JDBC

데이터베이스 연동

충북대학교 컴퓨터교육과
데이터베이스연구실
이선영

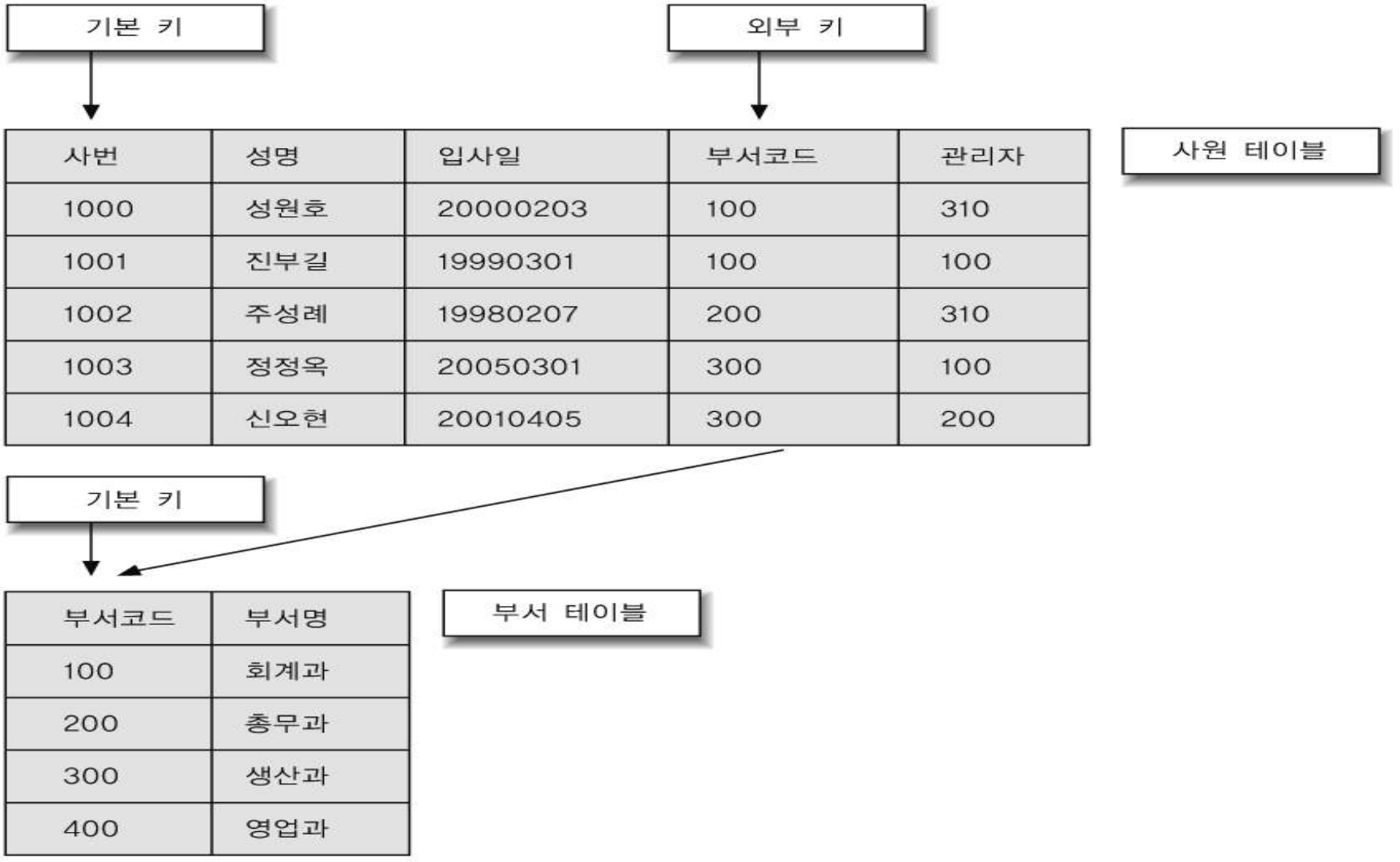
❑ 데이터베이스

- 데이터베이스
  - 지속적으로 저장되는 연관된 정보의 모음
  - 특정 관심의 데이터를 수집하여 그 데이터의 성격에 맞도록 잘 설계하여 저장하고 관리함으로써 필요한 데이터를 효율적으로 사용할 수 있는 자원
- DBMS(Database Management System)
  - 데이터를 효율적으로 관리할 수 있는 시스템
  - 데이터를 데이터베이스에 추가, 삭제, 변경, 검색을 할 수 있는 기능이 있어야 함
- DBMS 종류
  - 계층형, 네트워크용, 릴레이션형
  - 릴레이션형 DBMS : RDBMS
    - Oracle, DB2, MS-SQL, Infomix

❑ 관계형 데이터베이스

- 관계 데이터베이스
  - 관계들의 모임: 테이블 = 개체나 개체 간의 관계
  - 테이블의 행 : 튜플
- 기본 키(Primary Key)
  - 테이블의 각 행을 다른 행과 구분해주는 역할을 하는 필드(열)
  - '유일함' 만족, '값'이 있어야 함
  - NOT NULL 제약 조건과, UNIQUE 제약 조건을 포함
- 외부 키(Foreign Key)
  - 한 테이블의 기본 키에 기반한 관계를 가진 두 개의 테이블이 있는 경우를 위한 것
  - 외부 키는 테이블 내의 한 열의 필드인 동시에 다른 테이블의 기본 키인 열의 필드를 말함

❑ 관계형 데이터베이스


기본 키
외부 키
사번
성명
입사일
부서코드
관리자
1000
성원호
20000203
100
310
1001
진부길
19990301
100
100
1002
주성례
19980207
200
310
1003
정정옥
20050301
300
100
1004
신오현
20010405
300
200
사원 테이블
기본 키
부서코드
부서명
100
회계과
200
총무과
300
생산과
400
영업과
부서 테이블

❑ SQL

- SQL(Standard Query Language)
  - 관계데이터베이스 관리시스템(RDBMS)의 표준 언어
  - SQL문을 이용해서 단순한 쿼리뿐만 아니라 데이터 베이스 객체를 만들거나, 제거하고, 데이터를 삽입, 갱신, 삭제하거나 다양한 운영 작업을 할 수 있음
  - 1970년대 IBM에 의해 발표
    - 이후 ANSI/ISO 표준으로 편입되어 여러 차례의 개량되고 개발됨
  - SQL의 종류
    - DDL : 데이터와 구조를 정의
    - DML : 데이터의 검색과 수정
    - DCL : 데이터 베이스의 권한을 정의

❑ SQL

- DDL과 관련된 SQL문

SQL문	설명
CREATE	데이터베이스 객체를 생성한다.
DROP	데이터베이스의 객체를 삭제한다.
ALTER	기존에 존재하는 데이터베이스의 객체를 다시 정의하는 역할을 한다.

- DML과 관련된 SQL문

SQL문	설명
INSERT	데이터베이스 객체에 데이터를 입력한다.
UPDATE	데이터베이스 객체에 데이터를 갱신한다.
DELETE	데이터베이스 객체에 데이터를 삭제한다.
SELECT	데이터베이스 객체에 데이터로부터 데이터를 검색한다.
COMMIT	커밋 구문 전에 발생한 데이터베이스 액션을 영구히 저장한다.
ROLLBACK	마지막으로 발생한 커밋 후의 데이터베이스 액션들을 원시 데이터로 복구한다.

- DCL과 관련된 SQL문

SQL문	설명
GRANT	데이터베이스 객체에 권한을 부여한다.
REVOKE	이미 부여된 데이터베이스 객체의 권한을 취소한다.

❍ SELECT문

- 데이터베이스로부터 저장되어 있는 데이터를 검색

```
SELECT [ALL | DISTINCT] {*|컬럼,...}
FROM 테이블 명
[WHERE 조건]
[GROUP BY {컬럼,...}]
[HAVING 조건]
[ORDER BY {컬럼,...} [ASC, DESC]]
```

INSERT문

- 테이블을 사용하여 새로운 행 삽입

```
INSERT INTO 테이블명[(컬럼1[, 컬럼2, ... , 컬럼N ])]
VALUES(값1[, 값2, ... , 값N]);
또는
INSERT INTO 테이블명
VALUES(값1[, 값2, ... , 값N]);
```

UPDATE문

- 기존 행을 변경하기 위해서 사용

```
UPDATE 테이블 명
SET 컬럼1 = 값1 [ , 컬럼2 = 값2 , ... , 컬럼N = 값N]
[WHERE 조건];
```

DELETE 문

- 기존 행을 삭제하기 위해서 사용

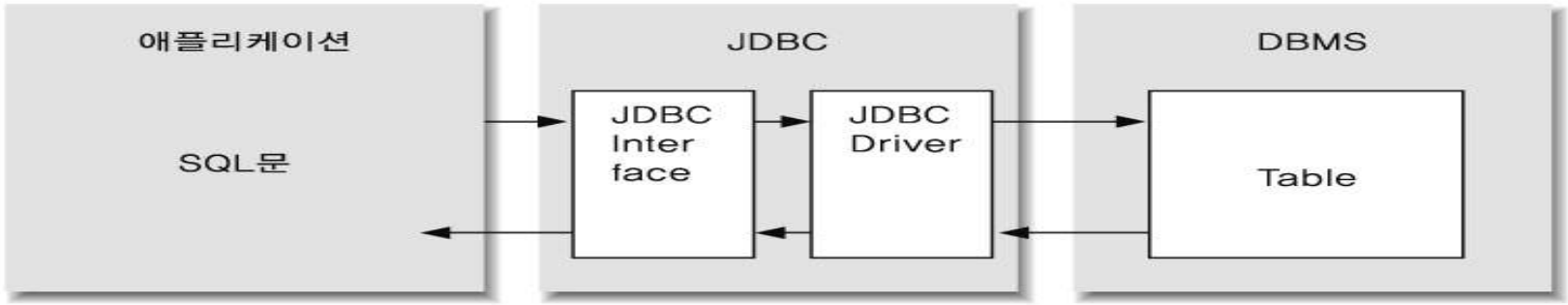
```
DELETE
FROM 테이블 명
[WHERE 조건];
```

❑ JDBC

- JDBC(Java Database Connectivity)
  - 자바를 이용하여 데이터베이스에 접근하여 각종 SQL문을 수행할 수 있도록 제공하는 API
- 각종 DBMS를 통합한 라이브러리 필요성
  - 자바가 데이터베이스에 접근하는 프로그램을 시도할 때 문제점
    - DBMS의 종류가 다양하고, 구조와 특징이 다름
  - 자바는 모든 DBMS에서 공통적으로 사용할 수 있는 인터페이스와 클래스로 구성하는 JDBC를 개발
  - 실제 구현은 DBMS의 밴더에게 구현하도록 함
  - 각 DBMS의 벤더에서 제공하는 구현 클래스를 JDBC 드라이버라고 함
  - JDBC로 코딩하기 위해서는 DBMS를 선택하고, DBMS에서 제공하는 JDBC 드라이버가 반드시 필요

JDBC의 구조와 역할

- JDBC 구성 : JDBC 인터페이스와 JDBC 드라이버로 구성

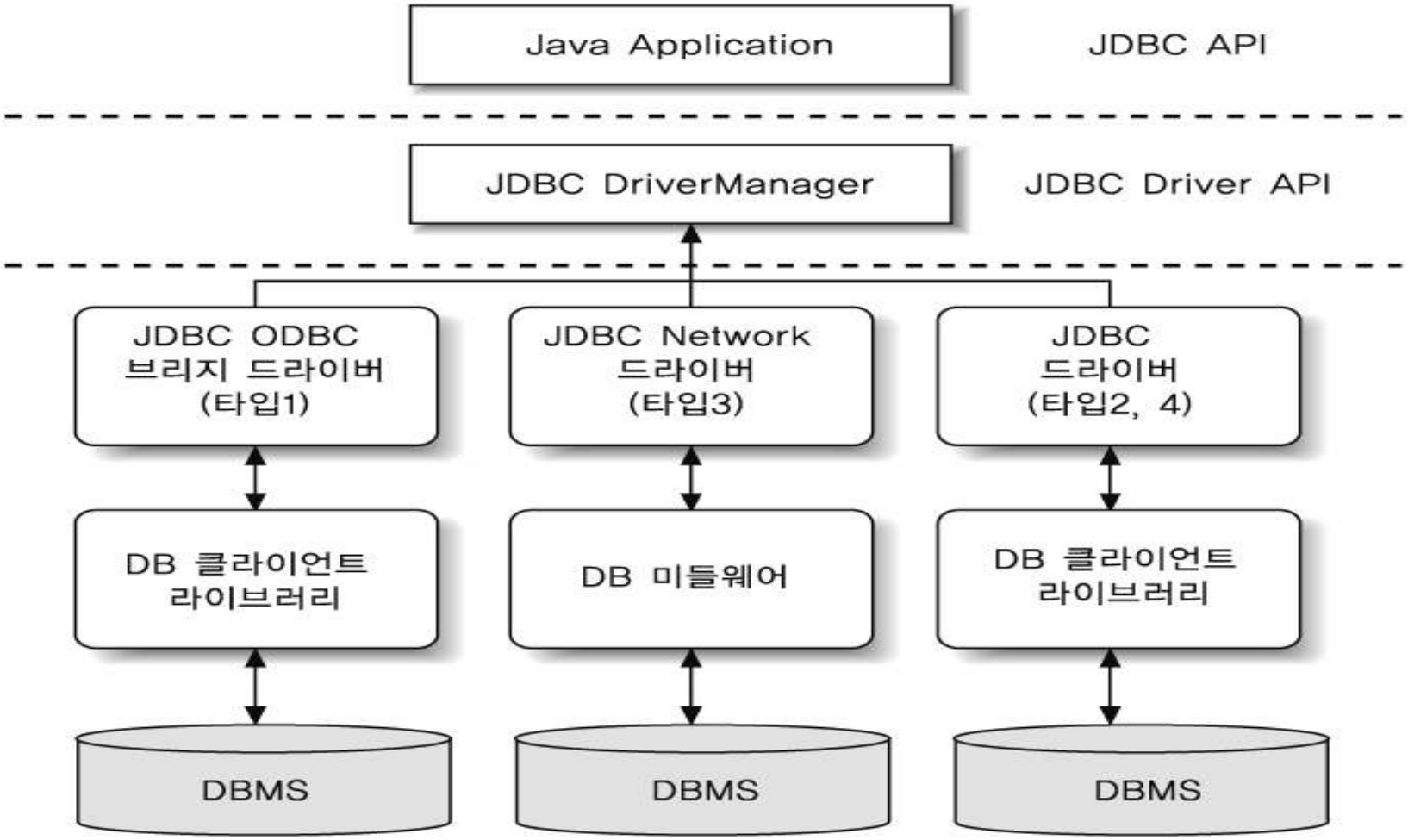


- 응용프로그램에서는 SQL문 만들어 JDBC Interface를 통해 전송하면 실제 구현 클래스인 JDBC 드라이버에서 DBMS에 접속을 시도하여 SQL문을 전송
- DBMS의 결과를 JDBC Driver와 JDBC Interface에게 전달되고 이를 다시 응용프로그램으로 전달 되어 SQL문의 결과를 볼 수 있음
- JDBC의 역할은 Application과 DBMS의 Bridge 역할을 하게 됨

❍ JDBC 드라이버의 종류

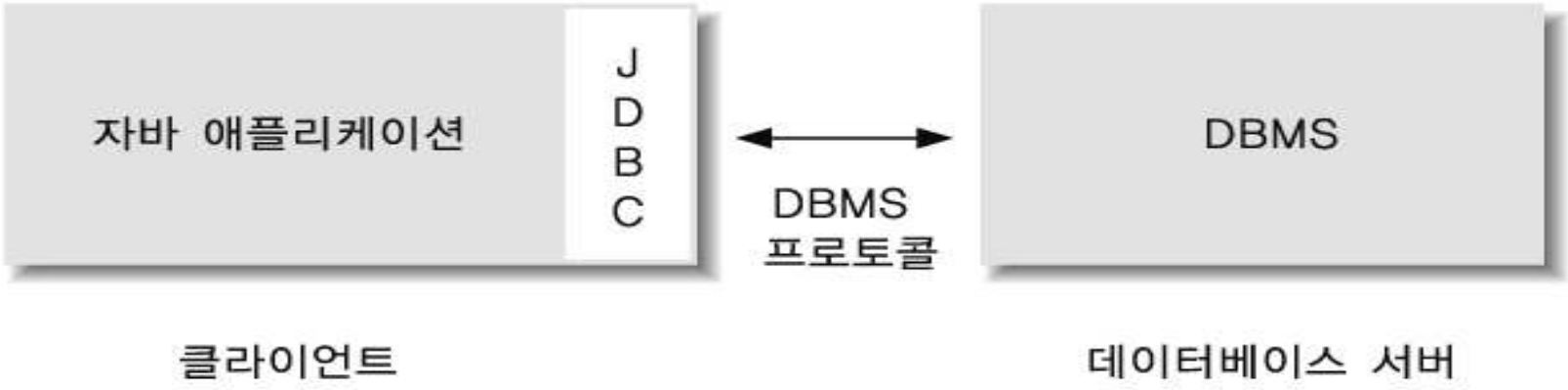
- JDBC-ODBC 브리지 드라이버
  - ➢ JDBC API로 작성된 프로그램이 JDBC-ODBC 브리지를 통해 ODBC 드라이버를 JDBC 드라이버로 여기고 동작하도록 함
  - ➢ 반드시 운영체제 내에 ODBC 드라이버가 존재해야 함
- 데이터베이스 API 드라이버
  - ➢ JDBC API 호출을 특정 데이터베이스의 클라이언트 호출 API로 바꿔주는 드라이버
  - ➢ 오라클 OCI 드라이버가 여기에 속한다.
- 네트워크 프로토콜 드라이버
  - ➢ 클라이언트의 JDBC API 호출을 특정 데이터베이스의 프로토콜과 전혀 상관없는 독자적인 방식의 프로토콜로 바꾸어 서버로 전송
  - ➢ 서버에는 미들웨어가 프로토콜을 특정 데이터베이스 API로 바꾸어 처리
- 데이터베이스 프로토콜 드라이버
  - ➢ JDBC API 호출을 서버의 특정 데이터베이스에 맞는 프로토콜로 변환시켜 서버로 전송하는 드라이버(Java Thin Driver)

❑ JDBC


Java Application
JDBC API
JDBC DriverManager
JDBC Driver API
JDBC ODBC
브리지 드라이버
(타입1)
JDBC Network
드라이버
(타입3)
JDBC
드라이버
(타입2, 4)
DB 클라이언트
라이브러리
DB 미들웨어
DB 클라이언트
라이브러리
DBMS
DBMS
DBMS

2tier

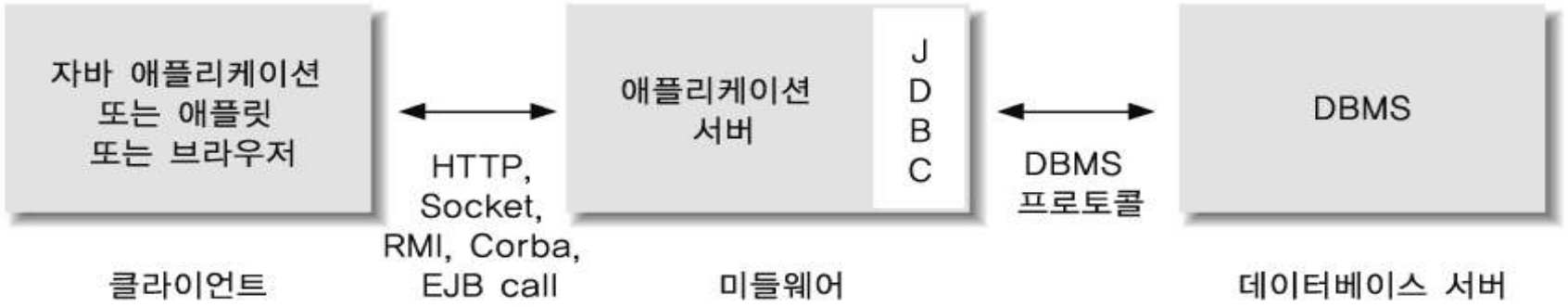
- 자바 애플리케이션이 JDBC 드라이버를 통해서 직접 데이터베이스를 접근하는 형식
- 이 모델에서 JDBC 드라이버는 JDBC API 호출을 통해 특정 DBMS에 직접 전달해 주는 역할을 함



❑ JDBC의 탄생 배경과 구조

❍ 3tier

- 2tier 모델에 미들웨어 계층이 추가된 형태
- 미들웨어에서 DBMS에 직접 질의하게 됨



- 2tier 모델 보다 유지보수(Maintenance) 비용 절약
- 단점: 2tier에 비해 속도는 다소 느림

❑ JDBC를 이용한 데이터베이스 연결방법

- JDBC를 이용한 데이터베이스 연결 방법
  - 1 단계 : import java.sql.*;
  - 2 단계 : 드라이버 로드
  - 3 단계 : Connection 객체 생성
  - 4 단계 : Statement 객체 생성
  - 5 단계 : SQL문에 결과물이 있다면 ResultSet 객체 생성
  - 6 단계 : 모든 객체를 닫음

❑ JDBC를 이용한 데이터베이스 연결방법

❍ 드라이버 다운 받기와 설정

- JDBC API를 이용해서 DBMS에 접근하기 위해서는 DBMS에서 제공되는 드라이버를 내려 받아야 함
- 오라클이 설치되어 있다면 아래의 경로에서 드라이버 제공

```
%ORACLE_HOME%₩ora92₩jdbc₩lib₩classes12.zip
```

- 오라클이 설치되어 있지 않다면 아래의 주소에서 다운받음

```
http://download.oracle.com/otn/utilities_drivers/jdbc/101020/ojdbc14.jar
```

❑ JDBC를 이용한 데이터베이스 연결방법

- ojdbc14.jar 파일이 C 드라이브에 있다는 가정에서 JDBC 드라이버를 설정
  - 이클립스 : 'project에서 오른쪽 클릭 => properties선택 => libraries(탭) 선택 => Add External JARs. 선택 => ojdbc14.jar를 찾아서 <열기> 버튼 클릭'

JDBC API import

- JDBC 코딩하기 위한 첫 번째 단계로 JDBC에서 사용되는 클래스와 인터페이스가 있는 패키지를 import 해야 함

```
import java.sql.*;
public class JdbcEx{
}
```

❑ JDBC를 이용한 데이터베이스 연결방법

❍ 드라이버 로드

- JDBC Driver를 로드

```
try{
        Class.forName("oracle.jdbc.driver.OracleDriver");
}catch(ClassNotFoundException cnfe){
        cnfe.printStackTrace();
}
```

- Class.forName(~) : 동적으로 JDBC 드라이브 클래스를 로딩
  - forName(~) 메서드에 매개변수로 오는 OracleDriver 클래스의 객체를 만들어 런타임 메모리에 로딩시켜 주는 메서드

- 아래와 같이 정의 해도 됨

```
oracle.jdbc.driver.OracleDriver driver =
                new oracle.jdbc.driver.OracleDriver ();
```

❑ JDBC를 이용한 데이터베이스 연결방법

❍Connection 객체 생성

- DBMS와 연결을 담당하는 Connection 객체를 생성

```
try{
        Connection con = DriverManager.getConnection(
        "jdbc:oracle:thin:@localhost:1521:orcl",
                                        "scott",
                                        "tiger");
}catch(SQLException sqle){
        sqle.printStackTrace();
}
```

- Connection 객체를 얻어 왔다면 DBMS와 접속이 성공적으로 이루어진 것임

❑ JDBC를 이용한 데이터베이스 연결방법

- getConnection() 메서드에 들어가는 url, user, password
  - URL-"oracle:jdbc:thin:@ip:port:ORACLE_SID"
    - IP – 오라클이 설치된 ip
    - PORT – oracle의 포트, 1521
    - ORACLE_SID – oracle를 설치할 때 설정하는 것인데 대부분 일정하지 않음. ORACLE_SID를 정확히 찾아 확인한 후 값 정해야 함
  - user – oracle의 user
  - password – oracle user에 대한 password

❑ Connection의 주요 메서드

반환형	메서드	설명
void	close( )	Connection 객체를 해제한다.
	commit( )	트랜잭션으로 설정된 모든 자원을 커밋한다.
Statement	createStatement( )	SQL문을 전송할 수 있는 Statement 객체를 생성한다.
	createStatement (int resultSetType, int resultSetConcurrency)	매개변수로 SQL문을 전송할 수 있는 Statement 객체를 생성한다. 매개변수값을 어떻게 설정하느냐 따라 Statement 객체의 기능이 달라진다.
boolean	getAutoCommit( )	Connection 객체의 현재 auto-commit 상태를 반환한다.
CallableStatement	prepareCall(String sql)	SQL문 전송과 Store Procedure를 호출할 수 있는 CallableStatement 객체를 생성한다.
	prepareCall(String sql, int resultSetType, int resultSetConcurrency)	매개변수로 CallableStatement 객체를 생성한다. 매개변수값을 어떻게 설정하느냐 따라 CallableStatement 객체의 기능이 달라진다.
PreparedStatement	prepareStatement (String sql)	SQL문을 전송할 수 있는 PreparedStatement 객체를 생성한다.
	prepareStatement (String sql, int resultSetType, int resultSetConcurrency)	매개변수로 PreparedStatement 객체를 생성한다. 매개변수값을 어떻게 설정하느냐 따라 CallableStatement 객체의 기능이 달라진다.
void	rollback( )	현재 트랜잭션에 설정된 모든 변화를 되돌린다.
	rollback(Savepoint savepoint)	Savepoint로 설정된 이후의 모든 변화를 되돌린다.
Savepoint	setSavepoint(String name)	현재 트랜잭션에서 name으로 Savepoint를 설정한다.

❑ JDBC를 이용한 데이터베이스 연결방법

❍ Statement 객체 생성

- 오라클과 연결되었다면 SQL문을 전송할 수 있는 Statement 객체를 생성해야 함

```
try{
        Statement stmt = con.createStatement();
}catch(SQLException sqle){
        sqle.printStackTrace();
}
```

- Statement 객체는 Connection 인터페이스의 createStatement() 메서드를 사용하여 얻어 올 수 있음
- Statement 인터페이스에는 SQL문을 전송할 수 있는 여러 가지 메서드 중에 3가지

❑ JDBC를 이용한 데이터베이스 연결방법

- executeQuery(String sql) – SQL문이 select 일 경우

```
Statement stmt = con.createStatement();
StringBuffer sb = new StringBuffer();
sb.append("select id from test ");
ResultSet rs = stmt.executeQuery(sb.toString());
```

- executeUpdate(String sql) – SQL문이 insert, update, delete문 등 일 경우

```
Statement stmt = con.createStatement();
StringBuffer sb = new StringBuffer();
sb.append("update test set id='syh1011' ");
int updateCount = stmt.executeUpdate(sb.toString());
```

❑ JDBC를 이용한 데이터베이스 연결방법

- execute(String sql) – SQL문을 알지 못하는 경우

```
Statement stmt = con.createStatement();
StringBuffer sb = new StringBuffer();
sb.append("update test set id='syh5055'");
boolean isResult = stmt.execute(sb.toString());
if(isResult){
        ResultSet rs = stmt.getResultSet();
        while(rs.next()){
                System.out.println("id : "+ rs.getString(1));
        }
}else{
        int rowCount = stmt.getUpdateCount();
        System.out.println("rowCount : "+ rowCount);
}
```

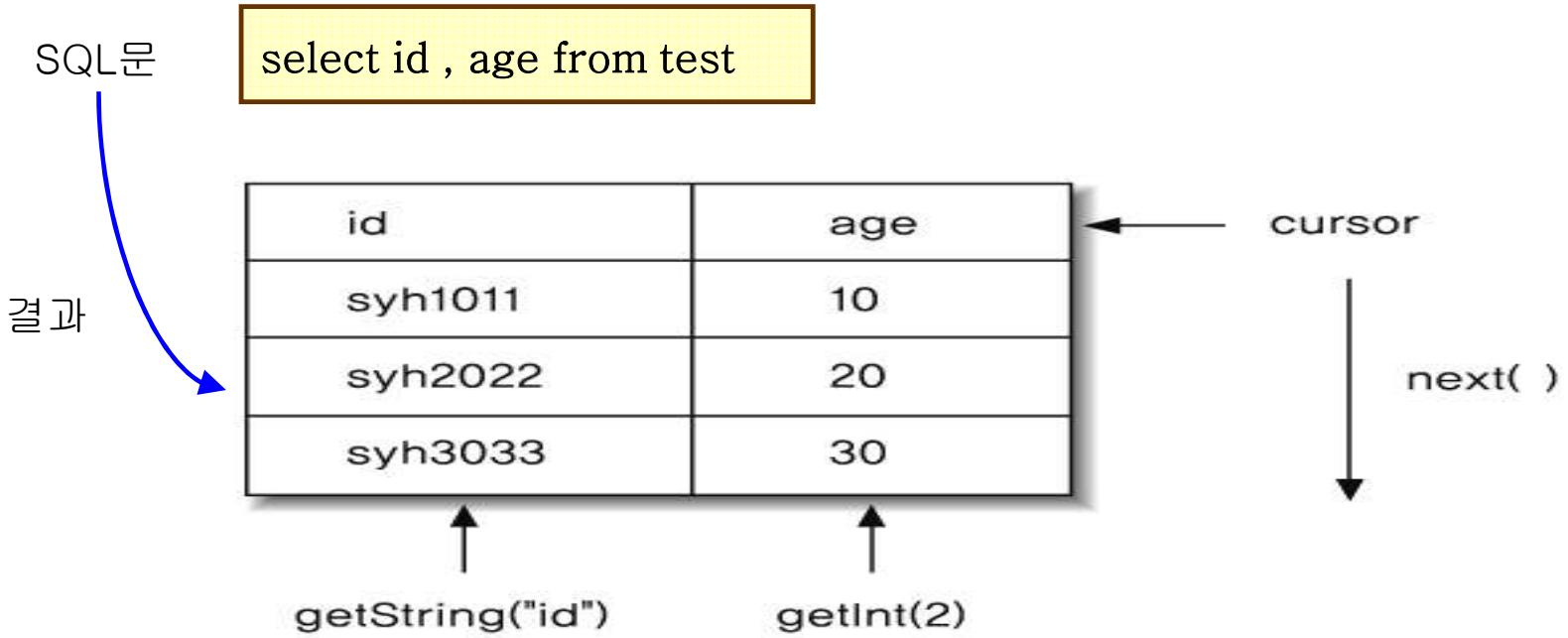
❑ Statement의 주요 메서드

반환형	메서드	설명
void	addBatch(String sql)	Statement 객체에 SQL문을 추가한다. 이 메서드를 이용해서 SQL의 일괄처리를 할 수 있다.
	clearBatch( )	Statement 객체에 모든 SQL문을 비운다.
	close( )	Statement 객체를 해제한다.
boolean	execute(String sql)	매개변수인 SQL문을 수행한다. 만약, 수행한 결과가 ResultSet 객체를 반환하면 true, 어떠한 결과도 없거나, 갱신된 숫자를 반환하면 false를 반환한다.
int[ ]	executeBatch( )	Statement 객체에 추가된 모든 SQL문을 일괄처리한다. 일괄처리된 각각의 SQL문에 대한 결과값을 int[ ]로 반환한다.
ResultSet	exectueQuery(String sql)	매개변수인 SQL문을 수행하고 ResultSet 객체를 반환한다.
int	executeUpdate(String sql)	매개변수인 SQL문을 수행한다. SQL문은 INSERT문, UPDATE문, CREATE문, DROP문 등을 사용한다.
ResultSet	getResultSet( )	ResultSet 객체를 반환한다.

❑ JDBC를 이용한 데이터베이스 연결방법

❍ ResultSet 객체 생성

- ResultSet은 SQL문에 대한 결과를 처리할 수 있는 객체
- Statement 인터페이스의 executeQuery() 메서드를 실행한 결과로 ResultSet 객체를 리턴 받음



❑ JDBC를 이용한 데이터베이스 연결방법

- ResultSet 인터페이스에는 결과물(필드)을 가져오는 수많은 메서드(getXXX())를 제공
  - getXXX() 메서드 : oracle의 자료형 타입에 따라 달라지게 됨
  - id 컬럼이 varchar2 타입이라면 getString(~) 메서드를 사용
  - age 컬럼이 number 타입이라면 getInt(~) 메서드를 사용
- 모든 데이터를 한번에 가져올 수 없기 때문에 cursor의 개념을 가짐
  - Cursor : ResultSet 객체가 가져올 수 있는 행을 지정해 줌
  - 처음 커서의 위치는 결과물(필드)에 위치하지 않기 때문에 cusror를 이동해야 함
  - 커서를 이동하는 메서드 : ResultSet의 next() 메서드
  - next() 메서드의 리턴 타입
    - boolean : 다음 행의 결과물(필드)이 있으면 true, 없으면 false를 리턴

❑ ResultSet의 주요 메서드

반환형	메서드	설명
boolean	absolute(int row)	ResultSet 객체에서 매개변수 row로 커서를 이동한다. 만약, 매개변수 row로 커서를 이동할 수 있으면 true, 그렇지 않으면 false를 반환한다.
void	afterLast( )	ResultSet 객체에서 커서를 마지막 로우 다음으로 이동한다.
	beforeFirst( )	ResultSet 객체에서 커서를 처음 로우 이전으로 이동한다.
boolean	last( )	ResultSet 객체에서 커서를 마지막 로우로 이동한다. 만약, ResultSet에 row가 있다면 true, 그렇지 않으면 false를 반환한다.
	next( )	ResultSet 객체에서 현재 커서에서 다음 로우로 커서를 이동한다. 만약, ResultSet에 다음 row가 있다면 true, 그렇지 않으면 false를 반환한다.
	previous( )	ResultSet 객체에서 현재 커서에서 이전 로우로 커서를 이동한다. 만약, ResultSet에 이전 row가 있다면 true, 그렇지 않으면 false를 반환한다.
void	close( )	ResultSet 객체를 해제한다.
boolean	first( )	ResultSet 객체에서 커서를 처음 로우로 이동한다. 만약, ResultSet에 row가 있다면 true, 그렇지 않으면 false를 반환한다.
InputStream	getBinaryStream (int columnIndex)	ResultSet 객체의 현재 로우에 있는 columnIndex의 값을 InputStream으로 반환한다.
	getBinaryStream (String columnName)	ResultSet 객체의 현재 로우에 있는 columnName의 값을 InputStream으로 반환한다.
Blob	getBlob (int columnIndex)	ResultSet 객체의 현재 로우에 있는 columnIndex의 값을 Blob으로 반환한다.
	getBlob (String columnName)	ResultSet 객체의 현재 로우에 있는 columnName의 값을 Blob으로 반환한다.
byte	getByte (int columnIndex)	ResultSet 객체의 현재 로우에 있는 columnIndex의 값을 byte로 반환한다.
	getByte (String columnName)	ResultSet 객체의 현재 로우에 있는 columnName의 값을 byte로 반환한다.
Clob	getClob (int columnIndex)	ResultSet 객체의 현재 로우에 있는 columnIndex의 값을 Clob으로 반환한다.
	getClob (String columnName)	ResultSet 객체의 현재 로우에 있는 columnName의 값을 Clob으로 반환한다.
double	getDouble (int columnIndex)	ResultSet 객체의 현재 로우에 있는 columnIndex의 값을 double로 반환한다.
	getDouble (String columnName)	ResultSet 객체의 현재 로우에 있는 columnName의 값을 double로 반환한다.
int	getInt (int columnIndex)	ResultSet 객체의 현재 로우에 있는 columnIndex의 값을 int로 반환한다.
	getInt (String columnName)	ResultSet 객체의 현재 로우에 있는 columnName의 값을 int로 반환한다.
String	getString (int columnIndex)	ResultSet 객체의 현재 로우에 있는 columnIndex의 값을 String으로 반환한다.
	getString (String columnName)	ResultSet 객체의 현재 로우에 있는 columnName의 값을 String으로 반환한다.

❑ JDBC를 이용한 데이터베이스 연결방법

- 모든 객체를 닫는다.
  - Connection, Statement, ResultSet 객체는 사용이 끝난 후에는 종료를 해 줘야 함
  - 종료 해주는 메서드는 모든 객체에 close() 메서드로 정의

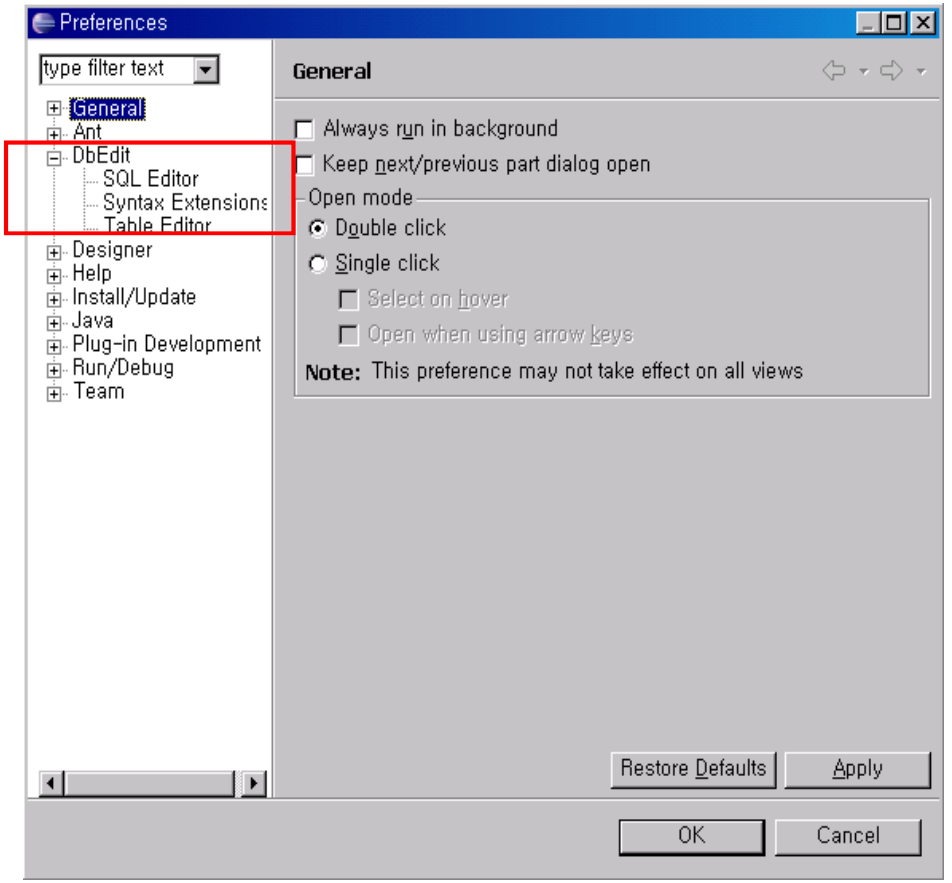
❑ 이클립스용 데이터베이스 접근 플러그인

- DBEdit
  - JDBC 드리아버를 지원하는 대부분의 데이터베이스에 대해 테이블 생성, 데이터 추가, 삭제, 검색, 수정등 일반적인 쿼리 작업 수행
  - http://www.geocities.com/uwe_ewald/dbedit.html
- JFaceDbc
  - 독립 애플리케이션으로도 실행 가능
  - http://jfacedbc.sourceforge.net
- Quantum
  - DBEdit에 비해 데이터베이스 관리 기능이 빠짐
  - 깔끔한 인터페이스 제공
  - http://quantum.sourceforge.net
- EclDBTool
  - http://ecldbtool.sourceforge.net

❑ DBEdit

❍ 설치

- DBEdit 플러그인 다운
- 이클립스 디렉터리의 plugins 디렉토리에 압축을 품
- Preferences 다이얼로그에 DBEdit 항목 생성



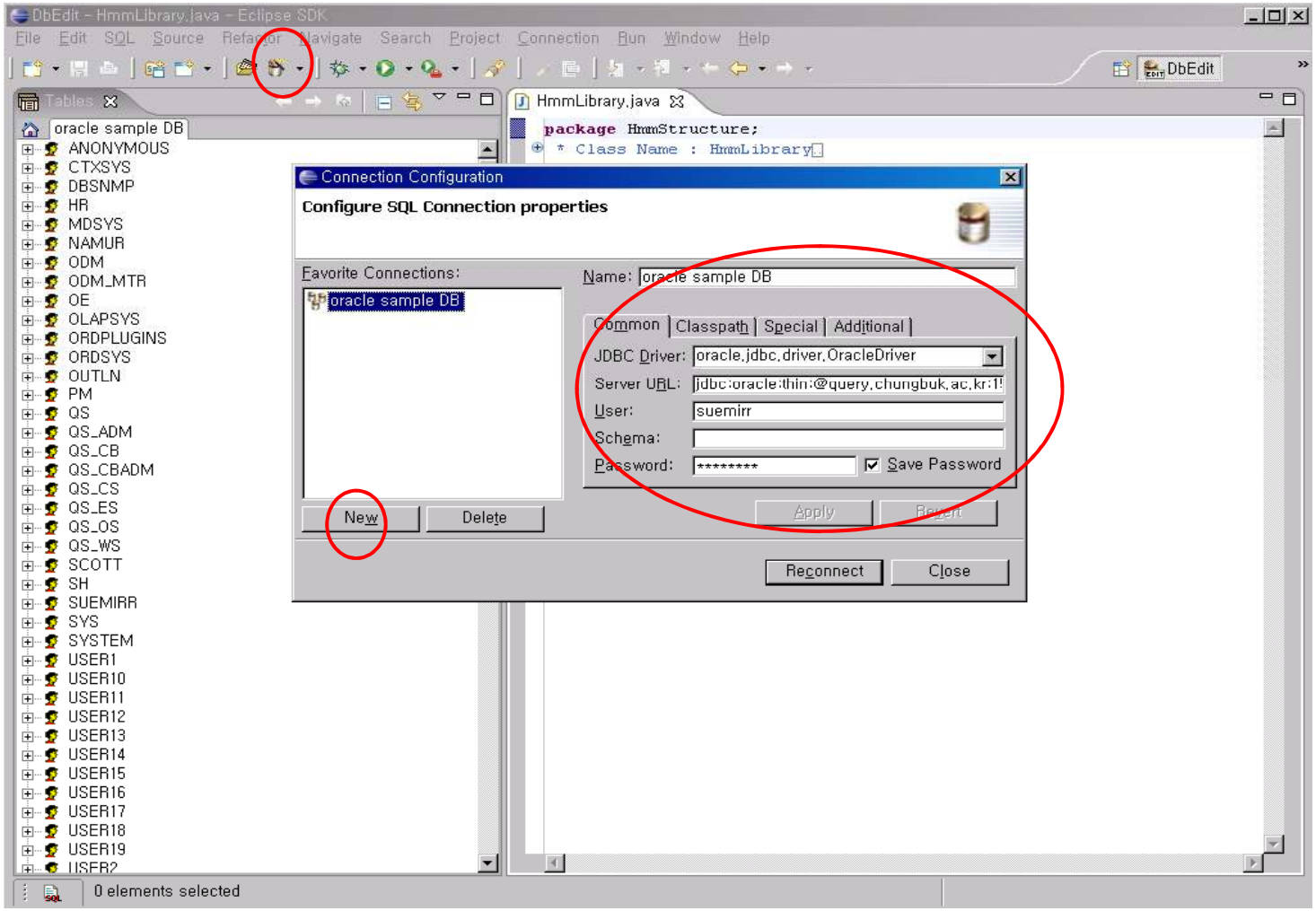
❑ DBEdit

❍ 연결 설정

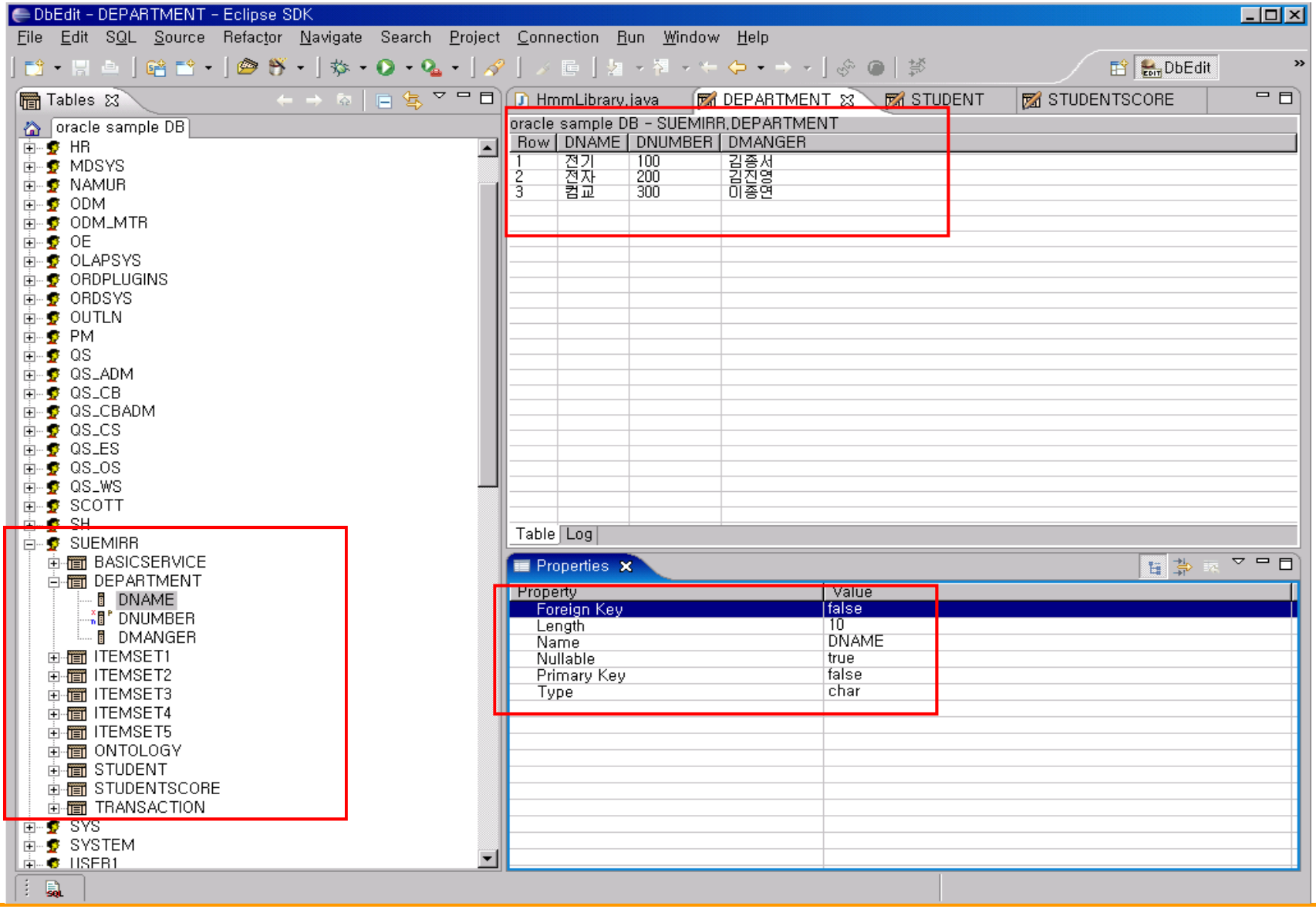
- JDBC Driver: oracle.jdbc.driver.OracleDriver
- Server URL :
  jdbc:oracle:thin:@220.73.234.30:1521:ora91

- User : system
- Password : oracle (Save Password Check)
- Classpath : Add Archive: 오라클 드라이버 경로 지정
- G:/jsp/eclipse/lib/classes12.jar

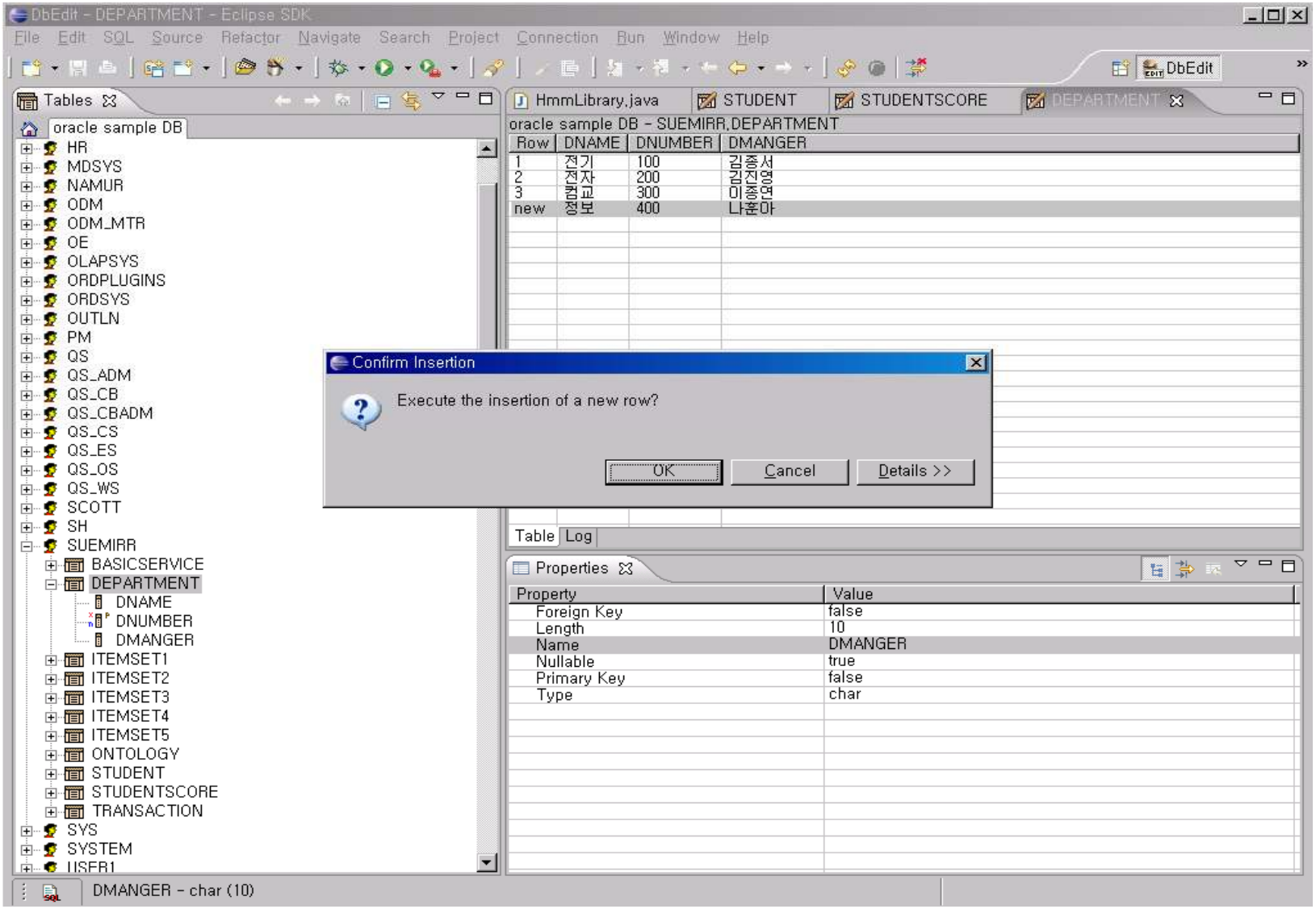
Window>Open perspective>Other



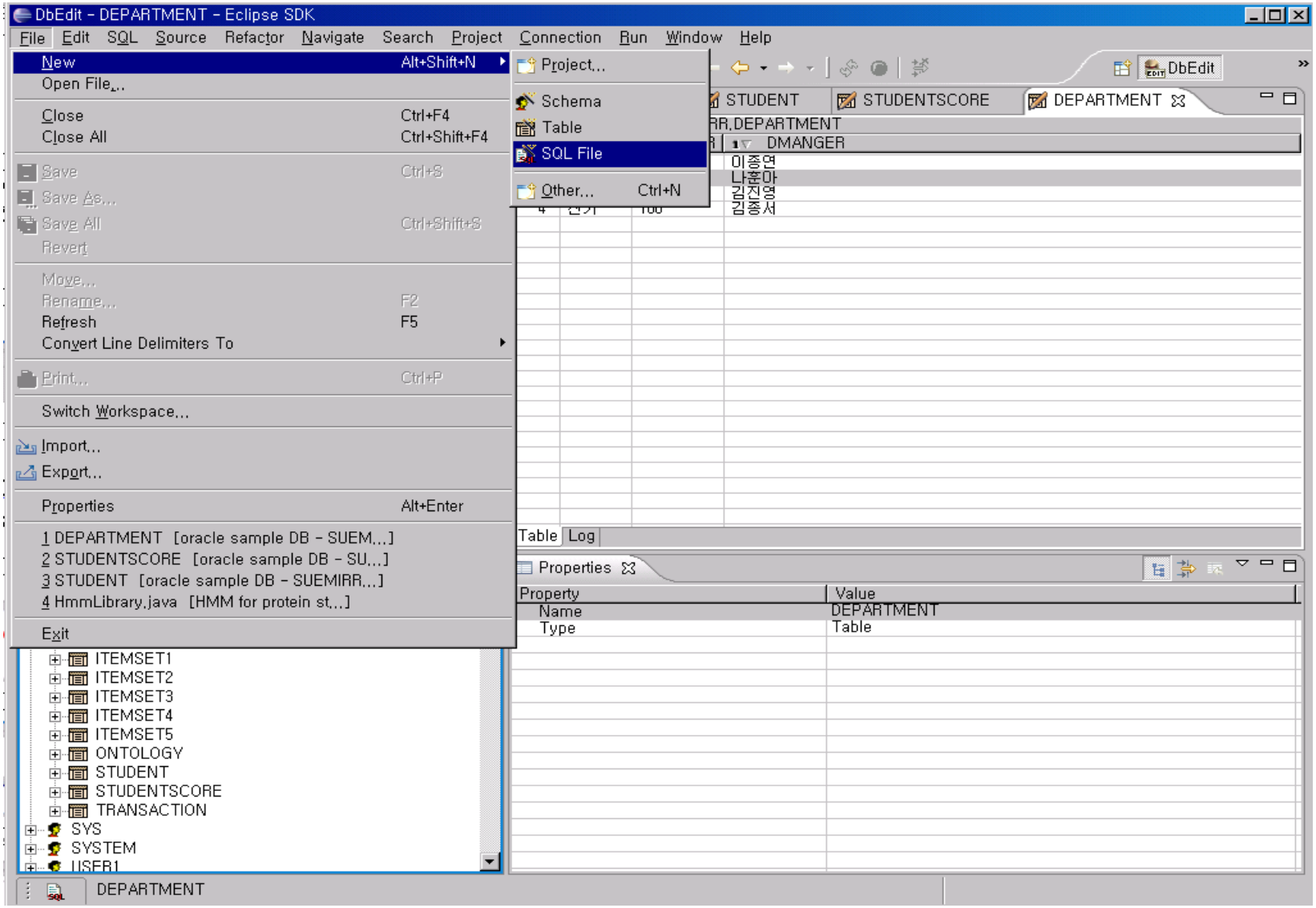
❑ 데이터베이스 확인


DbEdit - DEPARTMENT - Eclipse SDK
File Edit SQL Source Refactor Navigate Search Project Connection Run Window Help
DbEdit
Tables
oracle sample DB
HR
MDSYS
NAMUR
ODM
ODM_MTR
OE
OLAPSYS
ORDPLUGINS
ORDSYS
OUTLN
PM
QS
QS_ADM
QS_CB
QS_CBADM
QS_CS
QS_ES
QS_OS
QS_WS
SCOTT
SH
SUEMIRR
BASICSERVICE
DEPARTMENT
DNAME
DNUMBER
DMANGER
ITEMSET1
ITEMSET2
ITEMSET3
ITEMSET4
ITEMSET5
ONTOLOGY
STUDENT
STUDENTSCORE
TRANSACTION
SYS
SYSTEM
USER1
HmmLibrary.java
DEPARTMENT
STUDENT
STUDENTSCORE
oracle sample DB - SUEMIRR.DEPARTMENT
Row DNAME DNUMBER DMANGER
1 전기 100 김종서
2 전자 200 김진영
3 컴교 300 이종연
Table Log
Properties
Property Value
Foreign Key false
Length 10
Name DNAME
Nullable true
Primary Key false
Type char

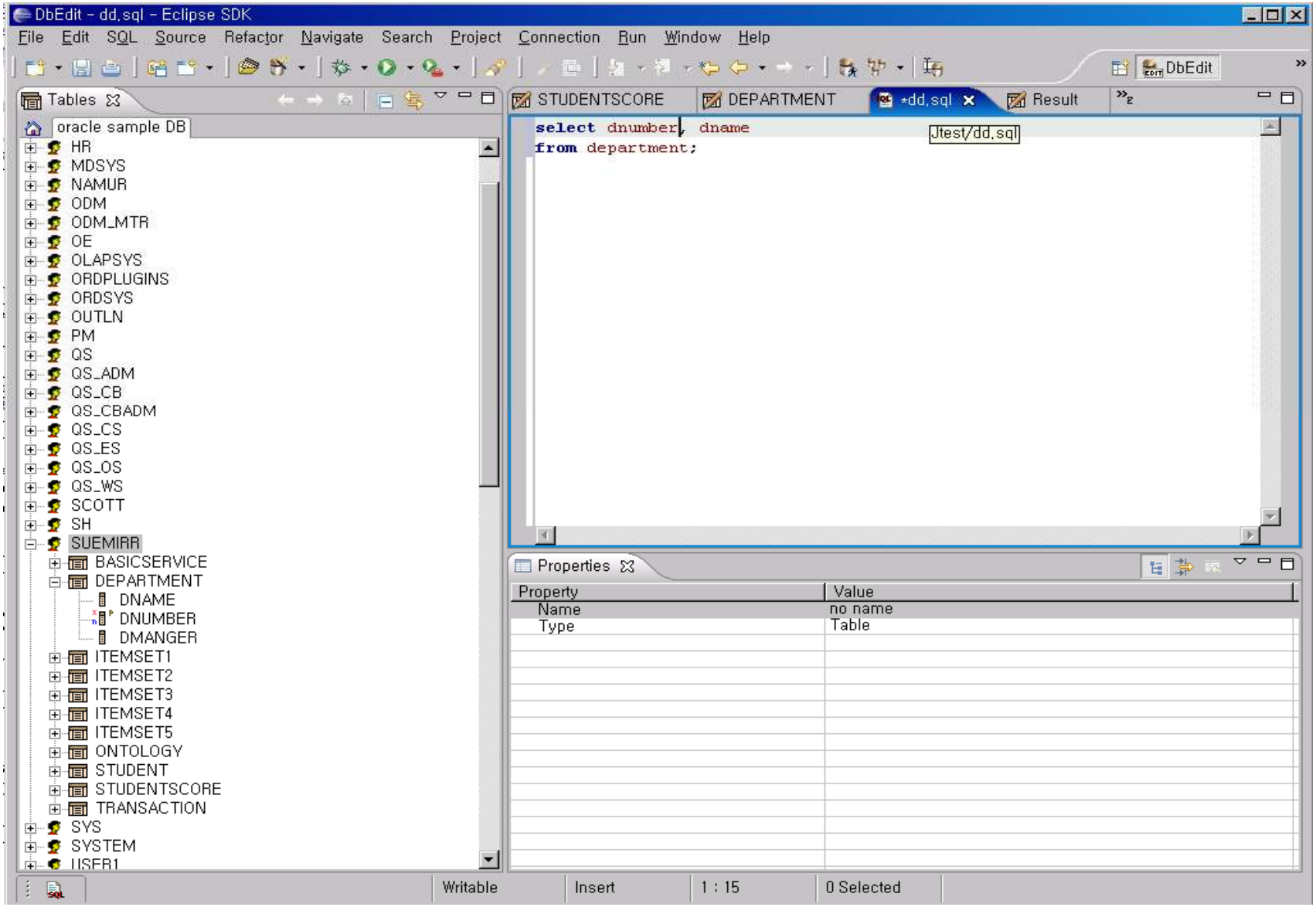
❑ 삽입


DbEdit - DEPARTMENT - Eclipse SDK
File Edit SQL Source Refactor Navigate Search Project Connection Run Window Help
DbEdit
Tables
oracle sample DB
HR
MDSYS
NAMUR
ODM
ODM_MTR
OE
OLAPSYS
ORDPLUGINS
ORDSYS
OUTLN
PM
QS
QS_ADM
QS_CB
QS_CBADM
QS_CS
QS_ES
QS_OS
QS_WS
SCOTT
SH
SUEMIRR
BASICSERVICE
DEPARTMENT
DNAME
DNUMBER
DMANGER
ITEMSET1
ITEMSET2
ITEMSET3
ITEMSET4
ITEMSET5
ONTOLOGY
STUDENT
STUDENTSCORE
TRANSACTION
SYS
SYSTEM
USER1
HmmLibrary.java
STUDENT
STUDENTSCORE
DEPARTMENT
oracle sample DB - SUEMIRR.DEPARTMENT
Row DNAME DNUMBER DMANGER
1 전기 100 김종서
2 전자 200 김진영
3 컴교 300 이종연
new 정보 400 나훈아
Confirm Insertion
Execute the insertion of a new row?
OK
Cancel
Details >>
Table Log
Properties
Property Value
Foreign Key false
Length 10
Name DMANGER
Nullable true
Primary Key false
Type char
DMANGER - char (10)

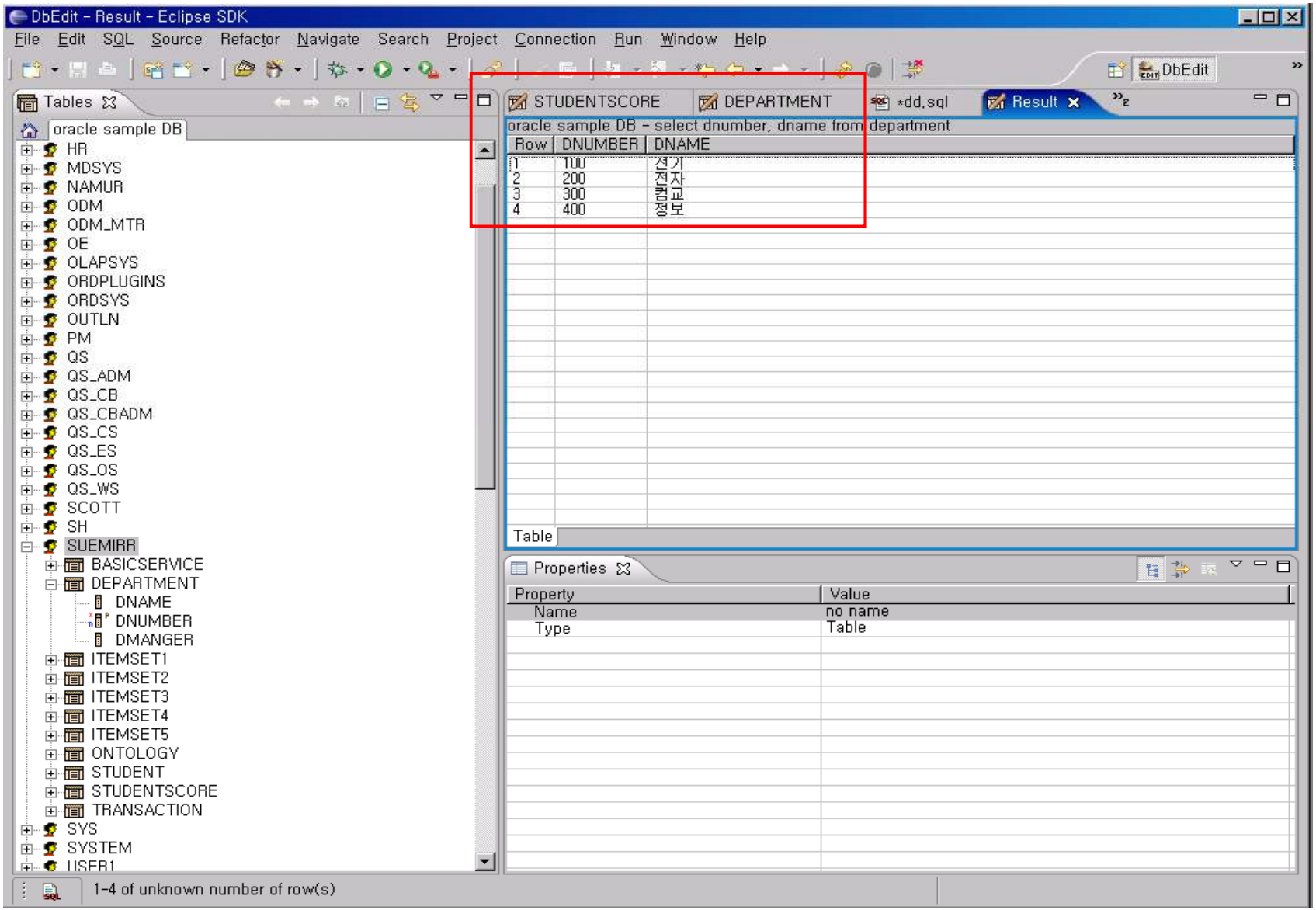
❑ SQL문 작성


DbEdit - DEPARTMENT - Eclipse SDK
File Edit SQL Source Refactor Navigate Search Project Connection Run Window Help
New Alt+Shift+N
Open File...
Close Ctrl+F4
Close All Ctrl+Shift+F4
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To
Print... Ctrl+P
Switch Workspace...
Import...
Export...
Properties Alt+Enter
1 DEPARTMENT [oracle sample DB - SUEM...]
2 STUDENTSCORE [oracle sample DB - SU...]
3 STUDENT [oracle sample DB - SUEMIRR...]
4 HmmLibrary.java [HMM for protein st...]
Exit
Project...
Schema
Table
SQL File
Other... Ctrl+N
DbEdit
STUDENT
STUDENTSCORE
DEPARTMENT
RR.DEPARTMENT
DMANGER
이종연
나훈아
김진영
김종서
Table Log
Properties
Property
Value
Name
DEPARTMENT
Type
Table
ITEMSET1
ITEMSET2
ITEMSET3
ITEMSET4
ITEMSET5
ONTOLOGY
STUDENT
STUDENTSCORE
TRANSACTION
SYS
SYSTEM
USER1
DEPARTMENT

❑ SQL문 작성


DbEdit - dd.sql - Eclipse SDK
File Edit SQL Source Refactor Navigate Search Project Connection Run Window Help
DbEdit
Tables
oracle sample DB
HR
MDSYS
NAMUR
ODM
ODM_MTR
OE
OLAPSYS
ORDPLUGINS
ORDSYS
OUTLN
PM
QS
QS_ADM
QS_CB
QS_CBADM
QS_CS
QS_ES
QS_OS
QS_WS
SCOTT
SH
SUEMIRR
BASICSERVICE
DEPARTMENT
DNAME
DNUMBER
DMANGER
ITEMSET1
ITEMSET2
ITEMSET3
ITEMSET4
ITEMSET5
ONTOLOGY
STUDENT
STUDENTSCORE
TRANSACTION
SYS
SYSTEM
STUDENTSCORE
DEPARTMENT
*dd.sql
Result
select dnumber, dname
from department;
Jtest/dd.sql
Properties
Property
Value
Name
no name
Type
Table
Writable
Insert
1 : 15
0 Selected

❑ 질의 결과


DbEdit - Result - Eclipse SDK
File Edit SQL Source Refactor Navigate Search Project Connection Run Window Help
DbEdit
Tables
oracle sample DB
HR
MDSYS
NAMUR
ODM
ODM_MTR
OE
OLAPSYS
ORDPLUGINS
ORDSYS
OUTLN
PM
QS
QS_ADM
QS_CB
QS_CBADM
QS_CS
QS_ES
QS_OS
QS_WS
SCOTT
SH
SUEMIRR
BASICSERVICE
DEPARTMENT
DNAME
DNUMBER
DMANGER
ITEMSET1
ITEMSET2
ITEMSET3
ITEMSET4
ITEMSET5
ONTOLOGY
STUDENT
STUDENTSCORE
TRANSACTION
SYS
SYSTEM
USER1
STUDENTSCORE
DEPARTMENT
*dd.sql
Result
oracle sample DB - select dnumber, dname from department
Row
DNUMBER
DNAME
1 100 전기
2 200 전자
3 300 컴교
4 400 정보
Table
Properties
Property
Value
Name
no name
Type
Table
1-4 of unknown number of row(s)