**살충제 패러독스**

: 동일한 테스트케이스로 반복 실행하면 => 더 이상 새로운 결함을 발견할 수 X

따라서 주기적 테스트케이스 점검&개선 필요

**데이터 마이닝**

: 많은 데이터 가운데 숨겨진 유용한 상관관계 발견 => 미래에 실행 가능한 정보 추출, 의사결정에 이용하는 과정

* Ex> 활용성공 사례: 아마존, 넷플릭스

프로토콜 3요소

: 구문 / 타이밍 / 의미

웹 브라우저 간 HTML 문법이 호환되지 않는 문제와 SGML의 복잡함을 해결하기 위하여 개발된 다목적 마크업 언어

: **XML**

속성-값 쌍(attribute-value pairs)으로 이루어진 데이터 오브젝트를 전달하기 위해 사용하는 개방형 표준 포맷 // AJAX에서 많이 사용 // XML 대체하는 주요 데이터 포맷 // 언어 독립형 데이터 포맷으로 다양한 프로그래밍언어에서 사용되고 있다

: Json

SQL 튜플 개수 구하기

* STUDENT 테이블에 컴퓨터정보과 학생 50명 / 인터넷정보과학생 100명 / 사무자동화과 학생 50명에 관한 데이터가 있음
* (1), (2), (3)을 각각 실행 => 결과 튜플 수 각각 몇 개? (DEPT는 학과 컬럼명)

1. SELECT DEPT FROM STUDENT ; 총 20 0개의 데이터(튜플)
2. SELECT **DISTINCT** DEPT FROM STUDENT ; 중복 제거 => 3학과
3. SELECT **COUNT** (DISTINCT DEPT) FROM STUDENT WHERE DEPT=’컴퓨터정보과’

컴퓨터정보과-조건처리 => 50명 / 중복제거&카운트 => 1

: 200, 3, 1

HRN 우선순위 계산식

: (대기시간 + 서비스 시간) / 서비스 시간

* SJF : 프로세스 실행시간이 가장 적은 프로세스를 먼저 실행시키는 스케줄링 기법(FIFO도 보기)
* **HRN**; HIGHEST Response ratio Next 비선점스케줄링

: 짧은 작업에 유리한 SJF의 단점을 개선 한 기법 / 각 작업의 우선순위로 서비스 해주는 스케줄링 ( 우선순위 = 최고 응답률(시스템 응답시간)의 값이 큰 순 )

* + - * 대기시간이 길수록 우선순위 ↑



트랜잭션 특성 **AICD**

: 원자성 Atomicity / 일관성 Consistency / 격리설(=독립성) Isolation / 지속성 Durability

* **트랜잭션** : 데이터베이스 시스템에서 하나의 논리적 기능을 정상적으로 수행하기 위한 작업의 기본 단위
* 원자성 : 트랜잭션 작업이 부분적으로 실행되거나 중단되지 않는 것을 보장하는 것
* 일관성 : 트랜잭션이 성공적으로 완료되면 일관적인 DB상태 유지하는 것
* 격리성 : 트랜잭션 수행 시 다른 트랜잭션의 작업이 끼어들지 못하도록 보장하는 것
* 지속성 : 성공적으로 수행된 트랜잭션은 영구적 반영이 되는 것

공격자가 패킷의 출발지 주소나 포트를 임의로 변경 => 출발지와 목적지 주소(또는 포트)를 동일하게 함으로써, 공격 대상 컴퓨터의 실행 속도를 느리게 하거나 동작을 마비 => 서비스 거부상태에 빠지도록 하는 공격 방법 // 수신되는 패킷 중 출발지 주소(또는 포트)와 목적지 주소(또는 포트)가 동일한 패킷들을 차단함으로써 피할 수 있다

: **LAND 공격**

128비트 암호화 해시 함수 // RFC 1321로 지정 // 주로 프로그램이나 파일이 원본 그대로인지를 확인하는 무결성 검사 등에 사용 // 1991, 로널드 라이베스트가 예전에 쓰이던 MD4 대체하기 위해 고안함

: **MD5**

모듈 설계 시 ((1)) 낮추고, ((2)) 높여야 한다

: 결합도 / 응집도

[문서 이름(릴리스 노트 이름) / 제품 이름 / 버전 번호 / 릴리스 날짜 / 참고 날짜 / 노트 버전,,, 정보]가 설명하는 제품패키지 릴리스 노트의 작성항목

: **헤더**

* 릴리스 노트 : 개발 과정에서 정리된 릴리스 정보를 소프트웨어의 최종 사용자인 고객과 공유하기 위한 문서입니다.
* 릴리스 노트 항목 : 헤더, 개요, 목적, 문제요약, 재현항목, 수정/개선 내용, 사용자 영향도, SW지원 영향도, 노트, 면책조항, 연락처
* 릴리스 노트 작성 순서 : 모듈 식별 > 릴리스 정보 확인 > 릴리스 노트 개요 작성 > 영향도 체크 > 정식 릴리스 노트 작성 > 추가 개선 항목 식별

C언어 – 버블정렬 (오름차순 결과입력)

#include <stdio.h>

void main() {

int i, j, temp;

int a[5] = {75, 95 85, 100, 50};

for (i=0; i<4; i++) {

for(j=0; j<4-i; j++) {

if (a[j] > a[j+1]) {

temp = a[j];

a[j] = a[j+1];

a[j+1] = temp;

}

}

for (i=0; i<5; i++) {

printf(“%d”, a[i]);

}

}

50, 75, 85, 95, 100

* break문 생략 ; 일치하는 실행문부터 switch문이 종료될 때까지 모든 문장 실행