**01. 관계형 데이터베이스 개요**

1. 데이터베이스

- 데이터베이스 : 기업이나 개인이 필요에 의해 데이터를 일정한 형태로 저장

- 1970년대에 계층형, 망형(네트워크형) 데이터베이스 등이 상용화됨

- 관계형 데이터베이스는 1970년대에 첫 소개, 1980년대에 상용화됨

1) 계층형 데이터베이스

2) 네트워크형 데이터베이스

3) 관계형 데이터베이스

- 관계형 데이터베이스는 릴레이션(Relation)과 릴레이션의 조인 연산으로   
합집합/ 교집합/ 차집합 등을 만들 수 있음

★데이터베이스 Vs. 데이터베이스 관리 시스템  
- 데이터베이스 관리 시스템(Database Management System)은 계층형, 네트워크형, 관계형 데이터베이스를 관리하기 위한 *“소프트웨어”* 임  
- Oracle, MS-SQL, MySQL, Sybase 등이 있으며, 모두 관계형 데이터베이스를 지원함

**[Quiz]**

**데이터베이스 종류 중에서 부모와 자식 관계를 표현하기 쉬운 데이터베이스는?**

① 계층형 데이터베이스  
② 네트워크 데이터베이스  
③ 관계형 데이터베이스  
④ 빅데이터

**2. SQL(Structured Query Language)**

- 관계형 데이터베이스에서 **데이터 정의/ 조작/ 제어**를 하기 위해 사용하는 언어.

- 관계형 데이터베이스에서는 SQL문을 사용하여 누구나 쉽게 데이터베이스를 사용할 수 있음

- SQL은 ANSI/ISO 표준을 준수 → 데이터베이스 관리 시스템이 변경되어도 그대로 사용 가능

|  |  |  |
| --- | --- | --- |
| **명령어 종류** | **명령어** | **설 명** |
| DDL | CREATE  ALTER  DROP  RENAME | 관계형 데이터베이스의 구조를 정의하는 언어 |
| DML | INSERT  UPDATE  DELETE  SELECT | - 테이블에서 데이터를 입력, 수정, 삭제, 조회  - 데이터베이스의 테이블에 들어있는 데이터에 변형을 가하는 종류의 명령어/ 또는 데이터를 조회하는 명령어 |
| DCL | GRANT  REVOKE  TRUMCATE | 데이터베이스 사용자에게 권한을 부여하고나 회수 |
| TCL | COMMIT  ROLLBACK  SAVEPOINT | 트랜잭션을 제어하는 명령어 |

1) 트랜잭션(Transaction) 이란

- 트랜잭션은 ***데이터베이스의 작업을 처리하는 단위***

|  |  |
| --- | --- |
| **트랜잭션 특성** | **설 명** |
| **원자성** | - 트랜잭션은 데이터베이스 연산의 전부가 실행되거나 전혀 실행되지 않아야 한다(ALL OR NOTHING)  - 특, 트랜잭션의 처리가 완전히 끝나지 않았을 경우는 실행되지 않은 상태와 같아야 함 |
| **일관성** | - 트랜잭션 실행 결과로 데이터베이스의 상태가 모순되지 않아야 함  - 트랜잭션 실행 후에도 일관성이 유지되어야 한다 |
| **고립성** | - 트랜잭션 실행 중에 생성하는 연산의 중간결과는 다른 트랜잭션이 접근할 수 없다  - 즉, 부분적인 실행결과를 다른 트랜잭션이 볼 수 없다 |
| **영속성** | - 트랜잭션이 그 실행을 성공적으로 완료하면 그 결과는 영구적 보장이 되어야 한다 |

**[Quiz]**

**트랜잭션의 특징 중에서 다른 트랜잭션이 접근하여 부분적인 실행 결과를 볼 수 없는 것은?**

① 원자성  
② 일관성  
③ 고립성  
④ 영속성

2) SQL문의 실행 순서

- 개발자가 작성한 SQL문은 3단계를 걸쳐서 실행됨  
 - 파싱(SQL 문법을 검사하고 구문을 분석) > 실행 > 데이터 인출

|  |  |
| --- | --- |
| **트랜잭션 특성** | **설 명** |
| **파싱**  **(Parsing)** | - SQL문의 문법을 확인 & 구문 분석  - 구문 분석한 SQL문은 Library Cache에 저장한다 |
| **실행**  **(Execution)** | 옵티마이저(Optimizer)가 수립한 실행 계획에 따라 SQL을 실행한다 |
| **인출**  **(Fetch)** | 데이터를 읽어서 전송 |

**[Quiz]**

**데이터베이스 관리 시스템이 SQL을 실행하는 순서로 올바른 것은?**

① 파싱 > 실행 > 인출   
② 실행 > 파싱 > 인출  
③ 인출 > 파싱 > 실행  
④ 실행 > 인출 > 파싱

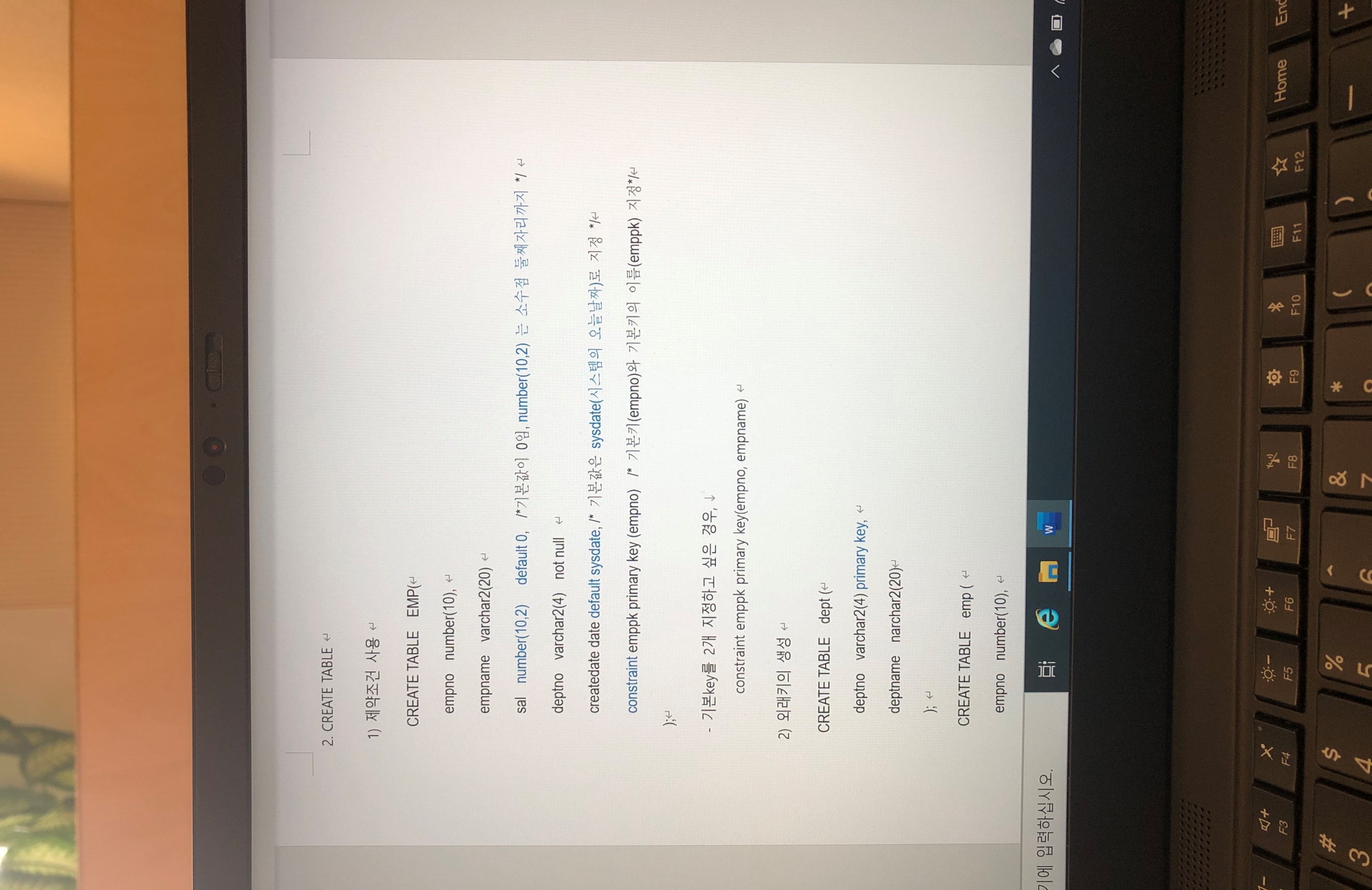
**02. DDL**

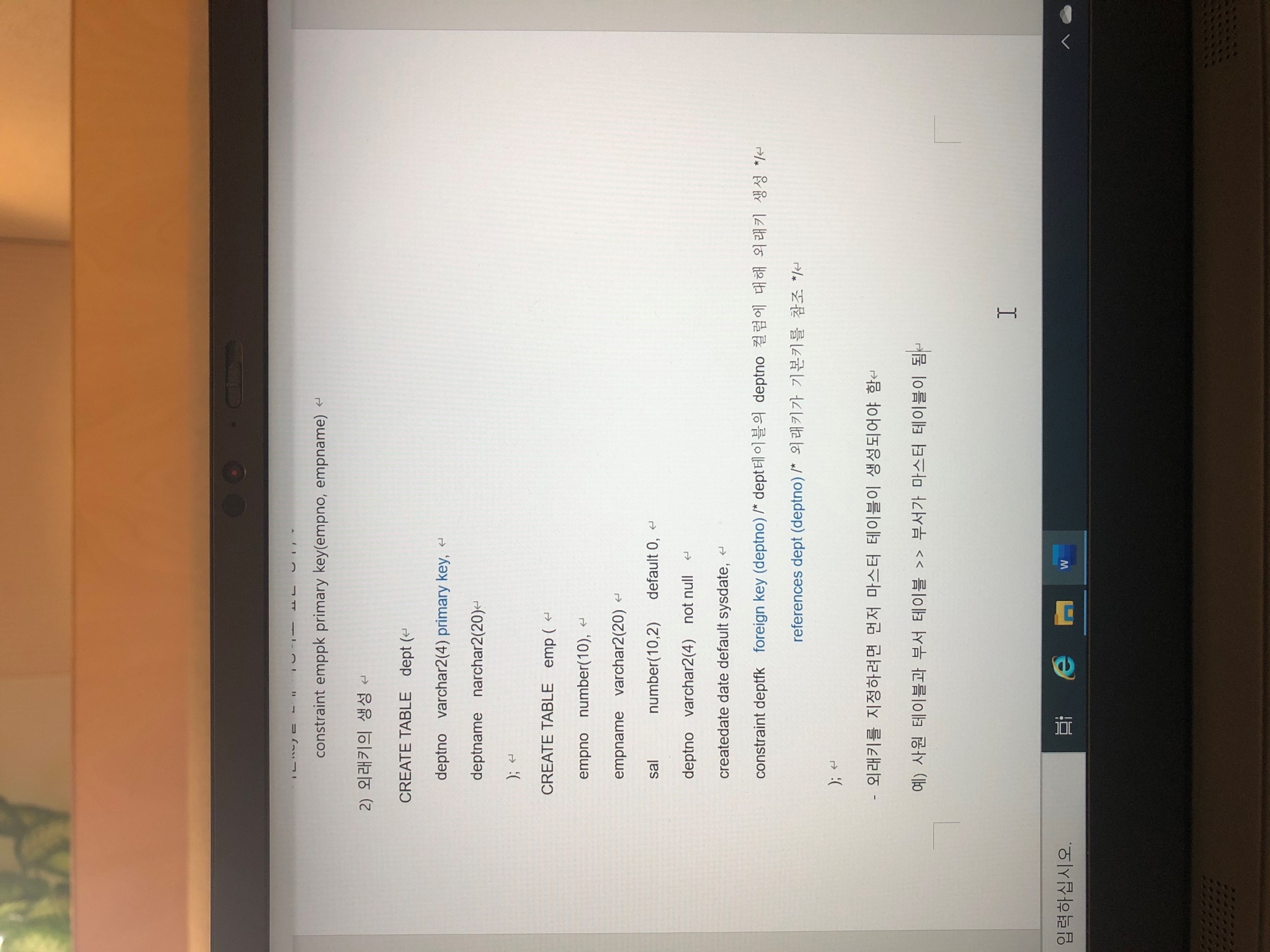
1. 데이터 유형

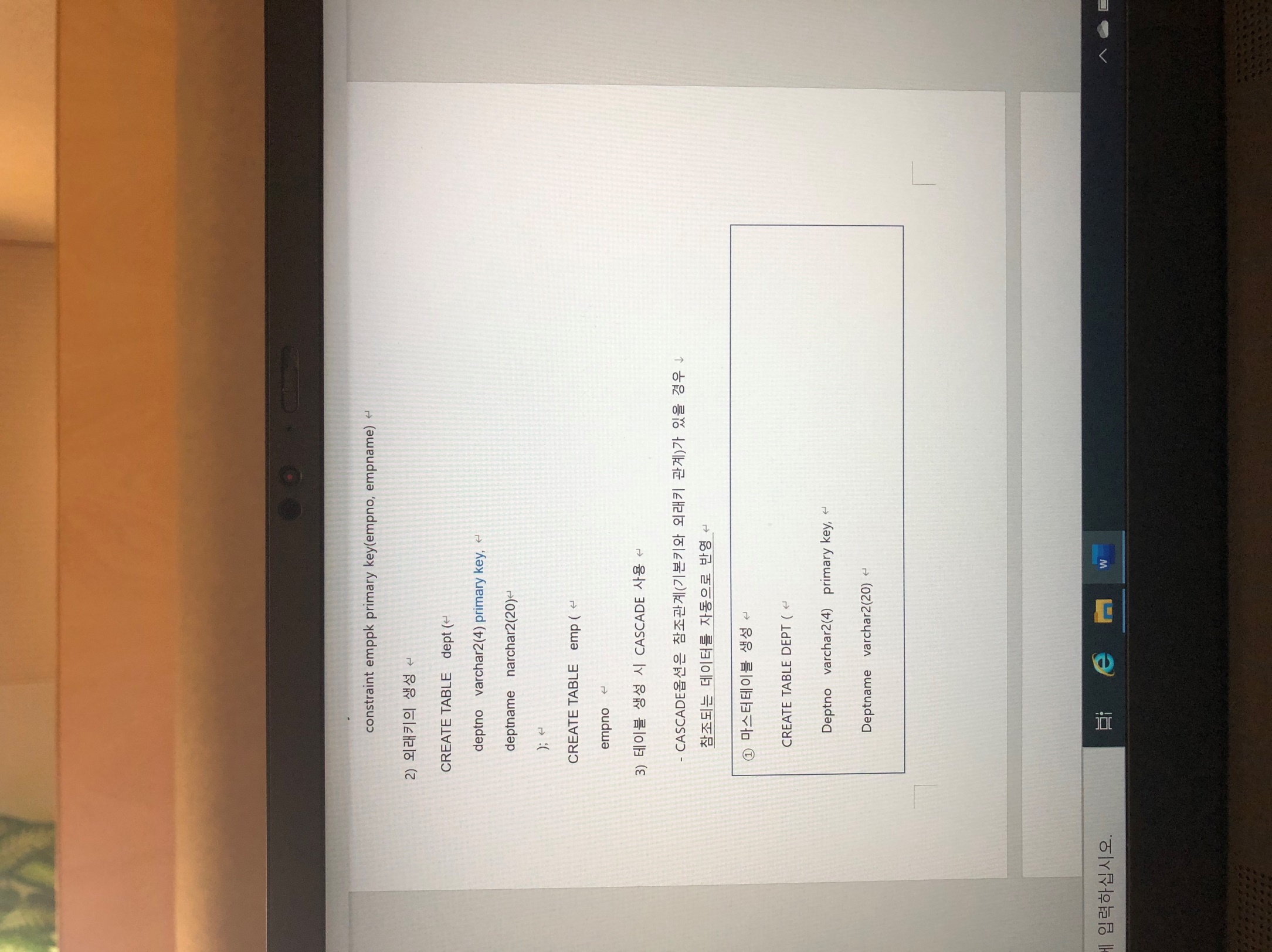
- 자주 쓰이는 데이터 유형으로는 **CHARACTER(s) / VARCHAR(s) / NUMERIC / DATETIME**

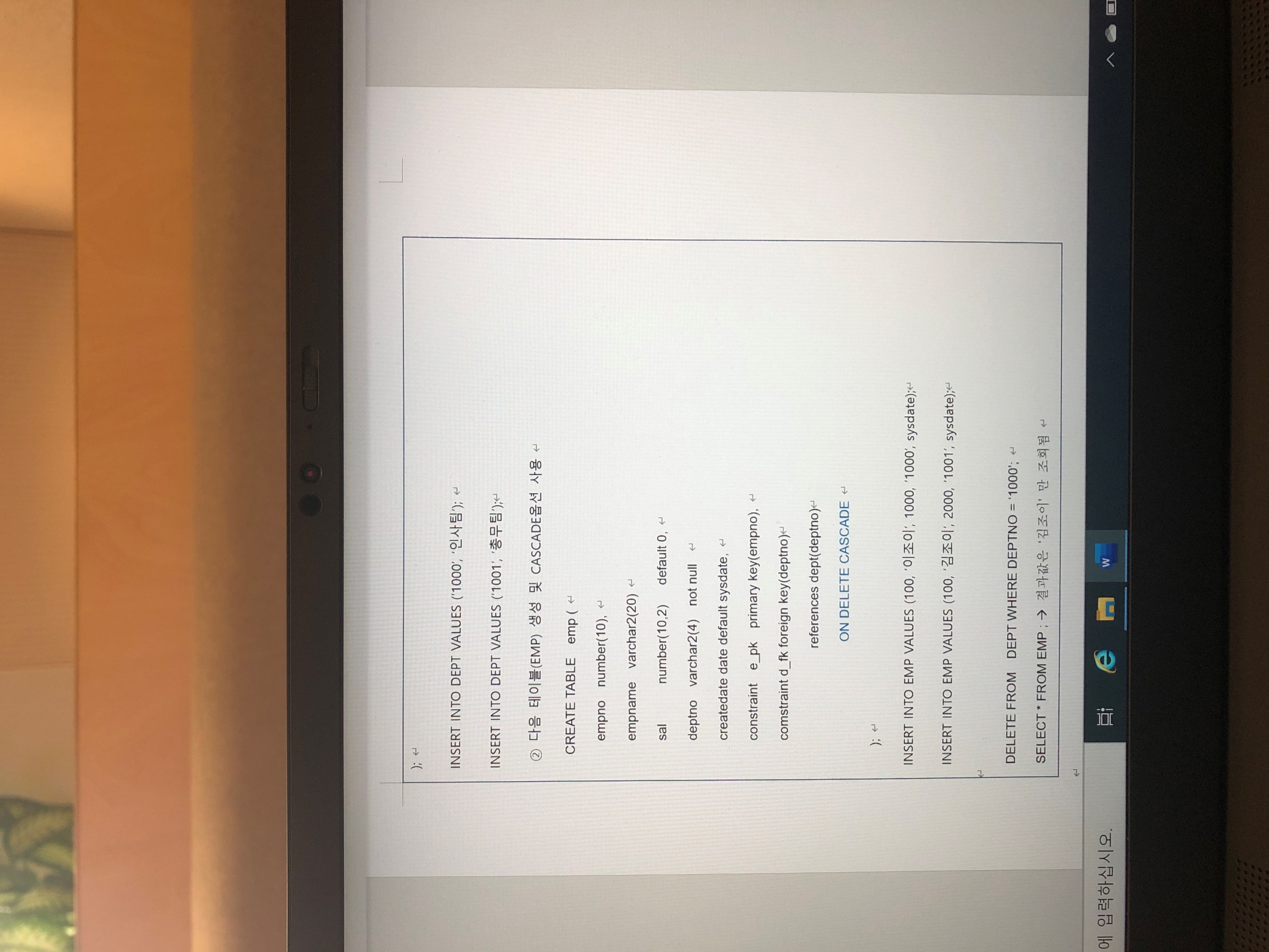
등이 있다.

2. CREATE TABLE









- ON DELETE CASCADE 옵션은 자신이 참조하고 있는 테이블(DEPT)의 데이터가 삭제되면  
 자동으로 자기 자신에서도 삭제되는 옵션임

- ON DELETE CASCADE 옵션을 사용하면 참조 무결성을 준수할 수 있음

※ 참조 무결성이 ‘위배’되는 경우 : 마스터 테이블(DEPT)에는 해당 부서번호(DEPTNO)가   
 없는데, 오히려 슬레이브 테이블(EMP)에는 부서번호가 존재하는 경우를 말함

3. ALTER TABLE

- ALTER TABLE문을 통해 테이블을 변경할 수 있음

- **테이블명 변경**, 컬럼 추가, 변경, 삭제 등을 할 수 있음

1) 테이블명 변경

- 테이블명 변경은 ALTER TABLE~ RENAME TO

ALTER TABLE CAMALL **RENAME TO** NEW\_CAMALL

2) 컬럼 추가

- ALTER TABLE~ ADD

**ALTER TABLE** CAMALL **ADD** (age number(2) default 1) ;

3) 컬럼 변경

- ALTER TABLE~ MODIFY : 데이터 타입 변경 또는 데이터 길이 변경

- 컬럼 변경 시 제약조건을 설정할 수 있음

- 단, 데이터 타입 변경 시에는 기존 데이터를 삭제해야 함

**ALTER TABLE** CAMALL **MODIFY** (ename varchar2(40) not null);

4) 컬럼 삭제

- 컬럼에 대한 삭제는 ALTER TABLE~ DROP COLUMN문으로 삭제

**ALTER TABLE** CAMALL **DROP COLUMN** age ;

5) 컬럼명 변경

- ALTER TABLE~ RENAME COLUMN ~ TO ~

**ALTER TABLE** CAMALL **RENAME COLUMN** cust **TO** new\_cust

4. DROP TABLE(테이블 삭제)

- 테이블 삭제는 DROP TABLE문을 사용하여 삭제할 수 있다

- 테이블의 ***구조와 데이터를 모두 삭제*** 한다

**DROP TABLE** CAMALL ;

- DROP TABLE 에서 ‘CASCADE CONSTRAINT’ 옵션을 사용할 수 있다

- CASCADE CONSTRAINT 옵션은 해당 테이블의 데이터를 외래키로 참조한   
 슬레이브 테이블과 관련된 제약사항도 삭제할 때 사용된다

DROP TABLE CAMALL **CASCADE CONSTRAINT** ; /\*참조된 제약사항까지도 **모두 삭제**\*/

5. 뷰(View) 생성과 삭제

1) 뷰의 생성

- 뷰 란, 테이블로부터 유도된 가상의 테이블이다

- 실제 데이터를 가지고 있지 않고, 테이블을 참조해서 원하는 컬럼만 조회할 수 있게 함

- 뷰는 데이터 딕셔너리(Data Dictionary)에 SQL문 형태로 저장하되 실행 시에 참조된다

★ 뷰View 의 특정   
 ① **참조한 테이블이 변경되면 뷰도 변경**된다   
 ② 뷰의 검색은 참조한 테이블과 동일하게 할 수 있지만,   
 **뷰에 대한 입력/ 수정/ 삭제에는 제약**이 있다  
 ③ 특정 컬럼만 조회시켜서 **보안성을 향상**시킨다  
 ④ 한 번 생성된 뷰는 변경할 수 없고, 변경을 원하는 삭제 후 **재생성**해야 한다  
 ⑤ ALTER문을 사용해서 뷰를 변경할 수 **없다**

**CREATE VIEW** T\_ emp AS

SELECT \* FROM EMP ;

- 뷰의 조회는 SELECT 문을 사용해서 일반 테이블처럼 조회한다

SELECT \* FROM t\_emp ;

2) 뷰의 삭제

DROP VIEW

- 뷰를 삭제했다고 해서 참조했던 테이블이 삭제되지는 않는다

3) 뷰의 장점과 단점

|  |  |
| --- | --- |
| **장점** | **단점** |
| - 특정 컬럼만 조회 🡪 **보안기능**이있음  - **데이터 관리**가 간단함  - **SELECT 문이 간단**해짐  - 하나의 테이블에 여러 개의 뷰 생성가능 | - 뷰는 독자적인 인덱스를 만들 수 없다  - 삽입, 수정, 삭제 연산이 제약됨  - **데이터 구조를 변경할 수 없음** |

**[Quiz]**

**다음 중 뷰(View)에 대한 설명으로 올바르지 않은 것은?**

① 데이터 관리가 간단하다  
② 개발자의 SELECT문이 간소해진다.   
③ 보안 기능은 제공되지 않는다.  
④ 데이터 구조를 변경할 수는 없다.

**03. DML (Data Manipulation Language)**

1. INSERT

1)

INSERT문은 테이블에 데이터를 입력하는 DML문임

INSERT INTO table(col1, col2,…) VALUES(expression1, expression2, …) ;

INSERT INTO CAMALL(EMPNO, ENAME) VALUES(1000, ‘김조이’) ;

🡪 특정 테이블의 모든 컬럼에 대한 데이터를 삽입하는 경우에는 컬럼명 생략 가능

- 주의사항 : INSERT문을 실행했다고 ***데이터 파일에 저장되는 것은 아님!*** 최종적으로 데이터를 저장하려면 TCL문인 COMMIT을 실행해야 한다.

- 만약 Auto COMMIT(set auto commit on)으로 설정된 경우에는 COMMIT을 실행하지 않아도  
 바로 저장됨

2) SELECT문으로 입력하기

- SELECT문을 사용하여 데이터를 조회해서 해당 테이블에 바로 삽입 가능

(단, 입력되는 테이블은 사전에 생성되어 있어야 함)

**INSERT INTO** DEPT\_TEST **SELECT \* FROM** DEPT ;

3) Nologging 사용

- 데이터베이스에 데이터를 입력하면 로그파일(Log file)에 그 정보를 기록한다

- Check point라는 이벤트가 발생하면 로그파일의 데이터를 데이터 파일에 저장한다.

- Nologging 옵션은 로그파일의 기록을 최소화시켜서 **입력 시 성능을 향상**시키는 방법

- Nologging 옵션으 **Buffer Cache라는 메모리 영역을 생략하고 기록한다**

ALTER TABLE DEPT **NOLOGGING** ;

2. UPDATE

- 입력된 데이터의 값을 수정할 때 사용한다.

- 주의사항 : 데이터를 수정할 때 조건절에서 검색되는 행들이 모두 수정된다

UPDATE EMP   
SET ENAME = ‘조조’  
WHERE EMPNO = 100 ;

🡪 EMPNO가 100번인 직원이 2명이면 2명 모두 ENAME이 ‘조조가 됨

3. DELETE

- 원하는 조건을 검색해서 해당되는 행을 삭제

- DELETE문에 조건문을 입력하지 않으면 모든 데이터가 삭제된다

- DELETE문으로 데이터를 삭제한다고 해서 테이블의 용량이 초기화되지는 않는다.

DELETE FROM EMP

WHERE EMPNO = 100 ;

|  |  |
| --- | --- |
| **DELETE FROM 테이블명;** | **TRUNCATE TABLE 테이블명;** |
| - 테이블의 모든 데이터를 삭제한다  - 데이터가 삭제되어도 테이블의 용량은  ***감소하지 않는다*** | - 테이블의 모든 데이터를 삭제한다(=)  - 데이터가 삭제되면  ***테이블의 용량은 초기화된다*** |

4. SELECT

1) SELECT문 사용

- 테이블에 입력된 데이터를 조회하기 위해 SELECT문 사용

- 특정 컬럼이나 특정 행만을 조회할 수 있다.

SELECT \*   
FROM CAMALL   
 WHERE MBR\_SN = 10001 ;

2) ORDER BY를 사용한 정렬

- SELECT문을 사용할 때 ORDER BY를 같이 사용할 수 있다.

- ORDER BY는 데이터를 **오름차순(ASCE) 혹은 내림차순(DESC)**으로 출력한다.

- ORDER BY가 정렬을 하는 시점은 모든 실행이 끝난 후에 데이터를 출력해 주기 바로 전이다.

- ORDER BY는 정렬을 하기 때문에 DB메모리를 많이 사용하게 된다.  
 🡪 대량의 데이터를 정렬학 되면 성능저하가 발생한다.

- 정렬을 회피하기 위해 인덱스(INDEX)를 생성할 때 사용자가 원하는 형태로 오름차순 혹은  
 내림차순으로 생성해야 한다.

- 특별한 지정이 없으면 ORDER BY는 오름차순으로 정렬한다.

SELECT \* FROM EMP  
 ORDER BY ENAME, SAL DESC ; 🡪 ENAME 은 오름차순, SAL은 내림차순으로 정렬됨

**[Quiz]**

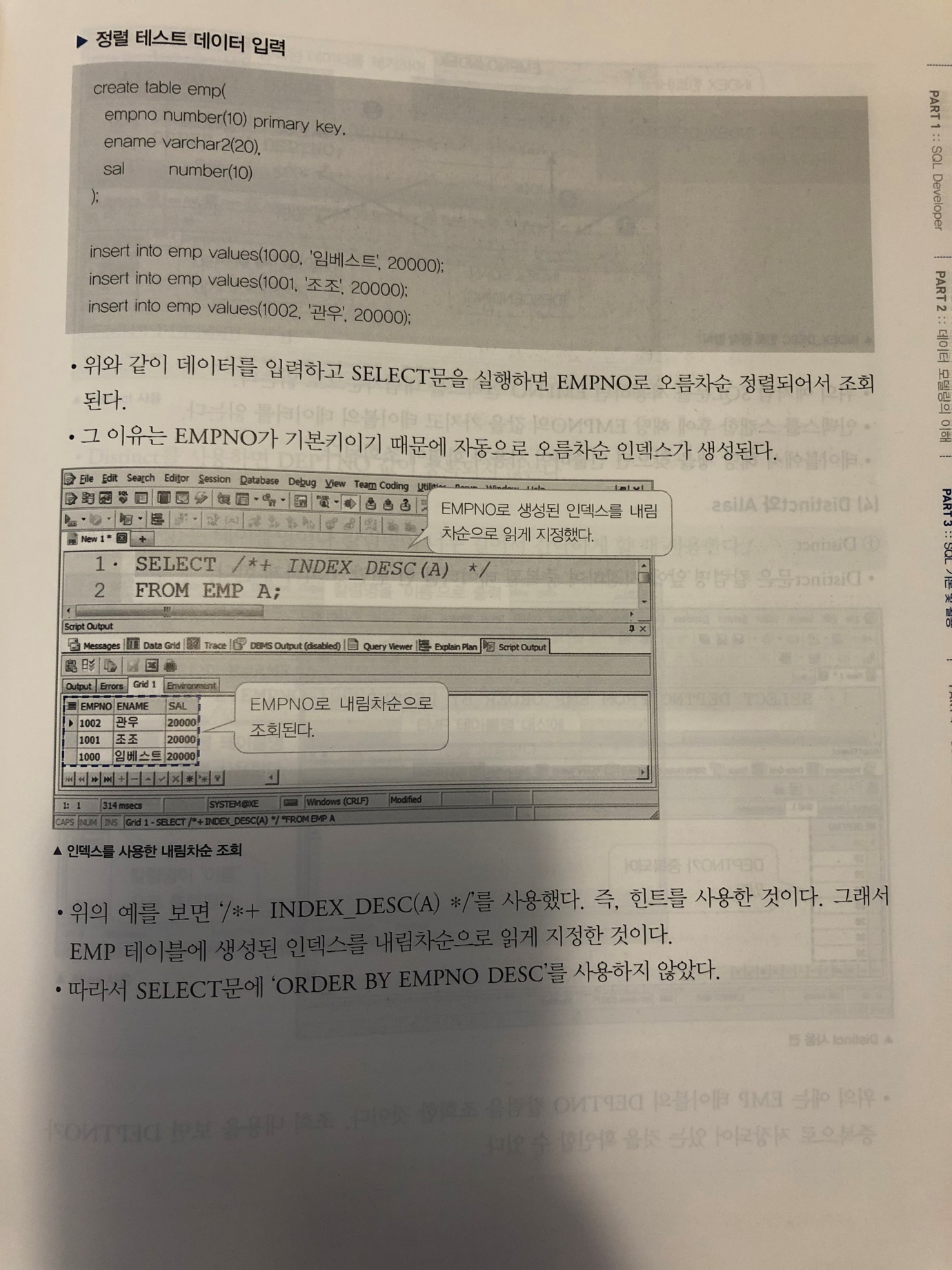
**SELECT문의 구문 중 가장 늦게 실행되는 것은?**

① ORDER BY   
② SELECT   
③ FROM  
④ WHERE

3) INDEX를 사용한 정렬 회피

- 정렬은 ORACLE DB에 부하를 주므로, 인덱스를 사용해서 Order by를 회피할 수 있음

- INDEX 를 먼저 스캔한 후 해당 EMPNO의 값을 가지고 EMP테이블을의 데이터를 읽고,  
 해당 행을 찾으면 인출하여 사용하 화면에 나타낸다.



4) DISTINCT 와 Alias

(1) Distinct

- Distinct문은 컬럼명 앞에 지정하여 중복된 데이터를 한 번만 조회하게 한다.

(2) Alias

- Alias(별칭)은 테이블명이나 컬럼명이 너무 길어서 간략하게 할 때 사용한다.

SELECT ENAME AS “이름”   
 FROM EMP\_3456 a  
 WHERE a.EMPNO = 1000 ;

**[Quiz]**

**DEPT 테이블에서 DEPTNO 값이 10, 10, 20, 30이 있을 때 다음의 SQL문을 실행하면   
 나오는 건수는?**

**SELECT DISTINCT DEPTNO FROM DEPT ;**

① 1   
② 2   
③ 3  
④ 4

**04. WHERE 절**

1. WHERE문이 사용하는 연산자

- 비교 연산자, 부정 비교 연산자, 논리 연산자, SQL 연산자, 부정 SQL 연산자가 있음

▶ 비교 연산자 : = , < , <= , >, >=

▶ 부정 비교 연산자

|  |  |
| --- | --- |
| **비교 연산자** | **설 명** |
| **!=**  **^=**  **<>**  **NOT 컬럼명 =** | 모두 ‘같지 않은 것’을 조회한다 |
| **NOT 컬럼명 >** | 크지 않은 것을 조회한다 |

▶ 논리 연산자

|  |  |
| --- | --- |
| **비교 연산자** | **설 명** |
| **AND** | 조건을 **모두** 만족해야 참(True)이 된다 |
| **OR** | 조건을 **하나만** 만족해도 참(True)이 된다 |
| **NOT** | 참이면 거짓(False)으로 바꾸고, 거짓이면 참(True)으로 바꾼다. |

▶ SQL 연산자

|  |  |
| --- | --- |
| **비교 연산자** | **설 명** |
| **LIKE ‘%비교 문자열%’** | 비교 문자열이 들어가는 모든 값을 조회한다 |
| **BETWEEN A AND B** | A와 B 사이의 값을 조회한다 |
| **IN (list)** | OR를 의미하며 list값 중에 하나만 일치해도 조회된다 |
| **IS NULL** | NULL 값을 조회한다 |

▶ 부정 SQL 연산자

|  |  |
| --- | --- |
| **비교 연산자** | **설 명** |
| **NOT BETWEEN A AND B** | A와 B 사이에 해당되지 않는 값을 조회 |
| **NOT IN (list)** | List 에 있는 값은 빼고 조회 |
| **IS NOT NULL** | NULL 값이 아닌 것을 조회 |

2. Like문 사용

- Like문은 와일드카드를 사용해서 데이터를 조회할 수 있다

|  |  |
| --- | --- |
| **와일드카드** | **설 명** |
| **%** | - 어떤 문자를 포함한 모든 것을 조회  - 예) ‘조%’ 는 ‘조’로 시작하는 모든 문자를 조회 |
| **\_(underscore)** | 한 개인 단일 문자를 의미 |

**[Quiz]**

**다음 중 SQL문의 의미로 올바른 것은?**

**SELECT \* FROM EMP WHERE ENAME LIKE ‘\_TEST’ ;**

① ENAME이 ‘TEST’인 직원을 조회한다  
② 첫글자가 숫자로 시작하는 직원을 조회한다  
③ ENAME이 ‘TEST’와 같지 않은 직원을 조회한다  
④ ENAME이 ‘TEST’이고 앞에 첫글자가 하나 더 있는 직원을 조회한다

3. NULL

- NULL은 모르는 값/ 값의 부재 를 의미한다.

- **NULL과 숫자 혹은 날짜를 더하면 NULL이 된다**.

- NULL과 어떤 값을 비교할 때, ‘알수 없음’이 반환된다.

- NULL 관련 함수

|  |  |
| --- | --- |
| **NULL 함수** | **설 명** |
| **NVL** | - NULL이면 다른 값으로 바꾸는 함수  - NVL(MGR, 0) 은 MGR컬럼이 NULL이면 0으로 바꾼다는 의미 |
| **NVL2** | - NVL함수와 DECODE 함수를 하나로 만든 것인다  - NVL2(MGR, 1, 0) 은 MGR컬럼이 NULL이 아니면 1,  NULL이면 0을 반환 |
| **NULLIF** | - 두 개의 값이 같으면 NULL을, 같지 않으면 첫번째 값을 반환한다  - NULLIF(col1, col2) 에서 col1과 col2가 같으면 NULL을, 같지 않으면 col1을 반환한다 |
| **COALESCE** | - NULL이 아닌 최초의 인자값을 반환한다  - COALESCE(col1, col2, col3, ..) 은 col1이 null이 아니면 col1을 반환한다 |

**[Quiz]**

**다음 중 null 관련 함수에 대한 설명으로 올바르지 않은 것은?**

① NVL함수는 컬럼에 NULL값이 있으면 지정된 다른 값으로 변경한다  
② NVL2 함수는 컬럼값이 NULL이 아니면 두 번째 컬럼값, NULL이면 세번째 컬럼값을   
 반환할 때 사용한다  
③ NULLIF 함수는 두 개의 컬럼값이 같으면 NULL을 되돌리고 같지 않으면 0을 되돌린다  
④ COALESCE함수는 컬럼값이 NULL이 아니면 첫번 째 컬럼을 반환한다

4. GROUP BY

- GROUP BY는 테이블에서 소규모 행을 그룹화하여 합계, 평균 등을 계산할 수 있다.

- HAVING구에 조건문을 사용한다.

- 집계함수의 종류에는 COUNT(), SUM(), AVG( ), MAX( ), MIN( ), STDDEV( ), VARIAN( )

등이 있다.

5. SELECT문의 실행 순서

- FROM > WHERE > GROUP BY > HAVING > SELECT > ORDER BY

6. 명시적(Explicit) 형변환과 암시적(Implicit) 형변환

- 형변환이란 두 개의 데이터의 데이터 타입(형) 이 일치하도록 변환하는 것

- 숫자와 문자열의 비교, 문자열과 날짜형의 비교

- **명시적 형변환은** 형변환 함수를 사용해서 데이터 타입을 일치시키는 것   
 🡪 **개발자가** SQL을 사용할 때 형변환 함수를 사용해야 함

예) TO\_NUMBER(문자열) , TO\_CHAR(숫자 혹은 날짜, [FORMAT]), TO\_DATE(문자열, FORMAT)

- 암시적 형변환은 개발자가 형변환을 하지 않은 경우 데이터베이스 관리 시스템이   
자동으로 형변환하는 것

- 인덱스 컬럼에 암시적 형변환이 일어나면 인덱스를 사용하지 못하는 경우가 있는데,

이런 경우 **명시적 형변환을 해주면 인덱스를 사용할 수 있다**.

**[Quiz]**

**다음 중 형변환에 대한 설명으로 올바르지 않은 것은?**

① TO\_NUMBER( )함수는 문자열을 숫자로 변환한다  
② TO\_CHAR( )함수는 숫자형과 날짜형을 문자로 변환한다  
③ TO\_DATE 함수는 문자열을 날짜형으로 변환한다  
④ 명시적 형변환은 형변환 함수를 사용하지 않고 데이터베이스 관리 시스템에서   
 내부적으로 형변환을 수행한다