

초보자를 위한
Oracle SQL Database



인덱스 및 뷰



한국기술교육대학교
온라인평생교육원

학습내용

- 인덱스
- 뷰

학습목표

- **인덱스**를 활용하여 원하는 데이터를 검색할 수 있다.
- **뷰**를 지정할 수 있다.

인덱스

인덱스 및 뷰

인덱스

인덱스(Index)란?

인덱스(Index)란?

- 인덱스는 데이터베이스 테이블에 있는 데이터의 검색 속도 향상을 위해서 사용
- 인덱스는 테이블에 있는 하나 이상의 컬럼으로 만들 수 있음
- 가장 일반적인 B-tree 인덱스는 인덱스 키와 이 키에 해당하는 테이블의 행이 저장된 주소 값으로 구성됨
- 간단하게 인덱스는 책의 맨 뒤에 있는 “찾아보기”와 같음
- 책에서 어떤 내용을 빨리 찾기 위해 찾아보기를 참고하여 해당 페이지를 보듯이 인덱스를 만들어 놓으면 데이터가 저장되어 있는 물리적 주소를 빨리 찾을 수 있음

인덱스 및 뷰

인덱스

인덱스의 장/단점

+ 인덱스의 장점

테이블을 생성하여 사용하다 보면
데이터를 입력과 삭제를 반복하게 됨

시간이 지나면 데이터들이 뒤섞이면서
WHERE 절의 조건에 맞는 데이터를 찾기 위해서는
테이블의 처음부터 끝까지 읽어서 찾는 FULL SCAN을 해야 함

데이터의 양이 많으면 많을 수록 검색하는데
걸리는 시간은 길어지게 되어 효율이 떨어짐

이때 테이블에 인덱스를 생성하여 사용하면
데이터를 빨리 찾을 수 있음

인덱스

인덱스 및 뷰

인덱스

인덱스의 장/단점

인덱스의 장점

인덱스의 장점

- 특히, ORDER BY에 의한 SORT에서 그 성능이 탁월함
- ORDER BY는 부하가 굉장히 많이 걸리는 작업
 - 정렬을 하기 위해 1차적으로 메모리에서 정렬이 이루어지고 메모리보다 큰 데이터들은 디스크에 읽기 쓰기를 하며 작업을하게 됨
- 인덱스를 사용하면 이러한 작업 없이 인덱스를 참조하여 데이터를 가져오기 때문에 성능에서 차이가 크게 발생

인덱스 및 뷰

인덱스

인덱스의 장/단점

인덱스의 장점

인덱스를 사용해야 할 경우

- 행의 개수가 많을 때
- WHERE 절에서 자주 조회되는 컬럼
- ORDER BY 절에서 자주 사용되는 컬럼
- 조인(join) 시 자주 사용되는 컬럼
- 해당 컬럼이 null을 포함하는 경우
(인덱스에는 null 값이 들어갈 수 없음)

인덱스

인덱스 및 뷰

인덱스

인덱스의 장/단점

인덱스의 단점

인덱스의 단점

- 인덱스를 유지하기 위하여 데이터의 INSERT, UPDATE, DELETE가 발생하면 그에 따른 인덱스를 다시 계산해야 함
- 데이터의 수정이 많이 발생하는 테이블의 경우에는 인덱스를 계산하는 시간이 발생하기 때문에 오히려 효율이 더 떨어질 수 있음
- 검색 시 검색 결과에 해당하는 데이터가 테이블의 전체 데이터 중에서 10~15% 이하인 경우에만 효율적이고 그 이상의 데이터를 처리할 경우에는 인덱스를 사용하지 않는 것이 더 좋음
- 인덱스를 관리하기 위해 저장공간이 추가로 필요

인덱스 및 뷰

인덱스

인덱스의 장/단점

인덱스의 단점

인덱스를 사용하지 말아야 하는 경우

- 행의 개수가 적을 때
- 테이블의 검색은 적고, 삽입, 수정, 삭제가 빈번하게 발생하는 경우

인덱스

인덱스 및 뷰

인덱스

인덱스 관리

+ INDEX 생성

INDEX 생성

```
CREATE [UNIQUE] INDEX 인덱스명
ON 테이블명 (컬럼1 ASC | DESC, 컬럼2, 컬럼3...)
```

- 지정한 컬럼에 중복 데이터가 입력되지 않는 경우 : 일련번호나 주민등록 번호와 같이 중복 값이 없을 때

UNIQUE INDEX는 지정한 컬럼에 중복 데이터가 입력되지 않는 경우에 지정

인덱스 및 뷰

인덱스

인덱스 관리

+ INDEX 생성

워크시트 질의 작성기

```
CREATE INDEX idx_학생_성명
ON 학생(성명);
```

스크립트 출력 x

작업이 완료되었습니다.(0.032초)

Index IDX_학생_성명이 (가) 생성되었습니다.

CREATE INDEX idx_학생_성명
ON 학생(성명);

UNIQUE INDEX 실행의 예

인덱스

인덱스 및 뷰

인덱스

인덱스 관리

+ INDEX 생성

| INDEX_OWNER | INDEX_NAME | UNIQUENESS | STATUS | INDEX_TYPE | TYPE | PARTITIONED | FUNCTIONAL | ... COLUMNS |
|-------------|------------|------------|--------|------------|------|-------------|------------|-------------|
| 1 STUDY01 | 학생_PK | UNIQUE | VALID | NORMAL | N | NO | (null) | NO 학번 |
| 2 STUDY01 | IDX_학생_성명 | NONUNIQUE | VALID | NORMAL | N | NO | (null) | NO 성명 |

“학생_PK” 인덱스를 보면 UNIQUE로 되어 있어
학번 컬럼은 중복 데이터를 허용하지 않음

“IDX_학생_성명” 인덱스는 NONUNIQUE로
성명은 중복 데이터를 허용하도록 지정

인덱스 생성 후 학생 테이블의 인덱스 탭 살펴보기

인덱스 및 뷰

인덱스

인덱스 관리

+ INDEX 삭제

DROP INDEX 인덱스명
ON 테이블명 (컬럼1 ASC | DESC, 컬럼2, 컬럼3...)

```
워크시트 질의 작성기
DROP INDEX idx_학생_성명;

스크립트 출력 x
작업이 완료되었습니다.(0.254초)

Index · IDX_학생_성명이 (가) 삭제되었습니다.
```

INDEX 삭제 실행의 예

인덱스

인덱스 및 뷰

인덱스

인덱스 관리

+ INDEX 삭제

| INDEX_OWNER | INDEX_NAME | UNIQUENESS | STATUS | INDEX_TYPE | P_TYPE | FUNCTION_NAME | COLUMNS |
|-------------|------------|------------|--------|------------|--------|---------------|---------|
| 1 STUDY01 | 학생_PK | UNIQUE | VALID | NORMAL | N | (null) | NO 학번 |

새로 생성되었던 “IDX_학생_성명” 인덱스가 삭제됨

인덱스를 삭제하고 학생 테이블의 인덱스 탭 살펴보기

인덱스 및 뷰

인덱스

인덱스 관리

+ INDEX가 지정되었는지 확인하는 SQL 문

| INDEX_NAME | INDEX_TYPE | TABLE_OWNER | TABLE_NAME | TABLE_TYPE | UNIQUENESS | COMPRESSION | PREFIX |
|------------|------------|-------------|------------|------------|------------|-------------|--------|
| 1 학생_PK | NORMAL | STUDY01 | 학생 | TABLE | UNIQUE | DISABLED | |

SELECT * FROM USER_INDEXES WHERE TABLE_NAME = 테이블명;

예)

SELECT * FROM USER_INDEXES WHERE TABLE_NAME = '학생';

INDEX가 지정되었는지 확인하는 SQL 문과 예시

인덱스

인덱스 및 뷰

인덱스

인덱스 관리

+ INDEX REBUILD

INDEX REBUILD

- 인덱스를 생성하여 사용하다 보면 검색 속도가 떨어지는 것을 느낄 수 있음
 - 이때는 인덱스를 리빌드해야 함
- 인덱스는 트리구조로 되어 있는데, 데이터를 INSERT, DELETE, UPDATE를 계속하다 보면 트리의 한쪽이 무거워져 전체적으로 트리의 깊이가 깊어지게 됨
- 이러한 현상으로 인해 인덱스의 검색속도가 떨어져 주기적으로 인덱스를 리빌드하는 것이 좋음

인덱스 및 뷰

인덱스

인덱스 관리

+ INDEX REBUILD

```
워크시트 질의 작성기
SELECT UR.TABLESPACE_NAME, UR.TABLE_NAME, UR.INDEX_NAME, UR.BLEVEL,
       DECODE(SIGN(NVL(UR.BLEVEL, 99)-3), 1, 'Check',
              DECODE(NVL(UR.BLEVEL, 99), 99, '?', 'Rebuild'), 'Check') CNF
  FROM USER_INDEXES UR
 WHERE UR.BLEVEL > 4;
```

```
SELECT UR.TABLESPACE_NAME, UR.TABLE_NAME,
       UR.INDEX_NAME, UR.BLEVEL,
       DECODE(SIGN(NVL(UR.BLEVEL, 99)-3), 1,
              DECODE(NVL(UR.BLEVEL, 99), 99, '?', 'Rebuild'), 'Check') CNF
  FROM USER_INDEXES UR
 WHERE UR.BLEVEL > 4;
```

리빌드가 필요한 인덱스 조회
다음은 인덱스 레벨이 4 이상인 것을 조회

인덱스

인덱스 및 뷰

인덱스

인덱스 관리

+ INDEX REBUILD

워크시트 질의 작성기

```
ALTER INDEX 학생_PK REBUILD;
```

스크립트 출력 x 질의 결과 x
작업이 완료되었습니다.(0.038초)

Index 학생_PK이 (가) 변경되었습니다.

ALTER INDEX 인덱스명 REBUILD;
예)
ALTER INDEX 학생_PK REBUILD;

INDEX REBUILD 문

인덱스 및 뷰

인덱스

인덱스 관리

+ INDEX REBUILD

리빌드를 하나씩 하지 않고 전체 인덱스를 리빌드 하려면,
다음과 같이 USER_INDEXES에 있는 인덱스를 조회하여
인덱스 리빌드 쿼리를 만들 수 있음

워크시트 질의 작성기

```
SELECT 'ALTER INDEX'||INDEX_NAME||' REBUILD;' FROM USER_INDEXES;
```

스크립트 출력 x 질의 결과 x
작업이 완료되었습니다.(0.010초)

1 ALTER INDEX 전공_PK REBUILD;
2 ALTER INDEX 학부_PK REBUILD;
3 ALTER INDEX 교과목_PK REBUILD;
4 ALTER INDEX 교수_PK REBUILD;
5 ALTER INDEX 학생_PK REBUILD;

SELECT 'ALTER INDEX'||INDEX_NAME||' REBUILD;' FROM USER_INDEXES;

전체 INDEX REBUILD 문

인덱스

인덱스 및 뷰

인덱스

인덱스 관리

+ ROWID 컬럼

| 워크시트 | | | |
|------------------------------------------|------------|-----|--|
| 질의 작성기 | | | |
| <pre>SELECT ROWID, 학번, 성명 FROM 학생;</pre> | | | |
| 스크립트 출력 x 질의 결과 x | | | |
| ROWID | 학번 | 성명 | |
| 1 AAATnqAASAAAAAVrAAA | 2020161106 | 이준희 | |
| 2 AAATnqAASAAAAAVrAAB | 2020161137 | 한황수 | |
| 3 AAATnqAASAAAAAVrAAC | 2020172008 | 김성빈 | |
| 4 AAATnqAASAAAAAVrAAD | 2020172012 | 김진만 | |
| 5 AAATnqAASAAAAAVrAAE | 2020172026 | 백성준 | |
| 6 AAATnqAASAAAAAVrAAF | 2020172028 | 신승석 | |
| 7 AAATnqAASAAAAAVrAAG | 2020172035 | 이재성 | |
| 8 AAATnqAASAAAAAVrAAH | 2020174011 | 김민석 | |
| 9 AAATnqAASAAAAAVrAAI | 2020174048 | 양진규 | |
| 10 AAATnqAASAAAAAVrAAJ | 2020174050 | 엄보나 | |
| 11 AAATnqAASAAAAAVrAAK | 2020174053 | 유승환 | |
| 12 AAATnqAASAAAAAVrAAK | 2020174092 | 한상구 | |
| 13 AAATnqAASAAAAAVrAAM | 2020180015 | 김선강 | |

ROWID를 확인하기 위해서는
SELECT ROWID, 컬럼1, 컬럼2... FROM 테이블명
WHERE 조건;
와 같이 컬럼 이름 부분에 ROWID를 넣어주면 됨

모든 테이블에는 인덱스 지정과 관계 없이 ROWID라는 컬럼이 있음
Oracle에서 ROWID는 “데이터 고유주소”를 의미

인덱스 및 뷰

인덱스

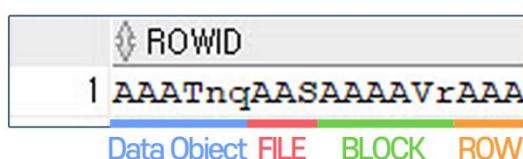
인덱스 관리

+ ROWID 컬럼

ROWID 컬럼의 구성

- ROWID = Data Object 번호 + FILE 번호 + BLOCK 번호 + ROW번호

18자리 6자리 3자리 6자리 3자리



Data Object : 데이터 오브젝트의 고유 번호

FILE : 데이터 파일에 할당되는 번호

BLOCK : 데이터 블록의 위치 번호

ROW : 블록 내의 행 고유번호

인덱스



인덱스 및 뷰

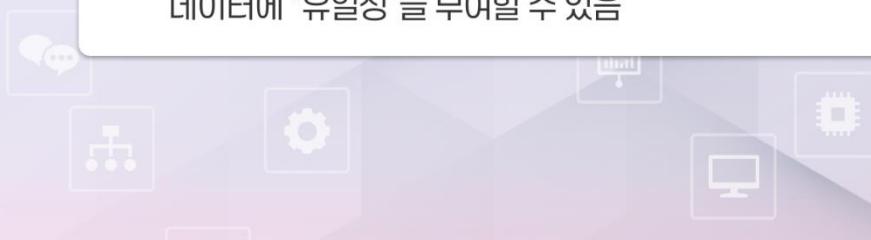
인덱스

인덱스 관리

+ ROWID 컬럼

ROWID의 특징

- ROWID는 테이블의 한 컬럼처럼 참조만 가능하며 데이터베이스에 값이 저장되지는 않음
- ROWID는 물리적인 ADDRESS를 가지고 있기 때문에 데이터에 접근하는 가장 빠른 조회 경로를 갖고 있음
- 데이터가 non-unique한 환경일 때, 데이터에 "유일성"을 부여할 수 있음



인덱스 및 뷰

인덱스

인덱스 관리

+ ROWID 컬럼

인덱스(Index) 추가 시 주의 사항

- 오라클 서버를 운영하다 보면 데이터의 누적으로 크기가 커지게 되면 속도가 느려지게 되는 일이 발생
 - 이때 데이터베이스 관리자가 생각하는 것이 인덱스를 추가로 생성하여 문제를 해결하려고 함
 - 이것이 좋은 해결책이 될 수 있지만, 나중에 다시 속도가 느려지게 되는 문제가 발생할 때마다 인덱스를 계속 추가할 수는 없음
- 하나의 테이블에 4, 5개 이상의 인덱스를 사용하는 것은 별로 좋은 방법은 아님
- 속도가 느려지는 문제가 발생할 때마다 인덱스를 생성하는 것은 하나의 쿼리문을 빠르게는 만들 수 있지만 데이터베이스의 전체적인 성능을 높이지는 못함
- 오히려 인덱스를 계산하는 시간이 더 늘어나 전체적인 성능을 떨어뜨림





❖ 뷰(VIEW)란?

뷰란

- 테이블 또는 테이블들에서 필요한 컬럼들을 모아 **가상의 테이블**을 만들 수 있음
- 뷰는 테이블처럼 **직접 데이터를 갖고 있지 않고** 뷰에 접근(검색)시 실행되어 데이터를 가져오는 방식
- 뷰는 테이블 뿐만 아니라 다른 뷰를 참조해서 새로운 뷰를 생성하여 사용할 수 있음



❖ 뷰의 사용 목적

뷰 사용 목적

- 여러 테이블의 데이터를 조인해서 사용할 때
- 복잡한 질의를 미리 만들어 놓거나 테이블의 조인이나 그룹핑 같은 복잡한 쿼리를 뷰로 만들어 놓고 필요할 때 사용
- 자주 사용해야하는 쿼리문을 뷰로 만들어 놓으면 함수를 사용하듯이 필요할 때 편리하게 사용



◆ 뷰의 사용 목적

뷰 사용 목적

- 데이터 보안 기능: 여러 테이블의 모든 컬럼의 데이터가 아닌 필요한 컬럼만 참조하기 때문에 불필요한 데이터를 볼 수 없게 함
- 예) 급여 담당자가 인사테이블에서 직책과 호봉 정보를 가져 와서 급여를 계산해야 하기 때문에 인사 테이블을 조회할 때
 - 급여 계산에 필요한 정보만 보는 것이 아니라 테이블에 들어 있는 개인 신상 정보도 같이 보게 된다면 문제가 발생
 - 업무에 필요한 최소한의 정보만 제공 받도록 뷰를 사용하면 이러한 문제를 해결하여 데이터 보안을 구현

◆ 뷰의 장/단점

뷰의 장점

- 데이터 보안
 - 보안 등급에 맞추어 테이블의 컬럼 및 범위를 지정
 - 연산 결과만 제공하기 때문에 데이터 원본을 숨길 수 있음
다음은 급여를 15% 인상한 결과를 보여 줌

```
CREATE VIEW v_table(v_no, v_fname, v_salary)
AS SELECT EMPLOYEE_ID, FIRST_NAME, (salary +
NVL(COMMISSION_PCT,0))* 1.15 FROM EMPLOYEES;
```
 - 뷰를 읽기 전용으로 지정하여 데이터의 INSERT나 UPDATE를 쉽게 막을 수 있음

```
CREATE VIEW v_READONLY_table(v_no, v_fname, v_salary)
AS SELECT EMPLOYEE_ID, FIRST_NAME, SALARY
FROM EMPLOYEES WITH READ ONLY;
```

뷰



인덱스 및 뷰

뷰

◆ 뷰의 장/단점

뷰의 장점

- 사용 편의
 - 검색 조건의 단순화
 - 조인 문장의 단순화



인덱스 및 뷰

뷰

◆ 뷰의 장/단점

뷰의 단점

- 독립적인 인덱스 사용 불가능
(테이블처럼 직접 데이터를 갖고 있지 않기 때문에 인덱스를 지정할 수 없음)
- 정의된 뷰는 변경 불가능



부

인덱스 및 뷰

부

뷰 권한

+ 뷰 생성 권한

```

Applications Places Terminal
Sat 21:42 • ④ ⑤ ⑥
Terminal
File Edit View Search Terminal Help
SQLcl: Release 19.1 Production on Sat Jan 15 21:40:26 2022
Copyright (c) 1982, 2022, Oracle. All rights reserved.
Username? ('system')
Password? ('*****')
Last Successful login time: Jan 15 2022 21:40:41 +09:00
Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

grant create view to study01;
Grant succeeded.
SQL> 
```

SQLcl 사용
cmd> sqlplus / as sysdba;
SQL> grant create view to 사용자;

뷰 생성 권한이 없는 사용자가 뷰를 생성하기 위해서는 권한이 필요
데이터베이스 관리자에게 필요한 권한을 요청하여 받아야 함

인덱스 및 뷰

부

뷰 권한

+ 뷰 생성 권한

시작 페이지 × 19c sys × 19c remote ×

워크시트 질의 작성기

```
GRANT CREATE VIEW TO STUDY01;
```

스크립트 출력 ×

작업이 완료되었습니다.(0.052초)

Grant을 (를) 성공했습니다.

SQL Developer 사용
Developer에서 Oracle 관리자로 접속을 한 후 다음과 같이 입력

부

인덱스 및 뷰

부

◆ 뷰 권한

+ 뷰 생성

뷰를 생성하는 SQL 문

CREATE [OR REPLACE] [FORCE | NOFORCE] **VIEW** 뷰이름
AS [WITH CHECK OPTION [constraint 제약조건]]
[WITH READ ONLY]

[] : 대괄호는 생략 가능, | : 여러 옵션 중 한 개 선택.

OR REPLACE : 이미 존재하는 뷰의 내용을 수정함

FORCE : 기본 테이블의 존재 여부 상관없이 View 생성

NOFORCE : 기본 테이블이 존재할 경우에만 View 생성(기본값)

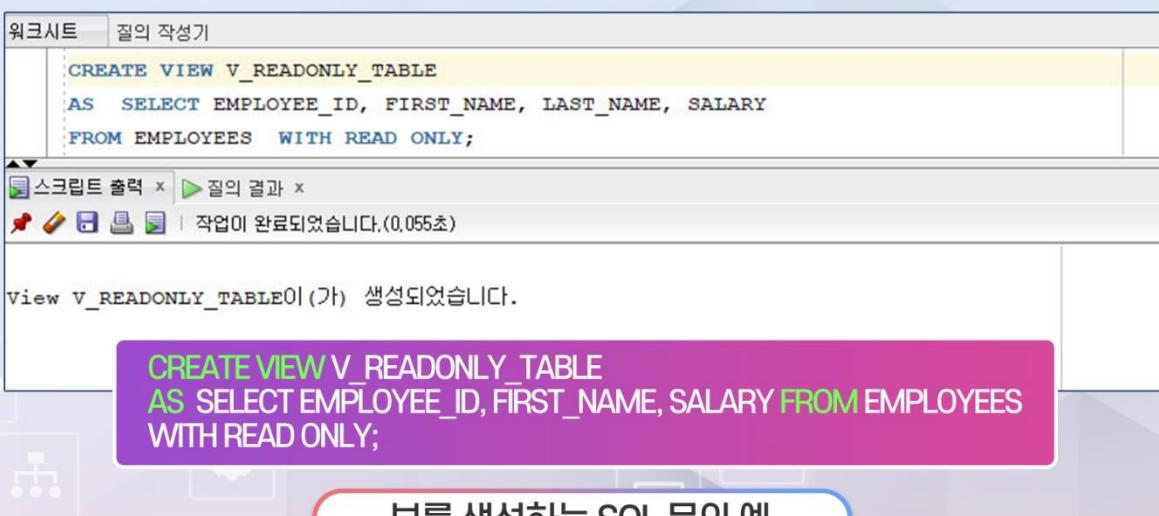
WITH CHECK OPTION : 주어진 제약조건에 맞는 데이터만 입력 및 수정 가능

인덱스 및 뷰

부

◆ 뷰 권한

+ 뷰 생성



워크시트 질의 작성기

```
CREATE VIEW V_READONLY_TABLE
AS SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY
FROM EMPLOYEES WITH READ ONLY;
```

스크립트 출력 x 질의 결과 x

작업이 완료되었습니다.(0.055초)

View V_READONLY_TABLE01 (가) 생성되었습니다.

**CREATE VIEW V_READONLY_TABLE
AS SELECT EMPLOYEE_ID, FIRST_NAME, SALARY FROM EMPLOYEES
WITH READ ONLY;**

뷰를 생성하는 SQL 문의 예

부

인덱스 및 뷰

부

▶ 뷰 권한

+ 뷰 수정하기

CREATE OR REPLACE

```
CREATE OR REPLACE [FORCE | NOFORCE] VIEW 뷰이름
AS [WITH CHECK OPTION [constraint 제약조건] ]
[WITH READ ONLY]
```



인덱스 및 뷰

부

▶ 뷰 권한

+ 뷰 수정하기

워크시트 질의 작성기

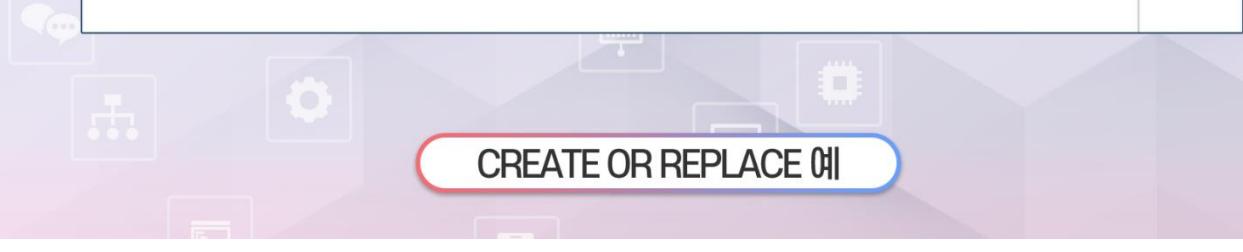
```
CREATE VIEW V_READONLY_TABLE
AS SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY
FROM EMPLOYEES WITH READ ONLY;
```

스크립트 출력 x ▶ 질의 결과 x
작업이 완료되었습니다.

**CREATE OR REPLACE VIEW V_READONLY_TABLE
AS SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY
FROM EMPLOYEES WITH READ ONLY;**

View V_READONLY_TABLE0 (1) 등록되었습니다.

CREATE OR REPLACE 예



부

인덱스 및 뷰

부

◆ 뷰 권한

+ 뷰 삭제하기

워크시트 질의 작성기

```
DROP VIEW V_READONLY_TABLE;
```

스크립트 출력 x 질의 결과 x
작업이 완료되었습니다.(0.05초)

View V_READONLY_TABLE0(가) 삭제되었습니다.

DROP VIEW 뷰 이름

인덱스 및 뷰

부

◆ 뷰 권한

+ 뷰 테이블 목록 확인

워크시트 질의 작성기

```
SELECT * FROM tab WHERE TABTYPE = 'VIEW';
```

스크립트 출력 x 질의 결과 x
SQL | 인출된 모든 행: 3(0.012초)

| TNAME | TABTYPE | CLUSTERID |
|------------------|---------|-----------|
| TESTVIEW | VIEW | (null) |
| V_READONLY_TABLE | VIEW | (null) |
| V_TABLE | VIEW | (null) |

SELECT * FROM tab WHERE TABTYPE = 'VIEW';

◆ 뷰의 종류

+ 단순 뷰

단순 뷰

- 하나의 테이블로 만들어진 뷰
- DML(INSERT, UPDATE, DELETE) 사용 가능
- 그룹함수 사용 불가, 별칭을 지정하면 사용 가능
- DISTINCT 사용 불가
단순 VIEW는 DML 사용 목적이기 때문에
SELECT 문의 DISTINCT, GROUP BY는 사용할 수 없음

◆ 뷰의 종류

+ 단순 뷰

단순 뷰의 예

```
CREATE VIEW V_READONLY_TABLE
AS SELECT EMPLOYEE_ID, FIRST_NAME, SALARY
FROM EMPLOYEES
WITH READ ONLY;
```



인덱스 및 뷰

뷰

◆ 뷰의 종류

+ 복합 뷰

복합 뷰

- 복수의 테이블로 만들어진 뷰
- DML 사용 불가
 - 테이블이 하나가 아니기 때문에 입력 테이블 확정 불가
 - JOIN, SELECT 사용 목적이기 때문에 DML 사용 불가
- 그룹함수 사용 가능
- DISTINCT 사용 가능

인덱스 및 뷰

뷰

◆ 뷰의 종류

+ 복합 뷰

복합 뷰의 예

```
CREATE VIEW TESTVIEW
```

```
AS SELECT 학번, 성명, 전공명, 학년, 성별 FROM 학생
```

```
INNERJOIN 전공
```

```
ON 학생.전공코드 = 전공.전공코드;
```



◆ 뷰 데이터 사전(VIEW DATA DICTIONARY)

뷰 데이터 사전(VIEW DATA DICTIONARY)

- 뷰 데이터 사전에는 뷰에 대한 데이터베이스 전반에 대한 정보를 담고 있음
- USER_VIEWS 테이블 : 뷰에 대한 정보가 담겨 있는 Oracle 테이블

◆ 뷰 데이터 사전(VIEW DATA DICTIONARY)

+ USER_VIEWS의 구조

| 질의작성기 | |
|-----------------------------------------|------------------------|
| desc USER_VIEWS; | |
| 스코프트 콜백 : > 파일 경로 : 작업자 환경변수(D:\0.0255) | |
| 이름 | 날짜 |
| VIEW_NAME | NOT NULL VARCHAR2(128) |
| TEXT_LENGTH | NUMBER |
| TEXT | LONG |
| TEXT_VC | VARCHAR2(4000) |
| TYPE_TEXT_LENGTH | NUMBER |
| TYPE_TEXT | VARCHAR2(4000) |
| OID_TEXT_LENGTH | NUMBER |
| OID_TEXT | VARCHAR2(4000) |
| VIEW_TYPE_ORDER | VARCHAR2(128) |
| VIEW_TYPE | VARCHAR2(128) |
| SUPERVIEW_NAME | VARCHAR2(128) |
| EDITIONING_VIEW | VARCHAR2(1) |
| REF_ID | VARCHAR2(1) |
| CONSTRAINTS_DATA | VARCHAR2(1) |
| RECREATES | VARCHAR2(12) |
| ORIGIN_CON_ID | NUMBER |
| DEFAULT_COLLATION | VARCHAR2(100) |
| CONTAINERS_DEFAULT | VARCHAR2(3) |
| CONTAINER_MAP | VARCHAR2(3) |
| EXTENDED_DATA_LINK | VARCHAR2(3) |
| EXTENDED_DATA_LINK_MAP | VARCHAR2(3) |
| HAS_SENSITIVE_COLUMN | VARCHAR2(3) |
| ADMITS_NULL | VARCHAR2(3) |
| PER_LOCAL_ONLY | VARCHAR2(3) |

desc USER_VIEWS;

DESCRIBE 명령으로 테이블의 구조를 보듯이
USER_VIEWS 테이블의 구조도 desc 명령으로 볼 수 있음

부

인덱스 및 뷰

부

◆ 뷰 데이터 사전(VIEW DATA DICTIONARY)

+ 생성된 VIEW 상세 정보 확인

```

    워크시트 | 풀의 작성기
    SELECT VIEW_NAME, TEXT_LENGTH, TEXT, READ_ONLY FROM USER_VIEWS;
    스크립트 출력 x | 질의 결과 x
    SQL | 인출된 모든 행: 3(0.018초)
    1 V_TABLE          TEXT_LENGTH TEXT
    2 TESTVIEW         129 select 학번, 성명, 전공명, 학년, 성별 from 학생 inner join 전공 on 학생.전공코드 = 전공.전공코드
    3 V_READONLY_TABLE 68 SELECT EMPLOYEE_ID, FIRST_NAME, SALARY FROM EMPLOYEES WITH READ ONLY
    READ_ONLY
  
```

컬럼 정보

- VIEW_NAME 컬럼 : 뷰 이름
- TEXT_LENGTH : 뷰의 길이
- TEXT : 생성된 뷰의 SQL 문
- READ_ONLY : 읽기 전용 여부

`SELECT VIEW_NAME, TEXT_LENGTH, TEXT, READ_ONLY
FROM USER_VIEWS;`

사용자가 생성한 뷰의 상세 정보를 확인하기 위해서는 위 SQL 문을 실행

인덱스 및 뷰

부

◆ 뷰 실습

+ 단순 뷰로 급여를 15% 인상한 결과를 보여주는 뷰

| COLUMN_NAME | DATA_TYPE | NULLABLE | DATA_DEFAULT | COLUMN_ID | COMMENTS | INSERTABLE | UPDATABLE | DELETABLE |
|-------------|---------------|----------|--------------|-----------|----------|------------|-----------|-----------|
| 1 v_fname | VARCHAR2 (20) | Yes | (null) | 2 (null) | | YES | YES | YES |
| 2 v_salary | NUMBER | Yes | (null) | 3 (null) | | NO | NO | NO |
| 3 v_no | NUMBER (6) | Yes | (null) | 1 (null) | | YES | YES | YES |

`CREATE VIEW v_table(v_no, v_fname, v_salary)
AS SELECT EMPLOYEE_ID, FIRST_NAME, (salary +
NVL(COMMISSION_PCT,0))* 1.15 FROM EMPLOYEES;`

열 탭을 선택하여 생성된 v_table을 보면
컬럼별 데이터의 INSERT, UPDATE, DELETE 여부가 표시되어 있음

부

인덱스 및 뷰

부

▶ 뷰 실습

- + 단순 뷰로 급여를 15% 인상한 결과를 보여주는 뷰

| V_NO | V_FNAME | V_SALARY |
|------|---------------|----------|
| 1 | 100 Steven | 27600 |
| 2 | 101 Neena | 19550 |
| 3 | 102 Lex | 19550 |
| 4 | 103 Alexander | 10350 |
| 5 | 104 Bruce | 6900 |
| 6 | 105 David | 5520 |
| 7 | 106 Valli | 5520 |
| 8 | 107 Diana | 4830 |
| 9 | 108 Nancy | 13800 |
| 10 | 109 Daniel | 10350 |

데이터 탭을 선택하여 v_table의 데이터를 보면
급여가 15% 인상된 것을 볼 수 있음

인덱스 및 뷰

부

▶ 뷰 실습

- + 복합 뷰로 학생 테이블과 전공 테이블을 조인하여 전공명을 조회하는 뷰

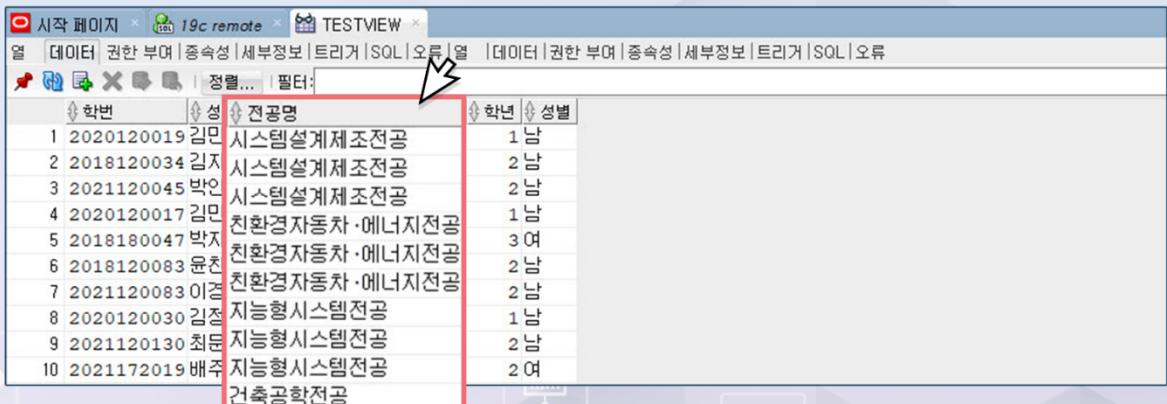
| COLUMN_NAME | DATA_TYPE | NULLABLE | DATA_DEFAULT | COLUMN_ID | COMMENTS | INSERTABLE | UPDATABLE | DELETABLE |
|-------------|----------------|----------|--------------|-----------|----------|------------|-----------|-----------|
| 1 학번 | NUMBER (10) | No | (null) | 1 (null) | YES | YES | YES | |
| 2 성명 | NVARCHAR2 (20) | Yes | (null) | 2 (null) | YES | YES | YES | |
| 3 전공명 | NVARCHAR2 (25) | Yes | (null) | 3 (null) | NO | NO | NO | |
| 4 학년 | NUMBER (1) | Yes | (null) | 4 (null) | YES | YES | YES | |
| 5 성별 | NVARCHAR2 (1) | Yes | (null) | 5 (null) | YES | YES | YES | |

```
CREATE VIEW TESTVIEW
AS SELECT 학번, 성명, 전공명, 학년, 성별 FROM 학생
INNER JOIN 전공
ON 학생.전공코드 = 전공.전공코드;
```

열 탭을 선택하여 생성된 testview를 보면
컬럼별 데이터의 INSERT, UPDATE, DELETE 여부가 표시되어 있음

▶ 뷰 실습

✚ 복합 뷰로 학생 테이블과 전공 테이블을 조인하여 전공명을 조회하는 뷰



| 학번 | 성 | 전공명 | 학년 | 성별 |
|--------|------------|-----|--------------|-----|
| 1 | 2020120019 | 김민 | 시스템설계제조전공 | 1 남 |
| 2 | 2018120034 | 김자 | 시스템설계제조전공 | 2 남 |
| 3 | 2021120045 | 박인 | 시스템설계제조전공 | 2 남 |
| 4 | 2020120017 | 김민 | 친환경자동차·에너지전공 | 1 남 |
| 5 | 2018180047 | 박자 | 친환경자동차·에너지전공 | 3 여 |
| 6 | 2018120083 | 윤천 | 친환경자동차·에너지전공 | 2 남 |
| 7 | 2021120083 | 이경 | 친환경자동차·에너지전공 | 2 남 |
| 8 | 2020120030 | 김정 | 지능형시스템전공 | 1 남 |
| 9 | 2021120130 | 최문 | 지능형시스템전공 | 2 남 |
| 10 | 2021172019 | 배주 | 지능형시스템전공 | 2 여 |
| 건축공학전공 | | | | |

데이터 탭을 선택하여 testview의 데이터를 보면
전공명이 조회되어 나타난 것을 볼 수 있음

실습하기

인덱스 및 뷰

실습하기

- 제공한 “14회차_SAMPLE_TABLE.sql” 파일로 테이블을 생성하고 다음과 같이 이데어로 새서해 주
 - STATIC_STRING_IDX 컬럼을 IDX_ST_SSI로 인덱스를 생성.
 - DYNAMIC_STRING_IDX 컬럼을 IDX_DY_DSI로 인덱스를 생성.
 - INTEGER_NUMBER_IDX 컬럼을 IDX_IN_INI로 인덱스를 생성.
 - FLOAT_NUMBER_IDX 컬럼을 IDX_FL_FNI로 인덱스를 생성.
- 인덱스를 생성한 고정 문자, 가변 문자, 정수, 실수 컬럼에서 각각 자료를 찾을 때, 인덱스 유무에 따라 검색 시간을 비교해 보세요.
 - STATIC_STRING 컬럼에서 'Jor2TEx4ZeiN7KWval6R'를 찾아 보기 바랍니다.
 - STATIC_STRING_IDX 컬럼에서 'Jor2TEx4ZeiN7KWval6R'를 찾아 보기 바랍니다.
 - DYNAMIC_STRING 컬럼에서 'w6tfRdPXfwEyLKhdxuUW'를 찾아 보기 바랍니다.
 - DYNAMIC_STRING_IDX 컬럼에서 'w6tfRdPXfwEyLKhdxuUW'를 찾아 보기 바랍니다.
 - INTEGER_NUMBER 컬럼에서 1047686951을 찾아 보기 바랍니다.
 - INTEGER_NUMBER_IDX 컬럼에서 1047686951을 찾아 보기 바랍니다.
 - FLOAT_NUMBER 컬럼에서 0.988432을 찾아 보기 바랍니다.
 - FLOAT_NUMBER_IDX 컬럼에서 0.988432을 찾아 보기 바랍니다.

+ Oracle Database 19c 버전 사용
+ 최신 버전인 Oracle Database 21c는 현재 cloud에서만 사용 가능하며, 지원 기간은 2년

인덱스 생성하기



실습단계

인덱스 생성하기

다운로드한 “14회차_SAMPLE_TABLE.sql” 파일을 워크시트로 드래그

인덱스는 데이터 입력 완료 후 생성

스크립트 실행 버튼을 클릭 또는 F5키

접속에서 “19c remote” 선택하고 “확인” 버튼

테이블 구조

CHAR형은 고정길이 데이터를 저장

VARCHAR2형은 가변길이 데이터를 저장

NUMBER(x,0)형은 정수 데이터를 저장

NUMBER(x,y)형은 y의 길이 만큼의 실수 데이터를 저장

실습하기

실습단계

인덱스 탭 선택하여 인덱스 확인

주석으로 막힌 인덱스 생성문을 주석 해제

마우스로 드래그하여 주석 해제할 부분을 선택

Ctrl + /, Ctrl + ? 키로 주석 해제 및 설정

커서의 SQL 문을 실행하기 위해 Ctrl + Enter키

“19c remote” 워크시트에 붙여넣기

인덱스가 없는 고정길이 문자 검색 시간 0.053초

인덱스가 적용된 고정길이 문자 검색 시간 0.004초

인덱스가 없는 가변길이 문자 검색 시간 0.043초

인덱스가 적용된 가변길이 문자 검색 시간 0.004초

인덱스가 없는 정수 검색 시간 0.047초

인덱스가 적용된 정수 검색 시간 0.003초

인덱스가 없는 실수 검색 시간 0.046초

인덱스가 적용된 실수 검색 시간 0.004초

실습하기

실습단계

뷰 생성하기

다운로드한 “14회차_DEPARTMENTS.sql” 파일을 워크시트로 드래그

DEPARTMENTS 테이블 생성

“19c remote” 워크시트에서 뷰 생성 권한이 있는지 확인

“19c 관리자”로 접속해서 뷰 생성 권한을 지정

셀프 조인으로 메니저명 검색

LEFT JOIN으로 부서명을 검색

뷰 생성

데이터 보안 뷰 생성하기

읽기 전용으로 지정

뷰 생성

읽기 전용이라 데이터 수정이 안됨