7장 인덱스된 순차 화일

❖ 인덱스된 순차 화일의 구조

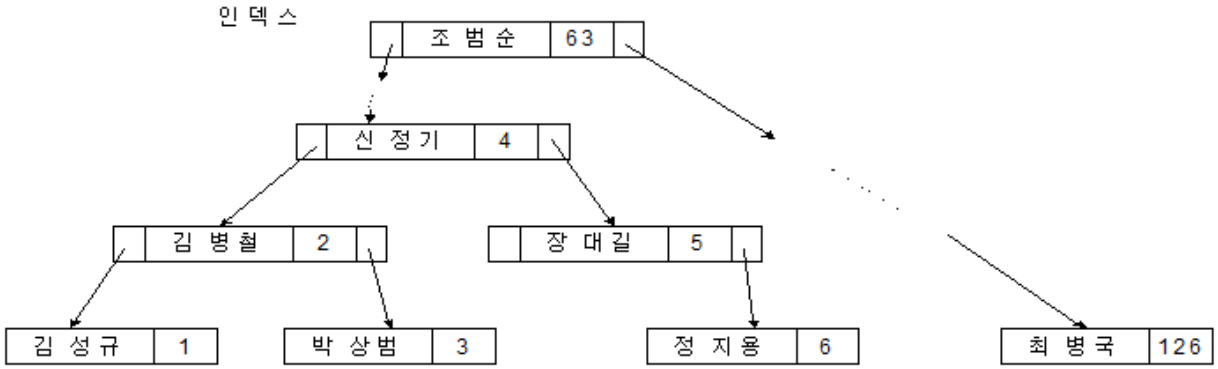
- **순차 화일 : 순차 접근 방법**
- **직접 화일 : 직접 접근 방법**

장점을 취함

- **구조**
  - 순차 데이터 화일 : 순차적으로 정렬
  - 인덱스 : 화일에 대한 포인터

▶ 인덱스된 순차 화일의 예

◆ 순차화일에 이원 탐색 트리의 인덱스를 추가해보자



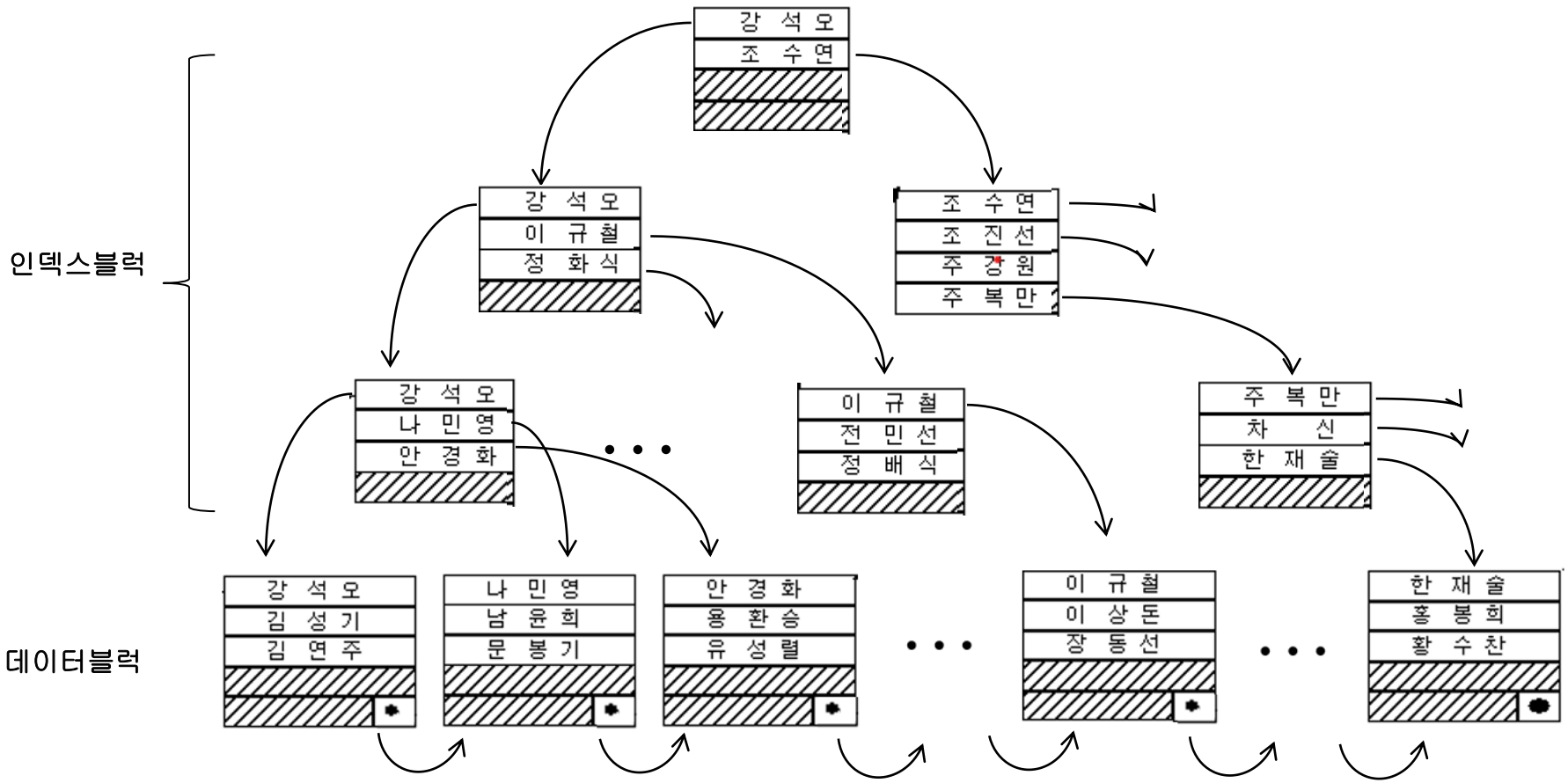
순 차 데 이 타 화 일

1	김 성 규		
2	김 병 철		
3	박 상 범		
4	신 정 기		
5	장 대 길		
6	정 지 용		
:			
63	조 범 순		
	:		
126	최 병 국		

◆ 이러한 구조가 가능할까?

– 삽입삭제가 매우 어려움

▶ 인덱스된 순차 화일의 예


인덱스블럭
데이터블럭
강 석 오
조 수 연
강 석 오
이 규 철
정 화 식
조 수 연
조 진 선
주 경 원
주 복 만
강 석 오
나 민 영
안 경 화
이 규 철
전 민 선
정 배 식
주 복 만
차 신
한 재 술
강 석 오
김 성 기
김 연 주
나 민 영
남 윤 희
문 봉 기
안 경 화
용 환 승
유 성 렬
이 규 철
이 상 돈
장 동 선
한 재 술
홍 봉 희
황 수 찬

▶ 인덱스화일의 구현 방법

- 데이터의 삽입, 삭제 시
  - 관리 사항
    - 레코드 순서 유지 방법
    - 인덱스 갱신 방법
  - 이에 따라 다음과 같이 두가지로 구현됨
    - 정적인덱스
      - 인덱스의 구조가 기억장치의 물리적 특성에 기반
      - 데이터의 삽입/삭제에도 구조가 **불변**
    - 동적인덱스
      - 블럭에 기초한 구현 : 동적인 구현
      - 데이터를 블록에 저장하고 순차접근을 위한 블록을 체인으로 연결
      - 인덱스 구조가 데이터의 삽입/삭제에 따라 **동적으로 변화**

▶ 구현 방법

	정적 인덱스(static index)	동적 인덱스(dynamic index)
인덱스	• 내용 변경에도 구조 불변 • 하드웨어 의존적설계 • (Hardware Dependent) • 보조기억장치의 물리적 특성 • (실린더, 트랙)	• B^+ 트리 • 하드웨어 독립적 설계 • Hardware Independent

데이터	• 오버플로우
	  →오버플로우구역 에 저장
		• 블록으로 저장
		• 오버플로우
		  → 분열(split) →예비 공간
		• 언더플로우
		  → 합병(merge)

구현 사례	• IBM의 ISAM	• IBM의 VSAM

❖ 정적 인덱스 방법

- 특성
  - 정적인 인덱스 구조
    - 기억장치의 물리적 특성에 기반한 인덱싱
    - 인덱스 구조에 변화가 없음
  - 데이터 삽입 공간을 위해
    - 여유공간을 가짐
    - 오버플로우 구역
- 인덱스의 구성
  - 엔트리 : <키 값, 포인터(실린더 번호, 트랙 번호)>
  - 3단계의 인덱스로 구성됨
    - 마스터 인덱스
      - 최상위 레벨 인덱스 (주기억장치에 상주)
      - <최고키값, 포인터>
    - 실린더 인덱스
      - <최고키값, 실린더 번호>
    - 트랙 인덱스
      - 실린더에 저장된 레코드들에 대한 인덱스
      - 트랙 0에 위치
      - <최저키값, 트랙 번호>

❖ 정적 인덱스 방법

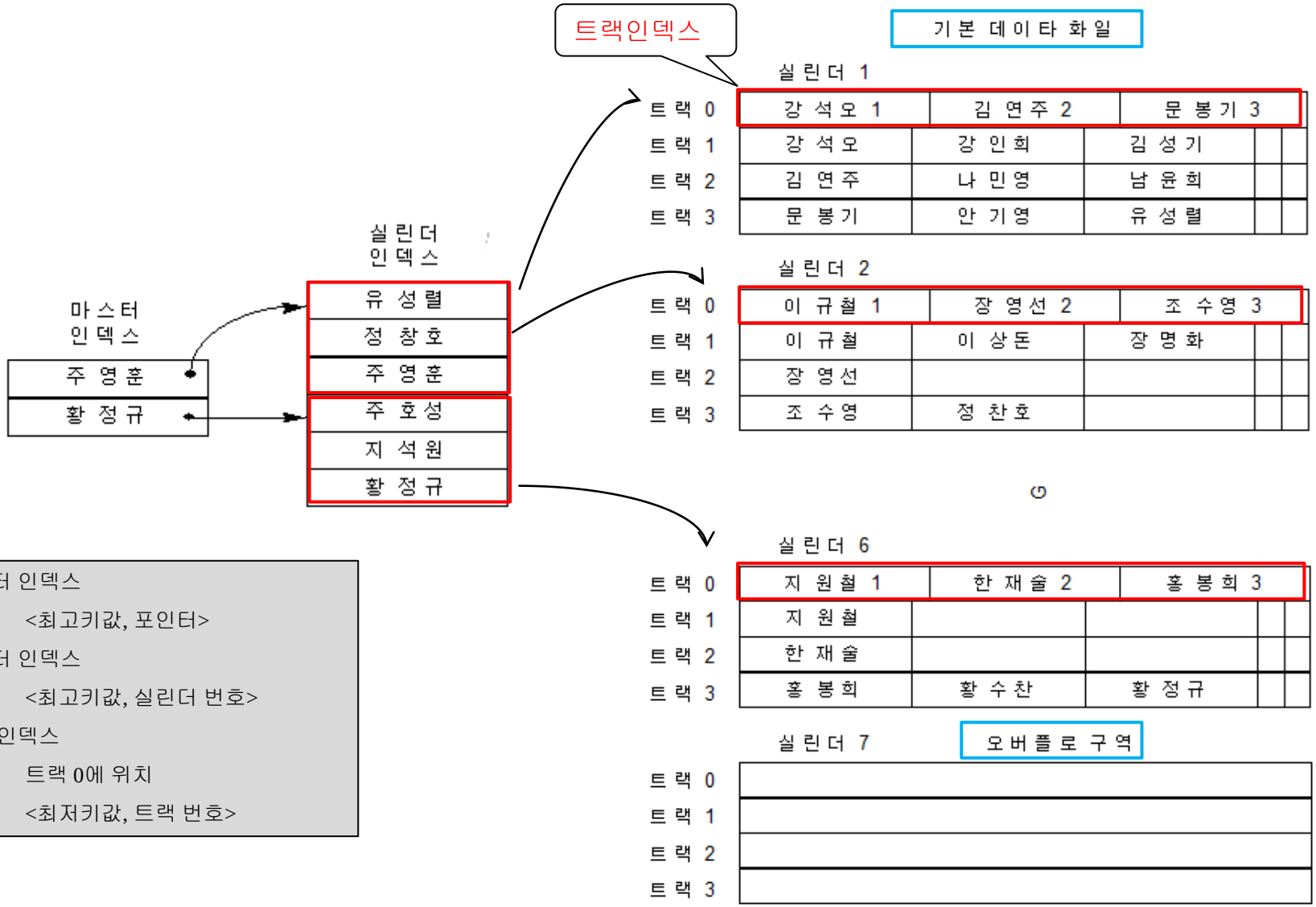
◆ 데이터 저장 구조

- 기본 구역(prime area)
  - 각 실린더
    - 트랙 0 : 트랙의 레코드에 대한 인덱스
    - 트랙 1~n : 데이터 레코드

- 오버플로우 구역
  - 분리된 화일
    - 추가적인 삽입
  - 오버플로우 포인터 : <실린더, 트랙, 레코드 번호>
  - 한 실린더 당 한 오버플로우 구역 또는 한 화일 당 한 오버플로우 구역

▶ 정적 인덱스 방법 예

◆ 정적 인덱스 방법으로 구성된 인덱스된 순차 파일의 예



마스터 인덱스

<최고키값, 포인터>

실린더 인덱스

<최고키값, 실린더 번호>

트랙 인덱스

트랙 0에 위치

<최저키값, 트랙 번호>

▶ 갱신 연산 (삽입)

★ 가정 : 각 트랙 : 5 레코드, 40 % 자유공간

1 INSERT 김수철

실린더 1

트랙 0	강 석오 1	김 연주 2	문 봉기 3		
트랙 1	강 석오	강 인희	김 성기		
트랙 2	김 연주	나 민영	남 윤희		
트랙 3	문 봉기	안 기영	유 성렬		

실린더 1

트랙 0	강 석오 1	김 연주 2	문 봉기 3		
트랙 1	강 석오	강 인희	김 성기	**김 수철**	
트랙 2	김 연주	나 민영	남 윤희		
트랙 3	문 봉기	안 기영	유 성렬		

2 INSERT 강원구

실린더 1

트랙 0	강 석오 1	김 연주 2	문 봉기 3		
트랙 1	강 석오	**강 원구**	강 인희	김 성기	김 수철
트랙 2	김 연주	나 민영	남 윤희		
트랙 3	문 봉기	안 기영	유 성렬		

3 INSERT 김시만

실린더 1

트랙 0	강 석오 1	김 시만 op	김 연주 2	문 봉기 3	
트랙 1	강 석오	강 원구	강 인희	김 성기	김 수철
트랙 2	김 연주	나 민영	남 윤희		
트랙 3	문 봉기	안 기영	유 성렬		

실린더7 오버플로우 구역

트랙 0	김 시만			
트랙 1				
트랙 2				
트랙 3				

※ 오버플로우 구역 → 기본 구역과 별개의 실린더

▶ 갱신 연산 (삽입)

4 NSERT 남창원

※ **남창원**은 **김연주**와 **문봉기** 사이 위치
(**김연주** < **남창원** < **문봉기**)

※ **트랙 2**에 입력해야함.

실린더 1

트랙 0	강 석오 1	김 시만 op	김 연주 2	문 봉기 3	
트랙 1	강 석오	강 원구	김 인희	김 성기	김 수철
트랙 2	김 연주	나 민영	남 윤희		
트랙 3	문 봉기	안 기영	유 성렬		

실린더7 오버플로우 구역

트랙 0	김 시만			
트랙 1				
트랙 2				
트랙 3				

실린더 1

트랙 0	강 석오 1	김 시만 op	김 연주 2	문 봉기 3	
트랙 1	강 석오	강 원구	김 인희	김 성기	김 수철
트랙 2	김 연주	나 민영	남 윤희	**남 창원**	
트랙 3	문 봉기	안 기영	유 성렬		

실린더7 오버플로우 구역

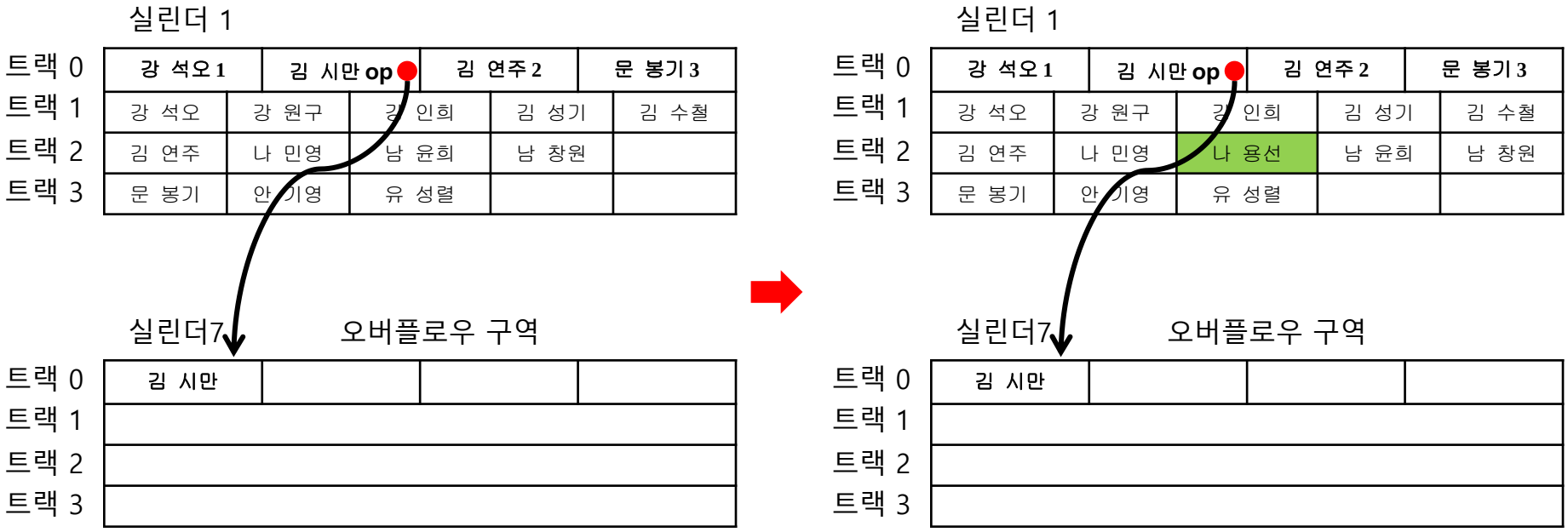
트랙 0	김 시만			
트랙 1				
트랙 2				
트랙 3				

▶ 갱신 연산 (삽입)

5 INSERT 나용선

※ **나용선**은 **김연주**와 **문봉기** 사이 위치
(**김연주** < **나용선**< **문봉기**)

※ **트랙 2**에 입력해야함.



▶ 갱신 연산 (삽입)

6 INSERT 나원규

※**나원규**는 **김연주**와 **문봉기** 사이 위치
(**김연주** < **나원규** < **문봉기**)

※ **트랙 2**에 입력해야 하나, **트랙2의 레코드는 꽉참.**

※ 오름차순을 고려했을 때 **나원규는 나용선 뒤 남윤희 앞에 위치.**

※ 나원규의 삽입으로 **나원규 이후의 레코드가 밀려** 트랙2 마지막 레코드 **남창원**은 **오버플로우 발생**

실린더 1

트랙 0	강 석오 1	김 시만 op	김 연주 2	문 봉기 3	
트랙 1	강 석오	강 원구	강 인희	김 성기	김 수철
트랙 2	김 연주	나 민영	나 용선	남 윤희	남 창원
트랙 3	문 봉기	안 기영	유 성렬		

실린더7 오버플로우 구역

트랙 0	김 시만			
트랙 1				
트랙 2				
트랙 3				

실린더 1

트랙 0	강 석오1	김 시만 op1	김 연주2	남 창원op2	문 봉기3
트랙 1	강 석오	강 원구	강 인희	김 성기	김 수철
트랙 2	김 연주	나 민영	나 용선	나 원규	남 윤희
트랙 3	문 봉기	안 기영	유 성렬		

실린더7 오버플로우 구역

트랙 0	김 시만	남 창원		
트랙 1				
트랙 2				
트랙 3				

▶ 갱신 연산 (삽입)

7 NSERT 김성복

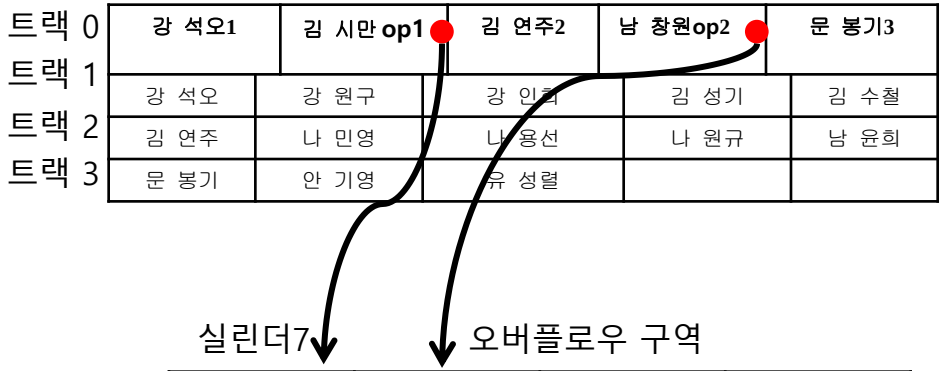
※김성복은 강석오와 김연주 사이 위치 (김연주 < 김성복 < 문봉기)

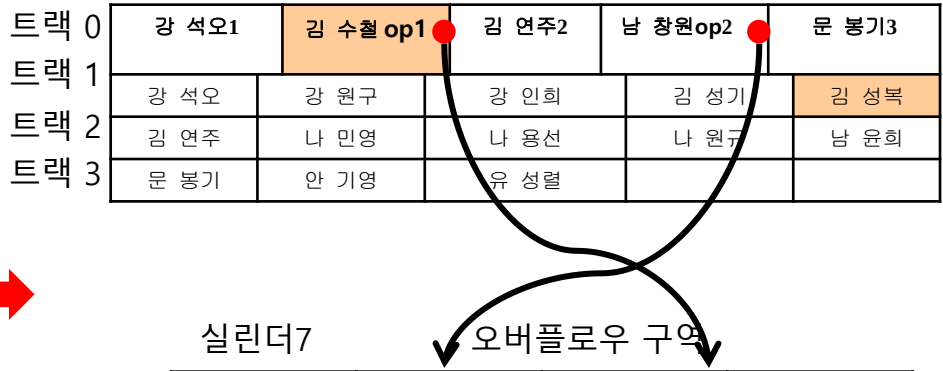
※ 트랙 1에 입력해야 하나, **트랙1의 레코드는 다참.**

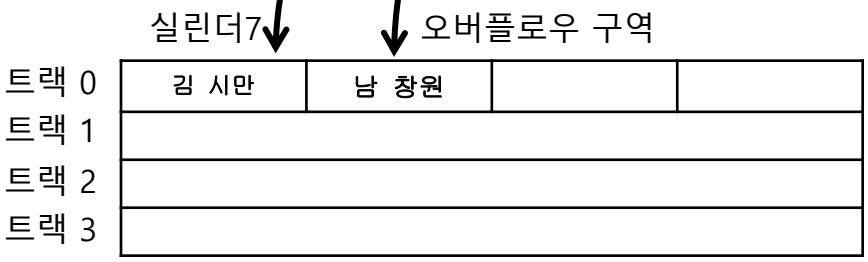
※ 오름차순을 고려했을 때 **김성복은 김성기 뒤 김수철 앞에 위치.**

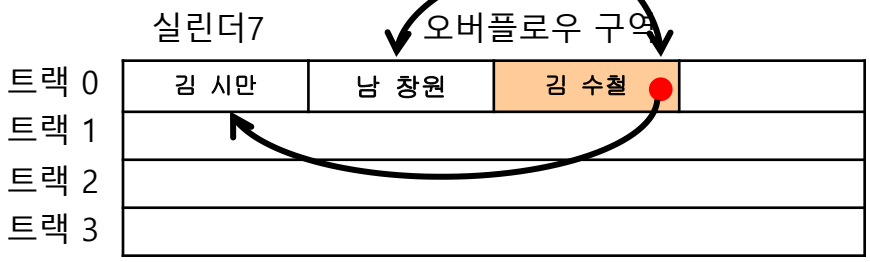
※ **김성복**의 삽입으로 **김성복 이후의 레코드가 밀려** 트랙1 마지막 레코드 **김수철**은 **오버플로우 발생**

※ **김시만**은 오름차순 순서상 **김수철 뒤에 위치**하므로, **op1의 대표 값은 김수철**이 되고, **김시만은 김수철과 오버플로우 체인으로 연결**









※ 오버플로우 구역의 레코드 : 체인으로 연결

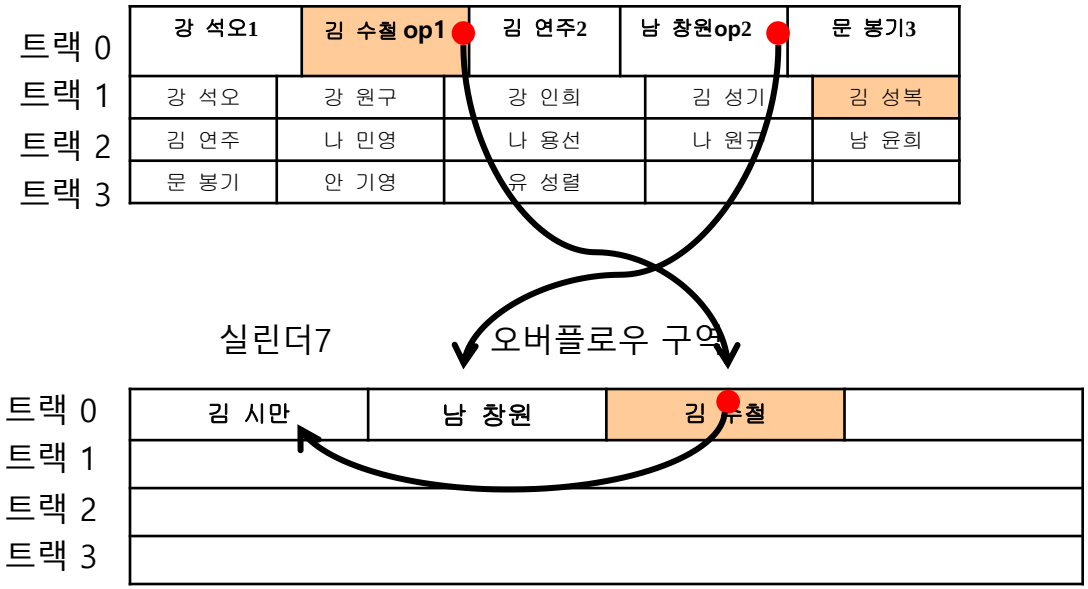
<레코드, 해당 트랙의 다음 오버플로우 레코드에 대한 포인터>

▶ 순차탐색(연습)

순차 검색

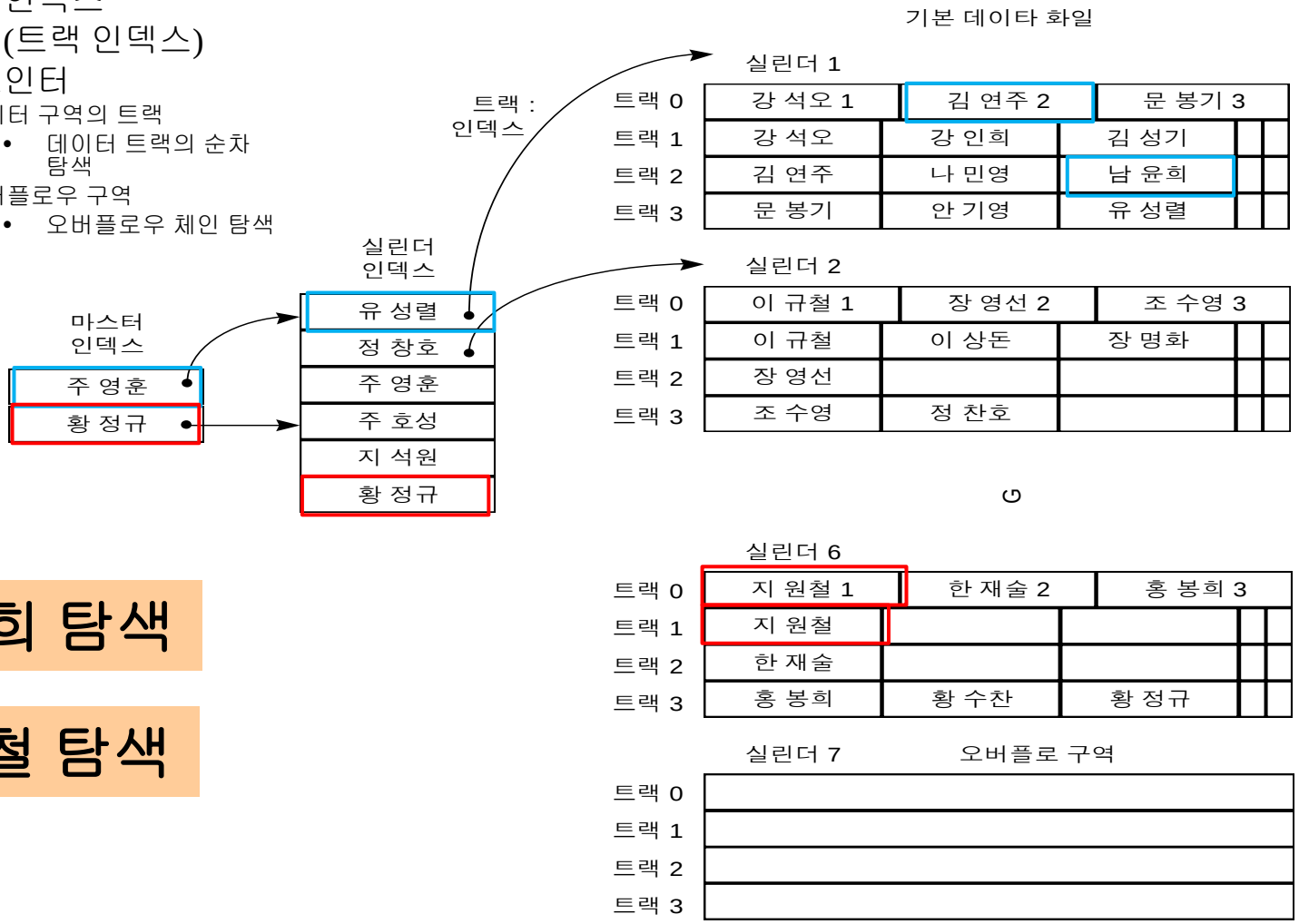
기본 데이터 트랙 검색

오버플로우 체인 접근



검색

- 직접 검색
  1. 마스터 인덱스
  2. 실린더 인덱스
  3. 실린더 (트랙 인덱스)
  4. 트랙 포인터
     - 데이터 구역의 트랙
       - 데이터 트랙의 순차 탐색
     - 오버플로우 구역
       - 오버플로우 체인 탐색



남윤희 탐색

지원철 탐색

▶ 검색 성능

- **기본 구역의 레코드 접근 : 4회**
  - 인덱스 화일 : 2
  - 트랙 인덱스 : 1
  - 데이터 트랙 : 1
- **오버플로우 구역 : 3 + k**
  - 인덱스 화일 : 2
  - 트랙 인덱스 : 1
  - 체인의 k번째 : 1
- **재구성**
  - 삽입 빈번 → 긴 체인(성능 저하)
  - 화일 관리자의 주기적 재구성
    - 순차적 완독
    - 재기록

▶ 삭제

- **동적 삭제**
  - 물리적 제거
    - ① 오버플로우 구역
      - 체인 조정
      - 만약체인의 첫 레코드인 경우 트랙 인덱스 수정
    - ② 기본 구역
      - 삭제레코드보다 큰 키 값의 레코드는 한자리 씩 이동
      - 오버플로우 레코드가 있는 경우
        - 체인의 첫 레코드가 기본구역 이동
        - ① 의 작업
- **삭제 표시**
  - 주기적인 쓰레기 수집(garbage collection)

❖ ISAM 화일

- **IBM의 ISAM(Indexed Sequential Access Method)**
- **특정 하드웨어의 특성에 맞도록 설계**
- **장점**
  - 접근 시간 단축
  - 기억 공간의 효율성
- **단점**
  - 기억장치의 유형 변경 또는 화일의 복사시 문제

❖ 동적 인덱스 방법

- **블럭에 기초한 구현 : 동적인 구현**

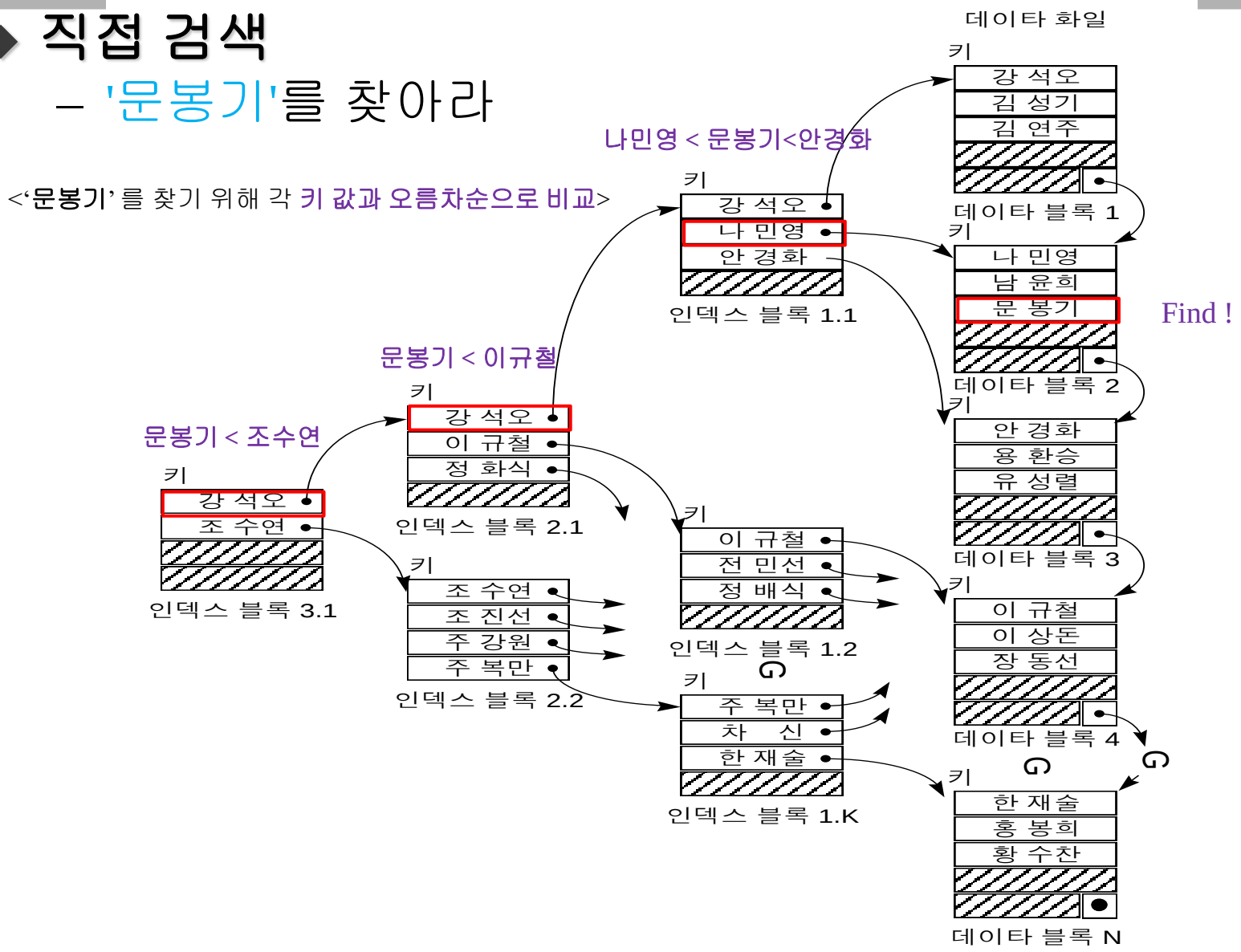
- **인덱스 화일**
  - 인덱스 블럭의 트리구조
    - 다중 레벨 인덱싱 (인덱스의 인덱스 화일)
    - 최고 레벨 인덱스(마스터 인덱스)는 주기억장치에 적합
  - 인덱스 엔트리 = <키 애트리뷰트 값, 포인터(데이터 블럭 또는 인덱스 블록)>

- **데이터 화일**
  - 순차적인 구조 - 데이터 블럭들
  - 블럭들 사이에 자유공간이 분포
    - 나중의 삽입을 위해
  - 데이터 블럭들은 논리적인 순서로 연결
    - 데이터 블럭 체인

▶ 동적 인덱스 방법의 인덱스된 순차 화일

◆ 직접 검색

– '문봉기'를 찾아라



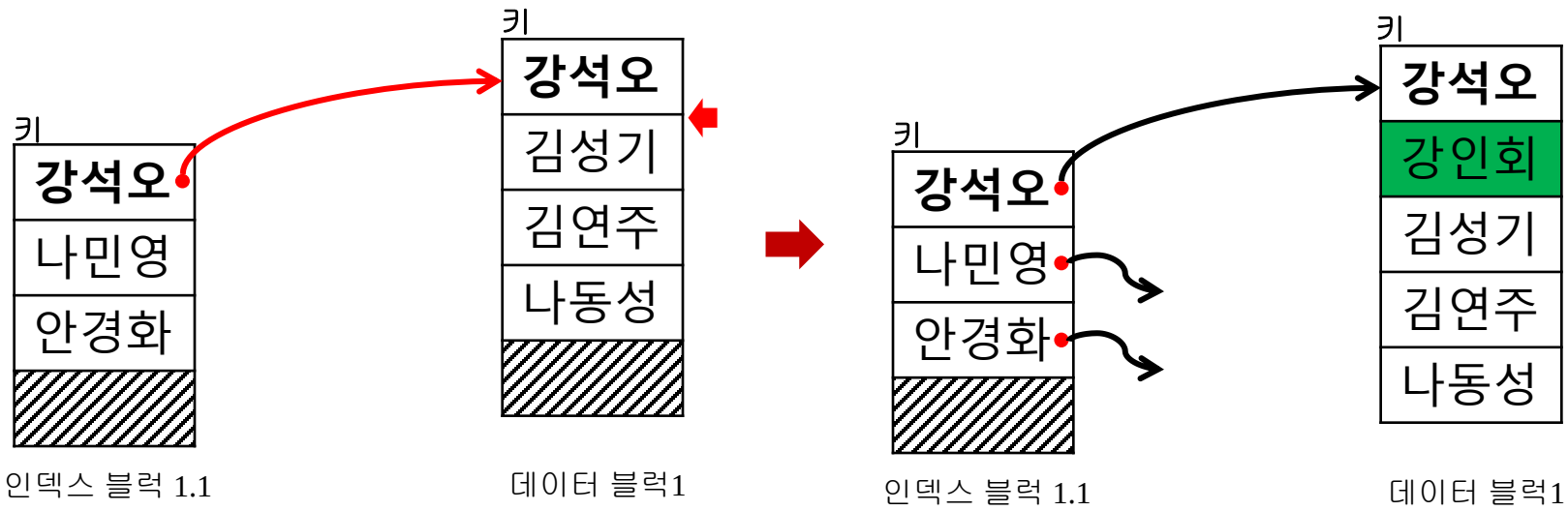
▶ 삽입 예

★ 가정 : 데이터 블럭 : 5레코드
인덱스 블럭 : 4개의 <키값,포인터> 쌍
5차 B+트리

2 INSERT 강인회

※ 강인회는 강석오와 나민영 사이 위치
(강석오 < 강인회 < 나민영)

※ 인덱스 블럭에서 키값 강석오가 가리키는 데이터 블럭에 오름차순을 고려해 강석오보다는 뒤에, 김성기보다는 앞에 삽입.



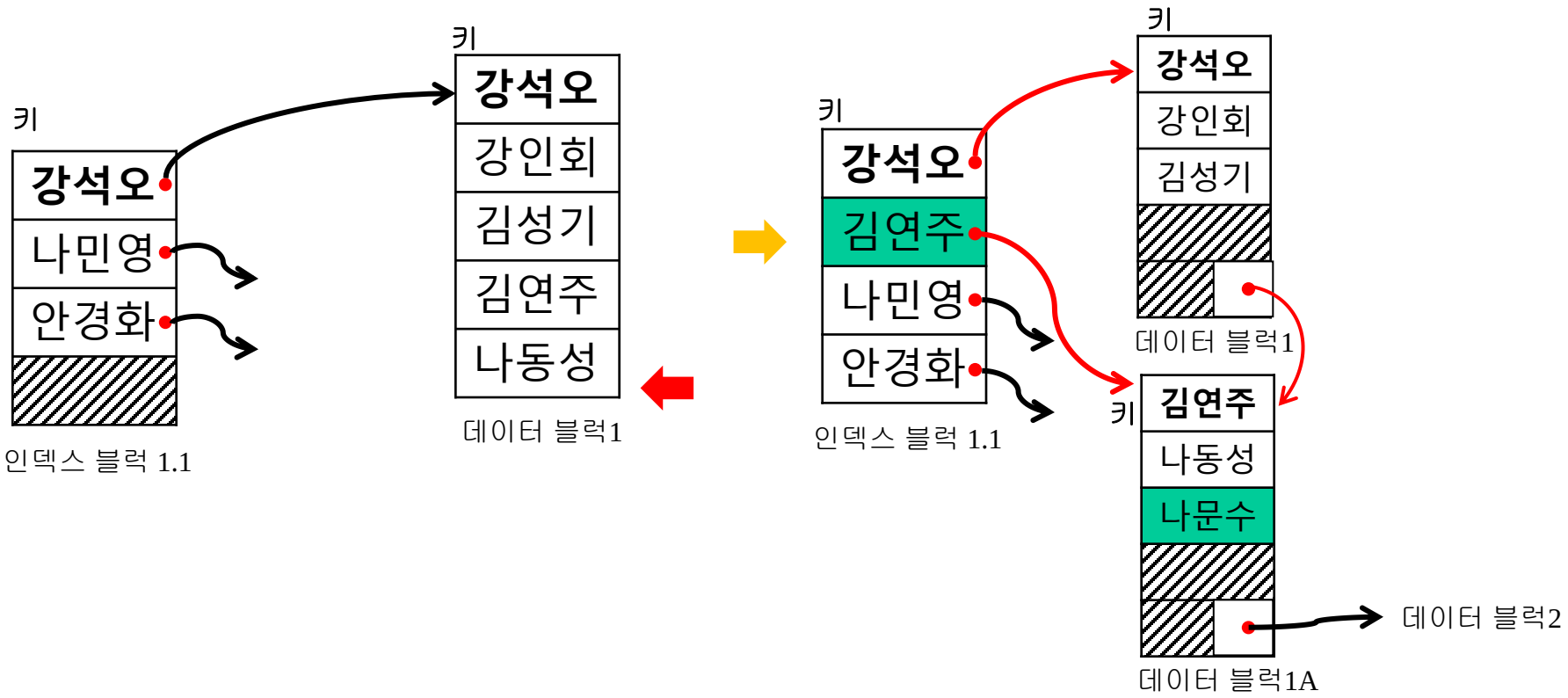
▶ 삽입 예

3 INSERT 나문수

※ **나문수**는 **강석오**와 **나민영** 사이 위치(**강석오** < **나문수** < **나민영**)

※ 인덱스 블럭에서 강석오의 데이터 블럭의 크기가 가득 차있으므로,
나문수를 입력 받기 위해 데이터 블럭의 동적인 분할 발생

※ 인덱스 블럭의 키값을 분열된 데이터 블럭의 키값(김연주)으로 추가한다.

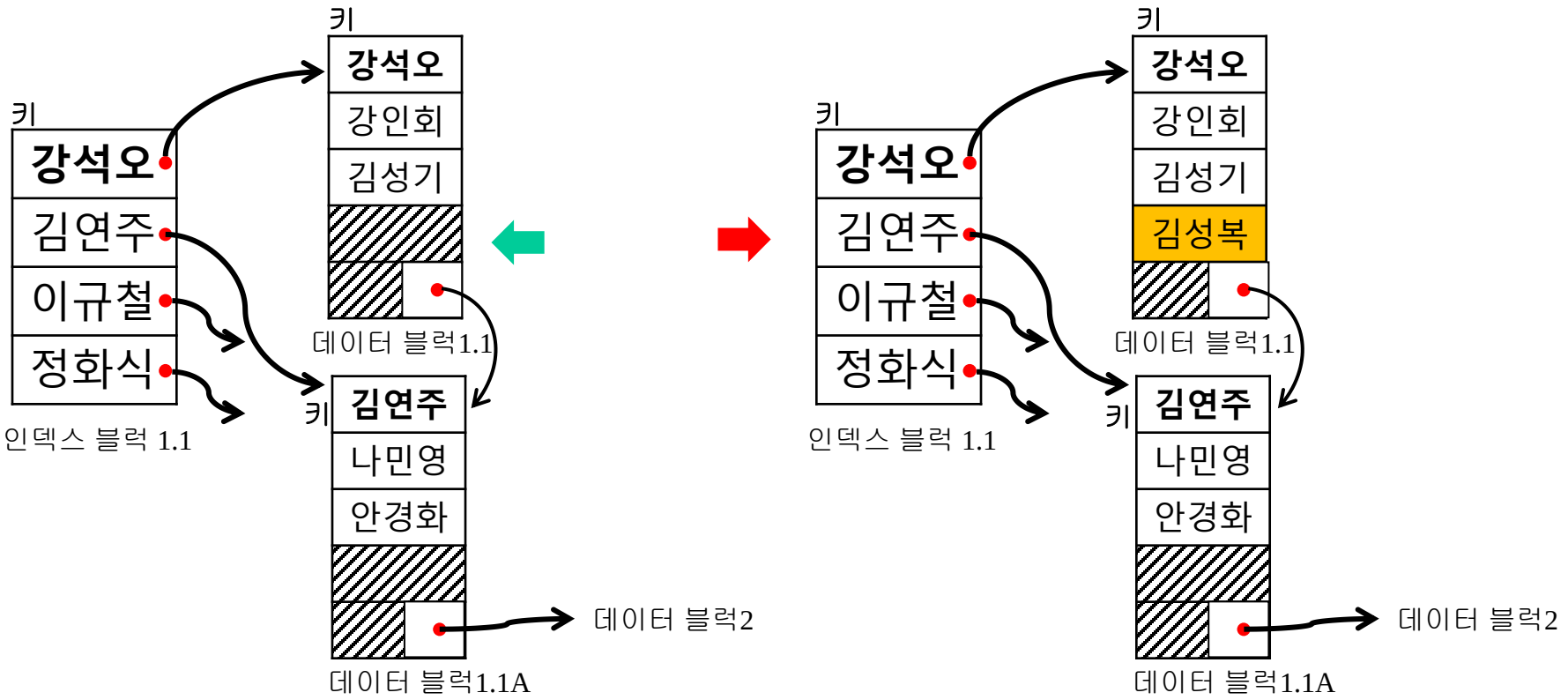


▶ 삽입 예

4 NSERT 김성복

※ **김성복**은 **감성기**와 **김연주** 사이 위치
(**김성기** < **김성복** < **김연주**)

※ 인덱스 블럭에서 강석오의 데이터 블럭에 오름차순 고려 삽입(김성기 다음 위치)



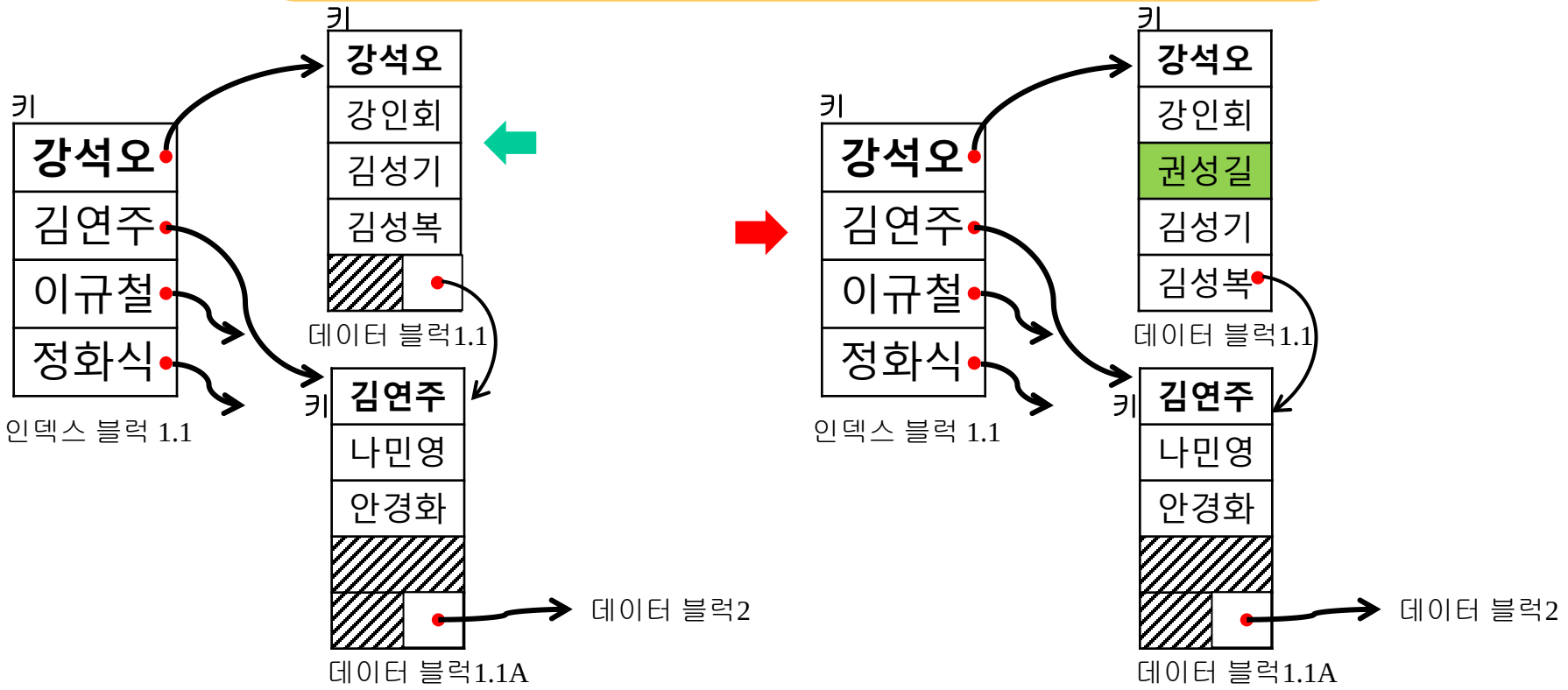
▶ 삽입 예

5 INSERT 권성길

※ **권성길**은 **강석오**와 **김연주** 사이 위치
(**강석오** < **권성길** < **김연주**)

※ 인덱스 블럭에서 강석오의 데이터 블럭에 오름차순 고려 삽입(강인회 다음 위치)

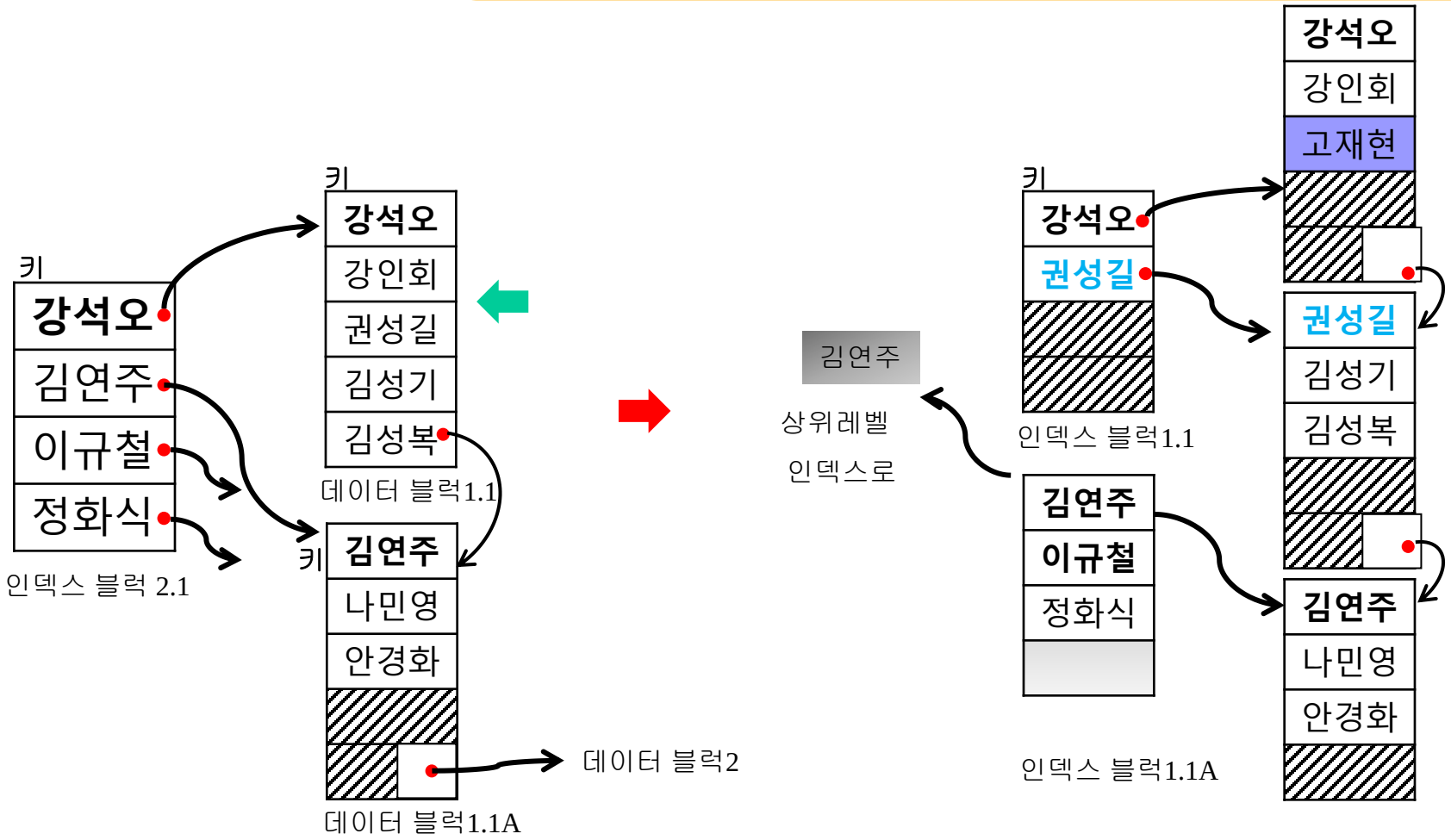
※ 권성길의 위치가 강인회 다음에 오므로, 권성길 이후의 레코드 1칸씩 밀림



▶ 삽입 예

6 INSERT 고재현

※ **고재현**은 **강석오**와 **김연주** 사이 위치 (**강석오** < **고재현** < **김연주**)
※ 인덱스 블럭에서 강석오의 데이터 블럭의 크기가 가득 차있으므로,
고재현을 입력 받기 위해 데이터 블럭의 동적인 분할 발생
※ 고재현은 **강인회 다음, 권성길 전에 위치**
※ 데이터 블럭 1.1을 강석오와 권성길을 기준으로 분할한 후, 인덱스 블럭으로 변경한다.

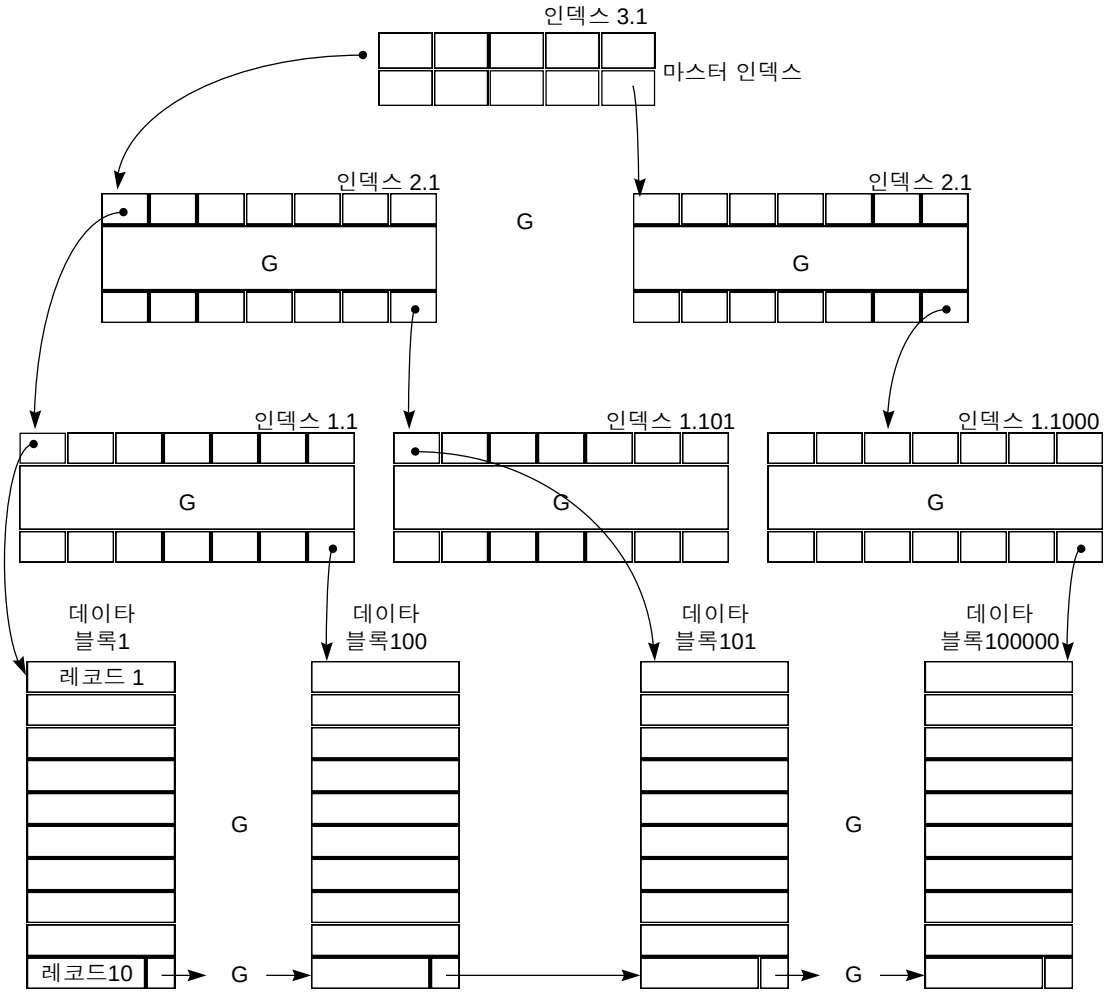


❖ VSAM 화일

- **VSAM : Virtual Storage Access Method**
- **동적 인덱스 방법**

- **VSAM 화일의 구조**
  - 제어 구간(control interval)
    - 데이터 레코드 저장
  - 제어 구역(control area)
    - 제어 구간의 모임
  - 순차셑(sequence set)
    - 제어 구역에 대한 인덱스 저장
  - 인덱스 셑(index set)
    - 순차셑의 상위 인덱스

동적 인덱스 설계 예


인덱스 3.1
마스터 인덱스
인덱스 2.1
G
인덱스 2.1
G
G
인덱스 1.1
G
인덱스 1.101
G
인덱스 1.1000
G
데이타
블록1
데이타
블록100
데이타
블록101
데이타
블록100000
레코드 1
G
G
레코드10
G
G