# 제124회 정보관리기술사 해설집

2021.05.28

# 국가기술자격 기술사 시험문제

기술사 제 124 회 제 3 교시 (시험시간: 100 분)

| 분야 | 정보통신 | 자격종목 | 정보관리기술사 | 수검번호 | | 성명 | |
|---|---|---|---|---|---|---|---|

**※ 다음 문제 중 4 문제를 선택하여 설명하시오. (각 10 점)**

1. 최근 우리나라 IT 기업은 개발자 공급부족이라는 매우 어려운 상황에 직면해 있다. 이의 원인과 해결방안을 다음의 관점에서 설명하시오.
   가. IT 서비스 분야의 산업 특수성과 노동 특성 관점에서의 원인
   나. 소프트웨어산업진흥법에 명시된 하도급 구조 개선 제도의 한계점 및 개선방안

2. 자료구조 Heap 의 2가지 유형인 Max-heap 과 Min-heap 을 설명하시오.

3. NoSQL 모델링 패턴 3가지 및 NoSQL 모델링 절차를 설명하시오.

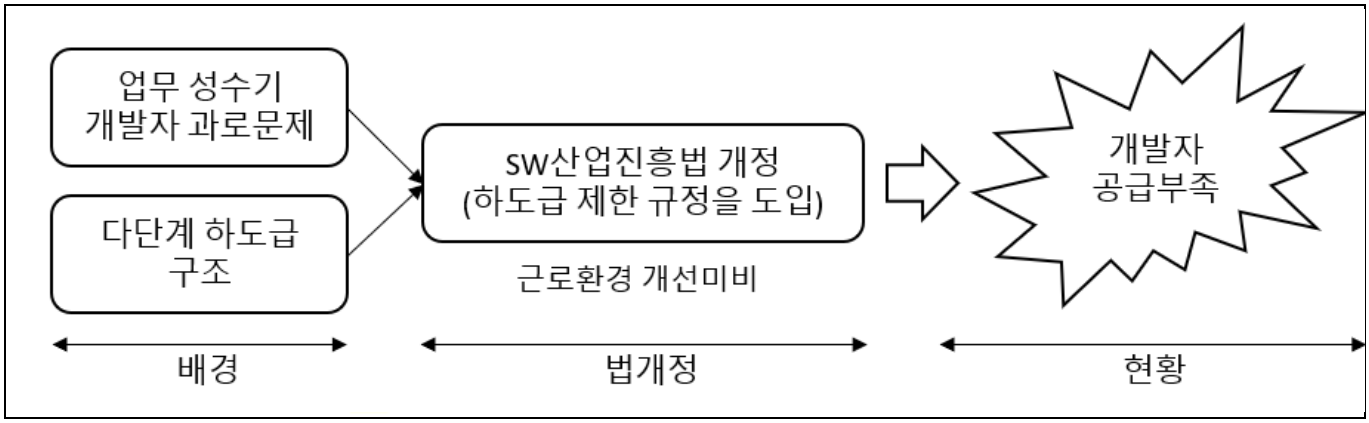
4. 교착 상태(Dead Lock)의 개념과 교착상태를 회피하기 위한 은행가 알고리즘(Banker's Algorithm)의 개념 및 자료구조를 설명하시오.

5. OSI 7 Layer 의 각 계층과 특징을 설명하시오.

6. 침입탐지시스템(Intrusion Detection System, IDS)과 침입방지시스템(Intrusion Prevention System, IPS)의 개념을 비교하여 설명하시오.

| 01 | 소프트웨어산업진흥법 | | |
|---|---|---|---|
| 문제 | 최근 우리나라 IT기업은 개발자 공급부족이라는 매우 어려운 상황에 직면해 있다. 이의 원인과 해결방안을 다음의 관점에서 설명하시오.<br>가. IT서비스 분야의 산업 특수성과 노동 특성 관점에서의 원인<br>나. 소프트웨어산업진흥법에 명시된 하도급 구조 개선 제도의 한계점 및 개선방안 | | |
| 도메인 | 소프트웨어 공학 | 난이도 | 중 (상/중/하) |
| 키워드 | 다단계 하도급, 소프트웨어산업진흥법, 법·제도 개선, 근로환경 개선 | | |
| 출제배경 | IT서비스 분야의 특수성 이해 및 개선방안을 통한 IT서비스 분야의 근로환경 개선의 필요 | | |
| 참고문헌 | IT서비스 분야의 특수성과 근로환경 개선을 위한 시사점(SPRI, 2020.12.21) | | |
| 해설자 | NS반 백기현 기술사(제 122회 정보관리기술사 / onlyride@naver.com) | | |

**I. IT서비스 분야의 개발자 공급 부족의 배경**




- IT서비스 분야의 문제를 해결하기 위해 2008년 SW산업진흥법 개정에서 하도급 제한 규정을 도입하였으나, 근로환경 개선은 중점적으로 다뤄지지 않았고
- IT서비스 분야의 근로환경은 개발자들의 이직과 전직으로 이어지며, 이는 IT서비스 기업의 부실화는 물론이고 개발자들의 번아웃은 IT서비스 분야와 나아가 SW산업 경쟁력을 저하시킬 것으로 예상됨

**II. IT서비스 분야의 산업 특수성과 노동 특성 관점에서의 원인**

**가. IT서비스 분야의 산업 특수성 관점에서의 원인**

| 구분 | 원인 | 설명 |
|---|---|---|
| 산업특성 | - 상품 특성 | - IT서비스 분야는 단순 노동력을 투입하여 주문 생산품을 산출하는 1차 산업의 특성과 1차 산업물을 가공해 재화를 생산하는 2차 산업의 특성 및 서비스를 제공하는 3차 산업 특성을 모두 보유 |
| | - 고도의 지식노동 | - 요구사항과 설계사양을 조정하며 과정에서 감정노동의 요소<br>- 구현, 테스트 단계를 거쳐 검수를 위해서는 고도의 지식노동 수행 |
| | - 사업구조 특성 | - IT서비스 기업은 공공/민간 발주자에게 자신의 전문성을 입증하여 사업을 수주하고 그 운영을 통해 영업이익을 발생시킴 |
| 사내도급 특성 | - 기업내 상주하는 사내 도급 형태 | - 파견인력을 활용하면 파견인력을 관리·감독해야 하는 부담이 있으나, 도급계약을 통해 인력을 활용할 경우 그러한 부담이 없음 |

| | | |
|---|---|---|
| 노사관계 특성 | - SW산업 종사자 대변 창구 부재 | - SW산업은 전반적으로 집단적인 이해당사자들의 의견을 교환하는 창구가 없어서 정부의 정책 추진에서 노동자의 이익을 대변할 수 있는 세력이 실질적으로 부재함 |

- IT서비스 특수성인한 근로 환경으로 인하여 개발자들의 이직과 전직의 반복으로 이어지고 있음

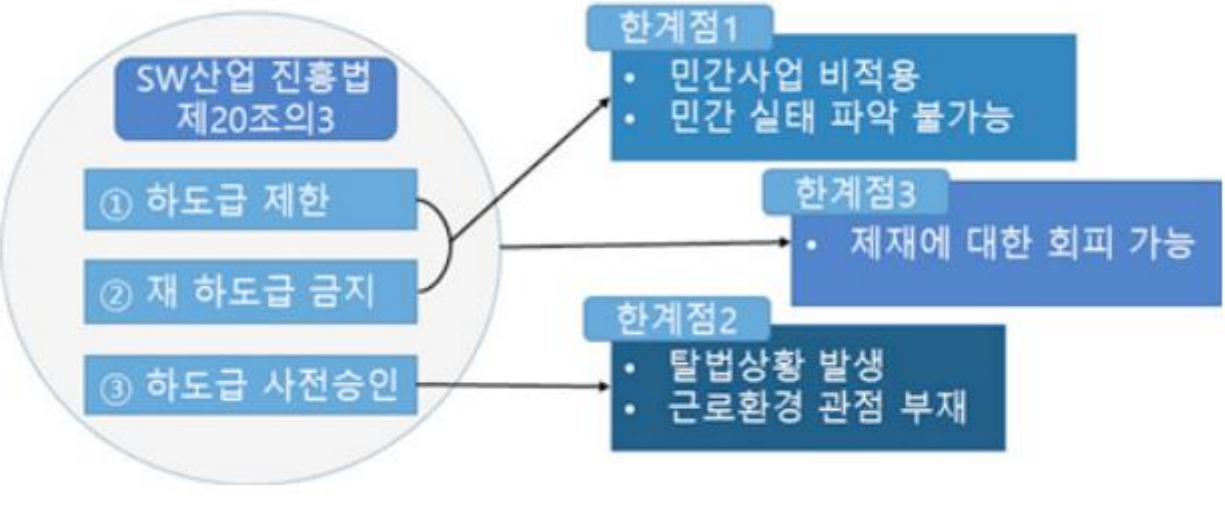
**나. IT서비스 분야의 노동 특성 관점에서의 원인**

| 구분 | 원인 | 설명 |
|---|---|---|
| 업무 강도 및 포괄임금계약 | - 업무강도 | - 특정 시기에 업무가 몰리는 '업무 성수기'에는 크런치 모드가 작동해 퇴근을 상상할 수 없을 정도의 과로에 노출됨 |
| | - 포괄임금계약 | - 제조업 못지않게 SW산업도 초과근로를 하고 있으나, 포괄임금제 관행으로 초과근로에 대한 보상을 받지 못하고 있음 |
| 다단계 하도급 관행 | - 아웃소싱 관행 | - IT서비스 분야는 기술력과 인력을 필요할 때마다 아웃소싱 하는 관례가 다단계 하도급 구조를 형성하게 됨 |
| | - 인건비 축소 현상 | - 다단계 구조에서 과도한 할인율 적용은 하도급 업체들은 수익률 저하시키고, 하도급 업체의 인건비 축소 현상으로 나타나게 됨 |
| 신규 일자리 질 감소 | - 프리랜서 양산 | - 아웃소싱을 통한 인력 수급 특성으로 프리랜서가 양산되고 있으며, 최하층부 업체들만 정규직 근로자로 채용하려 함 |
| | - 일자리 질 감소 | - 다단계 하도급 과정에서 중복적인 할인율은 하층부의 기업의 수익 보전을 위해 일자리 질을 위협함 |
| | - 노동시장 이중구조 | - 일자리의 질 문제로 인해 '노동시장 이중구조' 가 형성되었으며, 하층부 기업은 기술력 축적이 요원해 악순환에 빠짐 |
| | - 개발자 임금 감소 | - 다단계 하도급 과정에서 인력 소개로 수수료만 챙기는 'IT보도방'이 개입하면서 개발자의 임금에 영향을 주고 있음 |

- IT서비스 분야의 다단계 하도급 구조를 개선하는 소프트웨어산업진흥법 개정 진행이 되었지만 한계점 존재

**III. 소프트웨어산업진흥법에 명시된 하도급 구조 개선 제도의 한계점 및 개선방안**

**가. 소프트웨어산업진흥법에 명시된 하도급 구조 개선 제도의 한계점**



| 구분 | 한계점 | 설명 |
|---|---|---|
| 하도급 제한 & 재 하도급 금지 | - 민간사업 비적용 | - 공공사업의 경우 원청은 원칙적으로 사업 금액의 50% 초과한 하도급이 제한되고 재하도급이 금지되나, 민간의 다단계 하도급을 억제할 수 없어 산업 전반에 파급력을 미치지 못함 |
| | - 민간 실태 파악 불가능 | - 제도적 관리를 위해 '공공SW사업 하도급 관리 시스템 21)'을 도입하였으나, 민간사업은 강제사항으로 활용되지 않고 있어 민간사업의 다단계 하도급 실태 파악이 되지 못하고 있음 |
| 하도급 사전승인 | - 탈법상황 발생 | - 공공사업에서 하도급업체가 재하도급 금지 규정을 회피하기 위해서 탈법적으로 인력을 채용하고, 이를 알선하는 업체 등장 |
| | - 근로환경 관점 부재 | - 공공사업의 발주자는 하도급에 관한 관리·감독 책임이 있으나 23), 탈법 상황을 밝히는데 어려움이 있음 |
| 위반에 대한 제재수단 | - 제재에 대한 회피 가능 | - 공공사업에서 발주자 승인 없는 하도급이 있을 경우 부정당 업체로 입찰 참가제한, 과징금 처분 24) 을 받으나, 집행 저지 가능 |

- 현행 제도의 한계점을 법제도 측면에서 개선해 근로환경 개선방안 수립이 필요함

**나. 소프트웨어산업진흥법에 명시된 하도급 구조 개선 제도의 개선방안**

| 구분 | 개선방안 | 설명 |
|---|---|---|
| 법제도 일관성 형성 | - 등록제 | - 일정 요건을 갖추고 등록을 받아야만 산업에 진입하도록 제도화 |
| | - 하도급 제한제도 | - 수급인에게 주요 부분의 하도급을 금지해 재하도급이 원칙적으로 불가능해 다단계 형성이 불가능 |
| | - 파견금지 업종 | - 파견법 개정으로, 아웃소싱을 통해 인력이 유입되어 고용과 사용관계의 분리현상 봉쇄 |
| 처우개선 | - 임금 보전 특례 | - 근로기준법 제44조의 3 및 제44조의 4의 따라 연대책임 인정, 청구권 확대로 하청 근로자의 임금을 제도적으로 보전 |
| | - 처우개선 방안 | - 법·제도 개정으로 근로자 처우개선을 근거를 마련하고 고용관리 책임자 제도 도입으로 사업주의 관리책임을 규정화해 처우개선 |
| | - 사회보장 특례 | - 고용보험 및 산재보험의 보험료 징수 대상자와 퇴직공제의 가입자를 제도적으로 강자에게 전환함으로써 사회보장책 마련 |
| 적정임금제 도입 | - 기준마련 | - 다단계하도급의 근본 이유인 '단가삭감'을 적정임금제 제도 도입으로 단가 삭감을 억제해 다단계하도급을 원칙적으로 차단 |
| | - 기구마련 | - IT근로자를 위한 '근로자공제회'를 설치·운영하여 단일 중소기업에서 제공할 수 없는 각종 복지제도를 IT산업의 종사자를 위해 초기업 단위로 시행 |

- IT서비스 분야의 하도급 구조 형성의 특수성을 인정하고 이에 따른 근로자 환경 개선 위한 시사점 도출가능

**IV. IT서비스 분야의 특수성과 근로환경 개선을 위한 시사점**

| 구분 | 시사점 | 설명 |
|---|---|---|
| 특수성을 인정하고 이에 따른 접근 필요 | - 임금보장 | - 최하층부 개발자의 임금에 대한 실태조사와 적정 임금을 보전하는 방식에 대한 추가 연구 필요 |
| | - 모니터링 | - '일자리 질'의 문제를 해결하기 위한 신고센터를 마련하고, 현행 법 체계에서 보호받을 수 있는 방안 강구 |
| 범부처적 연대 방안 모색 필요 | - 개발자 보호 제도 | - SW개발자의 처우개선을 위한 입법적 해결방안 도모 |
| | - 동반성장 강화 | - SW업계의 '사회적 가치' 실현을 위해서 발주자 및 원청 등에게 부담을 강화하고, 중소기업 지원정책을 통해 하층부의 기업의 '임금 쥐어짜기'현상 방지하여 대·중·소기업의 상생을 도모 |

- 건설 산업은 다단계 하도급 구조 형성이 IT서비스 분야와 유사하고 해당 사항을 법제도 측면에서 개선해 왔으므로 이에 대해 분석해 벤치마킹할 필요가 있음

**"끝"**

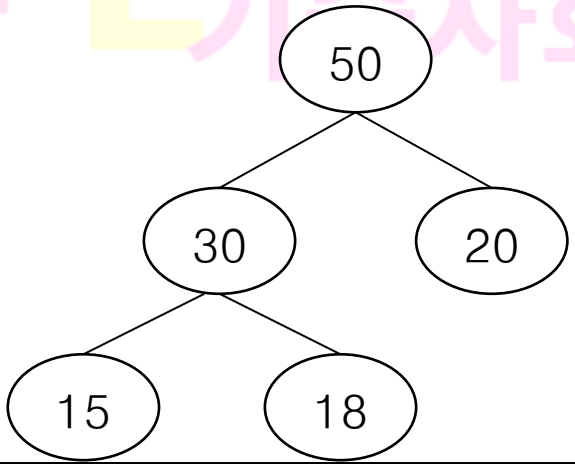
| 02 | Heap | | |
|---|---|---|---|
| 문제 | 자료구조 Heap의 2가지 유형인 Max-heap과 Min-heap을 설명하시오. | | |
| 도메인 | 자료구조 | 난이도 | 중(상/중/하) |
| 키워드 | 완전 이진 트리, 내림차순, 오름차순 | | |
| 출제배경 | 알고리즘 기본 토픽 이해 확인 | | |
| 참고문헌 | ITPE 기술사회<br>https://velog.io/@kjh107704/Heap | | |
| 해설자 | 강남평일야간반 전일 기술사(제 114회 정보관리기술사 / nikki6@hanmail.net) | | |

**I. 완전이진트리를 기본으로 한 자료구조, 힙(Heap)의 개념**

- 완전이진트리(Complete Binary Tree)에 있는 Node 중에서 Key 값이 가장 큰 Node나 가장 작은 Node를 찾기 위한 자료구조
- max-heap은 가장 큰 값을 빠르게 찾기 위한 것이고, min-heap은 가장 작은 값을 빠르게 찾기 위한 것

**II. 가장 큰 값을 빠르게 찾기 위한 Max-heap 상세 설명**

**가. Max-heap 상세 설명**

| 구분 | Max Heap(최대 힙) |
|---|---|
| 정의 | - 부모 Node의 키 값이 자식 Node의 키 값보다 항상 크거나 같은 완전이진트리 |
| 개념도 |  |
| 특징 | - Max Heap에 대해서 원소의 개수만큼 삭제 연산을 수행하여 큰 수부터 POP하여 내림차순으로 정렬 수행 |

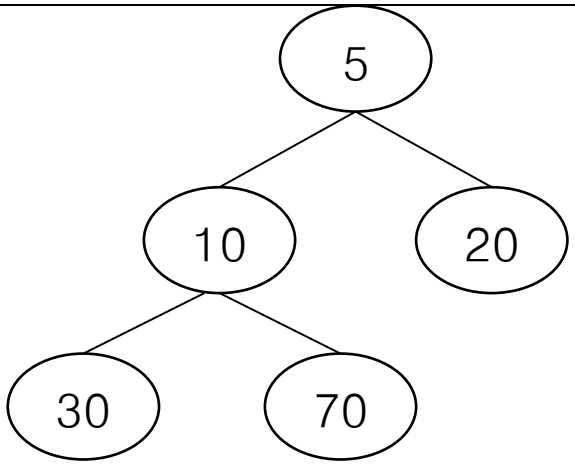
**나. Max-heap 알고리즘 구현 사례**

```
struct MaxHeap
{
    vector<unsigned int> heap;
    int count;
    MaxHeap(int n) : heap(n + 1, 0), count(0) {}
```

```
    void insert(unsigned int input)
    {
        heap[++count] = input;

        for (int i = count; i != 1 && heap[i] > heap[i / 2]; i = i / 2)
            swap(heap[i], heap[i / 2]);
    }

    unsigned int pop()
    {
        if (count == 0)
            return 0;

        int front = heap[1];

        swap(heap[1], heap[count]);
        heap[count--] =  0;

        for (int i = 1; i * 2 <= count;)
        {
            if (heap[i] > heap[i * 2] && heap[i] > heap[i * 2 + 1])
            {
                break;
            }
            else if (heap[i * 2] < heap[i * 2 + 1])
            {
                swap(heap[i], heap[i * 2 + 1]);
                i = i * 2 + 1;
            }
            else
            {
                swap(heap[i], heap[i * 2]);
                i = i * 2;
            }
        }
        return front;
    }
};
```

### III. 가장 작은 값을 빠르게 찾기 위한 Min-heap 상세 설명

#### 가. Min-heap 상세 설명

| 구분 | Min Heap(최소 힙) |
|---|---|
| 정의 | - 부모 Node의 키 값이 자식 Node의 키 값보다 항상 작거나 같은 완전이진트리 |
| 개념도 |  |
| 특징 | - Min Heap에 대해서 원소의 개수만큼 삭제 연산을 수행하여 작은 수부터 POP하여 오름차순으로 정렬 수행 |

#### 나. Min-heap 알고리즘 구현 사례

```
struct MinHeap // heap에 들어오는 범위가 1 ~ 2^31-1 인 경우
{
    vector<unsigned int> heap;
    int count;
    MinHeap(int n) : heap(n + 1, pow(2,31)), count(0) {}

    void insert(unsigned int input)
    {
        heap[++count] = input;

        for (int i = count; i != 1 && heap[i] < heap[i / 2]; i = i / 2)
            swap(heap[i], heap[i / 2]);
    }

    unsigned int pop()
    {
        if (count == 0)
            return 0;

        int front = heap[1];
```

```
        swap(heap[1], heap[count]);
        heap[count--] =   pow(2,31);

        for (int i = 1; i * 2 <= count;)
        {
            if (heap[i] < heap[i * 2] && heap[i] < heap[i * 2 + 1])
            {
                break;
            }
            else if (heap[i * 2] > heap[i * 2 + 1])
            {
                swap(heap[i], heap[i * 2 + 1]);
                i = i * 2 + 1;
            }
            else
            {
                swap(heap[i], heap[i * 2]);
                i = i * 2;
            }
        }
        return front;
    }
};
```

- Heap의 가장 큰 장점은 삽입/삭제 시 추가적인 Sort 미 존재

**IV. 효과적 사용을 위한 힙 정렬 알고리즘의 특징 분석**

| 구분 | 주요 내용 |
|---|---|
| Memory 사용 공간 | - n개의 원소에 대해 n개의 메모리 공간 사용<br>- 크기 n의 Heap 저장공간 |
| 연산 시간 | - Heap 재구성 연산 시간<br>1) n개의 Node에 대해 완전 이진트리는 $\log_2(n+1)$의 Level을 가지므로 Heap 구성 평균시간은 O(log n)<br>2) n개의 Node에 대해 n번의 Heap 재구성 작업 수행 |
| 평균 시간 복잡도 | - O(n log n) |
| 비교 회수 | - 2n log n |
| 특징 | - 다른 정렬들에 비해 최악의 연산 시간일 경우에도 O(n log n)으로 적게 듦<br>- 추가적인 메모리를 필요로 하지 않음 |

- 힙의 단점이라면 데이터 구조에 따라 다른 정렬보다 조금 늦게 정렬될 수 있음

**"끝"**

| 03 | NoSQL | | |
|---|---|---|---|
| 문제 | NoSQL 모델링 패턴 3가지 및 NoSQL 모델링 절차를 설명하시오. | | |
| 도메인 | 데이터베이스 | 난이도 | 중(상/중/하) |
| 키워드 | Atomic Aggregate, Dimensionality Reduction, Index Table | | |
| 출제배경 | NoSQL에 대한 모델링 패턴과 이를 활용한 실무적 관점의 모델링 수행과정 이해 확인 | | |
| 참고문헌 | 빅데이터 시대를 여는 첫걸음 NoSQL 데이터 모델링(지앤선)<br>조대협의 블로그(http://bcho.tistory.com/665)<br>단합반 서브노트 | | |
| 해설자 | 단합반 안경환 기술사(제 110회 정보관리기술사 / akh.itpe@gmail.com) | | |

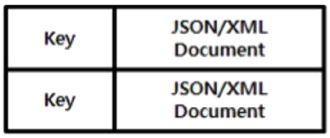
## I. 빅데이터 시대를 여는 인프라. NoSQL의 개요

### 가. NoSQL의 개념

| CAP 이론 | 정의 |
|---|---|
| 분산 환경에서 모든 Server가 **같은 시점**에 **동일**한 **데이터**를 바라보는 것을 의미<br>Oracle, MySQL, DB2 → RDBMS<br>HBase, MongoDB → NoSQL<br>Consistency, Availability, Partition Tolerance<br>CAP<br>NoSQL (Dynamo, Casandra)<br>클러스터를 구성하는 **일부 Server**가 **다운** 되도 **데이터베이스 시스템**은 **정상**적 **동작**을 의미<br>네트워크 장애로 인하여 **Message**가 **전달되지 않거나 유실**되었을 때 **시스템**이 **동작**하는 메커니즘 | - 관계형 데이터베이스(RDBMS)의 테이블-컬럼과 같은 스키마 없이, 분산 환경에서 단순 검색 및 추가 작업이 용이하고, 지연(latency)과 처리율(throughput)이 높은 Database |

### 나. NoSQL의 데이터 모델 유형

| 모델 유형 | 개념도 | 설명 |
|---|---|---|
| Key-Value | Key / Value<br>Key / Value | - 키와 값의 쌍으로 관리<br>- 키를 사용 값을 확인하는 가장 기본적이며 단순한 NoSQL 데이터 모델<br>- 단순한 모델 이므로 Key에 대한 단위연산 속도가 빠름<br>- 키 범위(Key Range)처리 요구에 대한 적용이 안 되는 단점 |
| Ordered Key-Value | Key / Column / Value<br>Sorted by Key | - 키 범위 처리 요구를 개선한 Key-Value 데이터 모델<br>- 키를 기반으로 Sorting 하여 저장 |

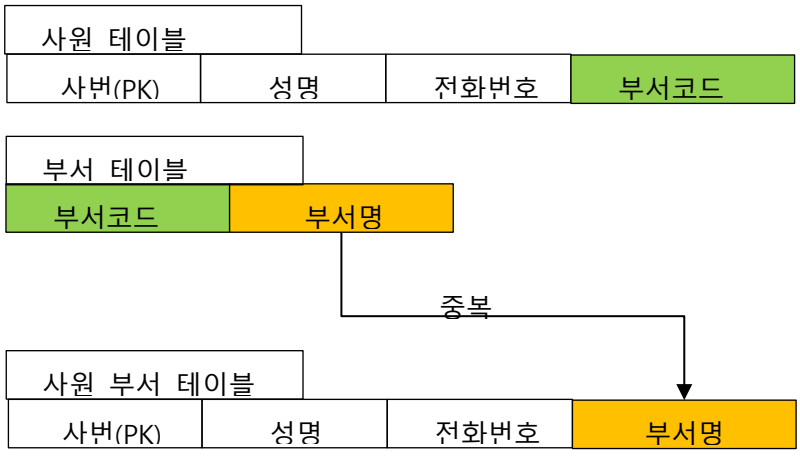
| | | |
|---|---|---|
| Document Key/Value Store |  | - 스키마 없이 임의의 속성을 추가할 수 있는 데이터 모델<br>- Document Id 또는 특정 속성값 기준으로 order-preserving 하여 키 범위 처리에 대한 효율적인 연산이 가능<br>- JSON, XML 형태로 구조적 문서 저장 가능<br>- 쿼리 처리시 데이터를 Parsing 하여 메모리 연산을 해야 하므로 처리 overhead가 Key-Value 데이터 모델에 비해 큼 |

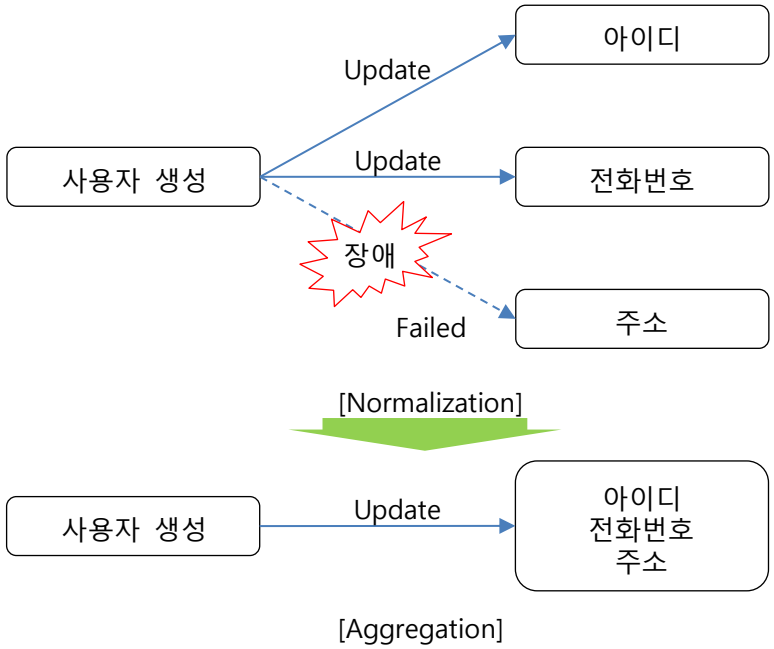
**II. NoSQL 모델링 패턴 3가지**

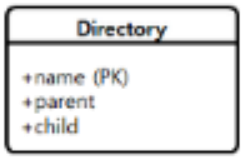
**가. NoSQL 모델링 패턴의 분류 3가지**

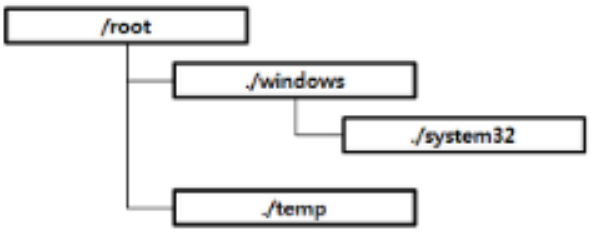
| 패턴 유형 | 세부 패턴 | 설명 |
|---|---|---|
| Conceptual Techniques | - Denormalization, Aggregation, Application Side Joins | - NoSQL 데이터 모델의 기본 원칙에 중점 |
| General Modeling Techniques | - Atomic Aggregates, Enumerable Keys, Dimensionality Reduction, Index Table, Composite Key Index | - NoSQL의 일반적 모델링 적용 기법 |
| Hierarchy Modeling Techniques | - Tree Aggregation, Adjacency List, Materialized Paths, Nested Sets | - 계층적으로 분석하는 모델링 패턴 |

**나. NoSQL 모델링 패턴 3가지**

| 패턴 | 항목 | 설명 |
|---|---|---|
| Denormalization 모델링 패턴 | 정의 | - Foreign Key로 분할되어 있는 테이블의 JOIN을 없애기 위해 데이터를 중복 저장하는 패턴 |
| | 개념도 |  |
| | 특징 | - Join 연산을 지원하지 않는 NoSQL의 단점을 보완<br>- Application에서 분할된 데이터를 조회할 경우 여러 번의 I/O 연산의 발생을 1회로 감소하고 Query 연산 성능 향상<br>- Query가 단순해지나 저장(Storage) 용량의 증가 |
| | 적용분야 | - Key-Value Stores, Document Databases, BigTable-style Databases |

<table>
<tr><td rowspan="4">Atomic Aggregate 모델링 패턴</td><td>정의</td><td>- 하나의 비즈니스 엔티티(Entity)를 하나의 문서, 행 또는 Key –Value쌍으로 저장하고 이를 원자적(Atomic)으로 업데이트 하는 패턴</td></tr>
<tr><td>개념도</td><td> 

[Aggregation]</td></tr>
<tr><td>특징</td><td>- 많은 NoSQL의 트랜잭션(Transaction) 처리의 문제점 해결하기 위한 패턴<br>- 분산된 Table에서 업데이트 도중 장애 등으로 인해 일부가 업데이트 안될 경우 원자성 문제 발생<br>- 분산된 테이블을 하나의 테이블로 구성하여 원자성 보장<br>- 구현상 Aggregation 패턴과 동일하나 Aggregation 패턴은 JOIN을 제거하기 위해 사용되며 Atomic Aggregation은 트랜잭션(Transaction)을 보장 받기 위해 사용되는 패턴</td></tr>
<tr><td>적용분야</td><td>- Key-Value Stores, Document Databases, BigTable-style Databases</td></tr>
<tr><td rowspan="3">Adjacent Lists 모델링 패턴</td><td>정의</td><td>- 자료 구조에서 사용하는 Linked List와 같은 자료 구조형을 사용하여, 각 Tree의 노드에 parent node에 대한 포인터와 child node들에 대한 포인터를 저장하는 모델링 패턴</td></tr>
<tr><td>개념도</td><td> 



<table>
<tr><th>key (name)</th><th>parent</th><th>child</th></tr>
<tr><td>root</td><td>Null</td><td>windows,temp</td></tr>
<tr><td>windows</td><td>root</td><td>system32</td></tr>
<tr><td>temp</td><td>root</td><td>null</td></tr>
</table></td></tr>
<tr><td>특징</td><td>- 특정 Node를 알 경우 상위 혹은 하위 Node를 자유롭게 Traversing 가능<br>- 데이터가 많을 경우 Traversing에 많은 I/O가 발생하므로 비 효율적</td></tr>
</table>

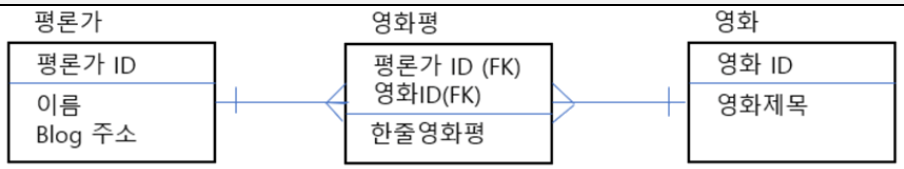
| | | |
|---|---|---|
| | | - 주어진 노드의 모든 하위 트리를 깊이 우선 탐색과 너비 우선 탐색으로 검색 시 비효율적 |
| | 적용분야 | - Key-Value Stores, Document Databases |

**III. NoSQL 모델링 절차**

| 단계 | 절차 | 설명 |
|---|---|---|
| 탐색 | 도메인 모델 파악 | - 저장하고자 하는 도메인 파악<br>- 어떤 데이터 개체가 있는지 개체 간의 관계는 어떻게 되는지 등을 분석하고 ERD활용하여 도식화 |
| 설계 | 쿼리결과 디자인 | - 도메인 모델 기반으로 애플리케이션에 의해서 쿼리 되는 결과값을 정하여 데이터 출력 내용 기반으로 디자인. |
| | 패턴을 이용한 데이터 모델링 | - RDBMS 기능을 제공하지 않기 때문에 이를 배제하고 Put/Get으로만 데이터를 가지고 올 수 있는 형태로 데이터 모델, NoSQL 내의 테이블로 재정의. |
| | 기능 최적화 | - RDBMS의 Index 와 같은 이 개념을 NoSQL 에서 'Secondary Index'을 이용하여 기능의 최적화 |
| 최적화 | 후보 NoSQL을 선정 및 테스트 | - NoSQL에 대한 구조 및 특성을 분석한 후에 실제로 부하테스트, 안정성, 확정성 테스트를 거친 후에 가장 적절한 솔루션을 선택. |
| | 선정된 NoSQL의 데이터 모델 최적화 및 하드웨어 디자인 | - 선정된 NoSQL 을 기반으로 그에 적합한 데이터 모델 최적화 후 이에 맞는 어플리케이션 인터페이스 설계와 구동시킬 하드웨어 디자인을 실시 |

- NoSQL의 key-value 구조로서 RDBMS와 차이점이 존재하며, 모델링 절차 시 구조와 특성의 고려가 필요

**IV. NoSQL 모델링 사례**

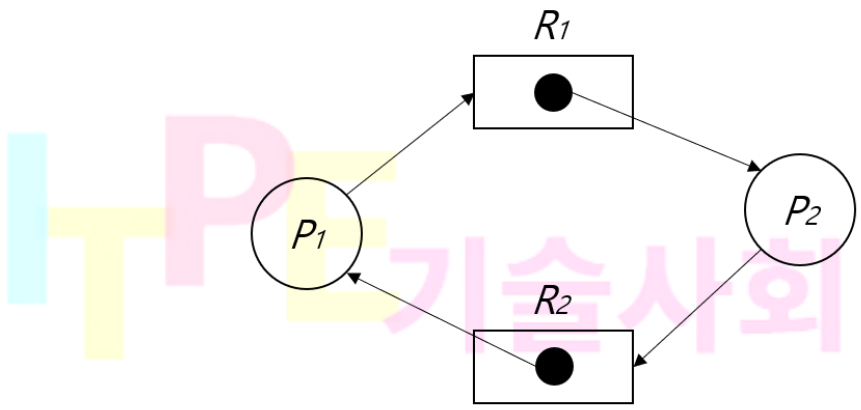
| 구분 | 모델링 사례 |
|---|---|
| RDBMS 모델링 |  |
| NoSQL 모델링 |  |

**"끝"**

| 04 | 은행가 알고리즘(Banker's Algorithm) | | |
|---|---|---|---|
| 문제 | 교착 상태(Dead Lock)의 개념과 교착상태를 회피하기 위한 은행가 알고리즘(Banker's Algorithm)의 개념 및 자료구조를 설명하시오. | | |
| 도메인 | 운영체제(Operating System) | 난이도 | 상(상/중/하) |
| 키워드 | 상호배제, 점유대기, 비선점, 환형대기, 예방, 회피, 발견, 복구, AVAILABLE(가용 자원), MAX(최대 요구), NEED(추가 요구), ALLOCATION(할당 자원), 안전 상태, 불안전 상태 | | |
| 출제배경 | 기본 운영체제의 교착 상태 개념과 해결 방안에 대한 이해 확인 | | |
| 참고문헌 | 운영체제 (곽덕훈, 백두권 공저, 2015.01, 한국방송통신대학교출판문화원)<br>단합반 서브노트 | | |
| 해설자 | 단합반 안경환 기술사(제 110회 정보관리기술사 / akh.itpe@gmail.com) | | |

## I. 운영체제(Operating System)의 무한 대기. 교착 상태(Dead Lock)의 개념

### 가. 교착 상태(Dead Lock)의 개념

| 개념도 |  |
|---|---|
| 개념 | - 특정 집합 내의 프로세스가 그 집합 내의 다른 프로세스에 의해서만 야기될 수 있는 Event를 무한정 대기하는 상태 |

### 나. 교착 상태(Dead Lock)의 발생 조건과 해결 방안

| 구분 | 조건과 해결 방안 | 설명 |
|---|---|---|
| 발생 조건 | 상호배제(Mutual Exclusion) | - 프로세스들이 필요로 하는 자원에 대한 배타적 통제권 요구(필요 하는 자원을 다른 Process가 점유 시 반드시 대기) |
| | 점유대기(hold and wait) | - 프로세스가 이미 다른 자원을 할당 받아 배타적 점유하는 상황에서 다른 프로세스가 점유하고 있는 자원이 해제되기를 대기 상황 |
| | 비선점(no premption) | - 프로세스에 할당된 자원은 그 프로세스가 사용을 마치고 스스로 반환 전까지 제거 불가 |
| | 환형대기(circular wait) | - 프로세스 자원 점유 및 점유된 자원의 요구 관계가 환형을 이루며 대기 |

| | | |
|---|---|---|
| 해결 방안 | 교착 상태 예방 | - 발생 조건 4가지 조건 중 하나라도 발생하지 않도록 제어 |
| | 교착 상태 회피 | - 안정 조건이 유지되도록 자원 관리하거나 프로세스의 시작을 거부 |
| | 교착 상태 발견 | - 주기적으로 환형 조건이 발생하는지 검사하고 발생 시 해결 |
| | 교착 상태 회복 | - 교착 상태 발생 시, 모든 프로세스를 중지하거나 롤백, 혹은 프로세스를 하나씩 종료 시키면서 교착 상태를 해결하는 기법 |

- 해결 기법 중 교착 상태를 회피하기 위한 기본 기법으로 은행가 알고리즘 존재

**II. 교착 상태 회피 기법. 은행가 알고리즘(Banker's Algorithm)의 개념**

**가. 은행가 알고리즘(Banker's Algorithm)의 정의**

시스템이 자원을 요청 받으면 그 자원을 할당해 주고 난 후의 상태를 가상하여 그것이 안전 상태인지 확인한 후 안전 상태가 보장되는 경우에만 할당하는 교착 상태 회피(Deadlock avoidance) 알고리즘(Algorithm)

**나. 은행가 알고리즘(Banker's Algorithm)의 목표. 안정 상태**

| 안정 상태와 불안정 상태의 개념도 | 안정 상태 정의 |
|---|---|
|  | - 시스템이 교착 상태를 일으키지 않으면서 각 프로세스에게(최대 요구량까지) 필요한 자원을 할당할 수 있는 상태 |
| | **불안정 상태 정의** |
| | - 시스템이 각 프로세스에게(최대 요구량) 필요한 자원을 할당한 경우 교착 상태가 발생할 수 있는 상태 |

- 불안정 상태는 교착 상태가 되기 용이하지만, 상태가 안정하다면 운영체제는 교착 상태를 회피 가능

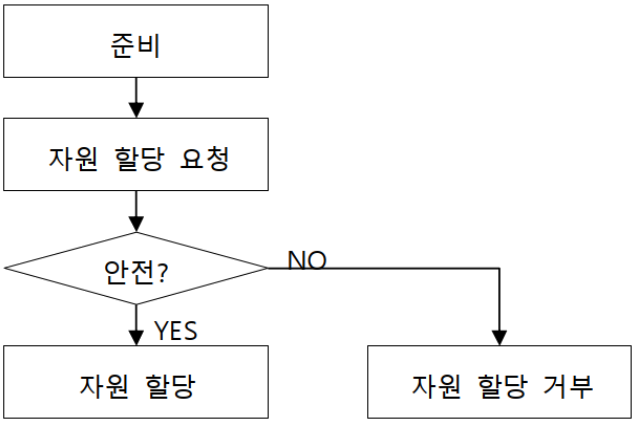
**II. 은행가 알고리즘(Banker's Algorithm)의 자료 구조**

**가. 은행가 알고리즘(Banker's Algorithm)의 자료 구조**

| 자료구조 | 구성 | 표기 | 설명 |
|---|---|---|---|
| AVAILABLE (가용 자원) | - 길이가 m인벡터 | - AVAILABLE[$j$] = $k$ | - 사용 가능한 자원의 수<br>- r$j$ 유형 자원 중 $k$ 개의 자원이 가용 상태 |
| MAX (최대 요구) | - n X m 행렬 | - MAX[$i$, $j$] = $k$ | - 각 프로세스의 자원에 대한 최대 요구량<br>- 프로세스 P$i$는 자원 유형 R$j$에 대해 총 $k$개 요청 가능 |
| NEED (추가 요구) | - n X m 행렬 | - NEED[$i$, $j$] = $k$ | - 현재 할당되어 있는 자원의 수<br>- 프로세스 P$i$는 작업 종료를 위해 자원 유형 R$j$의 자원을 총 k개 필요 |

| | | | |
|---|---|---|---|
| ALLOCATE<br>(할당 자원) | - n X m 행렬 | - ALLOCATE[*i*, *j*] = *k* | - 각 프로세스가 향후 더 요구할 수 있는 자원의 수<br>- 프로세스 P*i*는 현재 자원 유형 R*j*의 자원을 총 *k*개 할당 받은 상태 |

**나. 은행가 알고리즘(Banker's Algorithm)의 자료 구조를 활용한 동작 절차**

| 동작 순서도 | 동작 절차 |
|---|---|
|  | ① 가용 자원(Available)과 최대 요구(Max)를 사전 파악 |
| | ② 프로세스가 가용 자원의 추가 요구(Need) 요청 |
| | ③ 안전 알고리즘(Safety Algorithm)에 의해 안전 상태 여부 확인 |
| | ④ 안전 상태일 경우 자원 할당, 불안전 상태일 경우 할당 거부 |

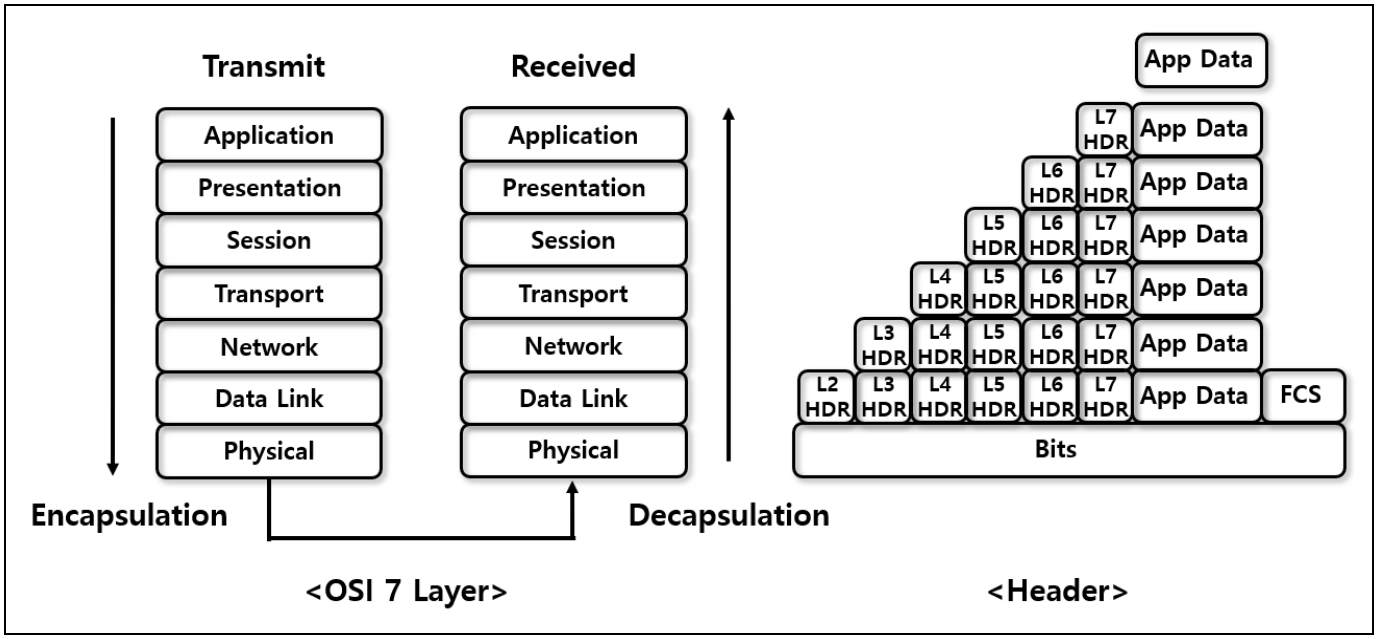
**"끝"**

| 05 | OSI 7 Layer | | |
|---|---|---|---|
| 문제 | OSI 7 Layer의 각 계층과 특징을 설명하시오. | | |
| 도메인 | 네트워크 | 난이도 | 하(상/중/하) |
| 키워드 | Application, Presentation, Session, Transport, Network, Data Link, Physical, 프로토콜 | | |
| 출제배경 | 네트워크 기본 토픽에 대한 이해, 기출(98, 116, 117, 122회) | | |
| 참고문헌 | https://ko.wikipedia.org/wiki/OSI_모형 | | |
| 해설자 | TOP반 유술사 (제 113회 컴퓨터시스템응용기술사 / itpe_you@naver.com) | | |

**I. 상호 연결성을 위한 표준화 모델, OSI 7 Layer의 개요**

**가. OSI(Open System Interconnection) 7 Layer의 정의**

- 모든 네트워크 통신에서 생기는 여러가지 충돌 문제를 완화하기 위하여, 국제표준기구(ISO)에서 표준화된 네트워크 구조를 제시한 기본 모델 (ISO 7498)




**나. OSI 7 Layer의 특징**

| 특징 | 설명 |
|---|---|
| 계층 구조 | - 한 계층의 변경이 다른 계층에 영향을 미치지 않아 관리 용이<br>- 네트워크에 대한 학습 및 이해 용이 |
| 캡슐화 | - 상위 계층에서 하위 계층으로 내려올 때 Header, Trailer 등 첨부 |
| 역 캡슐화 | - 하위 계층에서 상위 계층으로 올라갈 때 해당 Header 분석하고 분리 |
| 통합 표준 | - 현존하는 네트워크 표준들을 하나의 모델로 포괄<br>- 다른 네트워크 환경이라도 상호 통신이 가능하게 하는 지침 제공 |

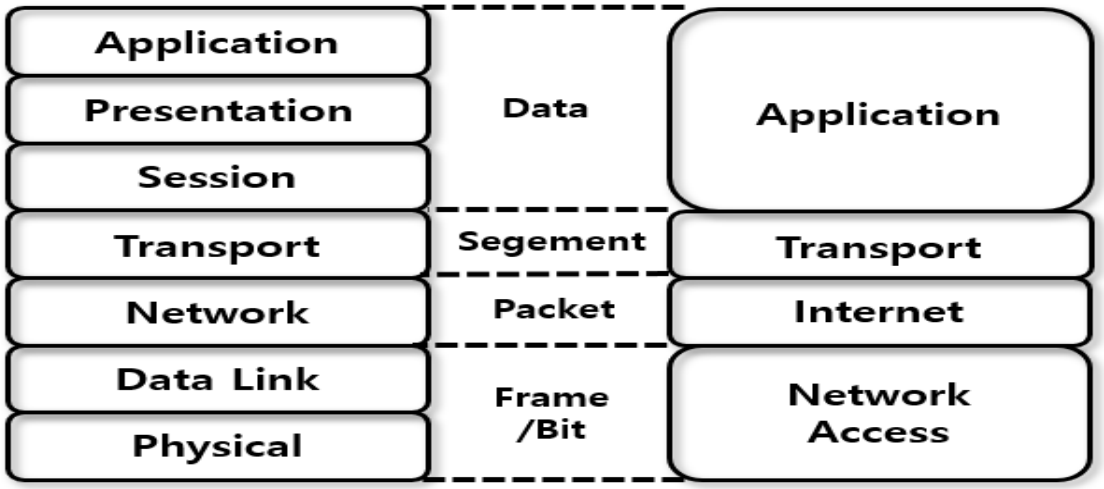
- Layer 4 ~ 7는 Host 부분, Layer 1 ~ 3 Media 부분으로 구분 가능
- OSI 7 Layer는 각 계층별 독립 구성이 가능하며 멀티 프로토콜 지원하는 특징 존재

**II. OSI 7 Layer의 각 계층과 특징**

| 계층 | 특징 | 계층 설명 | 프로토콜 |
|---|---|---|---|
| **7 계층 (Application Layer)** | - 사용자와 NW간의 연결<br>- 데이터 생성 | - 사용자가 네트워크에 접근할 수 있도록 해주는 계층<br>- 사용자 인터페이스, 전자우편, 데이터베이스 관리서비스<br>- 네트워크 소프트웨어 UI 부분 | HTTP, SMTP, SNMP, FTP, Telnet, SSH&SCP, NFS, RTSP, NTP |
| **6 계층 (Presentation Layer)** | - 데이터 형식 규정 | - 운영체제의 한 부분으로 I/O 데이터를 표현형태로 변환<br>- 번역을 수행하여 두 장치가 일관되고 이해할 수 있음<br>- 포장/압축/암호화 | JPEG, MPEG, XDR, SMB, AFP |
| **5 계층 (Session Layer)** | - 인증 및 서비스 제공 | - 통신세션을 구성하는 계층으로 포트연결 확인<br>- 통신장치간의 상호작용을 설정하고 유지하며 동기화<br>- 세션의 확립/유지/중단(운영체제) | TLS, SSH, RPC, NetBIOS, AppleTalk |
| **4 계층 (Transport Layer)** | - 프로세스 간의 데이터 전송 | - 전체 메시지를 발신지 대 목적지간 제어와 에러 관리<br>- 패킷들의 전송 유효확인, 실패한 패킷은 재전송하여 신뢰성 있는 통신 보장<br>머리말에는 세그먼트가 포함<br>- End to End 최하위 계층 | TCP, UDP, RTP, SCTP, SPX |
| **3 계층 (Network Layer)** | - 데이터 경로설정<br>- 스위칭, 라우팅 | - 다중 네트워크 링크에서 패킷을 목적지로 전달<br>- 패킷이 시작시점에서 최종 목적지까지 성공적으로 전달되도록 관리<br>- 주소부여(IP) 및 경로설정(Routing) | IP, ICMP, IGMP, X.25, CLNP, ARP, RARP, BGP, OSPF, RIP, IPX, DDP |
| **2 계층 (Data Link Layer)** | - 네트워크 기기 간의 데이터 전송 | - 오류 없이 한 장치에서 다른 장치로 프레임을 전달<br>- 스위칭 테이블을 참조하여 입력되는 패킷의MAC 주소를 보고 해당 포트로 패킷 전송<br>- 에러검출/재전송/흐름제어 | PPP, HDLC, Ethernet, TokenRing, ISDN, FDDI |
| **1 계층 (Physical Layer)** | - 시스템 간 물리적 연결<br>- 전기적 신호 변환 | - 물리적 매체를 통해 비트(Bit)흐름 전송<br>- 장치간의 물리적 접속을 제어하기 위한 기능 제공<br>- OSI 계층에서 가장 복잡한 계층 | RS-232C, V.35, V.34, Q.911 T1, E1, ISDN, DSL |

- 각 계층은 하위 계층의 기능만을 이용하고 상위 계층에게 기능 제공하며 TCP/IP 구조와 비교

**III. OSI 7 Layer와 TCP/IP의 구조 비교**

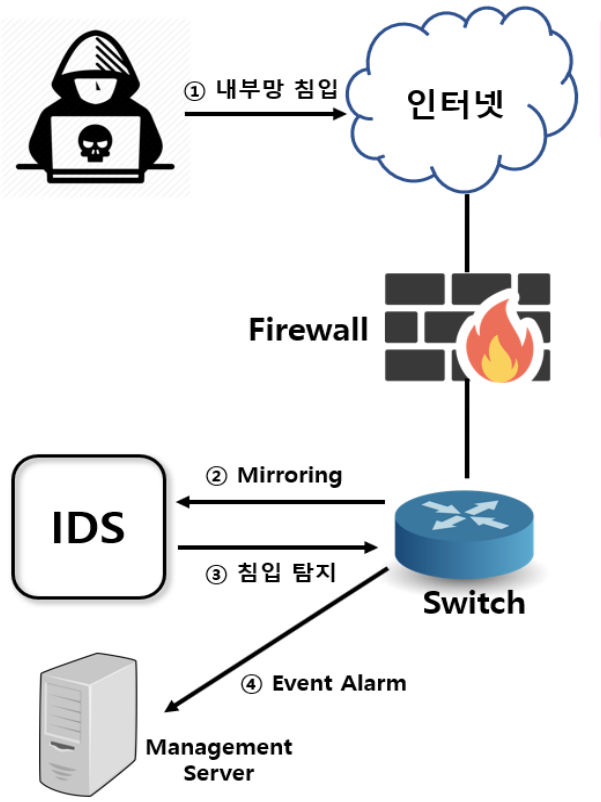
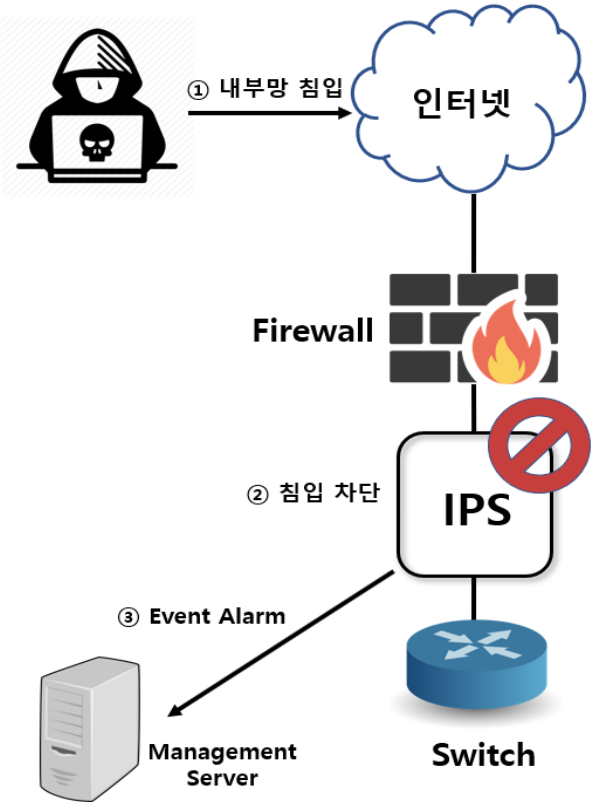
| 구분 | OSI 7 Layer | TCP/IP |
|---|---|---|
| 레이어 |  | |
| 표준화 | ISO 국제 표준(De jure) | 사실상 표준(De fact) |
| 목적 | 네트워크 통신 모델 표준 제시 | 네트워크 통신 구현 |
| 계층 | 7 계층 | 4 계층 |
| 속성 | 표준 참조 모델 | 프로토콜 규약 |

- TCP/IP는 인터넷의 전신인 ARPA Net을 구현하기 위해 개발된 실용 모델이며 OSI 7 Layer 기반으로 실체화

**"끝"**

| 06 | IDS / IPS | | |
|---|---|---|---|
| 문제 | 침입탐지시스템(Intrusion Detection System, IDS)과 침입방지시스템(Intrusion Prevention System, IPS)의 개념을 비교하여 설명하시오. | | |
| 도메인 | 보안 | 난이도 | 하(상/중/하) |
| 키워드 | 탐지, 차단/방어, 사후, 사전, In-line, Mirroring, Tap, 시그니처, Rule, 이상침입탐지(기존 정상 패턴과 비교), 오용침입탐지(알려진 패턴) | | |
| 출제배경 | 네트워크 침입에 따른 탐지/방어 시스템에 대한 기본 이해 | | |
| 참고문헌 | https://wiki.wikisecurity.net/wiki:침해방지_시스템_ips<br>https://ko.wikipedia.org/wiki/침입_탐지_시스템 | | |
| 해설자 | TOP반 유술사 (제 113회 컴퓨터시스템응용기술사 / itpe_you@naver.com) | | |

**I. 침입탐지시스템(IDS)와 침입방지시스템(IPS)의 기본 개념 비교**

| 구분 | 침입탐지시스템(IDS) | 침입방지시스템(IPS) |
|---|---|---|
| 정의 | - 비인가된 사용자가 자원의 무결성(integrity), 기밀성(confidentiality), 가용성(availability)을 저해하는 일련의 행동들과 보안 정책을 위반하는 행위, 즉 침입(intrusion)을 실시간으로 탐지하는 시스템 | - 침입탐지 시스템의 오판(False Positive)와 미탐(Miss Detection)의 문제 해결을 위해 등장한 정보 시스템<br>- 네트워크에서의 침입탐지와 방어가 가능 시스템 |
| 개념도 |  |  |
| 특징 | - Positive, Reactive 탐지<br>- 선 패턴 등록 후 반응 | - Active, Proactive 대응<br>- 공격전 사전 차단 |

**II. 침입탐지시스템(IDS)와 침입방지시스템(IPS)의 탐지/차단 측면 개념 비교**

| 구분 | 침입탐지시스템(IDS) | 침입방지시스템(IPS) |
|---|---|---|
| 목적 | - 침입 여부 탐지 | - 침입방지, 탐지 후 적극적 대응 |
| 분석방법 | - 시그니처 DB기반 패턴 매칭방법<br>- 알려진 공격패턴 감지(CVE) | - Rule DB 기반<br>- 비정상 행위 방지(비지도 학습) |
| 패킷공격 | - 첫번째 패킷 공격의 방어 어려움 | - 공격 방지 가능 |
| Zero-Day Attack | - 방어 곤란 | - 일부 가능 |
| 차단방법 | - Reset Signal, 방화벽 연동 | - 자체 차단 |
| One-Way Attack. | - 탐지 | - 탐지/차단 |
| DDoS, DoS차단 | - 탐지 | - 탐지/차단 |
| Worm Virus | - 탐지 | - 탐지/차단 |
| Ransomware | - 탐지 | - 탐지/차단 |
| 암호화 트래픽 | - 암호화 트래픽 분석 제한<br>- 복호화 트래픽 장비 연동 | - 암호화 트래픽 분석 가능<br>- 사설 인증서 적용 |

**III. 침입탐지시스템(IDS)과 침입방지시스템(IPS)의 구축/운영 측면 개념 비교**

| 구분 | 침입탐지시스템(IDS) | 침입방지시스템(IPS) |
|---|---|---|
| 연결 방법 | - Mirroring<br>- Out of Path | - In-Line<br>- TAP(Test Access Point) |
| 설치 위치 | - 코어 네트워크 스위치 | - 관문 네트워크 스위치<br>- 관문 방화벽 상/하단 |
| 서비스 전환 | - IDS Monitoring 및 트래픽 학습<br>- 오탐 패턴 탐지 예외 | - IDS Mode Monitoring<br>- 오탐 패턴 차단 예외 |
| 대응 방법 | - 관리자에게 경고<br>- 통합보안관리시스템을 통해 방화벽 Rule변경 | - 알려지지 않은 공격 탐지<br>- 자원 접근 차단 |
| 서비스 중단 시 장애극복 | - 서비스 영향 없음 | - FoD(Fail over Device)를 통한 장애대응<br>- Bypass Mode |
| 시스템 부하 | - 시스템 부하 영향 없음 | - 트래픽 처리 지연 발생 |
| 장점 | - 모든 패킷에 대해 자체 탐지 모듈 지원으로 네트워크 이상 경고 | - 모든 패킷에 대해 자체 탐지 및 차단 모듈을 지원으로 네트워크 보호 |
| 단점 | - 방화벽과 연동 방어를 통한 차단 가능(독립적 제한적) | - Transparent mode로 운영되며 NAT등 방화벽 고유 기능지원 불가 |

- IDS는 탐지, IPS는 차단 기능을 제공하며 방화벽과 유사한 기능을 제공하나 차이점 존재

**IV. 침입탐지시스템(IDS)과 침입방지시스템(IPS)과 방화벽 비교**

| 구분 | 침입탐지시스템(IDS) | 침입방지시스템(IPS) | 방화벽(Firewall) |
|---|---|---|---|
| 차단/탐지 | - 탐지 | - 탐지/차단 | - 탐지/차단 |
| OSI Layer | - Layer 3 ~ 7<br>- Application 분석 및 탐지 | - Layer 3 ~ 7<br>- Application 분석 및 차단 | - Layer 3 ~ 4<br>- IP/Port 기반 차단 |
| 주요 기능 | - 시그니처 이상 징후 탐지<br>- 비정상 트래픽 탐지<br>- 바이러스/웜 탐지 | - 패킷 필터링 및 차단<br>- 비지도학습 기반 차단<br>- 바이러스/웜 차단 | - 패킷 필터링 및 차단<br>- 바이러스/웜 차단 불가<br>- NAT/DMZ 제공 |
| 연결 방법 | - Mirroring | - In-Line | - In-Line |
| 장애 대응 | - 서비스 영향 없음 | - Bypass mode | - 이중화(HA 구성) |
| 시스템 부하 | - 부하 없음 | - 트래픽 지연 발생 | - 트래픽 지연 발생 |

- 최근 Application Layer 분석이 가능한 지능형 방화벽 등장으로 IDS/IPS와 상호 보완적 기업 보안 체계 구축

**"끝"**


# ITPE 기술사회
# 제124회 정보처리기술사 기출문제 해설집

**대 상** 정보관리기술사, 컴퓨터시스템응용기술사, 정보통신기술사, 정보시스템감리사 시험
**발행일** 2021년 05월 28일
**집 필** 강정배PE, 안경환PE, 유술사PE, 전일PE, 백기현PE
**출 판** **ITPE(Information Technology Professional Engineer)**
**주 소** ITPE 대치점 서울시 강남구 선릉로 86길 17 선릉엠티빌딩 7층
ITPE 선릉점 서울시 강남구 선릉로 86길 15, 3층 IT교육센터 아이티피이
ITPE 강남점 서울시 강남구 테헤란로 52길 21 파라다이스벤처타워 3층 303호
**연락처** 070-4077-1267 / itpe@itpe.co.kr