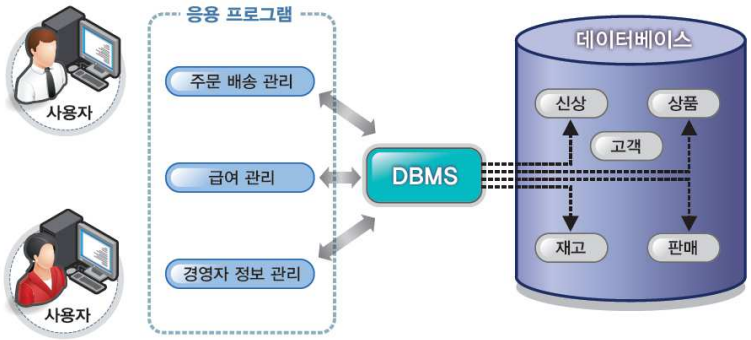
1. Introduction

데이터베이스 시스템

- 데이터베이스
  - 영속적으로 유지 관리되고 공유되어 사용되는 대용량 데이터들의 집합
- 데이터베이스 관리 시스템(DBMS)
  - 데이터베이스를 효율적으로 관리하고 이용할 수 있는 환경을 제공하는 시스템 소프트웨어
  - 데이터 모델 및 데이터 정의 언어(DDL) 제공
    - 데이터베이스의 구조 생성 및 변경 지원
  - 고수준 질의 언어(query language) 제공
    - 데이터에 대한 다양한 질의 처리 지원
  - 영속적인 저장 시스템(persistent storage)
    - 대용량 데이터의 지속적인 관리 및 효율적인 이용 방법 제공
  - 트랜잭션 처리(transaction processing)
    - ACID 특성 지원, 동시성 제어(concurrency control) 및 복구(recovery) 기능 제공

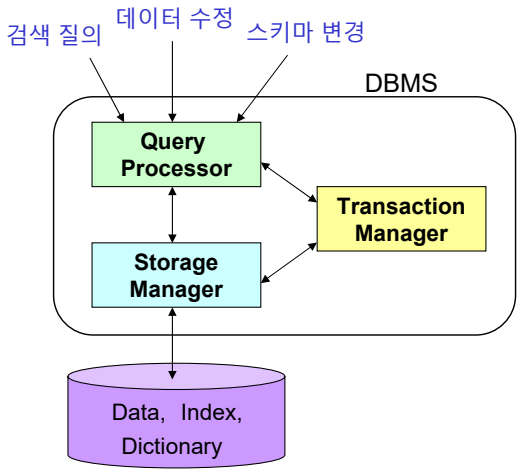
데이터베이스 시스템

- DBMS를 통한 데이터베이스 관리 및 이용



데이터베이스 시스템

- DBMS의 주요 구성요소
  - 질의 처리 시스템
  - 저장소 관리 시스템
  - 트랜잭션 처리 시스템



데이터베이스 시스템

- 관계형 데이터베이스 시스템
  - 관계형 데이터 모델(Relational Data Model) 이용
    - 연관된 테이블들의 집합
  - 고수준 질의 언어를 통해 데이터에 대한 검색 질의 표현
    - SQL(Structured Query Language)

```
SELECT customer_name
FROM   Customer
WHERE acc_number =  'A-101'
```

Customer Table

Column/Attribute/Field

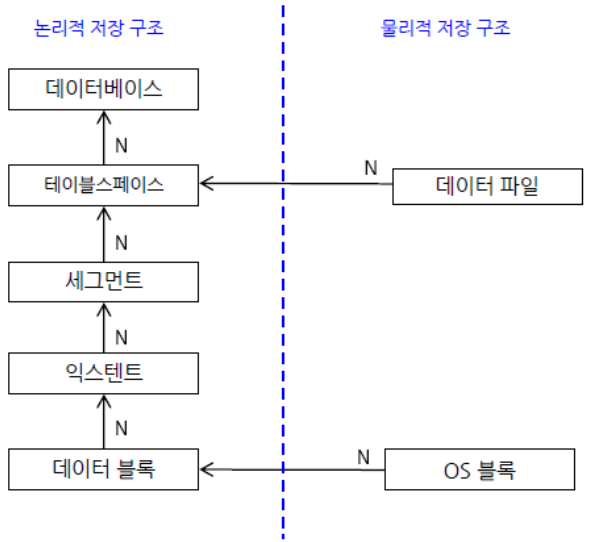
customer _id	customer_ name	address_ street	address_ city	acc_ number
192-83-7465	Johnson	Alma	Palo Alto	A-101
019-28-3746	Smith	North	Rye	A-215
192-83-7465	Johnson	Alma	Palo Alto	A-201
321-12-3123	Jones	Main	Harrison	A-217
019-28-3746	Smith	North	Rye	A-201

Row/ Tuple/ Record

Oracle DBMS

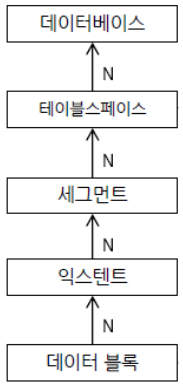
- Oracle DBMS
  - Oracle Corp.에서 개발한 객체 관계형 데이터베이스 관리 시스템
  - 가장 높은 시장 점유율과 신뢰도 확보
  - 다양한 하드웨어 및 운영체제 지원
    - UNIX, LINUX, MS Windows등 대부분의 운영체제를 지원
  - Cloud computing platform 지원
  - Multitenant Architecture 지원
    - 하나의 Container DB가 여러 개의 Pluggable DB들을 포함 가능
  - 버전
    - Oracle Database 11g (2007)
    - Oracle Database 12c (2013)
    - Oracle Database 18c (2018), 19c (2019), 21c (2021), 23ai (2024)

Oracle 데이터 저장 구조



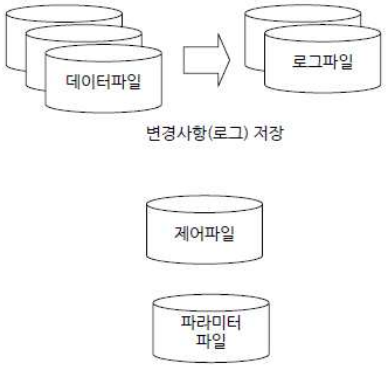
논리적 저장 구조

- Database
  - 하나 이상의 tablespace를 포함
- Tablespace
  - Table, index, view 등 스키마 객체들을 저장
  - SYSTEM, USERS, TEMP 등으로 구분됨
    - 성격이 다른 데이터들을 분리해서 저장 관리
  - Tablespace는 하나 이상의 segment를 포함
  - 물리적으로 하나 이상의 data file들로 구성됨
- Segment
  - 특정 유형의 스키마 객체를 저장하기 위해 할당되는 extent들의 집합
- Extent
  - 연속된 data block들의 집합
- Data block (또는 Page)
  - 데이터의 최종적인 저장 구조
  - 입출력 연산의 단위



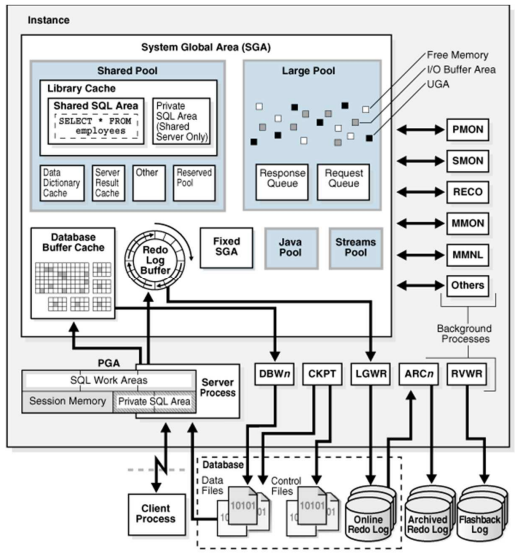
물리적 저장 구조

- Data file
  - 데이터를 저장
  - 데이터베이스 당 하나 이상 필요
- REDO log file
  - DB에 대한 변경 내역을 기록하고 고장 회복에 사용
  - 데이터베이스 당 두 개 이상 필요
- Control file
  - DB 이름, data file, REDO log file 이름과 위치 정보 등을 저장
  - 데이터베이스 당 하나 이상 필요
- Parameter file
  - Oracle 환경 구성을 위한 parameter 설정 값을 저장



Oracle Database Instance

- Processes & Memory structure



Oracle Process

- 서버 프로세스(server process)
  - 클라이언트의 요청을 처리하는 프로세스
  - 필요한 데이터를 SGA Buffer Cache를 통해 디스크(DB)로부터 읽음
- 백그라운드 프로세스(background process)
  - 필수 백그라운드 프로세스
    - DB 쓰기 프로세스(DBWR: database writer process)
    - 로그 쓰기 프로세스(LGWR: log writer process)
    - 시스템 모니터 프로세스(SMON: system monitor process)
    - 프로세스 모니터 프로세스(PMON: process monitor process)
  - 기타 프로세스
    - 아카이브 프로세스(ARCH: archiver)
    - 체크포인트 프로세스(CKPT: checkpoint process)
    - 고장회복 프로세스(RECO: recovery process)
    - 잠금 프로세스(LCK: lock process)

SGA(System Global Area)

- 서버 내의 여러 프로세스들이 공통으로 사용하는 메모리 영역

구성 요소	기능
버퍼 캐시 (buffer cache)	• 데이터베이스 내의 데이터를 블록 단위로 저장 • 서버 프로세스는 디스크로부터 블록들을 읽어 들임 • 버퍼 내의 변경 블록은 추후 DBWR 프로세스에 의해 디스크에 저장됨
REDO 로그 버퍼 (log buffer)	• 데이터베이스에 대한 변경 내역인 로그 레코드를 메모리에 유지 • LGWR에 의해 REDO 로그 파일에 기록됨
공유 풀 (shared pool)	• Library cache: 사용자가 실행한 SQL 질의의 분석정보 및 실행계획을 저장 • Data dictionary cache : 테이블, 인덱스, 함수, 트리거 등 Oracle 스키마 객체들에 대한 정보를 저장
대형 풀 (large pool)	• 선택적 영역으로, 대량의 I/O를 처리하기 위한 메모리 영역
자바 풀 (java pool)	• 자바 저장 프로시저(java stored procedure) 또는 SQLJ와 같은 서버 내의 자바 응용을 위한 영역
스트림 풀 (streams pool)	• DB 간의 데이터 이동을 처리하는 스트림 기능을 위한 영역

PGA(Program Global Area)

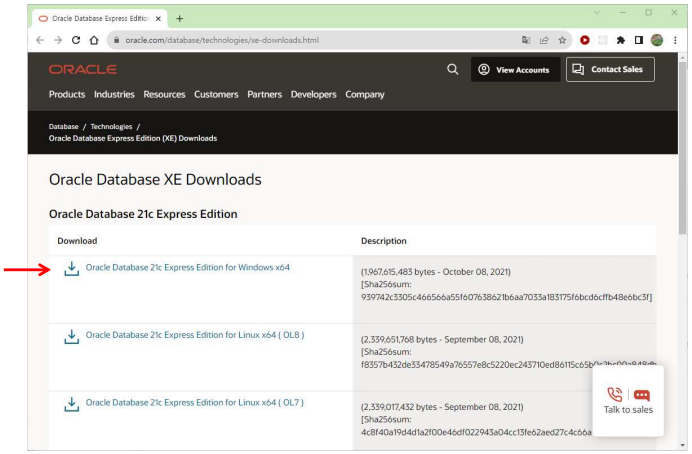
- 클라이언트에서 전달된 요청을 처리하기 위해 서버 프로세스 별로 독립적으로 사용하는 메모리 영역
  - 세션 정보(session information)
    - 클라이언트 접속정보, 사용자 정보와 같은 세션 정보 유지
  - 커서 상태 정보(cursor status)
    - 커서를 이용하여 다수의 레코드를 차례로 방문하는 경우, 현재 커서의 상태에 대한 정보
  - 변수 저장 공간(stack space)
    - SQL 문 안에서 사용되는 bind 변수 값을 저장
  - 정렬 공간(sort area)

Oracle Database Express Edition(XE)

- 개발자를 위한 무료 데이터베이스 시스템
  - Oracle Database 21c와 호환되고 주요 기능을 제공
  - Windows x64, Linux x64 운영체제 지원
  - 설치 크기가 작고 시스템 자원을 적게 사용
- 리소스 제약
  - Up to 12 GB of user data
  - Up to 2 GB of database RAM
  - Up to 2 CPU threads
  - Up to 3 Pluggable Databases
- 홈페이지
  - https://www.oracle.com/database/technologies/appdev/xe.html

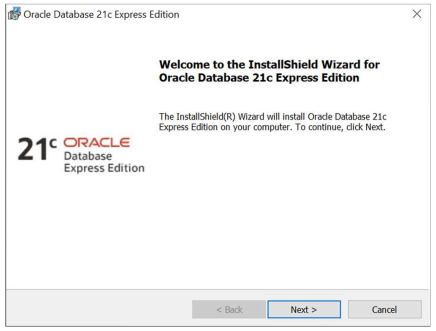
Oracle Database XE

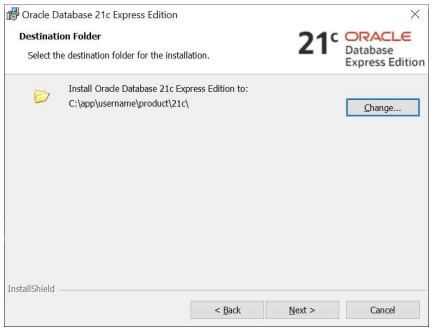
- 다운로드
  - https://www.oracle.com/database/technologies/xe-downloads.html



Oracle Database XE

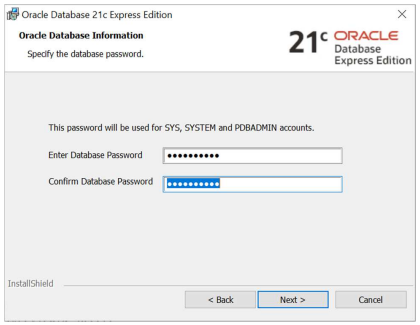
- 설치
  - 관리자 권한이 있는 계정으로 로그인 후 다운로드 실행
  - 다운로드된 파일의 압축을 풀고 setup.exe 파일을 실행하여 설치 시작
  - 설치 폴더 지정/변경

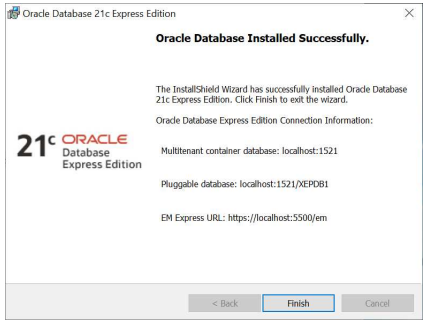




Oracle Database XE

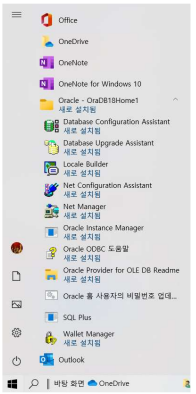
- 설치
  - 시스템 관리자 계정(SYS, SYSTEM)을 위한 암호 설정
  - 자세한 사항은 Installation Guide 참조
    - https://docs.oracle.com/en/database/oracle/oracle-database/21/xeinw/installing-oracle-database-xe.html

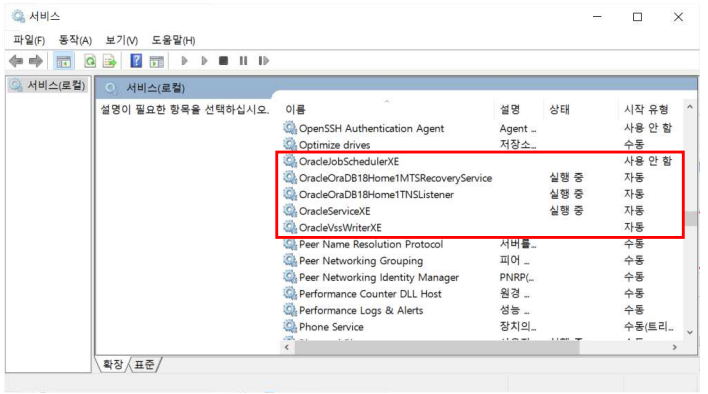




Oracle Database XE

- Windows 메뉴 및 Service
  - OracleServiceXE, TNSListener 서비스 실행 상태 확인





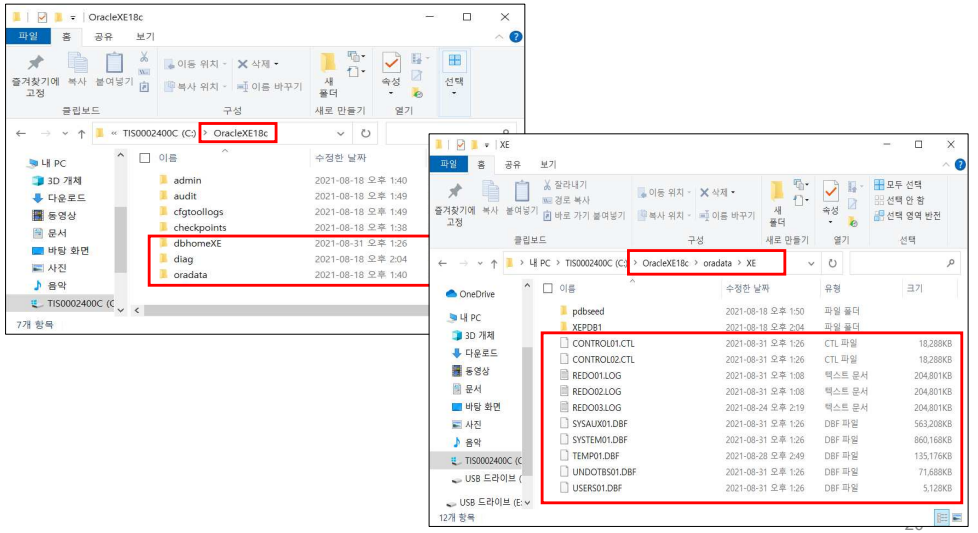
Oracle Database XE

- 관련 폴더

File Name and Location	Purpose
<INSTALL_DIR>	Oracle Base This is the root of the Oracle Database XE directory tree.
<INSTALL_DIR>\dbhomeXE	Oracle Home This home is where the Oracle Database XE is installed. It contains the directories of the Oracle Database XE executables and net work files.
<INSTALL_DIR>\oradata\XE	Database files
<INSTALL_DIR>\diag\rdbms\XE\XE\trace	Diagnostic logs The database alert log is <INSTALL_DIR>\diag\rdbms\XE\XE\trace\alert_XE.log
<INSTALL_DIR>\cfgtoollogs\	Database installation, creation, and configuration logs. The <INSTALL_DIR>\cfgtoollogs\dbca\XE\XE.log file contains the results of the database creation script execution.

Oracle Database XE

- 관련 폴더 및 데이터베이스 파일



Oracle Database XE

- 예제 계정 및 테이블 생성
  - <INSTALL_DIR>\dbhomeXE\rdbms\admin\scott.sql 파일 수정
    - 27행: CONNECT SCOTT/TIGER → CONNECT SCOTT/TIGER@localhost/xepdb1
    - 65, 71행 5번째 컬럼 값 변경: to_date('13-7-1987','dd-mm-yyyy')
  - 명령 프롬프트(cmd: 관리자 권한으로 실행)에서 다음 실행

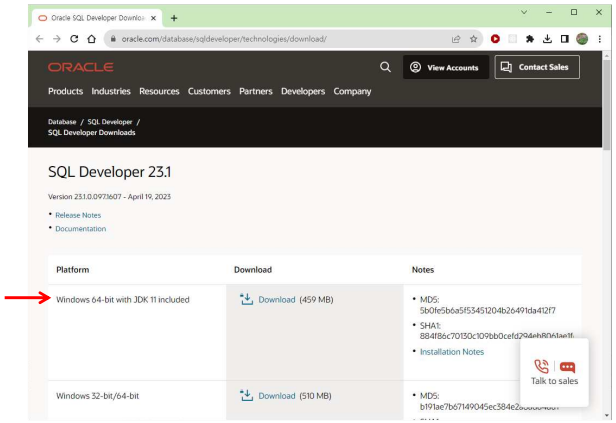
```
C:₩> cd <INSTALL_DIR>₩dbhomeXE₩rdbms₩admin                (스크립트가 있는 폴더로 이동)
C:₩<INSTALL_DIR>₩dbhomeXE₩rdbms₩admin> sqlplus system/<password>@localhost/xepdb1
                                  (sqlplus 실행 및 xepdb1의 관리자 계정으로 접속)
```

  - sqlplus에서 다음 실행

```
SQL> host more scott.sql                    (스크립트 파일 내용 확인)
...
SQL> @scott                                 (스크립트 파일 실행)
SQL> conn scott/TIGER@localhost/xepdb1      (생성된 scott 계정으로 접속)
SQL> select * from emp;                     (레코드들이 화면에 출력되는지 확인)
...
SQL> exit
```

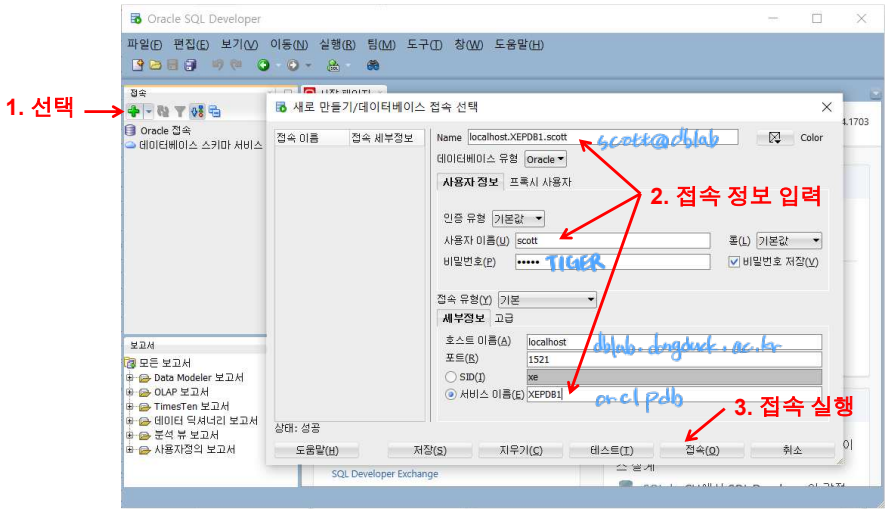
Oracle SQL Developer

- 다운로드 및 설치
  - https://www.oracle.com/tools/downloads/sqldev-downloads.html 에서 Windows 64-bit with JDK 11 included 버전을 선택하여 다운로드
  - Zip 파일을 압축 해제 후 sqldeveloper.exe 실행



Oracle SQL Developer

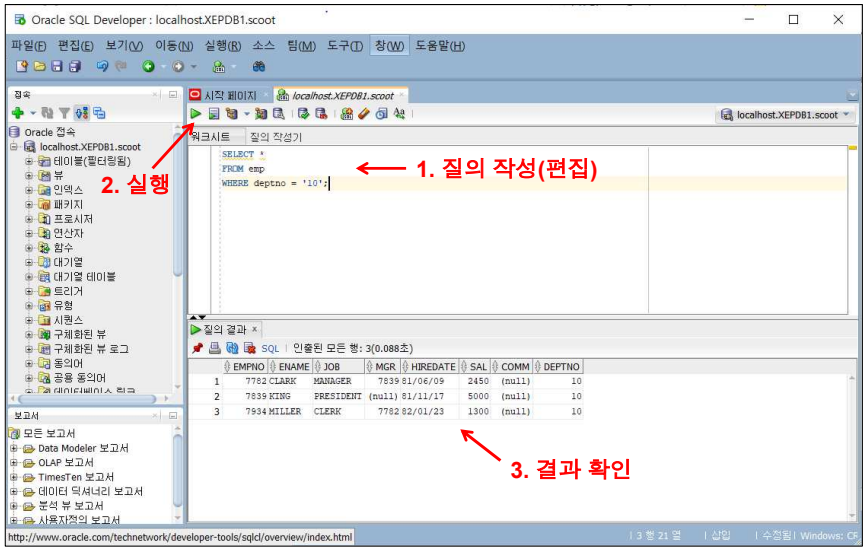
- 데이터베이스 접속



✓ 학과 Oracle Server: 호스트 이름→ dblab.dongduk.ac.kr, 서비스 이름→ orclpdb

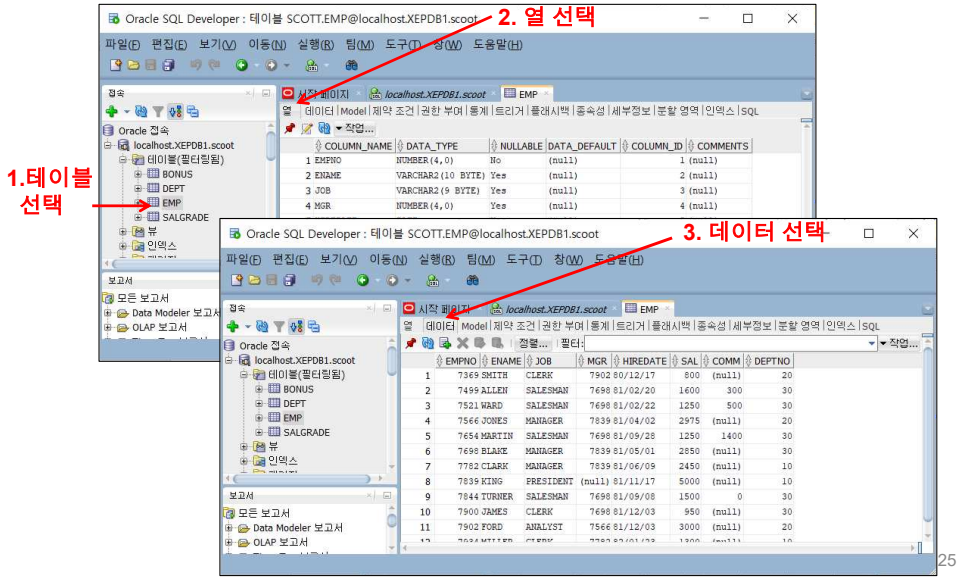
Oracle SQL Developer

- SQL 질의 작성 및 실행



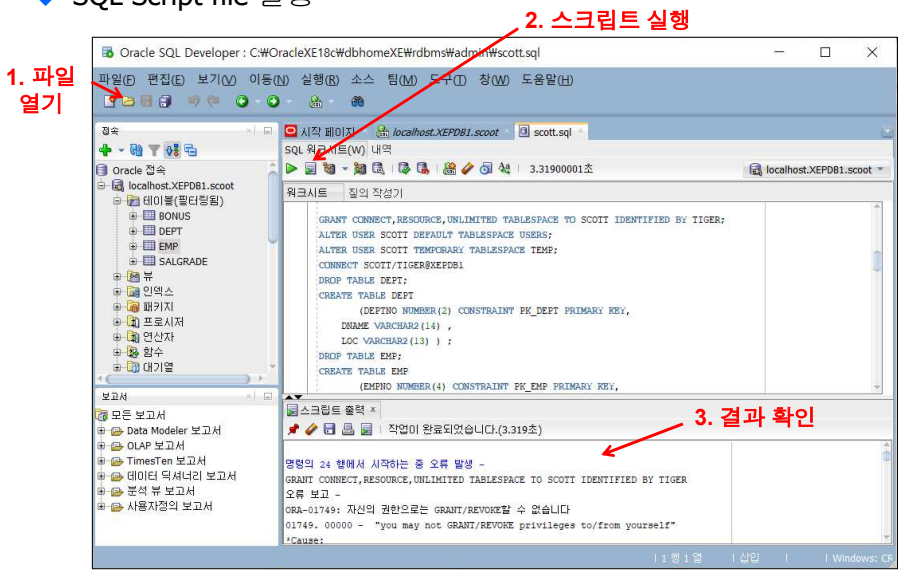
Oracle SQL Developer

- 테이블 구조 및 데이터 확인



Oracle SQL Developer

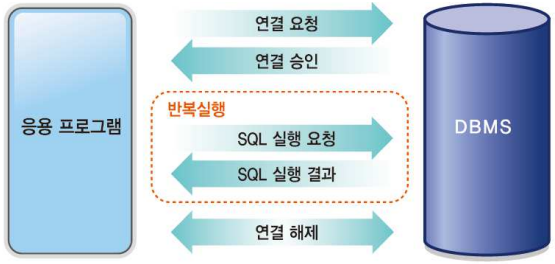
- SQL Script file 실행



데이터베이스 프로그래밍

- DBMS의 데이터베이스 관리 기능을 이용하는 응용 프로그램 개발
  - DBMS를 통해 데이터 저장 관리 및 이용
  - SQL 질의나 명령어를 DBMS에 전송 및 실행시키고 결과를 전달받아 사용

- 응용 프로그램과 DBMS의 연동 과정

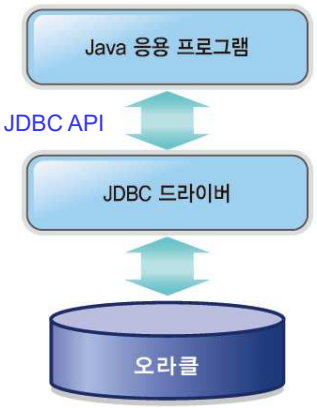


JDBC, ODBC 등 API(Application Programming Interface) 이용

JDBC

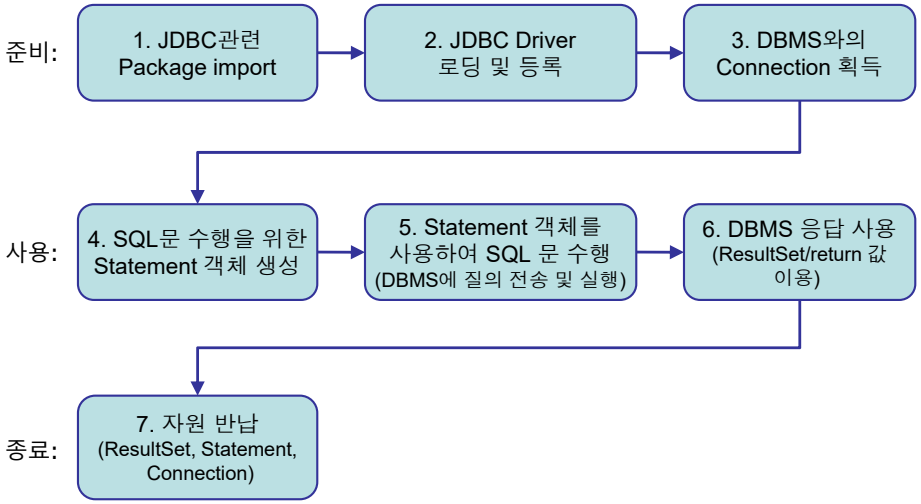
- JDBC(Java Database Connectivity) API
  - Java 응용 프로그램과 DBMS가 연동하기 위해 필요한 과정을 표준화 한 API
  - DBMS 연동에 필요한 메소드들의 이름, 매개변수, 결과 타입 등을 정의

- JDBC Driver
  - 특정 DBMS와 연동할 수 있도록 JDBC API를 구현한 클래스 라이브러리
  - DBMS 제조사(vender)에서 개발 및 배포
    - Oracle JDBC Driver
      - ✓ https://www.oracle.com/database/technologies/appdev/jdbc-downloads.html 에서 Oracle Database 버전에 맞는 프로그램을 다운로드
        - ojdbc8.jar: JDK 8+ 지원
        - ojdbc11.jar: JDK 11+ 지원



JDBC

◆ JDBC API를 이용한 프로그램 구현 방법



JDBC

◆ JDBC API 기반 프로그램 구현 예

```
package dbp.jdbc;
import java.sql.*;                                    // 1. JDBC 관련 package import

public class JdbcTest {
    public static void main(String args[])
    {
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        String url = "jdbc:oracle:thin:@localhost:1521:xe“, user = "scott“, passwd = "TIGER";
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");  // 2. JDBC Driver 로딩 및 등록
        } catch (ClassNotFoundException ex) { ex.printStackTrace(); }
        try {
            conn = DriverManager.getConnection(url, user, passwd);  // 3. DBMS와의 연결
                                                                    //    획득
            String query =  "SELECT * FROM EMP WHERE DEPTNO = 10";   // SQL 질의
            stmt = conn.createStatement();      // 4. SQL문을 위한 Statement 객체 생성
            rs = stmt.executeQuery(query);      // 5. Statement 객체를 사용하여 SQL문 실행
```

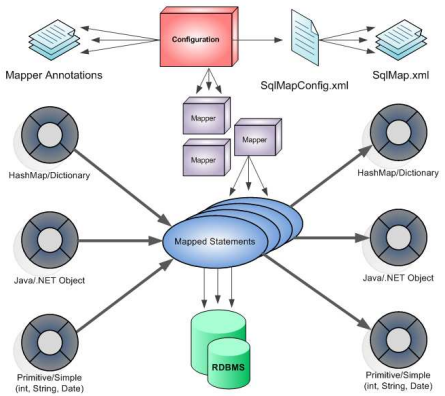
JDBC

```
            System.out.println("No    Name");
            while (rs.next()) {                                  // 6. DBMS 응답 사용
                int no = rs.getInt("EMPNO");
                String name = rs.getString("ENAME");
                System.out.println(no + " " + name);
            }
        } catch (SQLException ex) { ex.printStackTrace(); }
        finally {                                                // 7. 자원 반납
            if (rs != null) {
                try { rs.close(); } catch (SQLException ex) { ex.printStackTrace(); }
            }
            if (stmt != null) {
                try { stmt.close(); } catch (SQLException ex) { ex.printStackTrace(); }
            }
            if (conn != null) {
                try { conn.close(); } catch (SQLException ex) { ex.printStackTrace(); }
            }
        }
    }
}
```

MyBatis

◆ MyBatis

- "Data Mapper Framework"
  - Java 객체와 테이블 레코드 간의 데이터 이동을 처리
- SQL 질의에 대한 parameter mapping과 질의 결과에 대한 result mapping 실행
- XML과 annotation을 이용한 SQL 질의 설정 및 mapping 설정
- MyBatis API와 Mapper interface를 이용한 효율적인 프로그램 작성
  - 내부적으로 JDBC API를 이용해서 실행됨



MyBatis

- Mapper XML을 이용한 프로그램 구현 예

```
<mapper namespace="repository.mapper.EmpMapper">
    <select id="findEmployeesByDeptNo" parameterType="int"
        resultType="model.Employee">
        SELECT EMPNO AS empNo, ENAME AS name, SAL AS salary
        FROM  EMP
        WHERE  DEPTNO = #{deptNo}
    </select>  …
</mapper>
```

```
public class EmployeeRepository {
  private String namespace="repository.mapper.EmpMapper ";
  public List<Employee> getEmployeesInDept(int deptNo) {
        …
        return (List<Employee>)sqlSession.selectList(
                namespace + ".findEmployeesByDeptNo, deptNo);
  } …
}
```


MyBatis

- Mapper Interface를 이용한 프로그램 구현 예

```
package repository.mapper;
import model.Employee;
public interface EmpMapper  {
    @Select( {"SELECT EMPNO AS empNo, ENAME AS name, SAL AS salary",
                "FROM EMP  WHERE DEPTNO = #{deptNo}" } )
     List<Employee> findEmployeesByDeptNo(int deptNo);
}
```

```
import repository.mapper.EmpMapper;
public class EmployeeRepository {
    public List<Employee> getEmployeesInDept(int deptNo) {
        …
        return sqlSession.getMapper(EmpMapper.class).
                findEmployeesByDeptNo(deptNo);
              // EmpMapper interface에 정의된 method를 직접 호출
              // → Employee 객체들의 리스트가 생성 및  반환됨
    }
}
```