# 파이썬 실습1

## 음료수 얼려 먹기 프로그램

- N*M 크기의 얼음 틀이 주어지는 경우
- 0은 얼음틀 구멍을 1은 칸막이를 표현
- 0으로 연결된 묶음을 얼린 음료수 1개로 간주
- 다음 예시 4 * 5 에서는 3개의 얼린 음료수 생성

```
00110
00011
11111
00000
```

| | | | | |
|---|---|---|---|---|
| **0** | **0** | **1** | **1** | **0** |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 |

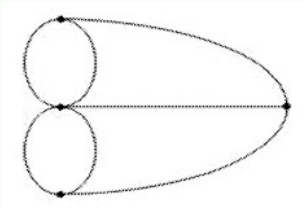
- 첫 번째 입력은 N과 M, 1 ~ 1000 사이의 자연수
- 두 번째 입력은 얼음 틀 형태가 주어짐

# 탐색 알고리즘

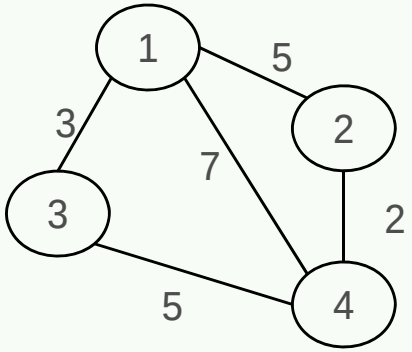
- 탐색이란
  - 많은 양의 데이터 중에서 원하는 데이터를 찾는 과정
  - 순차탐색, 이진탐색, 해시탐색 등의 방법이 있음
  - 순차(Sequential)
    - 왼쪽에서 오른쪽 방향으로 순서대로 하나씩 확인해 나가는 방법
    - 주로 리스트 구조이며 데이터 수가 많아지면 시간이 많이 소요되어 효율이 떨어짐
  - 이진(Binary)
    - 오름차순 또는 내림차순으로 정렬되어 있는 경우에 적합
    - 평균적으로는 순차보다는 빠르지만, 데이터의 양, 데이터 구조, 정렬 상황 등에 따라 선택
  - 해시(Hash)
    - 데이터의 내용과 저장한 곳의 요소를 미리 연계하여 짧은 시간 안에 탐색
    - 해시 함수를 이용하여 데이터가 저장될 곳의 위치를 계산하고 해시함수에 의해 계산된 위치에 해시 테이블을 사용하여 데이터 저장

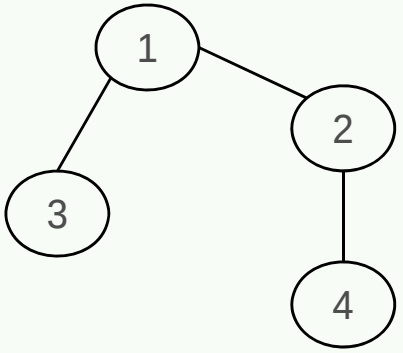
# 탐색 알고리즘

- 데이터가 저장된 형태가 트리 또는 그래프인 경우
  - DFS (Deep First Search), 깊이 우선 탐색
  - BFS (Breadth First Search), 너비 우선 탐색
- 그래프와 트리의 공통점과 차이점
  - 공통
    - 노드(node)와 간선(edge)로 구성
    - 방향성 유무
  - 차이점
    - 그래프는 개체간의 관계를 표현
    - 트리는 관계와 더불어 계층을 표현
      - 가중치가 없으며, 사이클이 존재하지 않음
      - 두 정점간 최단 경로는 단 하나만 존재
      - 간선의 개수는 정점개수 - 1 을 가짐



오일러의 쾨니히스베르그의 다리

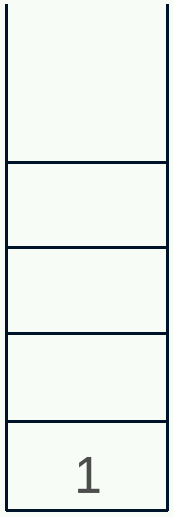


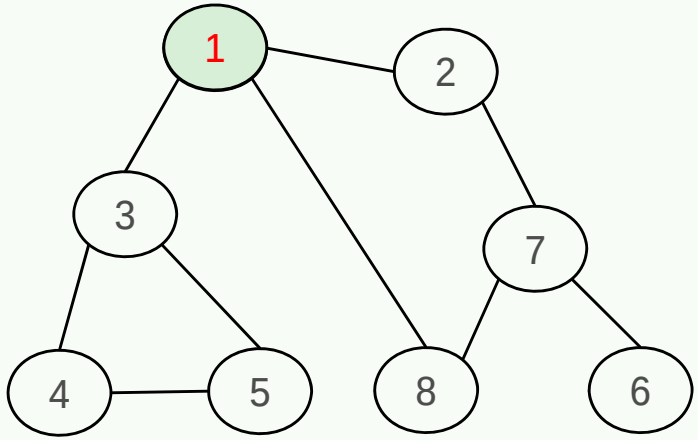



# 탐색 알고리즘 - DFS

## DFS 원리

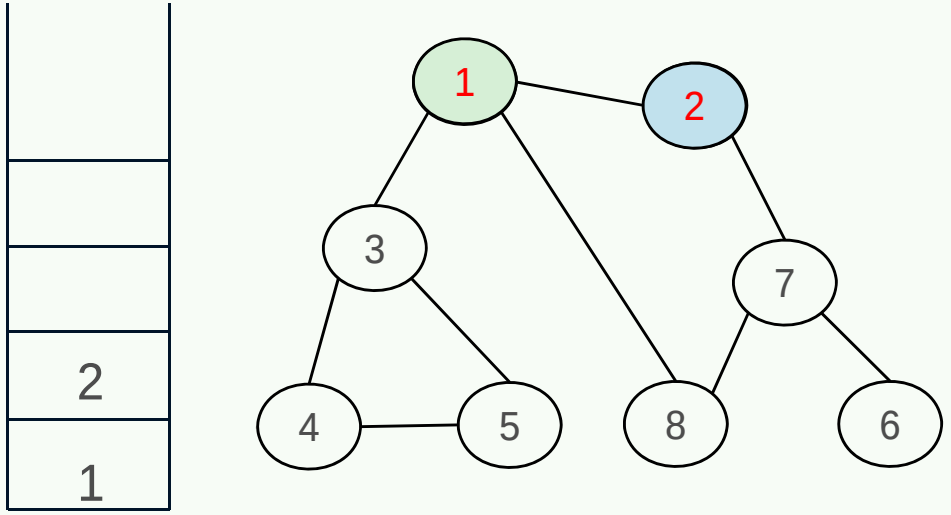
- 스택 자료구조를 이용하여 탐색
  - 탐색 시작 노드를 스택에 삽입하고 방문처리
  - 스택의 최상단 노드에 방문하지 않은 인접 노드가 있으면 그 인접 노드를 스택에 저장
  - 방문하지 않은 인접노드가 없을 시 스택의 최상단 노드를 제거
  - 위의 과정 2 단계를 모든 노드를 방문할 때까지 수행

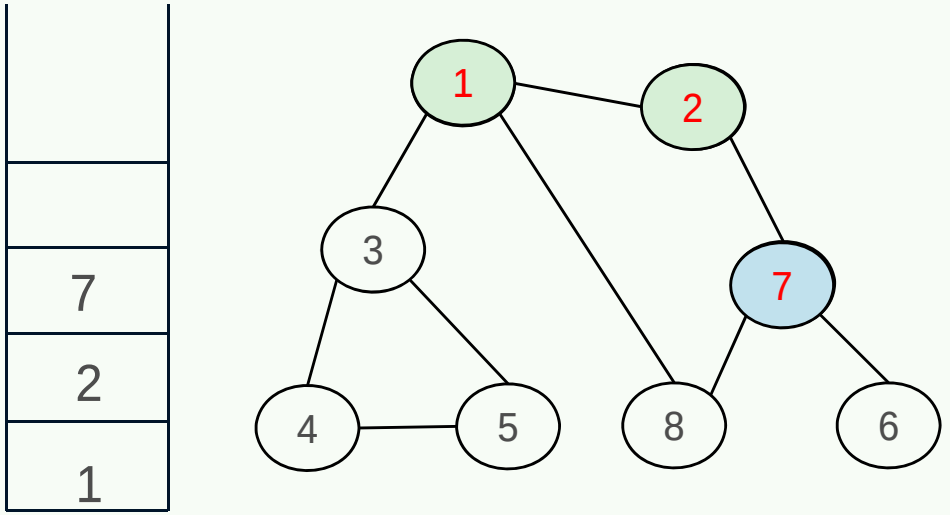


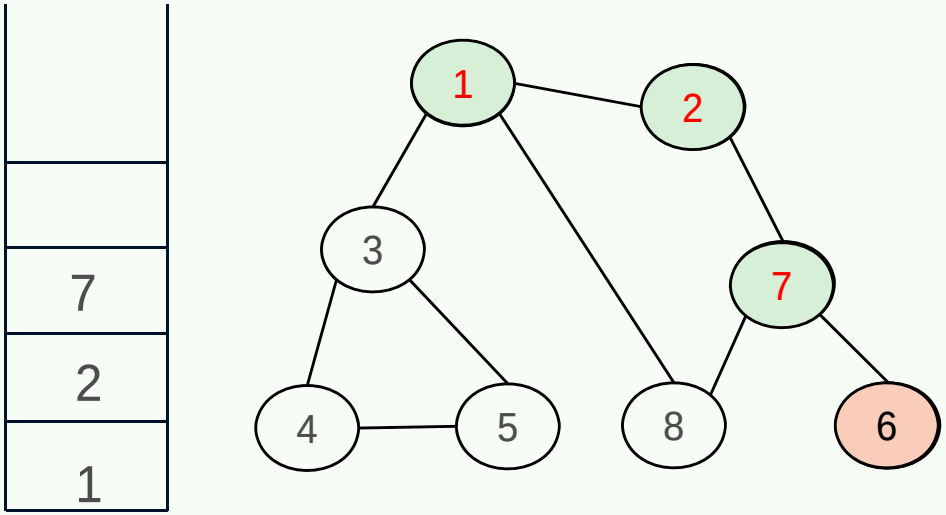



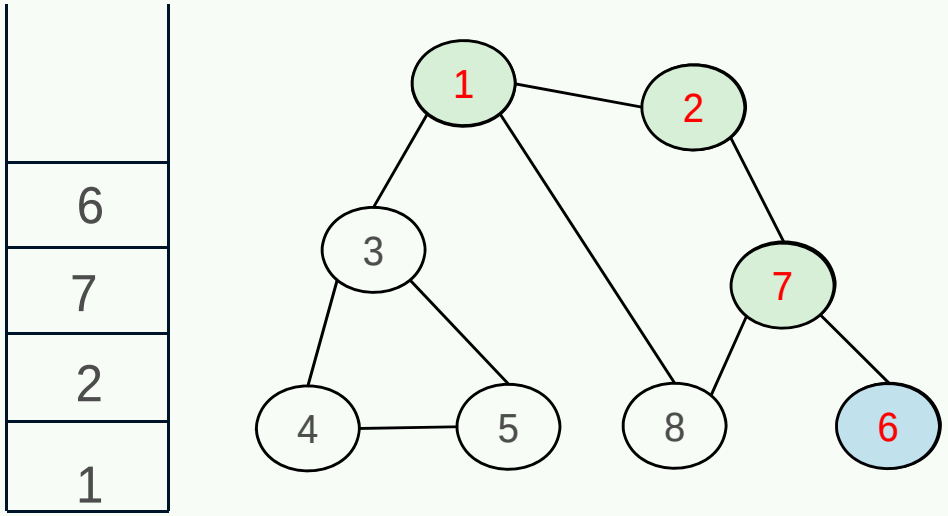

인접노드가 여러 개 일 경우
- 노드값 또는 가중치가 작은 순 /큰 순

# 탐색 알고리즘 - DFS

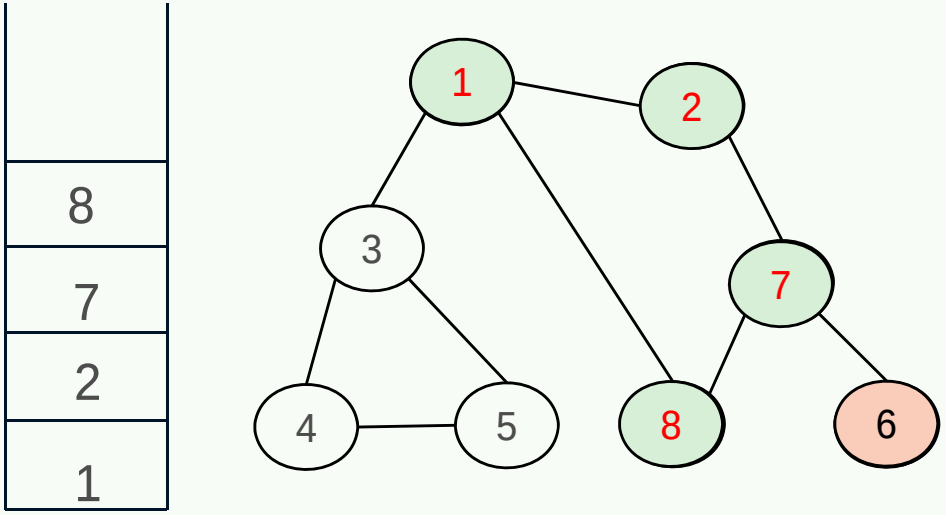


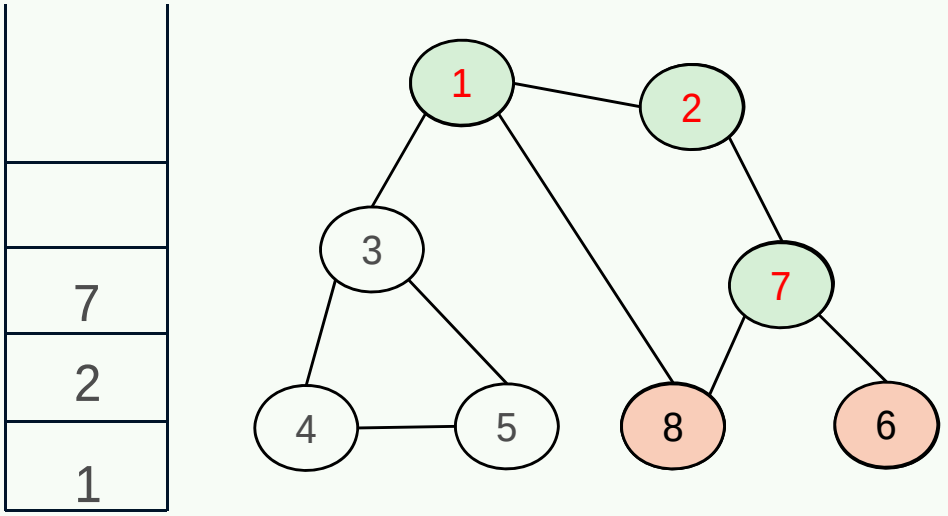



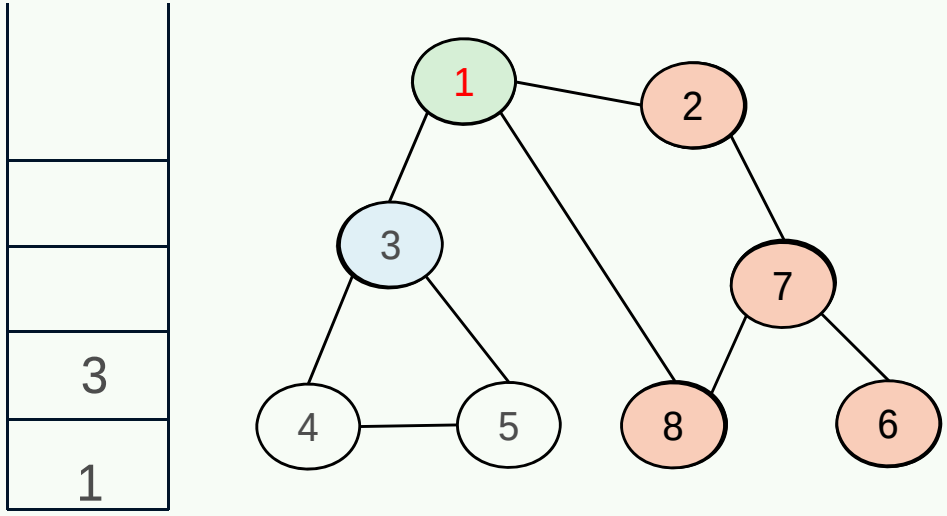



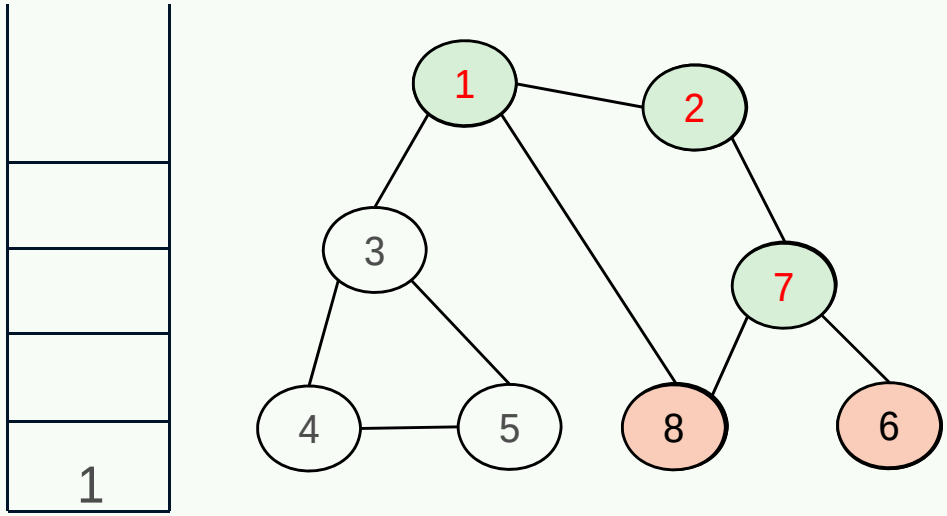



# 탐색 알고리즘 - DFS

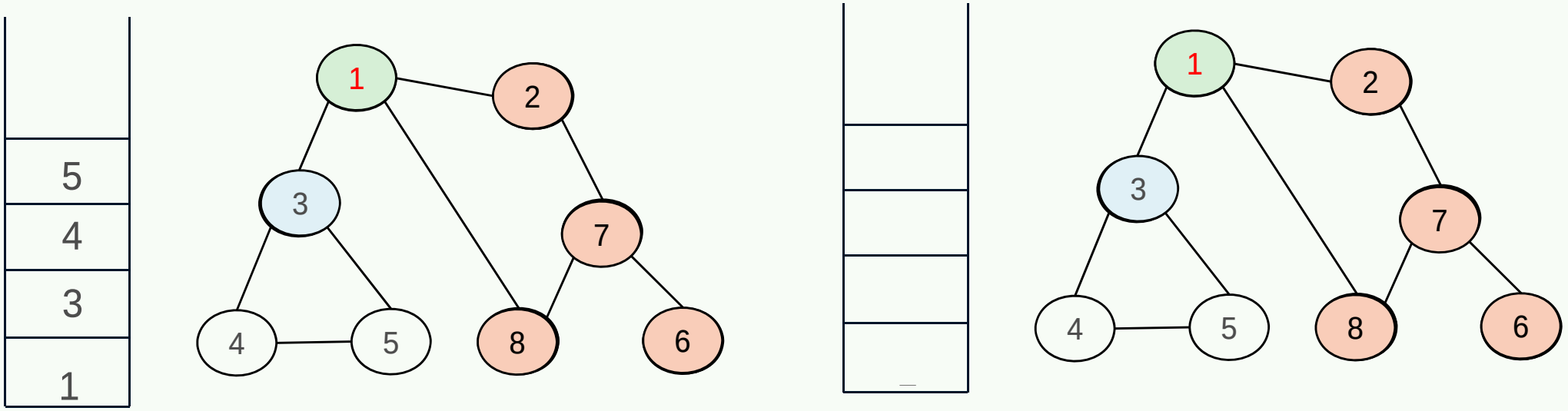


# 탐색 알고리즘 - DFS




노드 방문 순서(스택에 들어간 순서)

1 → 2 → 7 → 6 → 8 → 3 → 4 → 5

# 탐색 알고리즘 - DFS

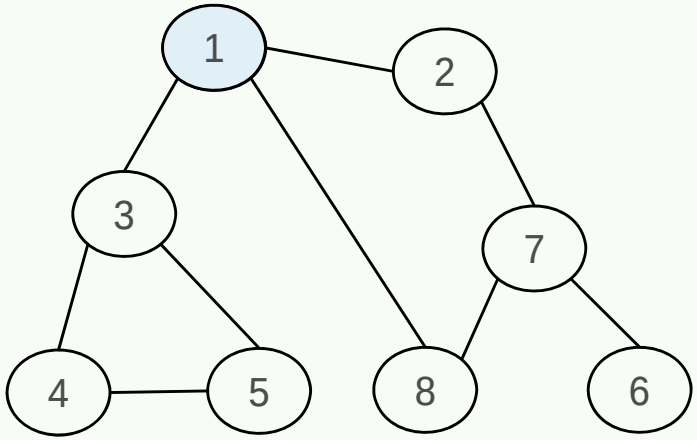
- 파이썬 코드

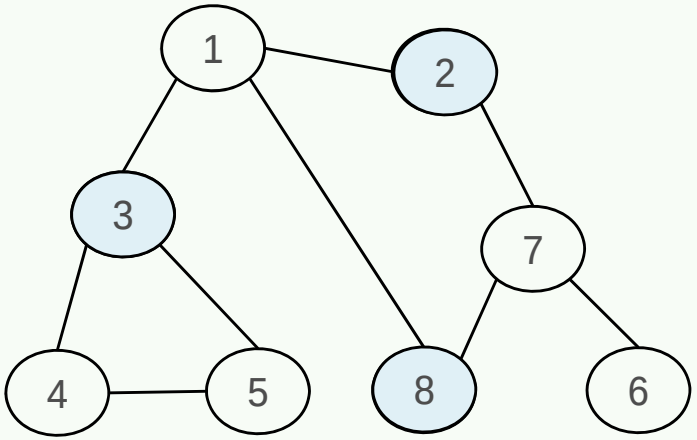
```
def dfs_f(data, visit_n):
    dfs_v[visit_n] = True
    print(visit_n, end='  ')
    for i in data[visit_n]:
        if not dfs_v[i]:
            dfs_f(data,i)


if __name__ == "__main__":
    data = [[],
            [2,3,8],
            [1,7],
            [1,4,5],
            [3,5],
            [3,4],
            [7],
            [2,6,8],
            [1,7],]
    dfs_v = [False]*len(data)
    dfs_f(data, 1)
```

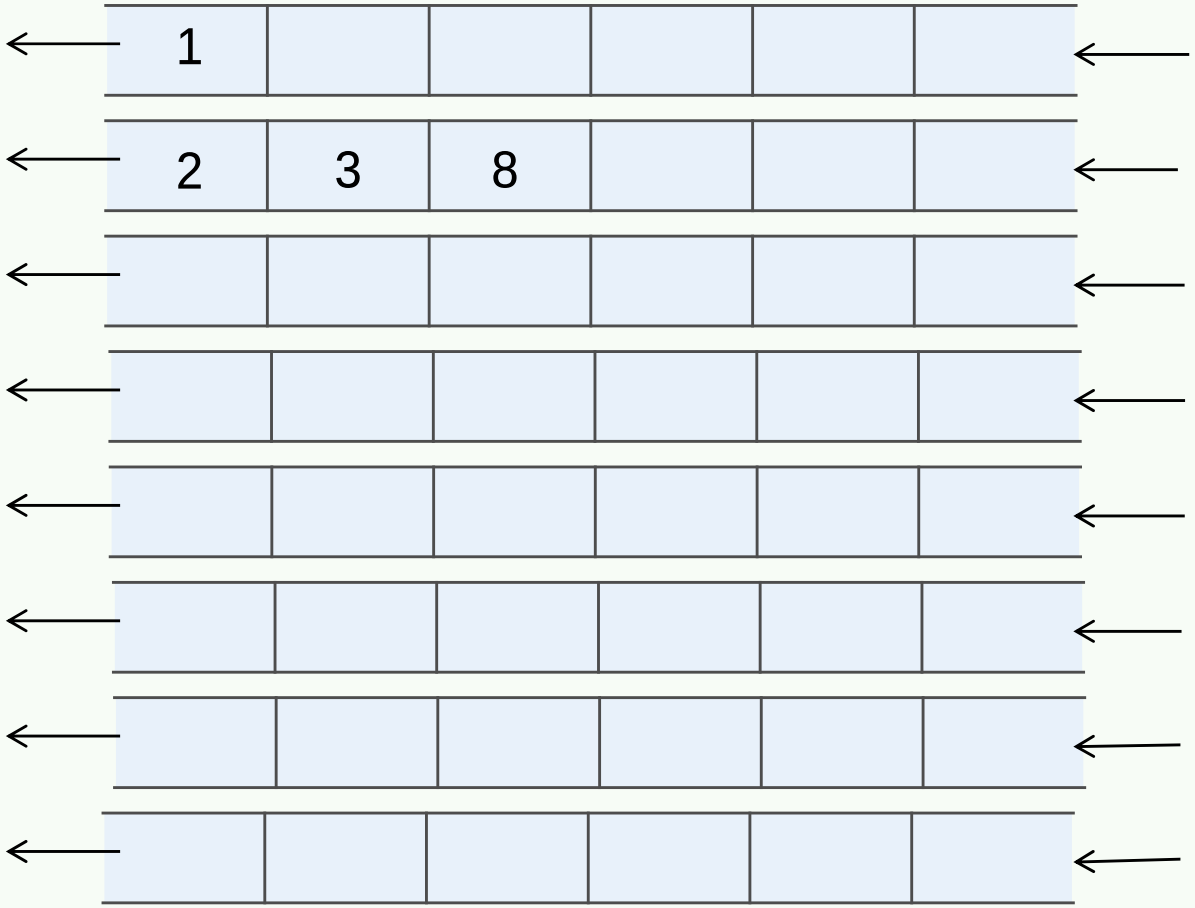
# 탐색 알고리즘 - BFS

- **BFS 원리**
  - DFS가 가장 멀리 있는 노드를 우선으로 탐색하는데 비해 가까운 노드부터 탐색하는 알고리즘
  - 큐 자료구조를 이용
    - 탐색 시작 노드를 큐 구조에 삽입
    - **큐에서 노드를 꺼내고 해당 노드의 인접 노드 중에서 방문하지 않은 노드를 모두 큐에 입력**
    - 바로 위의 과정 한 단계를 큐가 비워질때까지 실행

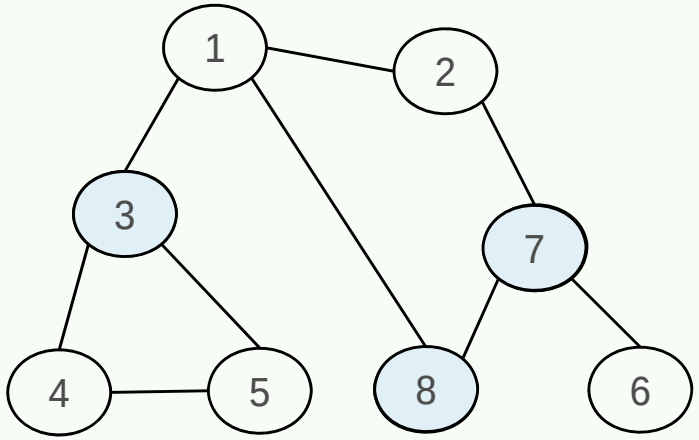
# 탐색 알고리즘 - BFS

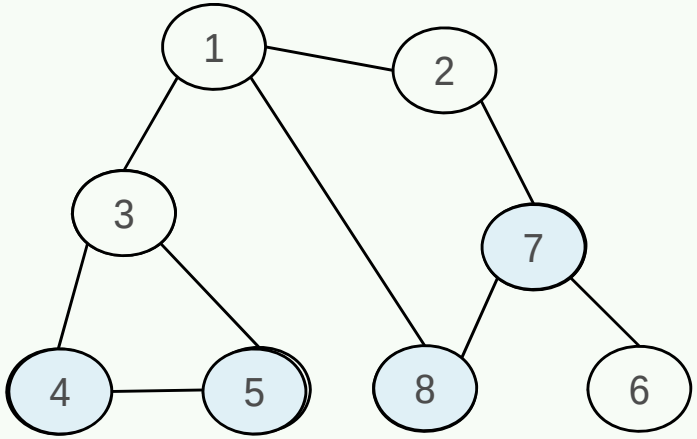


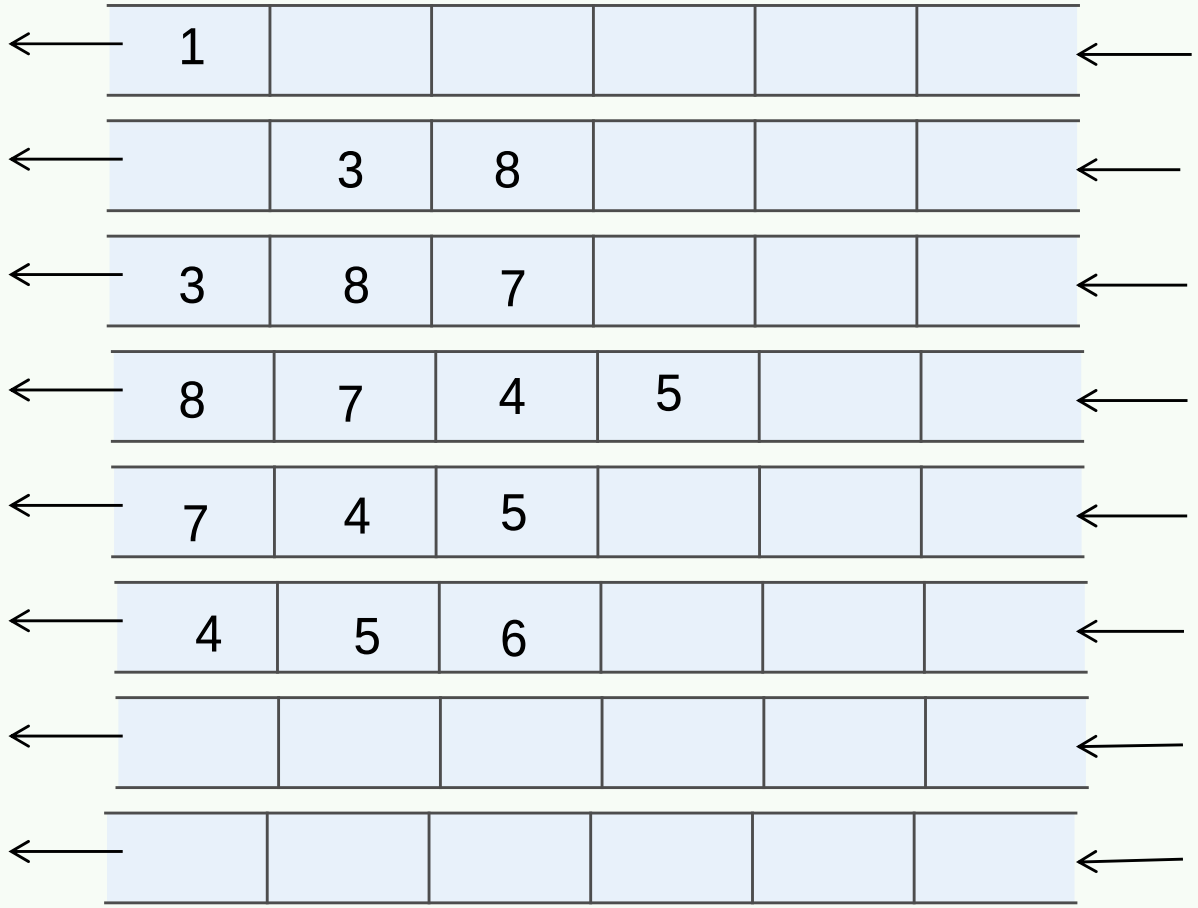



# 탐색 알고리즘 - BFS




노드 방문 순서(큐에서 빠져나온 순서)

1 → 2 → 3 → 8 → 7 → 4 → 5 → 6

# 탐색 알고리즘 - BFS

- 파이썬 코드

```
from collections import deque

def bfs_f(data,start):
    queue = deque([start])
    bfs_visited[start] = True
    while queue:
        n = queue.popleft()
        print(n, end= '  ')
        for i in data[n]:
            if not bfs_visited[i]:
                queue.append(i)
                bfs_visited[i] = True
```

# 파이썬 실습2

- 두 배열의 원소 교체를 통해 최대 합 구하는 프로그램
  - 두 배열의 크기는 N으로 동일하게 생성, N은 1 ~ 10,000 사이의 자연수
  - 두 배열의 원소는 1 ~ 1,000,000 사이의 자연수
  - 두 배열의 교체 횟수는 K로 0 <= K <= N 의 자연수
  - 예시

```
enter array num and change num >> 5 3
first array:  [1, 2, 5, 4, 3]
second array: [5, 5, 6, 6, 5]
max sum : 26
```

# 정규화 연습

- 다음 릴레이션에 정규화를 적용하시오

[학생과목담당강사 릴레이션]

| 학번 | 과목번호 | 주소 | 학년 | 성별 | 학점 | 강사번호 | 강사전공 |
|---|---|---|---|---|---|---|---|
| 2017123 | c001,c003 | 서울 | 4 | 여 | A,A | p01, p02 | 산공,컴공 |
| 2016124 | c002 | 인천 | 4 | 남 | B | p03 | 산공 |
| 2018125 | c001, c004 | 분당 | 3 | 여 | B,A | p01, p05 | 산공,기계 |
| 2018126 | c005 | 수원 | 3 | 남 | C | p06 | 전전 |
| 2019127 | c001,c002 | 서울 | 2 | 여 | B,B | p01,p03 | 산공,산공 |
| 2020128 | c003 | 서울 | 1 | 남 | A | p02 | 컴공 |

# 정규화 연습(2)

- 다음 릴레이션을 보고 정규화를 진행, 최종 릴레이션들을 작성

| 학번 | 이름 | 담임선생님 | 학교이름 | 학교전화번호 | 과목번호 | 성적 | 수업시간 | 회장이름 | 동아리이름 | 방번호 |
|---|---|---|---|---|---|---|---|---|---|---|
| 123 | 길동 | 희동 | 강남 | 555 | C01 | 90 | 2교시 | 영희 | 야구 | r10 |
| 124 | 둘리 | 철수 | 강남 | 555 | C02 | 97 | 1교시 | 서초 | 코러스 | r13 |

- **제1정규형은 만족**
  - 제1정규형일 경우 삽입, 삭제, 갱신의 이상현상 발생
  - 부분 함수종속성 확인

# RDB-mysql

## 데이터베이스 설계:관계데이터 연산

**2020.09. 02.** 수요일

최회련

En-CORE

Data Science Edu.

# 관계 데이터 연산 개념

- 데이터 모델 구성


- 관계 데이터 연산 - 관계 대수 / 관계 해석
  - 새로운 데이터 언어의 유용성 검증의 기준이 되며, 상용화된 관계 데이터베이스에는 실제로 사용되지 않는 개념임
  - 관계 대수(relational algebra)
    - 원하는 결과를 얻기 위해 데이터의 처리 과정을 순서대로 기술하는 절차 언어(procedural language)
  - 관계 해석(relational calculus)
    - 원하는 결과를 얻기 위해 처리를 원하는 데이터가 무엇인지만 기술하는 비절차 언어

# 관계 대수(relational algebra)

- 관계 대수는 릴레이션을 내부적으로 처리하기 위한 연산자들의 집합
- 따라서, 관계 대수의 연산 결과도 릴레이션이 됨 - 관계 대수의 폐쇄 특성
- 관계형 데이터모델의 이론적 언어로써 **SQL** 데이터베이스 언어의 이론적 토대
- 관계 대수는 대표적인 **8**개의 연산자가 있음
  - 일반 집합 연산자(set operation)
  - 순수 관계 연산자(relational operation)

| 관계 대수 연산자 | |
|---|---|
| 일반 집합 연산자 | 순수 관계 연산자 |
| 합집합 ∪ | 셀렉트 σ |
| 교집합 ∩ | 프로젝트 Π |
| 차집합 – | 조인 ⋈ |
| 카티션 프로덕트 × | 디비전 ÷ |

# 관계 대수 - 집합 연산

- 집합 연산**(set operation)**
  - 릴레이션을 튜플 집합 또는 속성 집합으로 간주하여 처리하는 연산들임
  - 합집합, 교집합, 차집합 및 카티션 프로덕트 연산자가 있음
  - 집합 연산 종류

| 연산 | 연산자 기호 | 형식 | 의미 |
|---|---|---|---|
| 합집합 | ∪ | **T1** ∪ R1 | 릴레이션 **T1**과 **R1**의 합집합 튜플 반환 |
| 교집합 | ∩ | **T1** ∩ R1 | 릴레이션 **T1**과 **R1**의 공통 튜플 반환 |
| 차집합 | – | **T1** – R1 | 릴레이션 **T1**에 포함된 **R1**의 튜플 제거한 **T1**만 반환 |
| 카티션 프로덕트 | × | **T1** × R1 | 릴레이션 **T1**과 **R1**의 각 튜플을 모두 서로 연결하여 만들어진 모든 튜플 조합으로 구성된 새로운 튜플 반환 |

# 관계 대수 - 집합 연산

- 집합 연산 제약 조건
  - 피연산자인 릴레이션은 반드시 2개 필요
  - 합집합, 교집합 및 차집합은 피연산자인 2개의 릴레이션이 합병가능(union-compatible) 해야 함
    - 두 릴레이션의 차수가 같아야 함, 속성의 개수가 동일
    - 두 개의 릴레이션에 서로 대응되는 속성의 순서와 도메인이 동일해야 함, 단 도메인이 같으면 속성의 이름은 달라도 됨
    - 연산 적용 후 결과 릴레이션의 속성 이름은 첫 번째 릴레이션의 속성 이름을 따름
  - 카티션 프로덕트 연산은 합병 가능 여부와 상관없이 실행 가능
- 합집합, 교집합 및 차집합 수행이 불가능한 사례 - 도메인이 동일하지 않음

| 고객번호 (int) | 고객이름 (char(30)) | 나이 (int) |
|---|---|---|
| 001 | 김고려 | 23 |
| 003 | 홍길동 | 33 |
| 006 | 이자연 | 29 |

✖ ∪ ∩ −

| 직원번호 (int) | 직원이름 (char(30)) | 직위 (char(30)) |
|---|---|---|
| 1001 | 박산공 | 부장 |
| 2001 | 최공학 | 대리 |
| 2003 | 김경영 | 사원 |

# 관계 대수 - 집합 연산

## 합집합 연산(union)

- 합집합 연산 사례

| 수강번호 | 이름 | 강의실 | 과목 |
|---|---|---|---|
| **k001** | 홍길동 | **4** | 기계학습 |
| **k004** | 김고려 | **2** | R개론 |
| **k005** | 이자연 | **3** | 컴퓨터 |

∪

| 수강번호 | 이름 | 강의실 | 과목 |
|---|---|---|---|
| **k001** | 홍길동 | **4** | 기계학습 |
| **k005** | 이자연 | **3** | 컴퓨터 |
| **k007** | 박산공 | **1** | 기계학습 |

↓

| 수강번호 | 이름 | 강의실 | 과목 |
|---|---|---|---|
| **k001** | 홍길동 | **4** | 기계학습 |
| **k004** | 김고려 | **2** | R개론 |
| **k005** | 이자연 | **3** | 컴퓨터 |
| **k007** | 박산공 | **1** | 기계학습 |

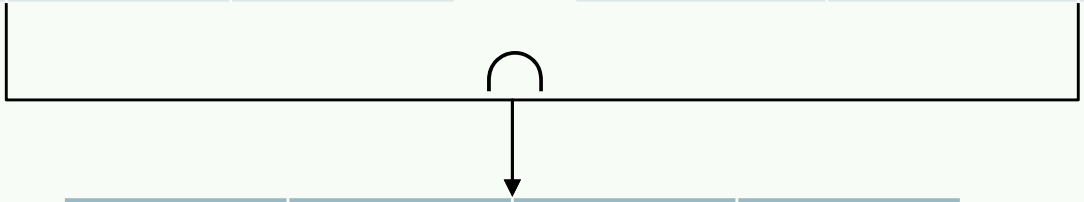
# 관계 대수 - 집합 연산

## 교집합 연산(intersect)

- 교집합 연산 사례

| 수강번호 | 이름 | 강의실 | 과목 |
|---|---|---|---|
| **k001** | 홍길동 | **4** | 기계학습 |
| **k004** | 김고려 | **2** | R개론 |
| **k005** | 이자연 | **3** | 컴퓨터 |

| 수강번호 | 이름 | 강의실 | 과목 |
|---|---|---|---|
| **k001** | 홍길동 | **4** | 기계학습 |
| **k005** | 이자연 | **3** | 컴퓨터 |
| **k007** | 박산공 | **1** | 기계학습 |



| 수강번호 | 이름 | 강의실 | 과목 |
|---|---|---|---|
| **k001** | 홍길동 | **4** | 기계학습 |
| **k005** | 이자연 | **3** | 컴퓨터 |

# 관계 대수 - 집합 연산

## 차집합 연산(difference)

- 차집합 연산 사례

T1

| 수강번호 | 이름 | 강의실 | 과목 |
|---|---|---|---|
| **k001** | 홍길동 | **4** | 기계학습 |
| **k004** | 김고려 | **2** | R개론 |
| **k005** | 이자연 | **3** | 컴퓨터 |

S1

| 수강번호 | 이름 | 강의실 | 과목 |
|---|---|---|---|
| **k001** | 홍길동 | **4** | 기계학습 |
| **k005** | 이자연 | **3** | 컴퓨터 |
| **k007** | 박산공 | **1** | 기계학습 |

—

| 수강번호 | 이름 | 강의실 | 과목 |
|---|---|---|---|
| **k004** | 김고려 | **2** | R개론 |

- S1 – T1의 결과는?

| 수강번호 | 이름 | 강의실 | 과목 |
|---|---|---|---|
| **k007** | 박산공 | **1** | 기계학습 |

# 관계 대수 - 집합 연산

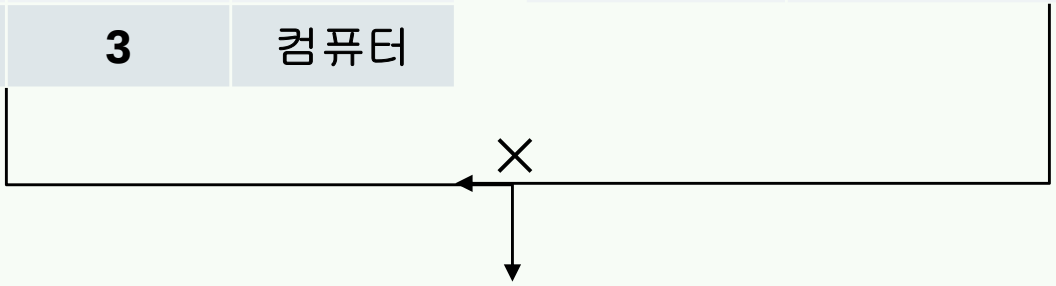
- 카티션 프로덕트 연산**(cartesian product)**
  - 카티션 프로덕트는 두 릴레이션의 모든 튜플간의 수평 결합임
  - 두 릴레이션의 기계적인 조합을 생성하는 연산자이므로, 다른 관계 대수 연산을 조합할 때 유용한 연산자임
  - 카티션 프로덕트 연산 사례

T1

| 수강번호 | 이름 | 강의실 | 과목 |
|---|---|---|---|
| **k001** | 홍길동 | **4** | 기계학습 |
| **k004** | 김고려 | **2** | R개론 |
| **k005** | 이자연 | **3** | 컴퓨터 |

S1

| 과목번호 | 담당강사 | 강의요일 | 강의단계 |
|---|---|---|---|
| **c0010** | **A** | 수요일 | **n001** |
| **c0020** | **B** | 화요일 | **n007** |



# 관계 대수 - 집합 연산

- 카티션 프로덕트 연산 결과

| 수강번호 | 이름 | 강의실 | 과목 | 과목번호 | 담당강사 | 강의요일 | 강의단계 |
|---|---|---|---|---|---|---|---|
| **k001** | 홍길동 | **4** | 산업공 | **c0010** | **A** | 수요일 | **n001** |
| **k001** | 홍길동 | **4** | 산업공 | **c0020** | **B** | 화요일 | **n007** |
| **k004** | 김고려 | **2** | 경영학 | **c0010** | **A** | 수요일 | **n001** |
| **k004** | 김고려 | **2** | 경영학 | **c0020** | **B** | 화요일 | **n007** |
| **k005** | 이자연 | **3** | 컴퓨터 | **c0010** | **A** | 수요일 | **n001** |
| **k005** | 이자연 | **3** | 컴퓨터 | **c0020** | **B** | 화요일 | **n007** |

- 두 릴레이션 속성의 이름이 동일한 경우는 릴레이션이름.속성이름 으로 표기함

# 관계 대수 - 집합 연산

- 집합 연산의 결과 정리( **T1** 연산자 **S1**)

| 연산자 | 구분 | 결과 |
|---|---|---|
| 합집합 | 차수 | **T1** 또는 **S1** 차수와 같음 |
| | 카디널리티 | **T1**과 **S1**의 카디널리티를 더한 수와 같거나 적음 |
| | 교환, 결합 | 성립 |
| 교집합 | 차수 | **T1** 또는 **S1** 차수와 같음 |
| | 카디널리티 | **T1(T1<S1)**과 **S1(S1<T1)**의 카디널리티 보다 적거나 같음 |
| | 교환, 결합 | 성립 |
| 차집합 | 차수 | **T1** 또는 **S1** 차수와 같음 |
| | 카디널리티 | **T1(T1-S1)** 또는 **S1(S1-T1)**의 카디널리티 보다 적거나 같음 |
| | 교환, 결합 | 성립하지 않음 |
| 카티션 프로덕트 | 차수 | **T1**과 **S1**의 차수를 더한 값과 동일 |
| | 카디널리티 | **T1**과 **S1**의 카디널리티를 곱한 값과 동일 |
| | 교환, 결합 | 성립 |

# 관계 대수 - 관계 연산

- 관계 연산**(relation operation)**
  - 릴레이션의 구조와 특성을 이용하는 연산자임
  - 관계형 데이터 모델을 위해 고안된 연산으로, 셀렉트, 프로젝트, 조인 및 디비전이 있음

| 연산 | 연산자 기호 | 형식 | 의미 |
|---|---|---|---|
| 셀렉트 | $\sigma$ | $\sigma_{선택조건식}(R)$ | 릴레이션 **R**에서 **(**선택**)**조건식을 만족하는 튜플 반환 |
| 프로젝트 | $\Pi$ | $\Pi_{속성리스트}(R)$ | 릴레이션 **R**에서 주어진 속성들의 값으로만 구성된 튜플 반환 |
| 조인 | $\bowtie$ | **T1** $\bowtie$ S1 | 공통속성을 이용해 릴레이션 **T1**과 **S1**의 튜플들을 연결하여 만들어진 새로운 튜플들을 반환 |
| 디비전 | $\div$ | **T1** $\div$ S1 | 릴레이션 **S1**의 모든 튜플과 관련 있는 릴레이션 **T1**의 튜플만 반환 |

# 관계 대수 - 관계 연산

## 셀렉트 연산(select)

- 릴레이션에서 주어진 조건을 만족하는 튜플만 선택하여 결과 릴레이션을 구성하는 연산임
- 연산자는 시그마(σ)를 사용하고, 피연산자가 1개 필요한 단항(unary) 연산자임
- 릴레이션을 수평 분할하는 효과가 있음

| 조건식 연산자 | 연산자 기호 | 제약 조건 |
|---|---|---|
| 크다 | **>** | 두 피연산자 속성의 도메인이 동일해야 함 |
| 크거나 같다 | ≥ | |
| 작다 | **<** | |
| 작거나 같다 | ≤ | |
| 같다 | **=** | |
| 같지않다 | ≠ | |
| 논리 연산자 | **∧(and) ∨(or) ¬(not)** | |

# 관계 대수 - 관계 연산

- 셀렉트 연산은 where 문을 사용하여 일반적 데이터 언어 형식으로도 표현 가능
  - 릴레이션 where 조건식
- 셀렉트 연산 예제

$\sigma_{등급='vip'}$ (회원) 또는 회원 where 등급 = 'vip'

$\sigma_{등급='gold' \wedge 나이 > 25}$ (회원) 또는 회원 where 등급 = 'gold' ∧ 나이 **> 25**

회원

| 회원번호 | 회원이름 | 등급 | 나이 | 할인율 | 적립금 |
|---|---|---|---|---|---|
| s001 | 박산공 | vip | 28 | 10 | 50000 |
| s003 | 이고려 | gold | 22 | 5 | 20000 |
| s007 | 김정보 | gold | 26 | 5 | 10000 |

$\sigma_{등급='vip'}$ (회원)

| 회원번호 | 회원이름 | 등급 | 나이 | 할인율 | 적립금 |
|---|---|---|---|---|---|
| s001 | 박산공 | vip | 28 | 10 | 50000 |

$\sigma_{등급='gold' \wedge 나이 > 25}$ (회원)

| 회원번호 | 회원이름 | 등급 | 나이 | 할인율 | 적립금 |
|---|---|---|---|---|---|
| s007 | 김정보 | gold | 26 | 5 | 10000 |

# 관계 대수 - 관계 연산

- 셀렉트 연산은 교환적 특징을 가짐

$$\sigma_{\text{조건식1}}(\sigma_{\text{조건식2}}(\text{릴레이션})) = \sigma_{\text{조건식2}}(\sigma_{\text{조건식1}}(\text{릴레이션})) = \sigma_{\text{조건식1} \wedge \text{조건식2}}(\text{릴레이션})$$

$$\sigma_{\text{적립금}> 20000}(\sigma_{\text{등급='vip'}}(\text{고객})) = \sigma_{\text{등급='vip'}}(\sigma_{\text{적립금}>20000}(\text{고객})) = \sigma_{\text{적립금}>20000 \wedge \text{등급='vip'}}(\text{고객})$$

# 관계 대수 - 관계 연산

- **프로젝트 연산(project)**
  - 릴레이션에서 특정 속성을 추출하는 연산임
  - 연산의 결과 릴레이션이 피연산자인 릴레이션의 일부 열로만 구성되므로 수직적 부분집합을 생성하는 효과가 있음
  - 연산의 표현식 - 수학적 표현과 언어적 표현

    $\Pi_{속성리스트}$(릴레이션)  또는  릴레이션[속성리스트]
  - 연산의 결과는 중복이 제거된 속성의 값으로 나타남
  - 프로젝트는 속성을 연산 대상으로 함
    - 속성리스트에 중복된 속성 이름을 작성할 수 없음

    $\Pi_{이름, 등급}$(회원)

# 관계 대수 - 관계 연산

- 프로젝트 연산 예제

회원

| 회원번호 | 회원이름 | 등급 | 나이 | 할인율 | 적립금 |
|---|---|---|---|---|---|
| s001 | 박산공 | vip | 28 | 10 | 50000 |
| s003 | 이고려 | gold | 22 | 5 | 20000 |
| s007 | 김정보 | gold | 26 | 5 | 10000 |

$\Pi_{\text{이름, 등급}}$(회원)

| 회원이름 | 등급 |
|---|---|
| 박산공 | vip |
| 이고려 | gold |
| 김정보 | gold |

$\Pi_{\text{등급}}$(회원)

| 등급 |
|---|
| vip |
| gold |

# 관계 대수 - 관계 연산

- 조인 연산(join)
  - 릴레이션 하나로 원하는 데이터를 얻을 수 없는 경우에 사용하는 연산
  - 즉, 관계가 있는 여러 릴레이션을 함께 사용해야하는 경우에 사용
  - 조인 연산은 조인 속성(join attribute)을 이용해 두 릴레이션을 조합하여 하나의 결과 릴레이션을 구성함 - 투플의 수평적 결합을 하는 연산임
  - 조인 속성은 두 릴레이션이 공통으로 가지고 있는 속성을 의미함
  - 카티션 프로덕트의 결과 릴레이션 중에서 [조인_조건식]을 만족하는 투플을 셀렉트한 것과 같음
  - 따라서 조인의 대상인 두 릴레이션 안의 속성 이름이 동일한 경우 릴레이션이름.속성이름으로 구별함
  - 연산의 표현식 - 자연조인, 조인_조건식(세타, 동등)

**릴레이션1 ⋈ 릴레이션2** 또는 **릴레이션1** $\bowtie_{\text{조인\_조건식}}$ **릴레이션2**

# 관계 대수 - 관계 연산

- 조인 연산 예제

회원

| 회원번호 | 회원이름 | 등급 | 나이 | 할인율 | 적립금 |
|---|---|---|---|---|---|
| s001 | 박산공 | vip | 28 | 10 | 50000 |
| s003 | 이고려 | gold | 22 | 5 | 20000 |
| s007 | 김정보 | gold | 26 | 5 | 10000 |

주문

| 주문번호 | 주문고객번호 | 주문제품 | 수량 |
|---|---|---|---|
| **p002** | **s001** | 마스크 | **20** |
| **p003** | **s007** | 라면 | **10** |

회원 $\bowtie$ 주문 또는 회원 $\bowtie_N$ 주문

| 회원번호 | 회원이름 | 등급 | 나이 | 할인율 | 적립금 | 주문번호 | 주문제품 | 수량 |
|---|---|---|---|---|---|---|---|---|
| s001 | 박산공 | vip | 28 | 10 | 50000 | p001 | 마스크 | 20 |
| s007 | 김정보 | gold | 26 | 5 | 10000 | p003 | 라면 | 10 |

# 관계 대수 - 관계 연산

- 조인 연산 예제 - 조인 속성이 다수인 경우
  - 조인 속성 값의 쌍이 같은 튜플만 조인 연산에 참여함

**A ⋈ B**

**A**

| a1 | d1 | d2 |
|---|---|---|
| aa1 | dd1 | dd1 |
| ~~aa1~~ | ~~dd2~~ | ~~dd1~~ |
| aa2 | dd1 | dd2 |

**B**

| d1 | d2 | g1 |
|---|---|---|
| dd1 | dd1 | gg1 |
| dd1 | dd1 | gg2 |
| dd1 | dd2 | gg3 |

| a1 | d1 | d2 | g1 |
|---|---|---|---|
| aa1 | dd1 | dd1 | gg1 |
| aa1 | dd1 | dd1 | gg2 |
| aa2 | dd1 | dd2 | gg3 |

# 관계 대수 - 관계 연산

- 세타 조인(theta join)
  - 조인_조건식에 6개의 세타로 구성
  - 세타는 관계 또는 비교 연산자(**>, ≥, <, ≤, = , ≠ )**를 말함
  - 연산 형식

$$\text{릴레이션1} \bowtie_{a\,\theta\,b} \text{릴레이션2}$$

  - 연산 예시

| 릴레이션1 $\bowtie_{A\,\theta\,B}$ 릴레이션2 |
|---|
| $A \bowtie_{a > b} B$ |
| $\bowtie_{a \geq b}$ |
| $\bowtie_{a < b}$ |
| $\bowtie_{a \leq b}$ |
| $\bowtie_{a = b}$ |
| $\bowtie_{a \neq b}$ |

# 관계 대수 - 관계 연산

- 세타 조인 연산 예제

학생

| 학번 | 이름 | 학년 | 전공 |
|---|---|---|---|
| **s0001** | 김고려 | **3** | 컴퓨터 |
| **s0003** | 이산공 | **4** | 산업공 |
| **s0007** | 박정보 | **2** | 기계공 |

수강

| 학번 | 수강번호 | 학점 |
|---|---|---|
| **s0007** | **c01** | **A** |
| **s0001** | **c03** | **B** |
| **s0003** | **c01** | **B** |
| **s0003** | **c04** | **A** |

학생 ⋈ 학생.학번 < 수강.학번 수강

| 학번 | 이름 | 학년 | 전공 | 수강.학번 | 수강번호 | 학점 |
|---|---|---|---|---|---|---|
| **s0001** | 김고려 | **3** | 컴퓨터 | **s0007** | **c01** | **A** |
| **s0001** | 김고려 | **3** | 컴퓨터 | **s0003** | **c01** | **B** |
| **s0001** | 김고려 | **3** | 컴퓨터 | **s0003** | **c04** | **A** |
| **s0003** | 이산공 | **4** | 산업공 | **s0007** | **c01** | **A** |

# 관계 대수 - 관계 연산

- 세타 조인 중에서 특별하게 = 관계를 사용하는 조인을 동등 조인이라 함
- 6개의 세타 조인 중 가장 많이 사용하는 조인이 동등 조인임
- 동등 조인 예제

회원

| 회원번호 | 회원이름 | 등급 | 나이 |
|---|---|---|---|
| s001 | 박산공 | vip | 28 |
| s003 | 이고려 | gold | 22 |
| s007 | 김정보 | gold | 26 |

주문

| 주문번호 | 주문고객번호 | 주문제품 | 수량 |
|---|---|---|---|
| **p002** | **s001** | 마스크 | **20** |
| **p003** | **s007** | 라면 | **10** |

회원 ⋈ 회원번호 = 주문고객번호 주문

| 회원번호 | 회원이름 | 등급 | 나이 | 주문번호 | 주문고객번호 | 주문제품 | 수량 |
|---|---|---|---|---|---|---|---|
| **s001** | 박산공 | **vip** | **28** | **p002** | **s001** | 마스크 | **20** |
| **s007** | 김정보 | **gold** | **26** | **p003** | **s007** | 라면 | **10** |

# 관계 대수 - 관계 연산

- 디비전 연산**(division operation)**
  - T1과 S1의 디비전 연산은 S1의 모든 튜플과 관련 있는 T1의 튜플로 결과 릴레이션을 구성하는 연산임
  - 단, T1이 S1의 모든 속성을 포함하고 있어야 T1 ÷ S1의 연산이 가능함
  - S1의 모든 속성과 도메인이 같은 속성을 T1이 포함하고 있어야 함을 의미함
  - 산술 연산의 나누기와 의미가 비슷함

$$12 \div 2 = (6 \times \mathbf{2}) \div \mathbf{2} = 6$$

# 관계 대수 - 관계 연산

- 디비전 연산 예제

회원

| 회원번호 | 회원이름 | 등급 | 나이 |
|---|---|---|---|
| s001 | 박산공 | vip | 28 |
| s003 | 이고려 | gold | 22 |
| s007 | 김정보 | gold | 26 |

| 등급 |
|---|
| gold |

**gold 등급**

회원 ÷ **gold 등급**

| 회원번호 | 회원이름 | 나이 |
|---|---|---|
| s003 | 이고려 | 22 |
| s007 | 김정보 | 26 |

# 관계대수 - 질의 표현

- 질의에 대한 관계 대수의 예제

회원

| 회원번호 | 회원이름 | 등급 | 나이 | 할인율 | 적립금 |
|---|---|---|---|---|---|
| s001 | 박산공 | vip | 28 | 10 | 50000 |
| s003 | 이고려 | gold | 22 | 5 | 20000 |
| s007 | 김정보 | gold | 26 | 5 | 10000 |

주문

| 주문번호 | 주문고객번호 | 주문제품 | 수량 |
|---|---|---|---|
| **p002** | **s001** | 마스크 | **20** |
| **p003** | **s007** | 라면 | **10** |

(1) 등급이 vip인 회원의 이름과 나이를 검색하시오

$$\Pi_{\text{이름, 나이}} ( \sigma_{\text{등급='vip'}} (\text{회원}) )$$

| 회원이름 | 나이 |
|---|---|
| 박산공 | 28 |

# 관계대수 - 질의 표현

(2) 회원이름이 김정보인 회원의 등급과 김정보 회원이 주문한 제품과 수량을 검색하시오

$\Pi_{등급,\ 주문제품,수량}\ (\ \sigma_{회원이름='김정보'}(\ 회원\ \bowtie\ 주문\ )\ )$

회원 ⋈ 주문

| 회원번호 | 회원이름 | 등급 | 나이 | 할인율 | 적립금 | 주문번호 | 주문제품 | 수량 |
|---|---|---|---|---|---|---|---|---|
| s001 | 박산공 | vip | 28 | 10 | 50000 | p001 | 마스크 | 20 |
| s007 | 김정보 | gold | 26 | 5 | 10000 | p003 | 라면 | 10 |

$\sigma_{회원이름='김정보'}$

| 회원번호 | 회원이름 | 등급 | 나이 | 할인율 | 적립금 | 주문번호 | 주문제품 | 수량 |
|---|---|---|---|---|---|---|---|---|
| s007 | 김정보 | gold | 26 | 5 | 10000 | p003 | 라면 | 10 |

$\Pi_{등급,\ 주문제품,수량}$

| 등급 | 주문제품 | 수량 |
|---|---|---|
| gold | 라면 | 10 |

```
SELECT 등급,주문제품,수량
FROM (회원 join 주문 on 회원.회원번호 = 주문.주문고객번호)
WHERE 회원이름 = '김정보 '
```

# 관계대수 - 확장연산

- 확장연산의 종류

| 연산 | 연산자기호 | 사용형식 | 설명 |
|---|---|---|---|
| 세미 조인 | $\ltimes$ / $\rtimes$ | $T1 \ltimes_{(a1,a2)} S1$<br>$T1 \rtimes_{(a1,a2)} S1$ | 릴레이션 T1과 S1의 자연조인결과에서 한쪽(왼쪽/오른쪽) 릴레이션의 속성만 반환 |
| 외부 조인 | $\bowtie^+$ / $\ltimes^+$ / $\rtimes^+$ | $T1 \bowtie^+_{(a1,a2)} S1$<br>$T1 \ltimes^+_{(a1,a2)} S1$<br>$T1 \rtimes^+_{(a1,a2)} S1$ | 릴레이션 T1과 S1의 자연조인결과에서 조인에 실패한 투플(양쪽/왼쪽/오른쪽)도 널 값으로 채워 결과에 포함하여 반환 |
| 외부 합집합 | $\cup^+$ | $T1 \cup^+ S1$ | 릴레이션 T1과 S1의 투플을 대응하는 속성이 없더라고 널 값으로 채워 모두 결과에 포함하여 반환 |

# 관계대수 - 확장연산

- 세미조인**(semi-join)**
  - 자연조인이 반환하는 결과 릴레이션 중에서 한쪽 릴레이션 속성만으로 한정하여 반환하는 제한적 자연 조인 연산임
  - 왼쪽 세미조인(⋉)
    - 자연조인결과 중 왼쪽 릴레이션의 속성만 반환함

$$T1 \ltimes_{(\text{조인속성리스트})} S1 = T1 \ltimes_{(a1,a2)} S1$$

$$= \Pi_{T1\_\text{속성리스트}}(T1 \bowtie_{N_{(a1,a2)}} S1)$$

    - 왼쪽 세미조인 예제

# 관계대수 - 확장연산

회원

| 회원번호 | 회원이름 | 등급 | 나이 |
|---|---|---|---|
| s001 | 박산공 | vip | 28 |
| s003 | 이고려 | gold | 22 |
| s007 | 김정보 | gold | 26 |

주문

| 주문번호 | 주문고객번호 | 주문제품 | 수량 |
|---|---|---|---|
| **p002** | **s001** | 마스크 | **20** |
| **p003** | **s007** | 라면 | **10** |

회원 $\ltimes_{(\text{회원번호},\text{주문고객번호})}$ 주문

| 회원번호 | 회원이름 | 등급 | 나이 |
|---|---|---|---|
| s001 | 박산공 | vip | 28 |
| s007 | 김정보 | gold | 26 |

# 관계대수 - 확장연산

- 오른쪽 세미조인(⋊)
  - 자연조인결과 중 오른쪽 릴레이션의 속성만 반환함

회원

| 회원번호 | 회원이름 | 등급 | 나이 |
|---|---|---|---|
| s001 | 박산공 | vip | 28 |
| s003 | 이고려 | gold | 22 |
| s007 | 김정보 | gold | 26 |

주문

| 주문번호 | 주문고객번호 | 주문제품 | 수량 |
|---|---|---|---|
| **p002** | **s001** | 마스크 | **20** |
| **p003** | **s007** | 라면 | **10** |

회원 ⋊(회원번호,주문고객번호) 주문

| 주문번호 | 주문고객번호 | 주문제품 | 수량 |
|---|---|---|---|
| **p002** | **s001** | 마스크 | **20** |
| **p003** | **s007** | 라면 | **10** |

# 관계대수 - 확장연산

## 외부조인(outer-join)

- 피 연산자인 두 릴레이션에 자연 조인 연산을 수행 시, 조인 속성값이 같은 투플이 상대 릴레이션에 존재하지 않아 조인 연산에서 제외된 모든 투플을 결과 릴레이션에 포함시키는 연산

회원

| 회원번호 | 회원이름 | 등급 | 나이 |
|---|---|---|---|
| s001 | 박산공 | vip | 28 |
| s003 | 이고려 | gold | 22 |
| s007 | 김정보 | gold | 26 |

주문

| 주문번호 | 주문고객번호 | 주문제품 | 수량 |
|---|---|---|---|
| p002 | s001 | 마스크 | 20 |
| p003 | s007 | 라면 | 10 |

회원 $\bowtie^{+}_{(회원번호,주문고객번호)}$ 주문

| 회원번호 | 회원이름 | 등급 | 나이 | 주문번호 | 주문제품 | 수량 |
|---|---|---|---|---|---|---|
| s001 | 박산공 | vip | 28 | p002 | 마스크 | 20 |
| s007 | 김정보 | gold | 26 | p003 | 라면 | 10 |
| s003 | 이고려 | gold | 22 | NULL | NULL | NULL |

# 관계대수 활용

- 고객과 주문 릴레이션

| 회원번호 | 회원이름 | 주소 | 연락처 |
|---|---|---|---|
| 1 | 손흥민 | 영국 런던 | 555-5555 |
| 2 | 김연아 | 한국 서울 | 222-2222 |
| 3 | 김연경 | 한국 서울 | 333-3333 |
| 4 | 박세리 | 한국 대전 | 777-7777 |

| 주문번호 | 회원번호 | 도서이름 | 가격 | 주문일자 |
|---|---|---|---|---|
| 1 | 2 | 파이썬 | 25000 | 2020-08-30 |
| 2 | 1 | R | 18000 | 2020-09-01 |
| 3 | 2 | 머신러닝 | 22000 | 2020-09-01 |
| 4 | 1 | R | 18000 | 2020-09-02 |
| 5 | 4 | 딥러닝 | 28000 | 2020-09-03 |
| 6 | 5 | 데이터분석 | 35000 | 2020-09-04 |
| 7 | 4 | 데이터분석 | 35000 | 2020-09-04 |

# 관계대수 활용

- 손흥민 고객의 이름, 도서이름과 판매가격을 검색하는 관계대수 작성

  - SQL 질의문

- 도서가격이 **25000**원을 넘는 책의 이름과 구매 고객의 이름

# RDB-mysql

## 데이터베이스 구축: SQL

**2020.09. 02.** 수요일

최회련

En-CORE

Data Science Edu.

# SQL (Structured Query Language)

- 관계 데이터베이스를 위한 표준 질의어로 사용되는 비절차적 데이터 언어
- **1974**년 **SEQUEL (Structured English Query language)**에서 유래
- **1986**년에 표준 질의어로 채택된 후, 표준안 작업을 지속적으로 행함
- **SQL-86**부터 **2011**년에 개정된 표준안인 **SQL:2011**을 거쳐 **SQL:2019** 등 지속적인 표준안 발표
- **SQL** 사용 형태
  - 데이터베이스 관리 시스템에 직접 접근하여 대화식의 질의를 작성하면서 사용
  - C++, 파이썬, 자바 등과 같은 프로그래밍언어로 작성된 응용 프로그램에 삽입하여 사용
- 각 회사의 **DBMS**의 특징에 따라 **SQL**이 조금씩 다름
  - ORACLE에서는 PL/SQL, SQL Server에서는 T-SQL, MySQL에서는 SQL로 부름

# SQL (Structured Query Language)

- **SQL 분류**
  - 데이터 정의어(DDL, Data Definition Language)
    - 테이블을 생성하고 변경 및 제거하는 기능 제공
    - CREATE. ALTER, DROP etc.
  - 데이터 조작어(DML, Data Manipulation Language)
    - 테이블에 새 데이터 삽입, 테이블에 저장된 데이터의 수정과 삭제, 검색하는 기능 제공
    - SELECT, INSERT, DELETE, UPDATE etc.
  - 데이터 제어어(DCL, Data Control Language)
    - 보안을 위해 데이터에 대한 접근, 사용 권한을 사용자별로 부여하거나, 취소하는 기능 제공
    - 데이터베이스 관리자가 주로 사용
    - GRANT, REVOKE etc.

# MySQL

- 오픈소스 관계형 **DBMS**
- **1995**년 스웨덴의 **MySQL AB** 사에 의하여 운영
- **2008**년 썬 마이크로시스템즈에 인수
- 썬 마이크로시스템즈가 **2010**년 오라클에 인수됨
- 현재 오라클에서 운영하고 있으며, **MySQL8.0** 버전이 사용되고 있음
- 구글, 페이스북, 트위터, 유튜브 등에서 사용되고 있음
- **MySQL**의 설치파일은 **https://dev.mysql.com/downloads/mysql/**에서 다운
- 정상적 설치 후 **MySQL** 실행은
  - GUI 방식인 MySQL Workbench 또는 명령어 방식인 MySQL Command Line Client으로 접속

# SQL - 데이터 정의어(DDL)

- 데이터를 저장할 테이블의 구조 생성
- **CREATE(생성), ALTER(구조 변경), DROP(구조 삭제)**
- **CREATE문**
  - 테이블 구성, 속성과 속성에 관한 제약 정의
  - 기본키 및 외래키 정의하는 명령문
  - 문법 형식

```
CREAT TABLE [테이블 이름]
(    {속성이름 데이터 타입
     [NULL | NOT NULL | UNIQUE | DEFAULT 기본값| CHECK 체크조건]
     }
     [PRIMARY KEY 속성이름(들)]
     [UNIQUE 속성이름(들)]
     [FOREIGN KEY 속성이름 REFERENCES 테이블이름(속성이름)]
          [ON DELETE {CASCADE | SET NULL}]
);
```

  - { } : 안의 내용 반복 가능, [ ] : 선택적 사용 , | : 1개 선택 의미

# SQL - 데이터 정의어(DDL)

- **Create Table**에 사용되는 정의
  - UNIQUE(속성이름 A) : A를 대체키로 지정
    - 대체키는 기본키와 같이 각 투플을 유일하게 식별하는 특성이 있음
    - 대체키로 지정된 속성의 값은 테이블에서 중복 안되며 유일성을 가져야함
    - 기본키와의 차이점은 NULL 값을 가질 수 있음
    - 또한 한 테이블에 여러 개의 대체키를 허용
  - FOREIGN Key 는 반드시 기본키로 사용되는 테이블을 명시해야 함
    - 참조 무결성의 제약조건을 유지하기 위해 REFERENCE 키워드 다음에 참조 테이블을 명시
    - FOREIGN KEY 선정 시의 제약 조건
      - ON DELETE NO ACTION: 투플을 변경하지 못하게 함
      - ON DELETE CASCADE: 관련 투플에서 외래키 값을 함께 변경
      - ON DELETE SET NULL: 관련 투플의 외래키 값을 NULL로 함
      - ON DELETE SET DEFAULT: 관련 투플의 외래키 값을 미리 지정한 기본값으로 변경

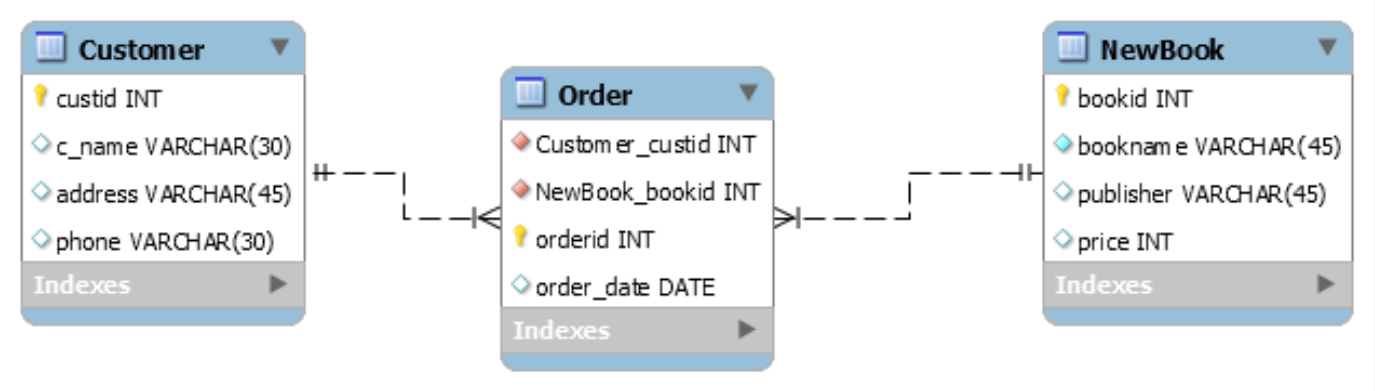
# SQL - 데이터 정의어(DDL)

- 속성의 데이터 타입

| 데이터 타입 | 설명 |
|---|---|
| INT / INTEGER | 정수 (기본 4byte) |
| SMALLINT | INT보다 작은 정수 |
| CHAR(n) / CHARACTER(n) | 길이가 n인 고정 길이의 문자열 |
| VARCHAR(n) / CHARACTER VARYING(n) | 최대 길이가 n인 가변 길이의 문자열 |
| NUMERIC(p,s) / DECIMAL(p,s) | 고정 소수점 실수, p는 소수점을 제외한 전체 숫자의 길이이며, s는 소수점 이하 숫자의 길이 |
| FLOAT(n) | 길이가 n인 부동 소수점 실수 |
| REAL | 부동 소수점 실수 |
| DATE | 연.월.일로 표현되는 날짜 |
| TIME | 시.분.초로 표현되는 시간 |
| DATETIME | 날짜와 시간 |

# SQL - 데이터 정의어(DDL)

- 서점정보시스템 구축(도서, 고객, 주문 등)
  - 도서개체유형, 고객개체유형은 다대다의 주문관계로 표현 함
  - 논리 스키마로 변형 시 다대다의 관계는 관계성릴레이션을 생성함




  - 논리 스키마를 데이터베이스로 구축 - 릴레이션을 테이블로 작성

# SQL - 데이터 정의어(DDL)

- 서점정보시스템 구축(도서, 고객, 주문 등) 중에서 도서 테이블 생성 예제
  - 테이블 이름: NewBook
  - 테이블 속성 및 데이터 타입: 도서번호(INTEGER), 도서이름(VARCHAR(20)), 출판사(VARCHAR(20)), 가격(INTEGER)

```
CREAT TABLE NewBook
(
        bookid          INTEGER,
        bookname        VARCHAR(20),
        publisher       VARCHAR(20),
        price           INTEGER,
        PRIMARY KEY     (bookid)
);
```

```
CREAT TABLE NewBook
(
    bookid          INTEGER     PRIMARY KEY,
    bookname        VARCHAR(20),
    publisher       VARCHAR(20),
    price           INTEGER,
);
```

  - 만약 primary key 가 단일이 아닌 두 개 속성의 조합이라면?
    - bookid 가 없고, bookname과 publisher가 primary key가 되는 경우?

# SQL - 데이터 정의어(DDL)

```
CREAT TABLE NewBook
(
        bookname        VARCHAR(20),
        publisher       VARCHAR(20),
        price           INTEGER,
        PRIMARY KEY     (bookname,publisher)
);
```

- 제약 사항을 추가하는 경우?
  - bookname은 NULL을 가질 수 없고, publisher는 동일한 데이터가 있으면 안됨,
  - price에 값이 입력되지 않는 경우는 기본값으로 10,000을 지정, 또한 최소값은 3,000으로 지정함

```
CREAT TABLE NewBook
(
        bookname        VARCHAR(20)     NOT NULL,
        publisher       VARCHAR(20)     UNIQUE,
        price           INTEGER  DEFAULT 10000 CHECK(price>=3000),
        PRIMARY KEY     (bookname,publisher)
);
```

# SQL - 데이터 정의어(DDL)

- 서점정보시스템 구축(도서, 고객, 주문 등) 중에서 고객 테이블 생성 예제

```
CREAT TABLE Customer
(
    custid          INTEGER       PRIMARY KEY,
    c_name          VARCHAR(30),
    address         VARCHAR(45),
    phone           VARCHAR(30),
);
```

- 주문 테이블 생성

```
CREAT TABLE Order
(
    orderid         INTEGER       PRIMARY KEY,
    order_date      DATE,
    custid          INTEGER,
    bookid          INTEGER,
    FOREIGN KEY (custid) REFERENCES Customer(custid) ON DELETE CASCADE
    FOREIGN KEY (bookid) REFERENCES NewBook(bookid) ON DELETE CASCADE
);
```

# SQL - 데이터 정의어(DDL)

## ALTER 문

- 생성된 테이블의 속성과 속성에 관한 제약 변경
- 기본키 및 외래키 변경
- 문법 형식

```
ALTER TABLE  테이블 이름
            [ADD    속성이름   데이터 타입];
            [DROP  COLUMN   속성이름 ] ;
            [MODIFY COLUMN   속성이름   데이터 타입];
            [MODIFY COLUMN 속성이름  [NULL | NOT NULL]];
            [ADD PRIMARY KEY(속성이름)] ;
            [[ADD | DROP] 제약이름] ;
```

- DROP 은 테이블 삭제에도 사용
  - DROP TABLE NewBook;

# SQL - 데이터 정의어(DDL)

- NewBook 테이블에 writer, VARCHAR(30)을 추가

```
ALTER TABLE NewBook ADD writer VARCHAR(30);
```

```
CREAT TABLE NewBook
(
    bookid  INTEGER       PRIMARY KEY,
    bookname              VARCHAR(20),
    publisher             VARCHAR(20),
    price                 INTEGER,
);
```

- NewBook 테이블에 price, SMALLINT로 변경

```
ALTER TABLE NewBook MODIFY price SMALLINT;
```

- NewBook 테이블에 writer 삭제

```
ALTER TABLE NewBook DROP COLUMN writer;
```

- NewBook 테이블의 bookid 속성에 NOT NULL 제약조건 추가

```
ALTER TABLE NewBook MLDIFY bookid INTEGER NOT NULL;
```

# SQL - 데이터 조작어(DML)

- **SELECT문, 질의어(query)라고도 함**
  - 기본 형식

```
SELECT  [ALL | DISTINCT] 속성 리스트
FROM    테이블 리스트;
```

  - SELECT: 질의 결과 추출되는 속성 리스트를 열거
  - FROM : 질의에 이용되는 테이블 리스트를 열거
  - 예시

```
SELECT  bookname, price
FROM    NewBook;
```

  - bookname을 중복에 상관없이 결과를 모두 표시

```
SELECT  ALL bookname
FROM    NewBook;
```

# SQL - 데이터 조작어(DML)

- bookname을 중복없이 표시

```
SELECT  DISTINCT bookname
FROM    NewBook;
```

- price에 대한 부가세를 계산하고, 이 결과를 tax로 출력 (tax라는 속성이 실제 저장되는 것은 아님)

```
SELECT  bookname, price*0.1 AS “tax”
FROM    NewBook;
```

- 조건 추가 시

```
SELECT  [ALL | DISTINCT] 속성 리스트
FROM    테이블 리스트
[WHERE   조건];
```

# SQL - 데이터 조작어(DML)

- 예시, 가격이 25000원 이상의 책 이름에 대한 결과

```
SELECT  bookname, price
FROM    NewBook
WHERE price >= 25000;
```

- 조건에 사용되는 연산자

| 연산자 | 설명 | 연산자 | 설명 |
|---|---|---|---|
| = | 같다 | AND | 모든 조건을 만족하는 경우 검색 |
| <> | 다르다 | OR | 여러 조건 중 한 가지만 만족해도 검색 |
| < | 작다 | NOT | 조건을 만족하지 않는 것만 검색 |
| > | 크다 | BETWEEN | 범위의 검색, price BETWEEN 1000 AND 3000 |
| <= | 작거나 같다 | LIKE | 패턴 연산으로 검색조건이 부분적이고, 문자열을 이용하는 조건에 사용 |
| >= | 크거나 같다 | IN, NOT IN | 집합연산, IN은 집합의 원소인지 판단, 하나라도 포함되면 검색 |
| IS NULL | 속성값의 NULL여부 | IS NOT NULL | 속성값이 NULL아 아닌 여부 |

# SQL - 데이터 조작어(DML)

- 예시, 출판사이름이 “고려“ 이며 가격이 3만원 이상인 책이름

```
SELECT  bookname
FROM    NewBook
WHERE publisher=“고려” AND  price >= 30000;
```

- 가격이 아직 입력되지 않은 책이름과 출판사 검색

```
SELECT  bookname, publisher
FROM    NewBook
WHERE  price IS NULL;
```

- LIKE 연산 검색에 사용되는 기호

| 기호 | 설명 |
|---|---|
| % | 0개 이상의 문자(문자 내용과 개수는 상관없음) |
| _ | 1개의 문자(문자 내용은 상관없음) |

# SQL - 데이터 조작어(DML)

- LIKE 기호 사용 예

| 기호 사용 예 | 설명 |
|---|---|
| LIKE '관계%' | 관계로 시작하는 문자열(길이는 상관 없음) |
| LIKE '%관계' | 관계로 끝나는 문자열 |
| LIKE '%관계%' | 관계가 포함된 문자열 |
| LIKE '관계_ _ _' | 관계로 시작하는 5자 길이의 문자열 |
| LIKE '_ _관%' | 세번째 글자가 관인 문자열 |

- 책이름이 시작이 파로 시작하는 책이름과 출판사 및 가격 출력

```
SELECT  bookname, publisher, price
FROM    NewBook
WHERE  bookname LIKE   ' 파%';
```

- 출판사이름이 7글자인 출판사이름 출력

```
SELECT   publisher
FROM    NewBook
WHERE  publisher LIKE  '_ _ _ _ _ _ _';
```

# SQL - 데이터 조작어(DML)

## 정렬검색, ORDER BY

- 형식

```
SELECT  [ALL | DISTINCT] 속성 리스트
FROM    테이블 리스트
[WHERE   조건]
[ORDER BY 속성 리스트 [ASC | DESC] ];
```

- 예시, 가격의 오름정렬 순으로 출력

```
SELECT  *
FROM    NewBook
ORDER BY price ;
```

- 출판사이름으로 오름정렬하고, 동일한 출판사는 책이름으로 내림정렬한 순으로 출력

```
SELECT  *
FROM    NewBook
ORDER BY publisher ASC,  bookname DESC;
```

# SQL - 데이터 조작어(DML)

- 집계함수(aggregate function) 이용 검색
  - 특정 속성값을 통계적으로 계산한 결과를 검색하기 위한 함수
  - 열함수라고도 하며, 개수, 합계, 평균, 최댓값, 최솟값의 계산 기능을 제공함
  - 집계함수는 NULL 인 속성값은 제외하고 계산
  - 집계함수는 WHERE 절에서는 사용할 수 없음, SELECT 또는 HAVING 절에서만 사용
  - 자주사용되는 집계함수

| 집계함수 | 설명 | 사용 가능한 속성 타입 |
|---|---|---|
| COUNT | 속성 값의 개수 | 모든 데이터 |
| MAX | 속성 값의 최댓값 | |
| MIN | 속성 값의 최솟값 | |
| SUM | 속성 값의 합계 | 숫자 데이터 |
| AVG | 속성 값의 평균 | |

# SQL - 데이터 조작어(DML)

- 예제, 책 가격의 평균을 검색

```
SELECT  AVG(price)
FROM    NewBook;
```

- 출력 시 컬럼이름을 average of books 로 하고 싶다면?

```
SELECT  AVG(price) AS “average of books”
FROM    NewBook;
```

- 책이름 컬럼 수를 출력

```
SELECT  COUNT(bookname) AS “num of book”
FROM    NewBook;
```

- 다음 구문의 결과는?

```
SELECT  COUNT(*)
FROM    NewBook;
```

# SQL - 데이터 조작어(DML)

- 출판사 이름의 중복을 없애고 전체 출판사 개수를 출력

```
SELECT  COUNT(DISTINCT publisher) AS “num of publisher”
FROM    NewBook;
```

## 그룹별 검색, GROUP BY

- 한 테이블에서 특정 속성의 값이 같은 투플을 모아 그룹을 생성 후
- 그룹별로 검색하기 위해 사용
- 그룹에 대한 조건을 추가하려면 GROUP BY 키워드를 HAVING 키워드와 함께 사용하면 됨
- GROUP BY 키워드가 없는 SELECT 문은 테이블 전체를 하나의 그룹으로 보고 검색하는 것임

# SQL - 데이터 조작어(DML)

- 형식

```
SELECT  [ALL | DISTINCT] 속성 리스트
FROM    테이블 리스트
[WHERE   조건]
[GROUP BY 속성 리스트 [HAVING 조건] ]
[ORDER BY 속성 리스트 [ASC | DESC] ];
```

- 예시, 출판사별 책 개수를 출력

```
SELECT  publisher, COUNT(*) AS “num of book”
FROM    NewBook
GROUP BY publisher ;
```

- 출판사별 책 개수와 그 책 중에서 제일 비싼 가격을 검색, 단 책 개수는 num of books 로 가격은 max price로 출력

```
SELECT  publisher, COUNT(*) AS “num of book”, MAX(price) AS “max price”
FROM    NewBook
GROUP BY publisher ;
```

# SQL - 데이터 조작어(DML)

- 책을 2권이상 출판한 출판사별로 책의 개수와 가장 비싼 단가를 검색