

# OBJECT





**VIEW**



# VIEW

## VIEW

- SELECT 쿼리의 실행 결과를 화면에 저장한 논리적인 가상 테이블
- 테이블과 다르게 실질적으로 데이터를 저장하고 있지 않지만, 사용자는 테이블을 사용하는 것과 동일하게 사용 가능
- 물리적인 실제 테이블과의 링크 개념
- 테이블 복사와의 가장 큰 차이점은 VIEW에서 데이터 수정 시 실제 테이블에 수정데이터 반영(테이블 복사 시에는 원본테이블에 영향 없음)

VIEW를 만들기 위해서는 추가적인 권한이 필요

관리자 계정 접속

```
GRANT CREATE VIEW TO kh;
```



# VIEW

## VIEW 생성 방법

CREATE VIEW 뷰이름 AS SELECT구문

CREATE VIEW EMP\_VIEW AS  
SELECT EMP\_NO, EMP\_NAME, EMAIL, PHONE FROM EMPLOYEE;

EMP_NO	EMP_NAME	EMAIL	PHONE
1 621235-1985634	선동일	sun di@kh.or.kr	01099546325
2 631156-1548654	송송기	song jk@kh.or.kr	01045686656
3 861015-1356452	노홍철	no hc@kh.or.kr	01066656263
4 631010-2653546	송은희	song eh@kh.or.kr	01077607879
5 660508-1342154	유재식	yoo js@kh.or.kr	01099999129
6 770102-1357951	정중하	jung jh@kh.or.kr	01036654875
7 630709-2054321	박나라	pack nr@kh.or.kr	01096935222
8 690402-2040612	하이유	ha iy@kh.or.kr	01036654488
9 870927-1313564	김해솔	kim hs@kh.or.kr	01078634444
10 750206-1325546	심보선	sim bs@kh.or.kr	0113654485
11 650505-2356985	윤해논	youn eh@kh.or.kr	0179964233
12 830807-1121321	전형준	jun hd@kh.or.kr	01044432222
13 780923-2234542	장쯔위	jang zw@kh.or.kr	01066682224
14 621111-1785463	하동운	ha dh@kh.or.kr	01158456632
15 856795-1313513	방명수	bang ms@kh.or.kr	01074127545
16 881130-1050911	대북훈	dae bh@kh.or.kr	01088808584
17 770808-1364897	차태연	cha ty@kh.or.kr	01064643212
18 770808-2665412	전지연	jun jy@kh.or.kr	01033624442
19 870427-2232123	이오리	loo or@kh.or.kr	01022306545
20 660712-1212123	임시환	im sw@kh.or.kr	(null)
21 770823-1113111	이송석	lee js@kh.or.kr	(null)
22 800808-1123341	유하진	yoo hj@kh.or.kr	(null)
23 760918-2854697	이태림	lee tr@kh.or.kr	01033000002

# VIEW

## VIEW 데이터 수정

```
UPDATE EMP_VIEW SET EMP_NAME='하동훈'  
WHERE EMP_NAME='하동운';  
UPDATE EMPLOYEE SET PHONE='01011111111'  
WHERE EMP_NAME='하동훈';  
SELECT * FROM EMPLOYEE WHERE EMP_NAME='하동훈';
```

EMP_ID	EMP_N...	EMP_NO	EMAIL	PHONE	DEPT_CODE	JOB_CODE	SAL_LEVEL	S...
213	하동훈	621111-1785463	ha dh@kh.or.kr	01011111111	(null)	J6	S5	23

```
SELECT * FROM EMP_VIEW WHERE EMP_NAME='하동훈';
```

	EMP_NO	EMP_NAME	EMAIL	PHONE
1	621111-1785463	하동훈	ha dh@kh.or.kr	01011111111

※ 테이블 복사와의 차이점은 DML사용시 원본데이터와 VIEW가 동시에 변경



# VIEW

## VIEW – DML 명령어 조작이 불가능한 경우

1. 뷰 정의에 포함되지 않은 컬럼을 조작하는 경우
2. 뷰에 포함되지 않은 컬럼 중에 베이스가 되는 테이블 컬럼이 NOT NULL 제약조건이 지정된 경우
3. 산술 표현식으로 정의된 경우
4. JOIN을 이용해 여러 테이블을 연결한 경우
5. DISTINCT를 포함한 경우
6. 그룹함수나 GROUP BY 절을 포함한 경우

※ 결국 VIEW의 컬럼과 테이블의 컬럼이 직접적으로 연관된 경우에 사용이 가능



# VIEW

## VIEW – 옵션

1. **CREATE OR REPLACE**
  - 생성한 뷰가 없으면 새로 생성하고, 이미 존재하면 갱신

EX) CREATE OR REPLACE VIEW TEST.....
2. **FORCE/NOFORCE**
  - FORCE 옵션은 기본 테이블이 존재하지 않더라도 뷰를 생성
  - 기본값은 NOFORCE로 지정

EX) CREATE FORCE VIEW VIEW\_TEST  
AS SELECT EMP\_NO, EMP\_NAME FORCE FROM EEE;
3. **WITH CHECK OPTION**
  - 옵션을 설정한 컬럼의 값은 수정 불가
4. **WITH READ ONLY**
  - 뷰에 대해 조회만 가능하고, 삽입, 수정, 삭제 등을 하지 못하게 함

※ CHECK OPTION과 READ ONLY의 차이점은 해당 컬럼인지 뷰 전체인지 차이





# SEQUENCE





# SEQUENCE

## SEQUENCE

- 순차적으로 정수 값을 자동으로 생성하는 객체로, 자동 번호 발생기의 역할

표현식

CREATE SEQUENCE 시퀀스이름

① [START WITH 숫자]

-- 처음 발생시킬 시작 값(기본1)

② [INCREMENT BY 숫자]

-- 다음 값에대한 증가 치(기본1)

③ [MAXVALUE 숫자 | NOMAXVALUE]

-- 최대값 지정( $10^{27}-1$ 까지 가능)

④ [MINVALUE 숫자 | NOMINVALUE]

-- 최소값 지정( $-10^{26}$ 까지 가능)

⑤ [CYCLE | NOCYCLE]

-- 시퀀스 최대값 도달 시 CYCLE은 START WITH값으로 되돌아가고

NOCYCLE은 에러

⑥ [CACHE | NOCACHE] – 메모리상에서 시퀀스값 관리(기본 20)



# SEQUENCE

## SEQUENCE

```
CREATE SEQUENCE SEQ_ID  
START WITH 100      -- 시작 값 100  
INCREMENT BY 1      -- 1씩 증가  
MAXVALUE 1000       -- 최대값 1000  
NOCYCLE             -- 1000 이후에 증가하지 않고 에러 발생  
NOCACHE             -- 캐쉬 사용 안함
```

### 시퀀스 사용법

시퀀스명.CURRVAL : 현재 시퀀스 값 반환

시퀀스명.NEXTVAL : 현재 시퀀스의 다음 값 반환

→ 시퀀스의 첫 시작 값은 없으므로 CURRVAL 사용전에 반드시 NEXTVAL을  
1번 이상 사용해야 함



# SEQUENCE

## SEQUENCE

```
SELECT SEQ_EMP_ID.NEXTVAL FROM DUAL;  
SELECT SEQ_EMP_ID.CURRVAL FROM DUAL;  
SELECT SEQ_EMP_ID.CURRVAL FROM DUAL;  
SELECT SEQ_EMP_ID.NEXTVAL FROM DUAL;  
SELECT SEQ_EMP_ID.CURRVAL FROM DUAL;
```

	NEXTVAL
1	100



	CURRVAL
1	100



	CURRVAL
1	100



	NEXTVAL
1	101



	CURRVAL
1	101

# SEQUENCE

## SEQUENCE

```
CREATE SEQUENCE SEQ_TEST  
START WITH 100  
INCREMENT BY 1  
MAXVALUE 1000  
NOCYCLE  
NOCACHE;
```

```
CREATE TABLE SEQ_TBL(  
    MEM_NO NUMBER PRIMARY KEY,  
    MEM_NAME VARCHAR2(20)  
);
```

```
INSERT INTO SEQ_TBL VALUES(SEQ_TEST.NEXTVAL,'user01');  
INSERT INTO SEQ_TBL VALUES(SEQ_TEST.NEXTVAL,'user02');  
SELECT * FROM SEQ_TBL;
```

	MEM_NO	MEM_NAME
1	100	user01
2	101	user02

# SEQUENCE

## SEQUENCE

- **NEXTVAL, CURRVAL 사용 가능 경우**
  1. 서브쿼리가 아닌 SELECT 문
  2. INSERT 문의 SELECT 절
  3. INSERT 문의 VALUE 절
  4. UPDATE 문의 SET 절
- **NEXTVAL, CURRVAL 사용 불가능한 경우**
  1. VIEW의 SELECT 절
  2. DISTINCT 키워드가 있는 SELECT 절
  3. GROUP BY, HAVING, ORDER BY 절이 있는 SELECT 문
  4. SELECT, DELETE, UPDATE의 서브 쿼리
  5. CREATE TABLE, ALTER TABLE의 DEFAULT 값



# SEQUENCE

## SEQUENCE – 수정/삭제

- 시퀀스 수정 시 **CREATE**에 사용한 옵션 변경 가능  
ALTER SEQUENCE 시퀀스이름  
INCREMENT BY 10  
MAXVALUE 500  
NOCYCLE  
NOCACHE;  
단, START WITH는 변경이 불가능하므로 변경해야 하는 경우 삭제 후 다시 생성 해야 함
- 시퀀스 삭제  
DROP SEQUENCE 시퀀스이름;





**INDEX**



# INDEX

## INDEX

- SQL 명령문의 처리 속도를 향상시키기 위해서 컬럼에 대해 생성하는 오라클 객체로 내부구조는 B\*트리 형식으로 구성

### INDEX 장점

- 검색속도가 빨라지고 시스템에 걸리는 부하를 줄여서 시스템 전체 성능 향상

### INDEX 단점

- 인덱스를 위한 추가 저장공간 필요
- 인덱스 생성 시간 필요
- 데이터 변경작업(INSERT,UPDATE,DELETE)이 자주 일어나는 경우 오히려 성능 저하 발생





# INDEX

## INDEX – 효율적인 사용 예

- 전체 데이터 중 10~15% 이내의 데이터를 검색하는 경우
- 두 개 이상의 컬럼이 WHERE절이나 JOIN 조건으로 자주 사용되는 경우
- 한번 입력된 데이터의 변경이 자주 일어나지 않는 경우
- 한 테이블에 저장된 데이터 용량이 매우 클 경우

## INDEX – 표현식

CREATE INDEX 인덱스이름 ON 테이블명(컬럼명,컬럼명,...);

인덱스 정보 조회

SELECT \* FROM USER\_IND\_COLUMNS;

※ 생성한 적 없는 INDEX들은 PRIMARY KEY, UNIQUE 같은 제약조건 생성 시에 자동으로 생성된 INDEX



# INDEX

## INDEX – 생성

```
CREATE INDEX EMP_IND ON  
EMPLOYEE(EMP_NAME,EMP_NO,HIRE_DATE);  
-- 자주 사용하는 검색일 시 인덱스로 미리 생성
```

```
SELECT * FROM USER_IND_COLUMNS WHERE  
INDEX_NAME='EMP_IND';  
-- 생성된 인덱스 확인
```

```
SELECT EMP_NAME,EMP_NO,HIRE_DATE FROM EMPLOYEE;  
-- F10을 눌러서 사용된 OBJECT_NAME을 보면 생성한 INDEX를 사용한것 확  
인 가능
```

OBJECT_NAME
<u>EMP_IND</u>



# INDEX

## INDEX – 생성

DROP INDEX EMP\_IND;                      -- 인덱스 삭제

SELECT EMP\_NAME,EMP\_NO,HIRE\_DATE FROM EMPLOYEE;  
-- F10을 눌러서 사용된 OBJECT\_NAME을 보면 EMPLOYEE 테이블 전체에  
서 조회한 것 확인 가능

OBJECT_NAME
<u>EMPLOYEE</u>

※ INDEX의 효율성을 확인하려면 많은 정보를 가지고있으며, 여러테이블이 연동된 상태여야 확인이 가능





**SYNONYM**



# SYNONYM

## SYNONYM

- 사용자가 다른 사용자의 객체를 참조할 때 [사용자ID].[테이블명]으로 표기
- 길게 표현되는 것을 동의어(SYNONYM)으로 설정하고 간단하게 사용 가능

### 동의어 종류

1. 비공개 동의어
  - 객체에 대한 접근 권한을 부여 받은 사용자가 정의한 동의어
  - 해당 사용자만 사용할 수 있음
2. 공개 동의어
  - 권한을 주는 사용자(DBA)가 정한 동의어
  - 모든 사용자가 사용할 수 있음

※ 동의어 생성을 위한 권한이 필요함

SYSTEM 계정 → GRANT CREATE SYNONYM TO kh;



# SYNONYM

## SYNONYM

표현식

CREATE SYNONYM 동의어이름 FOR 테이블명;

비공개 동의어 생성(kh 계정)

CREATE SYNONYM EMP FOR EMPLOYEE;

-- EMPLOYEE테이블을 EMP라는 동의어 생성

SELECT \* FROM EMPLOYEE;

SELECT \* FROM EMP;



# SYNONYM

## SYNONYM

표현식

CREATE SYNONYM 동의어이름 FOR 테이블명;

공개 동의어 생성(SYSTEM 계정)

CREATE PUBLIC SYNONYM DEPT FOR KH.DEPARTMENT;

KH계정

SELECT \* FROM DEPT;

-- SYSTEM이 만들었지만 KH계정 사용 가능



# SYNONYM

## SYNONYM

표현식

CREATE SYNONYM 동의어이름 FOR 테이블명;

타계정에서도 SELECT 권한만 있다면 사용 가능

GRANT SELECT ON KH.DEPARTMENT TO test01;

-- SYSTEM계정으로 test01 계정에 조회 권한 부여

SELECT \* FROM KH.DEPARTMENT; -- test01 계정

SELECT \* FROM DEPT; -- test01 계정

REVOKE SELECT ON KH.DEPARTMENT FROM test01;

-- SYSTEM계정으로 권한 회수 후 다시 조회 해볼 것





# SYNONYM

## SYNONYM – 삭제

표현식

공개 동의어 삭제

DROP PUBLIC SYNONYM DEPT; -- SYSTEM 계정

비공개 동의어 삭제

DROP SYNONYM EMP; -- kh계정

