

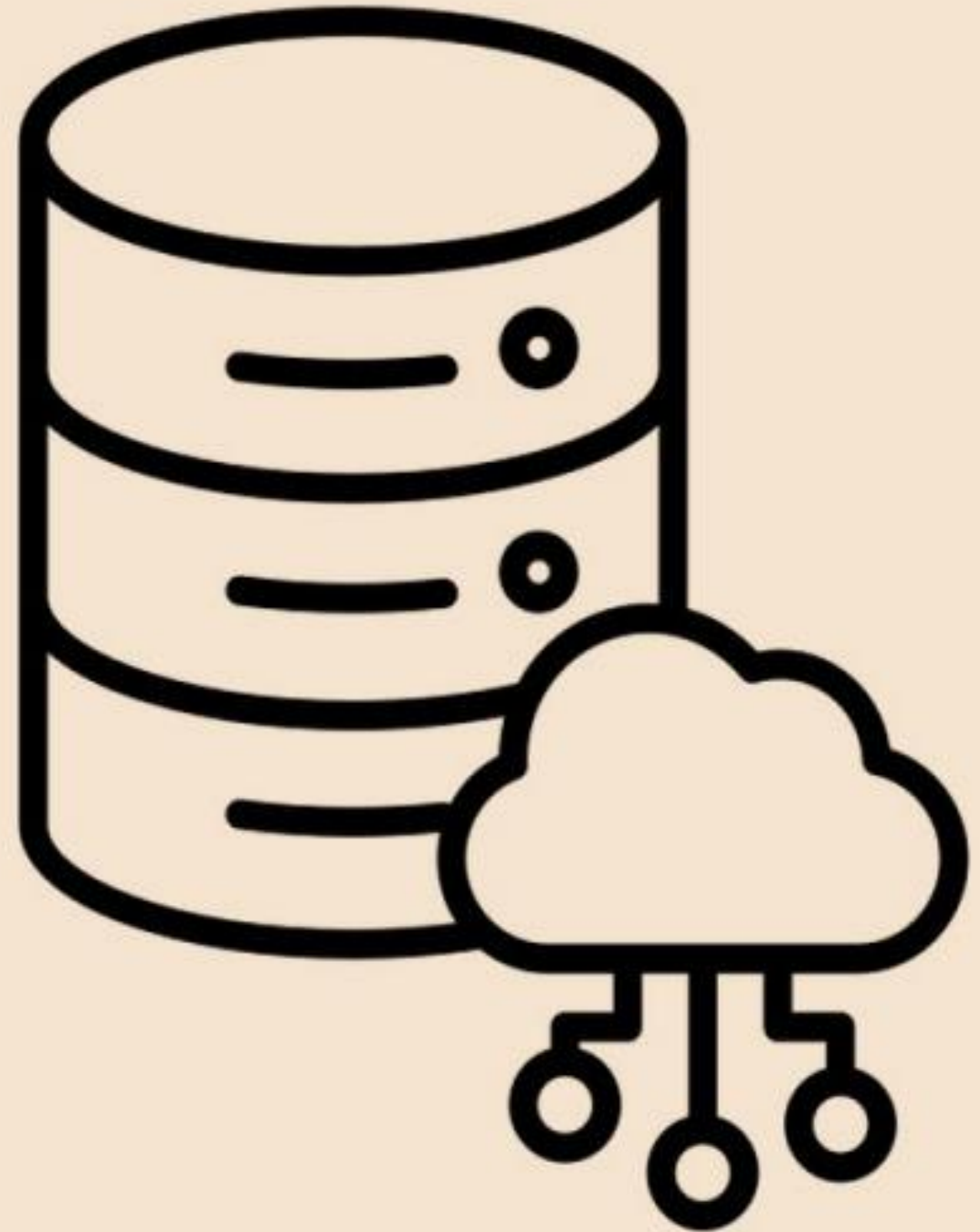
Database

(데이터베이스)

DDL

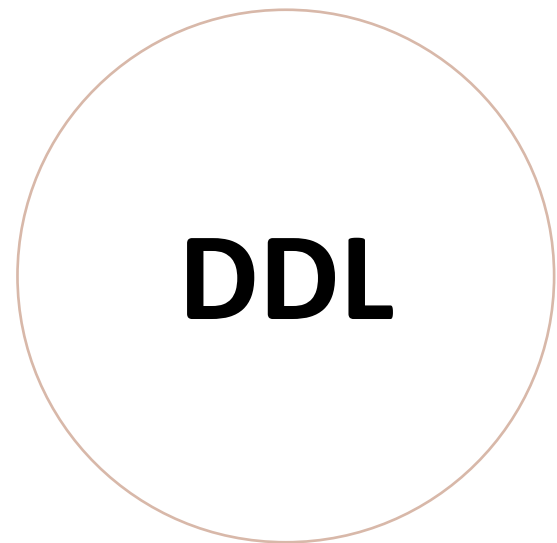
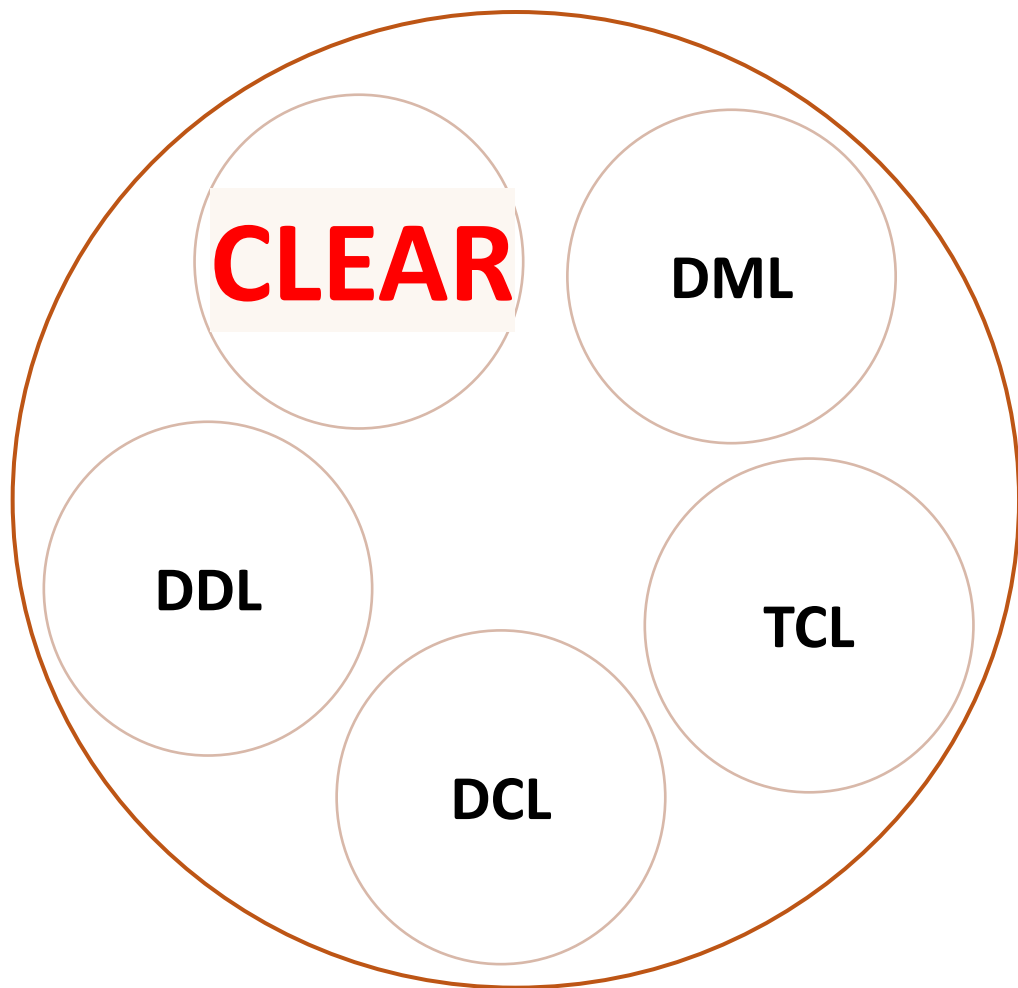
(Data Definition Language)

- 1 DDL 이란?
- 2 테이블의 자료형
- 3 테이블 생성하기
- 4 테이블 수정하기



DDL 이란?

SQL 문법



테이블 같은 데이터 저장소
객체의 생성/수정/삭제

DDL 이란?

DDL(Data Definition Language)

- 데이터 정의어
- 테이블과 같은 데이터 저장소 객체의 **생성/수정/삭제**
- **CREATE** : 새로운 저장소 객체를 생성
- **ALTER** : 테이블을 컬럼(열) 단위로 수정
- **DROP** : 테이블 객체와 데이터까지 전부 삭제
- **RENAME** : 컬럼의 이름을 변경
- **TRUNCATE** : 테이블 내부 데이터와 저장 공간 까지만 삭제

Step 2.

테이블의 자료형



테이블의 자료형

자료형 = DATA TYPE = 데이터를 저장하는 형식

↕ COLUMN_NAME	↕ DATA_TYPE	↕ NULLABLE	DATA_DEFAULT	↕ COLUMN_ID	↕ COMMENTS
1 직원ID	VARCHAR2(10 BYTE)	No	(null)	1	(null)
2 비밀번호	VARCHAR2(20 BYTE)	No	(null)	2	(null)
3 이름	VARCHAR2(30 BYTE)	Yes	(null)	3	(null)
4 성별	CHAR(3 BYTE)	Yes	(null)	4	(null)
5 나이	NUMBER(3,0)	Yes	(null)	5	(null)
6 입사일시	DATE	Yes	(null)	6	(null)
7 주민등록번호	VARCHAR2(15 BYTE)	No	(null)	7	(null)
8 연봉	NUMBER	Yes	(null)	8	(null)
9 부서ID	VARCHAR2(5 BYTE)	No	(null)	9	(null)

QUIZ) 자료형을 지정하는 이유?

가장 큰 이유는 **효율성** 때문!!

자료형에 따라서 테이블의 저장 공간을 효율적으로 관리할 수 있고,
연산을 위한 알고리즘도 최적화할 수 있다.

테이블의 자료형

테이블을 생성하는데 필요한 **자료형**

- VARCHAR2(n)** → 문자형 데이터 타입
(CHAR, NCHAR, LONG 등등...)
- NUMBER(n , m)** → 숫자형 데이터 타입
(FLOAT, BINARY_FLOAT 등등...)
- DATE** → 날짜형 데이터 타입
(TIMESTAMP)

< TIP > 위 3개의 대표적인 자료형만 알아도 무난하게 사용 가능!

테이블의 자료형

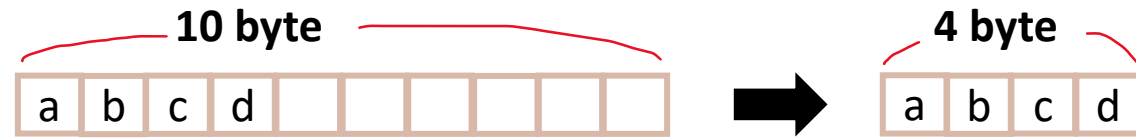
테이블을 생성하는데 필요한 자료형

VARCHAR2(n)

VARCHAR2(n)

문자형 값을 n 바이트까지 입력 받을 수 있는 **가변형 문자열**

예) varchar2(10) 인 컬럼에 'abcd' 를 입력한 경우



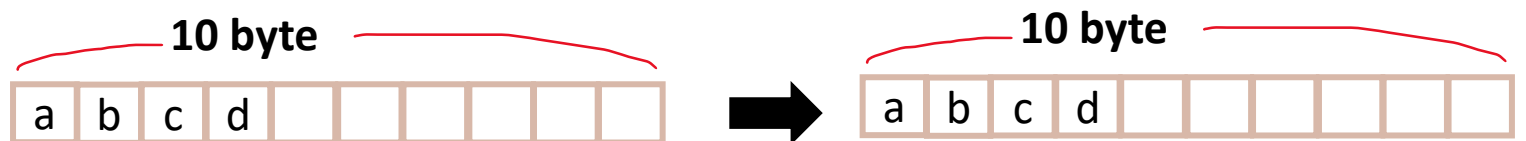
NUMBER(n , m)

DATE

CHAR(n)

문자형 값을 n 바이트까지 입력 받을 수 있는 **고정형 문자열**

예) char(10) 인 컬럼에 'abcd' 를 입력한 경우



테이블의 자료형

테이블을 생성하는데 필요한 자료형

VARCHAR2(n)

NUMBER(n , m)

DATE

NUMBER(n , m)

숫자 값을 n자리만큼 입력 받고 소수점 m자리만큼 입력 받는다.

NUMBER로 실수와 정수 모두 표현 가능

NUMBER 뒤에 n,m 은 생략 가능

< **TIP!!!** > 보통 n,m을 생략하고 NUMBER로 사용

예) NUMBER(3) 이면 999까지만 입력 가능

테이블의 자료형

테이블을 생성하는데 필요한 자료형

VARCHAR2(n)

NUMBER(n , m)

DATE

DATE

날짜 값을 입력 받는다.

이 외에도 **TIMESTAMP**라는 자료형도 존재

(둘 다 날짜 자료형이며 **TIMESTAMP**가 좀 더 구체적인 시간을 저장)

T_DATE	T_TIMESTAMP
23/01/07	23/01/07 20:52:43.000000000

Step 3.

테이블 생성하기



테이블 생성하기

CREATE

- 새로운 객체(OBJECT) 를 생성할 때 사용하는 명령어

CREATE TABLE ...

테이블 생성

CREATE USER ...

사용자 계정 생성

CREATE SEQUENCE ...

시퀀스 생성

CREATE VIEW ...

뷰 생성

테이블 생성하기

테이블 생성 문법 분석

[문법] CREATE TABLE 테이블명 (컬럼명 + 자료형 + DEFAULT로 지정할 값 + 제약조건);
(생략가능) (생략가능)

테이블명
CREATE TABLE QUIZ_TABLE(
Q_ID NUMBER(3, 0) NOT NULL, 컬럼에 NULL 입력 불가
Q_CONTENT VARCHAR2(200) NOT NULL ,
Q_ANSWER VARCHAR(100) ,
REG_DATE DATE DEFAULT SYSDATE ,
Q_BINGO VARCHAR(100) DEFAULT 'O'
);
새로운 튜플을 삽입할 때, 값을 지정하지 않으면
NULL값 대신 'O' 값이 기본으로 입력

테이블 생성하기

첫 시간에 생성했던 직원 테이블 DDL

(테이블 생성 정보 조회 : SHIFT + F4)

```
CREATE TABLE 직원 (
  직원ID VARCHAR2(10) ,
  비밀번호 VARCHAR2(20) NOT NULL ,
  이름 VARCHAR2(30) ,
  성별 CHAR(3) ,
  나이 NUMBER(3) ,
  입사일시 DATE ,
  주민등록번호 VARCHAR2(15) NOT NULL ,
  연봉 NUMBER ,
  부서ID VARCHAR2(5) NOT NULL
);
```

직원 -- 테이블이름을 드래그로 감싸서 SHIFT + F4 입력

열 제약 조건 권한 부여 통계 트리거 플래시백 종속성 세부정보 분할 영역 인덱스

새로고침: 0

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUM
1 직원ID	VARCHAR2(10 BYTE)	No	(null)	
2 비밀번호	VARCHAR2(20 BYTE)	No	(null)	
3 이름	VARCHAR2(30 BYTE)	Yes	(null)	
4 성별	CHAR(3 BYTE)	Yes	(null)	
5 나이	NUMBER(3,0)	Yes	(null)	
6 입사일시	DATE	Yes	(null)	
7 주민등록번호	VARCHAR2(15 BYTE)	No	(null)	
8 연봉	NUMBER	Yes	(null)	
9 부서ID	VARCHAR2(5 BYTE)	No	(null)	

도움말(H) 확인 취소

테이블 생성하기

<TIP> 테이블 생성 시 테이블 이름 규칙

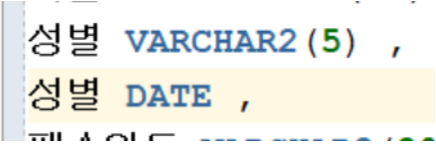
1. 대소문자 구분 X → 대문자로 변형

ex) create table aAa .. => AAA 테이블생성

2. 테이블 명 중복 불가

ex) create table AAA ... => 에러! 기존 객체가 있습니다.

3. 한 테이블 내에서 중복되는 컬럼 명 사용 불가

ex)  => 에러! 컬럼명이 중복되었습니다.

4. 문자로 시작해야 하며, 예약어 사용 불가

(테이블 이름으로 사용 가능한 문자 : a-z , A-Z , 0-9 , ㄱ-ㅎ , 특수문자는 _ , \$, # 만 사용가능)

Step 4.

테이블 수정하기



테이블 수정하기

ALTER

- 이미 만들어진 객체(OBJECT)를 수정할 때 사용하는 명령어

ALTER TABLE ADD ..

테이블에 컬럼 추가

ALTER TABLE DROP COLUMN..

테이블에서 컬럼 삭제

ALTER TABLE MODIFY ..

컬럼 속성 변경

ALTER TABLE RENAME ..

컬럼 이름 변경

테이블 수정하기

ALTER TABLE 테이블명 ADD 컬럼명 자료형 [default] [not null] ;

예) 직원들의 생년월일 정보를 저장할 수 있도록 “생년월일” 컬럼을 추가해주세요.

ALTER TABLE 직원 ADD (생년월일 VARCHAR2(8));

Table 직원이 (가) 변경되었습니다.

테이블에 컬럼 추가!

직원ID	패스워드	이름	성별	나이	입사일시	주민등록번호	연봉	부서ID	생년월일
A0001	12345	김철수	남	25	22/03/21	991212-1566123	2800	D001	(null)
A0002	hello123!	강홍수	남	28	21/09/12	950223-1562867	3000	D002	(null)
A0003	nono132	이현정	여	(null)	22/11/06	000112-4566123	2600	D003	(null)
A0004	123123!!	김선미	여	(null)	20/03/11	930722-2766443	4500	D004	(null)
A0005	test123	문현철	남	34	(null)	891231-1786155	5000	D005	(null)
A0006	774433	송대주	남	44	15/07/16	790903-1566127	7500	D001	(null)
A0007	pwd123	메이슨	남	40	16/08/19	830629-1676551	6200	D002	(null)
A0008	anjffhgkw1123	송진아	여	47	15/07/16	761212-2508143	7500	D003	(null)
A0009	test123	이서연	여	50	13/11/23	730317-259616	9000	D004	(null)
A0010	coffeegood!	김홍민	남	52	13/11/23	710513-1572876	9300	D005	(null)
A0012	pass1234	강감찬	남	31	23/01/07	931212-1212121	(null)	D001	(null)
A0014	hipasswd	공손한	여	50	23/01/06	740922-2555111	7000	D002	(null)

테이블 수정하기

ALTER TABLE 테이블명 DROP COLUMN 컬럼명 ;

예) 직원 테이블에서 생년월일 컬럼을 삭제해주세요.

ALTER TABLE 직원 DROP COLUMN 생년월일 ;

Table 직원이 (가) 변경되었습니다.

테이블의 컬럼 삭제!

직원ID	패스워드	이름	성별	나이	입사일시	주민등록번호	연봉	부서ID
A0001	12345	김철수	남	25	22/03/21	991212-1566123	2800	D001
A0002	hello123!	강홍수	남	28	21/09/12	950223-1562867	3000	D002
A0003	nono132	이현정	여	(null)	22/11/06	000112-4566123	2600	D003
A0004	123123!!	김선미	여	(null)	20/03/11	930722-2766443	4500	D004
A0005	test123	문현철	남	34	(null)	891231-1786155	5000	D005
A0006	774433	송대주	남	44	15/07/16	790903-1566127	7500	D001
A0007	pwd123	메이슨	남	40	16/08/19	830629-1676551	6200	D002
A0008	anjffhgkw1123	송진아	여	47	15/07/16	761212-2508143	7500	D003
A0009	test123	이서연	여	50	13/11/23	730317-259616	9000	D004
A0010	coffeegood!	김홍민	남	52	13/11/23	710513-1572876	9300	D005
A0012	pass1234	강감찬	남	31	23/01/07	931212-1212121	(null)	D001
A0014	hipasswd	공손한	여	50	23/01/06	740922-2555111	7000	D002

테이블 수정하기

ALTER TABLE 테이블명 MODIFY (컬럼명 자료형 [DEFAULT] [NOT NULL]);

예) 패스워드 컬럼의 자료형 길이를 300BYTE로 늘려주세요.

ALTER TABLE 직원 MODIFY (패스워드 VARCHAR2(300));

↕ COLUMN_NAME	↕ DATA_TYPE	↕ NULLABLE	DATA_DEFAULT
직원ID	VARCHAR2 (10 BYTE)	No	(null)
패스워드	VARCHAR2 (300 BYTE)	No	(null)
이름	VARCHAR2 (30 BYTE)	Yes	(null)

예) 부서ID 컬럼에 값이 꼭 들어갈 필요는 없으므로 NULLABLE 하게 바꿔주세요.

ALTER TABLE 직원 MODIFY (부서ID NULL);

연봉	NUMBER	Yes	(null)
부서ID	VARCHAR2 (5 BYTE)	Yes	(null)

테이블 수정하기

ALTER TABLE 테이블명 RENAME COLUMN 컬럼명 TO 바꿀컬럼명

예) 패스워드 컬럼의 이름을 “비밀번호” 로 변경을 해주세요.

ALTER TABLE 직원 RENAME COLUMN 패스워드 TO 비밀번호 ;

↕ COLUMN_NAME	↕ DATA_TYPE	↕ NULLABLE	DATA_DEFAULT
직원ID	VARCHAR2 (10 BYTE)	No	(null)
비밀번호	VARCHAR2 (300 BYTE)	No	(null)
이름	VARCHAR2 (30 BYTE)	Yes	(null)

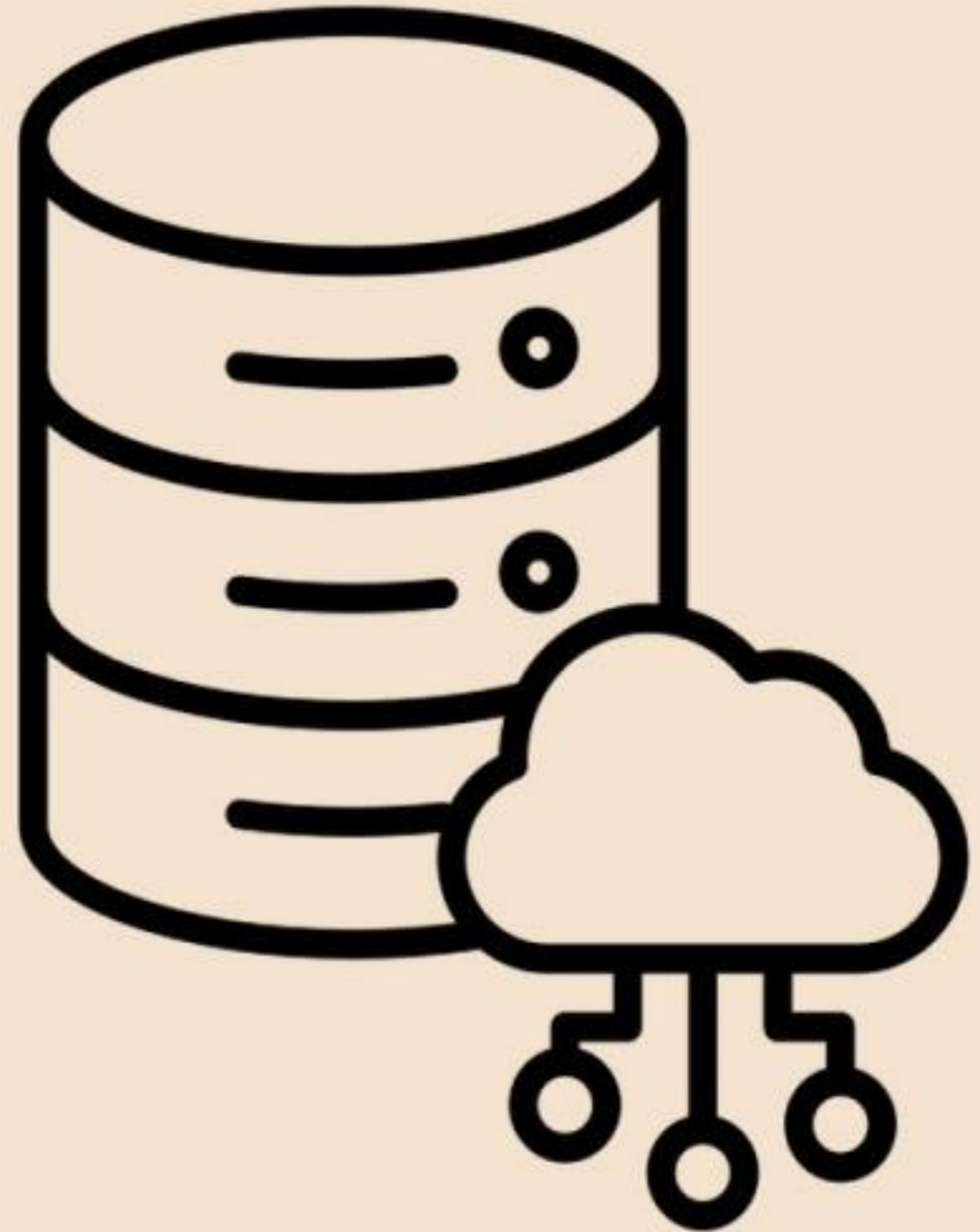
DDL

(Data Definition Language)

5 제약조건

6 테이블 삭제하기

7 시퀀스(SEQUENCE)와
뷰(VIEW)



제약조건

제약조건(CONSTRAINTS)

테이블에 입력 가능한 데이터를 조건으로 제약하는 것

NOT NULL

UNIQUE KEY (UK)

CHECK

★ **PRIMARY KEY (PK)** ★

★ **FOREIGN KEY (FK)** ★

제약조건

NOT NULL

컬럼이 **비어 있거나 NULL 값이 들어오지 않도록** 하는 조건

정보가 꼭 필요한 컬럼에 사용 예) 직원 테이블의 사원 이름

[문법1] ALTER TABLE 테이블명 MODIFY 컬럼명 + 자료형 + NOT NULL ;

[문법2] CREATE TABLE 테이블명 (컬럼명 + 자료형 + NOT NULL)

예시 1) ALTER TABLE QUIZ_TABLE **MODIFY Q_ANSWER VARCHAR(100) NOT NULL ;**

예시 2) CREATE TABLE QUIZ_TABLE(
Q_ID NUMBER(3, 0) PRIMARY KEY ,
Q_CONTENT VARCHAR2(200) UNIQUE ,
Q_ANSWER VARCHAR(100) **NOT NULL,**
REG_DATE DATE DEFAULT SYSDATE ,
Q_BINGO VARCHAR(100) DEFAULT 'O'
);

제약조건

UNIQUE KEY(UK) : UNIQUE

하나의 컬럼에 **중복되는 튜플이 존재할 수 없도록** 하는 조건

[문법1] ALTER TABLE 테이블명 ADD CONSTRAINT 제약조건명 UNIQUE (컬럼);

[문법2] CREATE TABLE 테이블명 (컬럼명 + 자료형 + UNIQUE)

예시 1) ALTER TABLE QUIZ_TABLE **ADD CONSTRAINT UK QUIZ_TABLE UNIQUE(Q_CONTENT)** ;

예시 2) CREATE TABLE QUIZ_TABLE(
Q_ID NUMBER(3, 0) PRIMARY KEY,
Q_CONTENT VARCHAR2(200) **UNIQUE**,
Q_ANSWER VARCHAR(100) NOT NULL,
REG_DATE DATE DEFAULT SYSDATE ,
Q_BINGO VARCHAR(100) DEFAULT 'O'
);

제약조건

CHECK

특정 컬럼에 데이터를 입력할 때 **조건에 해당하는 데이터만 입력할 수 있도록 제약**
CHECK 조건에 만족하는 정보만 튜플에 입력 가능 예) 성별, o/x 퀴즈 결과, 학점 등

[문법1] ALTER TABLE 테이블명 ADD CONSTRAINT 제약조건명 CHECK (조건 OR 범위);

[문법2] CONSTRAINT 제약조건이름 CHECK (조건 OR 범위)

예시 1) ALTER TABLE QUIZ_TABLE **ADD CONSTRAINT Q_BINGO_CK CHECK (Q_BINGO IN ('O', 'X')) ;**

예시 2) CREATE TABLE QUIZ_TABLE(
Q_ID NUMBER(3, 0) PRIMARY KEY ,
Q_CONTENT VARCHAR2(200) UNIQUE ,
Q_ANSWER VARCHAR(100) NOT NULL,
REG_DATE DATE DEFAULT SYSDATE ,
Q_BINGO VARCHAR(100) DEFAULT 'O'
CONSTRAINT Q_BINGO_CK CHECK (Q_BINGO IN('O', 'X'))
);

제약조건

PRIMARY KEY (PK) : NOT NULL + UNIQUE

식별자 규칙을 물리적 모델링 한 것. 기본 키, 주 키, 프라이머리 키라고도 부름
보통 튜플(행)이 지닌 **유일하고 고유한 특징**인 컬럼을 PK로 지정
NULL값 입력 불가, 중복 불가의 특징

생각해봅시다!

예) 새로운 휴대폰을 맞추려고 방문한 휴대폰 매장에서는 내 정보를 어떻게 찾을까?
라일락 회사의 '이지은' 사원의 연봉 정보를 찾으려면 어떤 정보를 기준으로 찾을까?
한국대학교의 '김민수' 학생을 찾기 위해서는 DB의 어떤 컬럼을 기준으로 찾을까?

제약조건

PRIMARY KEY (PK) : NOT NULL + UNIQUE

특정 컬럼을 식별자로 만들면 **자동으로 NOT NULL + UNIQUE 성질**을 가짐

[문법1] ALTER TABLE 테이블명 ADD CONSTRAINT 제약조건명 PRIMARY KEY (컬럼);

[문법2] CREATE TABLE 테이블명 (컬럼명 + 자료형 + PRIMARY KEY)

예시 1) ALTER TABLE QUIZ_TABLE **ADD CONSTRAINT PK_QUIZ_TABLE PRIMARY KEY(Q_ID) ;**

예시 2) CREATE TABLE QUIZ_TABLE(
Q_ID NUMBER(3, 0) **PRIMARY KEY**,
Q_CONTENT VARCHAR2(200) UNIQUE ,
Q_ANSWER VARCHAR(100) NOT NULL,
REG_DATE DATE DEFAULT SYSDATE ,
Q_BINGO VARCHAR(100) DEFAULT 'O'
);

이제 QUIZ_TABLE의 Q_ID에는
중복 값과 NULL 값 입력 불가!

제약조건

FOREIGN KEY (FK)

다른 테이블에 있는 **기본 키(PRIMARY KEY)**를 참조하는 컬럼을 FK로 지정
FOREIGN KEY는 외래 키라고 부르고 **데이터 무결성**을 지원
참조하는 테이블의 **기본 키 값**과 반드시 **동일한 값**으로 구성되어야 함

예) 직원ID 가 A0001인 직원의 이름과 휴대폰을 찾아봅시다

A 테이블

직원ID	이름
A1	김민수
B2	이지은
C3	장범준

B테이블의 직원 ID는
A테이블의 직원 ID를
참조하고 있다.

B 테이블

직원ID	구분코드	연락처
A1	휴대폰	010-111-1111
A1	집전화	062-111-1111
B2	집전화	062-222-2222

제약조건

FOREIGN KEY (FK)

[문법1] ALTER TABLE 테이블명 **ADD CONSTRAINT** 제약조건명
FOREIGN KEY (참조받을 컬럼) **REFERENCES** 참조할 테이블(참조할 컬럼) ;

[문법2] CREATE TABLE 테이블명 (생성할 컬럼 정보 , **CONSTRAINT** 제약조건명
FOREIGN KEY (참조받을 컬럼) **REFERENCES** 참조할 테이블(참조할 컬럼)) ;

예시1)

```
ALTER TABLE QUIZ_TABLE2 ADD CONSTRAINT TABLE1_TABLE_2_FK  
FOREIGN KEY(Q_ID) REFERENCES QUIZ_TABLE (Q_ID) ;
```

예시2)

```
CREATE TABLE QUIZ_TABLE2 (  
    Q_ID NUMBER(3,0) ,  
    Q_REAL VARCHAR2(200) ,  
    CONSTRAINT TABLE1_TABLE2_FK FOREIGN KEY (Q_ID)  
    REFERENCES QUIZ_TABLE (Q_ID)  
);
```

제약조건

실습 문제

1. 다음 조건에 맞춰 3개의 테이블을 생성 (CREATE) 하세요.

DEFAULT 나 NOT NULL 이 따로 없는 경우는 별도로 설정하지 않음을 의미합니다.

테이블명 : 회원정보

컬럼/자료형	: 회원ID	-	가변형문자형	10BYTE	, NOT NULL
	이름	-	가변형문자형	20BYTE	, NOT NULL
	가입일자	-	날짜형		
	나이	-	숫자형	DEFAULT 0	

테이블명 : 회원연락처

컬럼/자료형	: 회원ID	-	가변형문자형	10BYTE	, NOT NULL
	구분코드	-	가변형문자형	10BYTE	, NOT NULL
	연락처	-	가변형문자형	15BYTE	, NOT NULL

테이블명 : 회원주소

컬럼/자료형	: 회원ID	-	가변형문자형	10BYTE	, NOT NULL
	도로명주소	-	가변형문자형	200BYTE	, NOT NULL

제약조건

실습 답안

```
CREATE TABLE 회원정보 (
  회원 ID VARCHAR2(10) NOT NULL ,
  이름 VARCHAR2(30) NOT NULL ,
  가입일자 DATE ,
  나이 NUMBER DEFAULT 0
) ;
```

```
CREATE TABLE 회원연락처 (
  회원 ID VARCHAR2(10) NOT NULL ,
  구분코드 VARCHAR2(10) NOT NULL ,
  연락처 VARCHAR2(15) NOT NULL
) ;
```

```
CREATE TABLE 회원주소 (
  회원 ID VARCHAR2(10) NOT NULL ,
  도로명주소 VARCHAR2(200) NOT NULL
) ;
```

회원정보

열 제약 조건 권한 부여 통계 트리거 플래시백 종속성 세부정보 분할 영역

새로고침: 0

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT
1 회원 ID	VARCHAR2 (10 BYTE)	No	(null)
2 이름	VARCHAR2 (30 BYTE)	No	(null)
3 가입일자	DATE	Yes	(null)
4 나이	NUMBER	Yes	0

회원연락처

열 제약 조건 권한 부여 통계 트리거 플래시백 종속성 세부정보 분할 영역

새로고침: 0

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT
1 회원 ID	VARCHAR2 (10 BYTE)	No	(null)
2 구분코드	VARCHAR2 (10 BYTE)	No	(null)
3 연락처	VARCHAR2 (15 BYTE)	No	(null)

회원주소

열 제약 조건 권한 부여 통계 트리거 플래시백 종속성 세부정보 분할 영역

새로고침: 0

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT
1 회원 ID	VARCHAR2 (10 BYTE)	No	(null)
2 도로명주소	VARCHAR2 (200 BYTE)	No	(null)

제약조건

실습 문제

2. 1번에서 만든 3개의 테이블에 각각 주어진 조건에 맞춰 PK (PRIMARY KEY) 제약조건을 추가해주세요.

힌트 : 두 개 이상의 컬럼을 PK로 설정하려면 다음과 같은 형식으로 작성해주세요 => PRIMARY KEY (컬럼1 , 컬럼2)

각 테이블 별로 사용할 PK 컬럼과 제약조건 이름은 아래와 같습니다.

회원정보 테이블의 PK	: 회원ID	--제약조건명 : PK_회원정보
회원연락처 테이블의 PK	: 회원ID + 구분코드	--제약조건명 : PK_회원연락처
회원주소 테이블의 PK	: 회원ID	--제약조건명 : PK_회원주소

제약조건

실습 답안

ALTER TABLE 회원정보 ADD CONSTRAINT PK_회원정보 PRIMARY KEY (회원ID) ;

ALTER TABLE 회원연락처 ADD CONSTRAINT PK_회원연락처 PRIMARY KEY (회원ID, 구분코드) ;

ALTER TABLE 회원주소 ADD CONSTRAINT PK_회원주소 PRIMARY KEY (회원ID) ;

제약조건

실습 문제

3. 1번에서 만든 3개의 테이블에 대해 테이블 간의 관계 연결을 위해 **FK (FOREIGN KEY)** 제약조건을 설정하시오.

회원연락처 테이블의 [회원ID] 컬럼은 회원정보 테이블의 [회원ID] 를 참조하고자 합니다.

제약조건명 : **FK_회원연락처**

힌트 : **ALTER TABLE** 회원연락처 ...

회원주소 테이블의 [회원ID] 컬럼은 회원정보 테이블의 [회원ID] 를 참조하고자 합니다.

제약조건명 : **FK_회원주소**

힌트 : **ALTER TABLE** 회원주소 ...

제약조건

실습 답안

```
ALTER TABLE 회원연락처 ADD CONSTRAINT FK_회원연락처  
FOREIGN KEY (회원ID) REFERENCES 회원정보(회원ID) ;
```

```
ALTER TABLE 회원주소 ADD CONSTRAINT FK_회원주소  
FOREIGN KEY (회원ID) REFERENCES 회원정보(회원ID) ;
```

Step 6.

테이블 삭제하기



테이블 삭제하기

DROP

- 테이블 및 제약조건을 삭제할 때 사용하는 명령어

DROP TABLE ..

테이블을 삭제

DROP TABLE .. CASCADE CONSTRAINTS

테이블의 제약조건까지 함께 삭제

테이블 삭제하기

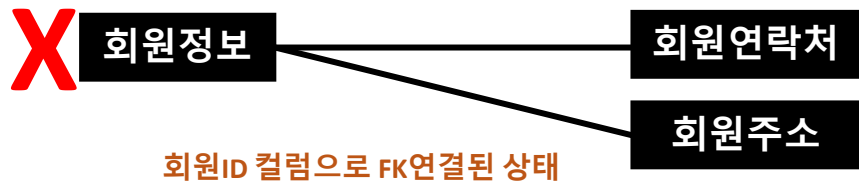
DROP TABLE 테이블명 [CASCADE CONSTRAINTS]

→ **테이블 영구 삭제** CASCADE CONSTRAINTS 옵션을 추가하면 관련 관계선(FK)도 모두 삭제 가능

예) 아까 만들었던 회원정보 테이블을 삭제해봅시다.

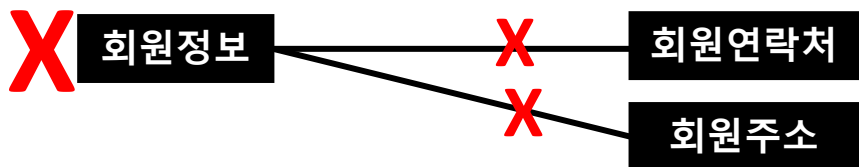
DROP TABLE 회원정보 ;

ORA-02449: 외래 키에 의해 참조되는 고유/기본 키가 테이블에 있습니다



현재 회원정보의 회원ID를 참조하는 테이블
존재 (회원연락처, 회원주소)
이 경우 테이블 삭제 불가능

DROP TABLE 회원정보 CASCADE CONSTRAINTS ;



회원ID 컬럼으로 FK연결된 관계도 같이 제거

CASCADE CONSTRAINT 옵션을 사용하면
회원정보와 연결된 관계선(FK)도 함께 제거 가능

테이블 삭제하기

ALTER TABLE 테이블명 DROP CONSTRAINT 제약조건명;

→ 설정해 놓은 제약조건 삭제

예) 아까 만들었던 회원연락처 테이블의 PRIMARY KEY 제약조건을 삭제해주세요.

2. 1번에서 만든 3개의 테이블에 각각 주어진 조건에 맞춰 PK (PRIMARY KEY) 제약조건을 추가해주세요.
 힌트 : 두 개 이상의 컬럼을 PK로 설정하려면 다음과 같은 형식으로 작성해주세요 => PRIMARY KEY (컬럼1 , 컬럼2)

각 테이블 별로 사용할 PK 컬럼과 제약조건 이름은 아래와 같습니다.

회원정보 테이블의 PK	: 회원ID	--제약조건명 : PK_회원정보
회원연락처 테이블의 PK	: 회원ID + 구분코드	--제약조건명 : PK_회원연락처
회원주소 테이블의 PK	: 회원ID	--제약조건명 : PK_회원주소

ALTER TABLE 회원연락처 DROP CONSTRAINT PK_회원연락처 ;

Table 회원연락처이 (가) 변경되었습니다.

Step 7.

시퀀스(SEQUENCE)
뷰(VIEW)



시퀀스(SEQUENCE)와 뷰(VIEW)

시퀀스(SEQUENCE)란?

연속적인 사건이라는 뜻이며, **자동으로 증가하는 값을 만들어주는 객체**

CREATE SEQUENCE 회원ID_SEQ

(INCREMENT BY 1 **증가할 시퀀스 폭**

START WITH 1 **시작할 시퀀스 값**

MINVALUE 1 **시퀀스 최소 값**

MAXVALUE 9999 **시퀀스 최대 값**)

시퀀스(SEQUENCE)와 뷰(VIEW)

SEQUENCE 값 확인하는 방법

```
SELECT 회원ID_SEQ.NEXTVAL  [시퀀스명.NEXTVAL] 형식으로 시퀀스 값 호출  
FROM DUAL ;
```

SEQUENCE를 사용하는 이유

1. PRIMARY KEY가 될 만한 컬럼이 존재하지 않을 때
2. 회원ID, 직원ID 처럼 규칙적으로 증가하는 값에 사용
→ NEXTVAL 키워드를 통해 이전 사원의 번호를 조회하지 않아도 다음 번호 사용 가능

시퀀스(SEQUENCE)와 뷰(VIEW)

[SEQUENCE 삭제하기]

DROP SEQUENCE 회원ID_SEQ;

Sequence 회원ID_SEQ이 (가) 삭제되었습니다.

시퀀스(SEQUENCE)와 뷰(VIEW)

뷰(VIEW)란?

일종의 “**가상테이블**” 을 의미, 테이블과 다르게 **물리적으로 존재 x**

BUT!!

사용자에게는 있는 것으로 간주됨

정보 보안 측면에서 유리하다는 장점

보통 자주 사용하는 쿼리를 뷰(VIEW)에 저장하여 사용

시퀀스(SEQUENCE)와 뷰(VIEW)

VIEW를 생성하려면 권한 필요

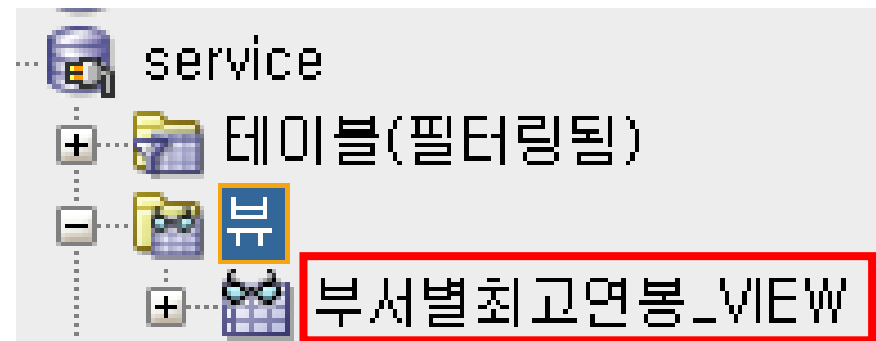
```
SQL> GRANT CREATE VIEW TO SERVICE;  
Grant succeeded.
```

ex) 만약 부서별 최고 연봉을 검색하는 쿼리를 자주 사용한다면

→ VIEW에 저장해두고 필요할 때 불러서 사용

```
CREATE VIEW 부서별최고연봉_VIEW AS  
SELECT 부서ID, MAX(연봉) AS 부서별최고연봉  
FROM 직원  
GROUP BY 부서ID  
ORDER BY 부서ID;
```

View 부서별최고연봉_VIEW이 (가) 생성되었습니다.



시퀀스(SEQUENCE)와 뷰(VIEW)

VIEW 사용 원리

테이블처럼 **FROM** 뒤에 **VIEW** 이름을 입력해 사용

```
SELECT *  
FROM 부서별최고연봉_VIEW ;
```



(1) VIEW 호출

```
CREATE VIEW 부서별최고연봉 VIEW AS  
SELECT 부서ID , MAX(연봉) AS 부서별최고연봉  
FROM 직원  
GROUP BY 부서ID  
ORDER BY 부서ID ;
```

(2) 해당 VIEW에 정의된 쿼리를 실행



(3) 해당 결과를 출력

부서ID	부서별최고연봉
D001	7500
D002	7000
D003	7500
D004	9000
D005	9300
D006	5000

시퀀스(SEQUENCE)와 뷰(VIEW)

VIEW는 실제 테이블과 JOIN도 가능

예) 부서별로 가장 높은 연봉을 가진 직원들의 정보를 출력해 주세요

```
SELECT A.이름, A.연봉, B.부서별최고연봉  
FROM 직원 A, 부서별최고연봉_VIEW B  
WHERE A.부서ID = B.부서ID  
AND A.연봉 = B.부서별최고연봉 ;
```



**VIEW와 직원 테이블을
조인하여 결과 출력!**

시퀀스(SEQUENCE)와 뷰(VIEW)

그럼 왜, WHY? VIEW를 사용할까?

(1) 자주 사용하는 쿼리를 저장해 놓고 이용할 수 있으므로 명령어 입력양 감소

```
CREATE VIEW 부서별최고연봉_VIEW AS  
SELECT 부서ID , MAX(연봉) AS 부서별최고연봉  
FROM 직원  
GROUP BY 부서ID  
ORDER BY 부서ID ;
```



```
SELECT *  
FROM 부서별최고연봉_VIEW ;
```

시퀀스(SEQUENCE)와 뷰(VIEW)

그럼 왜, WHY? VIEW를 사용할까?

(2) 원하는 데이터만 보여줄 수 있게 해 **보안 목적으로 사용 가능**

```
CREATE VIEW 직원_민감정보제외_VIEW AS  
SELECT 직원ID, 이름, 부서ID  
FROM 직원;
```

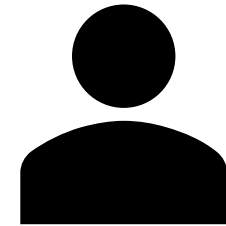
패스워드, 주민등록번호, 연봉 등
민감정보를 제외하고 뷰를 생성

직원ID	이름	부서ID
A0001	김철수	D001
A0002	강홍수	D002
A0003	이현정	D003
A0004	김선미	D004
A0005	문현철	D005
A0006	송대주	D001
A0007	메이슨	D002
A0008	송진아	D003
A0009	이서연	D004
A0010	김홍민	D005
A0012	강감찬	D001
A0017	새직원	D006

```
SELECT *  
FROM 직원_민감정보제외_VIEW;
```

접근 가능

외부업체 직원



```
SELECT *  
FROM 직원;
```

접근 불가

직원 테이블은 못 보게 하고
직원_민감정보제외_VIEW만
볼 수 있게 권한 설정 가능

시퀀스(SEQUENCE)와 뷰(VIEW)

VIEW 삭제하기

DROP VIEW 부서별최고연봉_VIEW ;

| View 부서별최고연봉_VIEW이 (가) 삭제되었습니다.