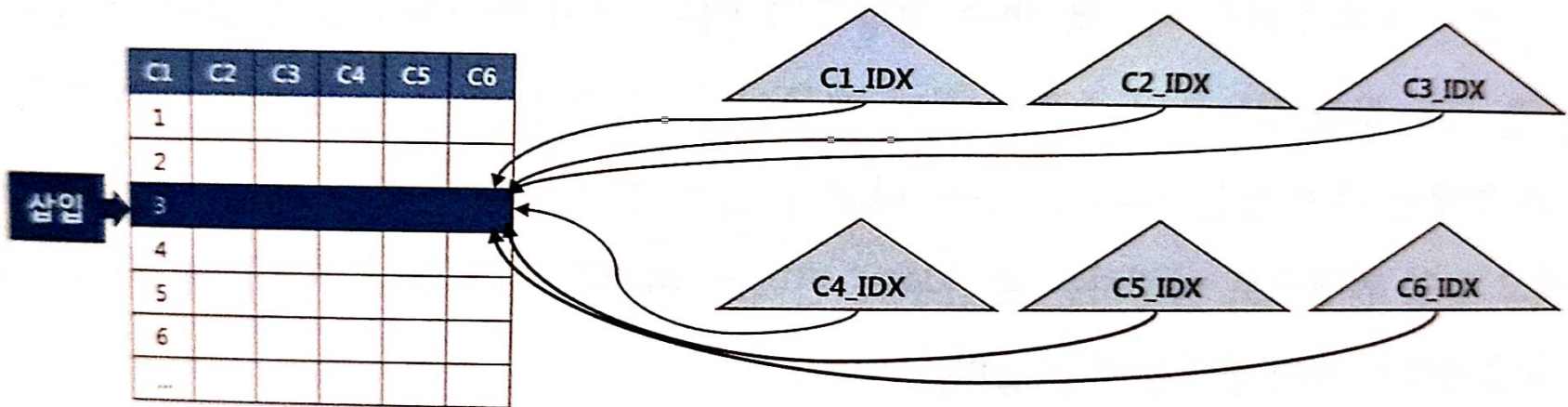


3장 인덱스 튜닝

3.4 인덱스 설계

3.4.1 인덱스 설계가 어려운 이유

- 인덱스가 많으면 아래와 같은 문제가 발생
 - DML 성능 저하(->TPS 저하)
 - 데이터베이스 사이즈 증가(->디스크 공간 낭비)
 - 데이터베이스 관리 및 운영 비용 상승.



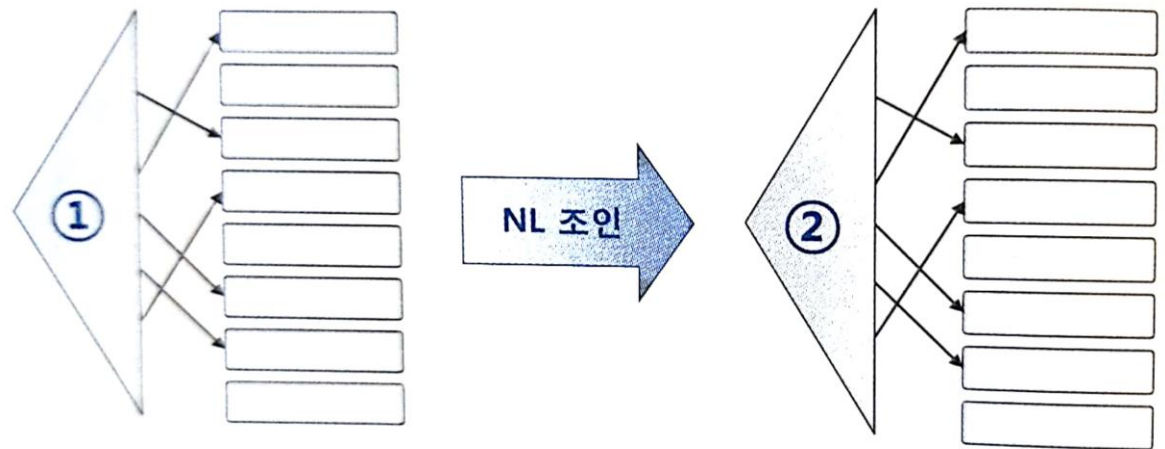
[그림 3 - 45]

3.4.2 가장 중요한 선택 기준

1. 조건절에 항상 사용하거나, 자주 사용하는 컬럼을 선정한다.
2. '=' 조건으로 자주 조회하는 컬럼을 앞쪽에 둔다.

3.4.3 스캔 효율성 이외의 판단 기준

- 수행 빈도
- 업무상 중요도
- 클러스터링 팩터
- 데이터량
- DML 부하(=기존 인덱스 개수, 초당 DML 발생량, 자주 갱신하는 컬럼 포함 여부 등)
- 저장 공간
- 인덱스 관리 비용 등



[그림 3 - 46]

A. [거래일자 + 거래구분코드]

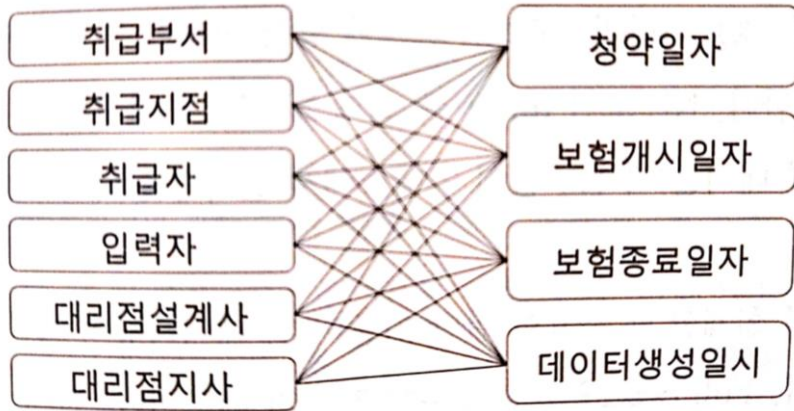
```
select /*+ leading(a) use_nl(b) */  
b.상품코드, b.상품명, a.고객번호, a.거래일자, a.거래량, a.거래금액  
from   거래 a, 상품 b  
where  a.거래구분코드 = 'AC'  
and    a.거래일자 between '20090101' and '20090131'  
and    b.상품번호 = a.상품번호  
and    b.상품분류 = '가전'
```

B. [거래일자 + 상품번호 + 거래구분코드]

```
select /*+ leading(b) use_nl(a) */  
b.상품코드, b.상품명, a.고객번호, a.거래일자, a.거래량, a.거래금액  
from   거래 a, 상품 b  
where  a.거래구분코드 = 'AC'  
and    a.거래일자 between '20090101' and '20090131'  
and    b.상품번호 = a.상품번호  
and    b.상품분류 = '가전'
```

← 비효율 발생

3.4.4 공식을 초월한 전략적 설계



[그림 3 - 47]

X01 : 청약일자 + 취급부서 + 취급지점 + 취급자 + 입력자 + 대리점설계사 + 대리점지사

X02 : 보험개시일자 + 취급부서 + 취급지점 + 취급자 + 입력자 + 대리점설계사 + 대리점지사

X03 : 보험종료일자 + 취급부서 + 취급지점 + 취급자 + 입력자 + 대리점설계사 + 대리점지사

X04 : 데이터생성일시 + 취급부서 + 취급지점 + 취급자 + 대리점설계사 + 대리점지사

- 일자 조회구간이 길지 않으면 인덱스 스캔 비효율이 성능에 미치는 영향 미미
- 인덱스 스캔 효율보다 테이블 액세스가 더 큰 부하요소

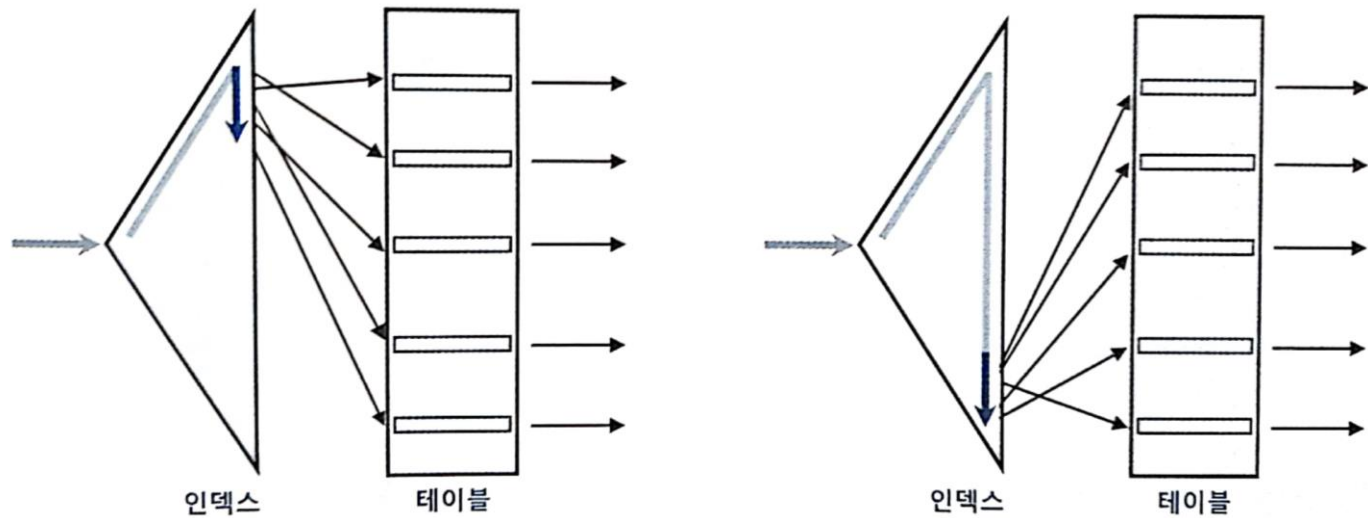
3.4.5 소트 연산을 생략하기 위한 컬럼 추가

```
select 계약ID, 청약일자, 입력자ID, 계약상태코드, 보험시작일자,보험종료일  
from 계약  
where 취급지점ID = :trt_brch_id  
and 청약일자 between :sbcg_dt1 and :sbcg_dt2  
and 입력일자 >= trunc(sysdate -3)  
and 계약상태코드 in ( :ctr_stat_cd1, :ctr_stat_cd2,:ctr_stat_cd3)  
order by 청약일자, 입력자ID
```

청약일자 + 입력자ID

청약일자 + 취급지점ID + 입력자ID

청약일자 + 입력자ID + 입력일자 + 계약상태코드



[그림 3 - 48]



1. '=' 연산자로 사용한 조건절 컬럼 선정
2. ORDER BY 절에 기술한 컬럼 추가
3. '=' 연산자가 아닌 조건절 컬럼은 데이터 분포를 고려해
추가 결정

IN 조건은 '=' 이 아니다

=> 계약상태 코드를 앞쪽에 두면 소트연산 생략?

```
select 고객번호, 고객명, 거주지역, 혈액형, 연령  
from 고객  
where 거주지역 = '서울'  
and 혈액형 in ( 'A' , 'O' )  
order by 연령
```

[거주지역 + 혈액형 + 연령]

```
select 고객번호, 고객명, 거주지역, 혈액형, 연령  
from 고객  
where 거주지역 = '서울'  
and 혈액형 = 'A'   
union all  
select 고객번호, 고객명, 거주지역, 혈액형, 연령  
from 고객  
where 거주지역 = '서울'  
and 혈액형 = 'O'   
order by 연령
```

거주지역	혈액형	연령
서울	A	23
서울	A	35
서울	A	48
서울	A	62
서울	O	29
서울	O	32
서울	O	45
서울	O	57

3.4.6 결합 인덱스 선택도

- 선택도: 전체 레코드 중에서 조건절에 의해 선택되는 레코드 비율
 - * 선택도 * 총 레코드 수 = 카디널리티
- 인덱스 선택도: 인덱스 컬럼을 모두 '='로 조회할 때 평균적으로 선택되는 비율

```
WHERE 성별 = :GENDER  
AND 고객번호 = :CUST_NO
```

< 조건절 1 >

WHERE 고객등급 = :V1
AND 고객번호 = :V2
AND 거래일자 >= :V3

<조건절 2 >

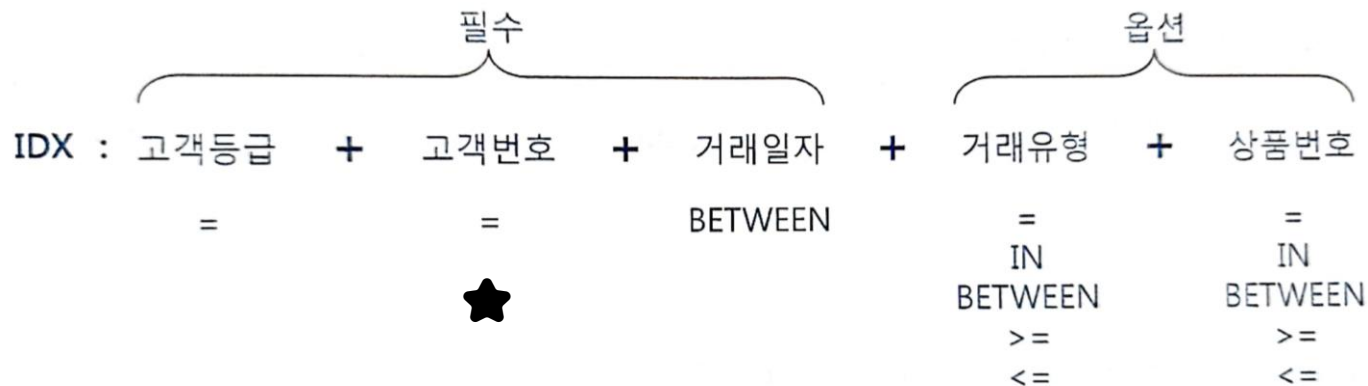
WHERE 고객등급 = :V1
AND 고객번호 = :V2
AND 거래일자 >= :V3

<조건절 3 >

WHERE 고객등급 = :V1
AND 고객번호 = :V2
AND 거래일자 >= :V3
AND 상품번호 = :V5


<조건절 4>

WHERE 고객등급 = :V1
AND 고객번호 = :V2
AND 거래일자 >= :V3
AND 거래유형 = :V4
AND 상품번호 = :V5



3.4.6 중복 인덱스 제거

완전중복

- X01: 계약ID + 청약일자
 - X02: 계약ID + 청약일자 + 보험개시일자
 - X03: 계약ID + 청약일자 + 보험개시일자 + 보험종료일자
- 

3.4.6 중복 제거 실습

컬럼 데이터분포

PK: 거래일자 + 관리지점번호 + 일련번호

N1: 계좌번호 + 거래일자

N2: 결제일자 + 관리지점번호

N3: 거래일자 + 종목코드

N4: 거래일자 + 계좌번호

컬럼명	NDV
거래일자	2,356
관리지점번호	127
일련번호	1,850
계좌번호	5,956
종목코드	1,715
결제일자	2,356

방법 1

PK: 거래일자 + 관리지점번호 + 일련번호

N1: 계좌번호 + 거래일자

N2: 결제일자 + 관리지점번호

N3: 거래일자 + 종목코드 + **계좌번호**

~~N4: 거래일자 + 계좌번호~~

PK: **관리지점번호** + 거래일자 + 일련번호

N1: 계좌번호 + 거래일자

N2: 결제일자 + 관리지점번호

N3: 거래일자 + 종목코드

~~N4: 거래일자 + 계좌번호~~

방법 2

PK: 거래일자 + 관리지점번호 + 일련번호

N1: 계좌번호 + 거래일자

N2: 결제일자 + 관리지점번호

N3: 거래일자 + 종목코드

~~N4: 거래일자 + 계좌번호~~

조건절	인덱스
관리지점번호 =, 거래일자 =	PK
관리지점번호 = , 거래일자 BETWEEN	PK
거래일자	N3
거래일자 BETWEEN	N3

3.4.6 중복 제거 실습

PK: 주소ID + 건물동번호 + 건물호번호 + 관리번호

N1: 상태구분코드 + 관리번호

N2: 관리번호

N3: 주소ID + 관리번호

컬럼 데이터분포

컬럼명	NDV
주소ID	736,000
건물동번호	175
건물호번호	3,052
관리번호	250,782
상태구분코드	3

PK: 주소ID + 건물동번호 + 건물호번호 + 관리번호

N1: 관리번호 + 상태구분코드

~~N2: 관리번호~~

N3: 주소ID + 관리번호

3.4.8 인덱스 설계도 작성

Range, Hash, List 등

인덱스 설계도

시스템명

작성자

홍길동

작성일

2004.04.01

테이블명

매물

설명

Owner

중간수

373,679

주파티션

주파티션 키

서브파티션

서브파티션 키

1

변경 전 인덱스 구성

2

변경 후 인덱스 구성

비고

NO	인덱스명		
P	PK	물건코드	물건코드
1	IDX01	지역대분류, 지역중분류	
2	IDX02	물건코드, 물건종류	물건종류, 지역대분류, 지역중분류, 읍면동
3	IDX03	입력일, 아파트시세코드, 평형, 평형타입	물건종류, 도시, 구시군, 읍면동
4	IDX04	입력일, 추천매물, 아파트시세코드, 평형, 읍면동	물건종류, 중개업소코드
5	IDX05	입력일, 지역대분류, 지역중분류, 마을명	물건종류, 평형
6	IDX06	입력일, 지역대분류, 지역중분류, 읍면동	
7	IDX07	입력일, 지역대분류, 지역중분류, CO14	
8	IDX08	입력일, 지역대분류, 평형	
9	IDX09	입력일, 전철호선, 역명, 전철도보시간	
10	IDX10	입력일, 중개업소코드, 물건코드	
11	IDX11	중개업소코드, 입력일, 인터넷매물	
12	IDX12	물건종류, 수정일	
13	IDX13	입력일, 매물구분, 추천매물, 지역대분류, 지역중분류	
14	IDX14	물건종류, 중개업소코드, 입력일, 인터넷매물	

number of distinct values

3

엑세스 경로

데이터 값 분포

기타

컬럼명	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	NDV	Avg	Max	Min	기타
물건코드	=	=																												
물건종류			=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=		11	36,865	70,446	1,394	
중개업소코드			=	=																										
아파트시세코드																										1,820		2,752	1	NULL : 280,656
평형																										1,122		13,959	1	NULL : 66,474
평형타입																										47		11,246	1	NULL : 161,795
추천매물																														
지역대분류					=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=		16		245,490	109	
지역중분류					=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=						
도시																														
구시군																														
읍면동																														
CO1																														
CO2																														
CO9																														
CO10																														
CO12																														
CO14																														
입력일																														
사용 인덱스																														
변경전	2	2	14	14	1	1	13	14	1	1	1	14	3	F	6	6	6	6	F	F	7	F	2	F	1					
변경후	P	P	4	4	2	2	2	2	2	2	2	2	5	5	3	3	3	3	3	3	3	5	P	3	2					

PK 인덱스 사용

between

order by

full table scan

메모

바인드 변수 사용 시, 도시가 '서울' 또는 '경기'인 매물을 조회할 때는 FTS로 처리되도록 실행계획 분리해

4.1 NL 조인

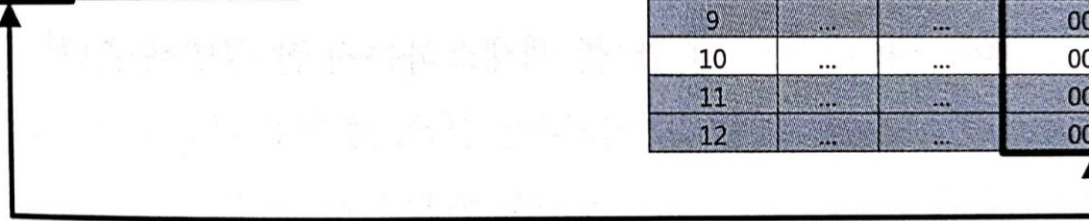
4.1.1 기본 메커니즘

사원 테이블

사원번호	사원명	입사일자
0001	홍길동	19970208
0002
0003	김철수	19960712
0004
0005	이영희	19970223
0006	손수희	19960101
0007
0008

고객 테이블

고객번호	고객명	전화번호	관리사원번호
1	0002
2	0004
3	0001
4	0003
5	0004
6	0001
7	0007
8	0006
9	0003
10	0008
11	0006
12	0005



```
select e.사원명, c.고객명, c.전화번호  
from 사원 e, 고객 c  
where e.입사일자 >= '19960101'  
and c.관리사원번호 = e.사원번호
```

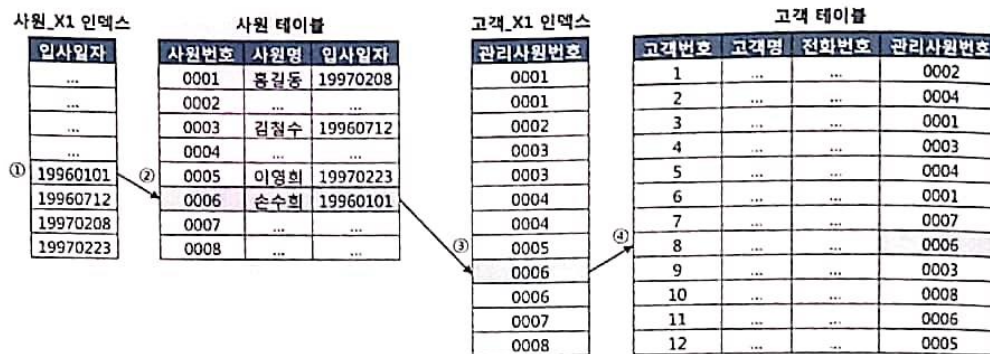
< C, JAVA >

```
for(i=0; i<100; i++){    -- outer loop
    for(j=0; j<100; j++){ -- inner loop
        // Do Anything ...
    }
}
```

< PL/SQL >

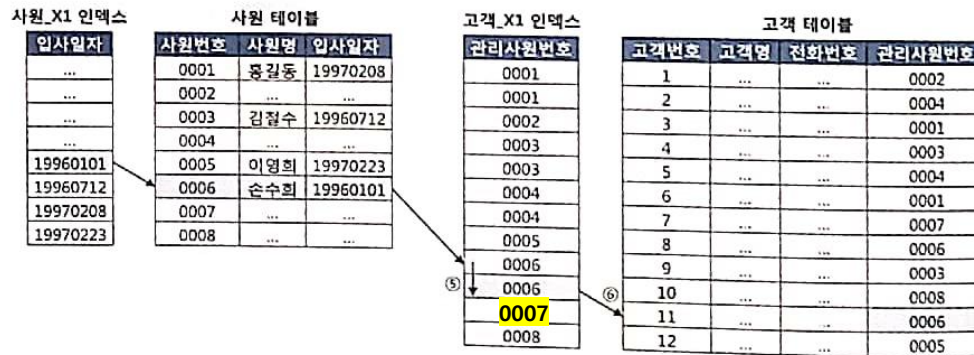
```
for outer in 1..100 loop
    for inner in 1..100 loop
        dbms_output.put_line(outer || ' : ' || inner);
    end loop;
end loop;
```

```
begin
    for outer in (select 사원번호, 사원명 from 사원 where 입사일자 >= '19960101')
    loop    -- outer 루프
        for inner in (select 고객명, 전화번호 from 고객
                        where 관리사원번호 = outer.사원번호)
        loop -- inner 루프
            dbms_output.put_line(
                outer.사원명 || ' : ' || inner.고객명 || ' : ' || inner.전화번호);
        end loop;
    end loop;
end;
```



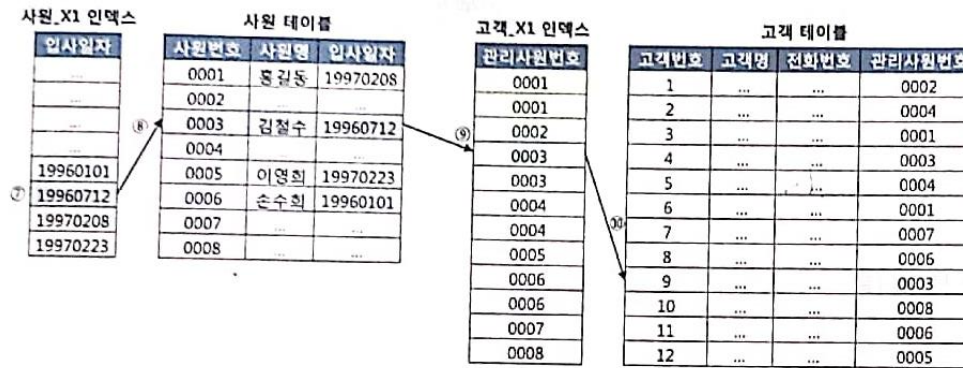
[그림 4 - 3]

- ① 사원_X1 인덱스에서 입사일자 >= '19960101' 인 첫번째 레코드를 찾는다
- ② 인덱스에서 읽은 ROWID로 사원 테이블 레코드를 찾아간다
- ③ 사원 테이블에서 읽은 사원번호 '0006'으로 고객_X1 인덱스를 탐색한다
- ④ 고객_X1 인덱스에서 읽은 ROWID로 고객 테이블 레코드를 찾아간다.



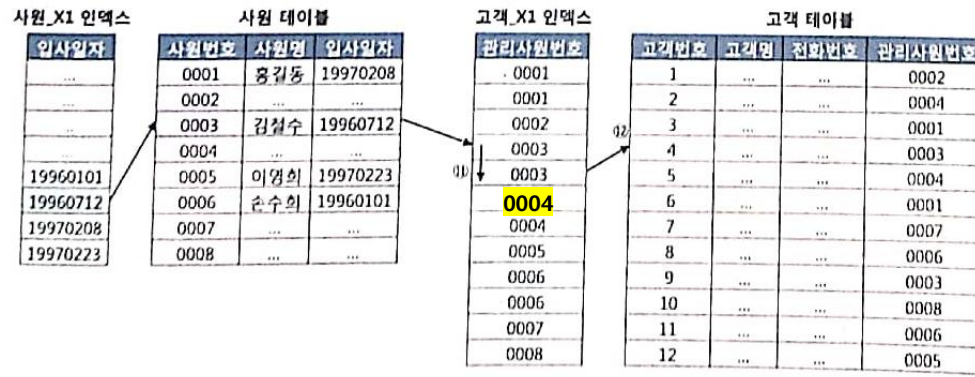
[그림 4 - 4]

- ⑤ 고객_X1 인덱스에서 한 건 더 스캔하고 관리사원번호가 '0006' 임을 확인한다.
- ⑥ 고객_X1 인덱스에서 읽은 ROWID로 고객 테이블 레코드를 찾아간다. (고객_X1 인덱스에서 한 건 더 스캔하고는 관리사원번호가 '0006'보다 크므로 거기서 인덱스 스캔을 멈춘다.)



[그림 4 - 5]

- ⑦ 사원_X1 인덱스에서 한 건 더 스캔해서 입사일자가 '19960712'인 레코드를 읽는다.
- ⑧ 인덱스에서 읽은 ROWID로 사원 테이블 레코드를 찾아간다.
- ⑨ 사원 테이블에서 읽은 사원번호 '0003'으로 고객_X1 인덱스를 탐색한다.
- ⑩ 고객_X1 인덱스에서 읽은 ROWID로 고객 테이블 레코드를 찾아간다.



[그림 4 - 6]

- ⑪ 고객_X1 인덱스에서 한 건 더 스캔하고 관리직원번호가 '0003' 임을 확인한다.
- ⑫ 고객_X1 인덱스에서 읽은 ROWID로 고객 테이블 레코드를 찾아간다. (고객_X1 인덱스에서 한 건 더 스캔하고는 관리직원번호가 '0003'보다 크므로 인덱스 스캔을 멈춘다.)

4.1.2 NL 조인 실행계획 제어

Execution Plan

```
0      SELECT STATEMENT Optimizer=ALL_ROWS
1  0      NESTED LOOPS
2  1          TABLE ACCESS (BY INDEX ROWID) OF '사원' (TABLE)
3  2              INDEX (RANGE SCAN) OF '사원_X1' (INDEX)
4  1          TABLE ACCESS (BY INDEX ROWID) OF '고객' (TABLE)
5  4              INDEX (RANGE SCAN) OF '고객_X1' (INDEX)
```

```
select /*+ ordered use_nl(c) */
      e.사원명, c.고객명, c.전화번호
from   사원 e, 고객 c
where  e.입사일자 >= '19960101'
and    c.관리사원번호 = e.사원번호
```

```
select /*+ ordered use_nl(B) use_nl(C) use_hash(D) */ *
from   A, B, C, D
where  .....
```

```
select /*+ leading(C, A, D, B) use_nl(A) use_nl(D) use_hash(B) */ *
from   A, B, C, D
where  .....
```

```
select /*+ use_nl(A, B, C, D) */ *
from   A, B, C, D
where  .....
```


4.1.3 NL 조인 수행 과정 분석

```
select /*+ ordered use_nl(c) index(e) index(c) */
      e.사원번호, e.사원명, e.입사일자
      , c.고객번호, c.고객명, c.전화번호, c.최종주문금액
from   사원 e, 고객 c
where  c.관리사원번호 = e.사원번호 ..... ①
and    e.입사일자    >= '19960101' ..... ②
and    e.부서코드     = 'Z123' ..... ③
and    c.최종주문금액 >= 20000 ..... ④
```

- 사원_PK: 사원번호
- 사원_X1: 입사일자
- 고객_PK: 고객번호
- 고객_X1: 관리사원번호
- 고객_X2: 최종주문금액

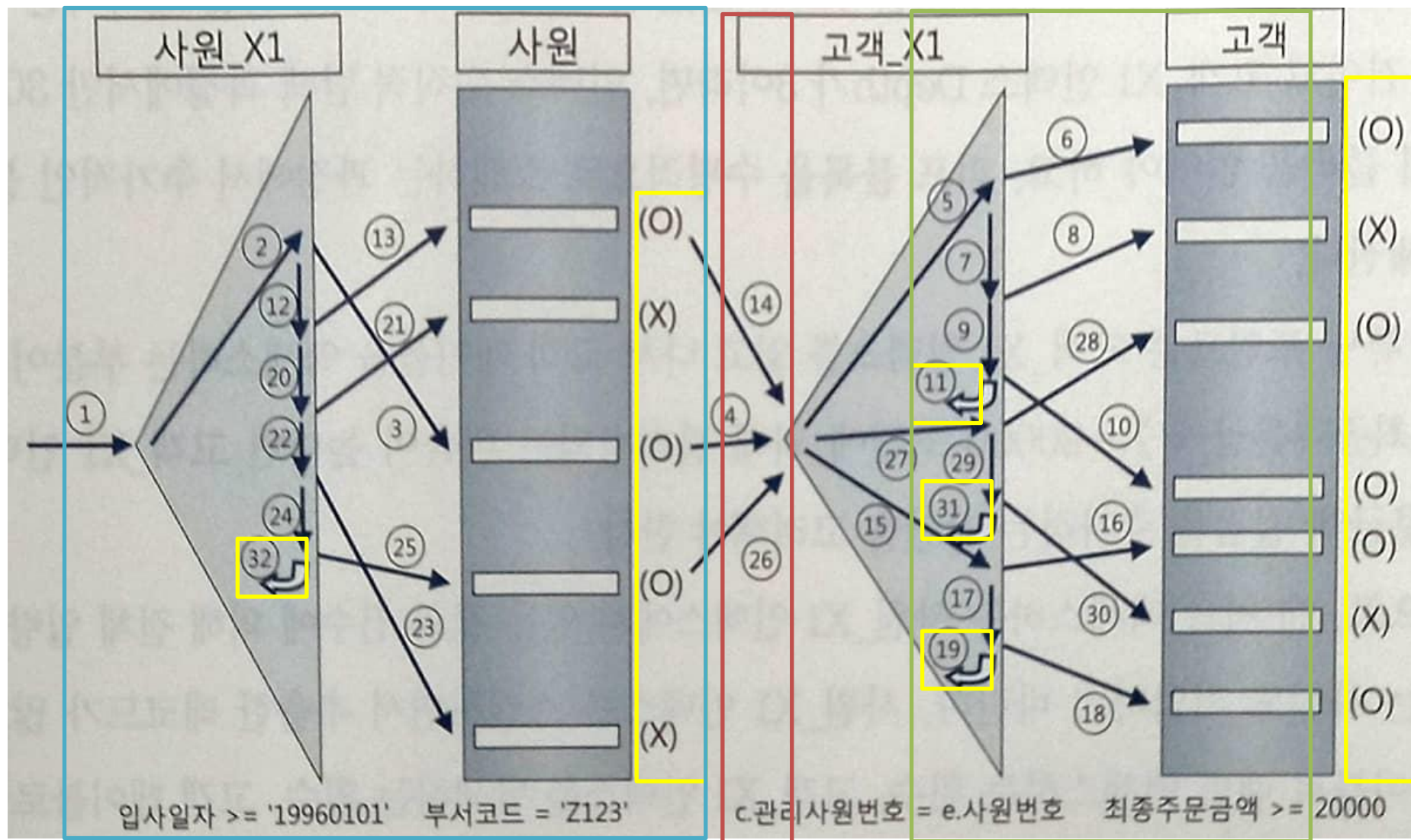
Id	Operation	Name	Rows	Bytes	Cost
0	SELECT STATEMENT		5	58	5
1	NESTED LOOPS		5	58	5
2	TABLE ACCESS BY INDEX ROWID	사원	3	20	2
3	INDEX RANGE SCAN	사원_X1	5		1
4	TABLE ACCESS BY INDEX ROWID	고객	5	76	2
5	INDEX RANGE SCAN	고객_X1	8		1

② → ③ → ① → ④

```
select /*+ ordered use_nl(c) index(e) index(c) */
      e.사원번호, e.사원명, e.입사일자
      , c.고객번호, c.고객명, c.전화번호, c.최종주문금액
from   사원 e, 고객 c
where  c.관리사원번호 = e.사원번호 ..... ①
and    e.입사일자     >= '19960101' ..... ②
and    e.부서코드     = 'Z123' ..... ③
and    c.최종주문금액 >= 20000 ..... ④
```

- 조건절 ② : 입사일자 >= '19960101' 조건을 만족하는 레코드를 찾으려고 사원_X1 인덱스(입사일자) 를 range 스캔한다.
- 조건절 ③ : 사원_X1 인덱스에서 읽은 ROWID로 사원 테이블을 액세스해서 부서코드='Z123' 필터조건을 만족하는지 확인한다.
- 조건절 ① : 사원 테이블에서 읽은 사원번호 값으로 조인 조건 을 만족하는 고객 쪽 레코드를 찾으려고 고객_X1 인덱스를 range 스캔한다.
- 조건절 ④ : 고객_X1 (고객사원번호) 인덱스에서 읽은 ROWID로 고객 테이블을 액세스해서 최종주문금액 >= 20000 필터 조건을 만족하는지 확인한다.

4.1.4 NL 조인 튜닝 포인트



4.1.5 NL 조인 특징 요약

1. 랜덤 액세스 위주의 조인 방식

2. 조인을 한 레코드 씩 순차적으로 진행

```
select /*+ ordered use_nl(b) index_desc(a (게시판구분, 등록일시)) */  
      a.게시글ID, a.제목, b.작성자명, a.등록일시  
from   게시판 a, 사용자 b  
where  a.게시판구분 = 'NEWS'          -- 게시판IDX : 게시판구분 + 등록일시  
and    b.사용자ID = a.작성자ID       -- 사용자IDX : 사용자ID  
order by a.등록일시 desc
```

3. 인덱스 구성 전략이 특히 중요

4.1.6 NL 조인 튜닝 실습

```
select /*+ ordered use_nl(c) index(e) index(c) */
       e.사원번호, e.사원명, e.입사일자
       , c.고객번호, c.고객명, c.전화번호, c.최종주문금액
from   사원 e, 고객 c
where  c.관리사원번호 = e.사원번호
and    e.입사일자    >= '19960101'
and    e.부서코드    = 'Z123'
and    c.최종주문금액 >= 20000
```

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.00	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	2	0.00	0.01	1	9	0	5
total	4	0.00	0.01	1	9	0	5

입사일자 + 부서코드

Rows	Row Source Operation
5	NESTED LOOPS
3	TABLE ACCESS BY INDEX ROWID OF 사원
3	INDEX RANGE SCAN OF 사원_X1
5	TABLE ACCESS BY INDEX ROWID OF 고객
8	INDEX RANGE SCAN OF 고객_X1

Rows	Row Source Operation
5	NESTED LOOPS (cr=112 pr=34 pw=0 time=122 us)
3	TABLE ACCESS BY INDEX ROWID OF 사원 (cr=105 pr=32 pw=0 time=118 us)
3	INDEX RANGE SCAN OF 사원_X1 (cr=102 pr=31 pw=0 time=16)
5	TABLE ACCESS BY INDEX ROWID OF 고객 (cr=7 pr=2 pw=0 time=4 us)
8	INDEX RANGE SCAN OF 고객_X1 (cr=5 pr=1 pw=0 time=0 us)

Rows	Row Source Operation
5	NESTED LOOPS (cr=2732 pr=386 pw=0 time=...)
2780	TABLE ACCESS BY INDEX ROWID 사원 (cr=166 pr=2 pw=0 time=...)
2780	INDEX RANGE SCAN 사원_X1 (cr=4 pr=0 pw=0 time=...)
5	TABLE ACCESS BY INDEX ROWID 고객 (cr=2566 pr=384 pw=0 time=...)
8	INDEX RANGE SCAN 고객_X1 (cr=2558 pr=383 pw=0 time=...)

4.1.7 NL 조인 확장 매커니즘

전통적인 실행계획

Rows	Row Source Operation
5	NESTED LOOPS
3	TABLE ACCESS BY INDEX ROWID OF 사원
5	INDEX RANGE SCAN OF 사원_X1
5	TABLE ACCESS BY INDEX ROWID OF 고객
8	INDEX RANGE SCAN OF 고객_X1

테이블 prefetch 실행계획

Rows	Row Source Operation
5	TABLE ACCESS BY INDEX ROWID OF 고객
12	NESTED LOOPS
3	TABLE ACCESS BY INDEX ROWID OF 사원
3	INDEX RANGE SCAN OF 사원_X1
8	INDEX RANGE SCAN OF 고객_X1

배치 I/O 실행계획

Rows	Row Source Operation
5	NESTED LOOPS
8	NESTED LOOPS
3	TABLE ACCESS BY INDEX ROWID OF 사원
3	INDEX RANGE SCAN OF 사원_X1
8	INDEX RANGE SCAN OF 고객_X1
5	TABLE ACCESS BY INDEX ROWID OF 고객

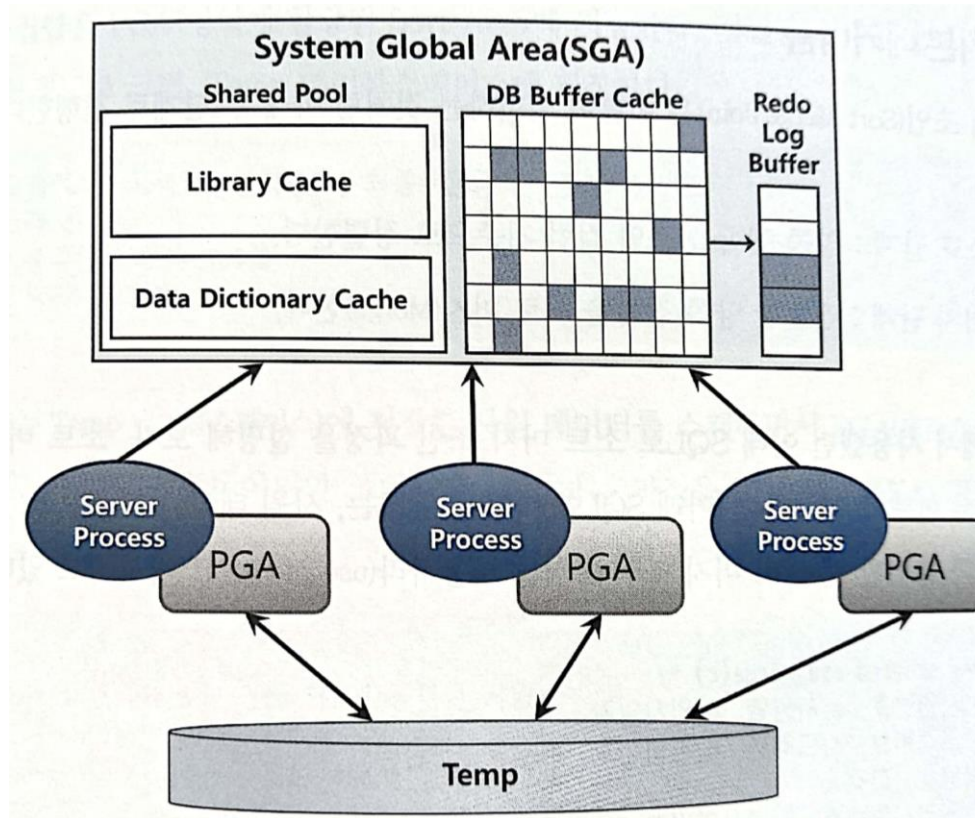
```

SELECT /** ordered use_n1(b) */
      A.등록일시, A.번호, A.제목, B.회원명, A.게시판유형, A.질문유형
FROM (
      SELECT A.*, ROWNUM NO
      FROM (
            SELECT 등록일시, 번호, 제목, 작성자번호, 게시판유형, 질문유형
            FROM   게시판
            WHERE  게시판유형 = :TYPE
            ORDER BY 등록일시 DESC  -- 인덱스 구성 : 게시판유형 + 등록일시
        ) A
      WHERE ROWNUM <= (:page * 10)
    ) A, 회원 B
WHERE A.NO >= (:page-1)*10 + 1
AND   B.회원번호 = A.작성자번호
ORDER BY A.등록일시 DESC      → 11g부터 여기에 ORDER BY를 명시해야 정렬 순서 보장

```


4.2 소트머지조인

4.2.1 SGA vs PGA



4.2.2 기본 메커니즘

1. 소트단계: 양쪽 집합을 조인 컬럼 기준으로 정렬한다.
2. 머지단계: 정렬한 양쪽 집합을 서로 Merge한다.

```

select /*+ ordered use_merge(c) */
       e.사원번호, e.사원명, e.입사일자
       , c.고객번호, c.고객명, c.전화번호, c.최종주문금액
from   사원 e, 고객 c
where  c.관리사원번호 = e.사원번호
and    e.입사일자    >= '19960101'
and    e.부서코드    = 'Z123'
and    c.최종주문금액 >= 20000

```

①

```

select 사원번호, 사원명, 입사일자
from   사원
where  입사일자 >= '19960101'
and    부서코드 = 'Z123'
order by 사원번호

```

②

```

select 고객번호, 고객명, 전화번호, 최종주문금액, 관리사원번호
from   고객 c
where  최종주문금액 >= 20000
order by 관리사원번호

```

③

```

begin
  for outer in (select * from PGA에_정렬된_사원)
  loop -- outer 루프
    for inner in (select * from PGA에_정렬된_고객
                  where 관리사원번호 = outer.사원번호)
    loop -- inner 루프
      dbms_output.put_line( ... );
    end loop;
  end loop;
end;

```

PGA에 정렬된 직원 데이터

직원번호	직원명	입사일자
0001	홍길동	19970208
0003	김철수	19960712
0005	이영희	19970223
0006	손수희	19960101

PGA에 정렬된 고객 데이터

관리직원번호	고객번호	고객명	최종주문금액
0001	3	...	25,000
0001	6	...	50,000
0002	1	...	100,000
0003	4	...	30,000
0003	9	...	20,000
0003	2	...	40,000
0004	5	...	70,000
0004	13	...	57,000
0005	12	...	60,000
0005	8	...	38,000
0006	11	...	200,000
0007	7	...	20,000
0008	10	...	45,000

①

②

③

④

4.2.2 소트머지 조인이 빠른이유

NL 조인	소트 머지
인덱스를 이용한 조인	일괄적으로 읽어 PGA 저장 후 조인
랜덤 액세스 방식	
래치 획득 및 캐시버퍼 획득 과정 거침	래치획득 과정 없음
대량 데이터 조인 불리	대량 데이터 조인 유리

4.2.2 소트머지 주용도

- 조인 조건식이 등치 조건이 아닌 대량 데이터 조인
- 조인 조건식이 아예 없는 조인

4.2.2 소트머지 조인 제어하기

Execution Plan

```
0      SELECT STATEMENT Optimizer=ALL_ROWS
1    0      MERGE JOIN
2    1        SORT (JOIN)
3    2          TABLE ACCESS (BY INDEX ROWID) OF '사원' (TABLE)
4    3            INDEX (RANGE SCAN) OF '사원_X1' (INDEX)
5    1        SORT (JOIN)
6    5          TABLE ACCESS (BY INDEX ROWID) OF '고객' (TABLE)
7    6            INDEX (RANGE SCAN) OF '고객_X1' (INDEX)
```

```
select /*+ ordered use_merge(c) */
      e.사원번호, e.사원명, e.입사일자
,      c.고객번호, c.고객명, c.전화번호, c.최종주문금액
from    사원 e, 고객 c
where   c.관리사원번호 = e.사원번호
and     e.입사일자      >= '19960101'
and     e.부서코드      = 'Z123'
and     c.최종주문금액 >= 20000
```

4.2.2 소트머지 조인 특징 요약

- 조인을 위해 실시간으로 인덱스를 생성하는 것
- NL 조인은 조인 컬럼에 대한 인덱스 유무에 크게 영향을 받지만, 소트머지 조인은 영향을 받지 않는다.
- 스캔 위주의 인덱스 방식을 사용한다