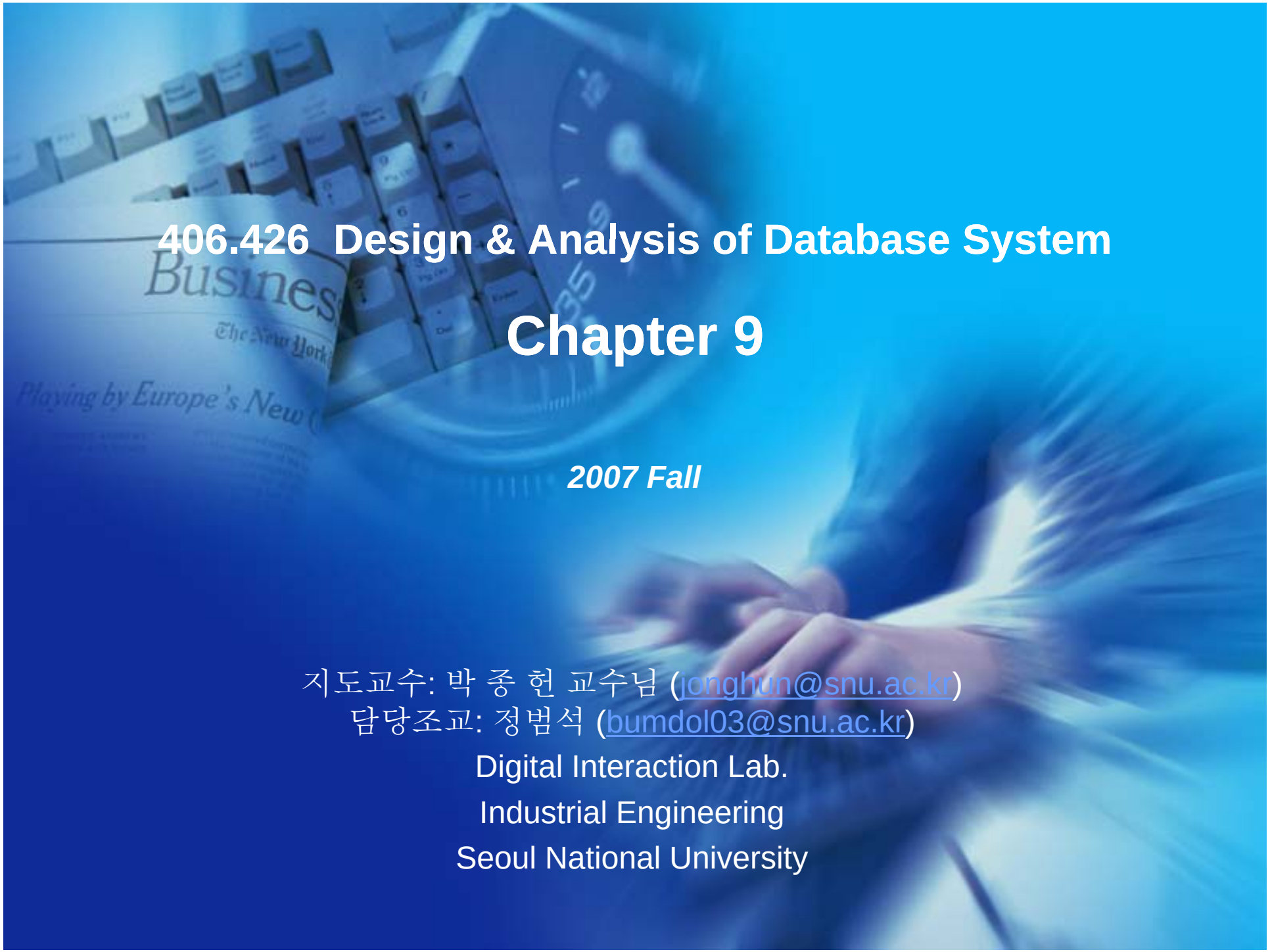


406.426 Design & Analysis of Database System

Chapter 9

2007 Fall

지도교수: 박 종 헌 교수님 (jonghun@snu.ac.kr)
담당조교: 정범석 (bumdol03@snu.ac.kr)
Digital Interaction Lab.
Industrial Engineering
Seoul National University

Database Programming

- Objective: to access a database from an application program (as opposed to interactive interfaces)

- Why? An interactive interface is convenient but not sufficient; a majority of database operations are made thru application programs (nowadays thru web applications)

Database Programming Approaches

- Embedded commands: database commands are embedded in a general-purpose programming language
- Library of database functions: available to the host language for database calls; known as an *API*
- A brand new, full-fledged language (minimizes impedance mismatch)

Impedance Mismatch

- Incompatibilities between a host programming language and the database model, e.g.,
  - type mismatch and incompatibilities; requires a new binding for each language
  - set vs. record-at-a-time processing
    - need special iterators to loop over query results and manipulate individual values

Steps in Database Programming

1. Client program opens a connection to the database server
2. Client program submits queries to and/or updates the database
3. When database access is no longer needed, client program terminates the connection

Embedded SQL

- Most SQL statements can be embedded in a general-purpose *host* programming language such as COBOL, C, Java

- An embedded SQL statement is distinguished from the host language statements by `EXEC SQL` and a matching `END-EXEC` (or semicolon)
  - *shared variables* (used in both languages) usually prefixed with a colon (:) in SQL

Embedded SQL in C Programming Examples

```
loop = 1;
while (loop) {
   prompt ("Enter SSN: ", ssn);
   EXEC SQL
       select FNAME, LNAME, ADDRESS, SALARY
       into :fname, :lname, :address, :salary
       from EMPLOYEE where SSN == :ssn;
       if (SQLCODE == 0) printf(fname, …);
       else printf("SSN does not exist: ", ssn);
       prompt("More SSN? (1=yes, 0=no): ", loop);
   END-EXEC
}
```

Dynamic SQL

- Objective: executing new (not previously compiled) SQL statements at run-time
  - a program accepts SQL statements from the keyboard at run-time
  - a point-and-click operation translates to certain SQL query

- Dynamic update is relatively simple; dynamic query can be complex
  - because the type and number of retrieved attributes are unknown at compile time

Dynamic SQL: An Example

```
EXEC SQL BEGIN DECLARE SECTION;
varchar sqlupdatestring[256];
EXEC SQL END DECLARE SECTION;
…
prompt (“Enter update command:“, sqlupdatestring);
EXEC SQL PREPARE sqlcommand FROM :sqlupdatestring;
EXEC SQL EXECUTE sqlcommand;
```

Embedded SQL in Java

- SQLJ: a standard for embedding SQL in Java
- An SQLJ translator converts SQL statements into Java (to be executed thru the *JDBC* interface)
- Certain classes, e.g., `java.sql` have to be imported

Java Database Connectivity

- JDBC: SQL connection function calls for Java programming
- A Java program with JDBC functions can access any relational DBMS that has a JDBC driver
- JDBC allows a program to connect to several databases (known as *data sources*)

Steps in JDBC Database Access

1. Import JDBC library (`java.sql.*`)
2. Load JDBC driver: `Class.forname("oracle.jdbc.driver.OracleDriver")`
3. Define appropriate variables
4. Create a connect object (via `getConnection`)
5. Create a statement object from the `Statement` class:
    1. `PreparedStatment`
    2. `CallableStatement`
6. Identify statement parameters (to be designated by question marks)
7. Bound parameters to program variables
8. Execute SQL statement (referenced by an object) via JDBC's `executeQuery`
9. Process query results (returned in an object of type `ResultSet`)
    - `ResultSet` is a 2-dimentional table

SQLJ

```
1)   import java.sql.* ;
2)   import java.io.* ;
3)   import sqlj.runtime.* ;
4)   import sqlj.runtime.ref.* ;
5)   import oracle.sqlj.runtime.* ;
     …
6)   DefaultContext cntxt =
7)       oracle.getConnection("<url name>", "<user name>", "<password>", true) ;
8)   DefaultContext.setDefaultContext(cntxt) ;
     …

//Program Segment J1:
1)   ssn = readEntry("Enter a Social Security Number: ") ;
2)   try {
3)      #sql{select FNAME, MINIT, LNAME, ADDRESS, SALARY
4)         into :fname, :minit, :lname, :address, :salary
5)         from EMPLOYEE where SSN = :ssn} ;
6)   }  catch (SQLException se) {
7)      System.out.println("Social Security Number does not exist: " + ssn) ;
8)      Return ;
9)      }
10)  System.out.println(fname + " " + minit + " " + lname + " " + address + " " +
     salary)
```

Database Programming with Functional Calls

- Embedded SQL provides static database programming
- API: dynamic database programming with a library of functions
  - advantage: no preprocessor needed (thus more flexible)
  - drawback: SQL syntax checks to be done at run-time

a C program segment with SQL/CLI

```
//Program CLI1:
0)   #include sqlcli.h ;
1)   void printSal() {
2)   SQLHSTMT stmt1 ;
3)   SQLHDBC con1 ;
4)   SQLHENV env1 ;
5)   SQLRETURN ret1, ret2, ret3, ret4 ;
6)   ret1 = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &env1) ;
7)   if (!ret1) ret2 = SQLAllocHandle(SQL_HANDLE_DBC, env1, &con1) else exit ;
8)   if (!ret2) ret3 = SQLConnect(con1, "dbs", SQL_NTS, "js", SQL_NTS, "xyz", SQL_NTS)
else exit ;
9)   if (!ret3) ret4 = SQLAllocHandle(SQL_HANDLE_STMT, con1, &stmt1) else exit ;
10)  SQLPrepare(stmt1, "select LNAME, SALARY from EMPLOYEE where SSN = ?", SQL_NTS) ;
11)  prompt("Enter a Social Security Number: ", ssn) ;
12)  SQLBindParameter(stmt1, 1, SQL_CHAR, &ssn, 9, &fetchlen1) ;
13)  ret1 = SQLExecute(stmt1) ;
14)  if (!ret1) {
15)      SQLBindCol(stmt1, 1, SQL_CHAR, &lname, 15, &fetchlen1) ;
16)      SQLBindCol(stmt1, 2, SQL_FLOAT, &salary, 4, &fetchlen2) ;
17)      ret2 = SQLFetch(stmt1) ;
18)      if (!ret2) printf(ssn, lname, salary)
19)          else printf("Social Security Number does not exist: ", ssn) ;
20)      }
21)  }
```

a JAVA program segment with JDBC

```
//Program JDBC1:
0)    import java.io.* ;
1)    import java.sql.*
      …
2)    class getEmpInfo {
3)        public static void main (String args []) throws SQLException, IOException {
4)        try { Class.forName("oracle.jdbc.driver.OracleDriver")
5)        } catch (ClassNotFoundException x) {
6)            System.out.println ("Driver could not be loaded") ;
7)            }
8)        String dbacct, passwrd, ssn, lname ;
9)        Double salary ;
10)       dbacct = readentry("Enter database account:") ;
11)       passwrd = readentry("Enter pasword:") ;
12)       Connection conn = DriverManager.getConnection
13)           ("jdbc:oracle:oci8:" + dbacct + "/" + passwrd) ;
14)       String stmt1 = "select LNAME, SALARY from EMPLOYEE where SSN = ?" ;
15)       PreparedStatement p = conn.prepareStatement(stmt1) ;
16)       ssn = readentry("Enter a Social Security Number: ") ;
17)       p.clearParameters() ;
18)       p.setString(1, ssn) ;
19)       ResultSet r = p.executeQuery() ;
20)       while (r.next()) {
21)           lname = r.getString(1) ;
22)           salary = r.getDouble(2) ;
23)           system.out.println(lname + salary) ;
24)       }  }
25)   }
```

a JAVA program segment with JDBC (cont.)

```
//Program Segment JDBC2:
0)    import java.io.* ;
1)    import java.sql.*
      …
2)    class printDepartmentEmps {
3)        public static void main (String args []) throws SQLException, IOException {
4)        try { Class.forName("oracle.jdbc.driver.OracleDriver")
5)        } catch (ClassNotFoundException x) {
6)            System.out.println ("Driver could not be loaded") ;
7)            }
8)        String dbacct, passwrd, lname ;
9)        Double salary ;
10)       Integer dno ;
11)       dbacct = readentry("Enter database account:") ;
12)       passwrd = readentry("Enter pasword:") ;
13)       Connection conn = DriverManager.getConnection
14)           ("jdbc:oracle:oci8:" + dbacct + "/" + passwrd) ;
15)       dno = readentry("Enter a Department Number: ") ;
16)       String q = "select LNAME, SALARY from EMPLOYEE where DNO = " +
          dno.tostring() ;
17)       Statement s = conn.createStatement() ;
18)       ResultSet r = s.executeQuery(q) ;
19)       while (r.next()) {
20)           lname = r.getString(1) ;
21)           salary = r.getDouble(2) ;
22)           system.out.println(lname + salary) ;
23)       }  }
24)   }
```

Database Stored Procedures

- Persistent procedures/functions (modules) are stored locally and executed by the database server (as opposed to execution by clients)

- Advantages:
  - if the procedure is needed by many applications, it can be invoked by any of them (thus reduce duplications)
  - execution by the server reduces communication costs
  - enhance the modeling power of views

Stored Procedure Constructs

- A stored procedure

```
CREATE PROCEDURE procedure-name (params)
local-declarations
procedure-body;
```

- A stored function

```
CREATE FUNCTION fun-name (params) RETRUNS return-type
local-declarations
function-body;
```

- Calling a procedure or function

```
CALL procedure-name/fun-name (arguments);
```

SQL Persistent Stored Modules

- SQL/PSM: part of the SQL standard for writing persistent stored modules

- SQL + stored procedures/functions + additional programming constructs
  - e.g., branching and looping statements
  - enhance the power of SQL

```
//Function PSM1:
0)    CREATE FUNCTION DeptSize(IN deptno INTEGER)
1)    RETURNS VARCHAR [7]
2)    DECLARE NoOfEmps INTEGER ;
3)    SELECT COUNT(*) INTO NoOfEmps
4)    FROM EMPLOYEE WHERE DNO = deptno ;
5)    IF NoOfEmps > 100 THEN RETURN "HUGE"
6)        ELSEIF NoOfEmps > 25 THEN RETURN "LARGE"
7)        ELSEIF NoOfEmps > 10 THEN RETURN "MEDIUM"
8)        ELSE RETURN "SMALL"
9)    END IF ;
```

MySQL GUI Tools

- download(http://dev.mysql.com/downloads/gui-tools/5.0.html)

Windows downloads

The install package uses the Windows Installer, which is built in to Windows XP and more recent Microsoft Windows versions. An update for Windows 2000 can be downloaded here.

Windows (x86)	5.0-r12	17.4M	Pick a mirror
		MD5: 7d6d546069554900e6d9a324b6840336	
Without installer (unzip in C:\)	5.0-r12	16.6M	Pick a mirror
		MD5: 9f396065bc095ff73dbd6e478554ee62	

Mac OSX downloads

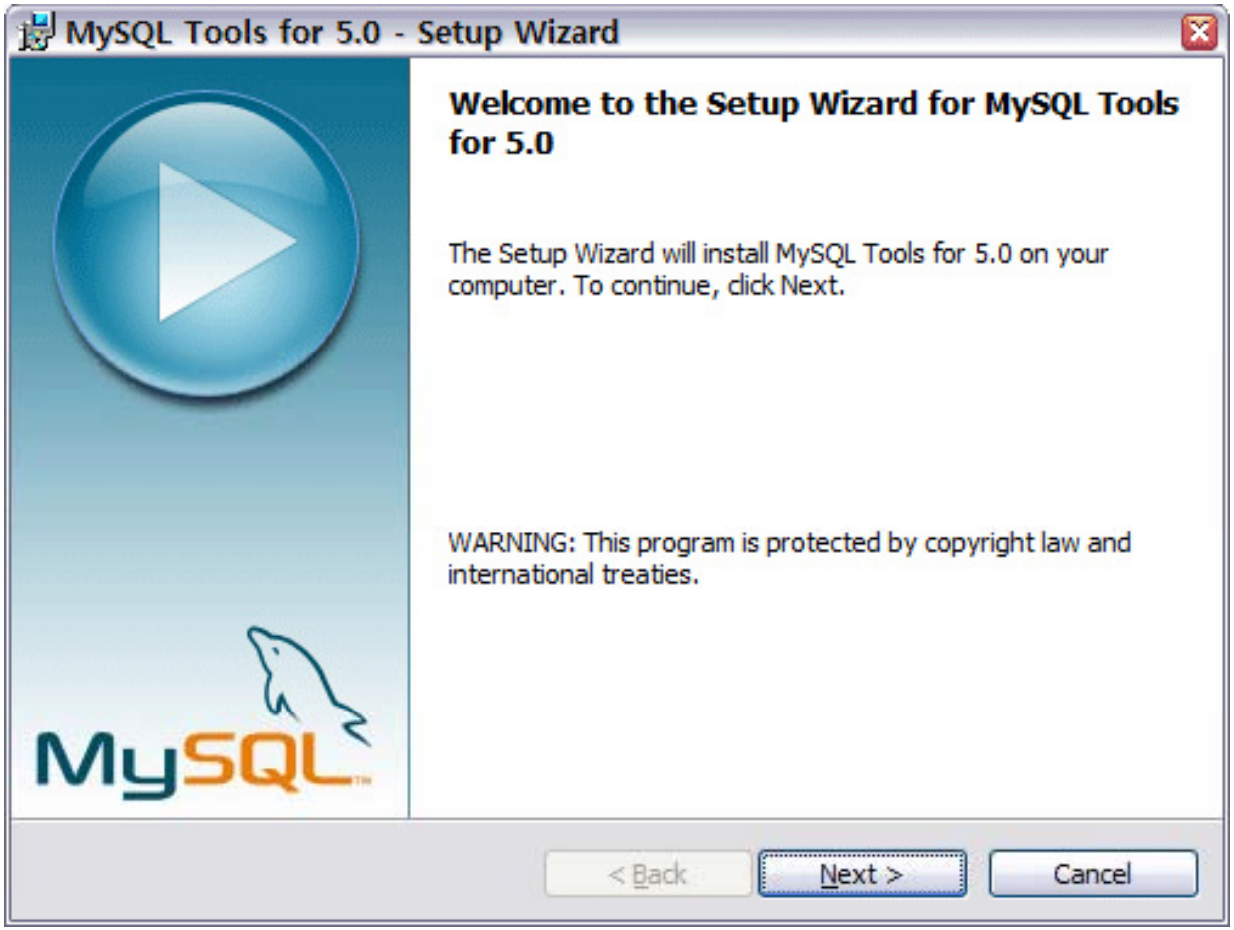
MAC OSX 10.4 (Universal binaries)	5.0-r12	12.4M	Pick a mirror
		MD5: 87989a094c5edb6efb28cdad50dd1a44	

Linux downloads

Individual RPM packages are supplied for the GUI Tools components and combined into a single TAR archive.

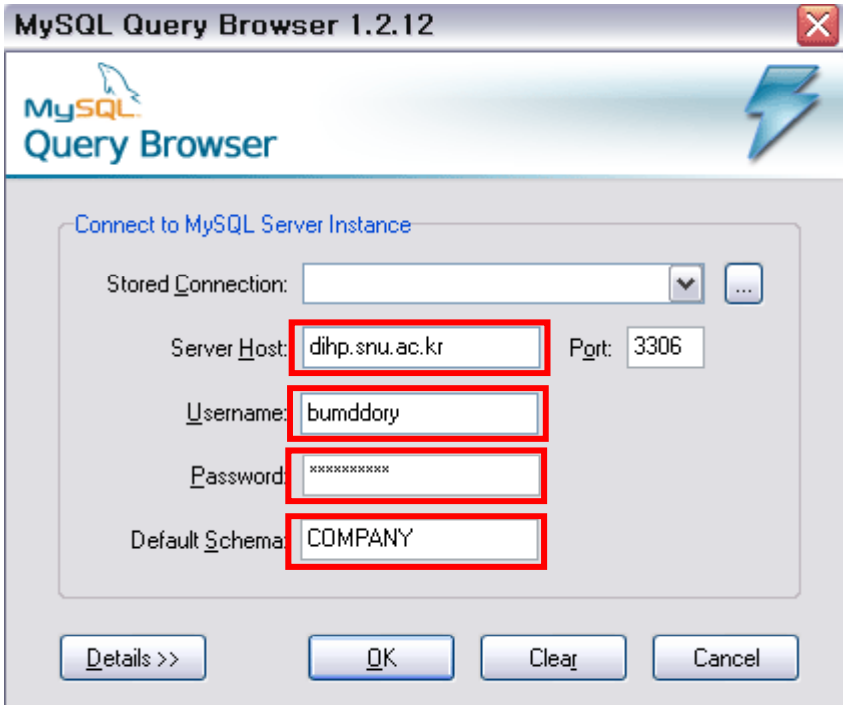
RedHat Enterprise Linux 3 (x86) RPM (bundled dependencies)	5.0r12	19.4M	Pick a Mirror
		MD5: 45efb53691698dae0a6badbb27f9c742	
RedHat Enterprise Linux 4 (x86) RPM (bundled dependencies)	5.0r12	17.8M	Pick a Mirror
		MD5: 96a346c760b4886cd30808cc0bab4c17	
Fedora Core 5 (x86) RPM	5.0r12	15.8M	Pick a Mirror
		MD5: ba81ca703a8f2b21945bcdfab35b6049	
SuSE Linux 10.x (x86) RPM	5.0r12	32.9M	Pick a Mirror
		MD5: cf6901d3bd906abb956a60c011b8ea04	

MySQL GUI Tools cont.

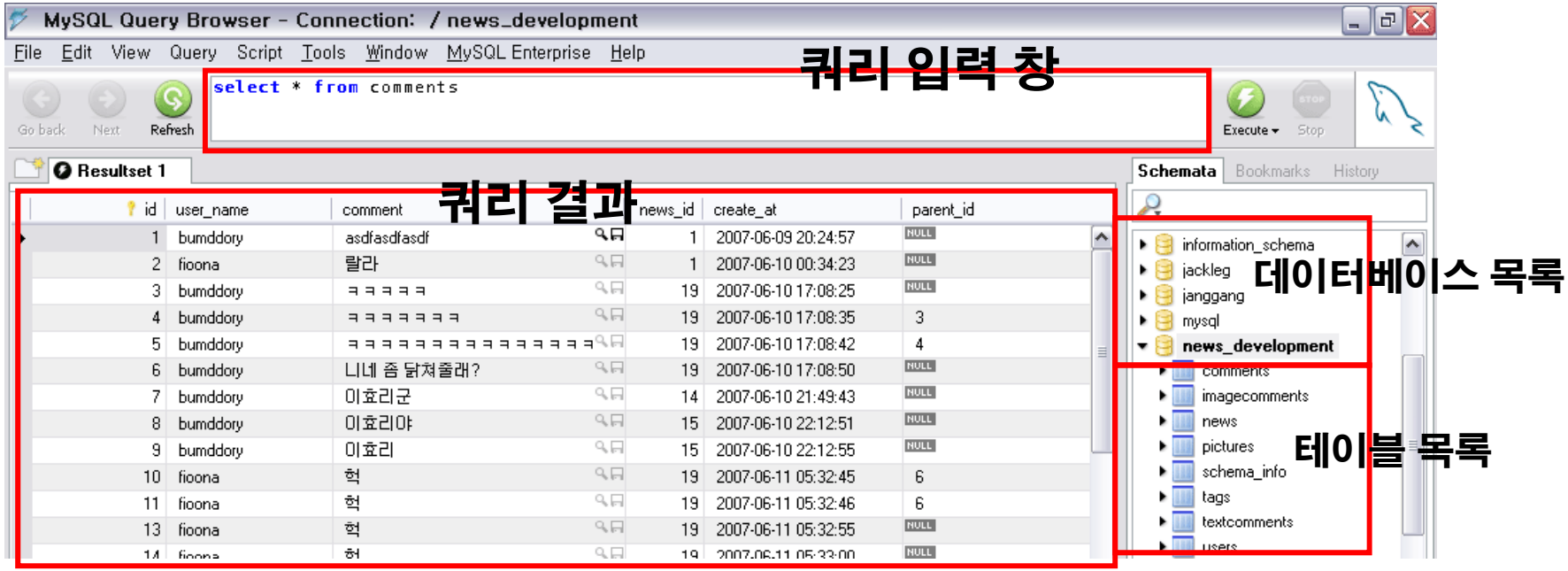


MySQL GUI Tools cont.

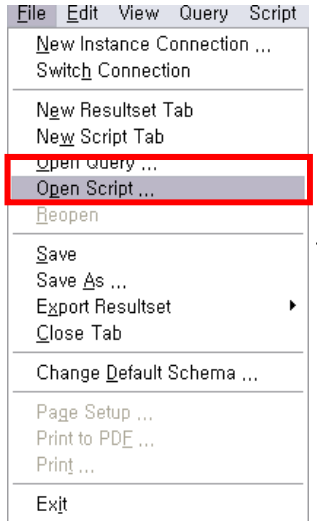
- Query Browser를 통해 DB 접속



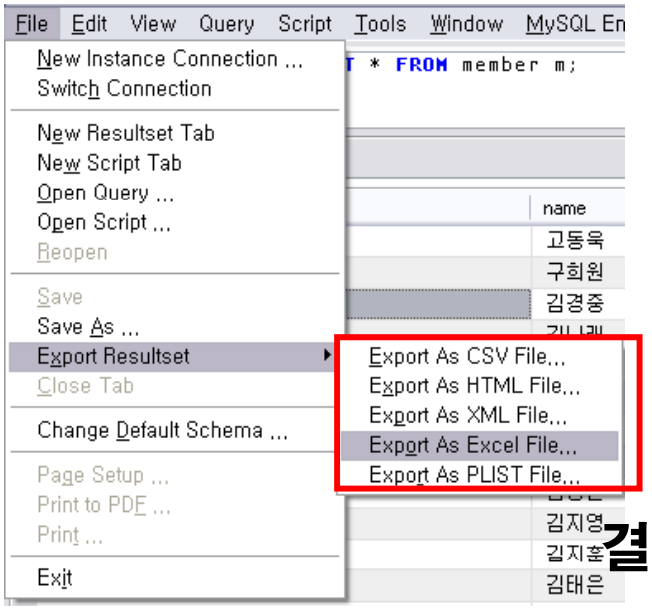
MySQL GUI Tools cont.



MySQL GUI Tools cont.



sql문 가져오기



결과를 파일로 저장


Thank You !