20차 수업

\*\* 환경변수

* 운영체제에게 알려주는 옵션
* Window는 GUI 환경에서 편집이 가능한데 나머지 운영체제는 텍스트를 직접 편집해야 합니다.
* 운영체제 공부를 하게 되면 텍스트 편집기 사용법을 먼저 공부해야 함

\*\* Path 환경변수

* 콘솔이나 실행 창에 명령어를 입력했을 때 명령어를 찾는 위치
* 어떤 프로그램들은 설치를 할 때 자신의 경로를 자동으로 path에 추가하고 어떤 프로그램들은 직접 추가하는 경우가 있습니다.
* 프로그램을 설치할 때 환경변수나 path라는 단어가 보이면 적적하게 옵션을 설정해야 합니다.
* 자바는 설치할 때 자동으로 path 설정이 안되고 oracle은 설치할 때 자동으로 path설정을 합니다.

\*\* java 개발환경 설정

1. Java(java development kit – 자바 개발 도구) 설치

* Java 개발에 필요한 명령어들을 가지고 있는 프로그램
* Jre(자바 실행 환경)는 자바에서 제공하는 기본 클래스들과 실행에 필요한 jvm(java virtual machine)을 포함하고 있는 프로그램
* 개발을 할 때는 jdk와 jre를 모두 설치해야 하는데 보통은 jdk안에 jre가 포함되어 있습니다.
* Jdk도 하나의 버전이 아니고 oracle에서 나온 버전이 있고 IBM에서 만든 버전도 있고 오픈소스로 만들어진 open jdk도 있습니다.
* Fedora(개발 중인 버전), cent os, redhat enterprise(유료- 안정화 된 버전)
* Jdk를 설치 할 때 1.8버전이상이면 최근의 실무에서 사용하는 모든 자바의 기능을 대부분 사용할 수 있습니다.
* 다운로드 받을 때 주의할 점은 64bit 운영체제에 32bit 버전이 설치가 되는데 jdk를 32bit 버전을 설치하면 이후에 java와 관련된 모든 프로그램을 전부 32bit 버전으로 설치해야 합니다.

1. 설치 확인

* Java – version : jre 설치 여부 확인
* Javac –version : jdk 설치 여부 확인 - windows에서는 jdk 경로를 환경변수에 추가하지 않기 때문에 처음에는 명령어가 없다고 에러가 납니다.

1. Jdk를 설치하고 나면 jdk의 bin이라는 디렉토리를 path라는 환경변수에 추가를 해야 java 명령어를 아무 곳에서나 사용 할 수 있습니다.

* 프로그램 안에 bin 이라는 디렉토리가 있으면 이 디렉토리는 실행가능한 명령어 또는 프로그램을 모아놓은 디렉토리입니다.
* lib라는 디렉토리가 있으면 프로그램 실행에 관련된 라이브러리를 가지고 있는 디렉토리입니다.

1. 자바로 만들어진 프로그램들이 쉽게 실행하도록 할 때는 JRE\_HOME 이라는 환경변수에 jre 디렉토리를 설정해주면 됩니다.
2. Eclipse 설치

* IDE(통합 개발 환경) : 소스 코드를 작성하고 번역을 해주고 실행을 할 수 있도록 해주는 프로그램
* Java IDE로는 Ecilpse(전자정부 프레임워크 나 대기업 프레임워크의 기반 – Open Source), Intelli J(개인 개발자나 중소기업이나 중견기업에서 최근에 많이 사용, 코드 센스 및 빌드가 Eclipse 보다는 우수 – 상용화, 자바 애플리케이션 제작만 한다면 무료 버전이 있고 이것을 기반으로 해서 Android Studio 가 만들어짐)
* 그 이외에도 NetBeans등이 있지만 국내에서는 거의 사용하지 않습니다.
* Eclipse IDE for Enterprise Java Developers 옵션을 선택해야만 Java 웹 프로그래밍을 할 수 있습니다.
* Eclipse를 설치하고 나면 환경설정(Preferences) 메뉴를 실행시켜서 자신에게 맞도록 환경설정을 해주는 것이 좋습니다.
* 윈도우에서 작업할 때는 인코딩 설정을 변경해주는 것이 좋습니다.

\*\* DTO(Data Transfer Object – Value Object)

* 여러 개의 속성을 묶어서 표현하기 위한 클래스
* 테이블 구조의 데이터(관계형 데이터베이스에서 테이블, NoSQL에서는 하나의 도큐먼트)에서 하나의 행을 표현하기 위한 클래스
* 생성을 할 때는 클래스를 만들고 표현할 속성들을 private 변수로 생성
* 변수에 접근하기 위한 접근자 메소드를 생성
* 빠르게 디버깅하기 위해서 toString 메소드를 재정의
* 상황에 따라서 서로 다른 컴퓨터끼리 데이터를 주고받는 경우에는 Serializable 인터페이스를 구현
* 크기 비교를 해야 하는 경우에는 Comparable 인터페이스를 구현
* 응용 프로그램을 만들 때는 Cloneable 인터페이스를 구현
* 동일성 여부를 일부분만 가지고 판단하고 할 때는 equals 와 hashCode 메소드를 재정의

외부에서 데이터를 가져와서 설정하는 경우에는 매개변수가 있는 생성자를 만들어서 초기화를 편리하게 수행하는 것이 좋습니다.

생성자(Constructor)를 만드는 경우는 외부에서 데이터를 가져와서 초기화를 하거나 객체가 생성될 때 무엇인가를 해야 하는 경우에 생성

\*\* 번호(정수), 이름(문자열), 점수(정수), 생년월일(날짜)

**package** test;

**import** java.util.Date;

**public** **class** DTO {

// 번호, 이름, 점수, 생년월일을 속성으로 갖는 DTO 클래스

**private** **int** num;

**private** String name;

**private** **int** score;

// java.util.Date 시간까지 포함

// java.sql.Date는 시간을 포함하지 않음

**private** Date birthday;

// 매개변수가 없는 생성자(Default Constructor)

// 초기값이 없는 상태로 생성할 수 있도록 해주는 생성자

**public** DTO() {

**super**();

}

// 초기값이 있는 경우에 사용할 생성자

**public** DTO(**int** num, String name, **int** score, Date birthday) {

**super**();

**this**.num = num;

**this**.name = name;

**this**.score = score;

**this**.birthday = birthday;

}

// 접근자 메소드 : 속성에 접근하기 위한 메소드

**public** **int** getNum() {

**return** num;

}

**public** **void** setNum(**int** num) {

**this**.num = num;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getScore() {

**return** score;

}

**public** **void** setScore(**int** score) {

**this**.score = score;

}

**public** Date getBirthday() {

**return** birthday;

}

**public** **void** setBirthday(Date birthday) {

**this**.birthday = birthday;

}

// 속성의 값을 빠르게 확인하기 위해서 사용하는 메소드

// 확인을 하는 이유는 맞는지 안맞는지 검증 - 디버깅

// 이 메소드는 출력하는 메소드에 객체참조만 대입하면 자동으로 호출

@Override

**public** String toString() {

**return** "DTO [num=" + num + ", name=" + name + ", score=" + score + ", birthday=" + birthday + "]";

}

}

**package** test;

**import** java.util.Date;

**import** java.util.Stack;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

// DTO 클래스의 객체를 5개 생성

DTO dto1 = **new** DTO();

dto1.setNum(1);

dto1.setName("조창훈");

dto1.setScore(98);

dto1.setBirthday(**new** Date());

DTO dto2 = **new** DTO(2, "양재건", 87, **new** Date());

DTO dto3 = **new** DTO(3, "강영준", 93, **new** Date());

DTO dto4 = **new** DTO(4, "지현섭", 87, **new** Date());

DTO dto5 = **new** DTO(5, "박대근", 15, **new** Date());

// 동일한 모양의 데이터가 여러개 있을 때는 하나로 모아야 합니다.

// 이 때 사용하는 자료구조가 List

// 종류 : 배열, ArrayList, LinkedList, Stack, Queue, Deque 등이 있음

// 위의 데이터들을 Stack에 저장

Stack <DTO> stack = **new** Stack<>();

stack.push(dto1);

stack.push(dto2);

stack.push(dto3);

stack.push(dto4);

stack.push(dto5);

**for**(**int** i=0; i<5; i=i+1) {

System.***out***.printf("%s\n", stack.pop());

}

}

}

\*\* Oracle 설치 및 접속 확인(비밀번호 : 92whckdgns)

1. [www.oracle.com](http://www.oracle.com)에서 오라클을 다운로드

* Enterprise Edition : 기업용
* Standard Edition : Enterprise Edition에서 기능을 조금 제거한 버전
* Express Edition : 가장 가벼운 버전
* Express Edition은 메모리를 1GB만 사용, 나머지 버전은 메모리를 무제한 사용

1. 버전은 11g, 12c, 18c 등이 있는데 SQL과 간단한 운영 정도를 공부할 때는 버전 차이는 없습니다.
2. 설치 확인

* Cmd 창에서 sqlplus 라고 입력하고 엔터

1. 접속 확인

* 사용자명 : system
* 비밀번호 : 설치할 때 비밀번호 입력

\*\* 오라클을 설치하고 나면 만들어지는 계정

1. Sys : 슈퍼관리자
2. System : 관리자

* 설치할 때 입력한 비밀번호로 계정이 생성됩니다.
* 관리자 비밀번호를 잃어버린 경우 아래처럼 접속해서 명령어를 수행
* 유저명에 /as sysdba : 비밀번호 없이 접속
* alter user system identified by 새로운 비밀번호;
* alter user sys identified by 새로운 비밀번호;

\*\* 오라클 접속 프로그램

1. sqlplus : 오라클을 설치하면 자동으로 설치

* console 창에서 작업
* DBA 들이 많이 사용

1. SQLDeveloper : 오라클에서 제공하는 GUI 접속 프로그램

* Java로 만들어져 있습니다.

1. Toad : GUI 접속 프로그램

* 무료 버전과 상용화된 버전이 존재
* 실무에서 많이 사용하는 접속 도구

1. dbeaver

* 여러가지 데이터베이스에 접속할 수 있는 프로그램
* jdbc(java 데이터베이스 연동)를 이용해서 데이터베이스에 접속
* 데이터베이스에 접속할 때 데이터베이스 드라이버가 필요
* 데이터베이스 드라이버는 프로그래밍 언어가 데이터베이스와 연결이 될 때 사용되는 인터페이스
* 오라클의 경우는 오라클 설치 디렉토리/product/버전/이름/jdbc/lib 라는 디렉토리에 드라이버가 존재

\*\* 실습 계정

* 오라클 11g 에는 오라클 학습을 위한 2개의 계정이 존재
* scott, hr 계정입니다.
* 관리자 계정(sys 또는 system)으로 접속
  + alter user scott identified by tiger account unlock;
  + scott 계정의 비밀번호를 tiger로 설정해서 계정을 풀기

\*\* 오라클의 데이터베이스 개체

1. Table : 실제 데이터를 저장하는 개체
2. View : 자주 사용하는 select(조회) 구문을 저장한 후 마치 하나의 테이블처럼 이용하는 개체
3. Index : 데이터를 빠르게 조회하기 위해서 데이터에 포인터를 만들어 놓은 개체
4. Sequence : 일련번호
5. Synonym : 개체에 대한 별명
6. Procedure : 자주 사용하는 SQL 구문을 하나의 이름으로 묶어 둔 것
7. Function : 연산의 결과를 반환하는 개체
8. Trigger : DML(삽입, 삭제, 갱신) 문장을 수행하기 전이나 수행한 후에 자동으로 동작을 할 수 있는 만들어 놓은 개체
9. Package : 자주 사용하는 프로시저나 함수 또는 트리거를 묶어놓은 개체

* 개발자는 테이블 -> 프로시저 -> 트리거 정도를 중요하게 생각하고 관리자는 인덱스를 중요하게 생각합니다.
* 오라클에서 프로시저나 트리거를 만드는 문법을 PL/SQL 이라고 합니다.
* 오라클 가능자와 PL/SQL 가능자는 다른 의미입니다.
* 오라클 가능자는 오라클에 데이터를 CRUD 할 수 있는 개발자를 의미하고 PL/SQL 가능자는 프로시저나 트리거를 만들 수 있는 개발자를 의미합니다.

\*\* SQL(Structured Query Language)

* 구조적 질의 언어 – 비절차적(입력한 순서대로 실행되지 않음)
* 관계형 데이터베이스(RDBMS – 테이블 구조 기반의 데이터베이스)에서 데이터베이스에 개체에 대한 명령문

1. DDL(Data Definition Language – 데이터 정의언어)

* 데이터베이스 개체를 생성하고 삭제하고 구조 변경하는 명령어
* Create : 데이터베이스 개체 생성
* Alter : 개체의 구조를 변경
* Drop : 개체를 삭제
* Truncate : 테이블 내의 데이터를 전부 삭제
* Rename : 개체에 대한 이름 변경

1. DML(Data Manipulation Language – 데이터 조작언어)

* 테이블 내의 데이터를 조작하는 명령어
* Insert : 데이터 추가
* Update : 데이터 갱신
* Delete : 데이터 삭제

1. DQL(Data Query Language – 데이터 조회언어)

* 테이블 내의 데이터를 검색하는 명령어 – 이론 책에서는 DML로 분류
* Select : 데이터 조회

1. TCL(Transaction Control Language – 트랜잭션 제어언어)

* commit : 현재까지의 작업내용을 원본에 반영
* rollback : 현재까지의 작업내용을 원본에 반영하지 않음

1. DCL(Data Control Language – 데이터 제어언어)

* Grant : 권한 부여
* Revoke : 권한 회수

1. 개발자는 DQL -> DML -> TCL -> DDL -> DCL 순으로 중요

* 개발자가 DCL까지 하는 경우는 드뭅니다.
* 관계형 데이터베이스를 사용하는 실무 프로젝트의 경우 DBA(데이터베이스 관리자)나 DB 설계를 담당하는 실무자가 따로 있어서 DDL도 개발자가 잘 하지 않는데 공부를 할 때는 테이블 설계를 해 줄 사람이 따로 없기 때문에 사용할 수 있어야 합니다.

1. SQL 작성법
   1. 하나의 명령문 끝은 ;(세미콜론)
   * 접속 프로그램을 이용하는 경우는 ;(세미콜론)을 생략해도 되고 java에서 sql을 입력할 때는 ;(세미콜론)을 붙이면 안됩니다.
   1. 문자열은 ‘ ‘(작은 따옴표) 사이에 입력
   2. 데이터는 대소문자를 구별하지만 명령어는 대소문자를 구별하지 않습니다.
   3. 줄 바꿈은 의미가 없어서 해도 되고 안 해도 되지만 되도록이면 하나의 절은 한 줄에 입력하는 것을 권장
   * 공부를 할 때는 명령문이 짧아서 한 줄에 입력해도 되지만 실무에서 사용하는 SQL 문장은 길어서 구분을 잘 해두지 않으면 읽기가 어렵습니다.(가독성이 떨어집니다.)

\*\* SCOTT에 제공되는 4개의 테이블

1. emp 테이블 – 사원 테이블

* select\*from emp;
  + empno : 사원번호로 숫자로 되어있으며 not null이고 unique(primary key)
  + ename : 사원이름으로 문자열로 구성
  + job : 직업으로 문자열로 구성
  + mgr : 관리자 사원번호로 숫자
  + hiredate : 입사일로 date 타입
  + sal : 급여로 정수
  + comm : 상여금으로 정수인데 null을 포함
  + deptno : 부서 번호로 뒤에 나오는 dept 테이블의 deptno를 참조합니다.(foreign key)

1. dept(부서) 테이블
   * deptno : 부서 번호로 숫자로 되어 있으면 not null이고 unique(primary key)
   * dname : 부서이름으로 문자로 되어있음
   * loc : 부서의 위치로 문자로 되어있음
2. salgrade(호봉) 테이블
   * grade : 숫자로 호봉수를 의미 not null 이고 unique(primary key)
   * losal : grade의 최소 급여로 숫자
   * hisal : grade의 최대 급여로 숫자
3. bonus 테이블
   * ename : 사원이름으로 문자
   * job : 직업으로 문자
   * sal : 급여로 숫자
   * comm : 상여금으로 숫자

\*\* 명령어

1. 현재 접속 중인 유저 확인

* show user;

1. 테이블 구조 확인

* desc 테이블 이름;

\*\* 윈도우에 오라클을 설치하면 자동 실행됩니다.

자동실행을 하지 않고 직접 실행하고 중지를 하려면 서비스에서 하면 됩니다.

OracleService는 오라클 자체를 중지하고 실행

TNSListener는 외부에서 접속할 때 접속을 시켜주는 서비스

\*\* 관계형 데이터베이스 용어

1. 테이블 – 릴레이션이라고도 합니다.
2. 테이블에서 하나의 행 – row 또는 record
3. 테이블에서 하나의 열 – column, attribute, field

\*\* 데이터베이스 접속 프로그램 – dbeaber

* sqlplus를 이용해서 오라클 작업을 해도 되지만 실제 업무에서는 오라클이 설치되지 않은 컴퓨터에서 작업을 많이 하기 때문에 sqlplus를 사용할 수 없습니다.
* 원격의 데이터베이스에 접속할 수 있는 프로그램을 이용해서 데이터베이스 작업을 하는 경우가 많습니다.

\*\* SELECT 구문 : 데이터를 조회하는 구문

* 5 - select\*또는 컬럼이름나열
* 1 - from 테이블이름
* 여기까지는 필수
* 2 - where 조건
* 3 - group by 그룹화 할 조건
* 4 - having group by 이후의 조건
* 6 - order by 정렬할 조건

1. 테이블의 전체 데이터 조회

* select\*
* from 테이블이름;

1. 테이블에서 특정 컬럼만 조회

* SELECT 컬럼이름을 나열
* FROM 테이블이름;

--EMP 테이블의 모든 데이터를 조회

**SELECT**\*

**FROM** EMP;

--EMP 테이블의 ENAME 과 SAL을 조회

**SELECT** ENAME, SAL

**FROM** EMP;

1. 숫자 데이터의 경우 산술연산을 해서 조회가 가능

* SELECT 절에 컬럼과 다른 컬럼 또는 숫자 데이터와 산술연산을 하면 산술연산의 결과를 출력을 해줍니다.
* SAL 컬럼은 한달 동안의 급여인데 일 년치 급여를 알고자 하는 경우
  + ENAME 과 SAL\*12 의 결과를 조회

--EMP 테이블에서 ENAME 과 SAL\*12 의 결과를 조회

**SELECT** ENAME, SAL\*12

**FROM** EMP;

1. 데이터베이스에서 NULL

* NULL은 아직 알려지지 않은 값
* NULL과 다른 데이터가 연산하면 결과는 무조건 NULL
* NULL이 있는 데이터와 연산을 할 때는 어떻게 할 것인가 고민을 해야 합니다.
* 이 경우 NULL 값을 대체하는 역할을 수행해주는 함수가 NVL
* NVL(컬럼이름이나 연산식, NULL일 때 대체값)

--EMP 테이블에서 ENAME 과 SAL 그리고 COMM 컬럼을 조회

**SELECT** ENAME, SAL, COMM

**FROM** EMP;

--EMP 테이블에서 ENAME 과 SAL\*12+COMM을 조회

--COMM이 NULL인 행은 결과가 NULL로 리턴

**SELECT** ENAME, SAL\*12 + COMM

**FROM** EMP;

--EMP 테이블에서 ENAME 과 SAL\*12+COMM을 조회

--COMM 컬럼의 값이 NULL인 경우는 0으로 계산

**SELECT** ENAME, SAL\*12 + **NVL**(COMM, 0)

**FROM** EMP;

1. SELECT 절에서 별명(Alias) 사용

* 산술 연산식을 출력할 때 별명을 사용하지 않으면 연산식이 그대로 보여지기 때문에 보기에 적당하지 않습니다.
* 별명을 만들 때는 공백을 한 칸 주고 별명을 작성하면 됩니다.
  + 다른 공백이 있는 경우라면 as 별명 의 형태로 입력해도 됩니다.
  + 별명에 대문자가 있거나 특수문자가 있으면 “ ”로 묶어서 입력합니다.

--EMP 테이블에서 ENAME 과 SAL\*12+COMM을 조회

--COMM 컬럼의 값이 NULL인 경우는 0으로 계산

--계산한 컬럼의 이름을 연봉으로 출력

**SELECT** ENAME, SAL\*12 + **NVL**(COMM, 0) **AS** 연봉

**FROM** EMP;

1. 연결 연산자

* ||를 이용하면 2개의 컬럼 또는 컬럼과 다른 데이터를 연결해서 출력할 수 있습니다.

--EMP 테이블에서 ENAME 과 SAL\*12+COMM을 조회

--COMM 컬럼의 값이 NULL인 경우는 0으로 계산

--ENAME 의 연봉은 SAL\*12 + NVL(COMM, 0) AS 연봉 입니다.

**SELECT** ENAME || '의 연봉은' || (SAL\*12+**NVL**(COMM,0)) || '입니다.' **AS** 연봉

**FROM** EMP;

1. DISTINCT

* select 절의 맨 앞에 나올 수 있는 명령으로 중복을 제거하고 조회
* 컬럼이름이 1개이면 그 컬럼의 중복만 제거하는 것이고 컬럼이름이 2개 이상이면 모든 컬럼의 값이 동일한 경우만 제거

--EMP 테이블에서 JOB의 종류를 확인

**SELECT** JOB

**FROM** EMP;

--중복을 제거하고 JOB을 조회

**SELECT** **DISTINCT** JOB

**FROM** EMP;

--DISTINCT 뒤에 컬럼이름이 2개이면 2개 모두 일치하는 데이터만 제거

**SELECT** **DISTINCT** JOB, DEPTNO

**FROM** EMP;

\*\* WHERE 절

* 조건을 설정해서 테이블을 행단위로 분할하는 절
* select는 열 단위로 분한
* from 다음에 수행

1. 기본 연산자

* >, >=, <, <=
* =, !=(<>)
* 컬럼이름 연산자 비교대상 의 형태로 입력
* 반대로 만들 때는 컬럼이름 앞에 NOT을 추가해서 가능
  + sal > 1000 문장을 아래처럼 작성해도 됩니다.
  + NOT SAL <= 1000
  + 부등호 연산자가 등호 연산자보다 우선순위가 더 높음

--EMP 테이블에서 SAL 이 3000이상인 데이터의 empno, ename, sal을 조회

**SELECT** EMPNO, ENAME, SAL

**FROM** EMP

**WHERE** SAL >= 3000;

--EMP 테이블에서 JOB이 CLERK 인 데이터의 empno, ename을 조회

**SELECT** EMPNO, ENAME

**FROM** EMP

**WHERE** JOB = 'CLERK';

1. Oracle에서는 날짜와 날짜 사이의 산술연산 그리고 날짜와 숫자의 산술연산 및 크기 비교가 가능

* 숫자 1을 하루로 계산합니다.
* 문자열을 날짜로 표현하는 함수 : to\_date(‘날짜문자열’, ‘날짜서식’)
  + 서식에서 년도는 y 월은 mm 날짜는 dd 시간은 hh 인데 hh는 12시간제이고 hh24가 24시간제이며 분은 mi 초는 ss로 표현
* 오늘 날짜는 sysdate

--EMP 테이블에서 hiredate가 입사일 입니다.

--EMP 테이블에서 ENAME 과 입사한 지 몇 일 되었는지 출력

**SELECT** ENAME, **SYSDATE**-hiredate

**FROM** EMP;

--EMP 테이블에서 1982년 12월 31일 이후에 입사한 사원의 ENAME과 HIREDATE를 조회

**SELECT** ENAME, HIREDATE

**FROM** EMP

**WHERE** HIREDATE > **TO\_DATE**('1982-12-31', 'yyyy-mm-dd');

1. and 와 or

* and 는 2개의 조건을 모두 만족한 경우에만 조회
* or 는 2개의 조건 중 하나만 만족해도 조회
* and 가 or 보다 우선 순위가 높습니다.

--EMP 테이블에서 1982년에 입사한 사원의 ENAME 과 HIREDATE를 조회

**SELECT** ENAME, HIREDATE

**FROM** EMP

**WHERE** HIREDATE >= **TO\_DATE**('1982-01-01', 'yyyy-mm-dd') **AND** HIREDATE <= **TO\_DATE**('1982-12-31', 'yyyy-mm-dd');