# Singly Linked-List

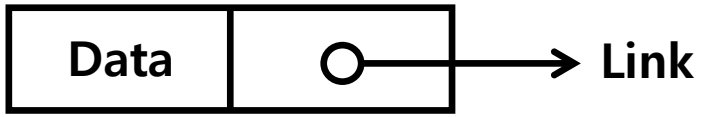
## 리스트의 개념

리스트(Linked List)는 자료를 순서대로 연결하여 저장하는 자료구조이며,

주로 포인터를 이용하여 리스트를 연결한다

## 단순연결리스트

단순연결 리스트는 다음과 같이 Data와 Link를 갖는 Node로 구성한다

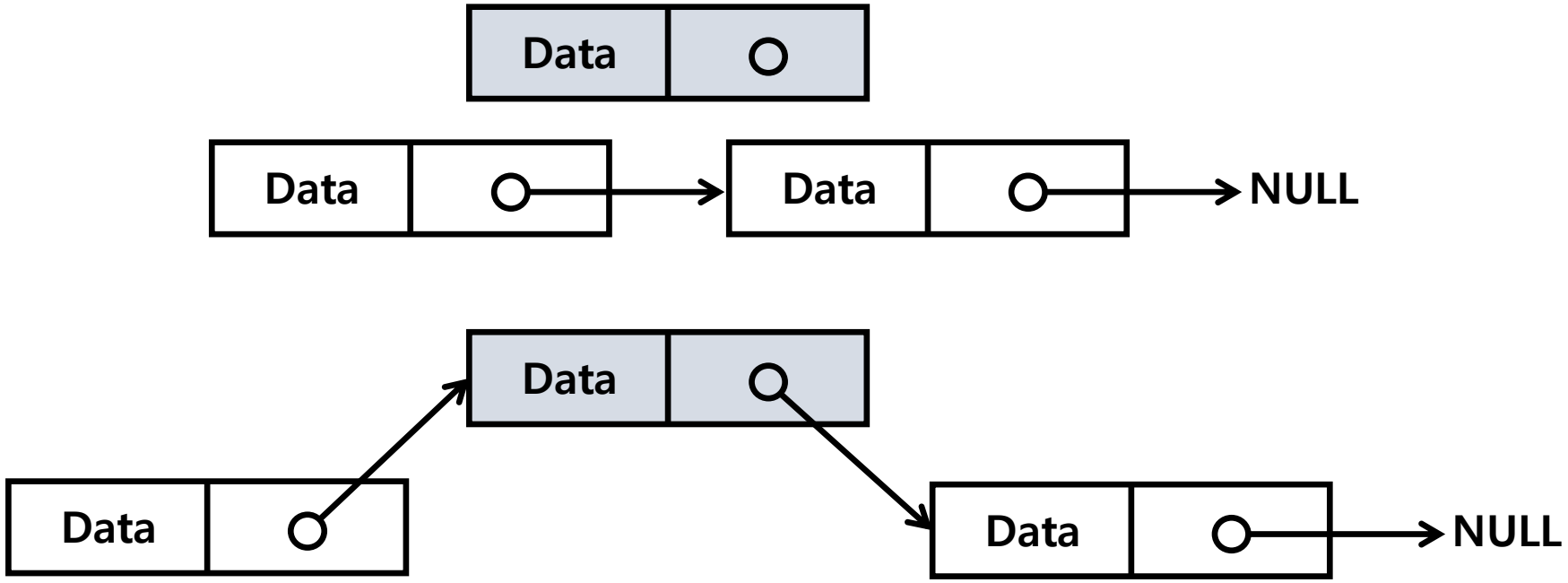



위의 Node를 포인터로 연결하여 연결리스트를 구성한다

## Singly Linked-List

### 단순연결리스트의 데이터 삽입

2번째 위치에 새로운 Node를 추가하는 방법은 다음과 같다




### 단순연결리스트의 특징

**장점 : 배열과 달리 Insert 동작에서 나머지 Node를 이동할 필요가 없다**

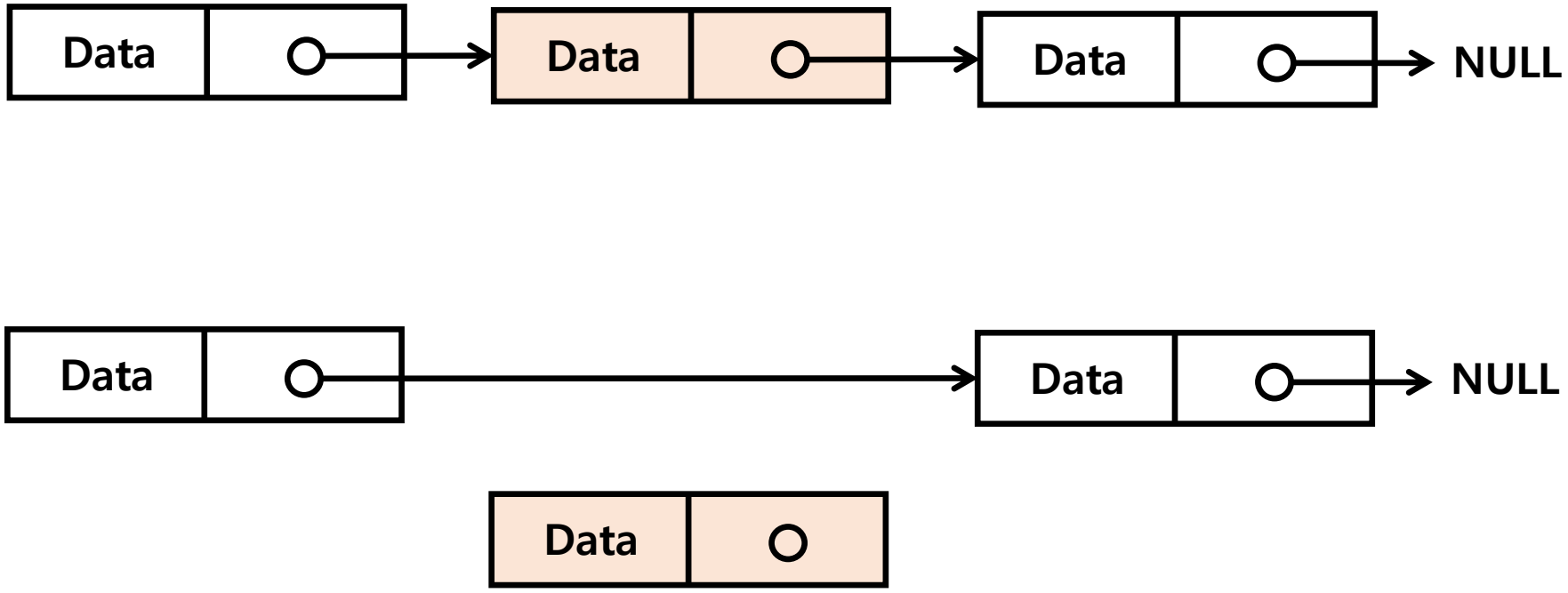
**단점 : 특정 위치의 자료에 임의 접근이 불가능하다 (포인터로 순차탐색 필요)**

# Singly Linked-List

## 단순연결리스트의 데이터 삭제

2번째 위치의 Node를 삭제하는 방법은 다음과 같다

**(단, 삭제한 Node는 반드시 동적할당을 취소해야 함)**

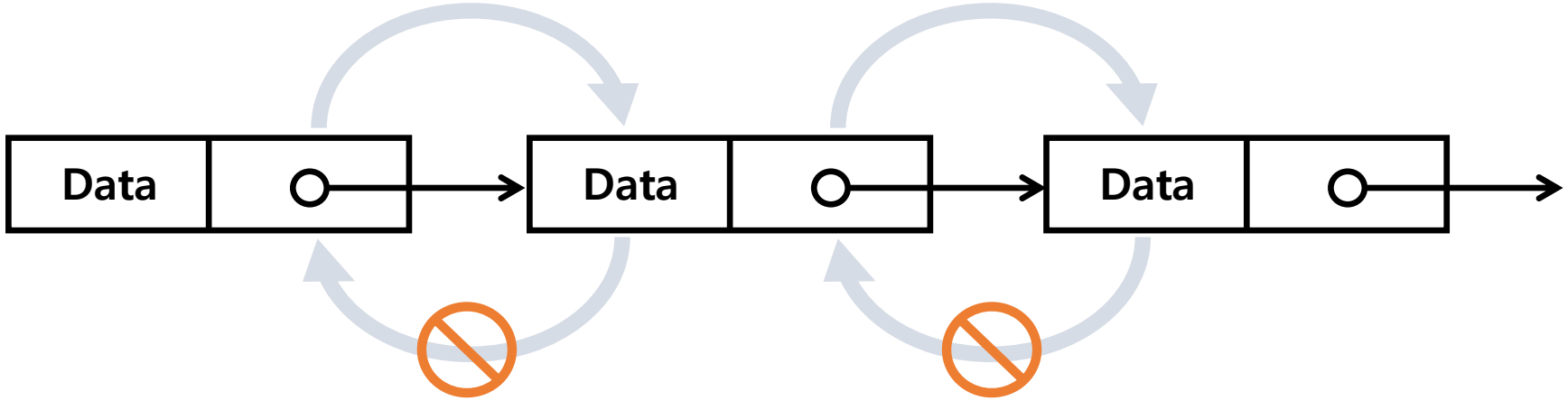


# Doubly Linked-List

## 단순연결리스트의 단점

단순연결 리스트는 Next Link만을 가지기 때문에,

특정 위치의 자료를 찾기 위해서는 항상 Root부터 전체를 탐색 해야 한다



# Doubly Linked-List

## 이중연결리스트의 개념

이중연결 리스트는 단순연결리스트의 단점을 극복하고자,

다음과 같이 Prev와 Next Link를 갖는 Node로 구성한다

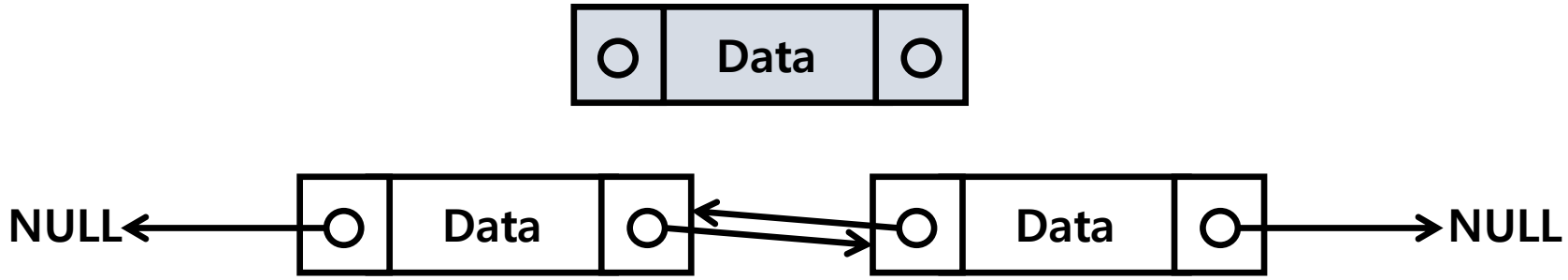

위의 Node를 포인터로 상호 연결하여 이중연결리스트를 구성한다

각 노드는 Prev, Next Link를 갖기 때문에 전방, 후방 탐색이 가능하다

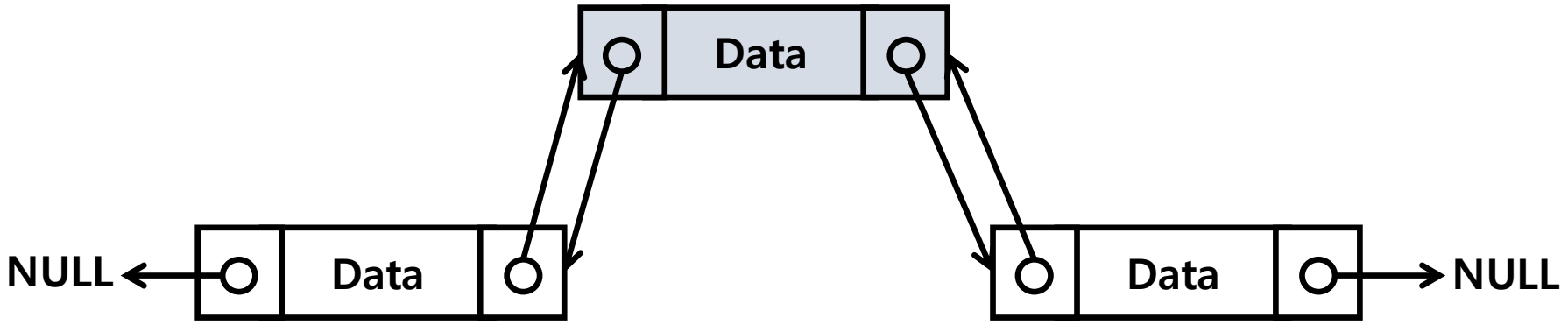
# Doubly Linked-List

## 이중연결리스트의 데이터 삽입

2번째 위치에 새로운 Node를 추가하는 방법은 다음과 같다



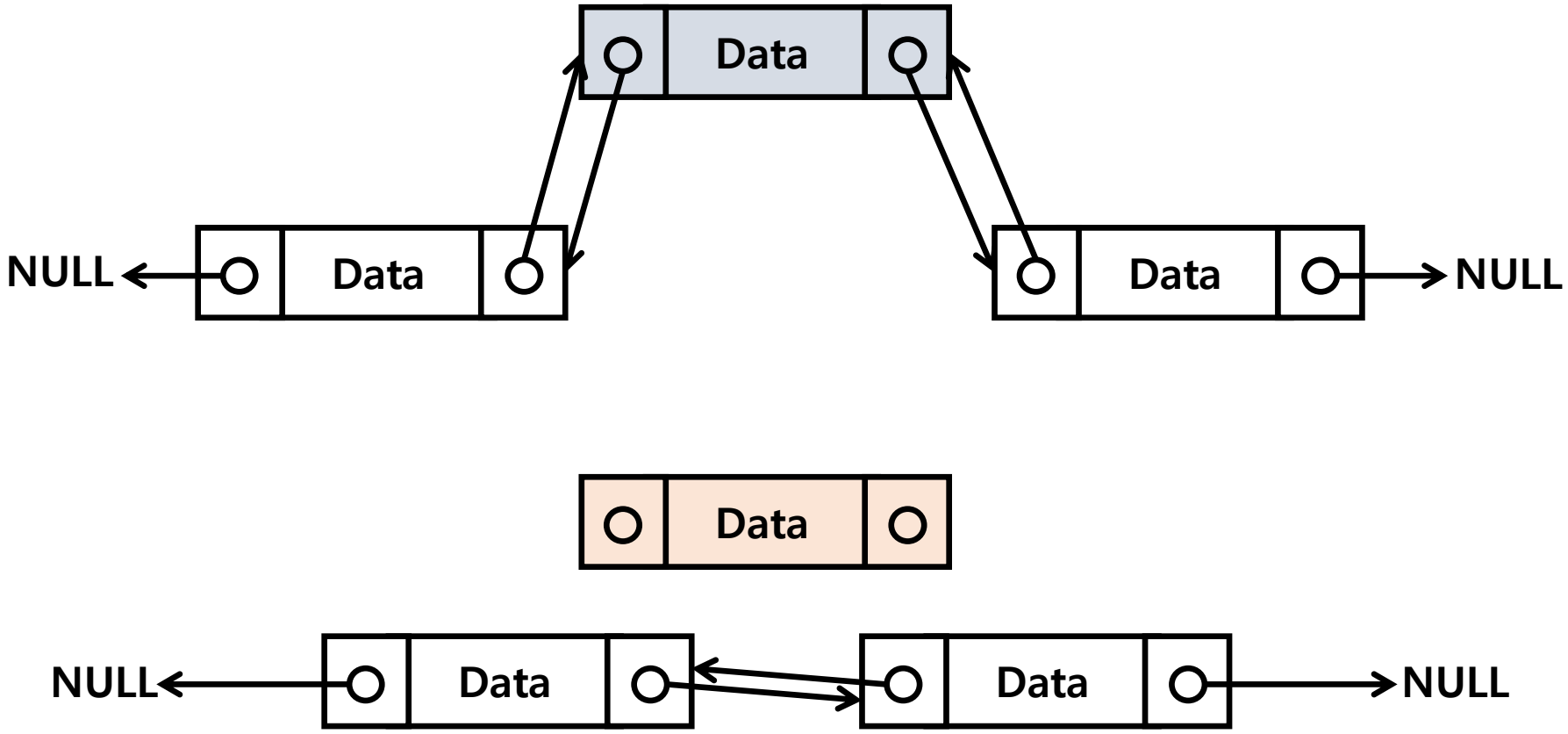

단순연결리스트와 달리 Prev, Next Link를 모두 변경하는 것이 중요하다



# Doubly Linked-List

## 이중연결리스트의 데이터 삭제

2번째 위치의 Node를 삭제하는 방법은 다음과 같다



# Doubly Linked-List

## 단순연결리스트와의 비교

장점 : 이전 Node 접근이 가능하므로, 자료를 처음 Node부터 찾지 않는다

단점 : Node에 포인터가 추가되어 리스트 메모리가 더 필요하며, 연산이 복잡하다

## 커서의 개념

이중연결리스트는 자료 탐색을 위해 매번 root부터 검색을 하지 않으므로,

현재 위치를 저장하고 있으면 **현 위치로부터 양방향으로 순차적 접근이 용이**하다