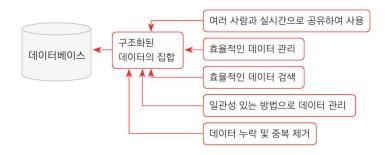
제1장 데이터베이스와 DBMS 개요

1. 데이터베이스

- 1) 데이터, 정보, 지식
 - 데이터는 관찰을 통한 결과를 정량적 혹은 정성적인 값으로 표현한 실제 값
 - 정보는 데이터에 의미이고 지식은 정보를 바탕으로 사물이나 현상에 대한 이해
 - DIKW(Data, Information, Knowledge, Wisdom) 피라미드 4단계를 통한 가치 창조

2) 데이터베이스 개요

- 데이터베이스는 특정 조직의 업무를 수행하는데 필요한 데이터들의 모임
- 데이터베이스는 저장매체에 저장되어 있는 중복을 배제한 통합된 데이터
- 데이터베이스는 응용 시스템들이 공동으로 소유하고 유지하는 자료



3) 데이터베이스 특징

- 실시간 접근성 : 수시적이고 비정형적인 질의에 대하여 실시간 처리에 의한 응답이 가능 - 계속적인 변화 : 새로운 데이터의 삽입, 삭제, 갱신으로 항상 최신의 데이터를 유지

- 동시공용 : 다수의 사용자가 동시에 같은 내용의 데이터를 이용

- 내용에 의한 참조 : 데이터베이스의 데이터는 사용자가 요구하는 데이터 내용으로 데이터를 참조

2. DBMS

- 1) 데이터베이스 관리시스템(DBMS)
 - DBMS는 사용자와 데이터베이스 사이에서 사용자의 요구에 따라 데이터베이스를 관리하는 소프트웨어
 - DBMS는 기존 파일시스템이 갖는 데이터의 중복성의 문제를 해결하기 위해 제안된 시스템
 - Oracle, MySQL, MS-SQL, MaridDB 등



2) 테이블

- 관계형 DBMS(RDBMS)의 핵심 개념은 테이블로 데이터를 효율적으로 저장하기 위한 구조체
- 데이터 저장을 여러 개의 테이블로 나눠 저장함으로써 불필요한 공간의 낭비를 줄이고 효율성을 보장학생 정보 테이블

학번	이름	생년월일	전화번호	집 주소	학년	학기	학과 코드	졸업 여부	•••
16031055	홍길동	971210	010-1111-1111	서울시…	1	2	COM		
12071632	성춘향	940424	010-2222-2222	부산시…	4	2	BNS	졸업	

15022655	박문수	960605	010-3333-3333	광주시…	2	2	MTH		
	테이블		<u>- ፲</u> - 열		1.		행 :		

3. SQL

1) SQL 개요

- SQL(Structured Query Language)은 RDBMS에서 사용되는 언어
- 1970년 후반 IBM이 SEQUEL이라는 이름으로 개발한 관계형 데이터베이스 언어
- 질의 기능 뿐 만 아니라 데이터 구조의 정의, 데이터 조작, 데이터 제어 기능을 갖음

2) SQL 분류

- 2.1) 데이터 정의 언어(DDL:Data Definition Language)
 - DDL은 DB의 구조, 데이터 형식, 접근방식 등 데이터베이스를 구축하거나 수정할 목적으로 사용
 - DDL은 번역한 결과가 데이터사전(Data Dictionary)이라는 특별한 파일에 테이블로 저장
 - DDL의 종류에는 CREATE, ALTER, DROP, TRUNCATE
- 2.2) 데이터 조작 언어(DML:Data Manipulation Language)
 - DML은 사용자로 하여금 데이터를 처리할 수 있게 하는 언어
 - DML의 종류에는 SELECT, INSERT, UPDATE, DELETE
- 2.3) 데이터 제어 언어(DCL:Data Control Language)
 - DCL은 데이터를 보호하고 데이터를 관리하는 목적으로 사용하는 언어
 - DCL의 종류에는 GRANT, REVOKE

4. Oracle

- 1) Oracle 개요
 - 오라클 데이터베이스는 1977년 Larry Ellison이 개발한 DBMS
 - 대표적인 상용 DBMS로 높은 라이센스 비용으로 대기업 위주로 사용
 - 다양한 오프소스 DBMS의 등장으로 과거에 비해 시장 점유율 감소 추세



2) Oracle 데이터베이스

- Oracle DB는 최초 설치 시 지정된 DB(orcl)로 고정이며, 추가적인 DB 생성은 지원하지 않음
- Oracle은 사용자 생성이 DB 생성으로 취급되며 사용자 중심으로 테이블을 관리
- ☞ 실습하기 1-1. 데이터베이스 생성(사용자 생성)

Microsoft Windows [Version 10.0.19045.3324]

(c) Microsoft Corporation. All rights reserved.

C:\Users\chhak0503>sqlplus

SQL*Plus: Release 19.0.0.0.0 - Production on 수 9월 6 14:42:12 2023

Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle. All rights reserved.

사용자명 입력: system 또는 sys as sysdba

비밀번호 입력:

마지막 성공한 로그인 시간: 수 9월 06 2023 14:09:49 +09:00

다음에 접속됨:

Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production Version 19.3.0.0.0

SQL> ALTER SESSION SET "_ORACLE_SCRIPT"=true; 세션이 변경되었습니다.

SQL> CREATE USER scott IDENTIFIED BY tiger; 사용자가 생성되었습니다.

SQL> GRANT CONNECT, RESOURCE TO scott; 권한이 부여되었습니다.

SQL> GRANT UNLIMITED TABLESPACE TO scott; 권한이 부여되었습니다.

SQL> exit

3) Oracle 자료형

- Oracle 자료형은 크게 숫자, 문자, 날짜/시간 데이터 형식 3가지
- VARCHAR2 자료형은 영문/숫자는 1Byte 크기, 한글은 3Byte 크기
- NVARCHAR2 자료형은 영문/숫자/한글(UTF-8) 1Byte 크기

자료형	설명
NUMBER(p, s)	최대 38 자릿수의 숫자 데이터, p는 최대 유효숫자 자릿수, s는 소수점 자릿수
CHAR(s)	고정 길이 문자열 데이터, 1byte ~ 4000byte
VARCHAR2(s)	가변 길이 문자열 데이터, 1byte ~ 4000byte
NVARCHAR2(s)	가변 길이 국가별 문자열 데이터, 1byte ~ 4000byte
DATE	날짜/시간 데이터
CLOB	Character Large Object, 최대 크기 4GB 대용량 텍스트 데이터 저장
BLOB	Binary Large Object, 최대 크기 4GB 대용량 이진 데이터 저장

〈숫자형 자료형 예시〉

타입	입력 데이터	저장 데이터	설명
NUMBER	- 1234.56	- 1234.56	- 이상 없음
	- 1234.567	- 1234.567	- 이상 없음
NUMBER(3)	- 123	- 123	- 이상 없음
	- 123.4	- 123	- 반올림
	- 1234	- 오류	- 자릿수 부족
NUMBER(6, 2)	- 1234.56	- 1234.56	- 이상 없음
	- 1234.567	- 1234.57	- 반올림
	- 12345.6	- 오류	- 자릿수 부족

〈문자형 자료형 예시〉

타입	입력 데이터	저장 데이터	설명
CHAR(10)	- 'Apple'	- 'Apple '	- 10 byte
	- '사과'	- '사과 '	- 10 byte
VARCHAR2(10)	- 'Apple'	- 'Apple'	- 5 byte
	- '사과'	- '사과'	- 6 byte
NVARCHAR2(10)	- 'Apple'	- 'Apple'	- 5 byte
	- '사과'	- '사과'	- 2 byte

```
☞ 실습하기 1-2. NUMBER 자료형 이해
> CREATE TABLE TYPE_TEST_NUMBER (
   num1 NUMBER,
   num2 NUMBER(2),
   num3 NUMBER(3,1),
   num4 NUMBER(4,2),
   num5 NUMBER(5,6)
);
//num1 NUMBER
insert into TYPE_TEST_NUMBER (num1) values (1);
insert into TYPE_TEST_NUMBER (num1) values (123);
insert into TYPE_TEST_NUMBER (num1) values (123.74);
insert into TYPE_TEST_NUMBER (num1) values (123.12345);
//num2 NUMBER(2)
insert into TYPE_TEST_NUMBER (num2) values (12);
insert into TYPE_TEST_NUMBER (num2) values (123);
insert into TYPE_TEST_NUMBER (num2) values (1.2);
insert into TYPE_TEST_NUMBER (num2) values (1.23);
insert into TYPE_TEST_NUMBER (num2) values (12.34567);
insert into TYPE_TEST_NUMBER (num2) values (12.56789);
insert into TYPE_TEST_NUMBER (num2) values (123.56789);
//num3 NUMBER(3,1)
insert into TYPE_TEST_NUMBER (num3) values (12);
insert into TYPE_TEST_NUMBER (num3) values (123);
insert into TYPE_TEST_NUMBER (num3) values (12.1);
insert into TYPE_TEST_NUMBER (num3) values (12.1234);
insert into TYPE_TEST_NUMBER (num3) values (12.56789);
insert into TYPE_TEST_NUMBER (num3) values (123.56789);
```

```
☞ 실습하기 1-3. 문자형 자료형 이해
> CREATE TABLE TYPE_TEST_CHAR (
   char1 CHAR(1),
   char2 CHAR(2),
   char3 CHAR(3),
   vchar1 VARCHAR2(1),
   vchar2 VARCHAR2(2),
   vchar3 VARCHAR2(3),
   nvchar1 NVARCHAR2(1),
   nvchar2 NVARCHAR2(2),
   nvchar3 NVARCHAR2(3)
);
//CHAR
insert into TYPE_TEST_CHAR (char1) values ('A');
insert into TYPE_TEST_CHAR (char1) values ('가');
//VARCHAR2
insert into TYPE_TEST_CHAR (vchar1) values ('A');
insert into TYPE_TEST_CHAR (vchar2) values ('AB');
insert into TYPE_TEST_CHAR (vchar3) values ('가');
//NVARCHAR2
insert into TYPE_TEST_CHAR (nvchar1) values ('A');
insert into TYPE_TEST_CHAR (nvchar2) values ('AB');
insert into TYPE_TEST_CHAR (nvchar3) values ('가나다');
```

제2장 SQL 기본

1. 테이블

1) 테이블 생성 및 삭제

테이블 생성

```
CREATE TABLE 테이블명 (칼럼명1 자료형1, 칼럼명2 자료형2, ...);
```

※ Oracle은 테이블명, 컬럼명 모두 대소문자 구분을 하지 않음, 구분을 할때는 큰따옴표 처리

테이블 제거

```
DROP TABLE 테이블명;
```

```
☞ 실습하기 1-1. 테이블 생성

> CREATE TABLE USER1 (
    ID VARCHAR2(20),
    NAME VARCHAR2(20),
    HP CHAR(13),
    AGE NUMBER
    );

> DROP TABLE USER1;
```

2) 데이터 추가(INSERT)

데이터 추가

```
INSERT INTO 테이블명 VALUES (데이터1, 데이터2, 데이터3 ...);
```

칼럼명을 지정해서 데이터 추가

```
INSERT INTO 테이블명 (칼럼명1, 칼럼명2 ... ) VALUES (데이터1, 데이터2 ...);
```

※ Oracle은 INSERT INTO SET 구문을 지원 안함

```
☞ 실습하기 1-2

> INSERT INTO USER1 VALUES ('A101', '김유신', '010-1234-1111', 25);

> INSERT INTO USER1 VALUES ('A102', '김춘추', '010-1234-2222', 23);

> INSERT INTO USER1 VALUES ('A103', '장보고', '010-1234-3333', 32);

> INSERT INTO USER1 (id, name, age) VALUES ('A104', '강감찬', 45);

> INSERT INTO USER1 (id, name, hp) VALUES ('A105', '이순신', '010-1234-5555');
```

3) 데이터 조회(SELECT)

데이터 조회

```
SELECT 칼럼명1, 칼럼명2 ... FROM 테이블명;
SELECT 칼럼명1, 칼럼명2 ... FROM 테이블명 WHERE 조건;
```

모든 데이터 조회

```
SELECT * FROM 테이블명;
SELECT * FROM 데이터베이스명.테이블명;
```

- ☞ 실습하기 1-3
- > SELECT * FROM USER1;
- > SELECT * FROM USER1 WHERE id='A101';
- > SELECT * FROM USER1 WHERE name='김춘추';
- > SELECT * FROM USER1 WHERE age > 30;
- > SELECT id, name, age FROM USER1;

4) 데이터 수정(UPDATE)

데이터 수정

UPDATE 테이블명 SET 칼럼명1=데이터1, 칼럼명2=데이터2 ...;

조건에 일치하는 레코드만 수정

UPDATE 테이블명 SET 칼럼명1=데이터1, 칼럼명2=데이터2 ... WHERE 조건;

- ☞ 실습하기 1-4
- > UPDATE User1 SET hp='010-1234-4444' WHERE id='A104';
- > UPDATE User1 SET age=51 WHERE id='A105';
- > UPDATE User1 SET hp='010-1234-1001', age=27 WHERE id='A101';

5) 데이터 삭제(DELETE)

데이터 삭제

DELETE FROM 테이블명 WHERE 조건;

- ☞ 실습하기 1-5
- > DELETE FROM User1 WHERE id='A101';
- > DELETE FROM User1 WHERE id='A102' AND age=25;
- > DELETE FROM User1 WHERE age >= 30;

2. 제약조건

- 1) 데이터 무결성과 제약조건
 - 데이터 무결성(Data Integrity)은 데이터베이스에 저장되는 데이터의 정확성과 일관성을 보장하는 의미
 - 제약조건(Constraint)은 이러한 데이터 무결성을 지키기 위한 제한된 조건을 의미
 - 제약조건을 지정한 컬럼에 제약 조건에 부합한 데이터만 저장함으로써 데이터 무결성을 보장

〈데이터 무결성〉

종류	설명
도메인 무결성	- 컬럼에 저장되는 데이터의 적정 여부를 확인 - 자료형, 원자값, NULL 여부 등 정해놓은 범위를 만족하는 데이터
개체 무결성	- 테이블에 저장되는 데이터를 구분할 수 있는 유일한 식별값이 반드시 존재 - 식별값은 중복될 수 없고 NULL이 되어서는 안됨
참조 무결성	- 테이블의 외래키 값은 참조하는 테이블의 기본키로서 존재

〈제약조건〉

종류	설명
PRIMARY KEY	- 지정한 컬럼에 유일한 식별값을 가짐 - 데이터 중복 허용 안함, NULL값 허용 안함, 테이블에 하나만 지정
UNIQUE	- 지정한 컬럼에 유일한 값 - 데이터 중복 허용 안함, NULL값 허용
FOREIGN KEY	- 다른 테이블의 컬럼을 참조하여 데이터 입력
NOT NULL	- 지정한 컬럼에 NULL값 허용 안함, 데이터 중복은 허용
DEFAULT	- 데이터를 입력하지 않아도 자동으로 입력되는 기본값 - 모든 컬럼의 DEFAULT는 NULL
CHECK	- 설정한 조건식을 만족하는 데이터만 입력

2) 기본키(Primary Key)

- 기본키(PK)는 테이블에 존재하는 데이터를 구분하는 식별값

테이블을 생성할 때 기본키 지정하기

CREATE TABLE 테이블명 (칼럼명 자료형 PRIMARY KEY...);

```
실습하기 2-1

> CREATE TABLE USER2 (

ID VARCHAR2(20) PRIMARY KEY,

NAME VARCHAR2(20),

HP CHAR(13),

AGE NUMBER(2)

);
```

3) 고유키(Unique)

- UNIQUE 제약조건은 '중복되지 않은 유일한 값'을 입력해야 하는 제약조건
- 기본키와 달리 UNIQUE는 NULL 값을 허용

테이블을 생성할 때 고유키 지정하기

CREATE TABLE 테이블명 (칼럼명 자료형 UNIQUE...);

```
☞ 실습하기 2-2
```

- 4) 외래키(Foreign Key)
 - 외래키를 가진 테이블이 자식 테이블이라고 하고 외래키 값을 제공하는 테이블을 부모 테이블
 - 외래키 값은 NULL이거나 부모 테이블의 기본키 값과 동일(참조 무결성)

테이블 외래키 지정하기

CREATE TABLE 테이블명 (칼럼명 자료형 FOREIGN KEY (column) REFERENCES Parent (column));

```
☞ 실습하기 2-3
> CREATE TABLE PARENT (
    PID
            VARCHAR2(20) PRIMARY KEY,
    NAME
            VARCHAR2(20),
    ΗP
            CHAR(13) UNIQUE
    );
> CREATE TABLE CHILD (
    CID
             VARCHAR2(20) PRIMARY KEY,
    NAME
             VARCHAR2(20),
    HP
             CHAR(13) UNIQUE,
    PARENT VARCHAR(20),
    FOREIGN KEY (PARENT) REFERENCES PARENT (PID)
   );
> INSERT INTO PARENT VALUES ('P101', '김서현', '010-1234-1001');
> INSERT INTO PARENT VALUES ('P102', '이성계', '010-1234-1002');
> INSERT INTO PARENT VALUES ('P103', '신사임당', '010-1234-1003');
> INSERT INTO CHILD VALUES ('C101', '김유신', '010-1234-2001', 'P101');
> INSERT INTO CHILD VALUES ('C102', '이방우', '010-1234-2002', 'P102');
> INSERT INTO CHILD VALUES ('C103', '이방원', '010-1234-2003', 'P102');
> INSERT INTO CHILD VALUES ('C104', '0|0|', '010-1234-2004', 'P103');
```

5) DEFAULT와 NOT NULL

- DEFAULT는 값을 입력하지 않아도 자동으로 입력되는 기본값

```
- 모든 컬럼의 DEFAULT는 NULL이고 반드시 데이터 입력하는 컬럼은 NOT NULL 명시
 ☞ 실습하기 2-4
 > CREATE TABLE USER4 (
     NAME
             VARCHAR2(20) NOT NULL,
     GENDER CHAR(1) NOT NULL,
     AGE
             INT DEFAULT 1,
     ADDR
             VARCHAR2(255)
     );
 > INSERT INTO USER4 VALUES ('김유신', 'M', 23, '김해시');
 > INSERT INTO USER4 VALUES ('김춘추', 'M', 21, '경주시');
 > INSERT INTO USER4 (NAME, GENDER, ADDR) VALUES ('신사임당', 'F', '강릉시');
 > INSERT INTO USER4 (NAME, GENDER) VALUES ('이순신', 'M');
 > INSERT INTO USER4 (NAME, AGE) VALUES ('정약용', 33);
6) CHECK
 - CHECK 제약조건은 입력되는 데이터를 검사
 - CHECK 제약조건은 저장할 수 있는 값의 범위 또는 범주를 정의
 ☞ 실습하기 2-5
 > CREATE TABLE USER5 (
             VARCHAR2(20) NOT NULL,
              CHAR(1) NOT NULL CHECK(GENDER IN('M', 'F')),
     AGE
              INT DEFAULT 1 CHECK(AGE > 0 AND AGE < 100),
     ADDR
             VARCHAR2(255)
     );
 > INSERT INTO USER5 VALUES ('김유신', 'M', 23, '김해시');
 > INSERT INTO USER5 VALUES ('김춘추', 'M', 21, '경주시');
```

> INSERT INTO USER5 (NAME, GENDER, ADDR) VALUES ('신사임당', 'F', '강릉시');

> INSERT INTO USER5 (NAME, GENDER) VALUES ('이순신', 'M'); > INSERT INTO USER5 (NAME, AGE) VALUES ('정약용', 33);

3. 데이터베이스 개체

- 오라클 데이터베이스는 데이터 보관 및 관리를 위한 여러 기능을 위해 개체 제공
- 테이블은 오라클에서 가장 많이 사용하는 개체이며 그 외 데이터 사전, 인덱스, 뷰, 시퀀스 개체 등
- 1) 데이터 사전(Data Dictionary)
 - 오라클 데이터베이스는 사용자 테이블과 데이터 사전으로 구성
 - 데이터 사전은 데이터베이스를 구성하고 운영하는데 필요한 모든 정보를 저장하는 특수 테이블

〈데이터 사전〉

접두어	설명		
USER_XXX	현재 데이터베이스에 접속한 사용자가 소유한 객체 정보		
ALL_XXX	현재 데이터베이스에 접속한 사용자가 소유한 모든 객체 정보		
DBA_XXX	데이터베이스 관리를 위한 정보(SYSTEM, SYS 사용자만 조회 가능)		
V\$_XXX	데이터베이스 성능 관련 정보		

- ☞ 실습하기 3-1. 데이터 사전 조회(system 접속)
- # 전체 사전 조회
- > SELECT * FROM DIC;
- # 테이블 조회(현재 사용자 기준)
- > SELECT TABLE_NAME FROM USER_TABLES;
- # 전체 테이블 조회(현재 사용자 기준)
- > SELECT OWNER, TABLE NAME FROM ALL TABLES;
- # 전체 테이블 조회(system 관리자만 가능)
- > SELECT * FROM DBA_TABLES;
- # 전체 사용자 조회(system 관리자만 가능)
- > SELECT * FROM DBA_USERS;

2) 인덱스(Index)

- 인덱스는 데이터 검색 성능 향상을 위해 테이블 특정 컬럼에 설정하는 개체
- 테이블 특정 컬럼에 설정한 데이터의 주소, 즉 데이터의 위치 정보를 목록 형태로 만들어 놓은 구조체
- 인덱스가 반드시 성능 향상으로 이어지기보다는 데이터 종류와 SQL 등 많은 요소를 고려한 설정
- ☞ 실습하기 3-2. 인덱스 조회/생성/삭제
- # 현재 사용자 인덱스 조회
- > SELECT * FROM USER_INDEXES;
- # 현재 사용자 인덱스 정보 조회
- > SELECT * FROM USER_IND_COLUMNS;
- # 인덱스 생성
- > CREATE INDEX IDX_USER1_UID ON User1(`uid`);
- > SELECT * FROM USER IND COLUMNS;
- # 인덱스 삭제
- > DROP INDEX IDX_USER1_UID;
- > SELECT * FROM USER_IND_COLUMNS;
- 3) 뷰(View)
 - 뷰는 하나 이상의 테이블을 조회하는 SELECT를 저장한 가상 테이블 개체
 - 뷰는 테이블의 특정 컬럼의 노출을 피하거나 자주 사용하는 복잡한 SELECT를 간편하게 사용할 목적
 - ☞ 실습하기 3-3. 뷰 생성 권한 할당

```
C:\Users\chhak0503>sqlplus system/1234
다음에 접속됨:
SQL> GRANT CREATE VIEW TO 아이디;
SQL> exit
```

- ☞ 실습하기 3-4. 뷰 생성/조회/삭제
- # 뷰 생성
- > CREATE VIEW VW_USER1 AS (SELECT NAME, HP, AGE FROM USER1);
- > CREATE VIEW VW USER2 AGE UNDER30 AS (SELECT * FROM USER2 WHERE AGE < 30);
- > SELECT * FROM USER_VIEWS;
- # 뷰 조회
- > SELECT * FROM VW USER1;
- > SELECT * FROM VW USER2 AGE UNDER30;
- # 뷰 삭제
- > DROP VIEW VW_USER1;
- > DROP VIEW VW USER2 AGE UNDER30;

4) 시퀀스(Sequence)

- 시퀀스는 특정 규칙에 맞는 연속 숫자를 생성하는 개체
- 시퀀스로 생성되는 연속적인 번호를 식별값으로 활용하여 지속적이고 효율적인 데이터 관리

```
☞ 실습하기 3-5. 시퀀스 적용 테이블 생성
```

```
> CREATE TABLE USER6 (
SEQ NUMBER PRIMARY KEY,
NAME VARCHAR2(20),
GENDER CHAR(1),
AGE NUMBER,
ADDR VARCHAR2(255)
);
```

- ☞ 실습하기 3-6. 시퀀스 생성
- > CREATE SEQUENCE SEQ USER6 INCREMENT BY 1 START WITH 1;
- ☞ 실습하기 3-7. 시퀀스값 입력
- > INSERT INTO USER6 VALUES (SEQ_USER5.NEXTVAL, '김유신', 'M', 25, '김해시');
- > INSERT INTO USER6 VALUES (SEQ_USER5.NEXTVAL, '김춘추', 'M', 23, '경주시');
- > INSERT INTO USER6 VALUES (SEQ_USER5.NEXTVAL, '신사임당', 'F', 27, '강릉시');

4. 사용자와 권한

- 1) 사용자 생성
 - 오라클 데이터베이스는 스키마와 사용자를 구별하지 않고 사용자 중심으로 스키마 생성
 - 사용자 생성은 일반 사용자가 아닌 데이터베이스 관리 권한을 가진 SYS, SYSTEM 사용자만 생성 가능
 - ☞ 실습하기 4-1. 사용자 생성(SYSTEM 접속)
 - // 사용자 생성
 - > ALTER SESSION SET " ORACLE SCRIPT"=true;
 - > CREATE USER 아이디 identified by 비밀번호;
 - ☞ 실습하기 4-2. 사용자 조회(SYSTEM 접속)
 - // 전체 사용자 조회
 - > SELECT * FROM ALL_USERS;
 - // 특정 사용자 조회
 - > SELECT * FROM ALL_USERS WHERE USERNAME='아이디(대문자)';
 - ☞ 실습하기 4-3. 사용자 변경(SYSTEM 접속)
 - // 사용자 비밀번호 변경
 - > ALTER USER 아이디 IDENTIFIED BY 변경비밀번호;
 - // 사용자 삭제
 - > DROP USER 아이디;
 - // 사용자와 해당 사용자 객체(테이블 등) 모두 삭제
 - > DROP USER 아이디 CASCADE;
- 2) 권한 관리
 - 데이터베이스 관리를 위해 개별적인 권한을 부여
 - 여러 종류의 권한을 그룹화 해놓은 롤(role)을 사용해 여러 권한을 한 번에 부여하고 관리 효율을 높임

〈주요 Role〉

Role	설명
CONNECT	데이터베이스 접속에 필요한 여러 권한을 부여하는 롤
RESOURCE	테이블 및 시퀀스 등 여러 객체를 생성할 수 있는 기본 권한을 부여하는 롤
DBA	데이터베이스를 관리하는 시스템 권한을 모두 갖는 롤

- ☞ 실습하기 4-4. Role 부여
- // 접속 및 생성 권한 부여
- > GRANT CONNECT, RESOURCE TO 아이디;
- // 테이블 스페이스(테이블 파일 생성 공간) 할당량 권한 부여
- > GRANT UNLIMITED TABLESPACE TO 아이디;

제3장 SQL 실습

1. 데이터 조회

- 1) 실습 테이블 생성 및 데이터 입력
 - 전통적인 오라클 데이터베이스의 SCOTT 계정 테이블을 이용한 SQL 실습
 - 오라클은 테이블명과 컬럼명을 대문자로 권장

〈부서 테이블〉

열	자료형	설명
DEPTNO	2자리 숫자	부서 번호
DNAME	14byte 크기의 가변형 문자	부서 이름
LOC	13byte 크기의 가변형 문자	부서가 위치한 지역

〈직원 테이블〉

열	자료형	설명
EMPNO	4자리 숫자	직원 번호
ENAME	10byte 크기의 가변형 문자	직원 이름
JOB	9byte 크기의 가변형 문자	직원 직책
MGR	4자리 숫자	직속 상관의 직원 번호
HIREDATE	날짜 데이터	입사일
SAL	2자리 소수점 포함, 7자리 숫자	급여
СОММ	2자리 소수점 포함, 7자리 숫자	COMMISSION, 급여 외 추가 수당
DEPTNO	2자리 숫자	직원이 속한 부서 번호

〈급여등급 테이블〉

열	자료형	설명
GRADE	숫자	급여 등급
LOSAL	숫자	급여 등급의 최소 급여액
HISAL	숫자	급여 등급의 최대 급여액

```
☞ 실습하기 1-1. 실습 테이블 생성
// 부서 테이블
> CREATE TABLE DEPT (
    DEPTNO
             NUMBER(2) CONSTRAINT PK_DEPT PRIMARY KEY,
    DNAME
              VARCHAR2(14),
    LOC
              VARCHAR2(13)
    );
// 직원 테이블
> CREATE TABLE EMP (
    EMPNO
             NUMBER(4) CONSTRAINT PK_EMP PRIMARY KEY,
    ENAME
              VARCHAR2(10),
    JOB
             VARCHAR2(9),
    MGR
              NUMBER(4),
    HIREDATE DATE,
    SAL
              NUMBER(7,2),
    COMM
              NUMBER(7,2),
    DEPTNO
              NUMBER(2) CONSTRAINT FK_DEPTNO REFERENCES DEPT
    );
// 급여 테이블
> CREATE TABLE SALGRADE (
    GRADE
              NUMBER,
    LOSAL
              NUMBER,
    HISAL
              NUMBER
    );
// 보너스 테이블
> CREATE TABLE BONUS (
    ENAME
             VARCHAR2(10),
    JOB
              VARCHAR2(9),
    SAL
              NUMBER,
    COMM
              NUMBER
    );
☞ 실습하기 1-2. 실습 테이블 데이터 입력
// 부서 테이블 데이터 입력
> INSERT INTO DEPT VALUES (10, 'ACCOUNTING', 'NEW YORK');
> INSERT INTO DEPT VALUES (20, 'RESEARCH', 'DALLAS');
> INSERT INTO DEPT VALUES (30, 'SALES', 'CHICAGO');
> INSERT INTO DEPT VALUES (40, 'OPERATIONS', 'BOSTON');
// 직원 테이블 데이터 입력
> INSERT INTO EMP VALUES (7369, 'SMITH', 'CLERK', 7902, '80-12-17', 800, NULL, 20);
> INSERT INTO EMP VALUES (7499, 'ALLEN', 'SALESMAN', 7698, '81-02-20',1600,300,30);
> INSERT INTO EMP VALUES (7521, 'WARD', 'SALESMAN', 7698, '81-02-22', 1250, 500, 30);
```

```
> INSERT INTO EMP VALUES (7566, 'JONES', 'MANAGER', 7839, '81-02-04',2975,NULL,20);
 > INSERT INTO EMP VALUES (7654, 'MARTIN', 'SALESMAN', 7698, '81-09-28',1250,1400,30);
 > INSERT INTO EMP VALUES (7698, 'BLAKE', 'MANAGER', 7839, '81-05-01',2850,NULL,30);
 > INSERT INTO EMP VALUES (7782, 'CLARK', 'MANAGER', 7839, '81-06-09',2450,NULL,10);
 > INSERT INTO EMP VALUES (7788, 'SCOTT', 'ANALYST', 7566, '87-04-19',3000,NULL,20);
 > INSERT INTO EMP VALUES (7839, 'KING', 'PRESIDENT', NULL, '81-11-17',5000,NULL,10);
 > INSERT INTO EMP VALUES (7844, 'TURNER', 'SALESMAN', 7698, '81-09-08',1500,0,30);
 > INSERT INTO EMP VALUES (7876, 'ADAMS', 'CLERK', 7788, '87-05-23',1100,NULL,20);
 > INSERT INTO EMP VALUES (7900, 'JAMES', 'CLERK', 7698, '81-12-03', 950, NULL, 30);
 > INSERT INTO EMP VALUES (7902, 'FORD', 'ANALYST', 7566,
                              to_date('3-12-1981', 'dd-mm-yyyy'), 3000, NULL, 20);
 > INSERT INTO EMP VALUES (7934, 'MILLER', 'CLERK', 7782,
                             to_date('23-1-1982', 'dd-mm-yyyy'), 1300, NULL, 10);
 // 급여 등급 테이블 데이터 입력
 > INSERT INTO SALGRADE VALUES (1, 700, 1200);
 > INSERT INTO SALGRADE VALUES (2, 1201, 1400);
 > INSERT INTO SALGRADE VALUES (3, 1401, 2000);
 > INSERT INTO SALGRADE VALUES (4, 2001, 3000);
 > INSERT INTO SALGRADE VALUES (5, 3001, 9999);
 ☞ 실습하기 1-3. 기본 데이터 조회
 > SELECT * FROM EMP;
 > SELECT EMPNO, ENAME, DEPTNO FROM EMP;
2) 정렬
 - 조회한 데이터의 정렬할 때 ORDER BY절 사용
 - 오름차순(ASC), 내림차순(DESC)를 지정해서 정렬
 ☞ 실습하기 1-4. 데이터 조회 정렬
 > SELECT * FROM EMP ORDER BY SAL;
 > SELECT * FROM EMP ORDER BY SAL DESC;
 > SELECT * FROM EMP ORDER BY DEPTNO ASC, SAL DESC;
3) 중복제거와 별칭
 - DISTINCT는 조회한 데이터의 내용에서 불필요한 중복을 제거
```

- 별칭(Alias)을 지정해서 컬럼명을 알기 쉽게 출력
- ☞ 실습하기 1-5. 중복제거와 별칭 지정
- > SELECT DISTINCT DEPTNO FROM EMP;
- > SELECT DISTINCT JOB, DEPTNO FROM EMP;
- > SELECT ENAME, SAL, SAL*12+COMM, COMM FROM EMP;
- > SELECT ENAME, SAL, SAL*12+COMM AS ANNSAL, COMM FROM EMP;

2. WHERE절과 연산자

```
1) WHERE절과 기본 연산자
 - WHERE절은 데이터를 조회할 때 특정 조건을 기준으로 원하는 데이터를 조회
 - 기본적인 다양한 연산자를 이용해 세밀하게 데이터 조회
 ☞ 실습하기 2-1. 다양한 조건으로 데이터 조회
 > SELECT * FROM EMP WHERE DEPTNO=30;
 > SELECT * FROM EMP WHERE DEPTNO=30 AND JOB='SALESMAN';
 > SELECT * FROM EMP WHERE DEPTNO=30 OR JOB='CLERK';
 > SELECT * FROM EMP WHERE SAL * 12 = 36000;
 > SELECT * FROM EMP WHERE SAL >= 3000;
 > SELECT * FROM EMP WHERE SAL != 3000;
 > SELECT * FROM EMP WHERE SAL <> 3000;
 > SELECT * FROM EMP WHERE NOT SAL = 3000;
 > SELECT * FROM EMP WHERE JOB IN ('MANAGER', 'SALESMAN', 'CLERK');
 > SELECT * FROM EMP WHERE JOB NOT IN ('MANAGER', 'SALESMAN', 'CLERK');
 > SELECT * FROM EMP WHERE SAL >= 2000 AND SAL <= 3000;
 > SELECT * FROM EMP WHERE SAL BETWEEN 2000 AND 3000;
 > SELECT * FROM EMP WHERE SAL NOT BETWEEN 2000 AND 3000;
 > SELECT * FROM EMP WHERE ENAME LIKE 'S%';
 > SELECT * FROM EMP WHERE ENAME LIKE ' L%';
 > SELECT * FROM EMP WHERE ENAME LIKE '%AM%';
 > SELECT * FROM EMP WHERE ENAME NOT LIKE '%AM%';
 > SELECT * FROM EMP WHERE COMM = NULL;
 > SELECT * FROM EMP WHERE COMM IS NULL;
 > SELECT * FROM EMP WHERE COMM IS NOT NULL;
2) 집합 연산자
 - UNION은 두 개 이상의 SELECT문에 사용하는 집합 연산자
 - 집합 연산자를 사용하기 위해 조회하는 컬럼 개수와 컬럼 순서가 모두 일치
 ☞ 실습하기 2-2. 집합 연산자를 이용한 데이터 조회
 > SELECT EMPNO, ENAME, SAL, DEPTNO
      FROM EMP
     WHERE DEPTNO = 10
   UNION
   SELECT EMPNO, ENAME, SAL, DEPTNO
     FROM EMP
     WHERE DEPTNO = 20;
```

3. SQL 함수

```
1) SOL 문자 함수
 - 문자 함수는 문자 데이터를 가공하거나 특정 결과를 출력할 때 사용
 - 오라클 DUAL 더미 테이블 활용
 ☞ 실습하기 3-1. 다양한 SQL 문자 함수 실습
 // LENGTH : 문자 길이
 > SELECT ENAME, LENGTH(ENAME) FROM EMP;
 // UPPER, LOWER : 문자 대소문자
 > SELECT ENAME, UPPER(ENAME), LOWER(ENAME), INITCAP(ENAME) FROM EMP;
 // SUBSTR : 문자 자르기
 > SELECT JOB, SUBSTR(JOB, 1, 2), SUBSTR(JOB, 3, 2), SUBSTR(JOB, 5) FROM EMP;
 // INSTR : 문자 위치
 > SELECT
      INSTR('HELLO ORACLE!', 'L') AS INSTR_1,
     INSTR('HELLO ORACLE!', 'L', -1) AS INSTR_2
   FROM DUAL;
 // REPLACE : 문자 교체
 > SELECT '010-1234-5678', REPLACE('010-1234-5678', '-', '') FROM DUAL;
 // LPAD, RPAD : 문자 채우기
 > SELECT LPAD('Oracle', 10, '#') AS LPAD, RPAD('Oracle', 10, '*') AS RPAD FROM DUAL;
 // CONCAT : 문자 연결
 > SELECT CONCAT(EMPNO, ENAME) FROM EMP WHERE ENAME = 'SCOTT';
 > SELECT EMPNO || ENAME FROM EMP WHERE ENAME = 'SCOTT';
 // TRIM : 문자 공백 제거
 > SELECT
      '[ _Oracle_ ]' AS BEFORE,
      '[' || TRIM('_Oracle_ ') || ']' AS TRIM
   FROM DUAL;
```

```
2) SQL 숫자 함수
 - 오라클에서 제공하는 숫자 함수 중 자주 사용하는 함수 위주로 실습
 - 오라클 DUAL 더미 테이블 활용
 ☞ 실습하기 3-2. 다양한 SQL 숫자 함수 실습
 // SUM : 합계
 > SELECT SUM(SAL) AS 합계 FROM EMP;
 // COUNT : 갯수
 > SELECT COUNT(*) AS 전체갯수 FROM EMP;
 > SELECT COUNT(COMM) AS COMM_갯수 FROM EMP;
 // AVG : 평균
 > SELECT AVG(SAL) AS 평균 FROM EMP;
 > SELECT COUNT(COMM) AS COMM 갯수 FROM EMP;
 // ROUND : 반올림
 > SELECT
     ROUND(123.56) AS ROUND,
     ROUND(123.56, 0) AS ROUND_0,
     ROUND(123.56, 1) AS ROUND_1,
     ROUND(123.56, 2) AS ROUND_2
   FROM DUAL;
 // CEIL, FLOOR : 소수점 올림, 내림
 > SELECT
     CEIL(3.14),
     CEIL(3.78),
     FLOOR(3.14),
     FLOOR(3.78)
    FROM DUAL;
3) SQL 날짜 함수
 - 오라클에서 날짜 데이터(DATE형)는 SYSDATE 함수 사용
 - SQL Developer/도구/환경설정/데이터베이스/NLS - 날짜 형식 YYYY-MM-DD HH24:MI:SS 적용
 ☞ 실습하기 3-3. SQL 날짜 함수 실습
 // SYSDATE : 날짜시간 조회
 > SELECT SYSDATE, SYSDATE - 1, SYSDATE + 1 FROM DUAL;
 // ADD_MONTHS : 몇 개월 이후 날짜 조회
 > SELECT
     ADD_MONTHS(SYSDATE, 1),
     ADD_MONTHS(SYSDATE, -1)
   FROM DUAL;
```

```
4) SQL 기타 함수
 - 저장할 데이터 종류, 즉 자료형을 필요에 따라 변경할 때 형 변환 함수 사용
 - NULL값 확인을 위해 NVL, NVL2 함수 사용
 ☞ 실습하기 3-4. SQL 기타 함수 실습
 // TO_CHAR : 날짜 데이터를 문자 데이터로 변환
 > SELECT
     TO_CHAR(SYSDATE, 'YYYY') AS YYYY,
     TO_CHAR(SYSDATE, 'MM') AS MM,
     TO_CHAR(SYSDATE, 'DD')
                             AS DD,
     TO_CHAR(SYSDATE, 'HH24') AS HH24,
     TO_CHAR(SYSDATE, 'MI') AS MI,
     TO_CHAR(SYSDATE, 'SS')
                             AS SS,
     TO CHAR(SYSDATE, 'YYYY/MM/DD HH24:MI:SS') AS 날짜시간
   FROM DUAL;
 > INSERT INTO EMP VALUES (1001, '김유신', '사원', 7566,
                    TO_CHAR(SYSDATE, 'YYYY/MM/DD'), 3600, NULL, 10);
 // TO_DATE : 문자 데이터를 날짜 데이터로 변환
 > SELECT
     TO_DATE('20180714', 'YYYY/MM/DD') AS 날짜1,
     TO DATE('180714', 'YY-MM-DD') AS 날짜2,
     TO_DATE(SYSDATE, 'YYYY/MM/DD HH24:MI:SS') AS 날짜시간
    FROM DUAL;
 > INSERT INTO EMP VALUES (1002, '김춘추', '사원', 7566, SYSDATE, 3600, NULL, 10);
 > INSERT INTO EMP VALUES (1003, '장보고', '대리', 7566,
                     TO DATE(SYSDATE, 'YYYY-MM-DD HH24:MI:SS'), 2600, NULL, 10);
 // NVL, NVL2 : NULL 체크 함수
 > SELECT EMPNO, ENAME, SAL, COMM,
     NVL(COMM, 0),
     NVL2(COMM, 'NULL 아님', 'NULL 맞음')
    FROM EMP;
```

4. 그룹화

```
1) GROUP BY
 - 여러 데이터에서 의미있는 하나의 결과를 특정 열 값별로 묶어서 조회
 - GROUP BY 절로 먼저 지정한 열로 대그룹을 나누고 그 다음 소그룹으로 그룹화
 ☞ 실습 4-1. 그룹화 실습
 > SELECT AVG(SAL), DEPTNO FROM EMP GROUP BY DEPTNO;
 > SELECT AVG(SAL), DEPTNO FROM EMP GROUP BY DEPTNO, JOB;
2) HAVING
 - GROUP BY로 그룹화된 결과에 대한 조건을 지정해서 조회
 ☞ 실습 4-2. 그룹화 조건 실습
 > SELECT
     DEPTNO,
     JOB,
     AVG(SAL)
   FROM EMP
   GROUP BY DEPTNO, JOB
   HAVING AVG(SAL) >= 2000;
 > SELECT
     DEPTNO,
     JOB,
     AVG(SAL)
   FROM EMP
   WHERE SAL <= 3000
   GROUP BY DEPTNO, JOB
   HAVING AVG(SAL) >= 2000
   ORDER BY DEPTNO, JOB;
```

5. JOIN

```
1) 내부 조인(INNER JOIN)
 - 두 개 이상의 테이블을 연결하여 하나의 테이블처럼 조회
 - 내부 조인은 JOIN 대상이 되는 양쪽 테이블 모두 일치하는 데이터를 출력하는 조인
 ☞ 실습 5-1. 내부 조인 실습
 > SELECT *
  FROM EMP E
   JOIN DEPT D
   ON E.DEPTNO = D.DEPTNO;
 > SELECT *
   FROM EMP E
   JOIN DEPT D
   USING (DEPTNO);
 > SELECT *
   FROM EMP E, DEPT D
   WHERE E.DEPTNO = D.DEPTNO;
 > SELECT
     E.EMPNO,
     E.ENAME,
     E.JOB,
     E.MGR,
     E.HIREDATE,
     E.SAL,
     E.DEPTNO,
     D.DNAME,
     D.LOC
   FROM EMP E
   JOIN DEPT D
   ON E.DEPTNO = D.DEPTNO
   WHERE SAL <= 3000
   ORDER BY E.DEPTNO, EMPNO;
2) 외부 조인(OUTER JOIN)
 - 두 개 이상의 테이블을 연결할 때 어느 한쪽(LEFT, RIGHT)의 모든 데이터를 포함하는 조인
 ☞ 실습 5-2. 외부 조인 실습
 > SELECT * FROM EMP E1 LEFT JOIN EMP E2 ON E1.MGR = E2.EMPNO;
 > SELECT * FROM EMP E1 RIGHT JOIN EMP E2 ON E1.MGR = E2.EMPNO;
```

제4장 PL/SQL

1. PL/SQL 구조

- 1) 블록
 - PL/SQL은 데이터베이스 관련 특정 작업을 수행하는 절차적 프로그래밍
 - 블록은 PL/SQL 프로그램의 기본 구성단위

〈PL/SQL 블록〉

```
DECLARE

[선언부(선택), 실행에 필요한 여러 요소(변수, 상수, 커서 등) 선언]

BEGIN

[실행부(필수), 작업을 위해 실제 실행하는 조건문, 반복문, SQL, 함수 등 작성]

EXCEPTION

[예외 처리부(선택), 작업 수행 중 발생하는 오류 처리]

END;
```

☞ 실습 1-1. Hello, Oracle 출력

```
01 SET SERVEROUTPUT ON; --실행 결과를 콘솔에 출력
02 BEGIN
03 DBMS_OUTPUT.PUT_LINE('Hello, Oracle!');
04 END;
```

2) 주석

- PL/SQL 주석은 코드에 포함되지만 실행되지 않는 문장, 코드 설명 또는 이력 등을 남길 때 사용
- ☞ 실습 1-2. 주석 처리하기

```
/*
날짜 : 2000/01/01
이름 : 김철학
내용 : PL/SQL 주석 실습
*/
DECLARE
   NO NUMBER(4) := 1001;
   NAME VARCHAR2(10) := '홍길동';
   HP CHAR(13) := '010-1000-1001';
   ADDR VARCHAR2(100) := '부산광역시';
BEGIN
   --DBMS_OUTPUT.PUT_LINE('번호 : ' | NO);
   DBMS OUTPUT.PUT LINE('이름 : ' | NAME);
   DBMS_OUTPUT.PUT_LINE('전화 : ' | HP);
   DBMS_OUTPUT.PUT_LINE('주소 : ' | ADDR);
END;
```

2. 변수와 상수

- 1) 변수 선언과 대입
 - 변수(Variable)는 데이터를 일시적으로 저장하고 상수(Constant)는 한번 저장된 값을 고정하는 문법 요소
 - 변수는 선언부에 선언 후 실행부에서 참조
 - ☞ 실습 2-1. 변수 선언 및 변수값 출력

```
-- 콘솔 출력이 안될 경우 실행
SET SERVEROUTPUT ON;
DECLARE
   NO CONSTANT NUMBER(4) := 1001;
   NAME VARCHAR2(10);
   HP CHAR(13) := '000-0000-0000';
   AGE NUMBER(2) DEFAULT 1;
   ADDR VARCHAR2(10) NOT NULL := '부산';
BEGIN
   NAME := '김유신';
   HP := '010-1000-1001';
   DBMS_OUTPUT.PUT_LINE('번호: ' | NO);
   DBMS_OUTPUT.PUT_LINE('이름 : ' | NAME);
   DBMS_OUTPUT.PUT_LINE('전화 : ' || HP);
   DBMS_OUTPUT.PUT_LINE('나이 : ' || AGE);
   DBMS_OUTPUT.PUT_LINE('주소 : ' | ADDR);
END;
```

2) 자료형

- 변수에 저장할 데이터의 종류를 특정 짓기 위해 사용되는 타입
- 자료형은 크게 스칼라형, 참조형, 복합형(Record, Table)으로 구분

스칼라형(Scalar type)

- 스칼라형은 숫자, 문자, 날짜 등 오라클에서 기본으로 정의한 기본 원시(Primitive) 자료형

자료형	설명
NUMBER(p, s)	최대 38 자릿수의 숫자 데이터, p는 최대 유효숫자 자릿수, s는 소수점 자릿수
CHAR(s)	고정 길이 문자열 데이터, 1byte ~ 4000byte
VARCHAR2(s)	가변 길이 문자열 데이터, 1byte ~ 4000byte
DATE	날짜/시간 데이터
BOOLEAN	PL/SQL에서 사용하는 논리 자료형으로 true, false, NULL 포함

참조형(Reference type)

- 참조형은 데이터베이스에 존재하는 특정 테이블 열의 자료형이나 하나의 행 구조를 참조하는 자료형

구분	설명
%TYPE	열 참조, 특정 테이블의 하나의 열의 자료형을 참조
%ROWTYPE	행 참조, 특정 테이블의 하나의 행의 구조를 참조

복합형(Composite type)

- 복합형은 여러 종류 및 개수의 데이터를 저장하기 위해 사용자가 직접 정의하는 자료형

구분	설명
RECORD	여러 종류의 자료형의 데이터를 저장(테이블의 행과 유사)
TABLE(연관배열)	한 가지 자료형의 데이터를 여러 개 저장(테이블 열 또는 배열과 유사)

☞ 실습 2-2. 열 참조형 변수 실습

```
DECLARE
NO DEPT.DEPTNO%TYPE;
NAME DEPT.DNAME%TYPE;
ADDR DEPT.LOC%TYPE;

BEGIN
SELECT *
INTO NO, NAME, ADDR
FROM DEPT
WHERE DEPTNO = 30;

DBMS_OUTPUT.PUT_LINE('부서번호:'|| NO);
DBMS_OUTPUT.PUT_LINE('부서명:'|| NAME);
DBMS_OUTPUT.PUT_LINE('주소:'|| ADDR);

END;
```

☞ 실습 2-3. 행 참조형 변수 실습

```
DECLARE
-- 선언
ROW_DEPT DEPT%ROWTYPE;
BEGIN
-- 처리
SELECT *
INTO ROW_DEPT
FROM DEPT
WHERE DEPTNO = 40;

-- 출력
DBMS_OUTPUT.PUT_LINE('부서번호 : ' || ROW_DEPT.DEPTNO);
DBMS_OUTPUT.PUT_LINE('부서명 : ' || ROW_DEPT.DNAME);
DBMS_OUTPUT.PUT_LINE('주소 : ' || ROW_DEPT.LOC);
END;
```

☞ 실습 2-4. 레코드 자료형 기본 실습

```
DECLARE
   -- Record Define
   TYPE REC_DEPT IS RECORD (
       deptno NUMBER(2),
       dname DEPT.DNAME%TYPE,
       loc DEPT.LOC%TYPE
   );
   -- Record Declare
   dept rec REC DEPT;
BEGIN
   -- Record Initialize
   dept_rec.deptno := 10;
   dept_rec.dname := '개발부';
   dept_rec.loc := '부산';
   -- Record Print
   DBMS_OUTPUT.PUT_LINE('deptno : ' || dept_rec.deptno);
   DBMS_OUTPUT.PUT_LINE('dname : ' || dept_rec.dname);
   DBMS_OUTPUT.PUT_LINE('loc : ' || dept_rec.loc);
   DBMS_OUTPUT.PUT_LINE('PL/SQL 종료...');
END;
```

☞ 실습 2-5. 레코드 사용한 INSERT 실습

```
-- 테이블 복사(데이터 제외)

CREATE TABLE DEPT_RECORD AS SELECT * FROM DEPT WHERE 1 = 0;

DECLARE

TYPE REC_DEPT IS RECORD (
    deptno NUMBER(2),
    dname DEPT.DNAME%TYPE,
    loc DEPT.LOC%TYPE
   );
   dept_rec REC_DEPT;

BEGIN

dept_rec.deptno := 10;
   dept_rec.dname := '개발부';
   dept_rec.loc := '부산';

INSERT INTO DEPT_RECORD VALUES dept_rec;

DBMS_OUTPUT.PUT_LINE('PL/SQL 종료...');

END;
```

☞ 실습 2-6. 레코드를 포함하는 레코드 실습

```
DECLARE
   TYPE REC DEPT IS RECORD (
       deptno DEPT.DEPTNO%TYPE,
       dname DEPT.DNAME%TYPE,
       loc
              DEPT.LOC%TYPE
   );
   TYPE REC_EMP IS RECORD (
       empno EMP.EMPNO%TYPE,
       ename EMP.ENAME%TYPE,
       dinfo REC_DEPT
   );
   emp_rec REC_EMP;
BEGIN
   SELECT E.EMPNO, E.ENAME, D.DEPTNO, D.DNAME, D.LOC
       emp_rec.empno,
       emp_rec.ename,
       emp_rec.dinfo.deptno,
       emp_rec.dinfo.dname,
       emp rec.dinfo.loc
   FROM EMP E, DEPT D
   WHERE E.DEPTNO = D.DEPTNO AND E.EMPNO = 7788;
   DBMS_OUTPUT.PUT_LINE('EMPNO : ' || emp_rec.empno);
   DBMS_OUTPUT.PUT_LINE('ENAME : ' || emp_rec.ename);
   DBMS_OUTPUT.PUT_LINE('DEPTNO : ' || emp_rec.dinfo.deptno);
   DBMS_OUTPUT.PUT_LINE('DNAME : ' || emp_rec.dinfo.dname);
   DBMS_OUTPUT.PUT_LINE('LOC : ' || emp_rec.dinfo.loc);
   DBMS_OUTPUT.PUT_LINE('PL/SQL 종료...');
END;
```

☞ 실습 2-7. 테이블(연관배열) 기본 실습

```
DECLARE

TYPE ARR_CITY IS TABLE OF VARCHAR2(20) INDEX BY PLS_INTEGER;
arrCity ARR_CITY;

BEGIN

arrCity(1) := '서울';
arrCity(2) := '대전';
arrCity(3) := '대구';

DBMS_OUTPUT.PUT_LINE('arrCity(1) : ' || arrCity(1));
DBMS_OUTPUT.PUT_LINE('arrCity(2) : ' || arrCity(2));
DBMS_OUTPUT.PUT_LINE('arrCity(3) : ' || arrCity(3));
DBMS_OUTPUT.PUT_LINE('PL/SQL 종료...');

END;
```

3. 제어문

- 1) 조건문
 - 조건문은 특정 조건에 따라 분기 처리되는 구문
 - PL/SQL는 IF문과 CASE문 사용
 - ☞ 실습 3-1. IF문 실습

```
DECLARE
NUM NUMBER := 1;
BEGIN
IF NUM > 0 THEN
DBMS_OUTPUT.PUT_LINE('NUM은 0보다 크다.');
END IF;
DBMS_OUTPUT.PUT_LINE('PL/SQL 종료...');
END;
```

☞ 실습 3-2. IF~ELSE 실습

```
DECLARE
NUM NUMBER := -1;

BEGIN

IF NUM > 0 THEN

DBMS_OUTPUT.PUT_LINE('NUM은 0보다 크다.');

ELSE

DBMS_OUTPUT.PUT_LINE('NUM은 0보다 작다.');

END IF;

DBMS_OUTPUT.PUT_LINE('PL/SQL 종료...');

END;
```

☞ 실습 3-3. IF~ELSIF 실습

```
DECLARE
   SCORE NUMBER := 86;
BEGIN
   IF SCORE >= 90 AND SCORE <= 100 THEN
       DBMS_OUTPUT.PUT_LINE('A 입니다.');
   ELSIF SCORE >= 80 AND SCORE < 90 THEN
       DBMS_OUTPUT.PUT_LINE('B 입니다.');
   ELSIF SCORE >= 70 AND SCORE < 80 THEN
       DBMS_OUTPUT.PUT_LINE('C 입니다.');
   ELSIF SCORE >= 60 AND SCORE < 70 THEN
       DBMS_OUTPUT.PUT_LINE('D 입니다.');
   ELSE
       DBMS_OUTPUT.PUT_LINE('F 입니다.');
   END IF;
   DBMS_OUTPUT.PUT_LINE('PL/SQL 종료...');
END;
```

☞ 실습 3-4. CASE 실습

```
DECLARE
SCORE NUMBER := 86;
BEGIN

CASE FLOOR(SCORE/10)

WHEN 9 THEN DBMS_OUTPUT.PUT_LINE('A 입니다.');
WHEN 8 THEN DBMS_OUTPUT.PUT_LINE('B 입니다.');
WHEN 7 THEN DBMS_OUTPUT.PUT_LINE('C 입니다.');
WHEN 6 THEN DBMS_OUTPUT.PUT_LINE('D 입니다.');
ELSE DBMS_OUTPUT.PUT_LINE('F 입니다.');
END CASE;
DBMS_OUTPUT.PUT_LINE('PL/SQL 종료...');
END;
```

2) 반복문

- 반복문은 특정 작업을 반복 수행하는 구문
- PL/SQL에서는 LOOP, FOR LOOP, WHILE LOOP 제공
- ☞ 실습 3-5. 기본 LOOP 실습

```
DECLARE
NUM NUMBER := 0;

BEGIN

LOOP

DBMS_OUTPUT.PUT_LINE('NUM : ' || NUM);
NUM := NUM + 1;

IF NUM > 3 THEN
EXIT;
END IF;
END LOOP;
DBMS_OUTPUT.PUT_LINE('PL/SQL 종료...');

END;
```

☞ 실습 3-6. 기본 FOR 실습

```
BEGIN

FOR i IN 1..3 LOOP

DBMS_OUTPUT.PUT_LINE('i : ' || i);

END LOOP;

DBMS_OUTPUT.PUT_LINE('PL/SQL 零료...');

END;
```

☞ 실습 3-7. 기본 WHILE 실습

```
DECLARE
NUM NUMBER := 0;

BEGIN
WHILE NUM < 4 LOOP

DBMS_OUTPUT.PUT_LINE('NUM : ' || NUM);
NUM := NUM + 1;
END LOOP;
DBMS_OUTPUT.PUT_LINE('PL/SQL 零료...');

END;
```

☞ 실습 3-8. CONTINUE 실습

```
DECLARE
NUM NUMBER := 0;
BEGIN
WHILE NUM < 5 LOOP

NUM := NUM + 1;

-- MOD(): 나머지를 구하는 SQL함수
IF MOD(NUM, 2) = 0 THEN
CONTINUE;
END IF;

DBMS_OUTPUT.PUT_LINE('NUM : ' || NUM);

END LOOP;

DBMS_OUTPUT.PUT_LINE('PL/SQL 종료...');
END;
```

☞ 실습 3-9. CONTINUE-WHEN 실습

```
FOR i IN 1..5 LOOP

CONTINUE WHEN MOD(i, 2) = 0;

DBMS_OUTPUT.PUT_LINE('i : ' || i);

END LOOP;

DBMS_OUTPUT.PUT_LINE('PL/SQL 零료...');

END;
```

4. 커서와 예외처리

- 1) 커서
 - 커서(Cursor)는 SQL을 실행했을 때 해당 SQL을 처리하는 정보를 저장한 메모리 포인터
 - SELECT의 결과가 여러 행으로 나올 때 각 행별로 특정 작업을 처리할 때 커서를 사용
 - ☞ 실습 4-1. 단일 행 결과를 처리하는 커서 사용

```
DECLARE
   -- 커서 데이터를 저장할 변수 선언
   V_DEPT_ROW DEPT%ROWTYPE;
   -- 커서 선언
   CURSOR c1 IS SELECT * FROM DEPT WHERE DEPTNO = 40;
BEGIN
   -- 커서 열기
   OPEN c1;
   -- 커서 데이터 입력
   FETCH c1 INTO V_DEPT_ROW;
   -- 커서 데이터 출력
   DBMS_OUTPUT.PUT_LINE('DEPTNO : ' || V_DEPT_ROW.DEPTNO);
   DBMS_OUTPUT.PUT_LINE('DNAME : ' || V_DEPT_ROW.DNAME);
   DBMS_OUTPUT.PUT_LINE('LOC : ' || V_DEPT_ROW.LOC);
   -- 커서 종료
   CLOSE c1;
END;
```

☞ 실습 4-2. 여러 행 결과를 처리하는 커서 사용(LOOP)

```
DECLARE

V_DEPT_ROW DEPT%ROWTYPE;

CURSOR c1 IS SELECT * FROM DEPT;

BEGIN

OPEN c1;

LOOP

FETCH c1 INTO V_DEPT_ROW;

EXIT WHEN c1%NOTFOUND;

DBMS_OUTPUT.PUT_LINE('-----');

DBMS_OUTPUT.PUT_LINE('DEPTNO : ' || V_DEPT_ROW.DEPTNO);

DBMS_OUTPUT.PUT_LINE('DNAME : ' || V_DEPT_ROW.DNAME);

DBMS_OUTPUT.PUT_LINE('LOC : ' || V_DEPT_ROW.LOC);

END LOOP;

CLOSE c1;

END;
```

☞ 실습 4-3. 여러 행 결과를 처리하는 커서 사용(FOR)

```
DECLARE

CURSOR c1 IS SELECT * FROM DEPT;

BEGIN

FOR c1_rec IN c1 LOOP

DBMS_OUTPUT.PUT_LINE('----');

DBMS_OUTPUT.PUT_LINE('DEPTNO : ' || c1_rec.DEPTNO);

DBMS_OUTPUT.PUT_LINE('DNAME : ' || c1_rec.DNAME);

DBMS_OUTPUT.PUT_LINE('LOC : ' || c1_rec.LOC);

END LOOP;

END;
```

2) 예외처리

- 오류(Error)는 PL/SQL이 정상적으로 수행되지 못하는 상황
- PL/SQL이 실행되는 도중 발생하는 오류를 처리하기 위해 예외처리(Exceptino) 수행
- ☞ 실습 4-4. 기본 예외처리 실습

```
DECLARE
v_wrong NUMBER;

BEGIN
SELECT DNAME INTO v_wrong FROM DEPT WHERE DEPTNO = 10;

EXCEPTION
WHEN VALUE_ERROR THEN
DBMS_OUTPUT.PUT_LINE('VALUE_ERROR(수치 또는 값) 에러발생...');

END;
```

☞ 실습 4-5. 예외처리 코드/메시지 실습

```
DECLARE
deptno DEPT.DEPTNO%TYPE;
dname DEPT.DNAME%TYPE;

BEGIN
SELECT DEPTNO, DNAME INTO deptno, dname FROM DEPT;

DBMS_OUTPUT.PUT_LINE('deptno : ' || deptno);
DBMS_OUTPUT.PUT_LINE('dname : ' || dname);

EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('에러 발생...');
DBMS_OUTPUT.PUT_LINE('에러코드 : ' || SQLCODE);
DBMS_OUTPUT.PUT_LINE('에러내용 : ' || SQLERRM);

END;
```