소프트웨어 개발
데이터 입출력 구현 > 통합 구현> 제품 소프트웨어 패키징 > 애플리케이션 테스트 관리 > 인터페이스 구현

- 자료 구조: 컴퓨터 자료를 효율적으로 저장하기 위해 만들어진 논리적 구조
- 자료 구조: 선형 구조 / 비선형 구조
    - 선형 구조 : 배열 / 연결리스트 / 스택 / 큐
        - 배열(선형 리스트) : 자료의 접근 구조가 빠르다.
        - 연결리스트 : 포인터를 통해 찾는 시간이 추가되어 선형리스트에 비해 느리다.
        - 스택 : 한 방향으로만 자료를 넣고 뺄 수 있는 구조(Last in First out)
        - 큐(Queue) : 한쪽 끝에서는 삽입이 이뤄지고, 반대쪽 끝에서는 삭제 작업이 이루어지는 구조(First in First out)
        - 데크(Deque) : 큐의 양쪽 끝에서 삽입과 삭제를 함께 할 수 있는 자료 구조
    - 비선형 구조: 트리 / 그래프
        - 트리
        - 그래프(주어진 그림에서 찾기)
            1) 전위 순회(Root > 왼쪽 > 오른쪽)
            2) 중위 순회(왼쪽 > Root > 오른쪽)
            3) 후위 순회(왼쪽 > 오른쪽 > Root)

소프트웨어 개발
데이터 입출력 구현 > 통합 구현> 제품 소프트웨어 패키징 > 애플리케이션 테스트 관리 > 인터페이스 구현

- 트리 용어 정리(주어진 그림에서 찾기)
    - 트리의 시작점: 루트 노드
    - 깊이(depth) : 트리의 최대 레벨
    - 차수: 특정 노드에 연결된 자식노드의 수
    - 단말 노드(terminal node)

- 이진 트리의 유형
    - 포화 이진 트리 : 모든 레벨에서 노드가 채워진 트리
    - 완전 이진 트리 : 마지막 레벨을 제외하고 노드가 채워진 트리
    - 편향 이진 트리 : 노드의 왼쪽 또는 오른쪽 한 곳만 노드가 존재하는 트리

- 그래프 유형(n:정점의 개수)
    - 방향 그래프: 최대 간선의 수 n(n-1)
    - 무방향 그래프: 최대간선의 수 n(n-1)/2

- 그래프 용어 정리
    - 경로 길이 : 경로상 있는 간선의 수
    - 단순 경로 : 한 경로의 모든 간선이 다를 때의 경로
    - 사이클 : 동일 정점에서 시작과 끝이 이어지는 경로

- 논리 데이터 저장소: 사용자 혹은 개발자가 이해하기 쉽게 제공하는 업무 모델링 표기법
- 논리 데이터 저장 구조: 개체 / 속성 / 관계

- 논리 데이터 저장소에서 물리 데이터 저장소 모델로 변환하는 절차

    1) 개체를 테이블로 변환한다.
    2) 속성을 컬럼으로 변환한다.
    3) UID(사용자 식별자, User ID)를 기본키(Primary key)로 변환한다.
    4) 관계를 외래 키(Foreign key)로 변환한다.

소프트웨어 개발
데이터 입출력 구현 > 통합 구현> 제품 소프트웨어 패키징 > 애플리케이션 테스트 관리 > 인터페이스 구현

- 물리 데이터 저장소 구성: 테이블 제약조건 설계 > 인덱스 설계 > 뷰 설계 > 클러스터 설계 > 파티션 설계 > 디스크 구성 설계
- 테이블 (삭제/갱신) 제약 조건: 연쇄 / 제한 / 무효

- 인덱스 설계

- 뷰 설계
    - 뷰의 속성 : REPLACE, FORCE, NOFORCE, WITH CHECK OPTION, WITH READ ONLY

- 클러스터 설계

- 파티션 설계
    - 파티션의 종류: 레인지 파티셔닝 / 해시 파티셔닝 / 리스트 파티셔닝 / 컴포지트 파티셔닝
    - 파티션의 장점: 성능 향상 / 가용성 향상 / 백업 가능 / 경합 감소

- 디스크 구성 설계

- ORM(Object-Relational Mapping) 프레임워크 : 데이터베이스, 프로그래밍 언어 간에 호환되지 않는 데이터를 변환하는 프로그래밍 기법
    - SQL Mapping 기반 기술 : iBatis, Mybatis
    - OR Mapping 기반 기술 : Hibernate

- 트랜잭션 인터페이스: 데이터베이스 트랜잭션의 골격 및 인터페이스를 정의하는 활동이다.
- 트랜잭션: 인가받지 않은 사용자로부터 데이터를 보장하기 위해 DBMS가 가져야 하는 특성
- 트랜잭션 인터페이스의 특징
    - 트랜잭션 연산을 데이터베이스 모두에 반영 또는 반영하지 말아야 함: 원자성
    - 트랜잭션이 실행을 성공적으로 완료할 시 일관성 있는 데이터베이스 상태를 유지: 일관성
    - 둘 이상 트랜잭션 동시 실행 시 한 개의 트랜잭션만 접근이 가능하여 접근 불가: 독립성
    - 성공적으로 완료된 트랜잭션 결과는 영구적으로 반영됨: 영속성

- 프로시저(Procedure): SQP을 이용해서 생성된 데이터를 조작하는 프로그램이다.
- 절차형 데이터 조작 프로시저를 Oracle PL/SQL 기반으로 설명한다.
- PL/SQL의 장점: 컴파일 불필요 / 모듈화 가능 / 절차적 언어 사용 / 에러 처리
- PL/SQL을 활용한 저장형 객체 활용
    - 저장된 프로시저
    - 저장된 함수
    - 저장된 패키지
    - 트리거: 특정 테이블에 데이터 변경 이벤트가 발생하면 DBMS에서 자동적으로 실행되도록 구현된 프로그램

- 프로그램 디버깅: 프로시저가 출력을 올바르게 도출하는지에 대해서 확인하는 과정이다.
- 단위 테스트 도구: 구현된 프로시저의 적합성을 확인하기 위한 방법을 제공하는 도구이다.

- 쿼리(Query) 성능 측정 : SQL실행 계획을 최소의 시간으로 원하는 결과를 얻을 수 있도록 프로시저를 수정하는 사전 작업이다.
- SQL 성능 개선 절차
    - 문제 있는 SQL 선별 > 옵티마이저 통계 확인 > SQL문 재구성 > 인덱스 재구성 > 실행계획 유지관리

    * 옵티마이저: 사용자가 질의한 SQL문 실행 계획에 대한 비용을 추적하여, 최적의 실행계획을 수립한다.

- 소스 코드 인스펙션: 프로시저 코드를 보면서 성능 문제점을 개선해 나가는 활동이다.
- 소스 코드 품질 분석 도구(정적도구): pmd, cppcheck, checkstyle
- 소스 코드 품질 분석 도구(동적도구): avalanche, valgrind
- SQL 코드 인스펙션 대상: 미사용 변수 / 미사용 서브쿼리 / Null 값 비교 / 과거의 데이터 타입 사용

소프트웨어 개발

데이터 입출력 구현 > 통합 구현> 제품 소프트웨어 패키징 > 애플리케이션 테스트 관리 > 인터페이스 구현

- 모듈이란 프로그램을 구성하는 구성요소의 일부이며, 관련 데이터와 함수들이 묶여서 모듈을 형성한다.
- 단위 모듈 구현: 소프트웨어 개발에서 기능을 분할하고 추상화하여, 성능을 향상시키기 위한 단위 구현 기법이다.
- 단위 모듈 구현의 원리 : 정보 은닉 / 분할과 정복 / 데이터 추상화 / 모듈 독립성
    - 데이터 추상화: 각 모듈 자료 구조를 액세스 하고 수정하는 함수 내에 자료 구조의 표현 내역을 은폐
    - 모듈 독립성: 낮은 결합도와 높은 응집도를 가진다.

- 단위 모듈 테스트: 단위 모듈 테스트를 위해 IDE(Integrated Development Environment) 도구를 활용하여 디버깅을 수행한다.
    - 화이트 박스 테스트: 소스 코드를 보면서 테스트 케이스를 다양하게 만들어, 테스트를 수행하는 단위 모듈 테스트
    - 메소드 기반 테스트
    - 화면 기반 테스트
    - 테스트 드라이버 / 테스트 스텁
        - 테스트 드라이버: 상향식 테스트 방식
        - 테스트 스텁(Stub): 하향식 테스트 방식
          : 하향식 통합에 있어 모듈 간의 통합 시험을 위해, 일시적으로 필요한 조건만 가지고 임시로 제공되는 시험용 모듈

소프트웨어 개발

데이터 입출력 구현 > 통합 구현> 제품 소프트웨어 패키징 > 애플리케이션 테스트 관리 > 인터페이스 구현

- 테스트 커버리지: 테스트 수행의 완벽성을 측정하는 도구이다.
- 테스트 커버리지 유형: 구문 커버리지 / 결정 커버리지 / 조건 커버리지 / 조건,결정 커버리지 / 변경 조건,결정 커버리지 / 다중 조건 커버리지

    - 프로그램 내 모든 문장을 적어도 한 번 이상 실행하는 것을 기준으로 테스트 커버리지 수행: 구문 커버리지
    - 전체 조건식이 최소한의 참/거짓 한 번의 값을 가지도록 측정하는 테스트 커버리지 수행: 결정 커버리지
    - 각 개별 조건식이 참/거짓 한 번 모두 갖도록 조합하는 테스트 커버리지 수행: 조건 커버리지
    - 전체 조건식이 참/거짓 한 번씩 가지면서, 개별 조건식이 참/거짓 한 번씩 모두 갖도록 조합하는 테스트 커버리지: 조건/결정 커버리지

소프트웨어 개발

데이터 입출력 구현 > 통합 구현> 제품 소프트웨어 패키징 > 애플리케이션 테스트 관리 > 인터페이스 구현

- IDE(Integrated Development Environment): 프로그램 개발과 관련된 모든 작업들을 처리하는 환경을 제공하는 소프트웨어
- IDE 도구의 기능: 개발환경 지원 / 컴파일 / 디버깅 / 외부 연계 / DB 연동
    - 컴파일: 문법에 어긋나는지 확인
    - 디버깅: 프로그래밍 과정에 발생하는 오류 및 비정상 연산 제거

- 대표적인 개발(IDE) 도구로 이클립스, 비주얼 스튜디오, IOS 가 있다.

소프트웨어 개발

데이터 입출력 구현 > 통합 구현> 제품 소프트웨어 패키징 > 애플리케이션 테스트 관리 > 인터페이스 구현

- 협업 도구: 개발자 간의 지속적인 커뮤니케이션이 가능하도록 하는 도구이다.
- 협업 도구를 통해 여러 개발자들의 아이디어를 공유해서 품질이 높아진다.
- 대표적인 협업도구
    - 문서 공유 : 구글 드라이브
    - 소스 공유 : 깃허브
    - 아이디어 공유 : 에버 노트
    - 디자인 공유 : 레드 펜
    - 마인드 맵 : 마인드 마이스터
    - 프로젝트 관리 : 트렐로(Trello), 레드마인(Redmine), 지라(JIRA)
    - 일정관리 : 구글 캘린더

- 협업도구의 기능
    - 개발자 간 커뮤니케이션 / 일정 및 이슈 공유 / 개발자 간 집단 지성 활용

소프트웨어 개발

데이터 입출력 구현 > 통합 구현> 제품 소프트웨어 패키징 > 애플리케이션 테스트 관리 > 인터페이스 구현

소프트웨어 형상 관리 란, 소프트웨어 개발 과정의 변화되는 사항을 관리하는 것이다.

- 형상 관리 도구: 소프트웨어 변경 사항 관리를 위해서 형상 식별, 통제, 감사, 기록을 수행하는 도구이다.
- 대표 제품으로 CVS, SVN, Git 이 있다. (앞의 협업 도구와 헷갈리지 말기!)
- CVS: 중앙 집중형 서버 저장소를 두고 클라이언트가 접속해서 버전관리를 실행한다.
- SVN
- Git: 분산형 방식으로, 필요에 따라 중앙 집중형 방식으로도 운영할 수 있다.

- 형상 관리 도구의 기능
    - 체크인: 개발자가 수정한 소스를 형상 관리 저장소로 업로드 하는 기능
    - 체크아웃: 형상 관리 저장소로부터 개발자 PC로 다운로드 받는 기능
    - 커밋(Commit): 최종적으로 업데이트 되었을 경우, 형상 관리 서버에서 반영하도록 하는 기능

소프트웨어 개발

데이터 입출력 구현 > 통합 구현> 제품 소프트웨어 패키징 > 애플리케이션 테스트 관리 > 인터페이스 구현

- 애플리케이션 패키징: 개발 완료된 소프트웨어 제품을 고객에게 전달할 수 있는 형태로 제작하고, 매뉴얼을 작성하는 활동이다.
- 애플리케이션 패키징은 사용자 중심으로 진행된다.
- 사용자 관점에서의 패키징 고려사항
    - 사용자 시스템 환경 정의 / UI 제공 / 관리 서비스 형태로 제공 / 패키징의 변경 및 개선 관리 고려

- 애플리케이션 패키징 릴리즈 노트: 고객과 공유하는 상세서비스, 개선되는 정보를 제공하는 문서이다.
- 패키징 릴리즈 노트 작성 프로세스: 모듈 식별 > 릴리즈 정보 확인 > 릴리즈 노트 개요 작성 > 영향도 체크 > 정식 릴리즈 노트 작성 > 추가 개선 항목 식별

소프트웨어 개발

데이터 입출력 구현 > 통합 구현> 제품 소프트웨어 패키징 > 애플리케이션 테스트 관리 > 인터페이스 구현

- 애플리케이션 배포 도구: 배포를 위한 패키징 시에 지적 재산권을 보호 및 관리하고, 안전한 유통과 배포를 보장하는 도구이다.
- 애플리케이션 배포 도구 구성요소 : 암호화 / 키 관리 / 식별 기술 / 저장권 표현 / 암호화 파일 생성 / 정책 관리 / 크랙 방지 / 인증 = 디지털 저작 관리 기술 요소
    - 암호화 : 공개 키 기반 구조(PKI), 대칭 및 비대칭 암호화, 전자서명
    - 식별 기술: DOI, URI
    - 저작권 표현: XrML/MPEG-21
    - 정책 관리: XML, CMS(콘텐츠 관리 시스템)
    - 크랙 방지: 난독화, Secure DB

- 디지털 저작 관리 기술(DRM)의 기술 요소 : 크랙 방지 기술 / 정책 관리 기술 / 암호화 기술

- 애플리케이션 배포 도구 활용시 고려사항
    - 암호화.보완 / 이기종 연동 / 복잡성 및 비효율성 문제 / 최적합 암호화 알고리즘 적용
    - 사용자에게 배포되는 소프트웨어이므로 보안을 고려해야 한다.
    - 사용자 편의성을 위한 복합성 및 비효율성 문제를 고려해야 한다.
    - 제품 소프트웨어 종류에 적합한 암호화 알고리즘을 적용한다.

소프트웨어 개발

데이터 입출력 구현 > 통합 구현> 제품 소프트웨어 패키징 > 애플리케이션 테스트 관리 > 인터페이스 구현

- 애플리케이션 모니터링 도구: 소프트웨어 제품을 사용자 환경에 설치하여 모니터링을 통해, 제품을 최적화하기 위한 도구이다.

- 애플리케이션 모니터링 도구 활용에 따른 효과
    - 서비스 가용성
    - 서비스 성능
    - 장애인지/리소스 측정
    - 근본 원인 분석

소프트웨어 개발

데이터 입출력 구현 > 통합 구현> 제품 소프트웨어 패키징 > 애플리케이션 테스트 관리 > 인터페이스 구현

- DRM(Digital Rights Management): 암호화 기술을 통해, 허가된 권한 범위 내에서 디지털 콘텐츠를 이용 가능하도록 통제하는 기술이다
- DRM의 특징: 거래 투명성 / 사용 규칙 제공 / 자유로운 상거래 제공
- DRM 구성: 콘텐츠 제공자 / 콘텐츠 소비자 / 클리어링 하우스
    - 콘텐츠 제공자: DRM 콘텐츠 / 패키저
    - 클리어링 하우스: 콘텐츠 정책 / 콘텐츠 라이선스 / 콘텐츠 관리정보 / 콘텐츠 사용정보
    - 콘텐츠 소비자: DRM 컨트롤러 / 보안 컨테이너

- DRM의 기술요소: 접속 제어 / 사용 제어 / 내용 제어
    - 접속 제어: 권한이 없는 사용자의 접근 자체를 막음
    - 사용 제어: 권한이 없는 사용자의 콘텐츠 사용을 막음
    - 내용 제어: 워터마킹 기술 등을 통하여 소유권 및 불법복제 제어 정보 삽입

소프트웨어 개발

데이터 입출력 구현 > 통합 구현> 제품 소프트웨어 패키징 > 애플리케이션 테스트 관리 > 인터페이스 구현

- 제품 소프트웨어 매뉴얼: 사용자 중심의 기능 및 방법이 적혀진 설명서와 안내서를 의미한다.
- 제품 소프트웨어 설치 매뉴얼

- 제품 소프트웨어 설치 매뉴얼 기본 작성 항목

- 제품 소프트웨어 설치 매뉴얼 구성요소

- 제품 소프트웨어 설치 매뉴얼 작성 프로세스

소프트웨어 개발

데이터 입출력 구현 > 통합 구현> 제품 소프트웨어 패키징 > 애플리케이션 테스트 관리 > 인터페이스 구현

- 국제 표준 제품 품질 특성: 제품의 품질을 평가하는 기준 항목이다.
- 국가 제품 품질 표준

- ISO/IEC 9126의 소프트웨어 품질 특성: 기능성 / 신뢰성 / 사용성 / 효율성 / 유지보수성 / 이식성

- ISO/IEC 14598의 소프트웨어 품질 특성: 반복성 / 재현성 / 공정성 / 객관성