

INDEX

현업 사용 정도 : ★ ★ ★ ★ ★

1. 인덱스(INDEX)란?
2. 인덱스(INDEX) 원리 이해하기
3. NL(Nested Loop) 조인 이해하기
4. 실습

1. 인덱스(INDEX)란?

인덱스(INDEX) 란?

책에 있는 색인 혹은 목차처럼 특정 데이터를 쉽게 찾을 수 있게 **특정 기준으로 정리해놓은 객체**

A씨는 몇 명이 있죠?



A씨



B씨



C씨



D씨



E씨



A씨



F씨



H씨



A씨



I씨

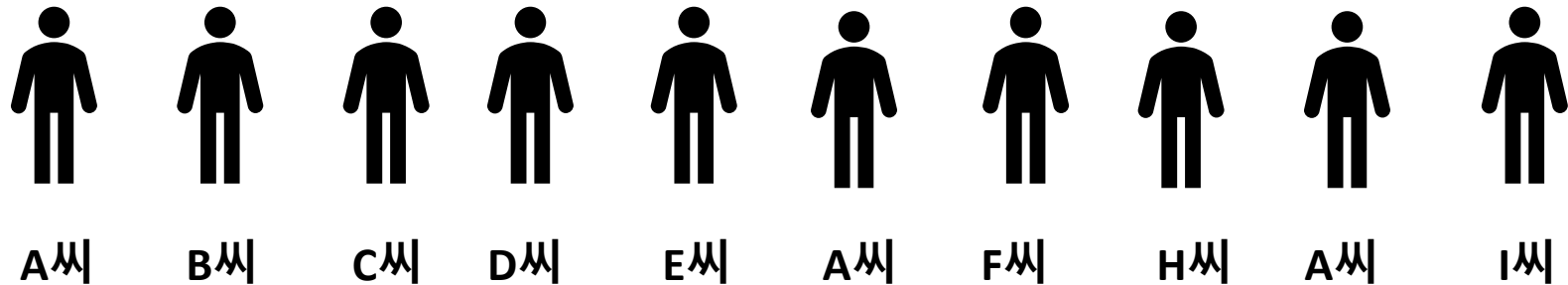
현재 무작위로 자리에 착석한 상황

1. 인덱스(INDEX)란?

인덱스(INDEX) 란?

책에 있는 색인 혹은 목차처럼 특정 데이터를 쉽게 찾을 수 있게 **특정 기준으로 정리해놓은 객체**

A씨는 몇 명이 있죠?



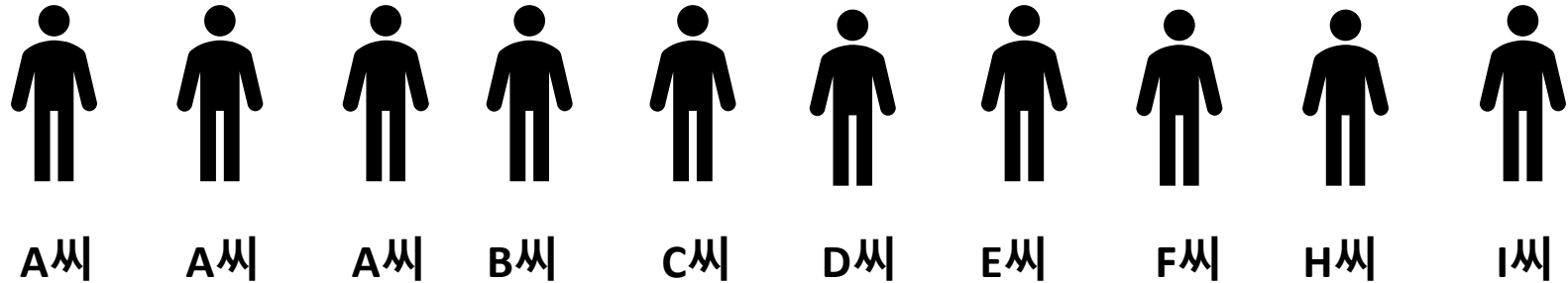
A씨가 맨 뒤에도 있을 수 있으니 끝까지 확인해봐야함

1. 인덱스(INDEX)란?

인덱스(INDEX) 란?

책에 있는 색인 혹은 목차처럼 특정 데이터를 쉽게 찾을 수 있게 **특정 기준으로 정리해놓은 객체**

A씨는 몇 명이 있죠?

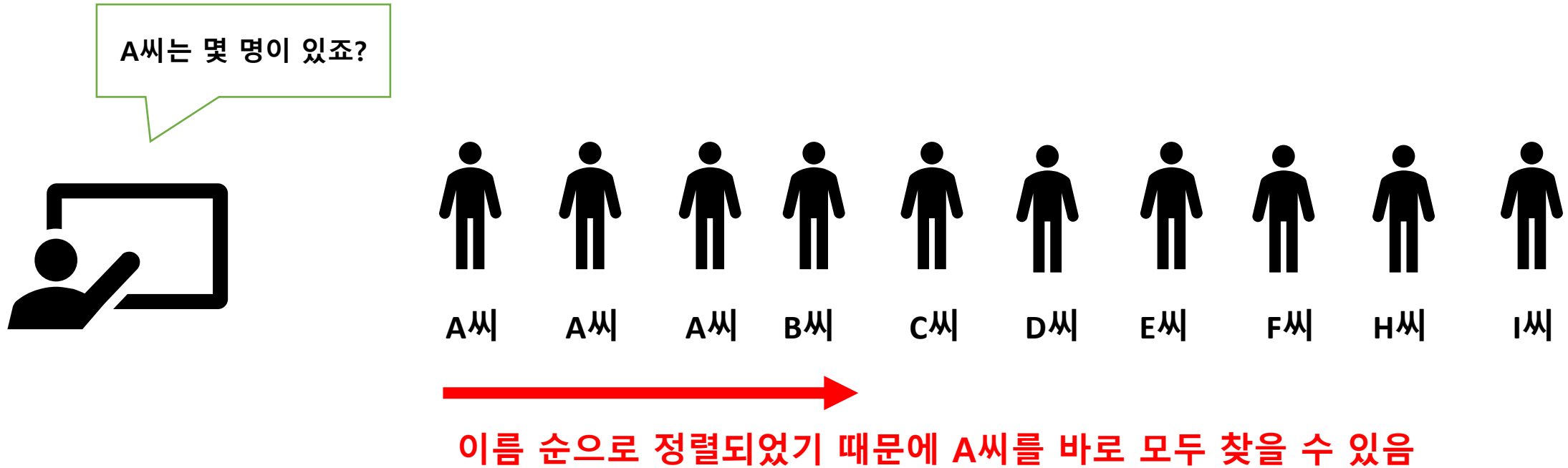


이름 순으로 오름차순하여 자리에 앉은 상황

1. 인덱스(INDEX)란?

인덱스(INDEX) 란?

책에 있는 색인 혹은 목차처럼 특정 데이터를 쉽게 찾을 수 있게 **특정 기준으로 정리해놓은 객체**



인덱스(INDEX) 가 없다면 ?

```
SELECT PRD_ID    --상품ID
      , PRD_NAME --상품명
      , PRD_TYPE --상품타입
FROM TB_PRD
WHERE PRD_TYPE = '스마트폰';
```

위 쿼리는 어떻게 실행될까요?

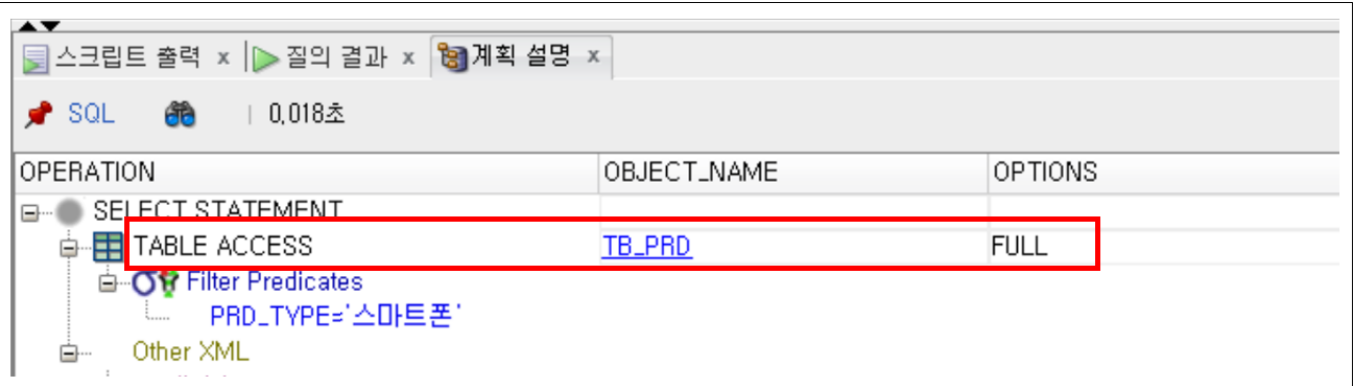


PRD_ID	PRD_NAME	PRD_TYPE
P0001	헤어드라이기	가전
P0002	에어컨	가전
P0003	세탁기	가전
P0004	건조기	가전
P0005	노트북	컴퓨터
P0006	데스크탑	컴퓨터
P0007	태블릿	컴퓨터
P0008	애플14	스마트폰
P0009	갤럭시S23	스마트폰
P0010	조아샴푸	욕실용품
P0011	주전자	주방용품
P0012	전기밥솥	주방용품
P0013	냄비	주방용품
P0014	칼	주방용품
P0015	수세미	욕실용품
P0017	곰팡이제거제	욕실용품
P0018	샤워기	욕실용품
P0019	린스	욕실용품
P0020	수건	욕실용품

인덱스(INDEX) 가 없다면?

```
SELECT PRD_ID    --상품ID
      , PRD_NAME --상품명
      , PRD_TYPE --상품타입
FROM TB_PRD
WHERE PRD_TYPE = '스마트폰';
```

계획설명 보는법 : 쿼리에 커서 대고 F10 클릭 (SQL DEVELOPER 기준)



OPERATION	OBJECT_NAME	OPTIONS
SELECT STATEMENT		
TABLE ACCESS	TB_PRD	FULL
Filter Predicates		
PRD_TYPE='스마트폰'		
Other XML		

실행 계획을 보면 **TABLE FULL SCAN** 이 떠있습니다.
즉, 위 조건에 맞는 데이터를 찾기 위해 테이블을 모두 조회했다는 의미입니다.

2. 인덱스(INDEX) 원리 이해하기

인덱스(INDEX) 를 만들어봅시다.

```
CREATE INDEX IDX_PRD ON TB_PRD ( PRD_TYPE [ ASC | DESC ] , ... ) ;
```

인덱스를 생성합니다.

생성할
인덱스명

TB_PRD 테이블의
PRD_TYPE 컬럼을 이용

오름차 혹은 내림차 순으로 생성
(기본값은 오름차순)

PRD_ID	PRD_NAME	PRD_TYPE
P0001	헤어드라이기	가전
P0002	에어컨	가전
P0003	세탁기	가전
P0004	건조기	가전
P0005	노트북	컴퓨터
P0006	데스크탑	컴퓨터
P0007	태블릿	컴퓨터
P0008	애플14	스마트폰
P0009	갤럭시s23	스마트폰
P0010	조아샴푸	욕실용품
P0011	주전자	주방용품
P0012	전기밥솥	주방용품
P0013	냄비	주방용품
P0014	칼	주방용품
P0015	수세미	욕실용품
P0017	곰팡이제거제	욕실용품
P0018	샤워기	욕실용품
P0019	린스	욕실용품
P0020	수건	욕실용품

< TB_PRD 테이블 >

PRD_TYPE
가전
가전
가전
가전
스마트폰
스마트폰
욕실용품
욕실용품
욕실용품
욕실용품
욕실용품
욕실용품
주방용품
주방용품
주방용품
주방용품
주방용품
주방용품
주방용품
컴퓨터
컴퓨터
컴퓨터

< IDX_PRD 인덱스 >

PRD_TYPE 컬럼 기준
오름차순 정렬된
인덱스를 생성!



2. 인덱스(INDEX) 원리 이해하기

인덱스(INDEX) 를 이용한 경우

```
SELECT PRD_ID      --상품ID
      , PRD_NAME   --상품명
      , PRD_TYPE   --상품타입
FROM TB_PRD
WHERE PRD_TYPE = '스마트폰';
```

WHERE 조건에 인덱스가 있는 컬럼이
사용되면 인덱스 사용 가능

스크립트 출력 x | 질의 결과 x | 계획 설명 x

SQL | 0.013초

OPERATION	OBJECT_NAME	OPTIONS
SELECT STATEMENT		
TABLE ACCESS	TB_PRD	BY INDEX ROWID
INDEX	IDX_PRD	RANGE SCAN
Access Predicates		
PRD_TYPE='스마트폰'		

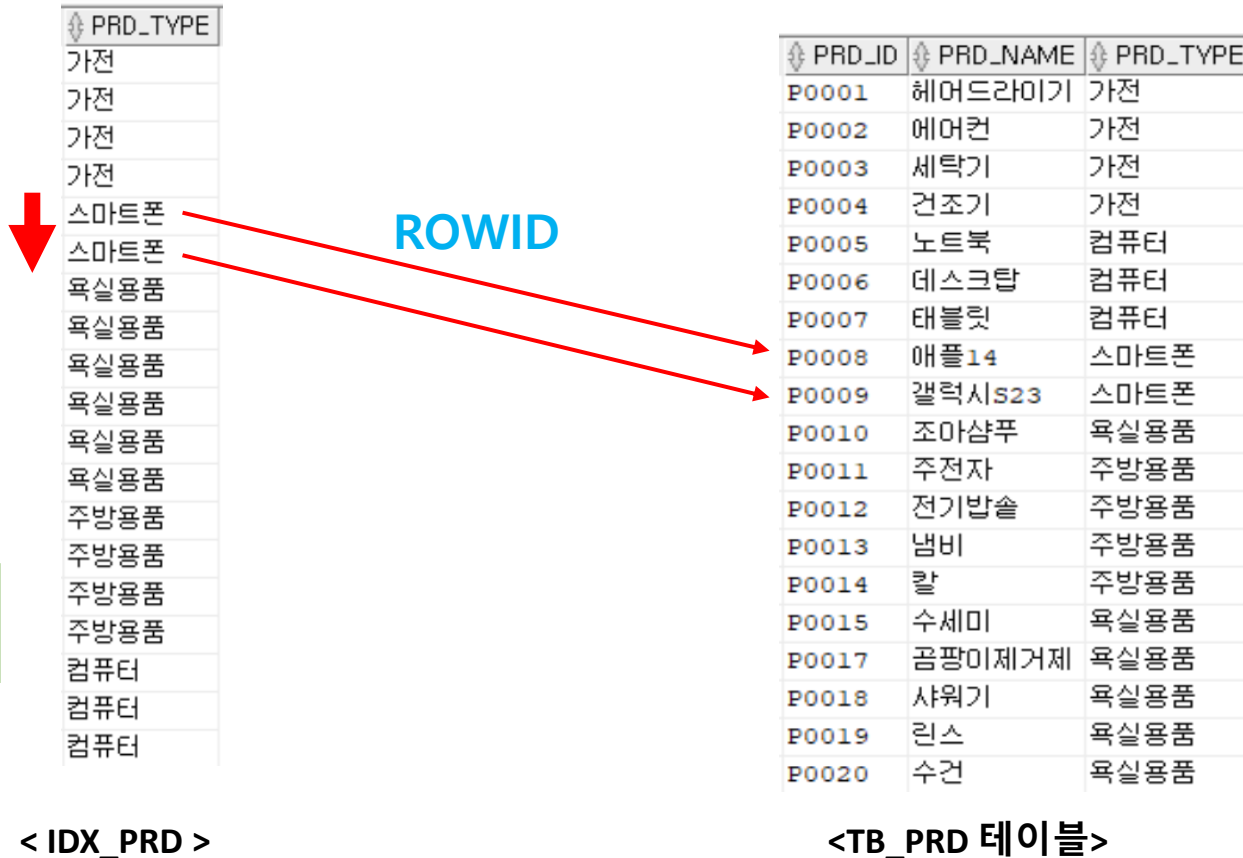
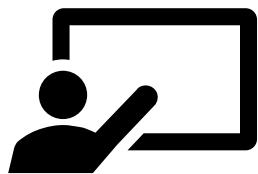
실행 계획을 보면 INDEX RANGE SCAN 이 떠있습니다.
이는 인덱스를 이용해서 테이블의 특정 범위의 데이터만 조회한 걸 의미합니다.

인덱스(INDEX) 원리 체크

```
SELECT PRD_ID      --상품ID
      , PRD_NAME    --상품명
      , PRD_TYPE    --상품타입
FROM TB_PRD
WHERE PRD_TYPE = '스마트폰';
```

WHERE 조건에 인덱스가 있는 컬럼이
사용되면 인덱스 사용 가능

인덱스는 이미 정렬이 되어있으므로 바로
"스마트폰" 부분을 찾아도 됩니다.



2. 인덱스(INDEX) 원리 이해하기

인덱스(INDEX) 의 목적은 조회성능 향상

```
SELECT PRD_ID      --상품ID
      , PRD_NAME    --상품명
      , PRD_TYPE    --상품타입
FROM TB_PRD
WHERE PRD_TYPE = '스마트폰';
```

PRD_ID	PRD_NAME	PRD_TYPE
P0001	헤어드라이기	가전
P0002	에어컨	가전
P0003	세탁기	가전
P0004	건조기	가전
P0005	노트북	컴퓨터
P0006	데스크탑	컴퓨터
P0007	태블릿	컴퓨터
P0008	애플14	스마트폰
P0009	갤럭시S23	스마트폰
P0010	조아샴푸	욕실용품
P0011	주전자	주방용품
P0012	전기밥솥	주방용품
P0013	냄비	주방용품
P0014	칼	주방용품
P0015	수세미	욕실용품
P0017	곰팡이제거제	욕실용품
P0018	샤워기	욕실용품
P0019	린스	욕실용품
P0020	수건	욕실용품

인덱스 없을 때

VS

PRD_TYPE	PRD_ID	PRD_NAME	PRD_TYPE
가전	P0001	헤어드라이기	가전
가전	P0002	에어컨	가전
가전	P0003	세탁기	가전
가전	P0004	건조기	가전
스마트폰	P0005	노트북	컴퓨터
스마트폰	P0006	데스크탑	컴퓨터
욕실용품	P0007	태블릿	컴퓨터
욕실용품	P0008	애플14	스마트폰
욕실용품	P0009	갤럭시S23	스마트폰
욕실용품	P0010	조아샴푸	욕실용품
욕실용품	P0011	주전자	주방용품
욕실용품	P0012	전기밥솥	주방용품
주방용품	P0013	냄비	주방용품
주방용품	P0014	칼	주방용품
주방용품	P0015	수세미	욕실용품
주방용품	P0017	곰팡이제거제	욕실용품
컴퓨터	P0018	샤워기	욕실용품
컴퓨터	P0019	린스	욕실용품
컴퓨터	P0020	수건	욕실용품

인덱스 있을 때

2. 인덱스(INDEX) 원리 이해하기

대신 DML 성능이 감소합니다.(TRADE-OFF)

```
INSERT INTO TB_PRD ( PRD_ID, PRD_NAME, PRD_TYPE, PRD_INFO, PRD_PRICE, SELL_COMP_NAME, REG_DATE )
VALUES ( 'P0021' , '샴푸' , '욕실용품' , '머리를 감을 수 있습니다' , 15000 , '다우니' , SYSDATE ) ;
```

PRD_ID	PRD_NAME	PRD_TYPE
P0001	헤어드라이기	가전
P0002	에어컨	가전
P0003	세탁기	가전
P0004	건조기	가전
P0005	노트북	컴퓨터
P0006	데스크탑	컴퓨터
P0007	태블릿	컴퓨터
P0008	애플14	스마트폰
P0009	갤럭시S23	스마트폰
P0010	조아샴푸	욕실용품
P0011	주전자	주방용품
P0012	전기밥솥	주방용품
P0013	냄비	주방용품
P0014	칼	주방용품
P0015	수세미	욕실용품
P0017	곰팡이제거제	욕실용품
P0018	샤워기	욕실용품
P0019	린스	욕실용품
P0020	수건	욕실용품
P0021	샴푸	욕실용품

PRD_TYPE
가전
가전
가전
가전
스마트폰
스마트폰
욕실용품
욕실용품
욕실용품
욕실용품
욕실용품
욕실용품
욕실용품
주방용품
주방용품
주방용품
주방용품
주방용품
주방용품
주방용품
컴퓨터
컴퓨터
컴퓨터

값이 입력/수정/삭제 가 되면
인덱스도 똑같이 반영됩니다.
(테이블 내용도 바뀌고 인덱스 내용도 바뀌고..)

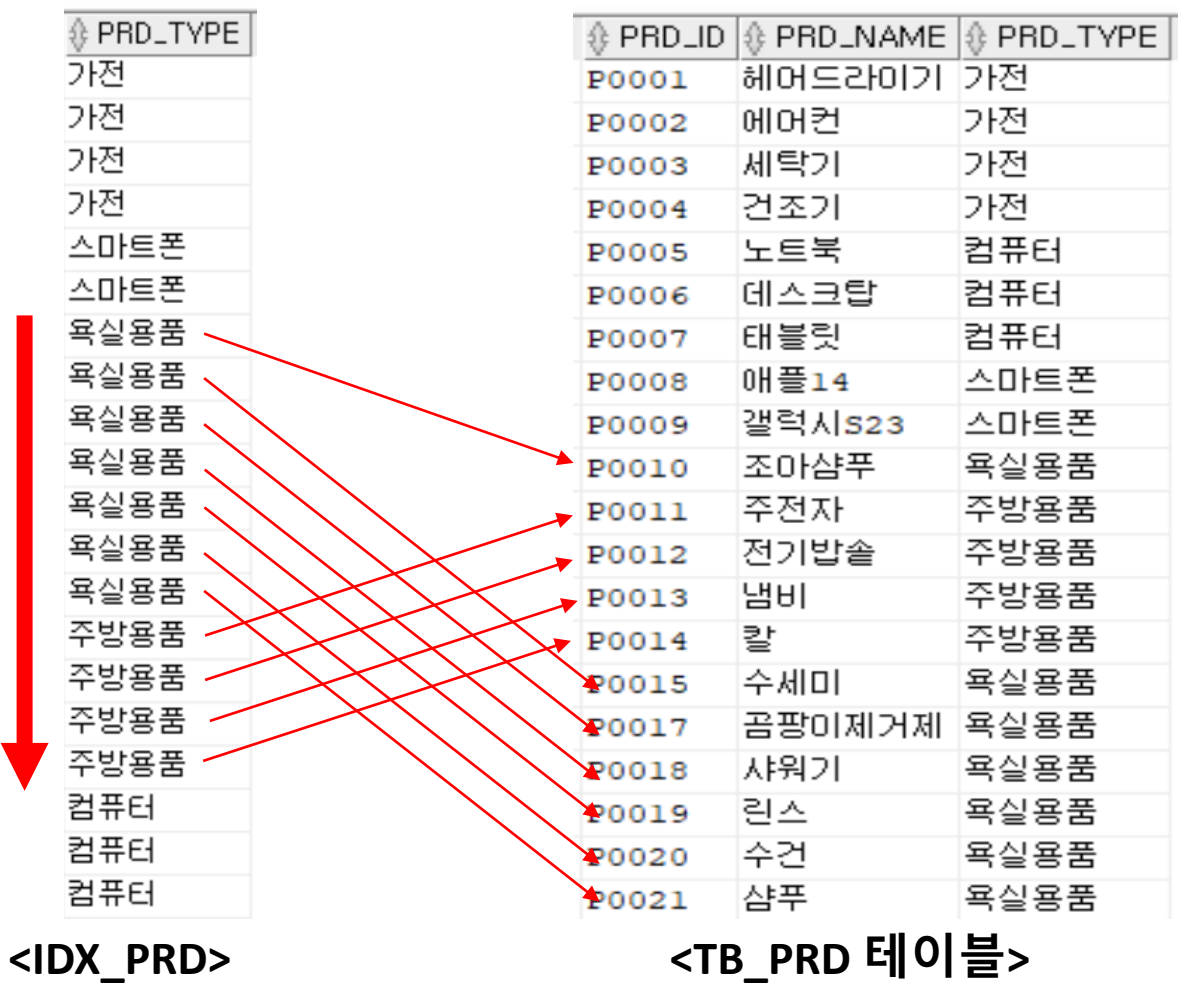


2. 인덱스(INDEX) 원리 이해하기

INDEX를 이용해 조회하는게 무조건 좋을까? **NO!**

```
SELECT PRD_ID      --상품ID
      , PRD_NAME    --상품명
      , PRD_TYPE    --상품타입
FROM TB_PRD
WHERE PRD_TYPE IN ( '주방용품' , '욕실용품' ) ;
```

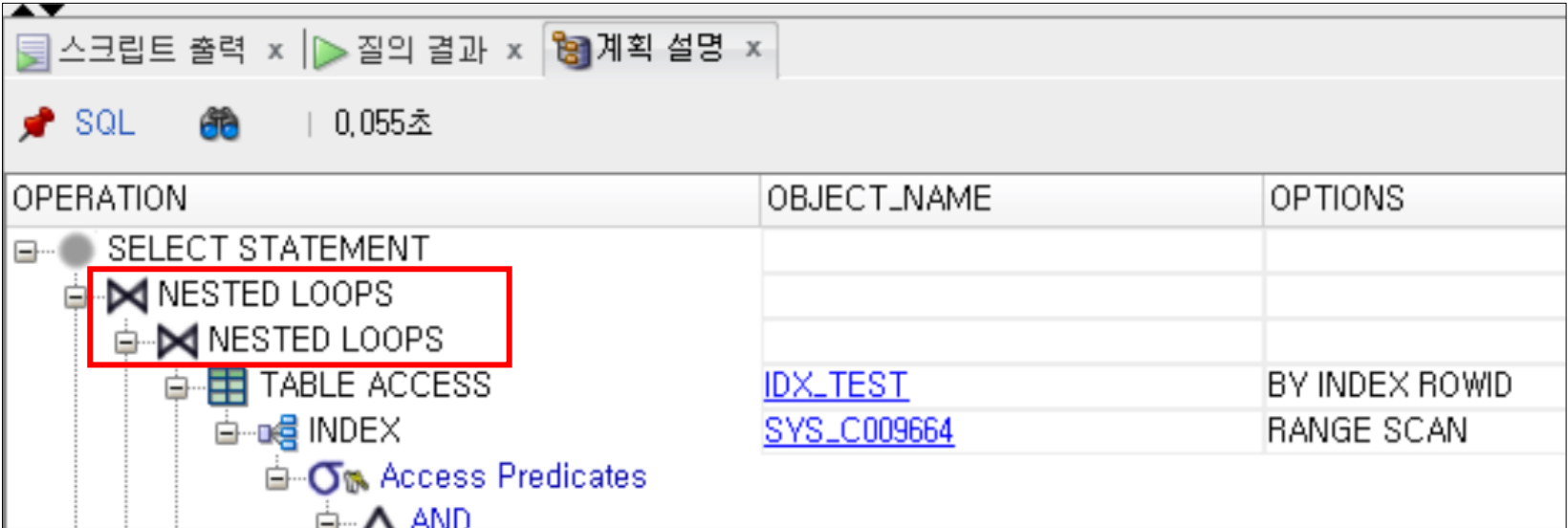
대량의 데이터 조회에는
인덱스가 오히려 안좋습니다.
차라리 TABLE FULL SCAN 이 좋습니다.
(배보다 배꼽이 더 큰 경우)



3. NL(Nested Loop) 조인 이해하기

NL(Nested Loop) 조인이란?

실제로 조인이 어떻게 발생하는지 물리적인 관점에서 보는 것으로 **인덱스가 있을 때만 쓸 수 있는** 기법
이 외에도 Hash Join , Sort Merge Join 이 있지만 대부분 웹사이트에서 쓰는 방식은 NL조인 방식



OPERATION	OBJECT_NAME	OPTIONS
SELECT STATEMENT		
NESTED LOOPS		
NESTED LOOPS		
TABLE ACCESS	IDX_TEST	BY INDEX ROWID
INDEX	SYS_C009664	RANGE SCAN
Access Predicates		
AND		

3. NL(Nested Loop) 조인 이해하기

NL 조인의 원리 (중첩 반복문 처럼 사용)

```
SELECT A.MEMBER_ID
      , A.MEMBER_NAME
      , A.GRADE_CD
      , B.GRADE_NAME
FROM TB_MEMBER A
      , TB_GRADE B
WHERE A.GRADE_CD = B.GRADE_CD ;
```

< TB_MEMBER 테이블 >

MEMBER_ID	MEMBER_NAME	GRADE_CD
AAAAA	사용자A	1
BBBBB	사용자B	2
CCCCC	사용자C	1
DDDDD	사용자D	3
EEEEE	사용자E	1
FFFFF	사용자F	3
GGGGG	사용자G	2
HHHHH	사용자H	5
IIIII	사용자I	4

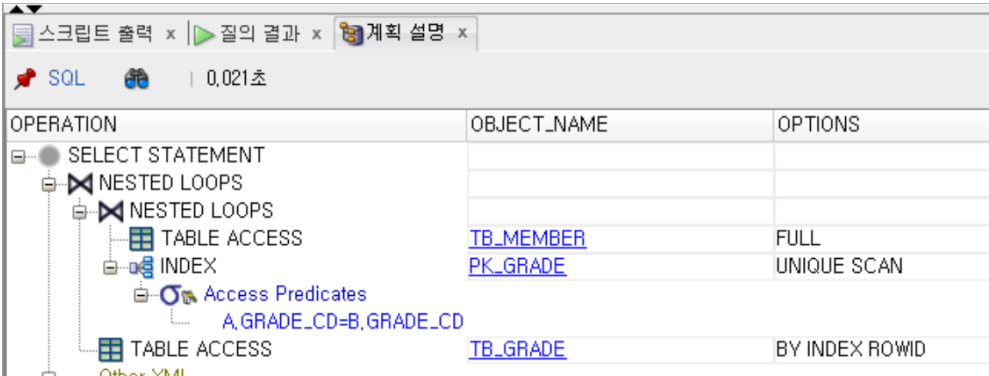
< TB_GRADE 테이블 >

GRADE_CD	GRADE_NAME
1	브론즈
2	실버
3	골드
4	VIP
5	VVIP

3. NL(Nested Loop) 조인 이해하기

NL 조인의 원리 (만약 중간에 **인덱스가 있다면?**)

```
SELECT A.MEMBER_ID
      , A.MEMBER_NAME
      , A.GRADE_CD
      , B.GRADE_NAME
FROM TB_MEMBER A
      , TB_GRADE B
WHERE A.GRADE_CD = B.GRADE_CD ;
```



< TB_MEMBER 테이블 >

MEMBER_ID	MEMBER_NAME	GRADE_CD
AAAAA	사용자A	1
BBBBB	사용자B	2
CCCCC	사용자C	1
DDDDD	사용자D	3
EEEEE	사용자E	1
FFFFF	사용자F	3
GGGGG	사용자G	2
HHHHH	사용자H	5
IIIII	사용자I	4

< PK_GRADE 인덱스 >

GRADE_CD
1
2
3
4
5

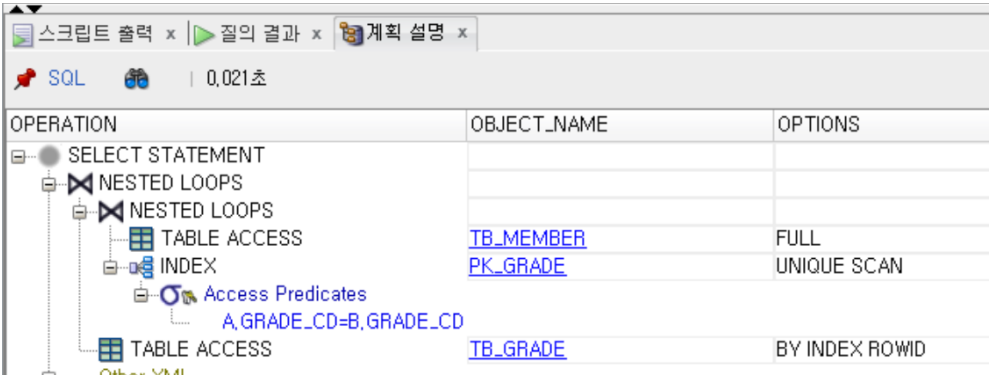
< TB_GRADE 테이블 >

GRADE_CD	GRADE_NAME
1	브론즈
2	실버
3	골드
4	VIP
5	VVIP

3. NL(Nested Loop) 조인 이해하기

NL 조인의 원리 (만약 **앞에도 인덱스가 있다면?**)

```
SELECT A.MEMBER_ID
      , A.MEMBER_NAME
      , A.GRADE_CD
      , B.GRADE_NAME
FROM TB_MEMBER A
      , TB_GRADE B
WHERE A.GRADE_CD = B.GRADE_CD
AND A.GRADE_CD = 1 ;
```



< PK_MEMBER 테이블 >

GRADE_CD
1
1
1
2
2
3
3
4
5

< TB_MEMBER 테이블 >

MEMBER_ID	MEMBER_NAME	GRADE_CD
AAAAA	사용자A	1
BBBBB	사용자B	2
CCCCC	사용자C	1
DDDDD	사용자D	3
EEEEE	사용자E	1
FFFFF	사용자F	3
GGGGG	사용자G	2
HHHHH	사용자H	5
IIIII	사용자I	4

< PK_GRADE 인덱스 >

GRADE_CD
1
2
3
4
5

< TB_GRADE 테이블 >

GRADE_CD	GRADE_NAME
1	브론즈
2	실버
3	골드
4	VIP
5	VVIP

[1] NL (Nested Loop) Join 장단점

장점

인덱스를 활용, 랜덤 액세스를 통해 **소량의 데이터 추출** 가능

대부분의 웹사이트에서는 NL조인을 주로 활용하여 활용도가 높다

단점

인덱스가 있어야만 사용할 수 있다.

소량의 데이터가 아니라면 NL조인은 매우 비효율적이다.

3. NL(Nested Loop) 조인 이해하기

아래 쿼리를 실행해봅시다.

```
DROP TABLE TB_TEST1 ;  
DROP TABLE TB_TEST2 ;
```

--150만건 임시데이터 생성하기

```
CREATE TABLE TB_TEST1 (  
    COL1 NUMBER NOT NULL , COL2 NUMBER NOT NULL , COL3 NUMBER NOT NULL , COL4 NUMBER NOT NULL );
```

```
INSERT INTO TB_TEST1  
SELECT /*+ APPEND */ LEVEL AS COL1 , LEVEL AS COL2 , LEVEL AS COL3 , LEVEL AS COL4  
    FROM DUAL  
CONNECT BY LEVEL <= 1500000 ;
```

```
COMMIT;
```

--150만건 임시데이터 생성하기

```
CREATE TABLE TB_TEST2 (  
    COL1 NUMBER NOT NULL , COL2 NUMBER NOT NULL , COL3 NUMBER NOT NULL , COL4 NUMBER NOT NULL );
```

```
INSERT INTO TB_TEST2  
SELECT /*+ APPEND */ LEVEL AS COL1 , LEVEL AS COL2 , LEVEL AS COL3 , LEVEL AS COL4  
    FROM DUAL  
CONNECT BY LEVEL <= 1500000 ;
```

```
COMMIT;
```

3. 인덱스 실습해보기

1.

```
SELECT /*+ ORDERED USE_NL(B) */ *  
FROM TB_TEST1 A  
    , TB_TEST2 B  
WHERE A.COL1 = B.COL1 ;
```

-> TB_TEST1 와 TB_TEST2 를 NL 조인한다. (인덱스가 없는 상태)

2.

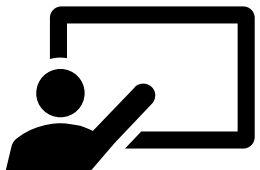
```
CREATE INDEX IDX_TEST2 ON TB_TEST2 (COL1) ;
```

 -> TB_TEST2 에 대한 인덱스를 생성한다.

3.

```
SELECT /*+ ORDERED USE_NL(B) */ *  
FROM TB_TEST1 A  
    , TB_TEST2 B  
WHERE A.COL1 = B.COL1 ;
```

-> 다시 실행해본다 (인덱스가 중간에 존재하는 상태)



실행 결과, 인덱스의 존재만으로 속도가 매우 빨라진 것을 알 수 있으며,
실제 현업에서도 인덱스를 이용해 성능을 높이는 편입니다.

인덱스(INDEX) 삭제하기

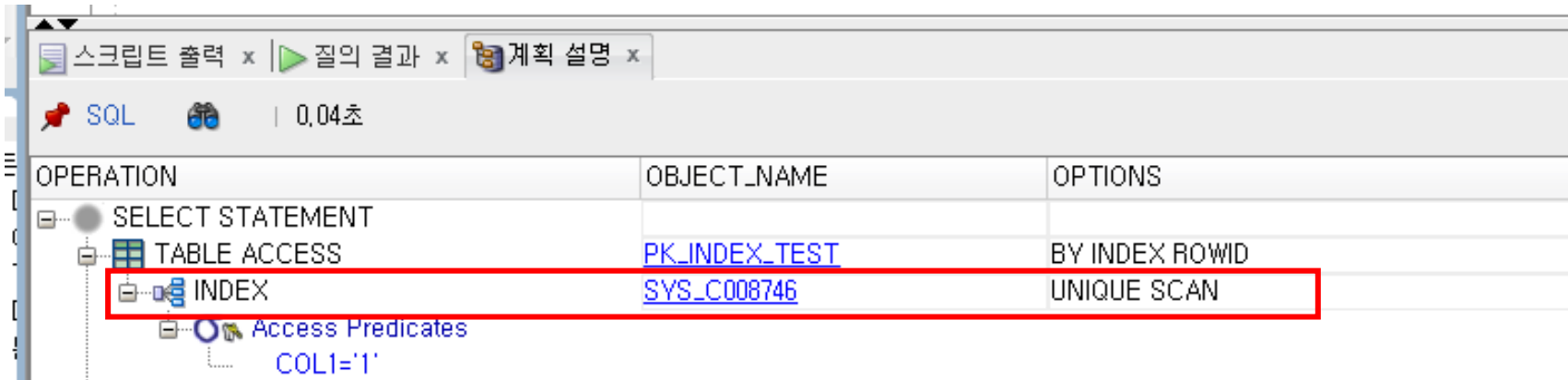
```
DROP INDEX IDX_PRD ;
```

Index IDX_PRD이 (가) 삭제되었습니다.

PK 생성시 기본 인덱스가 **자동으로 생성**됩니다.

(그 외에는 직접 생성을 해야 함)

```
CREATE TABLE PK_INDEX_TEST (  
COL1 VARCHAR2(10) PRIMARY KEY , --PK 설정한 컬럼에 대해 자동으로 인덱스 생성  
COL2 VARCHAR2(10) );  
  
INSERT INTO PK_INDEX_TEST VALUES ( 1,1);  
INSERT INTO PK_INDEX_TEST VALUES ( 2,2);  
INSERT INTO PK_INDEX_TEST VALUES ( 3,2);  
  
SELECT /*+ RULE */ * FROM PK_INDEX_TEST WHERE COL1 = '1' ;
```



스คร립트 출력 x | 질의 결과 x | 계획 설명 x

SQL | 0.04초

OPERATION	OBJECT_NAME	OPTIONS
SELECT STATEMENT		
TABLE ACCESS	PK_INDEX_TEST	BY INDEX ROWID
INDEX	SYS_C008746	UNIQUE SCAN
Access Predicates		
COL1='1'		

INDEX

END