. 프로그램을 짜는 수준으로 가버리면 사실상 비전공자 분들은 거의 못 푸시거나 코딩 파트만 한 달 넘게 잡으셔야 할 것입니다. ('코드로 프로그램을 짜라'와 '해당 코드의 결과 값을 써라'는 난이도 차이가 어마어마하게 난다는 말입니다) 아무래도 이제 시작하시는 것 같은데 포인트를 살짝 잘 못 잡으신 것 같아요~! 영상에서 언급한 정도면 충분 합니다~

필기 합격만 원하시면 개정된 후의 기출을 최대한 더 돌려보시고 지문과 해설을 최대한 눈에 익히고 반복하시면 됩니다.

단, 실기까지 고려하셔서 학습 하실 거면 이론공부가 병행되어야 하며, 프로그래밍 파트가 정말 이해가 안가고 모르겠다! 하시면 실기 책을 사셔서 프로그래밍 파트만 추가로 조금 더 보시는 것 정도는 괜찮습니다(다만 프로그래밍 인강을 사서 듣는다 or 프로그래밍 전용 책을 사서 본다 같은 행동은 합격과는 거리가 멀어지는 것이라고 할 수 있습니다)

Q2. 윤파고님이 판매 중인 개념서에 빠진 개념이 좀 있는 것 같아요 예를 들면 필기 기출(CBT 기출)에서 자료사전 문제가 나오던데 왜 개념서에는 포함하지 않은 건가요?

A2. 전혀 중요하지 않은 개념이거나(현업에서 거의 안 쓰고), 필기에서만 반짝 다루거나, 개념 학습을 하지 않고 기출 문제 해설만 읽어도 충분히 맞출 수 있는 개념은 제외했습니다. 그런 쓰지도 않는 개념은 개념을 익히는 것보다 문제로 익히는 게 훨씬 효율적입니다. 개념서라고 잘 쓰이지도 않고 문제만 봐도 맞출 수 있는 개념을 다 때려 넣으면 제가 추구하는 필기와 실기를 동시에 대비하는 효율적 학습과는 거리가 멀어지기에 전략적으로 개념서에서는 생략하고 CBT기출로 학습해도 필기 합격에 전혀 지장 없는 개념은 일부러 뺐다고 생각하시면 될 것 같습니다. 필기에 대한 학습은 기출 분석이 끝나야 완벽하게 준비가 되었다고 하실 수 있습니다. 기사 공부에서 개념서는 결국 필기 기출 해설에 대한 이해도를 높이고 실기에 대해 미리 준비하기 위해서 필요한 것입니다. 목적과 수단을 혼동하시면 안 됩니다. 필기 기출을 보시다가 제 개념서에 없는 내용은 당황하지 마시고 기출문제와 기출 해설에서 추가적인 지식 습득을 하시면 됩니다. **필기 합격에서 기출과 개념 중 우선순위와 중요도를 따지자면 필기에서는 기출, 실기에서는 개념 이라고 할 수 있습니다**. 해당 개념을 다 보시고 기출까지 N회독 하셨으면 이미 합격에 많이 가까워지셨으니 불안해하시지 않으셔도 됩니다!

Q3. 기출문제를 풀 때 , 그 문제와 비슷한 유형으로 나온다고 생각하고 보기 하나하나를 개념 공부하는 게 나을까요? 아니면 그냥 눈에 익어서 답이 보이도록 하기만 하면 되는 건가요?

A3. 기출 분석이라는 것 자체가 말씀하신 내용 두 가지를 다 포함한다고 보시면 되겠습니다. 필기 기출을 풀어보시면 아시겠지만 결국 나오는 유형의 문제가 반복해서 나오고 있고 그 안에서 정답지가 조금씩 변경되거나 단어가 변경되기 때문에 기출 학습을 하실 때는 문제 유형에 대한 파악 + 지문에 대한 분석이 병행되어야 합니다. 지문에 대한 분석은 문제의 해설을 봄으로서 필기시험에서만 나오는 추가적인 개념을 습득하는 것이고 그 개념의 습득을 용이하게 하고 이해를 돕기 위해서 개념서를 공부 하신 겁니다~! 개념서를 보신 분들은 기출에서 나오는 문제나 해설에 대한 내용이 대부분은 개념서로 한 번씩 훑어보셨던 내용일 거라 습득하시는데 훨씬 빠르고 용어에 대해서 어느 정도 알고 정리가 된 상태라 그냥 기출만 본 사람들에 비해 습득시간이나 변형에 대한 대비에서 훨씬 유리합니다.

2022

기적을 경험하라