UNIT 23

프로그래밍

이 장에서는 DB2가 제공하는 ESQL, CLI, API, JDBC에 대한 전반적인 내용에 대해 소개합니다.

DB2 9.7 개발자 가이드

Developer Edition

- ESQL
- CLI
- API
- JDBC

TOPIC 2301 ESQL

Point

SQL과 프로그래밍 언어가 혼합된 방식인 ESQL로 구현된 응용프로그램을 빌드하려면 프리컴파일이 필요합니다. 프리컴파일러와 헤더 파일, 라이브러리를 제공하려면 개발 머신에 DB2 Application Development Client 모듈 이상을 설치해야 합니다.

Tip

ESQL은 Embedded SQL 의 약자입니다.

Tip

prep 명령어에서 BINDFILE 옵션이 없으면, bnd 파일을 생성하지 않고 패키지를 생성하므로 bind 명령어를 실행할 필요가 없습니다.

Tip

ESQL/C의 소스 파일의 확장자는 sqc입니다. C 컴파일러를 위한 PATH, LIBPATH 등의 환경 변수가 적절히 구성되었는지 확인하십시오.

Tip

ESQL 방식의 프로그램의 최종 실행 모듈은 바이너리 코드로 된 C 실행 모듈과 데이터베이스 시스템 카탈로그에 저장된 패키지입니다. C 실행 모듈은 SQL문에 대한 액세스 플랜을 가진 패키지를 호출합니다.

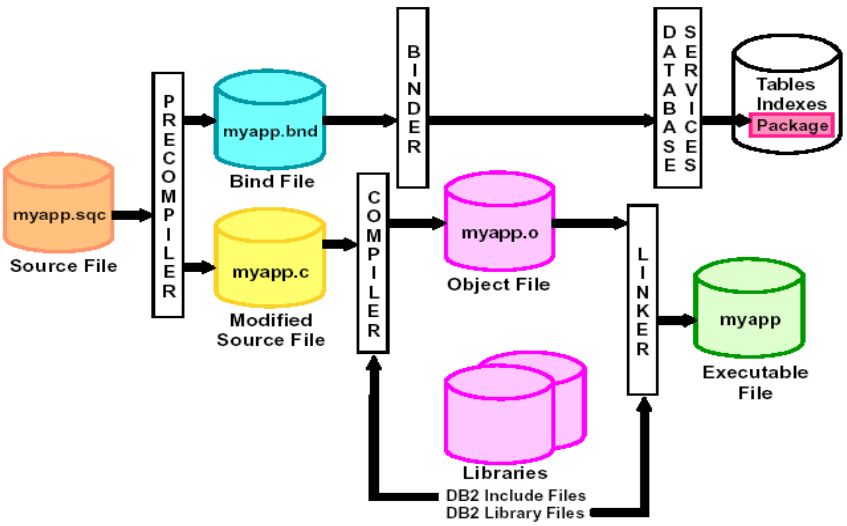
Tip

프로그램의 로직을 표현하는 프로그래밍 언어를 호스트 언어라고 합니다.

Tip

SQL문은 시작과 종료를 나타내는 특수한 구분자를 이용하여 표현합니다. C 언어를 사용하는 경우에는 EXEC SQL 로 시작하고, ; (세미콜론)으로 종료합니다.

1 ESQL은 데이터의 액세스는 SQL문으로 구현하고, 프로그램의 로직은 프로그래밍 언어로 구현하는 혼합 방식입니다. 프리컴파일과 SQL문에 대한 바인드, 프로그래밍 언어에 대한 컴파일과 링크 과정을 통해서 패키지와 바이너리 실행 파일을 생성합니다.




```
$ db2 connect to sample
$ db2 prep myapp.sqc BINDFILE
$ db2 bind myapp.bnd blocking all grant public
$ xlc -c myapp.c –I~db2inst1/sqllib/include
$ xlc -o myapp myapp.o -ldb2 –L~db2inst1/sqllib/lib
$ myapp
```

Figure 2301A ESQL/C 프로그램 빌드 과정

2 DB2 개발자로 로그인하여 ESQL/C 소스 파일을 작성합니다. 확장자를 sqc 로 지정합니다.

```
#include <stdio.h>

EXEC SQL INCLUDE SQLCA;

int main()
{
   EXEC SQL BEGIN DECLARE SECTION;
      char fname[13];
      char lname[16];
      char newfname[13];
   EXEC SQL END DECLARE SECTION;

   EXEC SQL CONNECT TO sample;
```

TOPIC 2301 ESQL

Point

SQL과 프로그래밍 언어가 혼합된 방식인 ESQL로 구현된 응용프로그램을 빌드하려면 프리컴파일이 필요합니다. 프리컴파일러와 헤더 파일, 라이브러리를 제공하려면 개발 머신에 DB2 Application Development Client 모듈 이상을 설치해야 합니다.

```
    printf("Enter the lastname : ");
    gets(lname);

    EXEC SQL SELECT FIRSTNME INTO :fname
            FROM employee
            WHERE LASTNAME = :lname;

    if (SQLCODE == 0)
      printf("First name = %s\n", fname);
    else if (SQLCODE == 100)
      printf("Data is not found\n");
    else {
      printf("Select Error : SQLCODE = %ld\n", SQLCODE);
      EXEC SQL ROLLBACK;
      EXEC SQL CONNECT RESET;
      return (-1);
    }

    EXEC SQL COMMIT;
    EXEC SQL CONNECT RESET;

    return(0) ;
}
```

Tip

프로그램을 실행할 때는 패키지에 대한 EXECUTE 권한이 있어야 합니다.

Tip

~<인스턴스명>은 DB2 인스턴스 개발자의 홈디렉토리를 의미합니다.

3 prep 명령어로 프리컴파일하고, bind 명령어로 패키지를 생성합니다.

```
$ db2 connect to <데이터베이스명>
$ db2 prep <소스파일명>.sqc bindfile
$ db2 bind <소스파일명>.bnd blocking all grant public
```

4 C 컴파일러를 이용하여 컴파일과 링크를 실행하십시오.

```
$ xlc -c <소스파일명>.c –I~<인스턴스명>/sqllib/include
$ xlc -o <실행파일명> <소스파일명>.o -ldb2 –L~<인스턴스명>/sqllib/lib
```

5 바이너리 형식의 실행 모듈을 실행하십시오.

```
$ <실행파일명>
Enter the lastname : LEE
First name = Newman
```

TOPIC 2302 CLI

Point

CLI는 C 라이브러리 형식으로 된 API를 이용하여 SQL문을 표현하는 프로그램 방식으로 직접적인 SQL문이 표현되어 있지 않으므로 프리컴파일은 필요하지 않으며, 이종 DBMS 간의 호환성이 높습니다. SQL문에 대한 패키지는 별도로 생성되지 않습니다.

Tip

CLI는 Call Level Interface의 약자입니다.

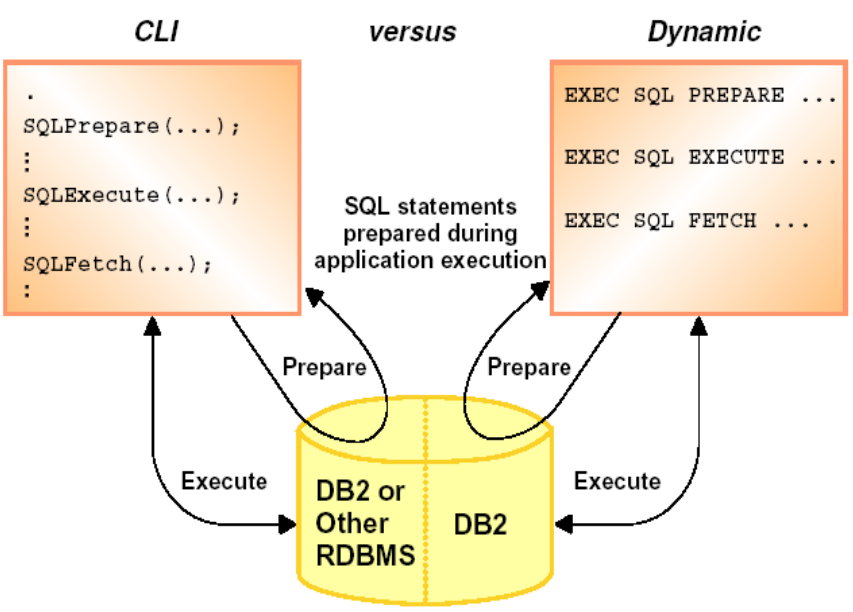
Tip

CLI의 소스 파일의 확장자는 c입니다. C 컴파일러를 위한 PATH, LIBPATH 등의 환경 변수가 적절히 구성되었는지 확인하십시오.

Tip

최종 실행 모듈은 바이너리 코드로 된 C 실행 모듈입니다. C 실행 모듈은 CLI를 위해 데이터베이스 시스템에 내장된 패키지를 호출합니다.

1 CLI는 데이터를 액세스하는 SQL문은 C 라이브러리 형식으로 된 API 로 구현하고, 프로그램의 로직은 C 언어로 구현하는 방식입니다. C 언어에 대한 컴파일과 링크 과정을 통해서 바이너리 실행 파일을 생성합니다. ESQL 에서 지원하는 동적 SQL문에 대응하는 API 가 모두 지원됩니다.




```
$ xlc -c mycli.c –I~db2inst1/sqllib/include
$ xlc -o mycli mycli.o -ldb2 –L~db2inst1/sqllib/lib
$ mycli
```

Figure 2302A CLI 와 ESQL

2 DB2 개발자로 로그인하여 CLI 소스 파일을 작성합니다. 확장자를 c 로 지정합니다.

```
#include <stdio.h>
#include <sqlcli1.h>

main() {

  SQLHANDLE   henv;
  SQLHANDLE   hdbc;
  SQLHANDLE   hstmt;

  SQLCHAR * stmt = ( SQLCHAR * ) "SELECT deptnumb, location
   FROM org";
  SQLRETURN   sqlrc = SQL_SUCCESS;
```

TOPIC 23 02 CLI

Point

CLI는 C 라이브러리 형식으로 된 API를 이용하여 SQL문을 표현하는 프로그램 방식으로 직접적인 SQL문이 표현되어 있지 않으므로 프리컴파일은 필요하지 않으며, 이종 DBMS 간의 호환성이 높습니다. SQL문에 대한 패키지는 별도로 생성되지 않습니다.

Tip

ESQL 방식의 프로그램에서 dynamic SQL로 구현했던 모든 로직은 대응되는 한 개 이상의 CLI로 변환될 수 있습니다. dynamic SQL이 지원하지 않는 기능을 구현하려면 CLI를 이용합니다.

```
struct
{  SQLINTEGER ind ;
   SQLSMALLINT val ;
} deptnumb ;

struct
{  SQLINTEGER ind ;
   SQLCHAR val[15] ;
} location ;

SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);
SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);
SQLConnect(hdbc,  (SQLCHAR *)"SAMPLE", SQL_NTS, NULL,
 SQL_NTS, NULL, SQL_NTS);
SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt);

SQLExecDirect( hstmt, stmt, SQL_NTS ) ;

printf("DEPTNUMB LOCATION     \n" ) ;
printf("-------- ------------\n" ) ;
sqlrc = SQLFetch( hstmt );

if (sqlrc == SQL_NO_DATA_FOUND)
     printf("\n    Data not found.\n");

while (sqlrc != SQL_NO_DATA_FOUND)
{
   sqlrc = SQLGetData( hstmt, 1, SQL_C_SHORT, &deptnumb.val,
                       0, &deptnumb.ind ) ;
   sqlrc = SQLGetData( hstmt, 2, SQL_C_CHAR, location.val,
                       15, &location.ind ) ;
   printf( "%-8d %-14.14s \n", deptnumb.val, location.val ) ;
   sqlrc = SQLFetch( hstmt );
}
```

Point

CLI는 C 라이브러리 형식으로 된 API를 이용하여 SQL문을 표현하는 프로그램 방식으로 직접적인 SQL문이 표현되어 있지 않으므로 프리컴파일은 필요하지 않으며, 이종 DBMS 간의 호환성이 높습니다. SQL문에 대한 팩키지는 별도로 생성되지 않습니다.

```
    SQLFreeStmt(hstmt, SQL_CLOSE);
    SQLEndTran(SQL_HANDLE_DBC, hdbc, SQL_COMMIT);
    SQLFreeHandle(SQL_HANDLE_STMT, hstmt);
    SQLDisconnect(hdbc);
    SQLFreeHandle(SQL_HANDLE_DBC, hdbc);
    SQLFreeHandle(SQL_HANDLE_ENV, henv);

    return(0);
}
```

Tip

~<인스턴스명>은 DB2 인스턴스 개발자의 홈디렉토리를 의미합니다.

3 C 컴파일러를 이용하여 컴파일과 링크를 실행하십시오.

```
$ xlc -c <소스파일명>.c –I~<인스턴스명>/sqllib/include
$ xlc -o <실행파일명> <소스파일명>.o -ldb2 –L~<인스턴스명>/sqllib/lib
```

4 바이너리 형식의 실행 모듈을 실행하십시오.

```
$ <실행파일명>
DEPTNUMB LOCATION
--------------------------------
      10 New York
      15 Boston
      20 Washington
      38 Atlanta
      42 Chicago
      51 Dallas
      66 San Francisco
      84 Denver
      10 New York
      15 Boston
      20 Washington
      38 Atlanta
      42 Chicago
      51 Dallas
      66 San Francisco
      84 Denver
```

TOPIC 2303 API

Point

API는 DB2 명령어에 대응하는 C 라이브러리 함수입니다. ESQL/C 소스에서 사용할 수 있으며, DB2가 제공하는 다양한 명령어에 대한 관리 API를 프로그램 로직과 함께 구현하는 방식입니다.

Tip

API는 Application Programming Interface의 약자입니다.

Tip

ESQL/C와 동일한 방식으로 빌드합니다.

1 API는 DB2 명령어를 C 라이브러리 형식으로 된 API 로 구현하는 방식입니다. 데이터 액세스에 SQL문을 사용하고, 프로그램 로직을 C 언어로 구현하는 것은 ESQL/C 와 동일합니다.

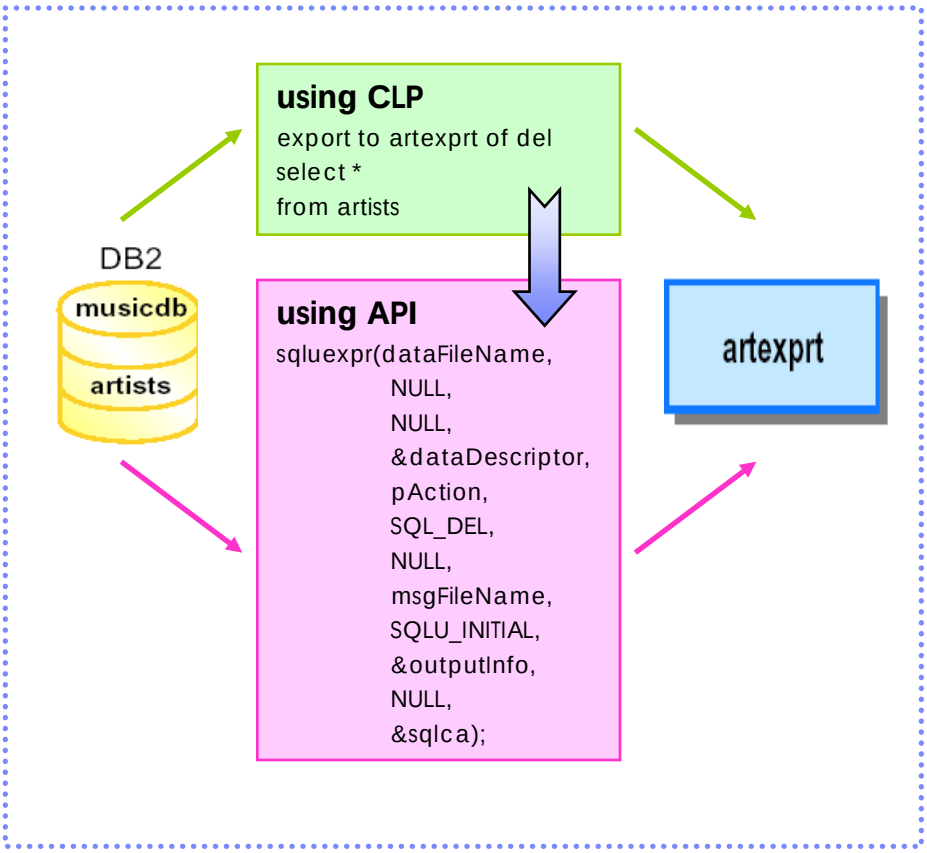


Figure 2303A • • • export 명령어와 sqluexpr 관리 API

2 DB2 개발자로 로그인하여 API 를 포함한 ESQL/C 소스 파일을 작성합니다. ESQL/C 소스이므로 확장자는 sqc 로 지정합니다.

```
$ login <DB2 개발자>
$ vi <소스파일명>.sqc
#include <stdio.h>
#include <stdlib.h>
#include <sqlutil.h>

int main()
{
 struct sqlca sqlca;

 char dataFileName[256];
```

TOPIC 23 03 API

Point

API는 DB2 명령어에 대응하는 C 라이브러리 함수입니다. ESQL/C 소스에서 사용할 수 있으며, DB2가 제공하는 다양한 명령어에 대한 관리 API를 프로그램 로직과 함께 구현하는 방식입니다.

Tip

소스에서 사용된 sqluexpr 은 DB2 명령어인 export 에 대응되는 API 입니다.

```
 struct sqldcol dataDescriptor;
 char actionString[256];
 struct sqlchar *pAction;
 char msgFileName[128];
 struct sqluexpt_out outputInfo;

 EXEC SQL connect to sample;

 strcpy(dataFileName, "ORG.DEL");
 dataDescriptor.dcolmeth = SQL_METH_D;
 strcpy(actionString, "SELECT deptnumb, deptname FROM org");
 pAction = (struct sqlchar *)malloc(sizeof(short) + sizeof(actionString)
   + 1);
 pAction->length = strlen(actionString);
 strcpy(pAction->data, actionString);
 strcpy(msgFileName, "tbexport.MSG");
 outputInfo.sizeOfStruct = SQLUEXPT_OUT_SIZE;

 printf("   client destination file name: %s\n", dataFileName);
 printf("   action                      : %s\n", actionString);
 printf("   client message file name    : %s\n", msgFileName);

 sqluexpr(dataFileName,
          NULL,
          NULL,
&dataDescriptor,
        pAction,
        SQL_DEL,
        NULL,
        msgFileName,
        SQLU_INITIAL,
        &outputInfo,
        NULL,
        &sqlca);
 free(pAction);
 EXEC SQL connect reset;
 return 0;
}
```

TOPIC 23 03 API

Point

API는 DB2 명령어에 대응하는 C 라이브러리 함수입니다. ESQL/C 소스에서 사용할 수 있으며, DB2가 제공하는 다양한 명령어에 대한 관리 API를 프로그램 로직과 함께 구현하는 방식입니다.

3 C 컴파일러를 이용하여 컴파일과 링크를 실행하십시오.

```
$ xlc -c <소스파일명>.c –I~<인스턴스명>/sqllib/include
$ xlc -o <실행파일명> <소스파일명>.o -ldb2 –L~<인스턴스명>/sqllib/lib
```

Tip

~<인스턴스명>은 DB2 인스턴스 개발자의 홈디렉토리를 의미합니다.

4 바이너리 형식의 실행 모듈을 실행하십시오.

```
$ <실행파일명>

$ cat ORG.DEL
10,"Head Office"
15,"New England"
20,"Mid Atlantic"
38,"South Atlantic"
42,"Great Lakes"
51,"Plains"
66,"Pacific"
84,"Mountain"
10,"Head Office"
15,"New England"
20,"Mid Atlantic"
38,"South Atlantic"
42,"Great Lakes"
51,"Plains"
66,"Pacific"
84,"Mountain"
```

TOPIC 23 04 JDBC

Point

JDBC 드라이버는 DB2 클라이언트 제품에 포함되어 있습니다. JDBC type2, type4 를 지원하며, ESQL 방식인 SQLJ 도 지원합니다.

Tip

JDBC 드라이버는 DB2 클라이언트 제품에 포함되어 있습니다.

Tip

SQLJ는 다른 ESQL 프로그램과 유사한 방법으로 프리컴파일과 바인드 과정을 거쳐 팩키지가 생성됩니다. SQLJ용 Run-Time 클래스는 DB2 클라이언트 제품에 포함됩니다.

1 SQLJ 또는 JDBC 방식으로 작성합니다. 데이터 액세스는 JDBC에서 제공하는 데이터베이스와 관련된 메소드를 이용합니다.

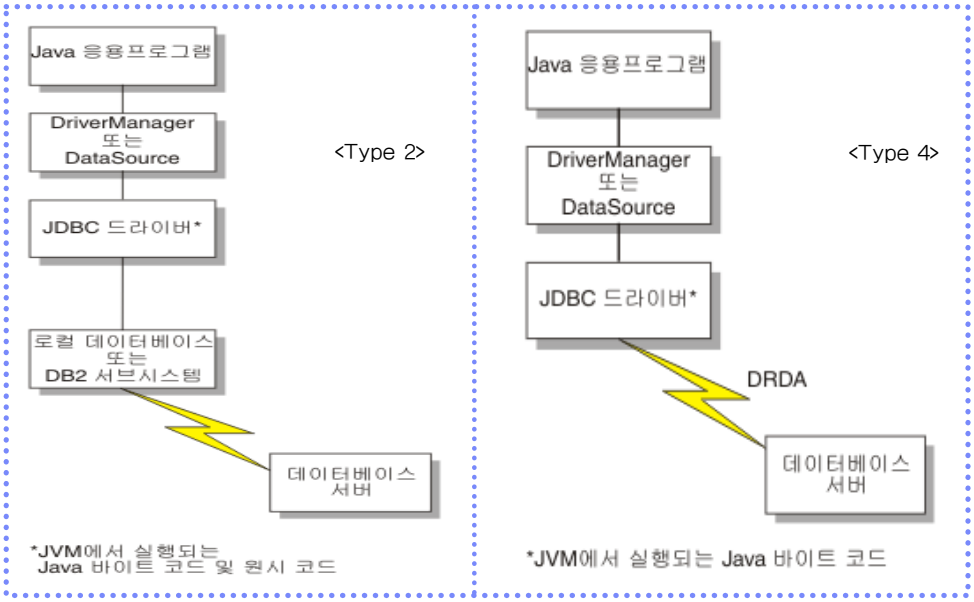


Figure 2304A JDBC 드라이버를 사용하는 자바 애플리케이션

2 DB2 개발자로 로그인하여 JAVA 소스 파일을 작성합니다. 확장자는 java 로 지정합니다.

```
$ login <DB2 개발자>
$ vi <소스파일명>.java
import java.sql.*;

class myjdbc {

  static {
    try {

    Class.forName("COM.ibm.db2.jdbc.app.DB2Driver").newInstance
    ();
    } catch (Exception e) {
       e.printStackTrace();
    }
  }

  public static void main(String argv[]) {
    Connection con = null;
    String url = "jdbc:db2:sample";
```

TOPIC 23 04 JDBC

Point

JDBC 드라이버는 DB2 클라이언트 제품에 포함되어 있습니다. JDBC type2, type4 를 지원하며, ESQL 방식인 SQLJ 도 지원합니다.

Tip

JDBC 방식의 프로그램에서 구현하는 SQL문은 동적 SQL문으로 처리됩니다. SQLJ 방식을 이용하면, 정적 SQL문을 구현할 수 있습니다.

```
try {
        con = DriverManager.getConnection(url);
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT * from employee");

        System.out.print("Received results:\n\n");
        while (rs.next()) {
          String a = rs.getString(1);
          String str = rs.getString(2);
          System.out.print(" empno= " + a + " firstname= " + str + "\n");
        }
        rs.close();
        stmt.close();
        con.close();
     } catch( Exception e ) {
        e.printStackTrace();
     }
  }
}
```

3 Java 버전을 확인하십시오.

```
$ java -version
java version "1.3.1“
Java(TM) 2 Runtime Environment, Standard Edition (build 1.3.1)
```

4 Java 버전에 맞는 컴파일러가 실행되고 있는지 확인하십시오.

```
$ which javac
/usr/java13_64/bin/javac
```

5 PATH, CLASSPATH 환경 변수가 적절히 구성되었는지 확인하십시오.

```
$ echo $PATH
$ echo $CLASSPATH
```

6 Javac 명령어로 컴파일합니다.

```
$ javac <소스파일명>.java
```

Point

JDBC 드라이버는 DB2 클라이언트 제품에 포함되어 있습니다. JDBC type2, type4 를 지원하며, ESQL 방식인 SQLJ 도 지원합니다.

7 java 명령어로 자바 클래스 파일을 실행합니다. 실행 결과는 다음과 같습니다.

```
$ java <클래스파일명>

Received results:

empno=000010 firstname=CHRISTINE
empno=000020 firstname=MICHAEL
empno=000030 firstname=SALLY
empno=000050 firstname=JOHN
empno=000060 firstname=IRVING
empno=000070 firstname=EVA
empno=000090 firstname=EILEEN
empno=000100 firstname=THEODORE
empno=000110 firstname=VINCENZO
empno=000120 firstname=SEAN
empno=000130 firstname=DOLORES
empno=000140 firstname=HEATHER
empno=000150 firstname=BRUCE
empno=000160 firstname=ELIZABETH
empno=000170 firstname=MASATOSHI
empno=000180 firstname=MARILYN
empno=000190 firstname=JAMES
empno=000200 firstname=DAVID
empno=000210 firstname=WILLIAM
empno=000220 firstname=JENNIFER
empno=000230 firstname=JAMES
empno=000240 firstname=SALVATORE
empno=000250 firstname=DANIEL
empno=000260 firstname=SYBIL
empno=000270 firstname=MARIA
empno=000280 firstname=ETHEL
empno=000290 firstname=JOHN
empno=000300 firstname=PHILIP
empno=000310 firstname=MAUDE
empno=000320 firstname=RAMLAL
empno=000330 firstname=WING
empno=000340 firstname=JASON
```

Memo