인덱스

배재대학교 컴퓨터공학과
김창수

목차

I. 인덱스 개념

II. 인덱스의 구조

III. 인덱스의 효율적인 사용 방법

IV. 인덱스 종류 및 생성 방법

V. 인덱스 실행 경로 확인

VI. 인덱스 관리

인덱스의 개념

- 인덱스란?
  - 인덱스는 SQL 명령문의 처리 속도를 향상시키기 위해 칼럼에 대해 생성하는 객체
  - 인덱스는 포인트를 이용하여 테이블에 저장된 데이터를 랜덤 액세스하기 위한 목적으로 사용
    - 실생활의 예 : 책의 색인(찾아보기)

- 데이터를 조회하는 대표적인 방법
  - 전체 테이블 검색(full table scan) 방법
    - 인덱스를 사용하지 않고 데이터를 검색하는 경우
    - 테이블 전체를 처음부터 끝까지 순차적으로 조회해야 함
    - 대량의 데이터 중, 한 두건의 데이터를 검색하기 위해 테이블 전체를 검색하는 것은 매우 비효율적
  - 인덱스 검색 (index scan) 방법
    - 검색 조건으로 사용하는 칼럼에 인덱스를 생성하여 랜덤 액세스하는 방법
    - 소량의 데이터를 검색하는 경우에 전체 테이블 검색에 비해 매우 효율적

목차

I. 인덱스 개념

II. 인덱스의 구조

III. 인덱스의 효율적인 사용 방법

IV. 인덱스 종류 및 생성 방법

V. 인덱스 실행 경로 확인

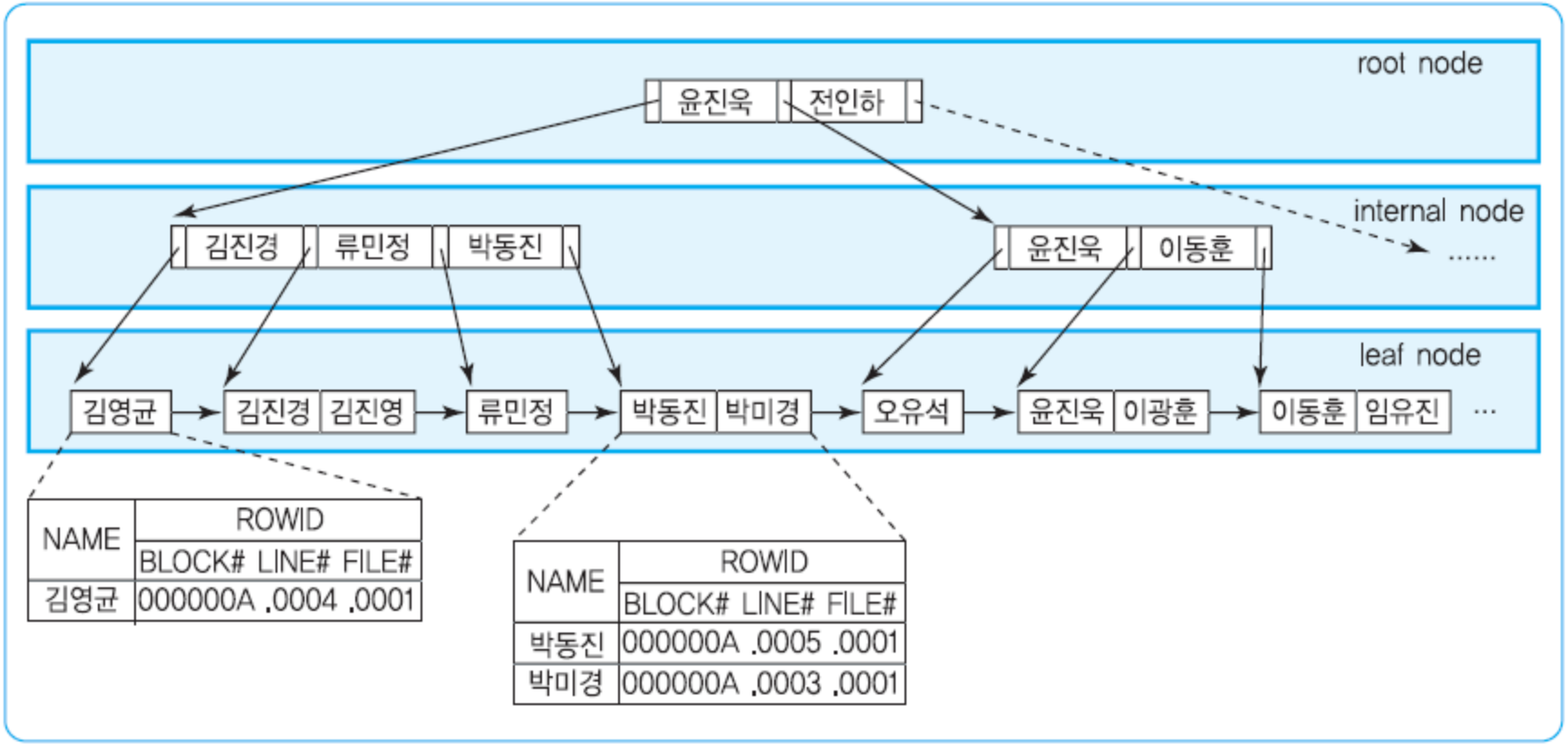
VI. 인덱스 관리

인덱스의 구조

- 오라클의 인덱스 구조
  - B*-트리 형식으로 구성
  - 구성요소
    - 테이블에서 인덱싱되는 칼럼의 값과 포인터의 역할을 하는 각 행의 ROWID
  - B*-트리에서 칼럼 값은 오름차순이나 내림차순으로 정렬
    - 칼럼 값이 같은 경우 각 행의 ROWID에 의해 결정
  - 테이블과는 독립적으로 생성하거나 삭제 가능
  - 하나의 테이블에 여러 개의 인덱스 생성 가능
  - 여러 인덱스에서 동일한 칼럼을 중복적으로 사용 가능

인덱스의 구조

- 인덱스의 내부 구조는 B*-트리 형식으로 구성



목차

I. 인덱스 개념

II. 인덱스의 구조

III. 인덱스의 효율적인 사용 방법

IV. 인덱스 종류 및 생성 방법

V. 인덱스 실행 경로 확인

VI. 인덱스 관리

인덱스의 효율적인 사용 방법

- **인덱스의 일반적인 사용 방법**
  - SQL 명령문의 처리 속도를 개선하기 위한 목적으로 주로 사용
    - SQL 명령문의 검색 결과는 인덱스 사용 유무와 상관없이 동일
    - 인덱스는 접근 경로나 처리 속도에만 영향을 줌
  - 물리적인 데이타베이스 설계 관점에서 인덱스는 테이블과는 다른 하드 디스크에 저장하는 것이 좋음

- **인덱스 사용이 효율적인 경우**
  - WHERE 절이나 조인 조건절에서 자주 사용되는 칼럼
  - 전체 데이터 중에서 10~15% 분포이내의 데이터를 검색하는 경우
    - 분포도가 높은 데이터 검색시 인덱스를 사용하면 전체 테이블 스캔 방식보다 더 느려지는 경우 발생
  - 두 개 이상의 칼럼이 WHERE절이나 조인 조건에서 자주 사용되는 경우
  - 테이블에 저장된 데이터의 변경이 드문 경우

인덱스의 종류 및 생성 방법

- **인덱스의 종류**
  - 인덱스 칼럼 값의 중복 여부에 따라
    - 고유 인덱스(unique index), 비고유 인덱스(non-unique index)
  - 칼럼의 결합 여부에 따라
    - 단일 인덱스(single index), 결합 인덱스(composite index)
  - 연산자 또는 함수의 적용 결과에 의해 생성되는 인덱스
    - 함수 기반 인덱스(function-based index)

- **인덱스 생성 방법**
  - 묵시적인 인덱스 생성
    - 테이블 생성시 기본 키나 고유 키 무결성 제약조건을 생성한 경우 고유 인덱스가 자동 생성
    - 인덱스 이름은 무결성 제약조건 이름과 동일
  - 명시적인 인덱스 생성
    - CREATE INDEX 명령문으로 사용자가 인덱스 필요시마다 생성

목차

I. 인덱스 개념

II. 인덱스의 구조

III. 인덱스의 효율적인 사용 방법

IV. 인덱스 종류 및 생성 방법

V. 인덱스 실행 경로 확인

VI. 인덱스 관리

인덱스의 생성

- **인덱스 생성 방법**

사용법

```
CREATE [ UNIQUE]  INDEX index
ON table (column1[ ASC|DESC] [ , column2[ ASC|DESC] , ...] );
```

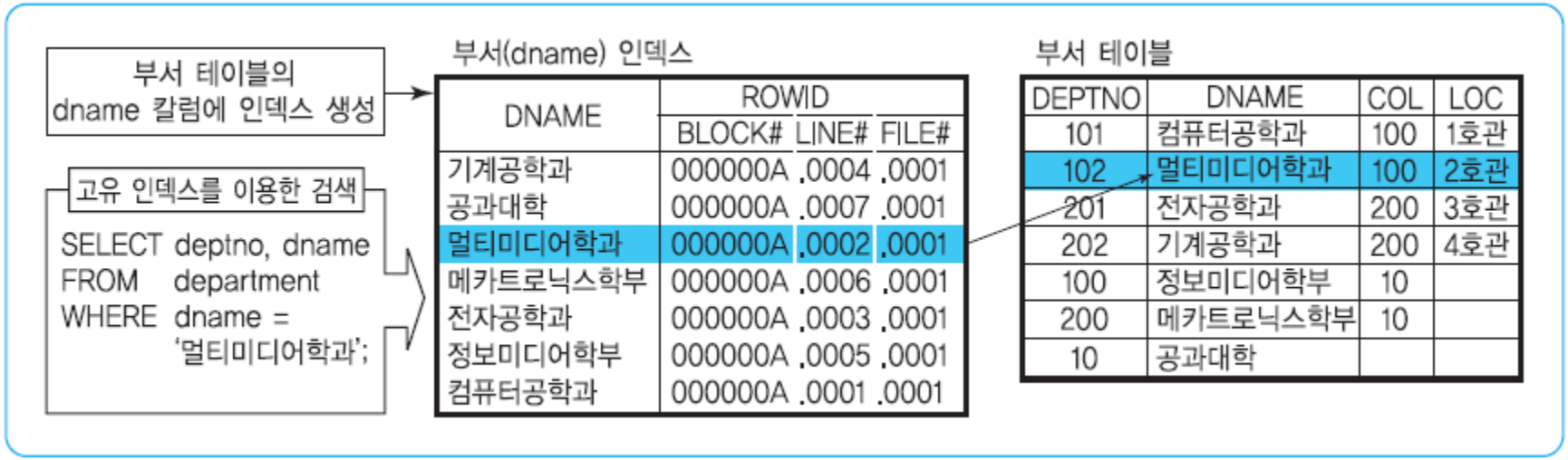
▶ UNIQUE : 고유 인덱스 지정

▶ ASC|DESC : 인덱스 키를 오름차순 또는 내림차순으로 정렬

- **고유 인덱스 생성**
  - 유일한 값을 가지는 칼럼에 대해 생성하는 인덱스
  - 모든 인덱스 키는 테이블의 하나의 행과 연결
  - 기본 키와 고유 키 무결성 제약조건을 정의하면 묵시적으로 고유 인덱스가 생성

고유 인덱스

- **고유 인덱스를 이용한 검색 예**



사용예

부서 테이블에서 name 칼럼을 고유 인덱스로 생성하여라. 단, 고유 인덱스의 이름을 idx_dept_name으로 정의한다.

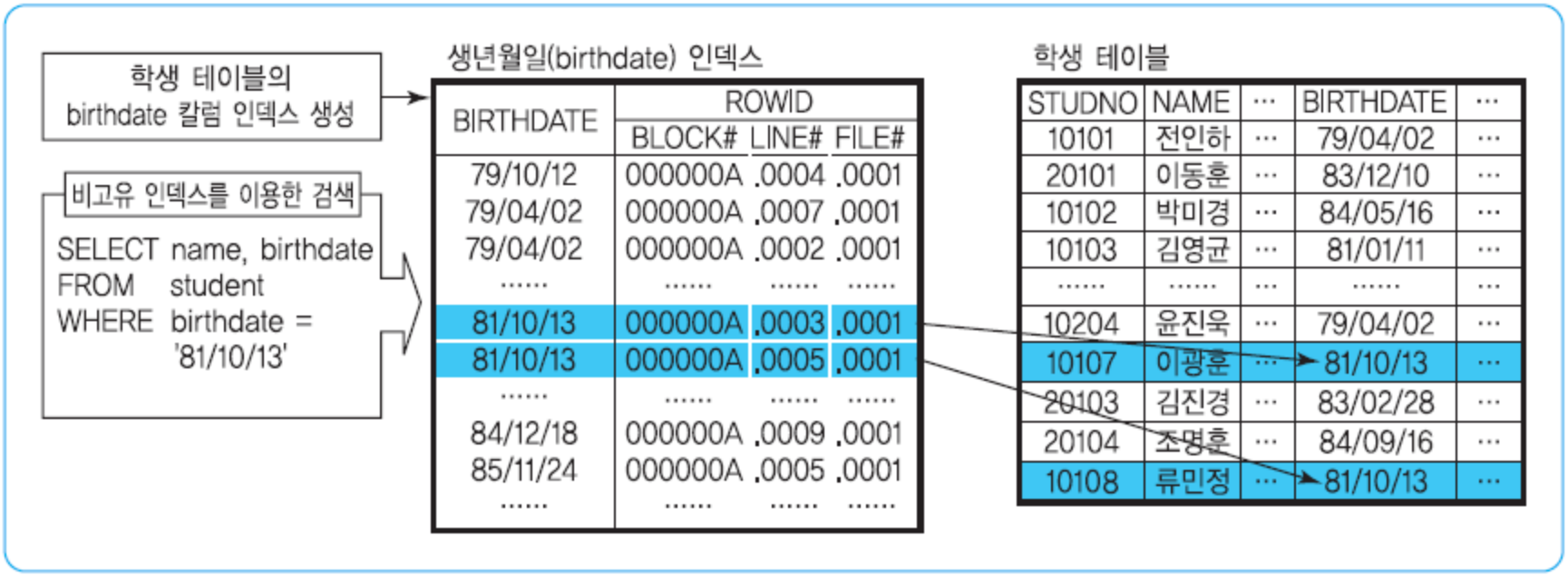
```
SQL> CREATE UNIQUE INDEX idx_dept_name
  2  ON department(dname);

인덱스가 생성되었습니다.
```

만일 부서 이름이 중복되면 고유 인덱스가 생성되지 않는다.

비고유 인덱스

- **비고유 인덱스 생성**
  - 중복된 값을 가지는 칼럼에 대해 생성하는 인덱스
  - 하나의 인덱스 키는 테이블의 여러 행과 연결 가능

- **비고유 인덱스를 이용한 검색 예**



비고유 인덱스, 단일 인덱스

- **비고유 인덱스 사용 예**

사용예

학생 테이블의 birtdate 칼럼을 비고유 인덱스로 생성하여라. 비고유 인덱스의 이름은 idx_stud_birthdate로 정의한다.

```
SQL> CREATE INDEX idx_stud_birthdate
  2  ON student(birthdate);

인덱스가 생성되었습니다.
```

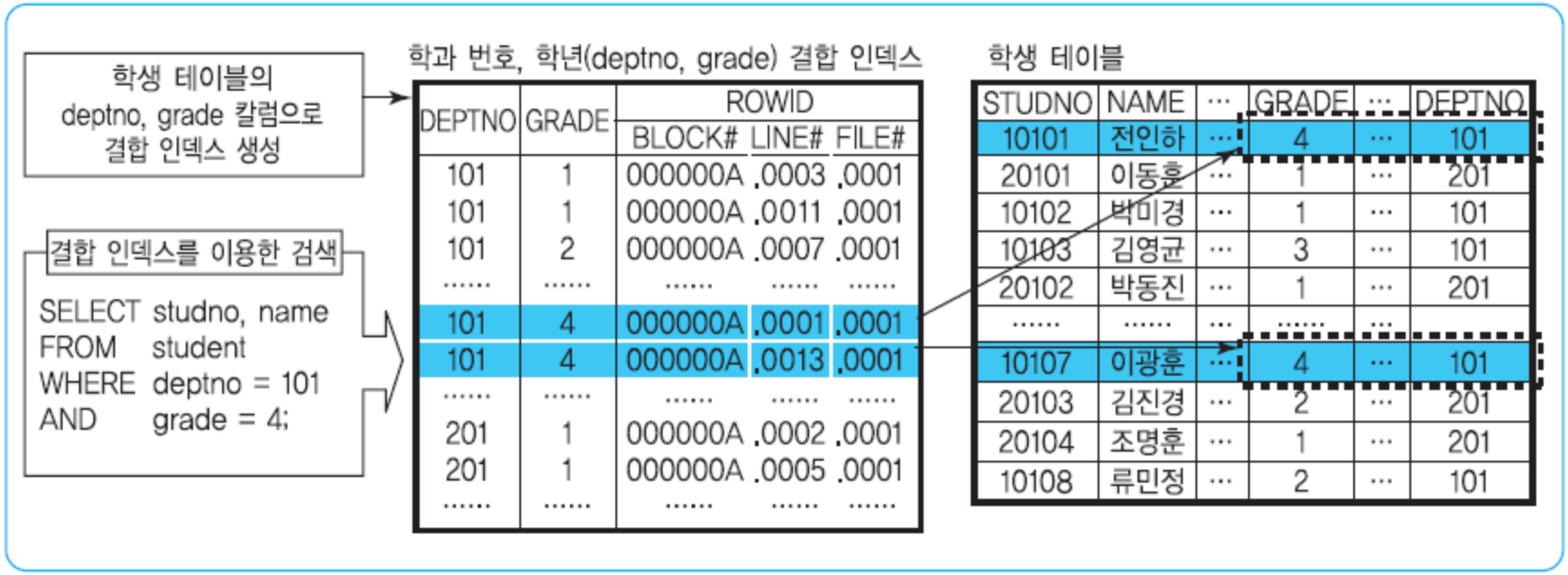
- **단일 인덱스**
  - 하나의 칼럼으로만 구성된 인덱스
  - 예 : 학생 테이블의 생년월일 칼럼에 생성한 idx_stud_birthdate

결합 인덱스

- **결합 인덱스**
  - 두 개 이상의 칼럼을 결합하여 생성하는 인덱스
  - WHERE 절의 조건 비교에서 두 개 이상의 칼럼이 AND 연산으로 자주 연결 되는 경우 주로 생성
  - 오라클에서는 결합 인덱스의 칼럼 수를 최대 32개까지 허용
  - 결합 인덱스를 구성하는 칼럼은 테이블의 칼럼 순서와 일치하거나 인접할 필요는 없음

결합 인덱스

- **결합 인덱스를 이용한 검색 예**



사용예

학생 테이블의 deptno, grade 칼럼을 결합 인덱스로 생성하여라. 결합 인덱스의 이름은 idx_stud_dno_grade로 정의한다.

```
SQL> CREATE INDEX idx_stud_dno_grade
  2  ON student(deptno, grade);

인덱스가 생성되었습니다.
```

DESCENDING INDEX

- DESCENDING INDEX 개념
  - 칼럼별로 정렬 순서를 별도로 지정하여 결합 인덱스를 생성하기 위한 방법
    - 기본적으로 결합 인덱스에서 칼럼의 정렬 순서는 오름차순

- DESCENDING INDEX 사용 예

사용예

학생 테이블에서 deptno와 name 칼럼으로 결합 인덱스를 생성하여라. 단, deptno 칼럼은 내림차순으로 name 칼럼은 오름차순으로 생성하여라.

```
SQL> CREATE INDEX idx_stud_no_name ON student(deptno DESC, name ASC);

인덱스가 생성되었습니다.
```

함수 기반 인덱스

- **일반적인 인덱스**
  - 칼럼에 저장된 값으로 생성
  - WHERE 절에서 칼럼 값에 연산을 적용하면 해당 인덱스 사용 불가
  - 예 : WHERE upper(userid) = 'MANDU'
    - USERID 칼럼의 값이 UPPER 함수로 인해 변형되어 USERID 인덱스 사용 불가
    - 전체 테이블 스캔 필요(질의처리 시간이 길어짐)

- **함수 기반 인덱스**
  - 오라클 8i 버전부터 지원하는 새로운 형태의 인덱스
  - 인덱스로 칼럼에 대한 연산이나 함수의 계산 결과를 인덱스로 생성 가능
  - INSERT, UPDATE시에는 새로운 값을 계산하여 인덱스에 추가

함수 기반 인덱스 생성 예

```
SQL> CREATE INDEX fidx_standard_weight ON student((height-100)*0.9);
CREATE INDEX fidx_standard_weight ON student((height-100)*0.9)
                                                          *

1행에 오류:
ORA-01031: 권한이 불충분합니다
```

함수기반 인덱스 생성을 위해서는 특정권한이 필요하다.

```
SQL> CONNECT
사용자명 입력: system
암호 입력: *******
연결되었습니다.
```

데이터베이스 관리자로 접속한다.

```
SQL> GRANT query rewrite TO scott;

권한이 부여되었습니다.
```

함수기반 인덱스를 사용할 수 있도록 'QUERY REWRITE' 권한을 SCOTT 사용자에게 부여한다.

```
SQL> CONNECT
사용자명 입력: scott
암호 입력: *****
연결되었습니다.
```

'QUERY REWRITE' 권한을 가진 사용자는 함수기반 인덱스를 생성할 수 있다.

```
SQL> CREATE INDEX fidx_standard_weight ON student((height-100)*0.9);

인덱스가 생성되었습니다.
```

목차

I. 인덱스 개념

II. 인덱스의 구조

III. 인덱스의 효율적인 사용 방법

IV. 인덱스 종류 및 생성 방법

V. 인덱스 실행 경로 확인

VI. 인덱스 관리

인덱스 실행 계획 확인

- 실행 계획(execution plan)
  - SQL 명령문이 내부적으로 처리되는 과정
    - SQL 명령문을 실행할 때, 인덱스를 이용하여 랜덤 액세스를 했는지, 전체 테이블 검색을 했는지 확인 필요
    - 실행 계획을 확인하여 더 나은 처리 속도를 지원하는 방식을 선택 가능
    - 처리 속도에 영향을 미치는 요인
      - cpu 사용시간
      - 하드디스크에서 읽은 물리적인 데이터 블록 수
      - 메모리에서 액세스한 논리적인 데이터 버퍼 수
      - 실행 단계별로 처리되는 행의 수 등
  - 오라클에서는 SQL 명령문을 처리하기 위해 사용된 내부적인 실행 경로를 확인할 수 있는 방법 제공
    - 세션별로 트레이스 파일(*.trc)을 생성하여 분석하는 방법
    - PLUSTRACE 롤을 사용자에게 부여하여 간단하게 분석하는 방법

인덱스 실행 계획 확인 예 1

사용예

학과 테이블에서 학과 이름이 '정보미디어학부'인 학과의 학과 번호와 학과 이름을 검색한 결과에 대한 실행경로를 분석하여라. dname칼럼에 고유 인덱스가 생성되어 있다.

☞ 학과 이름이 '정보미디어학부'인 학과 검색 : dname칼럼에 인덱스가 생성된 경우

```
SQL> SELECT deptno, dname
  2  FROM   department
  3  WHERE  dname = '정보미디어학부';
```

☞ 출력 결과

```
    DEPTNO DNAME
---------- ---------------
       100 정보미디어학부
```

dname 칼럼에 생성된 idx_dept_name 인덱스를 이용한 랜덤 액세스

☞ 실행 경로

```
Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   TABLE ACCESS (BY INDEX ROWID) OF 'department'
   2    1       INDEX (UNIQUE SCAN) OF 'IDX_DEPT_NAME' (UNIQUE)
```

인덱스 실행 계획 확인 예1

☞ idx_dept_name 인덱스 삭제

```
SQL> DROP INDEX idx_dept_name;
인덱스가 삭제되었습니다.

SQL> SELECT deptno, dname
  2  FROM   department
  3  WHERE  dname = '정보미디어학부';
```

☞ 출력 결과

```
    DEPTNO DNAME
---------- ---------------
       100 정보미디어학부
```

☞ 실행 경로

dname 칼럼에 생성된 인덱스가 삭제되어 전체 테이블 검색

```
Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   TABLE ACCESS (FULL) OF 'department'
```

인덱스 실행 계획 확인 예2

사용예

학생 테이블에서 생일이 '79/04/02'인 학생의 학생 이름과 생년월일을 검색한 결과에 대한 실행 경로를 분석하여라. birthdate 칼럼에 대해 비고유 인덱스가 생성되어 있다.

☞ 생일이 '79/04/02'인 학생 이름 검색: birthdate 칼럼에 인덱스가 생성된 경우

```
SQL>  SELECT name, birthdate
  2   FROM   student
  3   WHERE  birthdate = '79/04/02';
```

☞ 출력 결과

```
NAME       BIRTHDAT
---------- --------
윤진욱     79/04/02
```

☞ 실행 경로

```
Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   TABLE ACCESS (BY INDEX ROWID) OF 'student'
   2    1   INDEX (RANGE SCAN) OF 'IDX_STUD_BIRTHDATE' (NON-UNIQUE)
```

birthdate 칼럼에 생성된 idx_stud_birthdate 인덱스를 이용한 랜덤 액세스

인덱스 실행 계획 확인 예2

☞ 비고유 인덱스 삭제

```
SQL> DROP INDEX idx_stud_birthdate;
인덱스가 삭제되었습니다.

SQL> SELECT name, birthdate
  2  FROM   student
  3  WHERE  birthdate = '79/04/02';
```

☞ 실행 경로

```
Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   TABLE ACCESS (FULL) OF 'student'
```

birthdate 칼럼에 생성된 인덱스가 없으므로 전체 테이블 검색

인덱스 실행 계획 확인 예3

사용예

학생 테이블에서 101번 학과의 4학년 학생들의 학번과 이름을 검색한 결과에 대해 실행 경로를 분석하여라. 인덱스는 studno와 name 칼럼에 대해 결합 인덱스로 생성되어 있다.

☞ 101번 학과의 4학년 학생 이름 검색: studno와 nam칼럼에 결합 인덱스가 생성된 경우

```
SQL> SELECT studno, name
  2  FROM   student
  3  WHERE  deptno = 101 and grade='4';
```

☞ 출력 결과

```
    STUDNO NAME
---------- ----------
     10101 전인하
     10107 이광훈
```

☞ 실행 경로

deptno와 grade 칼럼에 생성된 idx_stud_dno_grade 결합 인덱스를 이용한 랜덤 액세스

```
Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   TABLE ACCESS (BY INDEX ROWID) OF 'student'
   2    1    INDEX (RANGE SCAN) OF 'IDX_STUD_DNO_GRADE' (NON-UNIQUE)
```

인덱스 실행 계획 확인 예3

☞ idx_stud_dno_grade 인덱스 삭제

```
SQL> DROP INDEX idx_stud_dno_grade;
인덱스가 삭제되었습니다.

SQL> SELECT studno, name
  2  FROM   student
  3  WHERE  deptno = 101 and grade='4';
```

☞ 실행 경로

deptno와 grade 칼럼에 생성된 인덱스가 삭제되어 전체 테이블 검색

```
Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   TABLE ACCESS (FULL) OF 'student'
```

목차

I. 인덱스 개념

II. 인덱스의 구조

III. 인덱스의 효율적인 사용 방법

IV. 인덱스 종류 및 생성 방법

V. 인덱스 실행 경로 확인

VI. 인덱스 관리

인덱스 정보 조회

- USER_INDEXES
  - 인덱스 이름과 유일성 여부 등을 확인 가능한 데이터 딕셔너리

사용예

학생 테이블에 생성된 인덱스를 조회하여라.

```
SQL> SELECT index_name, uniqueness
  2  FROM   user_indexes
  3  WHERE  table_name = 'STUDENT';

☞ 출력 결과
INDEX_NAME                     UNIQUENESS
------------------------------ --------------
FIDX_STANDARD_WEIGHT           NONUNIQUE
IDX_STUD_NO_NAME               NONUNIQUE
STUD_IDNUM_UK                  UNIQUE
STUD_NO_PK                     UNIQUE
STUD_USERID_UK                 UNIQUE
```

인덱스 정보 조회

- USER_IND_COLUMNS
  - 인덱스 이름, 인덱스가 생성된 테이블 이름과 칼럼 이름 등을 확인 가능한 데이터 딕셔너리

```
SQL> SELECT index_name, column_name
  2  FROM   user_ind_columns
  3  WHERE  table_name='STUDENT';

☞ 출력 결과
INDEX_NAME                     COLUMN_NAME
------------------------------ -------------
STUD_NO_PK                     STUDNO
STUD_USERID_UK                 USERID
STUD_IDNUM_UK                  IDNUM
IDX_STUD_NO_NAME               SYS_NC00012$
IDX_STUD_NO_NAME               NAME
FIDX_STANDARD_WEIGHT           SYS_NC00013$

6 개의 행이 선택되었습니다.
```

인덱스 삭제

- **인덱스 삭제**
  - DROP INDEX **명령문을 사용하여 인덱스 삭제 가능**
  - **기본 키나 고유 키 제약조건 생성으로 오라클에 의해 묵시적으로 생성된 인덱스는** DROP INDEX **명령문으로 삭제 불가**

사용법

```
DROP INDEX index;
```

인덱스 삭제 사용 예

사용예

학생 테이블에 생성한 idx_stud_no_name 인덱스와 기본 키 제약조건에 의해 자동으로 생성된 stud_no_pk 인덱스를 삭제하여라.

```
SQL> DROP INDEX idx_stud_no_name;

인덱스가 삭제되었습니다.

SQL> DROP INDEX stud_no_pk;
DROP INDEX stud_no_pk
           *
1행에 오류:
ORA-02429: 유일/일차 키 적용을 위한 인덱스를 삭제할 수 없습니다
```

idx_stud_no_name 인덱스 삭제

기본 키, 고유 키 제약조건 생성시 자동적으로 생성된 인덱스는 삭제할 수 없다.

인덱스 재구성

- **인덱스 재구성**
  - 인덱스가 생성된 칼럼의 데이터가 자주 변경되는 경우 인덱스의 재구성 필요
  - 불필요하게 생성된 인덱스 내부 노드를 정리하는 작업
    - B*-트리의 인덱스 키는 일정한 정렬 순서를 유지해야 하므로 새로운 값이 입력되거나 수정, 삭제될 경우, 인덱스 키의 정렬 순서를 유지하기 위해 노드를 조정하는 과정이 필요
    - 테이블 칼럼의 데이터가 입력, 수정, 삭제됨에 따라 B*-트리가 사향트리화되는 것을 막기 위해 내부 노드의 정리 작업이 필요

사용법

```
ALTER INDEX [ schema.] index REBUILD
[ TABLESPACE tablespace]
```

사용예

학생 테이블에 생성된 stud_no_pk 인덱스를 재구성하여라.

```
SQL> ALTER INDEX stud_no_pk REBUILD;

인덱스가 변경되었습니다.
```