자료 구조의 분류

- 선형 구조

> 배열, 큐, 스택, 데크, 선형 리스트

- 배열 = 같은 크기로 나열된 집합

- 큐 = FIFO

- 스택 = LIFO

- 데크 = 리스트의 양쪽 끝 삽입 삭제 모두 허용

\* 선형 리스트 중

- 연속 리스트 = 연속적, 이용 효율은 밀도 1

- 연결 리스트 = 임의 기억공간 기억, 노드 포인터 부분 이용

- 비선형 구조

> 트리, 그래프

- 트리 = Node(노드) Branch(가지), 그래프 형태

ex) a~m

- 노드 = a~m 전체

- 근 노드 = 맨 윗 대가리 a

- 디그리, 차수 = 노드에서 뻗어 나온 가지 수

- 단말 노드 = 자식이 하나도 없는 노드

- 자식 노드 = 부모 아래로 뻗은 자식들

- 부모 노드 = 부모님

- 현제 노드 = 동일 부모를 갖는 노드

- 트리 디그리 = 디그리 중 가장 많은 수

- 레벨 = 각 부모들의 순차 정의 숫치

데이터 저장소

- 논리적 구조 = 정적, 외적

- 물리적 구조 = 동적, 내적

DB

- 통합, 저장, 운영, 공용

DBMS

- 사용자 요구 따라 정보 생성, DB 관리 S/W

- 정의(definition), 조작(mainipulation), 제어(control) =DMC

- 제어 = 보안 유지, 권한 검사, 병행 제어

데이터 입출력

- SQL = IBM에서 개발함

- 관계형 DB = RDB

\*RDB = 2차원적, 표 이용

- 관계 대수 = 절차적(논리)

- 관계 해석 = 비절차적(물리)

-데이터 정의어(DDL) - DROP, ALTER, CREATE [DAC = 닼]

-데이터 조작어(DML) - SELECT, INSERT, DELETE [SID = 싣]

-데이터 제어어(DCL) - 보안, 무결성, 회복, 병행 수행 제어(Control)

데이터 접속(Mapping)

- SQL Mapping

- ORM(객체 - 관계 - 연결)

트랜잭션 - DB 상태 변화 시키는 논리적 기능

- 작업 단위 또는 한꺼번 모두 수행

TCL - 트랜잭션 명령어

- COMMIT = 정상 종료 후 DB에 반영

- ROLLBACK = 했던 모든 변경 작업 취소 및 이전 상태로 감

- SAVEPOINT = ROLLBACK 할 위치 저장점 저장 명령어

절차형 SQL

- C, JAVA, 언어같이 연속적 실행,분기,반복 제어 가능 SQL

- BEGIN ~ END 작성되는 블록(BLOCK)구조 기능별 모듈화

함수정의

- 프로시저 = 트랜잭션 언어로 미리 저장해 놓은 SQL 작업 수행 (SQL, TCL = 트랜잭션 언어)

- 트리거 = 입력, 갱신, 삭제, 이벤트 발생시 자동적 수행

- 사용자 함수 정의 = 프로시저와 유사, 종료 시 RETURN 사용

절차형 SQL의 테스트와 디버깅

- 구문 오류(Syntax Error) = 오타 내고 실행 시 출력되는 거

쿼리 성능 최적화

- SQL 코드 최적화

- APM 도구 사용

- 옵티마이저로 검토 및 SQL 코드, 인덱스 재구성

\* 옵티마이저 = 최적 경로를 찾아 주는 모듈

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

단위 모듈

- 여러 동작 중 한 가지 동작 수행 쌉가능 모듈

- 독립적, 처리, 명령, 데이터 구조

>단위 모듈 구현 순서

- 명세서 작성 > 기능 구현 > 알고리즘 구현

입'출력 기능 구현

- User Interface = CLI, GUI 연동

- Open Source API 이용 간편 구현

>IPC(Inter-Process Coummunication)

- 공유 메모리, 소켓, 세마포어, 파이프&네임드 파이프, 메세지 큐잉

단위 모듈 테스트

- Program 단위 기능 검증

- 단위 테스트라고도 함

- White Box, Black Box

테스트 프로세스 5단계

1. 계획

2. 분석

3. 구현

4. 평가

5. 완료

통합 개발 환경(IDE) - 그 말대로 개발에 필요한 환경 인터페이스 제공

- 다운로드 추가, 정보 공유, 수정 용이함

빌드 도구 - 소스코드 파일들을 컴퓨터에서 실행 가능 제품 S/W로 변환

- 전처리(Preprocessing), 컴파일(Compile) 수행 S/W

\* 대표적 도구 = Ant, Maven, Gradle

소프트웨어 버전 등록

- 형상관리(SCM : Software Configuration Management)

= S/W 변경 사항 관리

- 형상 관리 기능

- 형상 식별 = 이름, 관리 번호, 계층 구조

- 버전 제어 = 특정 절차 및 도구 결합

- 형상 통제 = 기준선(Base Line) 조정 작업

- 형상 감사 = 기준선 무결성 평가

- 형상 기록 = 식별, 통제, 감사 작업 및 결과 기록'관리 보고서

소프트웨어 버전 등록 관련 주요 용어

-저장소(Repository) = 정보 저장소

-가져오기(Import) = 아무것도 없는 저장소에 처음 파일 복사

-체크아웃(Check-Out) = 파일을 받아옴

-체크인(Check-In) = 파일을 새로운 버전으로 갱신

-커밋(Commit) = 충돌 알림 및 수정 후 갱신

\* diff 도구 사용 = 비교대상 파일 비교 후 다른 부분을 찾아 줌

-동기화(Update) = 최신 버전으로 읍그레이드

소프트웨어 버전 등록 과정

- 가져오기 > 체크 아웃 > 커밋(예치) > 업데이트(동기화) > 차이(diff 도구)

공유 폴더 방식

- 버전 관리 자료가 공유 폴더 저장 되어이 있는 거

\* 개발자와 당담자가 공유 폴더에서 이래 저래 활동 하는 뜻

클라이언트 / 서버 방식

- 관리 자료가 중앙 시스템에 저장 되어 관리 방식

\* 개발자 개인들 끼리 시스템에서의 일을 함, 모든 버전 관리는 시스템이 한다

\* 종류 = (C4,S,P) 라고 외우면 쉬움

\* 시스템 = 서버

분산 저장소 방식

- 원격 저장소와 개인 개발자 PC의 로컬 저장소에 함께 저장되 괸리 하는 방식

\* 로컬 저장소에 버전 관리 가능 해, 원격 저장소에서 문제 나도 로컬에서 작업 가능

\* 종류 = (G2,B2,D,M,T,P) 라고 외우면 쉬움

서브버전(Subversion, SVN)

- CVS 개선 한 것, 아파치 s/w에서 발표

\* CVS = 공동 개발 편리 작업 버전 관리 시스템

- 클라이언트/서버 구조

- Commit(반영, 저장 함)

- 커밋 할때마다 리비전 1씩 증가

\* 리비전 = 커밋 버전, 처음 저장소 만들면 0, 커밋 수행될 때마다 1씩 증가

- Open Source이므로 무료 사용

깃(Git)

- 리누스 토발즈 , 주니오 하마노

- 버전 관리는 쿠팡급 처리, 원격 저장소

- 네트워크 문제 있어도 작업 쌉가능

- 지역 저장소 , 로컬 저장소

\* 지역 저장소 = 개발자들이 노는 곳

\* 원격 저장소 = 여러 사람들이 협업 관리하는 곳

- 스냅샷(Snapshot) = 영문,숫자 혼합 40자리 문자열 표시

빌드 자동화 도구

- 빌드를 포함한 테스트 및 배포 자동화 도구

- 애자일 환경이 나오고, 지속적인 통합

Jenkins

- Java 기반 Open Source 자동화 도구

- 서블릿 컨테이너 실행 서버 기반 도구

\* 서블릿 컨테이너 = 클라이언트 요청 처리위해 서블릿(Server Side Applet)실행

서블릿 생명주기 관리

\* Web GUI

Gradle

- Groovy 기반 오픈소스 형태

\* Groovy = Java 에 Python, Ruby, SmallTalk 장점 결합 동적 객체지향 언어

- 안드로이드 앱 개발 환경

- Java, C/C++, Python 언어 가능

- 테스크 단위 실행

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

테스트 기반

- 정적 테스트 (워크스루, 인스펙션, 코드 검사)

- 동적 테스트 (블랙박스, 화이트 박스 테스트)

- 명세, 구조, 경험 기반 (단어의 뜻 의미만 생각하면 됨)

- Verification(검증 기반) = 제품이 (명세서)대로 완성 됐는지 테스트

- Validation(확인 기반) = 제품이 (정상적 동작)하는지 테스트

목적에 따른 테스트

- 회복 테스트(Recovery) = 결함 주어 실패토록 만들고 올바르게 (복구) 되는지 확인

- 안전 테스트(Security) = 불법적 침입으로 보호할 수 있냐 없냐

- 강도 테스트(Stress) = 과도한 정보 부과 후 (과부화) 시 정상적 실행 하냐 안하냐

- 성능 테스트(Performance) = 실시간, 응답 시간, 처리량

- 구조 테스트(Structure) = 내부 논리적 경로, 소스코드 복잡도 평가

- 회귀 테스트(Regression) = 변경 및 수정 시 결함 없음 확인

- 병행 테스트(Parallel) = 변경 전 후 데이터 입력 후 결과 비교

화이트박스 테스트(논리적으로 생각 하기)

- 소스 코드, 논리적 경로

- 종류 = 기초 경로, 제어 구조

\* 제어 구조 = 조건(Condition) , 루프(Loop), 데이터 흐름(Data Flow)

블랙박스 테스트(물리적으로 생각 하기)

- 완전 작동, 테스트

- 종류 = 동치 분할, 경계값 분석, 원인-효과 그래프, 오류 예측, 비교 검사

개발 단계 따른 APP Test

- S/W 개발 단계 = 요구사항 > 분석 > 설계 > 구현

- Test 단계 = 단위 > 통합 > 시스템 > 인수

- 알파 테스트 = 개발자 장소에서 사용자가 개발자 앞에서 테스트 기법

- 베타 테스트 = 선정된 사용자가 여러 명 사용자 앞에서 행하는 기법

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

통합 테스트 - 오류 및 결합 찾는 기법

- 비점진적 통합 = 모듈이 미리 결합되어 있는 프로그램 전체를 테스트 하는 방법

\* 빅뱅 통합 테스트

- 점진적 통합 = 모듈 단위로 단계적 통합 테스트 방법

하향식 통합 테스트

- 위에서 아래로 통합 하면서 테스트

- 깊이 우선 통합법, 넓이 우선 통합법

- Stub

상향식 통합 테스트

- 아래에서 위로 통합 하면서 테스트

- 클러스터(Cluster) 필요

- Driver

혼합식 통합 테스트

- 샌드위치 생각

회귀 테스팅

- 이미 테스트된 프로그램 테스팅 반복, 오류 있나 없나 확인

APP 테스트 프로세스

- 순서 = 테스트 계획 > 분석 및 디자인 > 시나리오 작성 > 테스트 수행 > 리포팅 > 추적 및 관리

테스트 케이스 = 명세 기반 테스트의 설계 산출물

테스트 오라클

- 테스트 결과가 올바른지 판단 위해 참 값을 대입 비교 기법.

- 종류 = 참 오라클, 샘플링 오라클, 추정 오라클, 일관성 검사 오라클(각자 단어 느낌만 알면 됨)

테스트 하네스 도구

- APP 컴포넌트 및 모듈을 테스트하는 환경 일부, 테스트 지원하기 위한 생성 코드, 데이터 의미

- 하네스 구성요소

- 테스트 드라이버 = 호출, 도출 도구

- 테스트 스텁 = 제어 모듈, 일시적 필요 조건 가짐

- 테스트 슈트 = 테스트 케이스 집합, 묶음

- 테스트 케이스 = 사용자 요구사항 정확 테스트 항목 명세서

- 테스트 스크립트 = 자동화된 테스트 실행 절차 명세서

- 테스트 목 오브젝트 = 사전 행위 조건부에 입력 시, 예정된 행위 수행 객체